Real-time 2D Static Hand Gesture Recognition and 2D Hand Tracking for Human-Computer Interaction

Pavel Alexandrovich Popov

A thesis submitted in partial fulfillment of the requirements for the
Doctorate in Philosophy degree in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The topic of this thesis is Hand Gesture Recognition and Hand Tracking for user interface applications. 3 systems were produced, as well as datasets for recognition and tracking, along with UI applications to prove the concept of the technology. These represent significant contributions to resolving the hand recognition and tracking problems for 2d systems. The systems were designed to work in video only contexts, be computationally light, provide recognition and tracking of the user's hand, and operate without user driven fine tuning and calibration. Existing systems require user calibration, use depth sensors and do not work in video only contexts, or are computationally heavy requiring GPU to run in live situations.

A 2-step static hand gesture recognition system was created which can recognize 3 different gestures in real-time. A detection step detects hand gestures using machine learning models. A validation step rejects false positives. The gesture recognition system was combined with hand tracking. It recognizes and then tracks a user's hand in video in an unconstrained setting. The tracking uses 2 collaborative strategies. A contour tracking strategy guides a minimization based template tracking strategy and makes it real-time, robust, and recoverable, while the template tracking provides stable input for UI applications. Lastly, an improved static gesture recognition system addresses the drawbacks due to stratified colour sampling of the detection boxes in the detection step. It uses the entire presented colour range and clusters it into constituent colour modes which are then used for segmentation, which improves the overall gesture recognition rates.

One dataset was produced for static hand gesture recognition which allowed for the comparison of multiple different machine learning strategies, including deep learning. Another dataset was produced for hand tracking which provides a challenging series of user scenarios to test the gesture recognition and hand tracking system. Both datasets are significantly larger than other available datasets. The hand tracking algorithm was used to create a mouse cursor control application, a paint application for Android mobile devices, and a FPS video game controller. The latter in particular demonstrates how the collaborating hand tracking can fulfill the demanding nature of responsive aiming and movement controls.

Acknowledgements

First and foremost I would like to thank my parents Nevenka and Alexandre. You have given me love and support throughout this journey and I am very grateful. Mom you taught me the value of cooking from scratches, and Dad you taught me how to dream and take risks. I love you both very much.

Next I would like to thank my wife Zoé. You have been with me through the ups and the downs of this thesis, and your optimism gave me strength when I needed it most. I love you.

Ray and Helene, you have welcomed me into your family, and you run the best retreat this side of lake Ontario. Thank you for your support, and I love you both as well.

To all of my friends, including Andrew, Sheldon G. Eric, Lucien, Leila, Maria, Justeen, and Rory, thank you for the good times and the company.

To all of my friends from the uOttawa Quidditch Team, including Gabe, Allison, Sean, Zoe G., Derek, Jean-Sebastien, Véro, David, Kevin, Julia, Martin, Josh, and Karen, thank you for one or more great seasons, tournaments, and moments on and off the field.

To my supervisor Robert, who had an incredible amount of patience, faith, and vision, thank you for giving me a chance to pursue my research interests and for the guidance and support. It was a terrific journey, which had fun points, challenging moments, and was more than anything else a great learning experience. Thank you for helping me finish this project, and for showing me how to do research.

And a to Karin, who gave me a great learning opportunity during my undergrad which motivated me to continue to graduate studies. Thank you.

And lastly to anyone that decides to read this thesis, I hope it helps.

Sincerely,

Pavel

Contents

Table of Figures

Table of Tables

Chapter 1 Introduction

1.1 Overview

Hand Gesture Recognition is a domain of computer vision which focuses on recognizing and tracking human hands in video. The goal is to use the recognition of the gestures and the tracking information of the hand and its members to input commands to user interfaces and improve the state of human computer interaction. A hand gesture is either a specific hand pose, which means that it is a static gesture, or a specific hand motion which may include a specific pose or sequence of poses, which makes it a dynamic gesture. Hand gesture recognition based user interfaces offer an intuitive and powerful way to interact with computers [1], [2]. A basic setup for gesture recognition is a camera, or a camera with depth sensor, providing input to a computer which the computer processes to determine user input. Depth sensors are used often to easily segment the user's hand from a background. However their resolution can be limited and adding an extra sensor adds cost to a system. Furthermore it is possible to achieve good recognition and tracking results without a depth sensor as will be discussed in the following chapters. Control mechanisms can be made with novel gesture based information, allowing users to click, drag and drop with their hands instead of a mouse.

With the decreasing cost of high quality input web cameras and depth sensors gesture recognition could allow for cheaper to maintain at-a- distance user interfaces which have a variety of interesting applications. Museum displays could show a large amount of content, allowing for a high degree of interaction, without requiring a user to actually physically touch anything, which would reduce the wear and tear of their equipment. 3D graphic design would benefit from allowing users to manipulate objects using intuitive hand commands. The novelty of gesture interface as well as the increased immersion would enhance the experience of playing video games. The ability to communicate at a distance with a machine would be useful in combat situations and industrial areas to control robots to aid human users in certain tasks, or simply as an added safety

precaution. Hand gesture recognition can also have cross over applications with human motion capture if good hand tracking is achieved.

Humans communicate quite a bit with their hands and use their hands to interact with the world, and the question of how we communicate with our hands is an area of continued cognitive and psychological research interest [3]–[6]. By solving the challenges that still surround hand gesture recognition human computer interaction will be improved and new avenues of creativity will be opened. Gesture recognition need not replace current mouse and keyboard, touch screen, or game controller input methods, but it has a great potential to enhance them, and it would be a viable option for user input for future computer applications.

Gesture Recognition in the HCI context

Gesture recognition acquires a more clearly defined role when it is viewed in the context of human computer interaction. Specifically for the purposes of user interfaces there are position independent and position dependent commands. These commands need to be mapped to hand detections and hand movements in order to make use of gesture recognition. The commands provide control input to applications. Position independent commands can be performed with static hand gestures. Position dependent commands with hand tracking. Dynamic hand gestures, depending on how sophisticated they are, have the potential for performing both position independent and position depended commands. However, due to the complexity of executing dynamic gestures, they are not used in the systems presented in this thesis. Dynamic hand gestures are generally needed when there is an insufficient amount of hand poses that can be mapped to all the desired commands, which is not the case in the work presented here.

A gesture recognition system must recognize and respond to static hand gestures in real-time. Real-time systems are usually at least 30 frames per second, so in order to have real-time respond a presented gesture should be recognized after a couple of seconds. Narrowing this requirement down to one second results in a minimum recognition capacity of 1 frame out of 30 of a presented hand gesture being recognized, or

equivalently a 3.3% minimum true positive recognition rate. A low false positive rate is also desired to avoid false detections. Building these types of systems typically requires machine learning based model building which depends on having large enough dataset of static hand gestures.

Static gesture recognition allows for UI applications where position independent commands can be used to interface with the machine. These applications can be used with volumes, or on/off controls for example. Each control is mapped to a static gesture. The UI detects the user's gesture and responds according with a command. When position dependent commands are desired that situation is better handled by hand tracking.

In order to achieve hand tracking, a gesture recognition system must be able to first recognize a user's hand and then track it afterwards. Hand tracking uses prior information about hand position as well as current detection of the hand in order to provided and continuous and stable tracking estimate of the hand's location and structure can be used for controlling position based commands for UI applications. Mouse cursors and aiming can be mapped to such commands Hand tracking however does requires some way to initialize. This is where the relationship to static gesture recognition becomes apparent. Static gesture recognition can be used to initialize the hand tracking which allows for position based commands. While static gesture recognition can be used for position based commands the instantaneous nature of the detections would make a smooth response difficult to achieve. Hand tracking provides the necessary grounding to position based commands and the prior information can be used to filter out unrealistic, erratic and choppy successive commands and movements.

Testing the performance of these systems can be tricky. The position dependent nature of hand tracking takes the performance question of: *did the system track?* and adds the question of *how well did the system track?* Therefore hand tracking and finger counting datasets are needed in order to test these types of systems. Additionally the system has to be tried out as a user interface in example applications. Building user interface applications with hand tracking systems test if the system is actually useable as a UI by a user. Assessing how well a system tracks a hand when it successfully tracks the hand is very difficult to evaluate quantitatively, because there are many parts of a hand, and

quantitative evaluation of tracking quality would in the best case require a lot of laborious and time consuming annotation of hand part location in addition to annotation of hand locations. It thus becomes important to demonstrate the tracking system's usability as a UI. Compared to position independent static gesture recognition, hand tracking must be assessed qualitatively as well as quantitatively. User studies can be difficult and can be expensive because they require time as well as many volunteers, sometimes paid, in order to do properly. However even a small proof of concept goes a long way to demonstrating the viability of a hand tracking system as a user interface. To this end the work presented here demonstrates several proof of concept UI applications with the thesis author (me) demonstrating how a user can interact with a computer using these applications.

1.2 Problem Statement

This thesis work examines what is needed for strong and robust hand gesture recognition and hand tracking. While it is easy to make a hand gesture detector using existing object detection methods, these results are often quite poor because hands are not rigid. They are highly deformable and learning good features for hands is often a very difficult process which results in frequent false positives. Solutions must be tailored to the shape and characteristics of the hand. Hand tracking is a complement to hand gesture recognition and becomes a must for user interfaces. While a single static pose can be detected, singular instance detection is a limited method for making user interfaces. What is of interest is continuous instance detection of hands and their static hand poses. This means hand tracking, which opens up avenues for increased interactivity for user interfaces. Furthermore not only the general location of a detected or tracked hand is of interest, but also the location of its members such as the palm and the fingers. If for example fingertips are tracked then their tracked locations must be stable, and only move with when the user's fingers move in real life. This is extremely important to ensure that user interfaces can receive responsive and reliable commands. Stability of tracking is essential for any member that is used for command input in a user interface.

It is very difficult to achieve contributions in hand recognition and in hand tracking in one system. Certain works achieve good multi scale recognition results, while others focus more on hand tracking. It appears that it is very uncommon to have contributions in

both areas. New works in the field typically present new ways to detect hands or recognize hand gestures [7], [8], [17]–[26], [9], [27]–[31], [10]–[16], or use a simple established detection or recognition method and use that to built a hand tracker[32], [33], [42]–[51], [34]–[41]. One of the systems that is presented in this thesis will endeavor to present a complete hand gesture recognition and tracking system with novel contributions in both areas.

Static hand gesture recognition methods have drawbacks which arise from small datasets and false positive detections from the machine learning models used. Methods of different types require different size datasets which can make comparisons difficult if the dataset is too small. Additionally real-time performance is always desired but sometimes the models become to computationally demanding. This work aims to solve these problems by using a large dataset to allow to train and compare a variety of machine learning models. The work in static gesture recognition uses simple machine learning models are used to detect in real-time and a geometric hand model is proposed to validate true positives and eliminate false positives. Criteria are defined for a false positive rate low enough and true positive rate high enough to have a responsive system in real-time. These are accomplished by the gesture recognition system presented in this thesis.

Hand tracking in 2 dimensional video only scenarios has not been satisfactorily solved[52]. Hand tracking in 2 dimensions has advantages over tracking in 3 dimensions because algorithms are computationally lighter, and more versatile. Hand tracking methods in 2D video rely on a range of limitations to constrain the problem. These include uniform background, initialization in a fixed area, fixed colour ranges for segmentation, and face presence. Furthermore as already mentioned steadiness is needed for user interface applications. Some methods are also not automatic, sometimes requiring manual user tweaking. In addition there is no consensus on how to estimate palm or fingertip location. Tracking methods can fail in certain situations so recoverability is an issue as well.

One of the systems presented is designed for hand tracking. The system was designed to impose as few restrictions on the user as possible. The operation restrictions were reduced to 3 user limitations. The system is designed for an indoor environment with

stable light. Occlusions with skin coloured objects have to be avoided. A hand gesture of a specific type has to be made in order to commence hand tracking. The hand tracking system removes the need for uniform backgrounds by actively sampling the user's skin colour. The system achieves automatic registration of the user's hand for tracking using a defined hand gesture verified by a hand model. Users are not required to do any further tweaking. This automatic registration is further expanded by using static hand gesture recognition to find the initial hand location. This allows the hand tracking to commence anywhere in the frame and removes the limitation of a fixed hand registration area. A novel hand tracking strategy is proposed that finds fingertips and palm locations and provides stable fingertip locations for user interface applications. Recoverability of the tracking is achieved both with 2-strategy hand tracking, as well as with static gesture recognition which can re initialize the tracking if it loses track of the user's hand.

The performance of the systems are tested with static hand gesture recognition and hand tracking datasets. Several UI applications are demonstrated with the hand tracking, including a mouse cursor controller, an Android Paint application, and a First Person Shooter video game controller.

The systems presented in this thesis address the major challenges in using 2d hand recognition and tracking for user interfaces. The work furthers the research in this area making significant contributions in static hand gesture recognition, hand tracking, and datasets for these research topics. It shows the viability of 2D video-only hand recognition and tracking for UI applications with important proofs of concept.

1.3 Thesis Statement

The goals of this thesis project are all in hand gesture recognition. The goals can be summarized as follows:

- The first goal is real-time multi scale hand gesture recognition from a video or webcam input
- The second is robust, articulated and steady hand tracking. Tracking will provide not only the general location of the hand but also the location of the fingers and the palm.

- The third goal is a very low false positive rate for gesture recognition.
- The fourth goal is to combine both hand gesture recognition and hand tracking into a robust and highly usable system.

All four goals are important for making hand gesture recognition systems capable to be used as control input systems for user interfaces.

1.4 Research Method

This section explains what the expectations were for the research problems of hand gesture recognition and hand tracking in a 2D video only context. It also explains how the evaluation would be done, and a general overview of the research approaches taken.

1.4.1 Hypothesis

The hypothesis of this thesis work contains two parts. One part is directed towards hand gesture recognition and one part is directed towards hand tracking.

Recognition of hand gestures is possible in a real-time video only context with low cost solutions. Real-time performance is possible which allows a user to have a responsive user interface. Gestures come from a live video stream and a solution should be possible with a camera recording user input and feeding it continuously. A solution should work in a complex background reducing user limitations. It is also assumed that the user is present in the video input in a front facing manner, including face, gesturing hand and arm, and upper body. A solution should be able to work with only a few gestures, and if successful, it is likely able to be generalized to more gestures. The key element is to work well for at least a few gestures.

Tracking of a user's hand is possible in a real-time video only context with low cost solutions. A solution should be possible using 2D only methods, which are computationally less demanding than any method that uses 3D data. Tracking should be continuous from frame to frame, and provide the location of the user's palm and fingertips. The locations of the fingertips should be stable in order to facilitate UI applications. A solution should work in a complex background reducing user limitations. A tracking solution should initialize tracking automatically once a user's hand is

recognized with no additional demands on the user other than presenting their hand for recognition. Tracking should continue until tracking is lost and the solution should be able to reset to track the next presented target.

1.4.2 Evaluation Criteria

Hand Gesture Recognition

The performance of any gesture recognition solution will be evaluated using true positive rates and false positive rates for each gesture. The true positive rate for a gesture refers to the frequency that a given system recognizes a gesture when it is presented. The false positive rate for a gesture refers to the frequency that objects other than the gesture, including backgrounds and other gestures, get falsely classified as the gesture in question.

For the goal of making a real-time gesture recognition system, false negatives can be accepted. False negatives would reduce the true positive rate of any system, however it is not significant if the system is fast and able to run in real-time. The speed of a lightweight system would make up for a low true positive rate. The goal is to minimize false positive rate as much as possible because issuing false commands to the system is undesirable and would deteriorate its usability. While low true positive rates can be accepted, in order to achieve low false positive rates and a lightweight system, there is a limit. In a real-time system 30 frames per second is an expected speed of execution. If a gesture is presented it is sufficient in a real-time system to only have 1 out of 30 frames successfully recognized as the presented gesture. This will have a response time of 1 gesture recognized per second. This corresponds to a 3.3% true positive rate, and as such a 3.3% true positive rate is the lower bound on true positive performance for any viable gesture recognition solution.

Any gesture recognition system will also be evaluated on its ability to run in a live online user scenario. It is one thing to have a system that evaluates well on collections of static images, it is a different scenario to have it run with a live video camera feed with a user

in a scene. This will be a demonstration of its usability and the demonstration will be recorded for presentation.

Hand Tracking

The performance of any hand tracking solution will be evaluated on its ability to track the user's hand and its usability as a user interface. Tracking of user's hand will be evaluated in an unconstrained 2D video only continuous context. A video dataset will be used to evaluate tracking performance by seeing if a given solution tracks the user's hand in the appropriate area of the frame, and to evaluate the given solution's finger counting performance.

Fingertip location stability is important for any tracking solution because fingertips are natural entities to which to map cursor type controls in user interfaces. Fingertip stability would be difficult to evaluate with a dataset because it would require too much laborious annotation. Furthermore users react to the applications that they are controlling, so testing for fingertip stability with videos where users don't see what are the consequences of their reactions would be unrealistic. That is why fingertip stability, which is important for UI, will be demonstrated with UI applications and demonstration videos.

Comparison

The assumptions of complex background, user presence, and the need for low cost real-time solutions make comparison to other systems difficult. This is further complicated by the lack of large quality annotated datasets that are designed for these problems. As such it is difficult to compare to other systems. However the gesture recognition system will be compared to CNN models for similar real-time applications in order to see the strengths and weakness of the system to models that are seen as universal solutions to general problems in this research area.

1.4.3 Method

Gesture Recognition Method

The approach for gesture recognition is to use machine learning algorithms with a dataset of gestures in order to train models which can recognize gestures in incoming frames from video. To train the machine learning models a dataset of hand gesture images was collected. The dataset is annotated according to gesture class.

The gesture recognition system has 2 steps. A detection step detects hand gestures in video frames using machine learning models. A validation step processes the detections to confirm true positives and reject false positives using skin colour and hand shape.

The system is evaluated using the dataset. A CNN model for mobile real-time applications is trained with the same dataset. The gesture recognition system is compared to the CNN model in order to see how the 2 step machine learning and hand gesture validation approach compares to CNN models for mobile vision applications. Comparison is made in terms of recognition performance and computational load.

The system is also tested in a live user scenario context with a camera. The demonstration of the qualitative performance is recorded and presented. This shows that the system works in live scenarios with continuous video feeds, not just with datasets of images.

Tracking Method

A hand tracking method is used which uses 2 collaborative tracking strategies. A potentially noisy and lightweight strategy based on skin colour and hand shape tracks the hand. A finer potentially heavier strategy using template matching tracks the fingertips with higher frame to frame stability which is needed for UI. The lightweight strategy guides the heavier template matching strategy limiting its tracking area which increases its speed. The tracking method tracks the user's hand from frame to frame after an initial target registration.

The tracking algorithm was developed using live experimentation of tracking strategies using a camera and a user's hand. It uses hand crafted features for tracking that can later be optimized if a dataset specific to problem is built. This is a laborious endeavor beyond the scope of the thesis but worth mentioning here as part of the process of obtaining an even better more optimal solution in future work.

To evaluate tracking performance a video dataset was collected containing users moving their hands and issuing sequences of extended visible fingers. The tracking method is evaluated using this dataset.

UI applications were built with the tracking algorithm. Demonstrations of these applications were recorded and presented. These applications show the tracking algorithm's usability as a UI.

1.5 Contributions

1.5.1 Proposed systems

This thesis presents 3 systems that push the envelope on what is possible with real-time hand gesture recognition and tracking systems using low cost hardware. These proposed systems are designed to work in real-time with low cost web cameras. They do not require IR or depth sensors or other more expensive hardware.

1.5.2 Hand Gesture Recognition system

The Gesture Recognition Algorithm recognizes hand gestures in real-time from a video, anywhere in the video frame. The algorithm was designed to achieve two goals of the thesis, real-time multi scale hand gesture recognition, and a very low false positive rate. There are two main steps to the system. The first is a detection step which achieves real-time recognition with simple machine learning models, specifically Adaboost and SVM. The second is a validation step which dramatically reduces false positives and validates true positives with geometric hand contour models. Figure 1 shows some successfully recognized gestures.

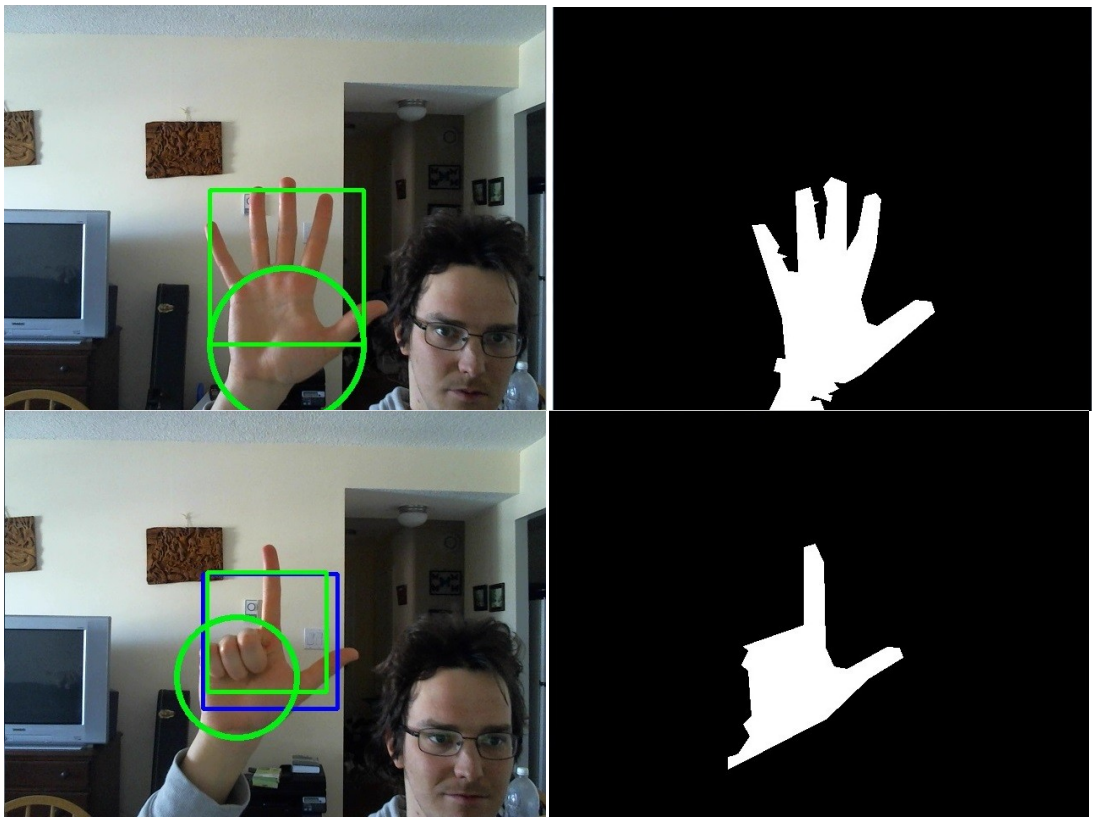


Figure 1 Example output of the 2-stage Hand Gesture Recognition system.

The major contributions of this system are the hand contour models, the improved colour filtering for segmenting hand gestures, the idea of breaking down the recognition task into two steps, and the comparison of the system to the MobileNets CNN model [53].

1.5.3 Hand Gesture Recognition and Hand Tracking system

The Hand Gesture Recognition and Hand Tracking system is a complete recognition and tracking system for human hands. This system will be able to recognize and track a user's hand anywhere in a video frame in real-time using a web camera. This system has two major components, which are hand gesture recognition and hand tracking. The algorithm uses the Gesture Recognition algorithm to find a user's hand in a frame from a video input feed, and then the colour and shape information of the recognized hand gesture is used to initialize the hand tracking portion of the algorithm. The hand tracking uses 2 collaborative strategies to achieve real-time and steady tracking of the user's hand. Each of the respective strategies, contour based tracking of the hand, and template based tracking of the fingertips, have strengths and shortcomings. Combining the two strategies

overcomes these shortcomings and achieves a good tracking result which is a viable option for user interfaces.

Figure 2 shows some examples of the hand tracking.

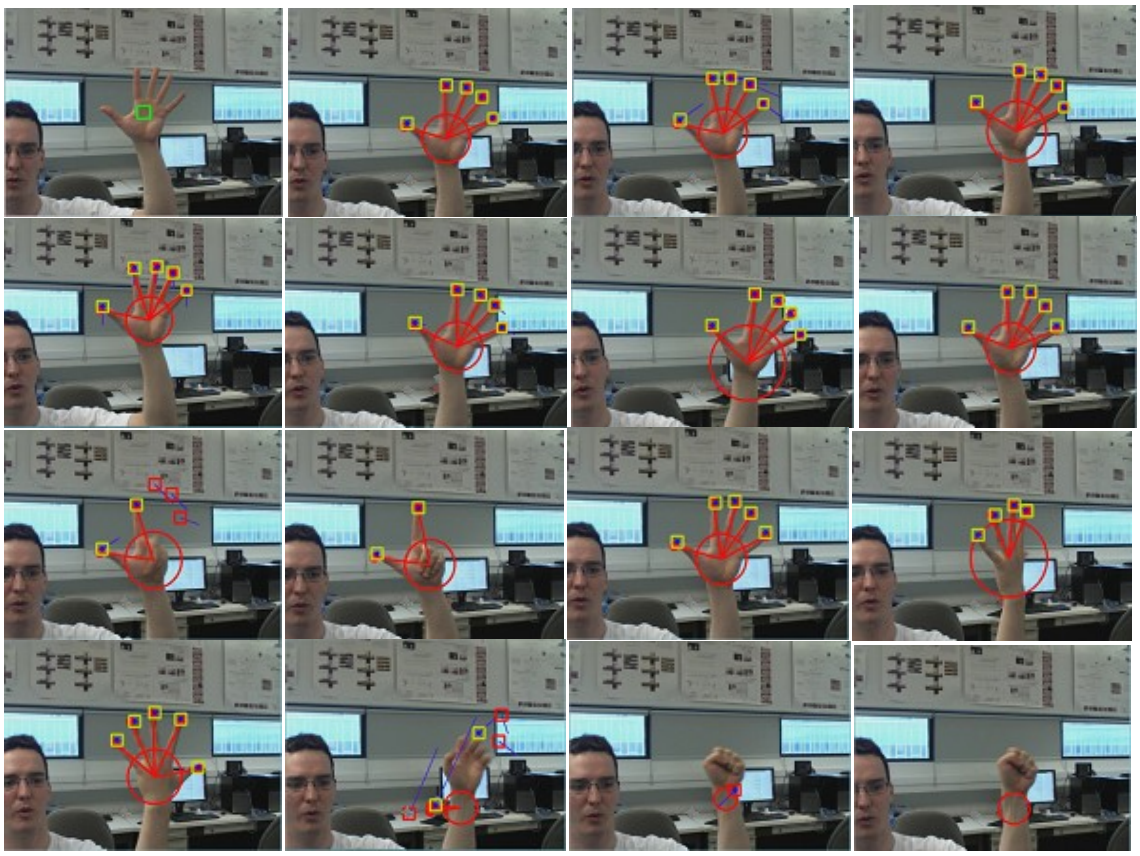


Figure 2 Examples of the Hand Tracking Algorithm

Major contributions of the system include combining hand gesture recognition and hand tracking into one robust real-time end-to-end system, and the collaborative multi strategy hand tracking algorithm. The hand tracking algorithm is also capable of recognizing the static gesture pose of the user's tracked hand from one of 5 distinct poses using a series of HOG Cascades. A variety of user interface applications are also demonstrated to prove the concept of the technology. These include mouse and keyboard controls, game controllers, and mobile Android applications.

1.5.4 Hand Gesture Recognition with Colour Clustering Segmentation system

The Gesture Recognition with Colour Clustering system is designed to improve upon the regular Gesture Recognition system by using all of the colour information available in the

hand gesture detection boxes. The hand gesture detection boxes produced by the detection step are not always well localized and the sampling process that is performed by the validation functions sometimes samples locations containing hand and background colours which results in bad segmentations and missed validations.

Figure 3 shows the effect of a misaligned sampling zone.

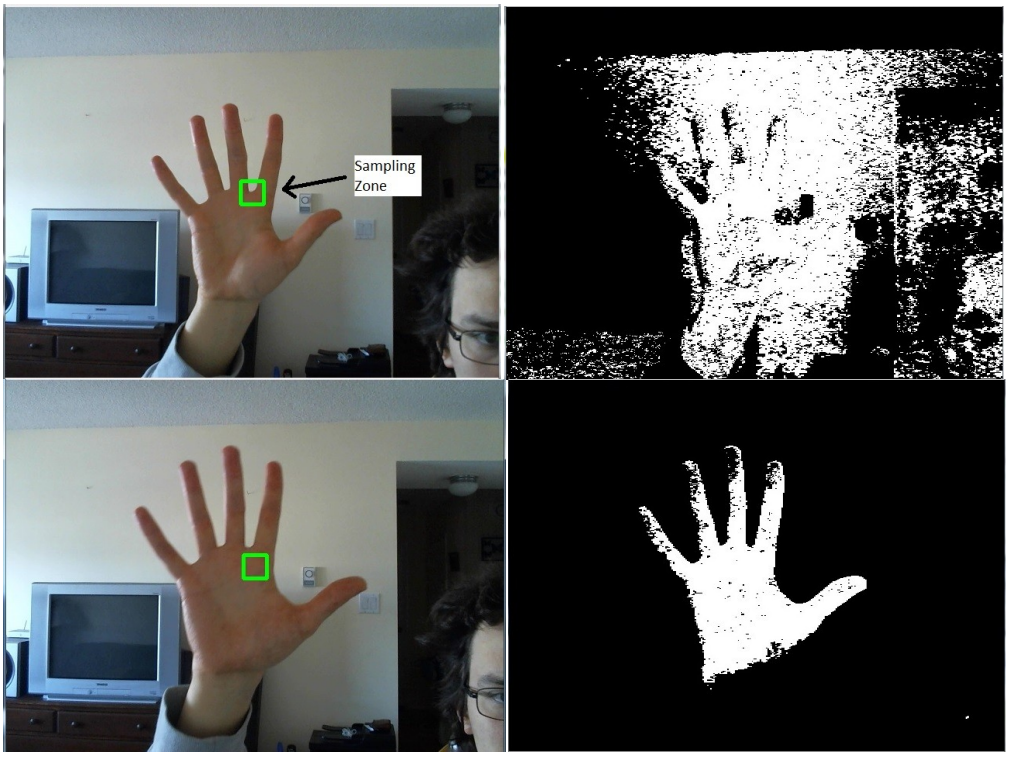


Figure 3 Colour segmentation using poorly placed and well placed sampling zones.

The Colour Clustering segmentation is the major contribution in this system. It's a novel method that uses all of the colour information to generate a complete 2D Colour Histogram of the area within a given detection bounding box which is then used to more thoroughly segment the scene and extract better hand contours to use for validation. It increases the overall recognition rate of the system because it uses all of the colour information provided by the detection boxes in the cases where it is applied.

1.5.5 Datasets

In addition to the 3 proposed systems, this thesis research has made 2 datasets, one for hand gesture recognition, and one for hand tracking.

The hand gesture recognition dataset contains between 8000 and 10 000 images for each of 5 different hand gestures, and approximately 2000 samples for 5 additional hand gestures. This dataset is a major thesis contribution enabling gesture recognition research for a variety of applications. The dataset is large it allows for comparisons between a variety of machine learning methods, from ones that use small datasets to ones that benefit from large amounts of samples. The dataset is available online at [54].

The hand tracking dataset contains 19 videos which have been annotated frame by frame indicating hand location and how many fingers are visibly displayed by a user in each frame. It would be useful for anyone interested in testing the performance of their hand tracking method. The hand tracking dataset is available online at [55].

1.6 Thesis Structure

The rest of this thesis is structured as follows: Chapter 2 examines techniques used to solve problems in Hand Gesture Recognition and Hand Tracking, the common ground between different applications and goals, and how they apply to the systems described in this thesis. Chapter 3 presents the machine learning models that were used in the Hand Gesture Recognition system. Chapter 4 shows the machine learning models of Chapter 3 were combined with hand contour models into a Hand Gesture Recognition system. The system presented in chapters 3 and 4 represents the first major feature of this thesis work: Static Hand Gesture Recognition. Chapter 5 presents the Gesture Recognition and Hand Tracking system which presents a robust hand tracking strategy that works with the presented recognition method in Chapter 4. This chapter forms the second major feature of this thesis work: real-time hand tracking. Chapter 5 presents the Hand Gesture Recognition with Colour Clustering segmentation system which presents the Colour Clustering segmentation technique designed to alleviate a shortcoming of the Gesture Recognition system. Chapter 6 concludes the thesis, and describes some future potential research directions.

Chapter 2 Literature Review

2.1 Overview

Hand Gesture Recognition is a diverse and varying field that is still being defined. It encompasses both static gestures and dynamic gestures, and there is significant interest in accurate and robust hand tracking. A variety of applications have also been proposed. There is also significant overlap between hand gesture recognition and human action recognition. Both use similar methods and both are linked to hands, the former explicitly, and the latter implicitly at the very least because many human actions of interest involve hands. Whatever the approach and whatever the goal of the hand gesture recognition or hand tracking approach, some common elements emerge. Features are needed to describe, detect, track, or capture the user's hand. Classification of these features is needed, and in general machine learning is preferred in order to find mathematically optimal solutions using datasets. And lastly, real-time performance is desired. This becomes a must when the proposed hand gesture recognition or hand tracking approach is presented as a method for user interfaces. It is assumed unless stated otherwise that all of the methods presented in this literature review work on images and video either in real-time or offline.

2.2 Features

A number of papers propose new methods dependant on new features or using existing features in a new way. Well known approaches that are reused by new proposed methods often make explicit reference to the type of features that were used for the hand gesture recognition or hand tracking. This indicates that there is still a significant amount of research interest in finding which features give the best performance for which gesture recognition applications and it remains an open problem.

2.2.1 Colour and Depth

Colour is the most basic feature of images and video and it remains a very prevalent feature that is used in hand gesture recognition and tracking methods. [7], [10], [37], [41], [44], [51], [56]–[59], [12], [13], [18], [24], [31]–[33], [35] all use colour in their proposed

methods. Colour features are often used to segment hands or to track them via segmentation. Skin detectors and colour histograms are interchangeable terms that define ranges of colours that are used for the segmentation, they can be pre defined or extracted adaptively. [7], [10], [51], [56]–[58], [12], [18], [24], [31], [33], [35], [37], [44] use skin detectors, which select a specific pre defined range of colours to keep. Whereas [32], [33], [41] use adaptive segmentation approaches that actively sample the colour of the user's hands either automatically or with user defined colour selection. [13] uses an adaptive colour segmentation approach that partitions the input image using k-means clustering. This approach also responds adaptively to the colour of the user's hand by grouping like colours together. Difference of colour is also used as a feature for segmentation. [33], [59] use simple background subtraction to segment user's hands. It is interesting to see that [33] uses a skin detector, which ranges can be adjust by the user to achieve a better result, and it also uses simple background subtraction. This means that these three techniques are not mutually exclusive.

Depth is another feature that is often used in hand gesture recognition and tracking, particularly for segmentation. It has become increasing used with the availability of cheap depth sensors such as the Microsoft Kinect and the Intel Creative Interactive Gesture Camera. [33], [50], [60], [61] use depth thresholding to segment the nearest object in their methods. A fusion of depth data and RGB data is proposed in [62] to form a shared feature space of vectors for classification with SVMs. [63] uses depth data to train their CNN model which operates on 2D images. Depth data is also used for 3D hand gesture recognition and tracking methods, and these will be mentioned later.

2.2.2 Geometric Hand Shape Features

Building upon simple colour and depth features, geometric shapes and properties of hands and their geometric shape features offer a way to analyze the shapes of hand gestures and extract meaningful information about pose, and the location and visibility of their members such as the palm and fingers. [9], [12], [41], [50], [51], [57], [59]–[61], [64]–[66], [13], [67], [68], [27], [28], [33], [34], [37], [39], [40] all use some sort of geometric shape features, these methods mostly deal with hands, with one exception [65] that deals with the edge features of objects. This method is still of interest because edge

features can be just as applicable to hand gesture recognition and tracking as was shown in [12], [64]. The authors of [12], [13], [65], [33], [34], [39], [40], [50], [57], [59], [64] all use some sort of edge or contour based analysis. [13], [33], [34], [39], [40], [50], [57], [59] specifically use different types of convexity defect, contour curvature analysis and other geometric tests to find the location of the palms and the extended visible fingers of the hands. These methods operate directly on the contour boundaries and produce stable results in the absence of significant noise. They apply a variety of logic and shape tests directly on the contour boundaries and the convexity defects to determine fingertip and palm locations [33], [50], [57], [59], finger count[13], [33], [59], gesture pose [13], [33], [34], [39], [40], [50], [59], and in some cases even the label of the extended fingers [13][50].

Another way to analyze the shape of the contour is to find its shape signature which was done by the authors of [41], [60], [61], [67], [68]. The works in [41], [60], [61], [67] take contours and find their 1 dimensional shape signatures and then use them for further processing and classification. [68] proposes a slightly different shape signature related method for counting valid fingers. The authors use a binary skeleton transformation of hand contours and a trained mathematical model to analyze the path of each potential finger skeleton to determine if it's a valid finger. It shares similarities to shape signature methods because it transforms each skeletal path with a normalized radial function, and normalization and radial properties are used in shape signatures.

2.2.3 Vectorizable Mathematical Transform

A third category of features used in hand gesture recognition and related fields are vectorizable features generated by mathematical transformations. In order to implement many different powerful machine learning classification methods vector features of input data are needed. Different feature extraction methods have been proposed and are used for producing vector features. The mathematical transformations that produce these vector features operate on raw images, and segmented contours. They also typically encode some shape information that is hopefully invariant to changes in scale and orientation. Thus there is some overlap with geometric shape features.

Some well known vector features are that work on raw images are HOG features [69], Haar-like features [70] , SURF [71], FREAK [72], and SIFT [73]. HOG features are extensively used and can be found in [8], [17]–[19], [22], [23], [31], and have been of interest in object classification and detection since they were proposed by Dalal and Triggs[69] for pedestrian detection with dense sampling. Research continues in the area of HOG feature with [17] proposing a skin colour modification of HOG called SCHOG for hand detection. Viola and Jones [70] proposed using Haar-like features for their face detection system, and Haar features have continued to be used in more recent work in [8], [34], [39], [40], [56], [70], [74], [75]. The authors of [56], [74], [75] have even gone as far as to propose improvements to Haar features for their applications. FREAK features are used in [47] for hand detection. The authors of [21] and [25] use SURF features, and [25] also uses SIFT, indicating that although these features are not as popular as Haar or HOG they are still of interest.

There are also several methods that use various descriptors to vectorize shape information from segmented hand contours. Hu moments are used in [9], [37], Contour Sequences are used in [9], [27], [28], [66]. The work in [51] uses Fourier Descriptors, and [67] uses wavelet transform. These methods serve to take the contour shape of indetermined length and vectorize it by encoding the shape information into a vector of fixed length. The works presented in [60], [61] both use near convex decomposition to threshold finger shapes that appear in the contour shape signature and these shapes are then catalogued by size and location to make fixed length shape signature vectors.

Certain authors use other vector features for their hand gesture recognition methods. [7] uses vectorized hand shape and texture features, [12] and [27] both use block based features around interest points which also serve to describe hand textures. It is interesting to note that [12] uses clustering to extract and categorize relevant brightness and texture feature information around edges detected in hand images.

Other vector features that have been proposed are by: Kong and Fu [62] who propose fusing RGB and Depth data into a common feature space for vectors, Rumyantsev et al whose work presents eigenvectors of scaled gesture contour shapes in [32], Dollar et al [76] who propose Weakly Invariant Pose Indexed Features, and Uwineza et al [77] which

use a fusion of Harlick Texture, Hu moments, and Colour histograms (1 dimensional). The work of [77] shows how colour can be used as a vector feature.

2.2.4 Other Features

Some other types of features have also been proposed in various works for hand gesture recognition and related applications. These can be mostly categorized into motion and statistics features. Motion features are used in [35], [37], [41], [43], [44], and in the case of [35], [37], [41], [44] the motion features are combined with skin or skin and depth segmentation. [43] on the other hand takes the motion features and the extracted motion path and then calculates its Log Path Signature to reduce complexity for classification. The works in [7], [47], [78] use statistical features based on saliency and entropy to find the most visually interesting areas of images. [16] uses Hough features in an ASL recognition system.

2.2.5 Relevance to proposed systems

In the Gesture Recognition and Hand Tracking systems presented in this thesis some of the previously mentioned features are used. Extensive use of colour information is made using 2D colour histograms. These are generated by adaptively sampling the user's skin colour to achieve very good segmentation results. The methods that will be presented build upon research done on colour histogram distributions which is highly applicable to the problem. These papers [79]–[85] were consulted when developing the colour filtering techniques used in all 3 proposed systems, it was of particular interest to learn how to determine a unimodal distribution from a non-unimodal one.

Geometric shape features of contours and their shape signatures are used to validate and invalidate gesture detections done by the machine learning trained multi scale detection components of the systems. The geometric shape features of contours along with the 2D colour histograms that segmented them are also used for hand tracking. Depth data is not used because none of the systems were designed to be used with a depth sensor. However

depth sensor based gesture recognition techniques are still of interest because once the initial hand segmentation is performed they use many shape analysis and gesture classification techniques which are highly applicable to video only gesture recognition techniques.

HOG features are also used by the hand gesture detection and pose classification methods that are used in the systems.

Feature extraction is important in order to find and segment hand candidates from background. Some works achieve good results but these are only designed for uniform backgrounds [8], [13], [39], [40], [86], [16], [18], [23], [24], [26], [28], [32], [34], [63], [87], [88]. Other works design their methods to work in cluttered and complex backgrounds [7], [9], [21], [22], [25], [27], [29]–[31], [33], [35], [36], [10], [37], [38], [41]–[45], [47]–[49], [11], [51], [56], [58], [60], [61], [66], [89]–[91], [12], [14], [15], [17], [19], [20], [92], [77], which dramatically increases the usability of their systems for user interfaces. What is interesting to note is that with good quality hand segmentation a method only designed for uniform backgrounds can be used for further gesture recognition and tracking. The 3 systems presented in this thesis were designed to perform hand gesture recognition and hand tracking in complex backgrounds. The good performance of the systems in cluttered environments is the result of robust and adaptive colour processing techniques which are used to create high quality segmentations. Cluttered and complex environments which can have variable light conditions also motivated the selection of HOG features for classification.

2.3 Feature Optimization and Kernelization Articles

Although they can provide very useful information for description and classification, feature extraction can sometimes be computationally expensive and thus prohibitive to real-time and mobile applications. Some features underperform in certain situations due to limitations that have not yet been overcome. There is continued interest in computer vision research to reduce the computational cost of features extraction, to optimize their evaluation, and to improve their performance.

Some examples can be found in these publications [14], [21], [25], [75], [89], [93]–[97] with these [14], [21], [25], [75], [89] being specifically explained in the context of hand detection, hand tracking, and hand gesture recognition, and the rest are highly applicable.

These works [14], [93], [96], [97] all use some kind of kernelization. [14], [93], [97] use kernelization to represent features in a different vector spaces where they are easier to separate by linear boundaries. Nemirko [93] uses Fisher Linear Discriminant to separate generic features. Nguyen et al. [14] proposes several interesting variants of a Kernel Descriptor (KDES), first proposed in [98], for static hand gesture recognition. Nguyen et al uses hand gesture classes with high interclass similarity and these benefit from the improved class separability achieved by the proposed KDES variants. [97] proposes the CovSVDK method for easier class separability of human actions which are represented by video clips of variable length. This method overcomes the variable length problem with covariance matrix calculation, and succeeds in making the action classes linearly separable. Henriques et al. [96] use Fourier descriptors as part of their tracking, and they use kernelization to exploit the circular structure of their video data. They achieve fast performance as a result.

Bag of Features is used by [21], [25], [94] and [95] in order to optimize features used for training and representation. Key features are found by clustering features. This helps train classifiers to only use the most relevant features and it solves problems cause by having too many samples and an imbalance of samples. It also makes classification simpler because the key features (or words) are used to approximate the input features in a simpler way. Thus differences are only encoded in feature vector members if they are significant.

Other computational improvements include exploiting redundancy in feature calculation. This was done in [89] for HOG features using their Distributed Object Detection framework. [95] improves the performance of Fisher Discrimination and VLAD which is used for object detection. It represents images as sparse integral images with code words found with Bag of Features. This results for a dramatic speed up. Finally [75] improves the classification power of Haar Feature Adaboost Cascades by exploiting feature

redundancy. By using rotated integral images, the method is able to add rotated Haar-like features in a computationally efficient way.

2.3.1 Relevance to proposed systems

Feature optimization and Kernelization techniques are not used by the systems presented here. The first reason for this is the lack of datasets. The hand contour models presented in this thesis could be optimized with a large contour dataset but the process of obtaining this data would be laborious and time consuming. It can be saved for future work on the topic. The second reason is that there are many research problems in this area that are still open, and solving these problems would take away from the focus of creating hand gesture recognition and tracking systems. It would become less a gesture recognition problem and more a mathematical optimization research question. This reason also makes these methods more difficult to use.

Even though feature optimization and kernelization techniques are not used, the need for using computationally fast features is taken into account because the systems have to run in real-time because the interest of this thesis is in real-time gesture recognition and hand tracking systems. Real-time gesture recognition systems offer exciting possibilities for creating Gesture Recognition based user interfaces. Real-time performance is only possible if computational efficiency is taken into account.

2.4 Gesture Detection and Classification

Classification methods are a common ground for Hand Gesture Recognition, be it static or dynamic gestures, and other related computer vision fields like Pedestrian Detection, Action Recognition, Object Detection, and Object Tracking. What works in one domain can be adapted and scaled to work in another, and the concerns and challenges are sometimes one and the same for different problems. Different methods are used for hand detection, hand gesture recognition, or both. Classification methods as they relate to Hand Gesture Recognition can be grouped as Adaboost Cascades and related methods, Support Vector Machines, Neural Networks, Other learning based methods, and Simple Logic methods. The first four categories all use machine learning and datasets to obtain solutions. The last category uses hand crafted logic and trial and error done by human

designers. Although this last category of methods cannot compete with the results provided by machine learning methods these methods do yield interesting results that are usually quite lightweight. These methods serve as a refreshing as well as introductory perspective to the hand gesture recognition problem and sometimes yield beneficial results which can be further improved with machine learning to bring competitive advantages to the state of the art.

2.4.1 Adaboost Cascades

The Adaboost Cascade of Weak Classifiers was proposed by Viola and Jones [70] for face detection in 2001 and since then it and Adaboost inspired methods have continued to be used for detection and recognition tasks [8], [14], [74]–[76], [99], [15], [19], [22], [29], [34], [39], [40], [46]. [8] and [99] use adaboost for Hand and Object detection, and the classification task is handled by PCA and SVMs, respectively. Rautaray and Agrawal use Haar feature cascades as a hand detection step in their work [34], [39], [40]. Nguyen et al [14] also use a Haar Cascade as part of their larger gesture recognition system. Haar features and Adaboost are used in [74] for object detection. Complete gesture recognition is done using Adaboost in [15] and [19]. [19] even extends the classification approach to give Adaboost a multiclass structure. Messom and Barczak also proposed to improve the Adaboost algorithm in [75] with diagonal Haar features to make the algorithm more robust to rotation. Also a number of other classifiers have also been used to make multi-stage architecture very similar to Adaboost Cascades, with each stage refining the detections of the previous stage to improve the overall detection or classification result. Cascades of Support Vector Machines have been used in [22], [29]. Cascades of weak regressors have been used in to make strong regressors in [46], [76]. The regressor works both use training algorithms similar to Adaboost with similar error minimization per stage.

2.4.2 Support Vector Machines

Support Vector Machines were first proposed by Cortes and Vapnik [100] building on earlier work in [101]. [102] provides a practical guide for parameter selection and training strategies. SVMs have become a popular classification method in Hand Gesture Recognition and related fields. They have been used for action recognition [62],

[94],[103] hand detection [17], [22], [29], [89],[7], hand gesture recognition [7], [14], [56], [91], [104], [18], [21], [23], [25]–[27], [37], [51], [105], and pedestrian detection [69]. The works presented here are not an exhaustive list and other applications as well as more examples of the mentioned applications can be found in the literature. SVMs are simple to train and with linearly separable data, or a good choice of a non-linear kernel, they perform well. Only one drawback has been identified. The drawback is that the more samples are used to train an SVM classifier the larger the model tends to become. This is because, as explained in [100], the training is done by finding the optimal margin of separation between the difficult to classify samples, which are referred to as Support Vectors. With larger datasets there tends to be more and more samples selected as support vectors. This means that while larger datasets may lead to a more optimal problem solution, the increasing amount of support vectors becomes more and more computationally costly to evaluate. Therefore it is difficult to train a SVM with a large dataset and to keep the model small enough to run in real-time. What is interesting to note with Support Vector Machines is their adaptability. While many works use SVMs to recognize static hand gestures [7], [14], [104], [18], [21], [23], [25]–[27], [51], [56], [105], other authors have adapted SVMS for dynamic hand gesture and action recognition [37], [62], [91], [94], [103]. Their dynamic gesture methods involve encoding the motion features as well as the shape features of the gestures into vectors classifiable by SVMs. As mentioned earlier SVMs have also been used to mimic the multi-stage classification structure of Adaboost Cascades in [22], [29]. Finally SVM related works mentioned in this section use a variety of features for classification, including Dense Trajectory Tracks[94], HOG [22][23], SURF[21], Shape and Texture Based Features [7], Fourier Descriptors [51], and Localized Contour Sequences combined with block based features [27] to name a few. An interesting approach is that of Sahoo et al. [105], which uses SVMs with Principle Component Analysis to classifier features extracted using a CNN called AlexNet. Rather than using the CNN for classification their work uses it for feature extraction. This variety of features used with SVMs shows the classifier's flexibility to work with different features depending on the nature of the problem and the designers' choices.

2.4.3 Neural Networks

Neural Networks are a competing classification method to SVMs. Neural Networks are very popular due to the fact that they can automatically select features to use for classification based on problem resolution constraints. These features would otherwise have to be selected by the designer often by using a trial and error method or running repeated experiments on test datasets. They require more data for training in general but have the advantage of keeping the model the same size even if the dataset increases. In addition, once trained, Neural Networks can be reused and adapted for different problems in a process called retraining. Retraining allows a neural network which was trained with a very large dataset for a generic task like object classification to be reused and adapted to a more specific task such as hand gesture recognition with a smaller dataset.

Neural Networks have also been used in a variety of ways to solve gesture recognition problems some of which can be seen in the works of [9], [10], [49], [53], [57], [58], [66], [67], [86], [90], [99], [106], [11], [107]–[111], [16], [20], [24], [28], [30], [36], [43], [63], [92], [77]. Both static [9], [10], [67], [108], [11], [16], [20], [24], [28], [30], [57], [66], [63], [88], [77], and dynamic hand gestures [43], [49], [90], [106], [107] have been recognized with neural networks. The applications to dynamic hand gestures show that Neural Networks are quite capable of using motion features for classification. Neural Networks have also been used to fit 3D hand models for hand pose estimation [36], [86] which is applicable to both static and dynamic hand gestures. Human action recognition which shares similarities to dynamic hand gestures has also been investigated with neural networks [109], [111]. Hand Detection is performed in [58] which uses a neural network to achieve very good skin segmentation which is then processed to find hand locations. The authors of [90] used neural networks to both temporally segment and classify dynamic hand gestures.

Hwang and Lee[9] using a neural network for static hand gesture recognition specifically for use as a computer interface. This idea is also echoed in [20] which proposes a framework called YOLSE which was used to classify ego centric hand static gestures as well as detect the locations of fingertips. The authors suggest augmented reality as their motivation for recognizing ego-centric gestures. User interfaces and Human Computer

Interaction (HCI) are also a motivation for the work done in [10], [11], [30], [49], [67], [86], [106], [107], [88], [92]. The work in [11] by Koller et al. presents how to use weakly labeled data to train neural networks on larger datasets. It is of particular interest because it achieves success in static hand gesture recognition that is independent of large changes in orientation using a large number of hand pose classes. This method can also cope with significant occlusion. Koller et al provide a very robust method for communicating with a computer using sign language. Sign language recognition is also the motivation for the CNN based work of Adithya and Rajesh [92]. [30] also uses ego centric data to train a neural network for hand gesture recognition for HCI. [67] maps recognized hand gestures to emojis. [106] and [49] analyze motion paths to allow users to draw numbers and letters that can be recognized by a computer. Molchanov et al [107] use dynamic hand gesture recognition to recognize swipe gestures. The 3D hand fitting model techniques of [36], [86] actively fit the model to the user's displayed hand and give a lot of control options in User Interface and Augmented Reality applications. The authors of [88] are motivated by HCI applications but they focus their efforts on dataset augmentation for CNN training. A simple type of neural network called an Extreme Learning Machine is used in[77] whose research motivation is human interaction with a robot. The authors use Extreme Learning Machines which are single hidden layer feed forward neural networks as well as dataset augmentation in their method. Human computer interaction and user interfaces are a continued research motivation and are still an open problem for hand gesture recognition which receives active attention.

The work of Liu et al [63] uses depth image data to train a CNN to perform static hand gesture recognition in 2D videos. The authors train a CNN to general pseudo depth images from 2D video and one to classify the gestures.

Object recognition is performed in [99] using a HOG Cascade for object detection and a Neural Network for classification. Object detection is used as an example application for the MobileNets Neural Network proposed in [53].

MobileNets [53] is a very interesting new approach because it is a powerful neural network classifier that is capable of running on mobile devices in real-time, and as such it has generated interest in computer vision. Computational savings are achieved with

separable convolutions. MobileNets runs on the TensorFlow framework [112] which is very popular in computer vision for neural network related methods. Sheng et al. [113] propose an improvement for MobileNets which improves performance for quantized models. MobileNets is used by [114], [115] for state of the art comparisons in a recent articles. The MobileNets neural network is of interest in the Hand Gesture Recognition system presented in this thesis.

2.4.4 Other classifiers

Some other classifiers have also been applied to hand gesture recognition and related problems. Rumyantsev el al. [32] uses Principle Components Analysis for hand gesture recognition and tracking. PCA is also used in [8]. Kerdibulvech [64] uses a Bayesian Classifier. [96] uses kernel regularized least squares with Fourier analysis for object tracking. This method also exploits the circular structure of object tracking data.

Zhou et al. use Finger Earth Mover's Distance for classification in their work in [60], [61]. Finger Earth Mover's Distance is an extension of Earth Mover's Distance which was described in the context of image retrieval in [116] by Rubner et al. [117] describes the work of Zhou et al. on Minimum Near-Convex Decomposition for shape representation. This method was used in [60], [61] to prepare samples for the Finger Earth Mover's Distance classification.

The work of [76] and [46] both use cascades of weak regressors to make strong regressors for pose estimation. Sun et. al. [46] extends the work of Dollar et al.[76] to use 3D hand models for hand pose estimation. It makes use of hierarchical skeletal properties to improve the result.

Hu et al. [87] have an interesting method which combines several types of classifiers to achieve both static and dynamic hand gesture recognition. Their works uses a custom regression based classifier to detect hand gestures in individual frames, then these labels are corrected temporally by a Bayesian classifier, and finally the sequences of gesture detections are classified as dynamic hand gestures by a Hidden Markov Model.

Jeon et al. use a fuzzy decision tree in [118] for hand gesture recognition. Random forests are used by the authors of [45], [47], [48], all 3 works focus on hand tracking.

[12] Lekova and Dimitrova use Symbol Relational Grammers to classify brightness and texture features around edges extracted with fuzzy clustering for static hand gesture recognition. Liu et al. [42] use dynamic movement primitives to classify dynamic hand gestures. Simple classification like nearest neighbor classification is used in [65] with scale invariant edge features. An interesting classification method is proposed in [119]. The authors use a MEMS accelerometer to record the motion of the user's hand which is then classified by using a sign sequence and template matching. This method presents itself as an alternative to time consuming model training methods. Determinantal Point Process is used to classify an interesting type of hand gestures in the work Huang et al. [31]. DPP is used for the classification of hand grasps. Hand grasps are like static hand gesture poses, with the addition of grasping different objects. These grasp gestures are very useful for understanding how humans interact with objects.

It is good to be aware of alternatives to popular classification methods. It may come in useful to solve future problems to either use a different method or just to approach it from that method's perspective.

2.4.5 Simple Logic Classifiers

A small number of works [13], [33], [34], [39], [40], [59], [68] use simple hand crafted logic for gesture classification. These methods do not use any learning and are entirely hand coded by designers. These methods typically classify gestures by using very evident features such as number of extended visible fingers, finger angle, and convexity defects. Although they do not compete with the performance of machine learning methods, these methods are quick to implement and provide reasonable performance for applications in controlled environments. They are quite useful when a hand gesture recognition problem can be solved in a simple way, and these methods often allow systems to use the position of user fingertips to control application activity.

2.4.6 Relevance to proposed systems

The systems presented in this propose make significant use Adaboost Cascades with HOG Features, Support Vector Machines with HOG features, and MobileNets Neural Networks. HOG features are light and scale transformation invariant when used with an image pyramid; which is why they were chosen. A more thorough comparison of all different feature types is beyond the scope of the thesis. Adaboost and SVMs were chosen because they are well studied models that are computationally lighter than other models such as neural networks. They were also relatively straight forward to train. MobileNets[53] was chosen because it is a popular neural network model designed for mobile vision applications. This means that its design is more suited to working in real-time vision applications, and thus it made an excellent choice for comparison to the Hand Gesture Recognition system.

Adaboost Cascades are trained with standard parameters.

Linear kernels are used with the Support Vector Machines for simplicity and performance. Non linear kernels are not used due to added design and computational complexity. A randomly selected subset of the whole dataset is used to keep the models small and running real-time.

MobileNets was selected as a Neural Network to use for the Gesture Recognition system because of its classification power and its ability to run in real-time on mobile devices and common desktop computers.

Machine learning is used for static hand gesture detection in all 3 proposed systems in this thesis. It is also used for static pose recognition of the tracked hand in the Hand Gesture Recognition with Hand Tracking system.

2.5 Energy Minimization and Optimization

Energy minimization and optimization method numerical methods are also used for gesture recognition and as components of larger gesture recognition systems. As opposed to using an Adaboost Cascade, an SVM, or a Neural Network, which can be approached

as black boxes, using optimization or energy minimization requires a more explicit statement from the designer of problem criteria that should be solved. If the engineer or designer can approach the problem whether its static or dynamic hand gesture recognition, hand detection or hand tracking from a mathematically rigorous perspective then optimization offers the advantage of allowing the designer to finely tune parameters of their problem or problem model using mathematical constraints.

The engineer or designer can directly define constraints and parameters for solving the problem, or their problem model, and they can pick and choose only what is useful for their system. This allows a higher degree of customization than the classification methods just mentioned. Optimization and energy minimization allows designers to tailor their solutions to their systems' needs. Evidently there also is a greater degree of complexity to using these methods.

Numerical optimization has been used in [11], [35], [38], [44], [48], [117], [120]–[123]. It was used in [123] by Khamis et al. for 3D hand model fitting on depth data, which is highly applicable for both hand tracking and hand gesture recognition. The method fits both mesh and skeleton of the hand model to the depth data. Similarly, 3D hand model fitting to depth data techniques are used in [38] which models the hand as a skeleton covered in simple spheres. Optimization is also used for hand tracking in [48], which also fits a 3D hand model to depth data. Koller et al. use expectation minimization, which is an iterative optimization method, in [11] in combination with a Convolutional Neural Network for hand gesture recognition. Their method allows them to use weakly and inaccurately labeled data to augment their dataset and thus their system's performance. Alon et al. [35] find optimal hand motion tracks from sequences of hand detections using optimization. These tracks get matched to dynamic hand gestures classes. Some authors of [35] also have also used optimization in their previous work in hand tracking [44]. [122] applies optimization for template tracking for objects. Their work builds on their previous work with Active Appearance Models [121] as well as that of others [120]. Active Appearance models use optimization, which the authors of [122] use as part of their tracking strategy. Numerical Optimization is also used for shape representation via

minimum near-convex decomposition in [117] which the authors successfully used in their work with hand gesture recognition with Finger Earth Mover's distance in [60], [61].

Optimization in some form is used in Adaboost Cascades, Support Vector Machines, and Neural Networks, as well as other classifiers. The classifiers all try to find the best or optimal solution to the classification problem presented by the training data while abstracting at least some of the optimization mechanics from the designer. Using optimization or energy minimization directly can be thought of as using computational classification in the raw sense. Which explains the added complexity in stating the problem criteria, and the higher degree of tuning that is available for solving the given problem.

Optimization also appears in other works in the later *Tracking by Optimization* section.

2.5.1 Relevance to proposed systems

The systems described in this thesis do not use direct numerical optimization or energy minimization for their applications. It is none the less useful to understand alternatives to optimization abstracting classifiers in order to be able to approach challenging problems with fresh perspectives and to make use of research contributions in related fields for new applications. Depending on the problem it may be easier to state and solve it as an optimization problem.

2.6 Hand Tracking

Hand tracking is a very useful thing for visual control systems. It allows a user's tracked hand position to be used to issue commands in a user interface. Hand tracking has its particular goals, and it also shares similarities with related Hand Gesture Recognition methods.

2.6.1 Hand Tracking vs. Hand Gesture Recognition

The goal of hand tracking is to retain and update the position of a user's hand as it moves in a video. Tracking methods are used in the Hand Tracking oriented works of [33], [34], [46]–[48], [52], [86], [123]–[127], [35], [128]–[137], [36], [138]–[147], [38], [148], [39]–[41], [44], [45], [149]–[153] and also in the Hand Gesture Recognition focused works in [32], [34], [50], [56], [90], [91], [97], [107], [128], [154]–[156], [35], [157]–[166], [37], [167], [39]–[43], [49]. The main difference is that while both are interested in keeping track of a user's hand and understanding what the hand is doing, Hand Gesture Recognition is focused on determining static [32], [34], [155], [158], [159], [165]–[167], [37], [39], [41], [50], [56], [91], [128], [154] or dynamic [35], [42], [155]–[164], [43], [49], [50], [56], [90], [97], [107], [128] hand gestures. Static gestures being hand in certain static poses, and dynamic gestures being hand motions in defined structured paths. However there is significant overlap which can be seen in the work of Bambach et al.[128]. Their human activity recognition system can classify single frames making it static gesture recognition, or a series of frames for dynamic gesture recognition. Hand tracking on the other hand has a different goal. The goal of hand tracking methods is to determine where the hand is within the frame or 3D space. When hand tracking methods reach a certain level of complexity they are also capable of understanding what the hand is doing as well meaning its pose structure. The two categories of methods differ also in the amount of prior information. A hand gesture recognition method generally does not know where the hand is or if there is one present when it begins operation. It determines this information using an incoming video frame. Hand tracking methods however need a more defined starting point and require initialization to run. These methods search for the hand in subsequent frames, determining a best hand location by observing the previous location and comparing it to the incoming information. Hand tracking techniques must also determine exit conditions to stop the tracking algorithm when the user's hand is no longer in view. Stable and temporally continuous output is also desired of hand tracking algorithms. The tracked location of a user's hand should be smooth from frame to frame and only move when the user's hand moves. Smoothness and stability are requirements for robust and responsive user interfaces. Choppy position and unstable motions can issue false commands and make an interface unusable. For example having a control system with a choppy mouse cursor would make it unreliable for use.

There is a significant and subtle difference between dynamic hand gesture recognition [35], [42], [155]–[164], [43], [49], [50], [56], [90], [97], [107], [128] and hand tracking [33], [34], [46]–[48], [86], [123]–[128], [35], [129]–[138], [36], [139]–[148], [38]–[41], [44], [45], [149]–[153]. Hand tracking seeks to find the explicit position of the user's hand in a video. Dynamic hand gesture recognition seeks to determine the structure of the hand motion and it also is interested in detecting when a specific type of structured motion occurs. The methods described in [42], [43], [160]–[164], [49], [50], [90], [97], [155], [156], [158], [159] perform dynamic gesture recognition using hand tracking. The authors of [42], [43], [162], [163], [49], [90], [97], [155], [156], [159]–[161] use hand trajectory in their methods. The hand trajectories extracted using hand tracking are then matched to specific gesture classes. The work of Alon et al [35] shows how it is possible to use trajectory without knowing the explicit position of a user's hand. Their dynamic gesture recognition system does trajectory matching without spatial or temporal segmentation. Simple colour and motion segmentation features are used to produce multiple potential "hands" in video frames. Then using these locations the system finds optimal trajectories over series of frames. Only motion trajectories that are temporally consistent are considered by the system. All other inconsistent trajectories are rejected. Alon el al are able to achieve dynamic hand gesture recognition, without explicitly knowing where a hand is in a video frame. Dynamic gesture recognition without explicit hand tracking is also achieved in [56], [107], [157]. The system in [157] detects the presence of hands in a ROI. Hand presence is then used to temporally segment video clips to classify as dynamic gestures. Hand motion gestures in 4 different directions are detected in [56] using motion detection features and classification with an SVM. A CNN is used for the classification of videos of dynamic hand gestures in [107]. The gesture class detected can be used to determined a general direction of motion. For example a swipe right gesture indicates that a hand moved to the right. Explicit hand position, however is not determined by the system.

Additionally while the locations of both hands are explicitly determine in the methods of [128], [158], [164] this information is not used in the gesture classification, rather it is used to localize the focus of the training so that it can be used to determine what the

hands are doing relative to each other rather than where they are. In these cases relative position is used rather than absolute position in the frame. The difference between hand tracking and dynamic hand gesture recognition comes down to the way the latter is interested in the structure of the hand motion, while the former is interested in location of the target. Dynamic hand gesture recognition often makes use of hand tracking but it is not limited to such methods so dynamic hand gesture recognition methods will not always be able to provide explicit hand location. Although the locations of both hands are determined explicitly in the works presented in [128], [158], [164], explicit hand location is not used in gesture classification. It is used to localize the focus of the training, but the goal is to determine where the hands are relative to one another instead of determining where they are in the frame. Instead of absolute position in the frame relative position is used.

The main difference between hand tracking and dynamic hand gesture recognition is there what aspect of the user's hand presence is the focus. Hand tracking is interested in the explicit location of the target hand. Dynamic hand gesture recognition is interested in the structure of the hand motion. Dynamic hand gesture recognition methods will frequently make use of hand tracking strategies, but dynamic gesture recognition is not limited to using hand tracking. Other strategies exist to detect hand motion structure. Therefore a dynamic hand gesture recognition method will not always be able to provide explicit hand location.

2.6.2 Similarities to object and face tracking

For the systems described in this thesis object tracking [78], [94], [96], [122], [168]–[173] and face tracking [122], [174] are also of interest. Object and face tracking uses colour segmentation and template matching both of which are used by the hand tracking component of the Gesture Recognition and Hand Tracking system presented in this thesis. Face detection and tracking is sometimes used in hand gesture recognition and tracking methods and as such the face tracking work of Bradski [174] using the proposed CAMShift algorithm is also of interest.

Template tracking as in done in [122], [170], [172] and template matching for the detection component of the Tracking-Learning-Detection framework [173], is of particular interest to the work presented in this thesis. A template based object tracker is used to track each individual fingertip of a tracked hand in the Gesture Recognition and Hand Tracking system.

2.6.3 2D vs 3D

Some hand tracking techniques are designed for 2D[32], [33], [47], [49], [52], [56], [78], [86], [94], [96], [122], [128], [34], [135], [142], [146], [155], [160], [165], [168]–[171], [35], [172]–[174], [39], [41]–[45],[149]–[151], while others are 3D techniques [33], [36], [91], [107], [123]–[127], [129]–[131], [37], [132], [134], [136]–[141], [143], [144], [38], [145], [147], [148], [154], [156]–[159], [161], [162], [40], [163], [164], [166], [167], [43], [46], [48], [50], [90], [152], [153]. 3D methods have become more popular in recent years due to the availability of the Microsoft Kinect and other cheap depth sensors. 2D hand tracking techniques operate using only video images. 3D techniques on the other hand use the added dimension of depth to track the 3D location of the target. Depth sensors have become popular among other reasons because depth can easily solve the problem of segmenting a user's hand from the background. Depth segmentation is often used in many different methods [33], [36], [136], [138], [140], [143], [144], [154], [156], [158], [159], [162], [37], [164], [166], [167], [38], [50], [90], [124], [129], [130], [134]. It is sometimes assumed that the user's hand is the closest object to the camera, or the user's body is the largest connected object [36]–[38], [136], [138], [156], and this largest or closest body is the target. Other methods set a depth threshold with which to segment with this threshold being either adaptive or fixed [33], [50], [124], [130], [134], [144], [158], [159], [166]. The works in [38], [140], [143] use visual colour cues to drive their depth segmentation. The SDKs of certain depth sensors provide the body joints of kinematic skeletons of detected users. These body joints are used as starting points for segmentation to segment are user's hand or body in the methods in [90], [129], [154], [156], [164], [167]. In [162] template matching is used to drive a region growing algorithm which segments out the user's body. In general more attention must be given to the segmentation problem when using a 2D tracking method. An interesting method in

recent literature by Simon et al [125] demonstrates the fitting of a 3D hand model to multiple 2D images using calibrated cameras. The fitting is performed with planar projection and optimization. A 3D hand model is also fitted to multiple 2D hand images by Sridhar et al [133]. 3D techniques allow using powerful 3D hand models that model joints and fingers of a hand it 3d space. While depths sensors can are be used without any 3D models such as is the case in [33], [37], [158], [159], [162], [40], [43], [50], [107], [124], [136], [144], [157], articulate 3D hand models are used in the methods presented in [36], [38], [131], [132], [134], [137]–[141], [143], [145], [46], [147], [148], [166], [48], [123], [125]–[127], [129], [130], [153], and body skeletons are used in [90], [91], [132], [154], [161], [163], [164], [167] which are provided by the SDKs of the depth sensors.

A unique hand tracking method is that of Mueller et al.[149] which shows the blurring between 2D and 3D methods. Their method fits a 3D hand model onto a single RGB video image. This is done with a RegNet CNN for joint detection, energy minimization for hand model fitting, and a synthetically generated 3D hand gesture dataset. This is an interesting work because it achieves good 2D hand tracking while relying on 3D data for training. So while it is a method intended for 2D applications, it cannot be used without 3D data. Thus it is probably more appropriate to consider it a 3D method because the limitations result from the training data.

2.6.4 Tracking by Detection

Tracking by detection is achieved by detecting a hand in a given video frame, and if there are multiple candidates the best one is chosen using some sort of metric. Often Euclidean distance to the previous tracked location is used for simplicity. Detection is performed on each frame and the position of the tracked hand is updated using the best candidate once it is found. Colour segmentation is often used by tracking by detection methods [33], [34], [56], [130], [133], [136], [138], [143], [160], [162], [165], [169], [35], [37], [41], [42], [45], [47], [49], [52], [151]. Segmentation is done with skin colour in the works of [33], [34], [56], [130], [133], [136], [138], [160], [162], [35], [37], [41], [42], [45], [47], [49], [52].

Depth is often used to perform segmentation as was explained in the earlier *2D vs 3D* section. The authors of [33], [36], [136], [138], [140], [143], [144], [154], [156], [158], [159], [162], [37], [164], [166], [167], [38], [50], [90], [124], [129], [130], [134] use depth segmentation in order to produce detections. Certain depth sensors can provide a body skeleton of a detected user. [90], [91], [129], [154], [156], [161], [163], [164], [167] make use of body skeletons to detect their users. Skeletons are used in combination with depth to produces segmentations by the authors of [90], [129], [154], [156], [164], [167].

Contours can be processed with a variety of geometric tests to produce detections as well. Contour analysis methods are used in [33], [37], [38], [50], [52], [129], [156], [158], [159]. The work of Plouffe and Cretu [158] takes the closest depth pixels in depth images and then uses region growing to find and segment hand contours. Region growing is also used to find fingertips in [38], [129]. Contour analysis is also used for fingertip detection in [33], [37], [50], [52], [156], [158], [159]. Palms [33], [50], [52], [158], [159] and wrists [50], [52], [159] can be detected by contour analysis as well.

In the works of [35], [41], [44], [47], [56], [171], [39] tracking is done with motion detection. Motion detection is performed by measuring the colour difference between successive frames of corresponding pixels. Pixels that have significant differences are labelled as containing motion. These pixels are then grouped into detections. Motion detection is actually a colour difference feature. It can also be thought of as a background subtraction feature with a rapidly updating background model. The challenge of motion detection is in how to update the background model in such a way that the object is entirely segmented from the background. Poor motion detection strategies can lead to detections only being found on the edges of objects of interest which can lead to poor segmentation. In the work of Alon et al.[35] the authors present an interesting case using motion detection. They combine simple motion features with skin detection features to produce multiple hand detections and the best candidates are determined via temporal optimization.

Detections are performed with other features as well. Saliency is used in [47], [78]. Saliency measures image variability and produce detections regions where significant objects are found. Using saliency produces detections by finding visually interesting

areas by measuring colour intensity, gradient , texture, or similar properties. Motion features can be considered saliency features as well. Another type of feature called Oriented Radial Distribution (ORD) is used by Suau et al [144]. Their method uses ORD at different scales to detect hands as well as fingertips.

Template matching can perform detections by search a region against a database as was done in the works of [52], [96], [162], [173], [152]. The database can have multiple candidates [173][96], or as in the cases of [52], [162], and [152], just one saved candidate template exists for each target. One type of template is used to detect extended fingertips, and another is used for folded fingers by Grzejszczak et al[52]. A nearest neighbour classifier is used to match templates by Kalal et al [173]. Matches produce detections which are then used by the tracking component of their algorithm. Hand detection is performed by Xu et al [162] using chamfer distance. Henriques et al.[96] use a kernel based detector to produce detections which are used to grow their database of templates. As the algorithm runs and more detections become available the template database is used to retrain their kernel based detector. [152] uses a single template to match to the user's hand in a depth image and updates the tracked location with the best match.

Tracking by detection methods make use of machine learning to produce detections for tracking. Different machine learning models are used including SVM [56], [96], [155], [157], Neural Networks[43], [86], [107], [125], [128], [141], [142], [145], [146], [164], and Random Forests [38], [45], [47], [127], [130], [132], [135], [138], [166]. Adaboost models are used to intialize hand tracking in the works of [34], [39], [40] with tracking in subsequent frames being performed by CAMShift and in the case of [39] Lucas-Kanade optical flow is used.

The contour based tracking portion of the Gesture Recognition with Hand Tracking system presented in this thesis is a tracking-by-detection method. It uses adaptive colour segmentation to segment out the user's hand. The closest candidate to the hand location in the previous frame is selected via Euclidean distance. The parts of the user's hand are detected using its contour shape and the geometric contour analysis of the system's hand models. The contour tracking method benefits from the unimodal filtering used in the Hand Gesture Recognition system. Unimodal filtering is important because it improves

the quality of the segmentation resulting in cleaner better defined contours. The contour tracking is initialized with by successfully recognized hand gesture found using the Hand Gesture Recognition system. This gesture's contour is taken as the first successful detection.

2.6.5 Tracking by Optimization

Tracking-by-optimization methods perform a numerical optimization in order to fit a model onto observed data. The result is a model that fits the observed data and that as the data changes over time the model changes to fit the data resulting in a tracking of the target producing the data. For hand tracking tracking-by-optimization generally means fitting a 3D articulated hand model to depth data [36], [38], [138]–[141], [143], [145], [147], [148], [48], [123], [127], [129]–[132], [137], [153]. There are exceptions such as the works in [125], [133] where a 3D model is fitted to multiple 2D images. Another interesting case is the work of Nolker and Ritter [86] where 2D fingertip detections from a single 2D image processed with a neural network are used to fit a 3D hand model. Their work proves that it's not necessary to have depth images or multiple views of a hand in order to fit a 3D hand model. This is also the case for the work of Mueller et al.[149] where a 3D model is fitted onto a single RGB video image. 3D methods are useful because they can handle self occlusion of the user's hand. They also give a complete estimate of the user's hand pose. These benefits come at the cost of additional hardware such as depth sensors and calibrated stereo cameras in order to acquire the 3D data. Computational cost is also higher for 3D methods, which is alleviated by using GPUs to run algorithms in real-time.

Many tracking by optimization approaches use energy minimization methods. Energy minimization is very flexible, because it allows customization of the terms in the mathematical models used for tracking. These terms can encourage or discourage a variety of desired properties [35], [38], [133], [138]–[141], [143], [145], [147], [148], [44], [48], [123], [127], [129]–[132], [149], which allows the designer to fine tune their minimization approach. Properties that have been used by tracking works include terms to enforce temporal continuity [35], [44], [48], [129], [138], [143], [145],[140], [149], to fit the hand model joint locations to data [48], [123], [129]–[132], [138], [141], [145],

[148], [149], to limit joint angles so that estimated hand poses are physically realistic [48], [123], [129], [133], [138], [143], [145], [147],[140], [149], to penalize self intersection of the hand model[38], [48], [129], [130], [143],[140], and terms that fit the mesh of a hand model to 3d depth images [38], [48], [148], [123], [129]–[131], [138], [139], [143], [147],[140]. A hierarchical particle filter such as the one used in [153] can also be used to fit the mesh of a hand model onto a depth image. Fitting of hand models onto 2d silhouettes with projections is also done in the algorithms presented in [131]–[133], [141], [143], [145], [148]. The drawback of using energy minimization is that the flexibility in design comes at the cost of increased design complexity. An algorithm designer is required to have more specific known of the problem that is to be solved. In comparison with other techniques energy minimization is not a black box that requires the setting of only a handful of parameters. Specific selection and definition of all the terms is required in an energy minimization method to obtain a good result.

The same type of tracking by optimization that is done by traditional energy minimization methods can also be accomplished with neural network models [36], [86], [125], [131], [132], [137], [146]. Using neural networks abstracts the term definition complexity from the algorithm designer. Auto encoders are also used in [137] for their 3d hand tracking.

Optical flow is used by the algorithms presented in [35], [39], [44], [45], [47], [94], [124], [151]. It is used in [94] for action recognition by Wang et al. The action recognition is done by tracking dense features which can be used for hand tracking. Optical flow is specifically used for hand tracking in [39], [45], [47], [124], [151]. There is an interesting case in the works of Alon et al [35] and Yuan et al [44]. In these works optical flow is used to generate motion detection features which are used by a tracking by detection method. Additionally the authors run a energy minimization method over these detections in order to find optimal paths of moving user hands. In this case optical flow which itself is an optimization method is used for detections.

There are simpler and well known optimization methods that are used by the authors of [33], [34], [168]–[171], [174], [37], [39], [40], [45], [47], [136], [140], [162]. In [140] Tkach et al. perform 3D hand tracking by propagating Kalman filter style uncertainty in

their energy minimization method. Kalman filters are also used for hand tracking in [33], [37], [136], [169]. In other algorithms hand tracking is done with Mean shift [45], [136], [168],[170], CAMshift [34], [39], [40], [162], [171],[174], and particle filters [45], [47], [168]. These optimization methods are used in the works in [33], [34], [37], [39], [40], [45], [47], [136], [162], [169],[174],[168] to temporally stabilize the detection inputs in order to produce smooth tracking trajectories.

The optimization approaches used by some tracking algorithms are unique and are not easy to categorize. One such example is proposed by Kalal et al. [175]. Their Median Flow Algorithm calculates forward and backward trajectories of interest points of a given object from a series of video frames. The trajectories are retained of only those points which produce the same trajectory path going forwards and backwards. This median flow algorithm is used by Kalal et al. in their Tracking-Learning-Detection framework [173]. SVMs are used to filter out incorrect poses in the human pose estimation work of Charles et al. [135]. SVMs are also used in [157] for a gesture recognition method which can infer crude motion from the gesture. Graph matching is used by Suau et al[144] for fingertip tracking. Tang et al. [126] fit a 3D hand model using regression forests in their approach. Dynamic Time Warping is used by Cheng et al[159]. Their work uses DTW to fit hand motion trajectory to a series of strokelets for their classification method. The work presented in Maqueda et al[155] uses a Bayesian tracker as proposed by del-Blanco et al[176]. Their gesture recognition system uses the tracker for hand tracking. A cascade of regressors is used to perform 3D hand tracking in the system of Sun et al[46] . Liu et al.[150] use a Kernalized correlation filter for hand tracking which they specifically formulate for scaling changes. Their method also uses a kalman filter with adaptive model updating to handle hand occlusion with other objects.

Template based methods are simple optimization methods. These methods use a stored template image and do a sum of least squares minimum value search within a frame image. They can be used for tracking as is the case in [122], [170], [172]. Template matching is also used as part of a larger tracking strategy by Kalal et al. [173]. The detection component of their Tracking-Learning-Detection framework uses template

matching. A comparison of template tracking methods and histogram matching methods is made by the authors of [170].

The hand tracking algorithm presented in the Gesture Recognition with Hand Tracking System chapter of this thesis uses template tracking. Template tracking can be viewed as a simple tracking by optimization method. An image of a user's given fingertip is the objective that is sought within the current frame. This current frame or input image is the search space. The search space is constrained with the applied skin colour segmentation which removes the influence of the background and simplifies the search. This skin segmentation is in fact a mask applied on the data term. The contour based tracking delaminates the search region for the template search for a given fingertip. This can be viewed as a range term of the optimization. A simple sum of least squares minimization method is used to find the optimal location of the user's fingertip in the search region. The result is that each fingertip is tracked with a template tracker that provides a more temporally stable tracking location. This smooth template tracking optimization method is guided by the less temporally smooth contour based tracking, allowing for the template tracking to run in real-time by using smaller search regions.

2.6.6 Using Multiple Strategies

A significant number of works use multiple strategies to obtain better tracking results [33], [34], [47], [86], [124], [127], [129], [130], [132]–[135], [35], [136], [138], [140], [141], [143]–[146], [155], [159], [36], [162], [169], [171], [173], [37]–[40], [44], [45], [153]. While a given method is able to track a hand or object by itself, using multiple strategies can overcome the short comings of each individual method. Using multiple strategies can simplifying the tracking problem as well by breaking it down into separate steps. Tracking methods can focus on simpler goals which results in better tracking. Tracking methods can require detections in order to perform tracking which necessitates the use a detection strategy.

Detections drive the optimization stages of tracking methods in a range of works. In the works of [35], [44] hand detections which are done with motion [44] and motion and

colour [35] features are used to drive energy minimization methods. The minimization methods find hand motion trajectories. Adaboost is used for hand detection in the methods in [34], [39], [40], which initialize CAMshift tracking stratgies. Colour segmentation and for [39] motion detection is used to drive these CAMshift tracking approaches subsequent frames. The CAMshift strategy of [39] is also combined with Lucas-Kanade optical flow to provide more accurate local tracking. Other works use CAMShift strategies as well [162], [171]. [162] uses colour, depth and template matching, and [171] uses motion features to drive the tracking with detections. A Mean-Shift and Kalman filter tracking strategy is used by Elmezain et al [136] with hand detections that were detected with depth and skin colour. The work presented in [37] also detects hands with depth and skin segmentation, and it uses a Kalman filter for tracking. Hand tracking is accomplished by Yeo et al.[33] also by using skin segmentation with a Kalman. The hand trajectory in [159] uses a dynamic time warping approach. The system detects hands with depth segmentation and uses these detections with its optimization method. Particle filters are used in [45], [47] for tracking. These particle filters are driven by hand pixel detections which are produced using random forests. In the hand tracking done by the work of Maqueda et al[130] SVMs are used to perform hand detections which drive the tracker proposed by del-Blanco[176].

A good example of combining two methods to overcome their individual shortcomings is the work of Tang and Zhang [168]. Their object tracking system combines mean shift and a particle filter its tracking approach. The slower performance of particle filters is overcome with mean shift tracking. The lower accuracy of mean shift is in turn overcome by a particle filter. The overall tracking result is improved because the two tracking methods share information.

Out of the 3D hand model fitting techniques that have already been mentioned, the works in [36], [38], [132], [134], [137]–[141], [143], [145], [147], [46], [148], [166], [48], [123], [126], [127], [129]–[131],[153], fit their hand models to depth images of hands. In order to obtain depth images of hands from general user in a scene depth images there has to be some kind of hand depth image extraction in place. In many of these works there in an explicit mention of which tracking by detection strategy they used to obtain their hand

depth images. However other works only explain their fitting method, assuming that the hand depth images are already available. It is implied that a detection method is used to produce the hand depth images. Therefore it can be assumed that in order to have a 3D model to depth fitting system run in real-time some sort of information sharing is used. It may be possible to obtain good 3d hand model fitting results on an entire depth image of a scene with a user and a displayed hand. Having a depth image of the user's hand however, will certainly help the fitting to converge faster.

It is worth mentioning among the 3D hand fitting methods already mentioned the interesting approach presented in [153] by Park et al. Their work uses a hierarchical particle filter to fit a 3D hand model to depth. In additional to using depth images the authors use a gyroscope to find angular velocity of the user's hand which is then used to correct the hierarchical particle filter optimization. This allows for better results in the presence of motion blur.

In certain tracking algorithms, initial detections of fingertips and joints are used to drive the optimization. Hand joint locations are estimated using CNNs by Ge et al.[141] who use these locations with their energy minimization method for hand model fitting. Neural networks are also used to find joint locations by Thompson et al.[132] who use these location estimates to constrain their fitting method. Mueller et al[145] in their 3d tracking method likewise use a CNN strategy to detect joints and hands. Fingertip detection is done using a neural network by Nolker and Helge [86], who use these detections as well as another neural network to fit a 3d hand model. Fitting a 3d model to fingertip detections is also done by Qian et al[38]. The authors of [129] use fingertip detections produced by the fingertip detection Qian et al[38]. The fingertip detections constrain their minimization method. Tzionas et al[130] also use fingertip detection to drive their 3d hand tracking energy minimization method. Tzionas et al use random forests for detecting fingertips.

There are two major components of the hand tracking algorithm used by the Gesture Recognition with Hand Tracking system presented in this thesis. A rough estimate of the user's hand is provided by a contour based tracking component. It tracks the general location of the user's hand and its palm and fingertips. A second template based fingertip

tracking component provides a refined and stable tracked location of the user's fingertips. The contour based tracking is a track-by-detection method. Contour tracking uses colour segmentation. The template tracking of the fingertips is a simple tracking-by-optimization method. It matches saved templates of the user's fingertips to locations in an input frame using a sum of least squares matching search method. The contour tracking guides the template tracking. It provides colour information which reduces the effects of the background in the template tracking. It also provides location range constraints which results in smaller search regions for the template tracking. Having smaller search regions simplifies the search and prevents a template tracking from jumping between fingertips of adjacent fingers. It also allows the template tracking to run in real-time, otherwise it would be too computationally costly to perform the template search multiple times on the entire frame. Lastly the contour tracking initializes new trackers whenever it finds a new fingertip. The template tracking is able to better track the user's fingertips providing better more temporally stable and smooth fingertip locations than the contour tracking can provide. In addition the template tracking deactivates any degenerate trackers that are no longer tracking a valid fingertip. When a tracker is deactivated the template tracking informs the contour tracking that an additional template tracker is available if there are new untracked fingertips. The stable fingertip locations provided by the template tracking makes the hand tracking algorithm a viable option for UI applications. The accurate hand motion can be used to map application controls. Using multiple tracking strategies and sharing information achieves a better result. The contour tracking overcomes the computational load of the template tracking by producing refined search regions. The template tracking overcomes the lack of location precision of the fingertips that the contour tracking suffers from and achieves a more stable tracking of the user's fingertips.

2.6.7 Relevance to proposed systems

The hand tracking component presented in the Gesture Recognition and Tracking system has two components: a contour hand tracking component most like [33], [34], [37], [39], [40], [45], [174] and a template based fingertip tracking component which is most like general object trackers found in [78], [94], [96], [122], [168]–[173]. The hand tracking

has the most similarities to [33] because of its shape analysis hand tracking methods used by the contour tracking. It uses simple nearest contour tracking like [41], [42], [49], [50] subject to a distance constraint limiting the maximum distance between successive frames for a valid contour. The tracking relies on the colour segmentation to accurately find the hand contours in each frame. The contour tracking is a tracking by detection method and it shares similarities with other tracking by detection methods as described earlier.

Because the goal of Gesture Recognition and Tracking system is the tracking of objects (hands) and multiple sub-objects (fingertips) the system shares similarities to general object trackers such as that of [173] where a detector and a tracker use information from the other to reduce errors in their own analysis. Some inspiration was drawn from [173] to improve the performance of the hand tracking algorithm's template tracking component by getting initial estimates of the fingertip locations from algorithm's contour based hand tracking. This allows the algorithm to create refined search regions for the fingertips in the given frame. The refined search regions only contain one fingertip which allows the template trackers to make stable matches and reduces the risk of switching to track the wrong fingertip. Thus the contour tracking guides the template tracking to reduce the error. The template update strategy of the template fingertip tracking is that the templates are updated when an untracked finger is detected by the contour based tracking. A template is removed when it cannot match to a valid finger contour which is extracted if present with colour segmentation. The template tracking is a tracking by optimization method and it shares some similarities with other tracking by optimization methods as discussed earlier.

2.7 3D Methods

3D vision techniques involve acquiring depth data for objects in video and using that data to solve vision problems in 3 dimensions. 3D vision has been of interest since the early work by Lucas and Kanade [177], Tsai [178], Horn[179], and Zhang [180]. These methods use image correspondences from either multiple image views or stereo cameras

to find depth information. Some additional examples of acquiring depth data with a single camera can be found here [181]–[183].

3D vision techniques have also been applied to hand gesture recognition and tracking [36], [38], [126], [127], [129]–[134], [137], [138], [46], [139]–[141], [143], [145], [147], [148], [154], [161], [163], [48], [164], [166], [167], [50], [86], [90], [91], [123], [125], [153], as well as 3D hand dataset collection [148]. These works benefit from recent advances in cheap depth sensors like the Microsoft Kinect and Intel's Creative Interactive Gesture Camera.

The works in [36], [38], [131]–[134], [137]–[141], [143], [46], [145], [147], [148], [166], [48], [123], [125]–[127], [129], [130], [153], specifically use 3D hand model fitting to estimate the fine pose of the user's hand to depth data. Works in [90], [91], [132], [154], [161], [163], [164], [167] use body skeletons of the user provided by the SDKs of the depth sensors used.

Depth data can be treated as a projection of the real 3D space and 3D modeling techniques can be applied to hand gesture recognition, via energy minimization and machine learning methods. This is further described in earlier sections on tracking by optimization.

The idea is highly intuitive because hands are 3D articulating objects, and what better way to understand what they are doing then by using a 3D articulated mesh and skeleton model and fitting it to input depth data. A 3D articulated model is fitted to a hand which is also a 3D articulated object. 3D models have hierarchy built into them which means that the individual bones of an object always have their position expressed and calculated relative to their parent bone which culminates at the root. This is also true of human hands, because the location of a finger is dependant of the location of the palm and that in turn is dependent on the location of the wrist. The authors of [46] found that by using a hierarchical fitting approach they were able to fit their hand model better to depth data than without considering hierarchy. These 3D hand model fitting works achieve fine pose estimation of the user's hand, with the location and orientation of the palm and each finger being known. As such they know the explicit orientation of the hand and this

effectively solves both the static hand gesture pose recognition and the hand tracking problem.

3D methods offer a powerful hand gesture recognition and tracking solution when a reasonably accurate depth sensor and sufficient computational power is available. The main drawback for 3D methods is primarily that high quality 3D joint labeled data of RGB-D datasets is lacking as explained in [148]. High quality RGB-D datasets with accurate 3D joint labels for the hand models are difficult to obtain. There is also a higher computational cost when working in 3 dimensions as opposed to 2. Despite these challenges the authors of [31], [39], [38], [48], [141], [143], [145], [147], [126], [129], [132]–[134], [138]–[140], achieve real-time results. These works demonstrate the 3D fitting techniques are viable for hand gesture recognition and hand tracking systems. The work presented in [149] even shows how to fit a 3D hand model to a 2D video image. This method is designed for 2D applications but it is at its core a 3D method because it requires 3D hand pose data.

2.7.1 Relevance to proposed systems

The proposed systems do not use depth data and are separate from 3D model fitting techniques. A web camera will continue to be cheaper than a camera of similar image quality with a depth sensor for the foreseeable future. The computational cost is typically lighter for vision methods with 2D video signals compared to those that use 2.5d video + depth signals. However 3D hand model fitting is very interesting because of the huge potential for robust user interfaces, and this is definitely an avenue for potential future research.

2.8 Dataset Generation

The authors of [148], [184] address some interesting ways to collect datasets for gesture recognition and tracking applications. These are of general interest to anyone who is working in gesture recognition.

2.9 Gap Analysis

While there are many different approaches in Gesture Recognition and related fields, only the [7], [9], [21], [22], [25], [27], [29]–[31], [33], [35], [36], [10], [37], [38], [41]–[45], [47]–[49], [11], [51], [56], [58], [60], [61], [66], [89]–[91], [12], [14], [15], [17], [19], [20] work in complex backgrounds. Of these only [7], [9], [25], [27], [29]–[31], [33], [35], [41]–[43], [10], [44], [45], [47], [49], [51], [56], [58], [66], [89], [11], [12], [14], [17], [19], [20], [22] work solely using video. Furthermore, static hand gesture recognition is only the focus of [7], [9], [27], [30], [33], [41], [51], [56], [66], [10]–[12], [14], [17], [19], [20], [25],[22], and [31], while other methods are concerned with dynamic gesture recognition[35], [42], [43], [49], [56], hand detection[17], [22], [29], [35], [43], [44], [58], [89], and hand tracking[35], [43]–[45], [47] respectively. Note that there are some overlapping cases such as [35] which is interested in hand detection, tracking, and dynamic hand gesture recognition. Hand detection being very similar however it does not account for pose of the user's hand. In the case of our Gesture Recognition system the goal is to be able to map commands to specific static hand poses to allow the user a higher level of initialization control.

Hardware based methods such as data gloves or accelerometers, limit the user and increase cost. 3D methods of any kind require well labeled 3D datasets which are difficult and laborious to obtain as explained in [148]. The research towards automated 3D dataset annotation [125], [132], [148] shows that obtaining 3d annotated data is a significant research challenge and active work is being done to reduce the complexity and difficulty of this problem.

Different authors use different static gestures. No standardized set has been defined. The authors of [7], [9], [29], [30], [33], [41], [51], [56], [66], [89], [10], [11], [14], [17], [19], [22], [25], [27], [21] use the same static hand gestures [7], [9], [33], [41], [51], [89], [10], [11], [14], [17], [19], [22], [25], [29], [21] or similar static hand gestures [7], [9], [33], [41], [51], [56], [66], [14], [17], [19], [22], [25], [27], [29], [30], [21] as our work. Other works use different gestures [12], [20], [30], [31]. The authors of [20], [30] use egocentric gestures where the back of the hand is visible instead of the palm, the gestures of [12] are similar to egocentric gestures because they also present the backs of the hands.

Lastly, [31] use grasping hand gestures which model hand object interactions and are not the focus of our work.

Datasets are often times too small for algorithms that require a large amount of training data. This limits small datasets to be used for SVM based approaches which don't require large amounts of data. This is the case for Ghosh and Ari who use 200 images per gesture in [66] and 240 per gesture in [27]. The work in [41] uses 720 images overall. [21] also only uses 200 images per gesture. Pisharady et al[7] use the NUS II dataset which is 275 images per class. Their dataset is publicly available online. However they do not report per class performance making comparison opportunities limited.[10] use only 55 images per gesture in their dataset. The work of Koller et al[11] has 3361 manually labeled images containing 45 hand pose classes. Their method uses weakly labeled data to train CNN models. Comparison to other methods is difficult however, because the weakly labeled data does not have frame level labeling so if your method cannot make use of weakly labeled data it cannot use this dataset. As such within the scope of the labeled frames, 3361 total instances for 45 hand poses is too small. The authors report that there are 14 significant poses out of the entire 45, but this still suggests only roughly 240 samples per gesture pose. In other instances the datasets are simply unavailable. This is the case in [9], [17], [19], [22], [29], [30], [33], [51], [89], [56]. In the work of Hsieh and Liou [56], in addition to the dataset being unavailable, there is only one true static hand gesture recognized by the system, with the others being dynamic gestures. Dardas and Georganas present a good static gesture recognition work in [25]. However their dataset is too small at 1100 frames per gesture, and is publically unavailable.

Some large hand datasets can be seen in [58], however these are only suited for hand detection because static pose information is not recorded. The datasets of A. Mittal, A. Zisserman, and P. H. Torr contains 13050 labeled hand Instances. The work of P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman contains 6000 annotates frames with left and right location annotations. S. Bambach, S. Lee, D. Crandall, and C. Yu also use a large dataset of 15 thousand hand instances. Please refer to [58] for

bibliographical details on these datasets. These three datasets are publically available and are representative of the quality that is required for static hand gesture recognition.

For the Gesture Recognition work a large static hand gesture dataset was collected. The goal was a minimum of 5000 images per gesture class. The final dataset contains between 8 and 10 thousand images per class. It also contains around 44000 negative background images. This dataset facilitates training and testing for a large variety of gesture recognition methods and it would allow for in depth comparisons. Having a dataset be too small means that the system will not be robust in differing environments. It also means that methods which require large datasets cannot be compared to methods that require small datasets. In the Gesture Recognition system methods are compared that benefit from large datasets as well as methods that require small datasets for good performance. Random sampling is used to select smaller training sets for the support vector machine models. All that dataset samples are used for Adaboost training. MobileNets being a deep neural network, benefiting from class balances in the data, as well as large datasets, was trained with all the positive class samples of the gestures used by the system, and with random sampling of the negative samples. These negative sample training datasets were much larger than the ones used for SVM training. Thus a large dataset allows for flexibility in classifier model choices. Having methods that are only trained and tested on small datasets makes them of too limited practical use.

The reporting of results differs from method to method. Some methods report per class results [9]–[11], [19], [25], [30], [41], [56], [14], while others do not [7], [17], [22], [27], [33], [66], preferring to report average results. In the case of Lekova and Dimitrova [12] quantitative results are not reported for their dataset. Another issue with static gesture recognition methods is that they don't always report results against negative background images [9]–[11], [19], [25], [30], [41], [56]. It is an important measure of the system to know how well it doesn't detect gestures when none are present.

The inconsistency in metrics, differing hand gestures, and the small and/or publically unavailable datasets indicate that the problem of video only static hand gesture recognition has not been sufficiently solved. In order to facilitate comparisons between future works a large publicly available dataset with pose labels will have to be made in

order to give a common metric on which to evaluate current and future work. Without such a dataset it will be very difficult to understand where improvement is needed in the field of static hand gesture recognition.

For the gesture recognition system a large hand gesture dataset was collected and used to train and evaluate the system performance. The system was tested for true and false positives for each gesture class as well as for the negative backgrounds. This large dataset allows for direct comparison between small data methods like SVMs and large data methods like deep neural networks. As such the performance of the Gesture Recognition work is compared to a deep learning model proposed by Google for mobile vision applications called MobileNets rather than other static gesture recognition algorithms. The reason is that the performance of deep learning, and the universality of using such methods to solve a variety of problems including static hand gesture recognition, will entice designers to use them. Therefore by comparing the benefits and drawbacks of the Gesture Recognition work to MobileNets, the gesture recognition method can be evaluated to the toughest competition. The incentive for using this gesture recognition work can then be seen. Since many researchers are familiar with deep learning, a comparison will serve as a very good justification for using the method presented here if the method is shown to have advantages. The work will show that while MobileNets achieves state-of-the-art performance, it is not so for every gesture class, and despite being able to run in real-time MobileNets relies on optimizations to offset its computational load which could also be applied to other classification methods. This work will show how good performance on the static gesture recognition task can be achieved with simpler and less computationally demanding methods.

MobileNets and other neural networks are more computationally costly than SVMs and Adaboost classification methods. They rely on proposal generators to run in real-time. Proposal generators could be used to speed up Adaboost or SVM based classification methods. Particularly SVM methods would benefit from having fewer windows to evaluated from input frames. The benefit of Adaboost and SVM methods is that they are computationally lighter, more familiar in terms of design because they have been used for longer, and are thus simpler to implement. However their major drawback is the amount

of false positive detections that they produce. This work makes simpler machine learning methods discriminative enough for static hand gesture recognition against a complex background. By reducing the false positive rates, Adaboost and SVM classification remain viable alternatives to Deep Learning where computational load is reduced at the expensive of true positive detections. For gesture recognition systems having low false positive detections is more important because false detections deteriorate the use of a method for user interfaces. Having a low true positive rate is compensated for by having real-time performance because the system becomes more responsive the faster that it is able to execute.

There are a variety of methods proposed specifically for 2d video only hand tracking. Many of these often use a segmentation and contour based tracking strategy. Out of these only [33], [50], [52], [68] have as sophisticated a hand model as the hand tracking method presented in this thesis which is capable of estimating the palm and fingertips.

The way the testing of these methods [33], [50], [52], [68] is performed becomes very significant for allowing comparison to other methods. Chernyshov and Mestetskiy[68] use uniform background segmentation during their testing which is much too constrained compared to the adaptive segmentation used by the hand tracking presented which lets it track hands in complex backgrounds. Likewise Chochai et al [50] use a dataset that represent a situation different from the intended use of the hand tracking system. Their dataset contains hands in complex backgrounds but relies on black gloves for segmentation.

Yeo et al[33] doesn't use a hand tracking dataset which has annotated frames of a user's hand for testing. They test the performance of their system with a qualitative and quantitative user study. The qualitative portion consists of users evaluating their experience with the system as a UI. The quantitative study involves testing the ability of users to issue clicks in specific places. While evaluating tracking performance is also important, this kind of user performance and experience study is none the less valuable for determining a system's effectiveness as a user interface. This could be an excellent future direction of the hand tracking research. However the system presented by Yeo et al[33] would certainly benefit from having its tracking performance tested using a hand

tracking dataset like the hand tracking system presented in this thesis.

The hand model presented by Grzejszczak et al [52] is a sophisticated hand model that is well tested with hand tracking datasets. In addition, unlike the dataset used to test the hand tracking system presented here, the authors of [52] have datasets that mark individual landmarks such as fingertips. This allows them to evaluate their method on the ability to find the correct locations of individual landmarks such as fingertips, as opposed to the more general hand location testing done with the hand tracking system presented in this thesis. This is an excellent direction for further research in hand tracking. However the work presented in [52] does show that good datasets for 2D video only hand tracking testing are lacking. Specifically for sequential testing of hand tracking. While there are several datasets available for still hands, annotated datasets for sequences of hands for 2D video applications are few. The dataset used to test the hand tracking system presented in this thesis is approximately 20 000 sequential annotated hand tracking frames which indicate general hand location and the amount of extended visible fingertips. There are only 2 sequential 2D datasets mentioned in [52] being roughly 900 and 5000 frames respectively. The hand tracking dataset using in this thesis would be a welcome addition in order to test hand tracking performance, and landmark annotation can also be added to further improve its use for comparative testing. As explained by the authors only some 2D hand tracking datasets contain landmarks and out of the 2 sequential 2D datasets mentioned in their work only the roughly 9000 frame dataset contained landmarks. Therefore the hand tracking dataset used in this thesis is a significant contribution to 2D hand tracking with or without landmark annotation. It must also be mentioned, that while well annotated, the datasets used for the landmark testing of [52] are small, in the range of 600 - 900 frames. The method of Grzejszczak et al [52] does have one significant shortcoming in that it runs at 5 frames per second which is significantly slower than our method, which also hasn't been formally measured, can certainly run around real-time speeds for 30 frames per second as demonstrated in the demonstration videos of the hand tracking system [185], [186] and of its user interface applications [187][188][189].

No other 2d hand tracking method combines guiding template tracking for fingertips with contour tracking to achieve real-time hand tracking. This novel contribution overcomes

the jitteriness of location of contour tracking methods with the more stable template tracking. It also overcomes the computational load of template tracking by reducing the size of the search regions with contour tracking and allowing for real-time operation. These contributions enables real-time UI applications with position based commands being mapped to fingertips.

Several of the works considered during the literature review were only considered after the formulation of the research direction and methods of this thesis. While these methods had no bearing on the direction or the results it is none the less interesting to see how the art has changed in the most recent publications.

Liu et al. [153] achieve good static hand gesture recognition with their CNN based method. However the method requires depth data for training and cannot be used in a strictly 2D image data context. Their dataset seems to only contain hand images against uniform backgrounds. In addition their dataset is not publically available. Adithya and Rajesh[92] use a CNN for static hand gesture recognition in a 2D image context. The datasets used are publically available, however they are small containing 200 images per class for the NUS dataset, and approximately 470 images per class for the American Finger Spelling dataset. Hu et al. [87] use an interesting combination of classifiers to detect static hand gestures, correct hand gestures to make sense temporally, and to identify sequences of static hand gestures as dynamic hand gestures. They use a regression based classify for static gesture recognition, and a Bayesian model, and a HMM model, for the temporal smoothing, and dynamic gesture recognition, respectively. Their static hand gesture dataset however only contains 200 samples per class. Islam et al.[88] use a CNN for static hand gesture recognition. They also use dataset augmentation in order to enrich their dataset for training which obtains improved results. Their base dataset however is only 960 images per class, which while better than other datasets is still on the small side. The authors also use a second dataset which has a much better 2000 samples per class, however these are near infrared images and not 2D colour video images and are thus less suitable for video only contexts. This dataset is still significantly smaller than the 8000-10000 images per class dataset that is presented in chapter 3 of this thesis. Furthermore if the authors are able to get improvements in performance by

augmenting their datasets with image transformations, imagine what could be achieved with the significantly bigger dataset presented in this thesis. Another issue with their method is that the authors use data with uniform backgrounds and rely on background subtraction. This of course can be corrected with better pre processing but suggests that easy to use and reliable 2D hand gesture pre processing methods would still be appreciated by the research community.

The work presented in [105] is an interesting combination of well known and novel classification methods. An SVM classifier is used with features extracted from an AlexNet CNN to classify static hand gestures. Their method makes no mention of image extraction however. It assumes that a hand image is already available for classification. Their dataset while publically available is also small at approximately 70 images per gesture class. What is interesting is that the method beats CNN performance while using SVMs which shows that SVMs are still viable for state of the art static hand gesture recognition.

Uwineza et al. [77] present a method based on an simple type of neural network called an Extreme Learning Machine for static hand gesture recognition. The authors train their model on the 200 samples per class NUS dataset which they augment to 1500 samples per class. While the authors use a neural network model they take time to craft their own features using a fusion of Harlick Textures, Hu moments, and Colour histograms. Their method is capable of working in complex backgrounds, however they use a GrabCut segmentation method which assumes that there is always a hand in the foreground. The method would need to be evaluated against negative backgrounds to make sure that detections do not occur when no gestures are present. Issues with their GrabCut segmentation method assuming a user's hand is always present could be remedied with a good hand detector. Furthermore this is another case of a small dataset. While the augmented sample count is certainly better than other methods, the core samples are still only 200 per class.

It is interesting that new methods continue to be published in static hand gesture recognition which still have the issues of having small datasets. Small datasets leads to methods being more context sensitive and lacking generality and it makes comparison

work difficult with methods that require larger datasets. Furthermore assumptions such as uniform backgrounds and hand presence allow authors to focus on the classification aspects of the problem. This suggests that the detection aspect is still an open area of research and new and simple solutions would be welcome. Additionally the focus on using CNNs to solve gesture recognition problems means that many newer works are GPU dependent, which means there are still contributions to be made with solutions of lower computational load. Seeing SVM classifiers and ELM classifiers, which were cited from a 2004 paper in [77], being used in the most recent literature demonstrates that there is still significant room for innovation with well known methods.

With regard to hand tracking, [149]–[153], [190] represent recent contributions, which while they did not influence the direction of this thesis research, are informative on progress in the field. Park et al[153] have shown a novel method capable of correcting for motion blur in depth image based hand tracking by using a gyroscope. Their multiple strategy approach echoes the idea that using multiple strategies provide better results. This allows their system to accommodate fast hand motion. This method is however a 3D method which is not directly applicable to the 2D hand tracking approach presented in Chapter 5.

Mueller et al[149] present a novel hand tracking approach for 2D video. It fits a 3D hand model onto an RGB image. The authors overcome dataset limitations by augmenting their 3D hand dataset with a Generated Adversarial Network. This augmented dataset is then used to train a RegNet CNN which produces hand joint heatmaps at run time from an RGB input image of a hand. The heatmap maximum likelihood locations are used to fit the 3D hand model with an energy minization approach. While this method is designed for 2D applications it is not a 2D method because it requires 3D data to train. Any improvement upon this work would likely involve more 3D training data so that the algorithm becomes more robust and accurate in a larger variety of environments and for a larger variety of users. As such it cannot be considered a purely 2D method and 3D methods are beyond the scope of this thesis work. However this 3D hand model fitting to 2D RGB images is a very promising and interesting direction for future work.

Liu et al [150] uses a Kernelized Correlation Filter for their 2D hand tracking method. It presents some interesting ways to solve the occlusion of the hand with other objects by using a Kalman filter and adaptive model updating criteria. However it doesn't describe their dataset or provide a reference for it. Cripanati et al.[151] use a Lucas-Kanade optical flow approach for 2D hand tracking in addition to YCrCb colourspace skin segmentation. The authors do not test on a dataset but rather use a series of user test cases to evaluate their work. The authors have also studied the impact of background using a wide variety of colours and they have found that red background colours can cause issues which echoes observations made about the hand tracking algorithm presented in chapter 5. Sun et al[152] use a simple template matching method with a hand template for their hand tracking method which operates on depth images. The authors achieve simple hand tracking without using machine learning. This is a relatively simple method that uses depth which is not considered for the systems presented in this thesis.

Lastly the work of Kapidis et al.[190] is a very inspired method that shows the applications of hand tracking for human action recognition in 2D video. They use a large publically available ego-centric dataset of human hand actions in a kitchen called Epic-Kitchens. The authors also use additional ego-centric hand datasets to train their hand detector. The authors use hand detection using YOLOv3 and the SORT algorithm for hand tracking. SORT is a combination of a Kalman filter with Hungarian algorithm. The authors achieve results similar to CNN approaches that process the entire videos of actions using just hand tracks and object detection. This is a promising direction for future research because it shows that by using hand tracking information a fraction of the input is needed for model training compared to using full videos.

The recent work in hand tracking shows that simple tracking methods continue to find favor even alongside more elaborate 3D hand model fitting methods. There is interest in using hand tracking to simplify input training of models and also to improve more well known techniques to achieve hand tracking without machine learning. There is also interest in improving the robustness of 3D hand model fitting to allow users to move their hands faster, and also to use it in a simpler 2D video context. This suggests a general desire for practical solutions that can work in more limited contexts which consequently

makes them more broadly applicable. There is also an evident need for datasets to allow for comparative work in 2D hand tracking. While, as demonstrated in [190], there are plenty of ego-centric hand datasets that are publically available, there doesn't seem to be publically available datasets of front facing users in 2D video which are used as benchmarks between different works. Without such benchmarks and specific evaluation criteria, it is very difficult to determine the performance of different methods. This means that a dataset such as the one presented in chapter 5 to evaluate hand tracking will be appreciated by the research community. Furthermore there are no 2D hand tracking methods that have been presented in the literature that have attempted to map aiming and moving functions of a video game requiring rapid response to a tracked hand target. While system usability is quite difficult to evaluate quantitatively, the usability of the 2D hand tracking system presented in chapter 5 is demonstrated by applications which give a higher degree of control to the user than is generally seen in the literature. The control applications presented in chapter 5 will be of interest to the research community.

2.10 The proposed Hand Gesture Recognition and Hand Tracking Systems

Three systems are presented in the following chapters of this thesis.

2.10.1 Hand Gesture Recognition

The first is a Hand Gesture Recognition system. It performs real-time static hand gesture detection on a video feed using one of 3 configuration options for multi scale detection: HOG Features with Adaboost, HOG features with SVMs, and a CNN called MobileNets. While the CNN MobileNets configuration is taken as a complete system, the first two configurations are complemented by validation functions. These functions use colour information, 2D colour histograms to adaptively sample skin pixels and perform segmentation, and geometric shape features to validate the shapes of the detected hand gestures. The validation functions reduce the false positive rate and increase the usability of the gesture recognition system. A gesture is considered recognized if it has been successfully detected and validated. Chapter 3 of the thesis explains in detail the machine

learning methods used to produce hand gesture detections. Chapter 4 explains the details of the complete hand gesture recognition system which uses the results of the machine learning detection presented in Chapter 3 and builds on it by adding validation functions to improve performance.

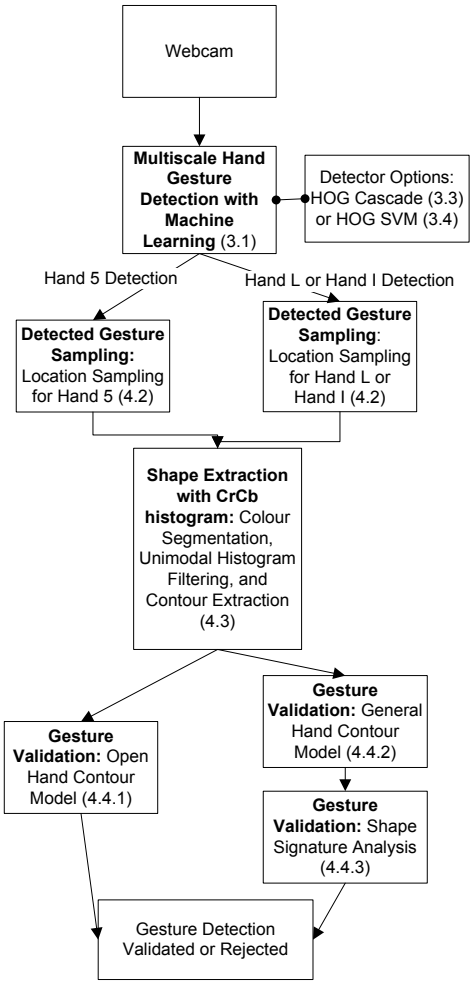


Figure 4 Gesture Recognition System Flow Chart

2.10.2 Hand Gesture Recognition with Hand Tracking

A Hand Gesture Recognition with Hand Tracking system is also presented which builds on the first system. It uses the Hand Gesture Recognition system for gesture recognition,

and then uses the colour histogram and the geometric shape features information obtained to track the user's hand once it is recognized. For tracking, the system uses the 2D colour histogram to segment and track the user's hand, using a nearest valid contour strategy subject to a distance constraint.

The tracking also uses the same geometric shape features that were used in the validation functions of the Gesture Recognition system to process the tracked hand contour and extract palm and fingertip location information. The geometric shape information, while robust and rapid to compute, tends to suffer from noise, so it is used to guide a complementary nearest neighbour template tracking approach to track the user's fingertips. This fingertip template tracking strategy provides more stable location information of the user's fingertips, which is essential for constructing user interfaces.

The hand tracking also uses HOG features with Adaboost to recognize the pose of the tracked hand. It uses a one vs. all classification strategy to recognize when a tracked hand forms specific static gesture poses.

2.10.3 Improved Hand Gesture Recognition with Colour Clustering Segmentation

The third presented system builds upon the Hand Gesture Recognition system and improves the quality of the colour processing with Colour Clustering Segmentation. It uses morphological processing of the 2D colour histograms used by the validation functions, and of the subsequent segmentations to improve the quality of the hand gesture segmentation.

The 2D colour histograms are decomposed into their constituent colour modes or clusters. Using this modal decomposition of the colour information, better hand segmentation can be achieved because hands are unimodal in colour. This improved segmentation quality results in sharper and clearer hand contours which means more correct hand gesture detections more frequently.

Colour clustering may in some cases produce more than 1 valid contour for a single hand gesture detection. This is solved by using HOG features with SVMs to select the best hand gesture contour.

Chapter 3 Hand Gesture Detection using Machine Learning

3.1 Overview

This chapter will demonstrate how to perform hand gesture detection in multiple scales on input images from a web camera with machine learning models. The models will be tested for their ability to detect gestures as well as ignoring objects that are not gestures.

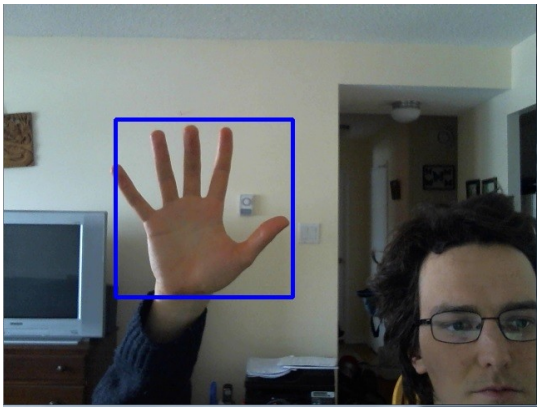


Figure 5 An example output from the machine learning detection step. The bounding box indicates the rough estimate of the hand.

The idea behind the Hand Gesture Recognition system is to use a two step process. The first step is a machine learning detector to find rough estimates of users' hands making specific hand gestures. This allows hand gestures to be detected anywhere in the frame, because the detectors chosen for this system are capable of multi scale detections. These rough estimate detections are processed and filtered by the second validation step. The validation step runs on the detections to confirm true positives and to discard false positives. Thus it is expected that the detectors will have some inaccuracies and false detections.

For this system 2 types of multi scale detectors were considered: Adaboost cascades with HOG features, otherwise known as HOG Cascades, and Support Vector Machines with HOG features, or more simply HOG SVMs.

The HOG feature was chosen because it is invariant to light changes which are a very big challenge with Visual Gesture Recognition systems.

The Gesture Recognition System, in both HOG Cascade detector configuration and HOG SVM detector configuration will have its performance compared to a MobileNets model trained with the same dataset. This will show how this Gesture Recognition system performs compared to recent advances in Convoluted Neural Networks (CNNs).

3.2 Gestures used

There are 3 hand gestures used for the hand gesture recognition system. These gestures are: an open hand with 5 extended fingers; also referred to as the hand-5 gesture, a hand with thumb and index fingers extended; referred to as a hand-L gesture, and a hand with only the index finger extended, also called the hand-I gesture.

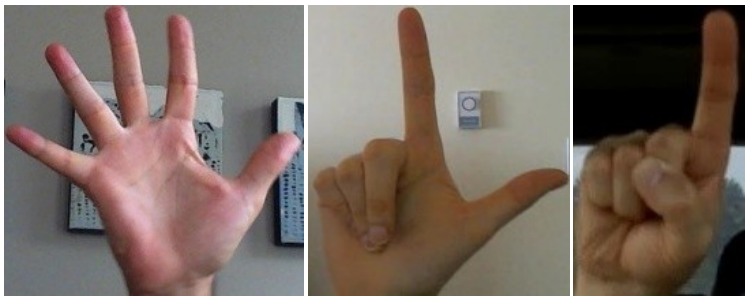


Figure 6 Images of the 3 gestures used by the hand gesture recognition system

All gesture samples that were used for the training of this system had to be front facing, and thus the system is only designed for front facing hand gestures. The hand-5 gesture was chosen because it is the most distinct hand gesture with 5 extended fingers. The large amount of surface area devoted to the palm means that good segmentations are easier to achieve because the palm best represents the colour range of the user's hand. Additionally the hand-5 gesture has some unique shape properties. The angles between consecutive fingers tend to be small and this property was used to create a validation function that is unique from the other 2 gestures. The small angle property is used to make a separate open hand contour model which is used to validate the gesture, whereas the general hand contour model lacks this property and is used to validate the other two gestures. The hand-L and hand-I gestures were chosen because of their distinctness from the hand-5 gesture and because they have distinct finger configurations that are different from each other. They are also two gestures that are easy for users to make. Arguably a closed fist gesture is also easy for users to make however this gesture was not used because its round

structure, due to a lack of extended fingers, makes it too similar to non hand objects. These gestures sometimes appear in other works in 2D hand gesture recognition, however that was not the primary motivation for why there were chosen. This is due to a lack of large publically available datasets for 2D static hand gesture recognition, as described in section Gap Analysis, which makes comparative work difficult. The gestures were primary chosen for their distinctness from each other, distinctness from the background, and how easy the gestures were to make for the user.

3.3 GR system with HOG Cascade Detectors

3.3.1 Method

Adaboost Cascades are a multiscale detector that uses a series of weak classifiers to make a strong classifier. Each weak classifier is trained to partially classify all samples generated from a sliding window. Each weak classifier uses several image filter masks or features to either accept or reject these generated samples. In the case of this system HOG features were used.

HOG cascades were used as a multi scale detector for one of the Gesture Recognition system configurations. They were simple to train and offered robust performance. HOG Cascades are trained by providing positive samples and negative backgrounds. In this system the positive samples were hand images of a given hand gesture.

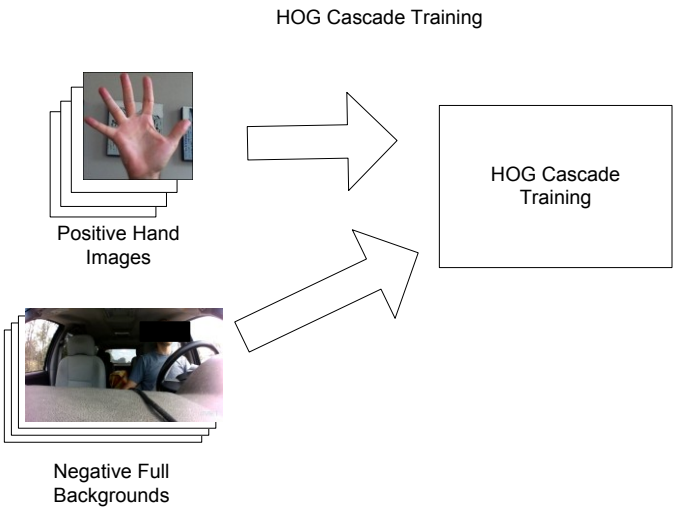


Figure 7 HOG Cascade Training

For each of the 3 hand gestures a separate classifier is trained and used to perform detections.

During system operation input images are taken from the video feed. For each input image all three cascades are run in sequence. These cascades produce bounding box detections when their specific hand gesture is detected. These detection boxes are then evaluated by the gesture validation steps.

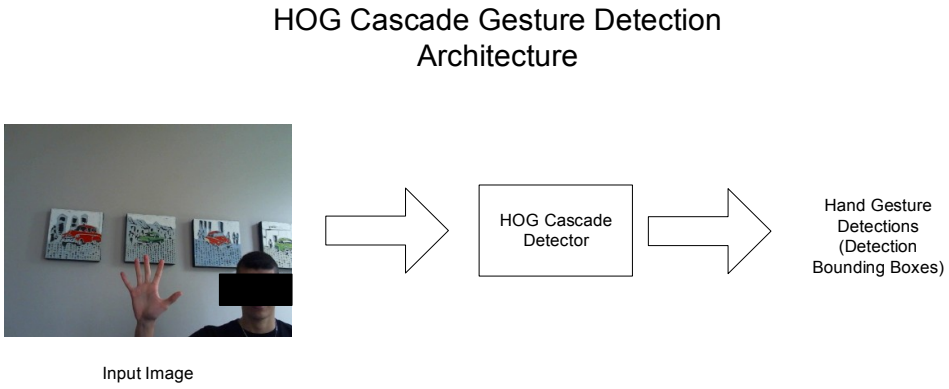


Figure 8 HOG Cascade Gesture Detection. This is performed for each hand gesture class.

3.3.2 Experiments

Three cascades were trained. One for the hand-5, the hand-L, and the hand-I gestures. For the hand-5 gesture 7813 positive samples were used vs. 35483 negative background samples. For the hand-L gesture 6401 positive samples were used vs. the same 35483 negative background samples. For the hand-I gesture 6958 positive samples were used vs. the same 35483 negative background samples. The dataset can be found here [54].

The cascades were trained with the following standard parameters for the samples and the stage training:

Table 1 Cascade Detector Training Parameters

Training Parameter	Parameter Value
height	32
width	32
minHitRate	0.990
numStages	20
maxFalseAlarmRate	0.5

The height and width define the scale at which the samples are resized in the Adaboost training algorithm. This also becomes the minimum detection size. The Adaboost algorithm trains a series of stages where each stage is a weak classifier. By linking the weak classifiers one after the other a strong classifier can be trained. This strategy is otherwise called a Cascade of classifiers and it's from here that the name Cascade comes. The minHitRate parameter defines the minimum true positive rate that each stage must have, and the maxFalseAlarmRate defines the maximum false positive rate that each stage can have. The overall true positive rate can be determined by:

$$Overall\ true\ positive\ rate = (minHitRate)^{numStages}$$

and the overall false positive rate can be found with:

$$Overall\ false\ positive\ rate\ = (maxFalseAlarmRate)^{numStages}$$

The numStages parameter determines the number of stages that the classifier attempts to train. It can be seen that by setting the minHitRate to .99 and the maxFalseAlarmRate to .5 results at the output of stage 20 in .81 or 81% true positive rate with a corresponding 0.00000095 or 0.000095% false positive rate. In practice training is stopped when the training of a single stage becomes very slow and the overall false positive rate with the currently trained stages is approximately 0.00005 or lower. This result can be achieved between 12 and 14 stages, depending on the false positive rates achieved by the individual stages. Training beyond 0.00005 can result in over trained classifiers that do not perform well in real-time online operation.

The final hand-5 cascade detector was 12 stages, the hand-I detect was 14 stages, and the hand-L detector was 14 stages.

3.3.3 Discussion

Training Adaboost classifiers comes with the advantage that excellent results can be achieved with little tweaking of parameters. The gestures themselves seem to indicate that more uniquely structured poses such as the hand-5 pose require less classification computation and thus less stages than the less distinct hand-L and hand-I poses. The results of the HOG Cascade Gesture Recognition System can be found later in this chapter where the detectors' ability to recognize gestures is evaluated as well as the complementary effects of using the detectors with the validation functions.

3.4 GR system with HOG SVM Detectors

3.4.1 Method

HOG SVMs are another type of multi scale detector used to make a second configuration of the Gesture Recognition system. Support Vector Machines are a machine learning classification algorithm proposed by Vapnik and Cortes [100]. SVMs classify feature vectors. Feature vectors are created by taking feature transformations of training images of the classes used by the classification problem. SVMs are trained by finding the optimal separation plan on a set of feature vectors. These vectors are generated from a set of image patches of the classes obtained from the training dataset. The optimal separation plan is defined by the a minimal set of vectors known as the Support Vectors. SVMs are used in the Gesture Recognition system by using a sliding window to generate input vectors and then classifying these vectors with the SVMs. In this way multi scale detection of the gesture classes is achieved.

The Histogram of Oriented Gradients, or HOG feature, was chosen because its invariant to light transformations. It is also scale invariant when an image pyramid is used, which is standard in sliding window based approaches. HOG is a mathematical feature transform that can take an input image and produce a vector that can then be used by a Support Vector Machine. The HOG features are calculated for each image patch from the

training dataset and this produces the feature vectors that train the SVMs. During operation of the Gesture Recognition system the HOG features are calculated on the image patches produced by the sliding window run on the input image. These features are then classified by SVMs to achieve detections for each class of hand gesture.

In the case of this gesture recognition system 2 class SVMs were used. Hand gesture samples of a given gesture formed the positive class, and background samples formed the negative class. A positive to negative ratio of samples must be selected so that an SVM performs well in real-time online testing. Training is done by feeding in the samples to the SVM training process, and by randomly selecting the appropriate amount to give the desired positive to negative ratio. Unlike the Adaboost cascades SVMs must be trained on image patches of negative backgrounds for the negative class as opposed to full background images. Additionally, because the intent is to make a real-time system, linear kernels were used for the SVM training because of computational simplicity. A significant part of the false detections were expected to be handled by the validation functions.

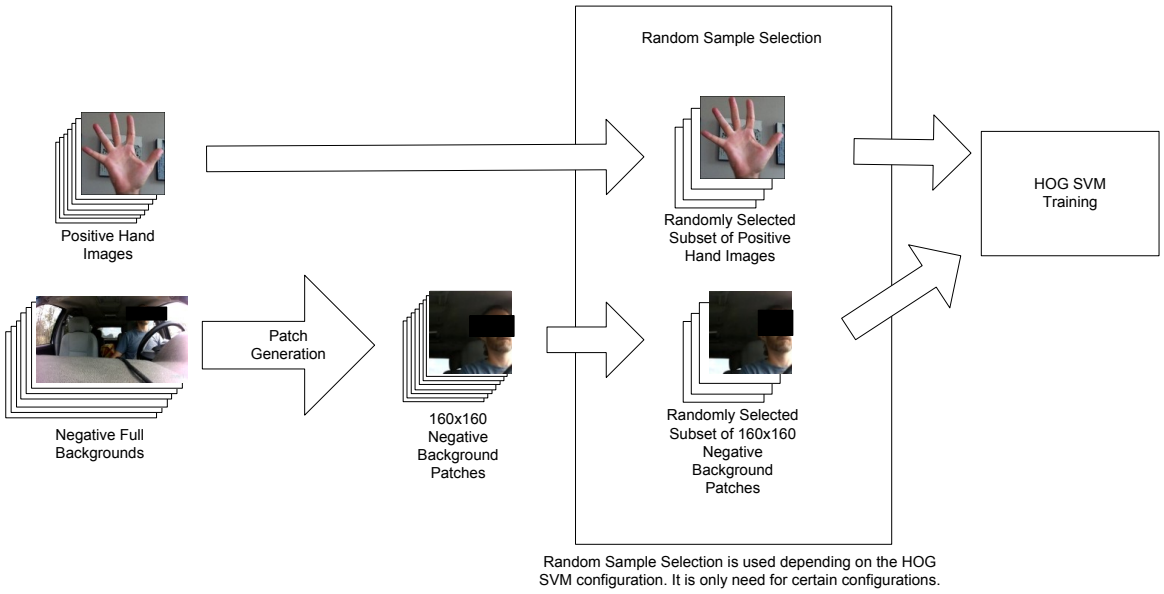


Figure 9 HOG SVM Training

In order to make more robust and discriminative detectors a multi-stage architecture can be used. Multiple SVM stages can be trained from different partitions in a partitioned

dataset. In the case of this system the partitioning was done using the magnitude of the HOG gradient.

3.4.2 System Operation

Four multi-stage configurations were investigated. One configuration is the 2-stage architecture with a 1-stage HOG SVM system, because it was found that a 1-stage SVM system benefits from a second pass of the same SVM. The other 3 configurations are a 1-stage system, a 2-stage system, and a 3-stage system.

The SVM detectors are run in sequence on the input image from the video feed. Each gesture's support vector machines produce detection bounding boxes when that hand gesture is detected, which as before are then evaluated by the gesture validation steps. What is specific to the SVM configuration is the way the multi-stage detector architecture works. Whereas the cascade based system used the cascades as a black box, the SVM based system uses multiple SVMs for each gesture in sequence to refine the detections.

The way the multi-stage SVM detection specifically works is illustrated in Figure 10. The first SVM is run on the input image. Then if there are any detections of the specific gesture, the detection bounding boxes are taken, copied, and enlarged by 15 pixels in each dimension, so long as the enlarged bounding box still fits on the original frame. These enlarged bounding boxes are run through the second stage SVM, and any bounding box that does not yield detections after the second stage is discarded. The remaining enlarged bounding boxes are then run through the 3rd stage SVM if it's a 3-stage system, and once again only kept if they produce detections in that stage. Then original detection bounding boxes from the 1st stage corresponding to the remaining enlarged bounding boxes are declared to be the refined detections. These refined detection bounding boxes are then passed to the validation steps. This SVM detection procedure is repeated in sequence for each of the 3 gestures.

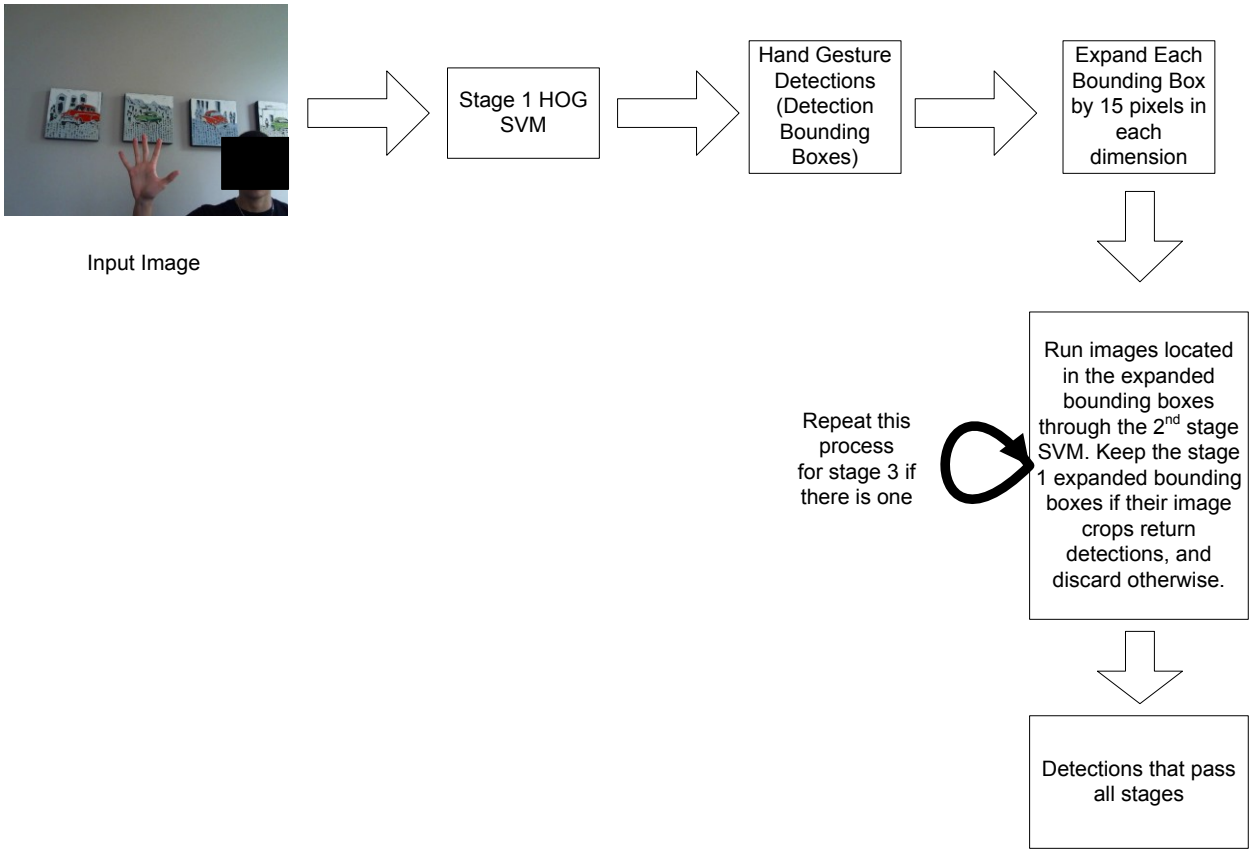


Figure 10 HOG SVM Detection Architecture. This is performed for each class of gesture.

3.4.3 Rationale for using multi-stage SVMs

The idea behind using multi-stage SVMs is that better partitioning of the vector space created by the features can be achieved with multiple linear SVMs as opposed to just one.

The dimensionality of the problem, and thus the SVM model, can be reduced according to this article [29]. This means that if you separate the negative samples according to gradient criteria you can then make multi-stage SVMs. Each stage of the multi-stage SVM can be trained using this configuration:

Positives vs. Negatives from a given gradient category

This increases performance because reduced dimensionality means a reduced number of support vectors which reduces computational load resulting in faster performance. Also multiple decision boundaries partitioning a vector space should increase accuracy.

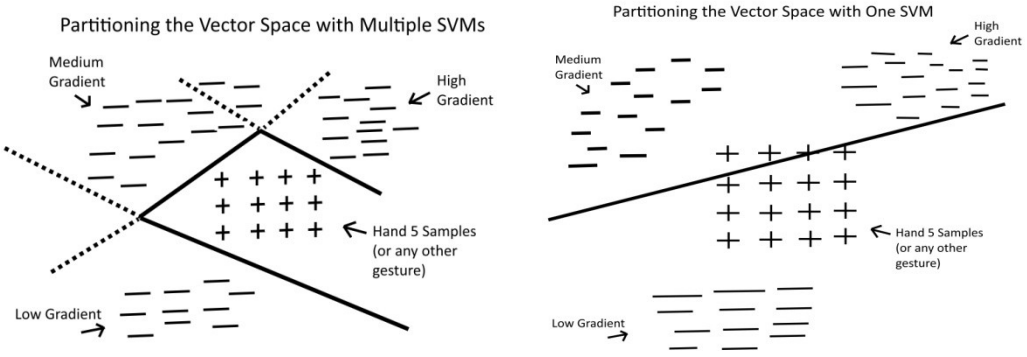


Figure 11 Vector Space Partitioning with Linear SVMs

This figure shows the effect of using multiple decision boundaries. Even simple linear decision boundaries can be combined to make better decision surfaces in the vector space.

Experimental results show that a 1-stage SVM produces better results on a second pass in a 2-stage system. Even a 1-stage SVM seems to benefit by using multi-stage architecture to self filter its detections with a second pass. That is why the 2-stage architecture is used even for the 1-stage configuration.

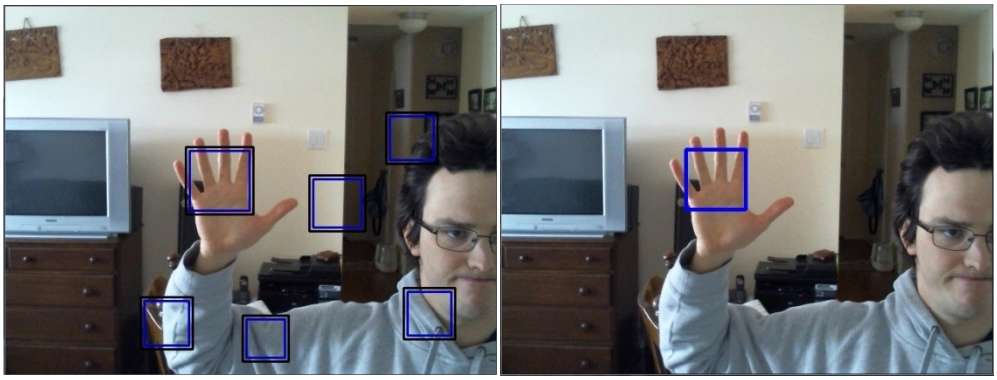


Figure 12 The left image shows the detections after 1 pass with a 1-stage HOG SVM. The right image shows the detections that remain after a 2nd pass with the same 1-stage HOG SVM.

3.4.4 How to partition and train multi-stage SVMs

The negative patches are partitioned according to the average magnitude of the HOG of all negative patches. In order to make a multi-stage system the patches can be partitioned once to make the 2-stage system negative patches resulting in low gradient patches and high gradient patches. The high gradient set using the dataset collected for this work is bigger, and it can be partitioned again to make 3 sets of negative patches for the 3-stage system.

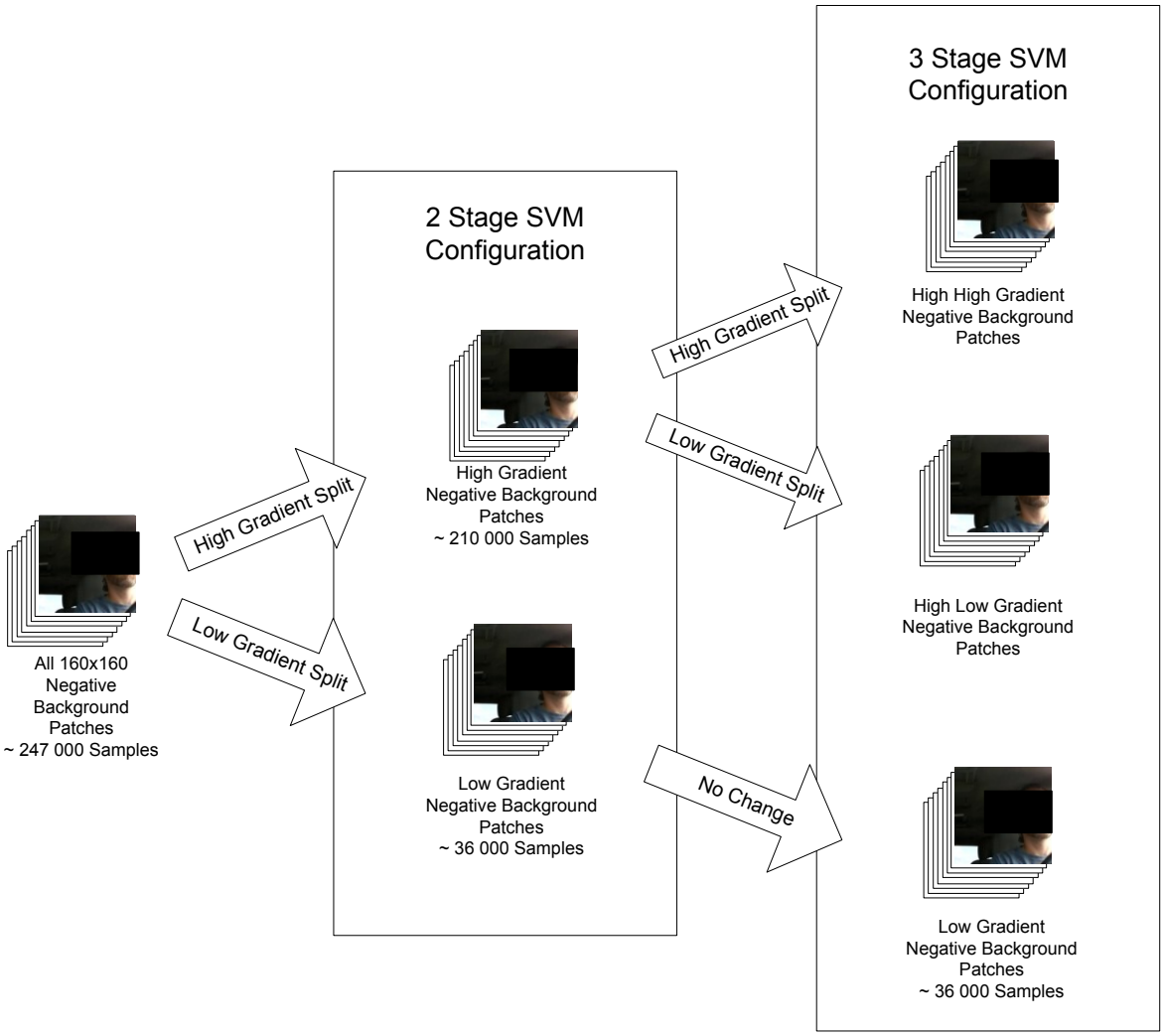


Figure 13 Negative Patch Partitioning for Multi-stage HOG SVM training

In order to do the training, the negative patches have to be partitioned according to the average magnitude of the HOG gradient, then the samples have to be randomly selected for the SVM training, and then each SVM has to be trained with its randomly selected set of samples.

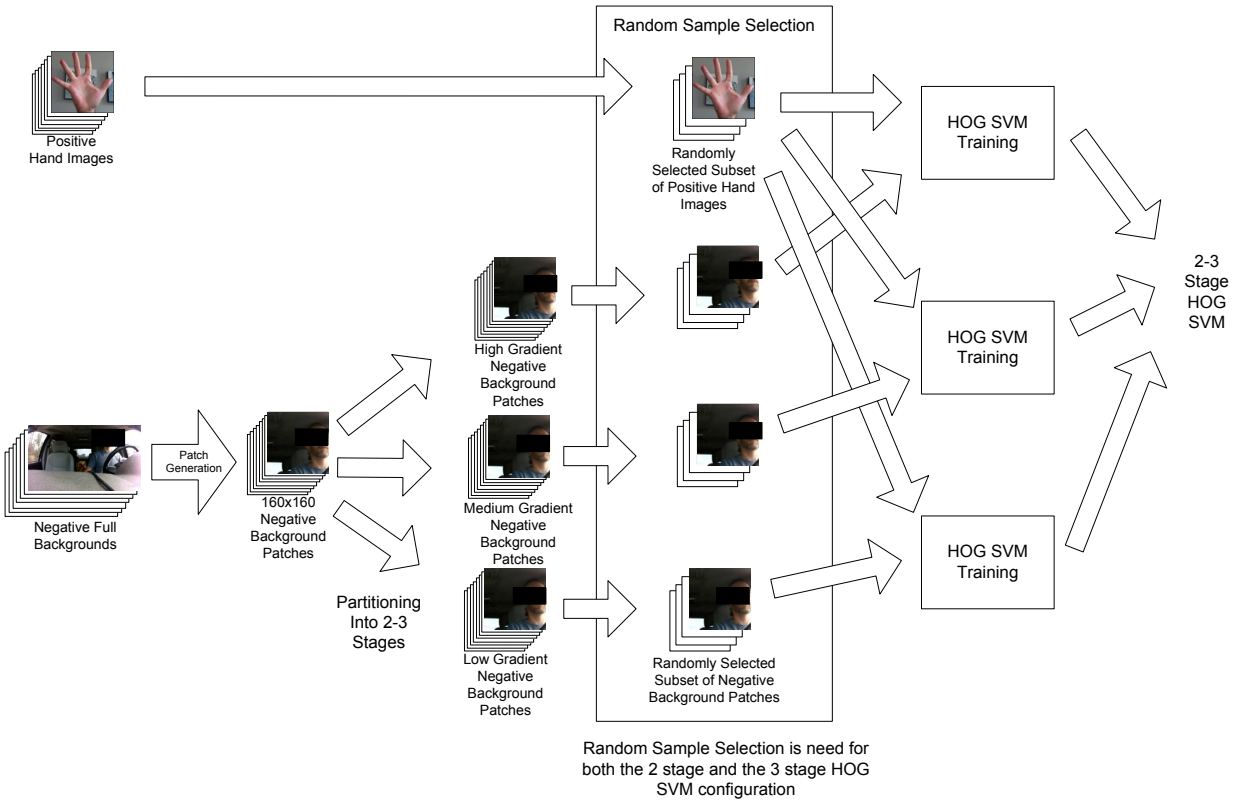


Figure 14 HOG SVM Training for Multiple Stages

3.4.5 Experiment

The Support Vector Machines used linear kernels which were chosen for simplicity and speed. The SVMs were trained with the parameters described in

Table 2.

Table 2 SVM Detector Training Parameters

Training Parameter	Parameter Value
window size	64x64
block size	8x8
block stride	4x4
cell size	4x4
bins	9

The window size defines the size that the sample images will be resized to in order to compute the HOG feature. The Histogram of Oriented Gradients, or HOG feature, was chosen because it is invariant to light transformations. It is also invariant to scale when an image pyramid is used such as is the case in multi scale windowing. The HOG feature takes the resized samples divides the whole area, 64 by 64 pixels in this case, into blocks. The block size is 8x8 and it defines the size of these blocks. The block stride defines the vertical and horizontal step size between adjacent blocks. The block stride is 4x4 which corresponds to a 50% overlap with adjacent blocks. This is typical of HOG feature calculation. Each block consists of cells which are determined by the cell size parameter. These cells have their gradient orientations computed and the orientations are stored in a histogram which has its bins defined by the bin size parameter. This bin size determine how the 180% orientation angle range is divided. A bin size of 9 results in angle ranges of 20 degrees per bin, which the standard choice for HOG computation. The histograms of all the cells of each of the blocks are concatenated to form the final feature vector that is used for the SVM training.

Training was done with positive hand gesture samples vs. negative patches. The whole negative frames were not used for training like with the cascades. The negative background frames were broken down into 160x160 non overlapping patches.

Three configurations were tested using online testing to determine sampling ratio:

1. 7813 hand-5 samples vs. 247 148 background patches (all positive:all negative ratio)

2. 7813 hand-5 samples vs. 21883 background patches (1.5:4 ratio) **Best performance**

3. 7813 hand-5 samples vs. 11210 background patches (1.5:2 ratio)

Two configurations of the 1.5:4 ratio were compared.

1. 7813 hand-5 samples vs. 21883 background patches (1.5:4 ratio all positive samples)

2. 1452 hand-5 samples vs. 4013 background patches (1.5:4 ratio small sample configuration) **Best performance**

The best configuration small sample configuration of approximately 1500 postives vs 4000 negatives was used to train all further SVMs. Four multi-stage configurations were investigated which used this configuration.

A performance comparison was done of 4 Multi-stage SVM configurations. The first configuration is a 1-stage SVM. The second configuration is a 2-stage configuration with a 1-stage SVM run twice. The third configuration is a 2-stage SVM, and the fourth configuration is a 3-stage SVM. These configurations have been run with their validation functions and only for the hand-5 gesture. The 1-stage SVM was also tested in a 1-stage configuration for comparison. Details on how the multi-stage SVMs were trained can be found in the discussion section. The results are presented in Table 3.

Table 3 Multi-stage SVM Results

Multi-stage Configuration Number	Number of Stages	True Positive Rate %	False Positive Rate %
1	1-stage System	31.47	0.1456
2	1-stage run 2x	29.05	0.1294
3	2-stage	26.33	0.07281
4	3-stage	17.93	0.08090

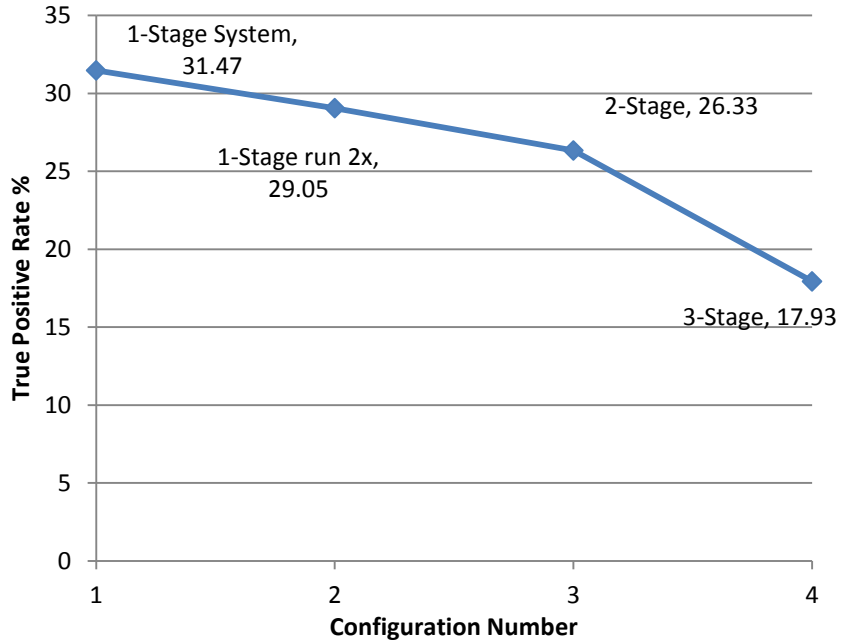


Figure 15 Multi-stage SVM True Positive Graph

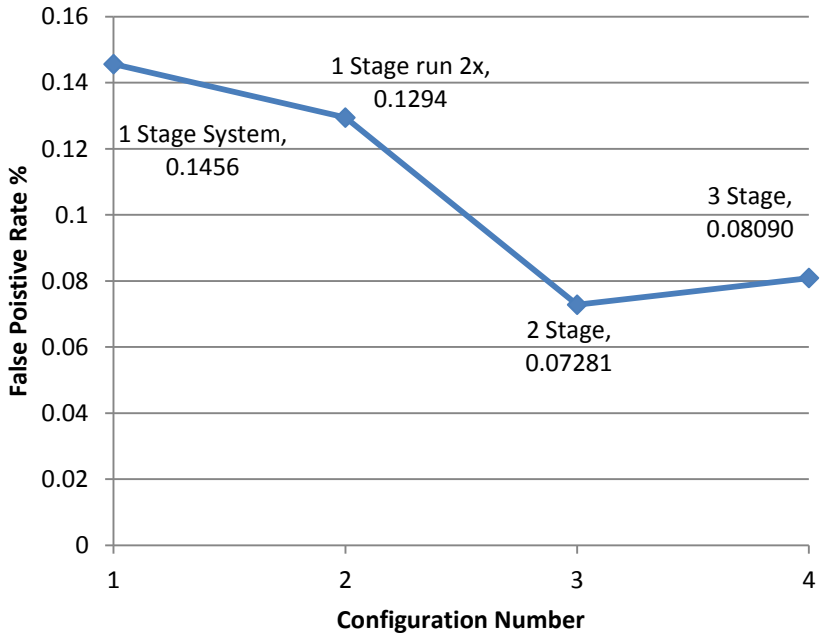


Figure 16 Multi-stage SVM False Positive Graph

As it can be seen there is a trade off. The false positive rate decreases as you add a second stage, and so does the true positive rate. It was decided that the 2-stage configuration

would be used because it had the lowest false positive rate, and it used the most samples from the dataset while still retaining high true positive performance. By using more stages the detector becomes more discriminatory and thus produces fewer false positives, even if this comes at a slight drop in true positive performance. By using more samples that the 1 SVM configurations, the 2 SVM detector makes better use of the training dataset to solve the detection problem. This is not only reflected in the lower false positive rate, but it also results in better localized bounding boxes which crop the visible hand gestures more closely. These better localized bounding boxes was the reason why the 2-stage SVM detector was chosen over the 1-stage SVM run twice detector, even though they had very similar true positive performance. The results of the 3-stage configuration suggest that using more than 2 stages accrues a significant drop in performance. The 2-stage SVM configuration was selected as the best overall configuration, and this configuration was used to build the 3 gesture SVM Gesture Recognition system.

3.4.6 Discussion

3 configurations were tried for one stage SVM multi scale detectors, in order to determine the best sample ratio. The first configuration was all positive training hand-5 samples (7813) vs. all negative training background patches (247 148). The second configuration was a 1.5:4 sample ratio with all 7813 positive training hand-5 samples and 21883 randomly selected negative training background patches. This ratio was chosen because the first small sample dataset produced good results with this ratio. The third configuration was a 1.5:2 sample ratio with all 7813 positive training hand-5 samples and 11210 randomly selected negative training background patches.

The all patches SVM configuration performed poorly. Probably because the classes were highly imbalanced and the model did not generalize well. 1.5:4 was the ratio that works visually qualitatively the best. 1.5:2 also worked poorly.

The best ratio all positive sample configuration (1.5:4) was investigated with a small sample set and the same ratio. Using a small randomly selected sample set of approximately 1500 positives to approximately 4000 negatives worked better than the

full 7813 positive to 21883 negative samples. During the initial dataset collection a small dataset collected from the data collection produced good results with approximately this number of patches. The 1500 positive vs. 4000 negative configuration was the first well performing configuration produced for hand-5 detection with the small initial dataset. That is why it was taken as one of the configurations to test with the full dataset using random sample selection of the complete dataset. The configuration produced well performing SVMs using the complete dataset. So that is why approximately 1500 positives to 4000 negatives using random selection is the configuration used for the SVMs.

Using multiple SVM stages is a promising option to increase the performance of the system. The idea is to use for each hand gesture several SVM detectors, one after the other, to refine the detections. 3 more configurations in addition to the 1-stage configuration were investigated to determine the best amount of stages to use. Each stage is trained with the same amount of randomly selected samples that was determined to work best in the 1-stage configuration tests. The first configuration is a 2-stage configuration with a 1-stage SVM run twice. The second configuration is a 2-stage SVM, and the 3rd configuration is a 3-stage SVM.

Training is done on the same randomly selected hand-5 samples vs. different sets of randomly selected negative patches for each stage from the corresponding gradient patches set. Initially the patch selection was done in the manner described in the preceding method section. It was thought that it would produce well performing SVMs. Specifically, the 2-stage configuration was to be trained like this:

Table 4 Initial 2-Stage SVM configuration idea

Stage	Gradient	Positive Samples	Negative Samples
1	Hand-5 vs. Low Gradient	1500 Hand-5 Samples randomly Selected	Low Gradient Randomly selected set (4000 from 36 000 Low Gradient Samples)
2	Hand-5 vs. High Gradient	1500 Hand-5 Samples randomly Selected (Same set as above)	High Gradient Randomly selected set (4000 from 210 000 High Gradient Samples)

The 3-stage configuration was to be trained as follows:

Table 5 Initial 3-Stage SVM configuration idea

Stage	Gradient	Positive Samples	Negative Samples
1	Hand-5 vs. Low Gradient	1500 Hand-5 Samples randomly Selected	Low Gradient Randomly selected set (4000 from 36 000 Low Gradient Samples)
2	Hand-5 vs. Medium Gradient	1500 Hand-5 Samples randomly Selected (Same set as above)	High Low Gradient Randomly selected set (4000 from 60 000 High Low Gradient Samples)
3	Hand-5 vs. High Gradient	1500 Hand-5 Samples randomly Selected (Same set as above)	High High Gradient Randomly selected set (4000 from 150 000 High High Gradient Samples)

However this did not result in configurations that were usable for real-time performance. At least 1 SVM stage performed poorly in both cases. Further partitioning was done based on the size of the dataset partition and whether or not it produced an SVM useable for real-time gesture detection.

The final 2-stage, and 3-stage hand-5 configurations were as follows:

2-stage Hand-5:

Table 6 Final 2-Stage SVM configuration Hand-5

Stage	Gradient	Positive Samples	Negative Samples
1	Hand-5 vs. Low Gradient	1500 Hand-5 Samples randomly Selected	High High Low Gradient Randomly selected set (4000 from High High Low Gradient Samples)
2	Hand-5 vs. High Gradient	1500 Hand-5 Samples randomly Selected (Same set as above)	High High High Gradient Randomly selected set (4000 from High High High Gradient Samples)

3-stage Hand-5:

Table 7 Final 3-Stage SVM configuration Hand-5

Stage	Gradient	Positive Samples	Negative Samples
1	Hand-5 vs. Low Gradient	1500 Hand-5 Samples randomly Selected	High High Low Low Gradient Randomly selected set (4000 from High High Low Low Gradient Samples)
2	Hand-5 vs. Medium Gradient	1500 Hand-5 Samples randomly Selected (Same set as above)	High High Low High Gradient Randomly selected set (4000 from High High Low High Gradient Samples)
3	Hand-5 vs. High Gradient	1500 Hand-5 Samples randomly Selected (Same set as above)	High High Gradient Randomly selected set (4000 from High High High Gradient Samples)

The summary of the observations of the multi-stage hand-5 training can be seen in Figure 17.

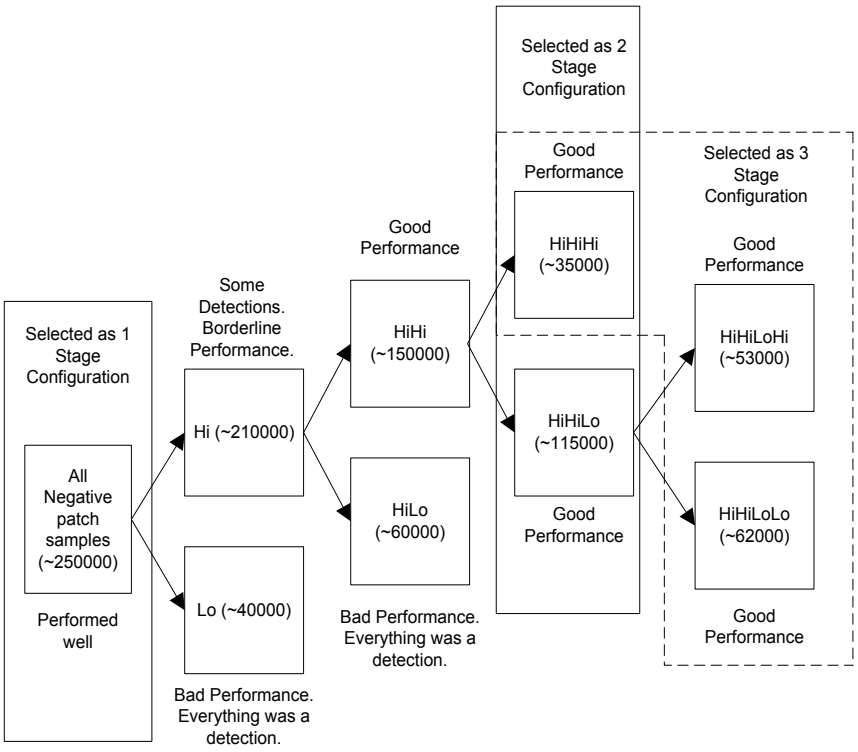


Figure 17 Hand-5 SVM Negative Patch Gradient Selection Tree

As it can be seen from the multi-stage comparison results in the previous section, the 2-stage SVM configuration was selected as the best multi-stage configuration. The SVM gesture detectors for the hand-L and hand-I gestures were also trained using a two stage configuration. These gestures did not produce the same quality of real-time results when trained with the same gradient configurations. Further investigation into which partitions produced good results for real-time online testing was done. The gradient partitions selected for the Hand-L gesture were:

2-stage Hand-L:

Table 8 Final 2-Stage SVM configuration Hand-L

Stage	Gradient	Positive Samples	Negative Samples
1	Hand-L vs. Low Gradient	1500 Hand-L Samples randomly Selected	High High Low Low Low Gradient Randomly selected set (4000 from High High Low Low Low Gradient Samples)
2	Hand-L vs. High Gradient	1500 Hand-L Samples randomly Selected (Same set as above)	High High Low Low High Gradient Randomly selected set (4000 from High High Low Low High Gradient Samples)

The summary of observations of the Hand-L gradient selection can be seen in the next figure.

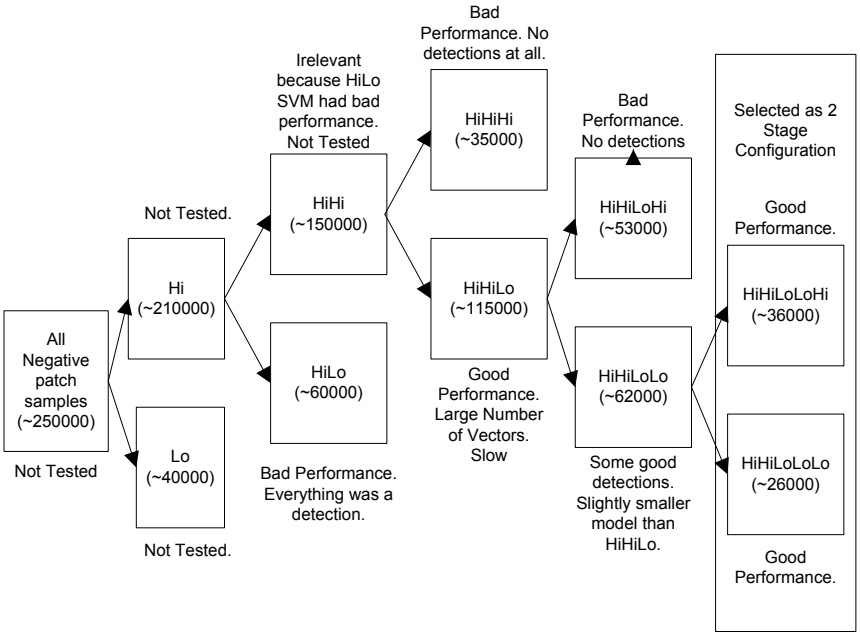


Figure 18 Hand-L SVM Negative Patch Gradient Selection Tree

The gradient partitions selected for the Hand-I gesture were:

Table 9 Final 2-Stage SVM configuration Hand-I

Stage	Gradient	Positive Samples	Negative Samples
1	Hand-I vs. Low Gradient	1500 Hand-I Samples randomly Selected	High High Low Low Low Low Gradient Randomly selected set (4000 from High High Low Low Low Low Gradient Samples)
2	Hand-I vs. High Gradient	1500 Hand-I Samples randomly Selected (Same set as above)	High High Low Low Low High Gradient Randomly selected set (4000 from High High Low Low Low High Gradient Samples)

The summary of observations of the Hand-I gradient selection can be seen in Figure 19.

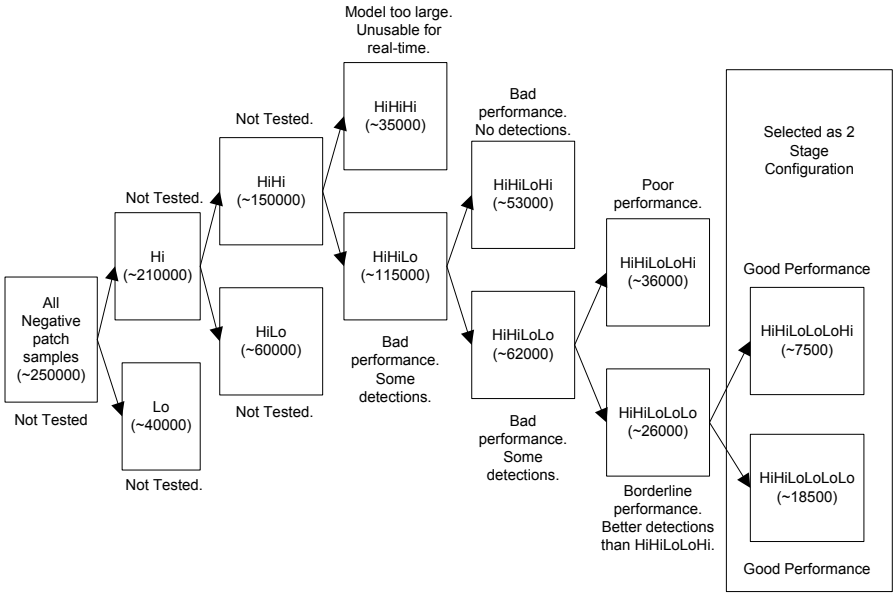


Figure 19 Hand-I SVM Negative Patch Gradient Selection Tree

The results of the HOG SVM Gesture Recognition System can be found later in this chapter where the detectors' ability to recognize gestures is evaluated as well as the complementary effects of using the detectors with the validation functions.

Each different gesture had a configuration which was found by following a different path in the negative patch partition tree. A configuration was chosen when both its upper and

lower gradient split resulted in a well performing SVM. The first well performing SVM configurations found using the tree search were selected for the Hand-L and Hand-I detectors. A heuristic criteria for training SVMs was formulated from the experience of performing this search. It is a simple and effective training heuristic criteria which captures the decision making process that was followed. This heuristic criteria can be used to effectively select training patches based on a chosen type of data separation for an SVM model. The result is a two stage SVM.

Heuristic Training Criteria for training a 2-stage SVM:

1. Split the negative sample data according to a criteria (Magnitude of HOG for this work)
2. Train and Test both SVMs
3. If both produce good results, use them in sequence to improve the results.
4. Otherwise follow the branch that performs better, and repeat the process from step 1. Stop when both SVMs of a split produce good detection results. And if the detections are good but the bounding boxes of said detections are poorly localized the designer can try flipping the order of the SVMs to see if it produces better results.

This heuristic criteria can be used for creating multi-stage SVMs with more than 2 stages. The designer first finds the 2-stage configuration using the heuristic. Then it must be determined which of the SVMs was trained on a larger partition of negative samples. Then a new 2-stage configuration is found on that data partition using the heuristic. These 2 SVMs then replace the SVM from the previous 2-stage configuration which used the larger data partition of negative samples. This strategy was used to find the 3-stage hand-5 SVM configuration. For a higher number of stages the process is simply repeated.

It was also found that flipping the order of the SVMs for the hand-I gesture resulted in better localized bounding boxes of the hand-I detections. By flipping the SVM stage order the higher gradient negatives SVM was used as the first stage, and the lower gradient negatives SVMs was used as the second. A stage SVM can be used as a detector in either lowest to highest or highest to lowest ordering. The ordering is not important, what is important is that the first stage produces well localized detection bounding boxes

which the desired hand gesture. The flipping idea is included in the heuristic to improve the localization of the bounding boxes.

3.5 Comparison to MobileNets

MobileNets [53] is a Convolutional Neural Network (CNN) proposed by Google for Computer Vision applications. Convolutional Neural Networks are recently developed deep learning models that can be used for gesture recognition.

MobileNets is a CNN that uses an efficient architecture whose chief contribution is the use of depth-wise separable convolutions. Instead of using computationally costly standard convolution filters, MobileNets separates the computation of these filters into depth-wise convolutional filters and point-wise convolutional filters. This has the effect of breaking down larger more computationally costly filters into smaller filters that can be convoluted together to produce the same effect. This piece wise composite evaluation achieves an 8-9 times reduction in computational load with only a small loss in accuracy. CNN classification solutions perform quite well in terms of correctly classifying objects in images, however, computational load is a significant concern. By using its depth-wise separable convolutions MobileNets models are lightweight enough for real-time Mobile Vision applications when combined with a region proposal algorithm. With these computational savings MobileNets is an excellent choice to compare to the Gesture Recognition system presented in this chapter.

The Cascade and SVM based Gesture Recognition system will be compared in performance to MobileNets to evaluate how well it does compared to recent neural network based methods used for gesture recognition and other vision applications.

Only gesture detection performance will be measured in the Cascade and SVM Gesture Recognition system. Due to high difficulty gesture localization performance will not be measured. This is why the Gesture Recognition system will be compared to Convoluted Neural Networks only in the gesture detection sense, not in the gesture localization sense.

MobileNets can be directly trained for Object localization using the TensorFlow with Object Localization API [191]. However this method requires slightly different data, that

is positive samples located within full background images. This method also uses a different approach to object localization than the Gesture Recognition system. Instead of the sliding window method employed by the Gesture Recognition system in both HOG Cascade and HOG SVM configurations, Tensorflow with Object Localization, being a CNN based method, uses heatmaps showing the probability of the sought after object appearing across the image. This means that using the TensorFlow with Object Localization API to make a hand gesture detection system with MobileNets would result in a significantly different architecture and would not be the best system to use for comparison to the HOG Cascade and HOG SVM Gesture Recognition system.

MobileNets is compared to the Gesture Recognition system by using a sliding window approach for gesture detection and localization. A MobileNets model is trained for each hand gesture to function as a positive sample vs. negative background image classifier. These classifiers will be run on each image produced by a sliding window on the input frame to determine hand gesture detections. Using the sliding window approach allows the comparison of MobileNets to the Gesture Recognition system to be made using the same data, and the same object localization approach.

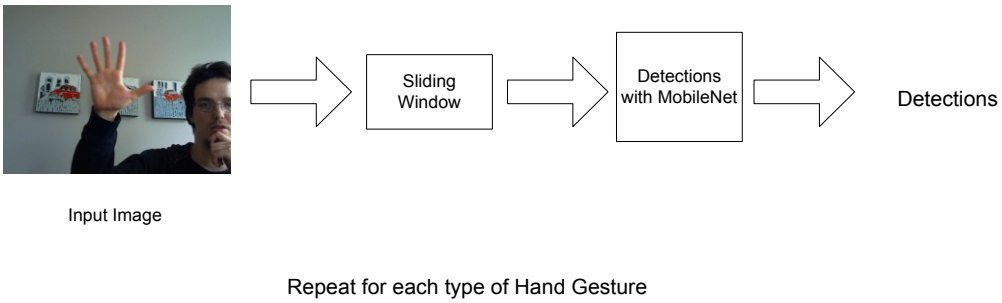


Figure 20 MobileNets Gesture Detection Architecture

The gesture detection performance of the Gesture Recognition system is measured against the MobileNets sliding window gesture detection system. The MobileNets system is not augmented by the validation procedure, it is used a complete system.

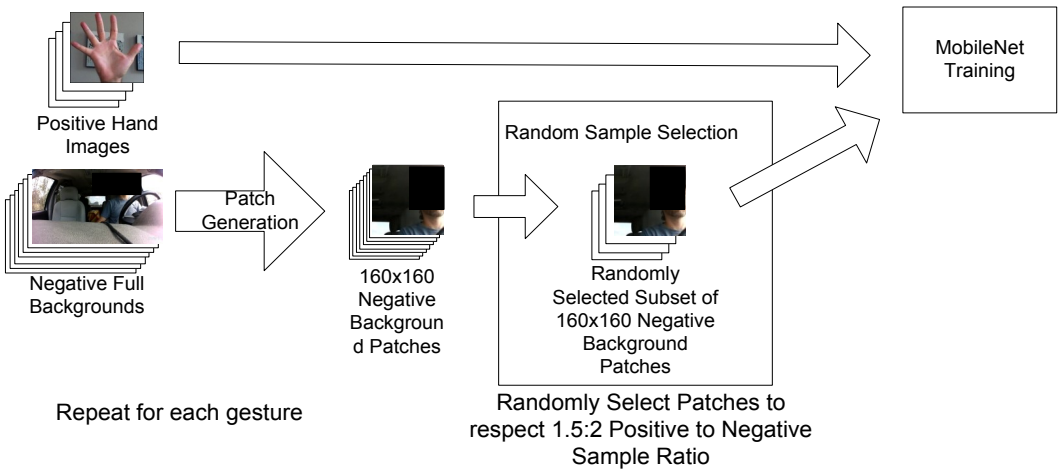


Figure 21 MobileNets Training

Training of the MobileNets classifiers is done on positive hand gesture samples against negative background patches, much like the training of the HOG SVM detectors. Using background patches is typically not how you train a Convoluted Neural Network. However when MobileNets is used with the sliding window approach the negative background patches will represent the type of negative background images generated by the sliding window during the testing of real-time hand gesture detection using a web camera. The training of each MobileNets classifier is done using all of the positive samples of the Hand-5 (7813), the Hand-L (6401), and the Hand-I gestures (6958), that are part of the training dataset. The negative samples for each classifier is provided by randomly selected negative background patches from the total training set of 247 148 patches such that for each MobileNets Classifier there is a 1.5:2 positive to negative ratio relative to the hand 5 gesture patches.

1.5:2 for the positive to negative samples is a ratio that is close to 1:1 which means balanced classes, which generally works better for neural networks. However the slightly larger amount of negative patches will allow the models to benefit a little more from the large negatives dataset in order to be more robust against false detections.

3.6 Results

This section will briefly present results that are more thoroughly discussed in Chapter 4 as part of the complete Hand Gesture Recognition system. The results presented here will

demonstrate the motivation for using the validation functions to complement the performance of the Cascade and SVM gesture detectors. The results presented in Table 10 show how the detectors perform with and without the validation functions that together with the detectors form the Hand Gesture Recognition system presented in the following chapter.

Gesture classification Cascade Configuration with Validation Functions Overall Results

Gesture	% True Positives	% False Positives
hand-5	31.67	0.0485
hand-L	25.58	0.8943
hand-I	16.192	1.0084

Gesture Classification Cascade Configuration No Validation Functions Overall Results

Gesture	% True Positives	% False Positives
hand-5	84.68	2.216649
hand-L	92.49	49.4077
hand-I	98.88	46.39511

Gesture Classification SVM Configuration with Validation Functions Overall Results

Gesture	% True Positives	% False Positives
hand-5	26.33	0.07281
hand-L	20.51	1.828
hand-I	10.11	2.898

Gesture Classification SVM Configuration No Validation Functions Overall Results

Gesture	% True Positives	% False Positives
hand-5	97.13	81.89
hand-L	91.82	81.13
hand-I	100	96.26

Table 10 Comparison of performance of the Cascade and SVM hand gesture detectors with and without the validation functions that together with the detectors form the Hand Gesture Recognition System presented in Chapter 4.

The overall false positive rate for both Cascade and SVM hand gesture detectors is too high to be usable for a real-time context. As will be explained in Chapter 4 only a 3.3% minimum true positive rate is needed to have a 1 detection per second performance which is sufficient for real-time use. However having a high false positive rate is unacceptable.

Comparing the performance of the two detectors with validation functions against the MobileNets model shows another motivation for making the Hand Gesture Recognition described in the following chapter which adds the gesture validation functions to the Cascade and SVM gesture detectors.

	Cascade System		**MobileNets Best Individual Thresholds System**			**MobileNets 0.99 Best Global Threshold System**	
Gesture	% TP	% FP	% TP	% FP	Confidence Threshold Necessary	% TP	% FP
hand-5	31.67	0.04854	**62.3**	0.04854	0.9995	89.08	0.7766
hand-L	**25.58**	0.8943	18.68	0.8943	0.9936	29.91	1.498
hand-I	16.19	1.008	**56.47**	1.008	0.9968	85.04	3.144
mean Precision	82.41		**87.08**			82.09	
mean Recall	24.48		45.82			**68.01**	
mean Accuracy	89.92		92.77			**94.62**	

	SVM System		**MobileNets Best Individual Thresholds System**			**MobileNets 0.99 Best Global Threshold System**	
Gesture	% TP	% FP	% TP	% FP	Confidence Threshold Necessary	% TP	% FP
hand-5	26.33	0.07281	**65.33**	0.07281	0.9993	89.08	0.7766
hand-L	20.51	1.828	**31.61**	1.828	0.9888	29.91	1.498
hand-I	10.11	2.898	**81.74**	2.898	0.9908	85.04	3.144
mean Precision	63.51		**82.78**			82.09	
mean Recall	21.69		59.56			**68.01**	
mean Accuracy	89.73		93.61			**94.62**	

	Cascade System	**SVM System**	**MobileNets System**
Approximate total runtime on test dataset	1 h 44min	4 h 50 min	94 h 4 min
Approximate average runtime per frame	434ms	1210ms	23 541ms

Table 11 Comparison table for Cascade and SVM systems to MobileNets

Using the MobileNets model using the same sliding window approach as the Cascade and SVM detectors shows that while MobileNets beats the true positive rate of the other detectors for a given false positive rate in most cases, this comes at a heavy computational cost resulting in a much slower system when using the same approach. There is a trade off between performance and speed which means that using the Cascade and SVM detectors is more desirable for real-time applications. This is further explained in the results of Chapter 4.

Chapter 4 Hand Gesture Recognition System

4.1 Overview

The previous chapter explained how hand gesture detection can be achieved with machine learning models. This will be an important building block of the Hand Gesture Recognition system which will be presented in this chapter. The hand gesture recognition system presented in this chapter will use the results from the previous chapter to detect hand gestures and then it will add validation tests to alleviate issues with false gesture detections. This makes the system viable for real-time live applications. The testing of the complete system is done with a large hand gesture dataset which allows the system's performance to be tested using both its Adaboost Cascade and Support Vector Machine hand gesture detector configurations. It allows the comparison of the system to neural network MobileNets models which are designed for mobile vision applications.

The Gesture Recognition Algorithm detects hand gestures in real-time from a video, anywhere in the video frame. The algorithm was designed with two goals in mind, real-time hand gesture recognition, and a very low false positive rate. The algorithm has two main steps. The first is a detection step, that uses detectors trained with machine learning to locate potential hand gestures in the frame. The second step uses shape information to either validate or reject the hand gesture detections, and is appropriately named the validation step. The first step achieves the goal of real-time hand gesture recognition by achieving multi scale detections in a video feed in real-time. The second step achieves the low false positive rate by rejecting many of the false detections. The first step is done using the results of the previous chapter. The two steps combine to form a powerful and robust real-time gesture recognition system.

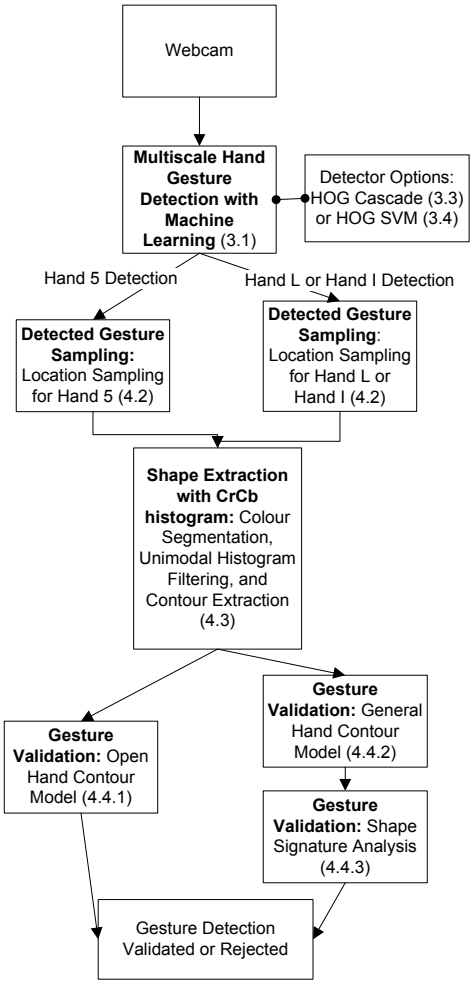


Figure 22 Gesture Recognition System Flow Chart

The chart above shows the way the Gesture Recognition algorithm works.

The machine learning detection step is done with one of two different configurations. One is a HOG Cascade configuration, and the other is a HOG SVM configuration. Both of these were implemented and tested to determine which is better. These detectors are also compared against competing Convolutional Neural Network methods for computer vision applications, specifically MobileNets [53] proposed by Google. All of the machine learning models used in the hand gesture recognition system are the resulting models of the previous chapter. The MobileNets comparison is also made using the results of the previous chapter. For results and comparisons please refer to section *Results*.

The shape validation relies on colour features in order to extract hand contours which are subsequently validated by shape analysis. Colour features are used because they are a simple and reliable way to segment objects in a 2D video only context. Motion and background subtraction features can also be used, however these impose constraints on the system. Background subtraction requires a good knowledge of the background model, which requires training and is not always readily available. Ultimately relying on a background model makes a system less flexible. Motion features impose the constraint that a user's hand must move in order to be detected. This limitation makes a system more difficult to use, because ideally a system should respond to a stationary hand. This is why colour features were chosen for this system. Additionally colour features are used in the tracking method described in the next chapter for the same reasons of lower overhead and more flexibility.

The shape validation step is performed using both geometric contour shape analysis, and contour shape signature analysis. These are a series of tests that were determined experimentally, and they work to eliminate false detections and to confirm true detections of hand gestures. Using these shape analysis methods, the validation step determines the number of extended visible fingers and the palm of a potential hand shape resulting from a potential hand gesture detection, and this information serves as a powerful discrimination tool to verify the presence of specific hand gestures.

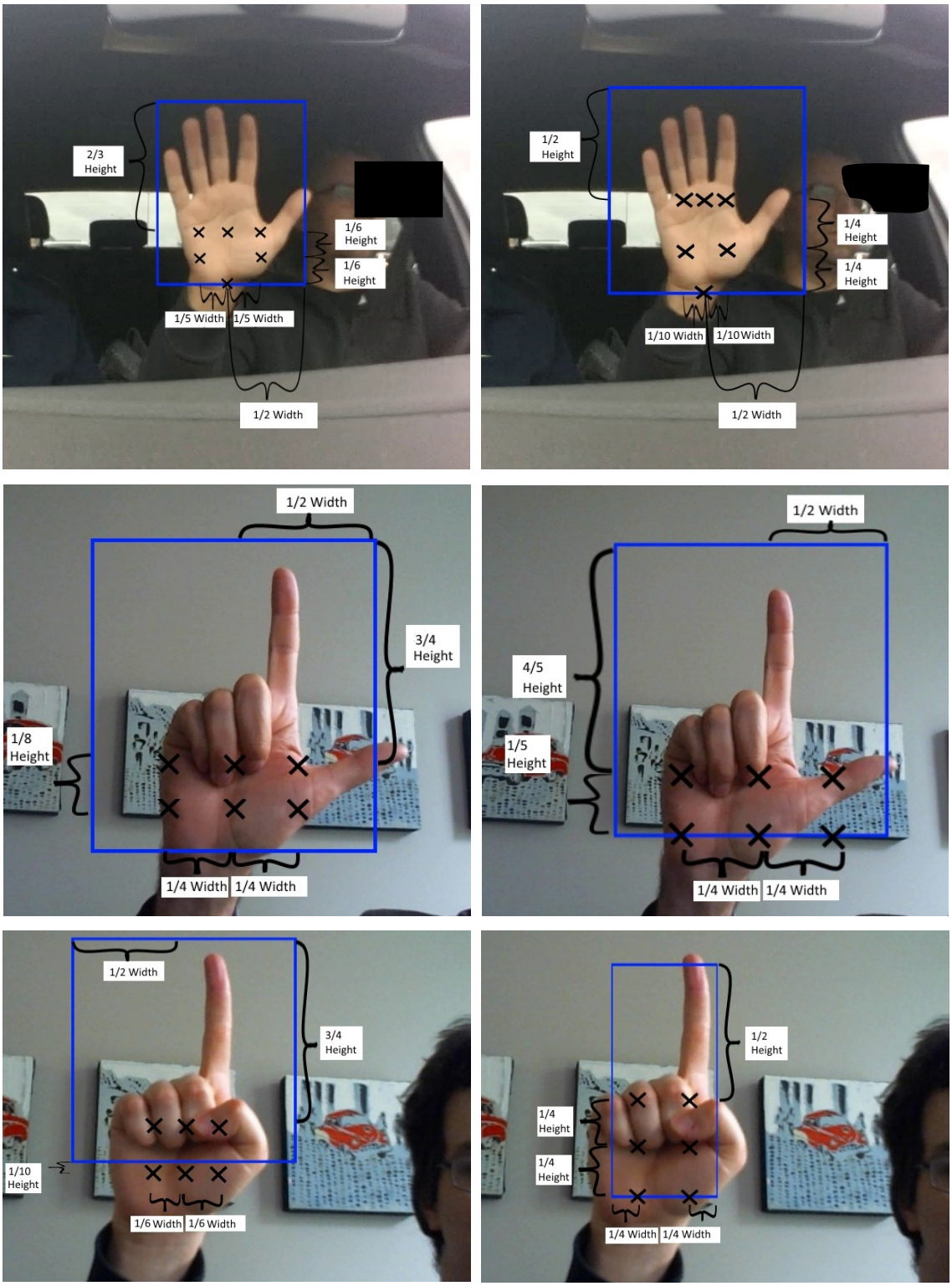
When a hand gesture has been detected by the machine learning detection step, and then validated by the shape validation step, the Gesture Recognition algorithm declares that that specific hand gesture is recognized. The input video frame is displayed with the validated hand gesture highlighted.

4.2 Detected Gesture Sampling

For each gesture detection produced in the detection step a validation procedure is applied using the gesture detection bounding box. The validation procedure requires an input image which is the original video frame, and a sampling point. The goal is to segment the hand shape and determine if it corresponds to the detected gesture. The area around the sample point has all of its colours recorded, and this colour sampling is used

for the segmentation. So in order for the validation procedure to work correctly, and validate a hand gesture when it is present, it has to receive an image of the hand gesture and a sampling point located somewhere within that hand gesture. The location of this sampling point is defined by the location of the gesture detection bounding box. Logically it makes sense that if a detection box is well centered on a present hand gesture then the center of the box becomes the sampling point to use for the validation procedure. In practice however a given detection bounding box is often not well centered. That is why the placement of the sampling point has to be determined experimentally. It is also better to use several sampling points and make several calls to the validation procedure, instead of just using one. By using a sampling pattern of sampling points it increases the chances of validating a hand gesture when it is present. The positions of the points as mentioned before have to be determined experimentally based on where the detection boxes for a given gesture tend to appear relative to the hand gesture itself. In the case of this system, these sampling patterns where determined with online testing using a webcam and a user's hand.

Each type of gesture has a different sampling pattern for its detection box. The validation procedure is applied at each sampling point. Figure 23 shows the sampling patterns for each type of gesture, with bounding boxes detecting hand gestures. The figures represent detection results that often occur. The left column shows the sampling patterns for the 3 gestures as they are used in the Gesture Recognition system with HOG Cascade detection, and the right column shows the sampling patterns in the Gesture Recognition system with HOG SVM detection. The sampling patterns differ because the nature of the bounding boxes from the HOG Cascade and HOG SVM detectors differ. They differ in terms of location relative to the presented hand gestures, and they also differ in size. The sampling patterns try to select the best locations that correspond to points inside the hand gestures for the majority of hand gesture detections.



Cascade Detection Boxes Sampling Patterns **SVM Detection Boxes Sampling Patterns**

Figure 23 Sampling Patterns of the Cascade and SVM detector bounding boxes

The gesture validation procedure uses the original input image and the sample point to validate the shape of the hand gesture that should appear in the detection box.

For a given sample point the validation procedure uses unimodal histogram filtering to find the best binary back projection of the current input image using colour information sampled around the sampling point. These back projections are afterwards used to validate the hand gesture shape of any hands that appear in the bounding box.

Valid hand shapes result in validated detections. If a single sample point yields a valid hand shape then that detection is declared valid. If every sample point results in only invalid hand shapes then that detection box is discarded as false.

The sample patterns were selected experimentally depending on where the hand gestures were often located within the detection boxes during typical operation of the Gesture Recognition system. In practice these patterns can be readjusted to suit the performance of different detectors.

4.3 Shape Extraction with CrCb histogram: Unimodal Histogram Filtering and back projection

This algorithm takes an input of a colour image and a sample point defined by the sampling pattern of the hand gesture detection. It extracts a CrCb colour histogram around the sample point, filters it for unimodality, and does a back projection. The output is a high quality back projection which has clear hand contours when the sample point is located within a hand in the original input image.

The details of the process are as follows:
1) Define a 30x30 region of interest (ROI) in the image centered around the sample point. Convert the ROI and the input image into YCrCb colour space. and then calculate the 2 dimensional Cr-Cb histogram for the ROI. Additionally, calculate the Y (Luma), Cr (Red-difference), and Cb (Blue difference) mono channel histograms for the ROI and their corresponding cumulative distribution functions (CDFs).

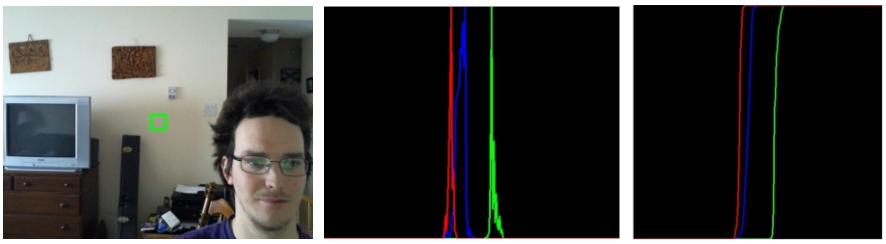


Figure 24 Input image with a region of interest defined in green (Left), Y, Cr, and Cb histograms (Middle), with the corresponding CDFs (Right)

2) Determine whether or not each CDF is unimodal. Unimodality is a property that manifests itself in a given CDF as a single step in the graph [79], [192]. A CDF can only be of a unimodal histogram distribution if it has one and only one distinct step, meaning only one distinct region of positive increase.

3) If the Cr, and Cb channels are unimodal then the original 2d Cr-Cb histogram should be replaced by an improved Cr-Cb histogram. The limits of the unimodal steps, or regions of positive increase, in the Cr and the Cb channels must be known. The step limits are used to create a square of white (1s) in a new binary CrCb histogram which is identical in size to the original histogram. The step limits are converted to use the CrCb histogram's binning system. If unimodality is not confirmed for the two channels then the process ends here for the given input image, and the back projection is performed with the ordinary Cr-Cb histogram.

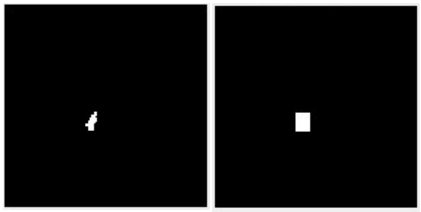


Figure 25 Regular unimodal CrCb histogram (left) and corresponding morphologically generated CrCb histogram (right).

5) Perform the back projection. The CrCb histogram is a 2d binary map that records whether or not a given Cr and Cb colour value pair is present. It is used as a look up table in order to segment the input image.

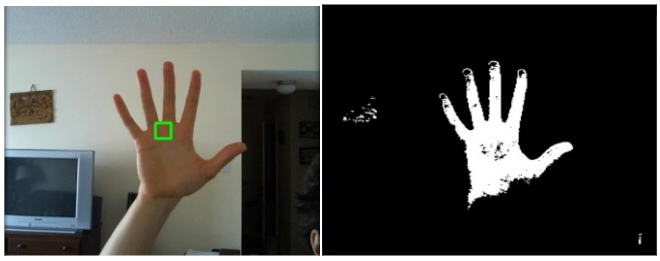


Figure 26 Example of back projection segmentation

Figure 26 demonstrates the results of a back projection. The CrCb histogram that is generated from the colour information in the ROI, indicated in green on the left image, is used as a look-up table to segment that same image.

4.3.1 Benefits of Unimodal Filtering

When a CrCb histogram is calculated for a given region of interest only the colours within that region are represented. But what if two shades of skin colour (human pink for example) are present but an intermediate shade is not? If that intermediate shade appears anywhere in the rest of the hand those areas will not register in the back projection. This effect occurs because of quantization noise and binning.

Additionally other colours that are part of the hand simply get cut out because they didn't register in the region of interest. Because of colours being omitted, hand shapes can degenerate quickly from fairly distinct to unrecognizable and disjoint blobs.

Figure 27 CrCb histogram (left) and corresponding back projection (right).

The problem areas are highlighted in red on the back projection. The areas which are likely containing useful colour information in the histogram are also highlighted. The following figures show some different results obtained using unimodal filtering. Note how the holes in the hand were reconstructed.

Figure 28 Example Result 1. The left back projection has holes that are fixed in the right back projection

Figure 29 Example Result 2. The left back projection in missing large portions of the fingertips. The fingertips are much better segmented in the right back projection.

Figure 30 Example Result 4. The left back projection has some occlusion with the wooden shelf. The right back projection has more significant occlusion. More of the hand is selected by the right segmentation but the occlusion is also made worse.

Figure 30 is another good reconstruction. However the occlusion of the hand with the neighbouring wooden shelf is made worse using the enhanced CrCb histogram back projection. Occlusion of hands with bright red objects have also been known to cause issues.

Overall the histogram filtering works to improve the quality of the shapes of the back projection. Clearer hand shapes enable faster and more reliable recognition which improves the hand recognition rate.

It is important to note that in the segmentation examples in this section, objects in the background have also been segmented. This is due to having similar colours as the

colours in the sampling zone marked by the green rectangle. In practice only the contours that overlap or intersect with the sampling zone are retained as valid.

4.3.2 Results of Unimodal Histogram Filtering

Experiments have been run with the HOG Cascade Gesture Recognition and the HOG SVM Gesture Recognition systems to determine how unimodal histogram filtering affects the performance of the validation functions. Both systems were run through using the testing dataset using full validation functions with unimodal filtering, and using validation functions without unimodal filtering.

Cascade System with Validation Functions with Unimodal Filtering Confusion Matrix Overall

samples \results	hand-5	hand-L	hand-I	back ground	total
hand-5	**641**	19	2	1362	2024
hand-L	1	**419**	52	1166	1638
hand-I	0	41	**188**	1562	1791
back ground	5	54	34	**8839**	8932

Cascade System with Validation Functions without Unimodal Filtering Conf Matrix Overall

samples \results	hand-5	hand-L	hand-I	back ground	total
hand-5	**449**	34	12	1529	2024
hand-L	2	**433**	65	1138	1638
hand-I	0	67	**285**	1439	1791
back ground	3	118	47	**8764**	8932

Cascade System with Validation Functions with Unimodal Filtering Gesture Classification Overall Results

Gesture	% True Positives	% False Positives
hand-5	31.67	0.04854
hand-L	25.58	0.89433
hand-I	10.4969	0.69875

Cascade System with Validation Funtions without Unimodal Filtering Gesture Classification Overall Results

Gesture	% True Positives	% False Positives
hand-5	22.1838	0.04045
hand-L	26.4347	1.718051
hand-I	15.9129	0.984596

Background classification with Validation Functions with Unimodal Filtering

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	8839	93	5	54	34
percent	N\A	98.9588	1.0412	0.056	0.605	0.3807

Background classification with Validation Functions without Unimodal Filtering

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	8764	168	3	118	47
percent	N\A	98.119122	1.88088	0.034	1.3211	0.526

Table 12 Side by side comparison tables of the effects of Unimodal Filtering on performance of HOG Cascade Gesture Recognition System

It can be seen that in almost every case the unimodal filtering helps to reduce the false positive rate. The only exception to this is for the hand-5 gestures where it actually adds 0.008% to the false positive rate while at the same time increasing the true positive rate by almost 10%. Another interesting thing to note is that for the hand-I gesture the false positive rate gets cut by 0.3% when using the unimodal filtering, but the true positive rate drops by 5%.

Overall the unimodal filtering has a positive effect on the HOG Cascade Gesture Recognition system, increasing the true positive rate of the hand-5 gesture while keeping its false positive rate marginally higher, and reducing the false positive rate of everything else. It should be noted that because the hand-I gesture had a low false positive rate of about 1% without the unimodal filtering, and it only experienced a 0.3% reduction with unimodal filtering at the expense of 5% of its true positive rate, the unimodal filtering was not used for the validation functions of the hand-I gesture in the HOG Cascade Gesture Recognition system. It was however still used for the hand-5 and hand-L validation functions to positive effect. Despite the drop of 5% in the true positive rate of the hand-I gesture, it is still over 10% with the unimodal filtering. This is still a very good true positive rate for the purposes of this project, and more than enough to provide robust performance in a real time context. However this is clearly a case of trade-off between true positive rate and false positive rate.

Detection Results SVM System with Validation Functions with Unimodal Filtering Confusion Matrix Overall

samples \results	hand-5	hand-L	hand-I	back ground	total
hand-5	**533**	37	13	1441	2024
hand-L	2	**336**	57	1243	1638
hand-I	3	60	**181**	1547	1791
back ground	4	136	295	**8497**	8932

Detection Results SVM System with Validation Functions without Unimodal Filtering Confusion Matrix Overall

samples \results	hand-5	hand-L	hand-I	back ground	total
hand-5	**366**	62	20	1576	2024
hand-L	1	**321**	86	1230	1638
hand-I	0	84	**226**	1481	1791
back ground	9	357	392	**8174**	8932

Gesture Classification SVM System with Validation Functions with Unimodal Filtering Overall Results

Gesture	% True Positives	% False Positives
hand-5	26.33399	0.07281
hand-L	20.51282	1.827881
hand-I	10.10609	2.898205

Gesture Classification SVM System with Validation Functions without Unimodal Filtering Overall Results

Gesture	% True Positives	% False Positives
hand-5	18.083	0.0809
hand-L	19.5971	3.94603
hand-I	12.6186	3.95426

Background classification SVM configuration with Validation Functions with Unimodal Filtering

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	8497	435	4	136	295
percent	N\A	95.12987	4.87013	0.045	1.523	3.303

Background classification SVM configuration with Validation Functions without Unimodal Filtering

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	8174	758	9	357	392
percent	N\A	91.513659	8.48634	0.101	3.997	4.389

Table 13 Side by side comparison tables of the effects of Unimodal Filtering on performance of HOG SVM Gesture Recognition System

The unimodal filtering works very well with the validation functions in the SVM Gesture Recognition system. The false positive rates are lowered for all three gestures, with significant reductions for the hand-L and hand-I gestures. There is also a significant boost in the true positive rate of the hand-5 gesture when using unimodal filtering. That is why

for the SVM Gesture Recognition system unimodal filtering is used for the validation functions in all three gestures.

Overall, the contributions of the unimodal filtering significantly improve the performance of both the Cascade and the SVM systems, helping to reduce the false positive rates, while keeping the true positive rates high.

4.4 Gesture Validation

The back projection produced using a given sample point has its contours analyzed to determine if there is a valid hand shape for the specific hand gesture detection. The hand shape contours are evaluated using tests specific to the hand gesture detected by the multi scale detector. There are 2 different procedures that are applied depending on the what type of gesture was detected.

If the back projection was produced by a hand-5 detection box, then that back projection and its sample point, are processed by the Open Hand contour model to determine if it has a hand-5 shape at the sample point. If the back projection was produced by a hand-L or a hand-I detection box, then the back projection and its sample point are processed by the General Hand model, and shape signature analysis to determine how many valid extended fingers are present in the contour shape at the sample point. If this number is 2 for both tests for a hand-L gesture, or 1 for a hand-I gesture, then that gesture is declared valid. If a gesture detection box fails to produce a contour shape with the correct extended finger characteristics required for its hand gesture, then that detection is discarded as false.

4.4.1 Validating the hand-5 gesture detections with the open hand model

When there is a hand-5 gesture detection in the multi scale gesture detection step, the detections are validated with the open hand contour model. The open hand contour model is a series of geometric tests that is applied to a contour to determine if it corresponds to a hand-5 shape. The algorithm calls the open hand contour model with a sampling point and a segmented binary image. The constraint is imposed that a contour found in the segmented image by the model must contain the sampling point within its boundaries in

order to be considered valid. If the open hand model finds a hand-5 shape containing the original sampling point then the hand gesture detection is declared valid.

The open hand model used for hand-5 recognition is a series of point calculations and proportionality ratios that are applied to a given contour shape and its corresponding convexity defects. The model any contour found at the sampling point and determines if it has a valid hand-5 gesture shape. The hand-5 gesture is an open hand with 5 visible extend fingers. The open hand contour model was determined experimentally with online testing with video input from a web camera. The specifics of the geometric calculations that are involved are explained in the appendix. The open hand contour model helps to validate the hand-5 gesture and discard false positives.

4.4.2 Finger counting with the General Hand Model for Hand-L and Hand-I detections

When there is a hand-L or a hand-I gesture detection, the detections are processed with the general hand contour model. The general hand contour model is a series of geometric tests that is applied to a contour to determine its palm location and count any visible extended fingers. The algorithm calls the general hand contour model with a sampling point and a segmented binary image. The constraint is imposed that a contour found in the segmented image by the model must contain the sampling point within its boundaries in order to be considered valid. If a valid contour is found the model it treats the contour as a potential hand and estimates the location of its palm and any visible extended fingers. The valid contour and its finger count are retained for use with the shape signature analysis which is the next step for validating hand-L and hand-I gesture detections.

The model is a series of point calculations and proportionality ratios that are applied to a given contour shape to find the palm and fingertips. Many of the ratios are slightly different than the open hand contour model. This is due to the hand shape changing when not all fingers are extended. The central palm or fist area becomes larger as fingers are folded into the hand and this changes the proportions compared to the hand-5 gesture. In contrast to the open hand model, any number of fingers are accepted as valid for a given contour and that finger count is then compared against the detected gesture to determine

if that gesture is valid. A valid contour must also be of a certain minimum size defined by the model. The model is also used for hand tracking as described in Chapter 4.

The generalized hand contour model was determined experimentally with online testing with video input from a web camera. The specifics of the geometric calculations that are involved are explained in the appendix. The general hand contour model helps to validate the hand-L and hand-I gestures, and discard false positives.

4.4.3 Finger counting with Shape Signature Analysis for Hand-L and Hand-I detections

When validating either the Hand-L or Hand-I gesture, a second shape validation test is done alongside the finger counting with the General Hand model called shape signature analysis. This is a test that analyzes a contour's shape signature to determine the amount of extended visible fingers, with the hope that it matches the number found by the General Hand model. If that is 2 fingers for Hand-L and 1 finger for Hand-I, then that gesture detection is declared valid. Otherwise it is declared invalid and discarded.

The shape signature analysis method takes as input the hand contour found at the sample point by the General Hand model. It returns the number of extended visible fingers found using the shape signature. The following procedure explains the process.

1. Obtain the contour from the input. Find the distance transform of the contour, then find its center of mass. The next step is to filter out the wrist points of the hand contour. Measure the angle of each point along the contour relative to this center of mass. Then the points that fall within 15 degrees of the downward facing normal are filtered out of the contour.

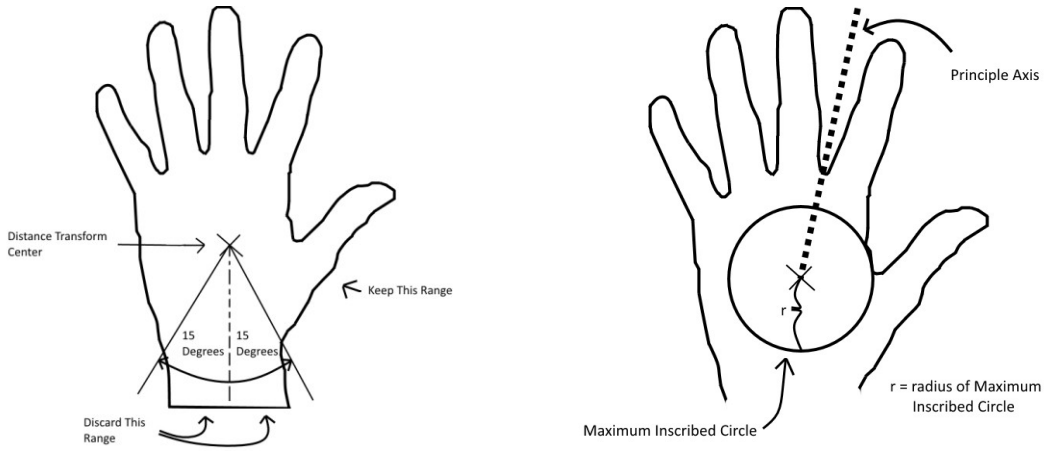


Figure 31 Filtering out the wrist (Left), PCA Analysis and Maximum Inscribed Circle(Right)

4. Find the principle axis of the filtered contour using Principle Components Analysis (PCA). Then find the maximum inscribed circle of the contour using the contour's center of mass. This is done by finding the closest contour point to the center of mass and the distance becomes the circle radius.

6. Calculate the shape signature of the contour using the max inscribed circle. The signature is found by plotting angle vs. distance for each point in the contour relative to a central location. Use the center of mass as the center, and then scale the distance values of the contour boundary points by the radius of the maximum inscribed circle. The points should remain connected to each other as they are plotted.

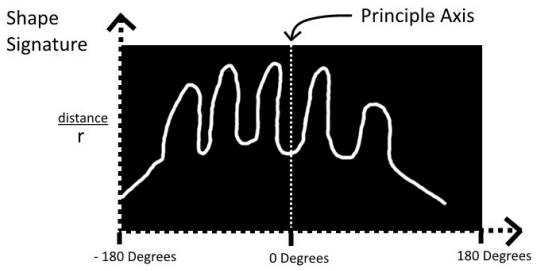


Figure 32 Shape Signature Example

7. Threshold the shape signature. Use 1.5 (distance/circle radius) as the threshold. The idea to use 1.5 as a threshold for finger separation was presented in [60], [61]. It is used in this algorithm because it performs well.

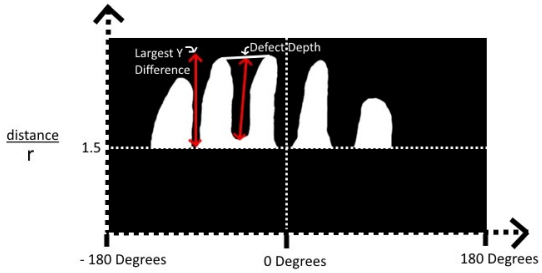


Figure 33 Shape signature with threshold. Red lines shows example of finding fused contours.

8. Find the finger contours that should now be separated after the thresholding. Some of these may actually be multiple visible extended finger contours fused together and these should be separated. Take the found contours and find the largest difference in y values amongst all the boundary points of each contour. Then compute the convexity defects of each contour. If the depth of a convexity defect is generally upright, and has a y value difference that is equal to or greater than 70% of the largest y value difference of the corresponding contour, then a cut needs to be made at that convexity defect's inner depth point in order to separate that contour.

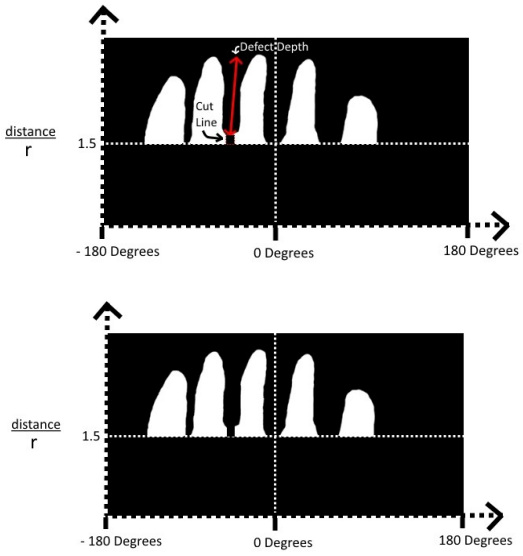


Figure 34 Separating the fused finger contours(Left) with Final Shape Signature with Finger Separation (Right)

9. Find the amount of contours present in the shape signature after cutting. This is the number of extended visible fingers. This is the number that is returned by the shape signature analysis algorithm.

If the finger counts of both the general hand model, and the shape signature analysis match the sought after number of fingers for a given gesture, that gesture detection is considered valid, otherwise that detection is discarded as false.

4.4.4 Affects of using the validation functions

The HOG Cascade Gesture Recognition System and the HOG SVM Gesture Recognition system benefit tremendously from the validation functions. The functions serve to confirm the presence of true gesture detections and reject false positives.

Background classification Cascade Configuration No Validation Functions

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	3660	6379	127	3793	2459
percent	N\A	40.976265	71.4174	1.422	42.47	27.53

Table 14 compares the results for the HOG Cascade Gesture Recognition system with and without using validation functions.

Detection Results Cascade System Confusion Matrix Overall

samples \results	hand -5	hand -L	hand -I	back ground	total
hand-5	**641**	19	7	1357	2024
hand-L	1	**419**	71	1147	1638
hand-I	0	41	**290**	1460	1791
back ground	5	54	49	**8824**	8932

Detection Results Cascade System No Validation Functions Confusion Matrix Overall

samples \results	hand -5	hand -L	hand -I	back ground	total
hand-5	**1714**	1225	1833	23	2024
hand-L	129	**1515**	1551	16	1638
hand-I	18	1280	**1771**	16	1791
back ground	127	3793	2459	**3660**	8932

Gesture classification Cascade Configuration with Validation Functions Overall Results

Gesture	% True Positives	% False Positives
hand-5	31.67	0.0485
hand-L	25.58	0.8943
hand-I	16.192	1.0084

Gesture Classification Cascade Configuration No Validation Functions Overall Results

Gesture	% True Positives	% False Positives
hand-5	84.68	2.216649
hand-L	92.49	49.4077
hand-I	98.88	46.39511

Background classification Cascade Configuration with Validation Functions

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	8824	108	5	54	49
percent	N\A	98.790864	1.20914	0.056	0.605	0.549

Background classification Cascade Configuration No Validation Functions

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	3660	6379	127	3793	2459
percent	N\A	40.976265	71.4174	1.422	42.47	27.53

Table 14 Side by side comparison tables of the effects of using validation functions on the performance of the HOG Cascade Gesture Recognition System

The validation functions significantly reduce the false positive rate of the HOG Cascade system. It is important for the system to be usable in a real-time user interface context to have this low false positive rate. It is interesting to note that the false positive rate of the hand-5 gesture is the lowest out of the 3 gestures in the Cascade system without validation functions. This could be because of the more distinct nature of the hand-5 shape. It is also important to note that unimodal histogram filtering is only used for the

validation function hand-5 and hand-L gestures in HOG Cascade Gesture Recognition system. The false positive rate for the hand-I gesture was only marginally lower when using the filtering, and by not using it the true positive rate for the hand-I gesture is significantly improved. More information on unimodal histogram filtering and its effects is available earlier in this chapter.

Detection Results SVM System Confusion Matrix Overall

samples \results	hand -5	hand -L	hand -I	back ground	total
hand-5	**533**	37	13	1441	2024
hand-L	2	**336**	57	1243	1638
hand-I	3	60	**181**	1547	1791
back ground	4	136	295	**8497**	8932

Detection Results SVM System No Validation Functions Confusion Matrix Overall

samples \results	hand -5	hand -L	hand -I	back ground	total
hand-5	**1966**	1644	2024	0	2024
hand-L	1568	**1504**	1638	0	1638
hand-I	1781	1187	**1791**	0	1791
back ground	6774	7511	8461	**301**	8932

Gesture Classification SVM Configuration with Validation Functions Overall Results

Gesture	% True Positives	% False Positives
hand-5	26.33	0.07281
hand-L	20.51	1.828
hand-I	10.11	2.898

Gesture Classification SVM Configuration No Validation Functions Overall Results

Gesture	% True Positives	% False Positives
hand-5	97.13	81.89
hand-L	91.82	81.13
hand-I	100	96.26

Background classification SVM configuration with Validation Functions

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	8497	435	4	136	295
percent	N\A	95.13	4.870	0.045	1.523	3.303

Background classification SVM configuration No Validation Functions

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	301	22746	6774	7511	8461
percent	N\A	3.370	254.7	75.84	84.09	94.73

Table 15 Side by side comparison tables of the effects of using validation functions on the performance of the HOG SVM Gesture Recognition System

It can be clearly seen that while the validation functions reduce the true positive rate of detection of the SVM system, they are instrumental in ensuring a low false positive rate.

This is required to make the system viable for use as a user interface. The true positive rate is never the less high enough to make the system responsive in a real-time setting. The validation functions are therefore a very important step in this gesture recognition system.

Both the Cascade and SVM based Gesture Recognition systems require validation functions to achieve a low false positive detection rate for their gestures. Without validation functions too many false positive detections occur which reduces the usability of these methods in the context of user interfaces where false positives could mean false commands. By using the validation functions the false positive rates are significantly reduced, and enough of the true detections are validated in order to make the system usable, responsive, and robust in a real-time user interface context.

4.5 Results

4.5.1 Testing

The Gesture Recognition system was tested using the testing dataset. The testing dataset is comprised of 2024 hand-5 samples, 1638 hand-L samples, 1791 hand-I samples, and 8932 negative backgrounds. This corresponds to 20% of the complete dataset. These samples are taken from situations where the system was intended to function, namely indoor or semi indoor settings with sufficient lighting. That is why the samples are generated from office and in car environments. The negative backgrounds are samples of office and in car environments as well as a range of other indoor environments without hand gestures being displayed.

For testing each gesture sample is randomly placed on a generic office background. Negative background samples are just used as they are. All of these samples are processed by the algorithm of the system configuration that is being tested. The algorithm is run on a given sample and the detection result is recorded as a bit string. An example of a result bit string is:

1 0 0 0

^ ^ ^ ^

h5 hL hI background

Each digit in the bit string represents one of the hand gesture detections or the background, with h5 being the hand-5 gesture, hL being hand-L, hI being hand-I, and background indicating no detections, respectively.

The performance of the system was tested in both HOG Cascade and HOG SVM configurations, and it was compared to the MobileNets comparison system. Each system was run on all the testing samples and the results were tallied. The results were recorded in a confusion matrix.

The testing was done on each sample with each system, and it was recorded what kind of detections were made if any. If no detections were made that sample will be recorded as having been detected to be a background. The raw sample numbers are recorded and the sums are tallied and recorded in the confusion matrix. The HOG Cascade and the HOG SVM Gesture Recognition systems were tested with and without their validation functions. This is a brief summary of all 5 system configurations:

- HOG Cascade Gesture Recognition System without validation functions

- HOG Cascade Gesture Recognition System with validation functions

- HOG SVM Gesture Recognition System without validation functions

- HOG SVM Gesture Recognition System with validation functions

- MobileNets comparison Gesture Recognition system

The execution time of processing the whole testing dataset was recorded for each of the 5 system configurations.

In addition to the confusion matrix the gesture detection performance was measured via true positive detection rate and false positive detection rate. These quantities are calculated for all 3 gestures from the results in the confusion matrix.

For a given gesture the rates are calculated as follows:

$$True\ Positive\ \% = \frac{\#\ of\ gesture\ samples\ of\ a\ given\ class\ detected\ as\ samples\ of\ that\ same\ gesture\ class}{\#\ of\ gesture\ samples\ of\ that\ class} * 100\%$$

$$False\ Positive\ \% = \frac{\left(\begin{array}{c}\#\ of\ gesture\ samples\ of\ other\ classes\ detected\ as\ samples\ of\ a\ given\ gesture\ class \\ + \\ \#\ of\ background\ samples\ detected\ as\ samples\ of\ that\ same\ gesture\ class\end{array}\right)}{\#\ of\ gesture\ samples\ of\ other\ classes + \#\ of\ background\ samples} * 100\%$$

For the hand-5 gesture this would look like this:

$$Hand\ 5\ True\ Positive\ \% = \frac{\#\ of\ hand\ 5\ samples\ \ detected\ as\ samples\ of\ hand\ 5\ class}{\#\ of\ hand\ 5\ samples} * 100\%$$

$$Hand\ 5\ False\ Positive\ \% = \frac{\left(\begin{array}{c}\#\ of\ gesture\ samples\ of\ other\ classes\ detected\ as\ samples\ of\ hand\ 5\ class \\ + \\ \#\ of\ background\ samples\ detected\ as\ samples\ of\ hand\ 5\ class\end{array}\right)}{\#\ of\ gesture\ samples\ of\ other\ classes + \#\ of\ background\ samples} * 100\%$$

The other two gestures have similar rate calculations with the hand-5 gesture being replaced by those particular gesture classes in the equations.

The background sample misclassification is also evaluated. An example of the way the results are tallied can be seen in Table 19. All the background samples were processed by the Gesture Recognition system in all its configurations. These sometimes produced gesture detections which were of course false positives. These results as well as correct background sample classifications, which did not produce gesture detections, were tallied in background sample classification tables. This gives a sense of how good the system is at telling gestures apart from the background.

Discussion of testing methods

In configurations where there are no validation functions it is possible to detect multiple hand gestures in one sample. The systems with validation functions are made to operate in a hierarchy; meaning that only one gesture can be validated for a sample. Therefore the percentages may not end up adding to 100% for a given row (representing one class of testing samples) for the configurations without validation functions.

The localization of the gesture detection is not tested. The system is only tested for gesture detections with each given sample. This means the position of where the system detects the gesture versus where it was placed in the scene is not evaluated. This is because from a user interface perspective if a gesture is correctly identified it can be used to issue a command even if it's not well localized.

There is a limitation to note in testing without localization. During the testing of the systems on a few occasions a gesture was validated using the contour of an incorrect object because its shape resembled that of a true gesture contour. The gesture thus detected would correspond to the correct gesture class of the sample image and it would be registered as a true positive. Every system was tested in the same manner. Therefore it is still valid to compare the performance of the different systems, and this false true positive event did not occur often. However this is something that can be improved in future work by overlap testing of recognition results with the target ground truth bounding boxes of the gesture samples placed on the generic background. This would test the gesture detection localization.

There are further questions to solve in how to evaluate localization. How much overlap is considered good overlap? Does the detection produced by the system have to be completely inside the gesture sample's area within the generic background or must it envelope this entire area? Furthermore multiple detections are produced by the gesture recognition system for a single validated gesture sample. One is the gesture detection bounding box for a given detection, whereas the other is the hand contour itself. So which one should be used? These questions need to be resolved in order to evaluate detection localization. These are not of the most pressing interest when trying to determine if a

system correctly identifies gestures. The most immediately interesting measurement is to see if a system finds the correct class of gesture when one is displayed.

If a gesture's class is correctly identified that is a successful recognition even if the extracted hand contour fuses or connects with objects in the background. Likewise if somehow an incorrect object gave the right contour shape to validate a gesture, correct identification of the gesture class still took place even though this occurred with an incorrect object. Evaluating how well a detection is localized is not an evaluation of gesture detection performance, but more of an evaluation of gesture detection quality.

Evaluation localization would require implementing non-max suppression for the MobileNets comparison system. With the current comparison system the same gesture sample can get detected at multiple scales.

Evaluating localization is however, important to determine the quality of gesture detection. Detection quality can affect subsequent hand tracking that is performed using a given gesture detection as an initialization for the tracking. Localization is evaluated in the Gesture Recognition with Hand Tracking system presented in the next chapter.

For this Gesture Recognition system gesture detection performance is measured and not gesture detection localization. The focus is on evaluating the system's responsiveness to the presence of user gestures and negative backgrounds and not on detection quality. Gesture detection localization testing can be implemented in future work as an additional evaluation metric for the Gesture Recognition system to evaluate the quality of detections. Gesture detection localization testing would solve the limitations that have been mentioned. The current testing is however sufficient for evaluating the Gesture Recognition system's responsiveness to the presence of user gestures.

False positive testing is considered in testing the performance of this system because false positives limit the usability of the system, because false detections can result in false commands. False negative performance also can affect the usability of the system. A false negative rate that is too high means that detections do not occur often enough. However for a real-time system running at 30 fps, 1 detection every second is sufficient, and this corresponds to a 3.3% true positive rate. Likewise a 3.3% true positive rate is

equivalent to a 96.7% false negative rate for a given gesture. Therefore by considering true positive rate as has been done in this work false negative rate is also account for, and true positive rate is a more intuitive way of presenting the performance of the system rather than the false negative rate.

4.5.2 Qualitative Results

The HOG Cascade based Gesture Recognition system is capable of recognizing all 3 hand gestures, the hand-5, hand-L, and the hand-I gestures, anywhere in an input frame. It runs in real-time with a web cam and detects and validates all 3 gestures. A video demonstration of the HOG Cascade Gesture Recognition system can be found here [193]. The system shown in the video was trained with a smaller dataset and the current trained system performs even better.

Here are a few examples of the recognition results from the current HOG Cascade Gesture Recognition system. Each pair of images show the hand gesture recognition in the input video frame, and the extracted hand shape which validated the hand gesture detection.

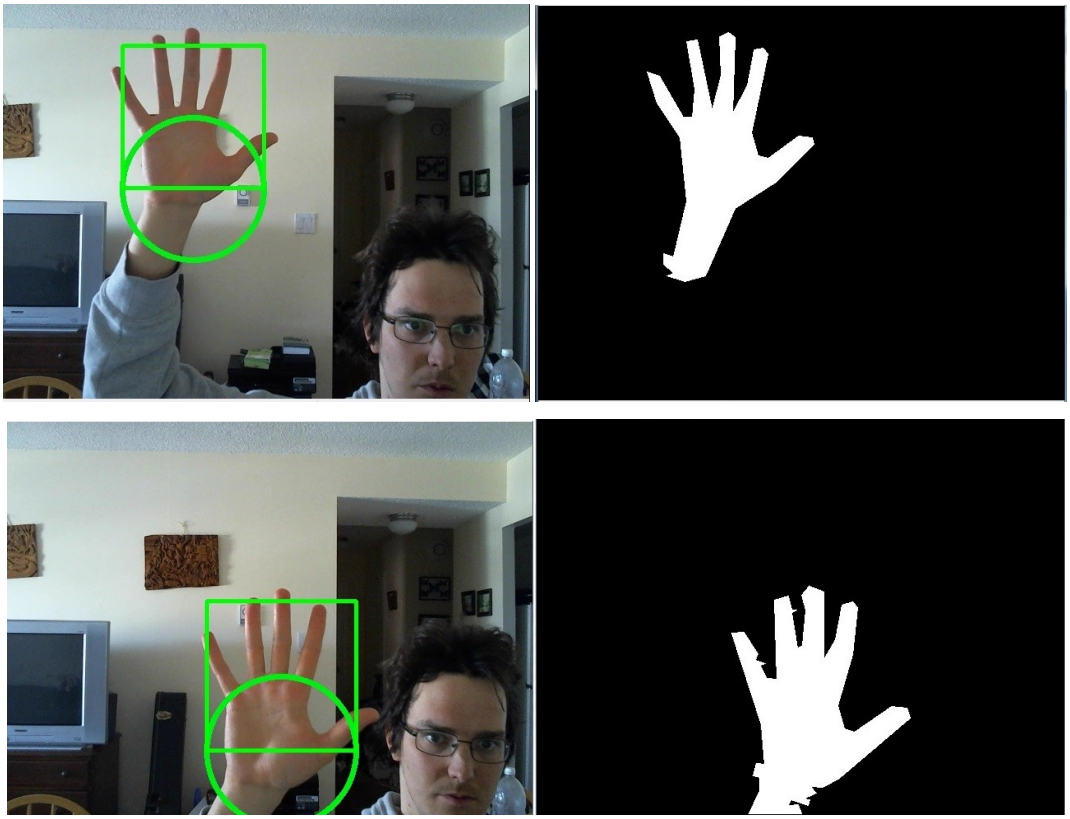


Figure 35 Successful Hand-5 Recognitions using the HOG Cascade Gesture Recognition system

The figures above show hand-5 gesture recognition results from the HOG Cascade based Gesture Recognition system. The images on the left show the hand gesture detection bounding boxes at the time of validation. The validated hand-5 gesture is shown in green. A green circle indicates the approximate location of the validated gesture, and it is centered on the sampling point which produced the successful validation. The images to the right show the extracted contours from the image segmentation.

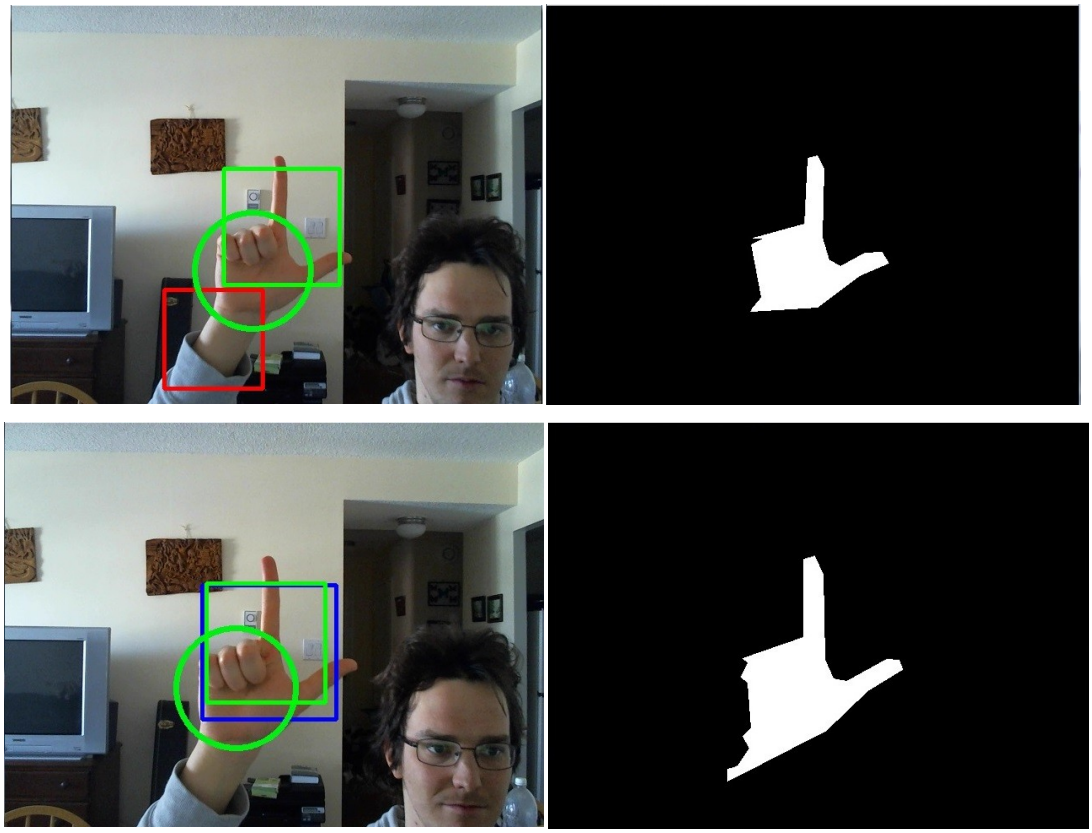


Figure 36 Successful Hand-L Recognitions using the HOG Cascade Gesture Recognition system

The figures above show hand-L gesture recognition results from the HOG Cascade based Gesture Recognition system. The images on the left show the hand gesture detection bounding boxes at the time of validation. The validated hand-L gesture detection is shown in green. Invalidated gesture detections are shown in blue for hand-5 and red for hand-L. A green circle indicates the approximate location of the validated gesture, and it is centered on the sampling point which produced the successful validation. The images to the right show the extracted contours from the image segmentation.

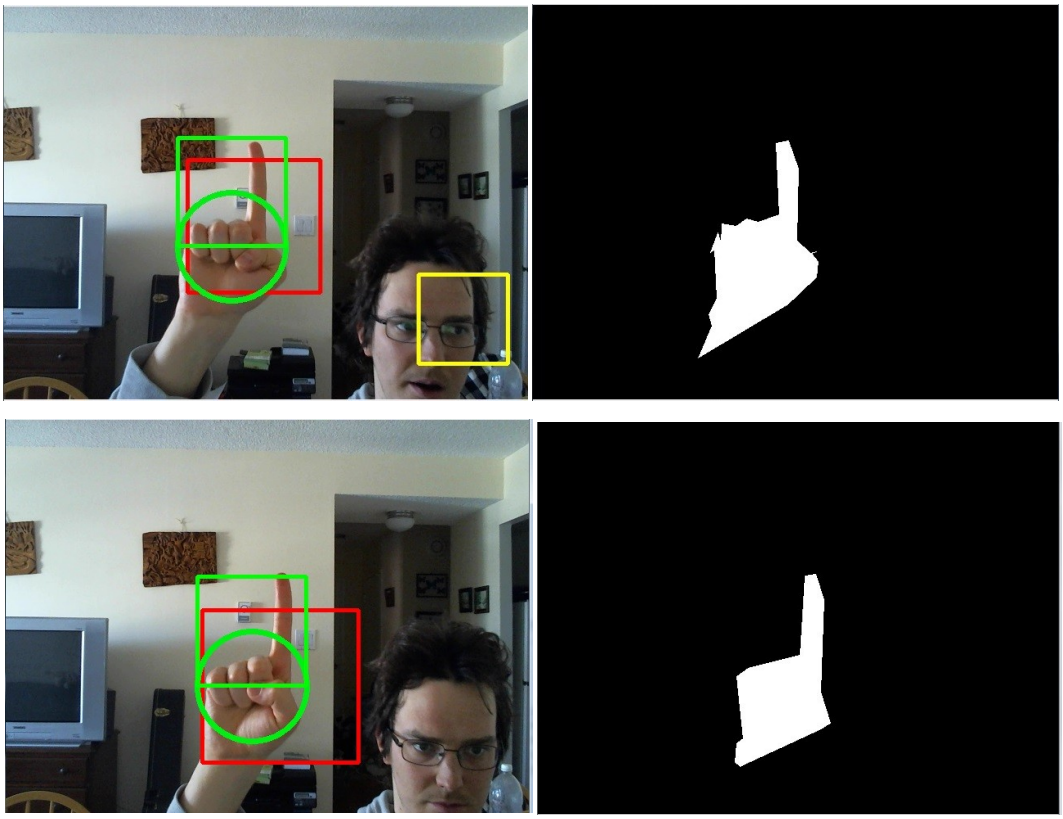


Figure 37 Successful Hand-I Recognitions using the HOG Cascade Gesture Recognition system

The figures above show hand-I gesture recognition results from the HOG Cascade based Gesture Recognition system. The images on the left show the hand gesture detection bounding boxes at the time of validation. The validated hand-I gesture detection is shown in green. Invalidated gesture detections are shown in red for hand-L, and yellow for hand-I. A green circle indicates the approximate location of the validated gesture, and it is centered on the sampling point which produced the successful validation. The images to the right show the extracted contours from the image segmentation.

The HOG SVM based Gesture Recognition system is also capable of recognizing all 3 hand gestures, the hand-5, hand-L, and the hand-I gestures, anywhere in an input frame. It works at close to real-time with a webcam. It uses a 2-stage SVM detection approach.

Here are a couple recognition results using the HOG SVM based Gesture Recognition system using an earlier 2-stage configuration of the system. This configuration used the

same 1-stage SVM twice, works well for the hand-5 gesture. Each set of images show the hand gesture recognition in the input video frame, and the extracted hand shape which validated the hand gesture detection.

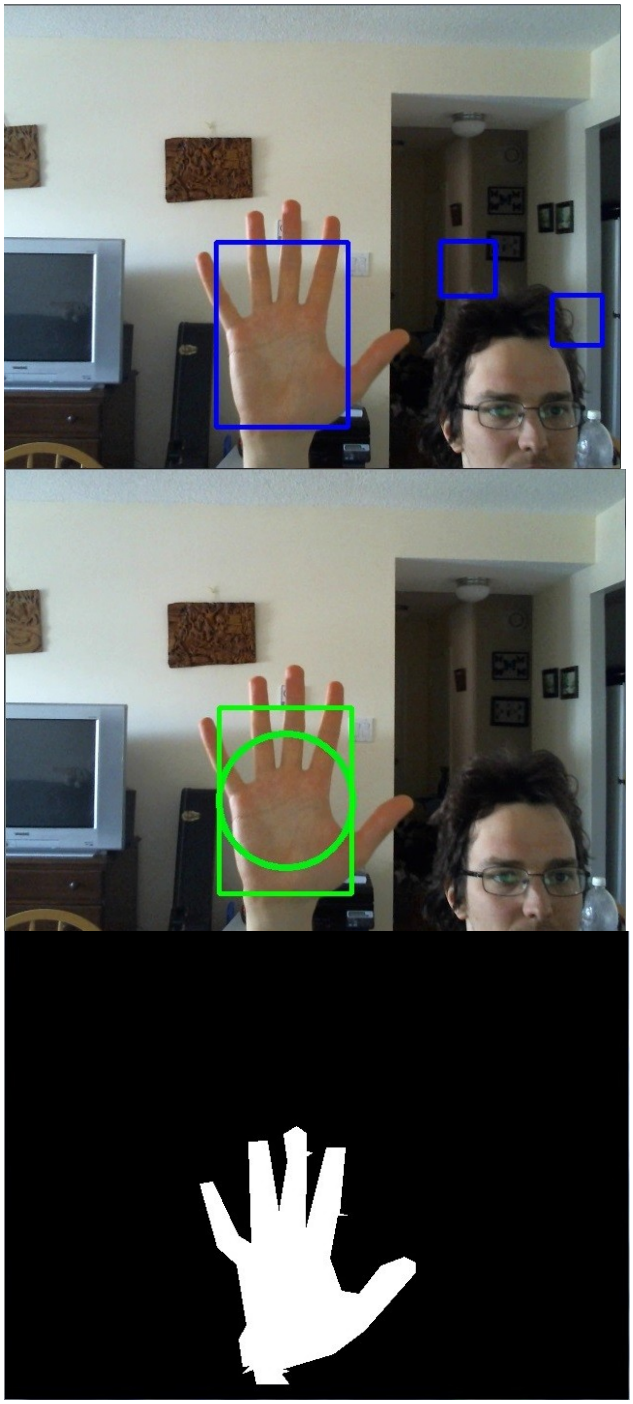


Figure 38 Hand-5 recognition using 2-stage HOG SVM Gesture Recognition system

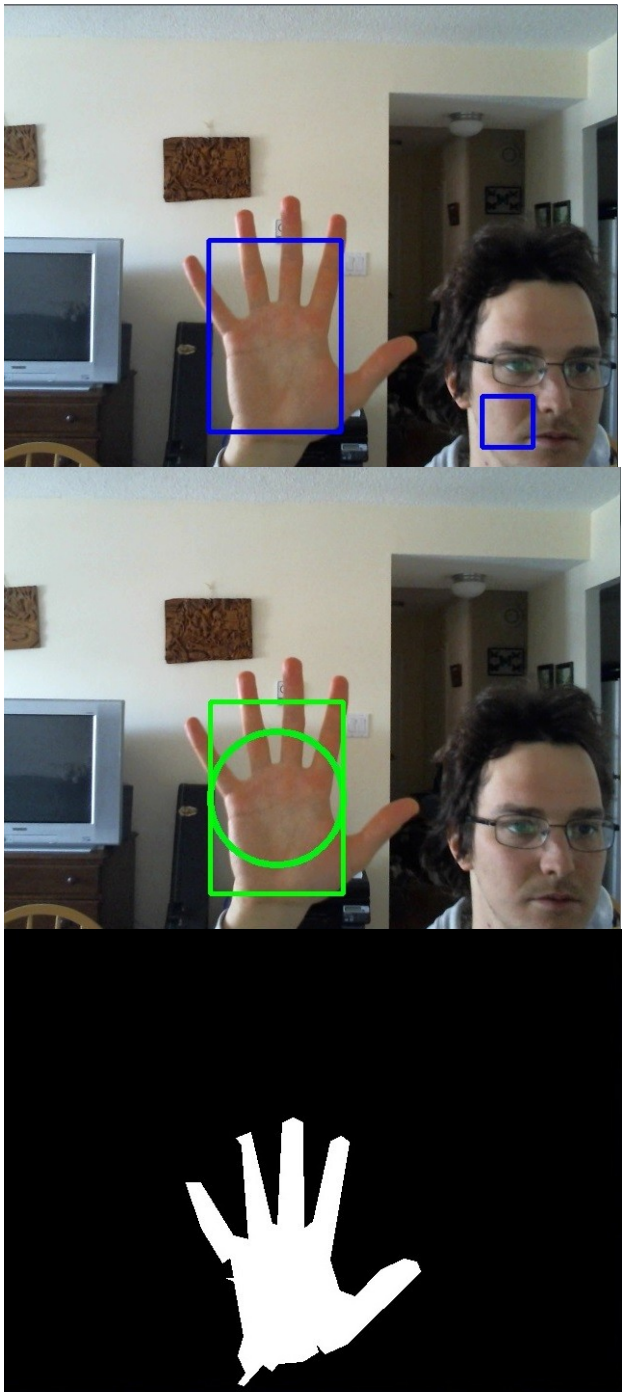


Figure 39 Another example of hand-5 recognition using the 2-stage HOG SVM Gesture Recognition system.

The figures above show hand-5 gesture recognition results from the HOG SVM based Gesture Recognition system. These results are from an older 2-stage configuration where a 1-stage SVM run twice. The first image in each set of 3 shows the hand gesture

detection bounding boxes after the first stage. The second image shows the remaining filtered bounding boxes after the second stage at the time of validation. The validated hand-5 gesture is shown in green. A green circle indicates the approximate location of the validated gesture, and it is centered on the sampling point which produced the successful validation. The third image in each set shows the extracted contour from the image segmentation. 1, 2 and 3-stage configurations have since been trained and tested online with a webcam and it has been determined that a 2-stage configuration works best.

As it can be seen from the result images, specifically the segmented hand shapes, the validation procedure works really well to validate all three gestures. It works well to extract the hand shapes from the input frames, and then to confirm that those hand shapes have the correct shape characteristics to belong to the hand gestures defined by the detections.

4.5.3 Quantitative Results

The Gesture Recognition Systems using the HOG Cascade and HOG SVM configurations have been built and tested. The testing dataset was split up by source. Part of the samples came from a medium resolution camera from a device called Raven. This part of the samples at times had poor colour separation, and was taken in a wider range of lighting. The other part of the dataset was taken from a higher resolution Logitech webcam. This part of the dataset generally had good colour separation and more stable lighting. These are referred to as Raven, and Logitech test datasets respectively. If either of these two names are not mentioned the result refers to the combined dataset.

Table 16 shows the results of the HOG Cascade Gesture Recognition system.

Detection Results Cascade System Confusion Matrix

samples\results	hand-5	hand-L	hand-I	background	total
hand-5 Raven	**281**	13	5	985	1284
hand-L Raven	0	**189**	35	602	826
hand-I Raven	0	18	**142**	685	845
hand-5 Logitech	**360**	6	2	372	740
hand-L Logitech	1	**230**	36	545	812
hand-I Logitech	0	23	**148**	775	946
background	5	54	49	**8824**	8932

Table 16 Gesture Recognition results for HOG Cascade system split by dataset

These results are split by Raven and Logitech test datasets. The confusion matrix in Table 17 shows the overall results.

Detection Results Cascade System Confusion Matrix Overall

samples\results	hand-5	hand-L	hand-I	background	total
hand-5	**641**	19	7	1357	2024
hand-L	1	**419**	71	1147	1638
hand-I	0	41	**290**	1460	1791
background	5	54	49	**8824**	8932

Table 17 Overall Gesture Recognition results for the HOG Cascade Gesture Recognition system

The best HOG Cascade Gesture Recognition system configuration uses unimodal histogram filtering for Hand-5 and Hand-L but not Hand-I gestures. Details on unimodal histogram filtering can be found earlier in this chapter.

The charts in Table 18 show the true positive and false positive results for the Gesture Recognition Cascade configuration system.

Gesture classification Cascade Configuration results

Gesture and source	% True Positives	% False Positives
hand-5 Raven	21.88	0.04716
hand-L Raven	22.88	0.7685
hand-I Raven	16.80	0.8060
hand-5 Logitech	48.65	0.05613
hand-L Logitech	28.33	0.7817
hand-I Logitech	15.64	0.8298

Gesture classification Cascade Configuration Overall Results

Gesture	% True Positives	% False Positives
hand-5	31.67	0.04854
hand-L	25.58	0.8943
hand-I	16.19	1.008

Table 18 True positive and false positive gesture recognition results for HOG Cascade system

Table 19 shows the results of the background samples for the Cascade system.

Background classification Cascade Configuration

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	8824	108	5	54	49
percent	N\A	98.79	1.209	0.05598	0.6046	0.5486

Table 19 Background classification results for HOG Cascade Gesture Recognition system

A rough estimate of the execution time of this experiment with the Cascade system was 1:44:13.

1, 2, and 3-stage SVM configurations have been trained and tested for the Gesture Recognition system. Based on online testing with a webcam, it has been determined that a 2-stage SVM configuration works best. Here are the results of the Gesture Recognition SVM configuration system using the best experimentally determined 2-stage configuration for Hand-5, Hand-L, and Hand-I gestures.

Detection Results SVM System Confusion Matrix

samples\results	hand-5	hand-L	hand-I	background	total
hand-5 Raven	**208**	29	6	1041	1284
hand-L Raven	0	**200**	17	609	826
hand-I Raven	0	18	**73**	754	845
hand-5 Logitech	**325**	8	7	400	740
hand-L Logitech	2	**136**	40	634	812
hand-I Logitech	3	42	**108**	793	946
background	4	136	295	**8497**	8932

Table 20 Gesture Recognition results for the HOG SVM system split by dataset

These results are split by Raven and Logitech test datasets. The confusion matrix in Table 21 shows the overall results.

Detection Results SVM System Confusion Matrix Overall

samples\results	hand-5	hand-L	hand-I	background	total
hand-5	**533**	37	13	1441	2024
hand-L	2	**336**	57	1243	1638
hand-I	3	60	**181**	1547	1791
background	4	136	295	**8497**	8932

Table 21 Overall Gesture Recognition results for the HOG SVM Gesture Recognition system

The charts in Table 22 show the true positive and false positive results for the Gesture Recognition SVM configuration system.

Gesture classification SVM Configuration results

Gesture and source	% True Positives	% False Positives
hand-5 Raven	16.19938	0.0377252
hand-L Raven	24.21308	1.6544616
hand-I Raven	8.639053	2.8799131
hand-5 Logitech	43.91892	0.0841908
hand-L Logitech	16.74877	1.7517423
hand-I Logitech	11.41649	3.2621137

Gesture Classification SVM Configuration Overall Results

Gesture	% True Positives	% False Positives
hand-5	26.33399	0.0728096
hand-L	20.51282	1.8278811
hand-I	10.10609	2.8982055

Table 22 True positive and false positive gesture recognition results for the HOG SVM Gesture Recognition system

Table 23 shows the results of the background samples.

Background classification SVM configuration

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	8497	435	4	136	295
percent	N\A	95.13	4.870	0.04478	1.523	3.303

Table 23 Background Classification results for the HOG SVM Gesture Recognition system

A rough estimate of the execution time of this experiment with the SVM system was 4:50:16.

True positives for both systems are calculated as the percentage of the given dataset's true samples of a given gestures that were classified as that particular gesture. False positives are calculated as the percentage of all other samples from the dataset that were classified as a particular gesture, this includes other gesture samples as well as background samples. It's important to note that the same background sample results are used as the background sample results for both the Logitech and the Raven result calculations.

MobileNets Comparison System

The MobileNets CNN[53] proposed by Google for mobile vision applications was used to compare the performance of the Gesture Recognition system to state of the art Neural Network models. The MobileNets model was taken to be an entire gesture recognition solution and so does not use any validation functions. The MobileNets comparison system was trained with a MobileNets V2 model with a window size of 128x128. The MobileNets model was used as a multiclass gesture recognition solution and it was used with a sliding window architecture. The sliding window architecture is used in order to make a very like to like comparison with the HOG SVM and HOG Cascade configurations which also use sliding windows to generate their proposals for classification. Normally CNNs use more sophisticated proposal generation to reduce the amount of window proposals generated from a frame which in turn reduces the computational load of the system and allows it to run in real-time. However in order to be able to accurately compare different classification strategies for static hand gesture recognition the same architecture must be used in order to see what kind of performance

tradeoffs exist. A proposal generator can also be applied to models other than neural networks in order to boost their speed as well.

The MobileNets model used for the comparison system was trained on all of the Hand-5, the Hand-L, and the Hand-I samples of the training dataset which corresponds to 7813 for Hand-5, 6401 for Hand-L, and 6958 for Hand-I. Out of the approximately 250000 negative background patches that have been generated from the negative background of the training dataset, 10986 were randomly selected for training the negative background class. This achieves approximately a 1.5:2 ratio with the hand-5 samples and some class balance is achieved while still benefiting from the large and diverse variety of negative backgrounds in the training dataset.

MobileNets is a neural network whose classification output for a given image is a confidence value for each class that was used for training. The performance of the MobileNets comparison system was investigated using a series of confidence thresholds to determine the best choice. The results are in Table 24.

Confidence Threshold vs. Gesture Recognition Performance for MobileNets Comparison system

Threshold	hand-5 TP %	hand-5 FP %	hand-L TP %	hand-L FP %	hand-I TP %	hand-I FP %
0.95	95.50395	6.285899	62.02686	13.58751	94.8074	14.95156
0.96	94.66403	4.400938	57.57021	10.57504	93.8024	11.75163
0.97	93.92292	2.742497	52.1978	7.374284	92.9648	9.012228
0.98	92.53953	1.577542	43.58974	4.157841	90.3964	6.106082
0.99	89.08103	0.776636	**29.9145**	**1.4984**	85.0363	3.144354
0.999	71.39328	0.121349	2.197802	0.007845	47.2362	0.317612
0.9995	**62.3024**	**0.0485**	0.671551	0	**34.618**	**0.1588**

Table 24 Performance Comparison Table of the MobileNets Comparison System. The Confidence Threshold is compared to the performance of gesture classification. The best false positive rate for each gesture with a true positive rate of 3.3% or greater is shown in bold. The best global threshold is also shown in bold.

0.99 was deemed to be the best overall confidence threshold. It achieves a low false positive rate for each gesture while maintaining a sufficiently high enough true positive rate for real-time performance of 3.3% or higher. Increasing the threshold beyond 0.99 resulted in a significant drop in true positive performance for the gestures particularly for

the hand-L gesture. The trend can be best visualized by looking at the graphs in Figure 40.

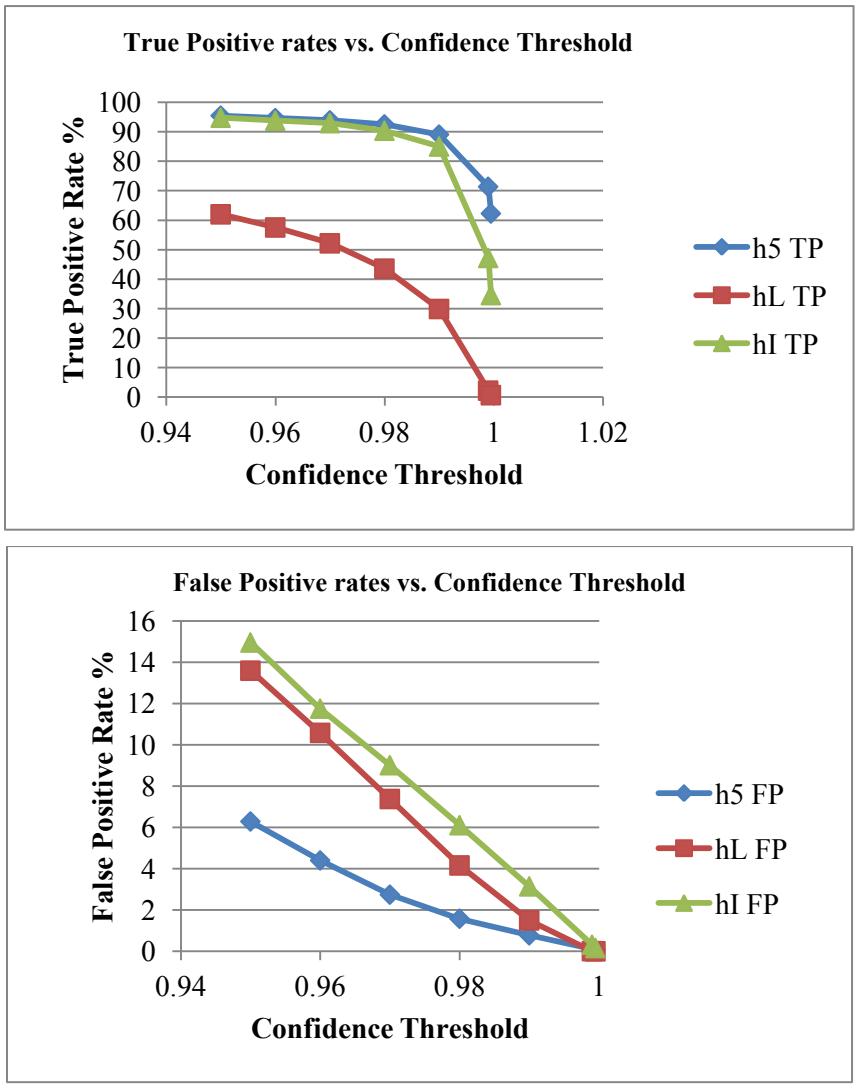


Figure 40 MobileNets Comparison System True Positive and False Positive result graphs

The results for the best global confidence threshold of 0.99 are as follows:

MobileNets Comparison System Confusion Matrix

samples\results	hand-5	hand-L	hand-I	background	total
hand-5	**1803**	13	15	217	**2024**
hand-L	4	**490**	249	1002	**1638**
hand-I	1	18	**1523**	262	**1791**
background	91	160	132	**8558**	**8932**

Table 25 MobileNets Comparison System Gesture Recognition Confusion Matrix

MobileNets Gesture Classification per dataset results

Gesture and source	% True Positives	% False Positives
hand-5 Raven	88.23988	0.87711
hand-L Raven	33.53511	1.627339
hand-I Raven	87.33728	2.209745
hand-5 Logitech	90.54054	0.879326
hand-L Logitech	26.23153	1.610473
hand-I Logitech	82.98097	2.70889

Table 26 MobileNets Comparison System Gesture Recognition Results per dataset

MobileNets Gesture Classification Overall Results

Gesture	% True Positives	% False Positives
hand-5	89.08103	0.776636
hand-L	29.91453	1.498392
hand-I	85.03629	3.144354

Table 27 MobileNets Comparison System Overall Gesture Recognition Results

MobileNets background classification

	all	classified as background (ie no detection)	falsely classified as a gesture	as hand-5	as hand-L	as hand-I
samples	8932	8558	383	91	160	132
percent	N\A	95.81281	4.287953	1.018809	1.791312	1.477833

Table 28 MobileNets Comparison System Background Classification

The MobileNets comparison system was tested on the sample Intel Core i7 PC that was used to test the Cascade and SVM configurations of the Hand Gesture Recognition system. The machine has an NVIDIA GTX 950M graphics card with a CUDA 5.0 compatibility. The runtime of testing of the MobileNets system on the testing dataset was over 90 hours.

Comparison of results

Comparing the results of the Gesture Recognition system using Cascades and using SVMs, it is seen that both systems achieved true positives of at least 10% for each gesture and very low false positives. Both systems correctly classify a very large amount

of the background samples as background, meaning that no detection was present and validated and marked as a gesture.

Gesture classification Cascade Configuration Overall Results

Gesture	% True Positives	% False Positives
hand-5	31.6699605	0.04854
hand-L	25.5799756	0.894328
hand-I	16.1920715	1.008417

Gesture Classification SVM Configuration Overall Results

Gesture	% True Positives	% False Positives
hand-5	26.33399	0.07281
hand-L	20.51282	1.827881
hand-I	10.10609	2.898206

Table 29 Gesture recognition overall results side by side comparison between Cascade and SVM systems

By comparing the overall classification results for both systems it can be seen that the hand-5 gesture is the best performer, followed by the hand-L, and lastly hand-I. This can be understood intuitively from the shape of each gesture. The hand-5 gesture with its five extended fingers is the most distinguished gesture with the most shape features, and thus it's the most easily distinguished from background objects. The hand-L gesture with its two extended fingers is less so, and lastly the hand-I is the least distinguished with only one finger, and thus the hardest to differentiate from background objects.

Both of the configurations can run at real-time or close to real time. The Cascade configuration can run at real-time. It had an approximate average runtime per frame of 434ms during testing with the test dataset. In a live on-line setting with a web camera it can run as fast as 75ms per frame or 13fps. The SVM configuration can be used as a real-time system because it only has a minor delay in terms of responsiveness when it runs in a live on-line setting with a web camera. The SVM system had an approximate average runtime of 1210ms per frame during testing with the test dataset. The system can run as fast as 350ms per frame when it runs in a live on-line setting with a web camera, or 3fps. The runtime speed of the system depends on how many detections are present and have

to be processed by the validation step. The runtime of the SVM system's experiment on the testing dataset vs. the runtime of the Cascade system's experiment is about 3 times longer. But it must be noted that both of these systems can be used as real-time systems, the SVM configuration albeit with a minor delay in responsiveness, and this has been achieved on a Intel Core i7 laptop computer without a dedicated effort for optimization. Therefore it is a safe assumption that, if optimized, the SVM system would run in real-time as well.

The criteria for success in our gesture recognition systems is that a hand gesture should be detected after being visibly presented in front of a camera for a couple of seconds, this combined with a very low false positive rate would make a robust system. Narrowing that done to 1 second means that in a real-time system running at 30 frames per second, a minimum of 1 frame with a hand gesture out of 30 should be detected to meet this requirement. Which corresponds to a true positive rate of 3.3%. It can be seen that this has been achieved for each gesture in each configuration. A low false positive rate is also important for a robust system. The overall false positive rates are 1.2% for the Cascade system, and 4.9% for the SVM system respectively. While the Cascade system outperforms the SVM system, both systems achieve very low false positive rates. It is important to note the false positive rates can be reduced further by adding data to the dataset that represents more users' hands, as well as trains for application specific background samples for specific backgrounds, such as an indoor office, a train station, or a museum.

Two systems that can be used online in real-time with a webcam have been successfully built and tested.

The MobileNets comparison system also achieves good performance using its best global confidence threshold of 0.99. It is seen that all 3 gestures have low false positive rates and very high true positive rates of 29.9% or higher.

MobileNets Gesture Classification Overall Results

Gesture	% True Positives	% False Positives
hand-5	89.08103	0.776636
hand-L	29.91453	1.498392
hand-I	85.03629	3.144354

Table 30 MobileNets Comparison System Overall Gesture Recognition Results

However both the Cascade and SVM configurations of the Gesture Recognition system in general achieve lower false positive rates than MobileNets using its best overall confidence threshold. Both Cascade and SVM systems achieve lower false positive rates for the Hand-5 and Hand-I gestures. The cascade system even achieves a lower false positive rate for the Hand-L gesture. The SVM configuration trails the hand-L false positive performance by only 0.3%.

The MobileNets system can outperform both systems in terms of a low false positive rate while having a true positive rate of 3.3% or higher for Hand-5 and Hand-I gestures if separate confidence thresholds are used for each gesture. These are highlighted in bold in Table 24. MobileNets is able to reach the very low false positive rates of the hand-5 gesture of the Cascade system while having a much larger true positive rate. However the hand-L true positive performance quickly drops with confidence thresholds higher than 0.99 and it reaches lower than 3.3% making it unusable for a real-time system. Linear interpolation was used to see the performance of gesture classification of the MobileNets system at the same false positive rate as the HOG Cascade and HOG SVM hand gesture recognition systems.

	Cascade System		**MobileNets Best Individual Thresholds System**			**MobileNets 0.99 Best Global Threshold System**	
Gesture	% TP	% FP	% TP	% FP	Confidence Threshold Necessary	% TP	% FP
hand-5	31.67	0.04854	**62.3**	0.04854	0.9995	89.08	0.7766
hand-L	**25.58**	0.8943	18.68	0.8943	0.9936	29.91	1.498
hand-I	16.19	1.008	**56.47**	1.008	0.9968	85.04	3.144
mean Precision	82.41		**87.08**			82.09	
mean Recall	24.48		45.82			**68.01**	
mean Accuracy	89.92		92.77			**94.62**	

	SVM System		**MobileNets Best Individual Thresholds System**			**MobileNets 0.99 Best Global Threshold System**	
Gesture	% TP	% FP	% TP	% FP	Confidence Threshold Necessary	% TP	% FP
hand-5	26.33	0.07281	**65.33**	0.07281	0.9993	89.08	0.7766
hand-L	20.51	1.828	**31.61**	1.828	0.9888	29.91	1.498
hand-I	10.11	2.898	**81.74**	2.898	0.9908	85.04	3.144
mean Precision	63.51		**82.78**			82.09	
mean Recall	21.69		59.56			**68.01**	
mean Accuracy	89.73		93.61			**94.62**	

	Cascade System	**SVM System**	**MobileNets System**
Approximate total runtime on test dataset	1 h 44min	4 h 50 min	94 h 4 min
Approximate average runtime per frame	434ms	1210ms	23 541ms

Table 31 Comparison table for Cascade and SVM systems to MobileNets

Linear interpolation revealed that the 0.89 % false positive rate for the hand-L gesture can be approximately achieved at 0.99365 confidence threshold for the Cascade system. However the corresponding true positive rate of 18.7% for the Hand-L gesture with the MobileNets system is lower than the 25.6% rate of Cascade system. This means that the Cascade Hand Gesture Recognition system beats the MobileNets comparison system in terms of Hand-L recognition performance. It can also be seen that in order to achieve the low false positive rates of both systems very high confidence thresholds need to be used. This highlights the robustness of the HOG Cascade and HOG SVM configurations and shows that both systems perform well when compared to the best threshold configuration of the MobileNets comparison system. The Cascade system is even able to beat the MobileNets model in some cases.

The runtime of the MobileNets system is significantly greater for the test dataset than the Cascade or the SVM configurations of the Gesture Recognition system. Using the same sliding window strategy for all three systems the Cascade system runs the testing experiment in 2 hours, the SVM system in 5, whereas there is an over 90 hour runtime for MobileNets. This corresponds to an approximate average runtime per frame of 23 541ms. Whereas the Cascade and SVM systems both were responsive enough to run in a live on line setting, the MobileNets comparison system is too slow. That is not to say that MobileNets cannot run in real-time. A CNN can run in real-time with a proposal generator which allows it to skip evaluating many windows at different scales of an input frame. The 90 hour runtime of MobileNets shows clearly its dependence on using a proposal generator to run in real-time.

Using a sliding window strategy with all the systems makes for a very good comparison. It is clear that MobileNets is more accurate in terms of classification performance. However this accuracy comes at a huge computational trade off, and this accuracy is not absolute as the Hand-L gesture results have shown. Furthermore a proposal generator can be used to speed up either a cascade or an SVM model in the same way it speeds up neural networks. This further illustrates the need for like to like comparisons between different machine learning models. It is important to see the overall accuracy with a given model using the same windowing strategy as its competitors in order to assess the benefits and drawbacks in a clearer light.

A consideration must be made for the accuracy of the runtimes. When running the Cascade and SVM systems on the testing dataset sometimes the systems would error at a random sample in the middle of a test set run. It was always a different sample. The issue seemed to occur when too many samples would get processed too quickly. The issue was possible due to a hardware limitation on the testing computer, such as a buffer being overloaded, or circuitry overheating. This error could not be replicated consistently. The only way around it was that if it occurred during a testing run of a given Cascade or SVM configuration, was to pause testing for 30 seconds to a minute several times in the middle of the test run. As a result the times for the whole test dataset processing could be too large by as much as 5-10 minutes for the Cascade and the SVM system. This has little

effect on the comparisons made because the differences in runtimes of the systems are at least an order of magnitude larger than this 5-10 time addition. At other times slowdowns occurred due to system processes starting or stopping. This affected all 3 systems. As such the testing times printed in this thesis are approximate. They serve as a useful comparison between systems but cannot be taken as exact. Exact measurements on a given computer architecture would require a more finely controlled setup which would require at least one machine to run the experiment while a separate computer monitors the results. This kind of setup would be instructive for benchmarking purposes but it is beyond the scope of this thesis.

To conclude the analysis of the results, both configurations of the Gesture Recognition system perform well relative to the MobileNets comparison system. The SVM Gesture Recognition system is a close contender to the 0.99 best global confidence threshold MobileNets system, in terms of low false positive performance and at least a 3.3% true positive rate. The Cascade Gesture Recognition system beats the 0.99 MobileNets system outright because it has lower false positive rates for all 3 gestures. Using separate thresholds the MobileNets system can beat the SVM system for all gestures, and it can beat the Cascade system for 2 out of 3 gestures. However the Cascade system beats MobileNets in terms of Hand-L performance.

A real-time hand gesture recognition system is successfully implemented in 2 different configuration with one configuration running in real-time and the second being usable in a real-time live user scenario despite running close to real-time. The Gesture Recognition system performs well compared to a state of the art neural network deep learning approach even beating it in some cases.

4.6 Summary

A real-time hand gesture recognition system was presented in this chapter. The Gesture Recognition system was implemented using HOG Cascades. The system is capable of detecting and validating the 3 classes of gestures in real-time from anywhere in a video input frame. The Gesture Recognition system was also implemented using HOG SVMs. It was implemented with 1, 2, and 3-stage SVMs. It was determined that a 2-stage SVM

configuration works best. The system is capable of detecting and validating the 3 classes of gestures in close to real-time from anywhere in a video frame. Close to real-time in this context means that the system can detect the 3 gestures in real-time with a small delay in video output responsiveness. Both Gesture Recognition system configurations, the Cascade based system, and SVM based system, work with a web camera to recognize hand gestures in real-time, and close to real-time, respectively.

A large dataset was collected for hand gesture recognition. It has between 8000 - 10000 samples for the three gestures used by the gesture system, and a significant amount of samples for several other gestures. It also has approximately 44 000 negative images, covering indoor environments and people. This dataset enabled the training of the system. It is an excellent opportunity for comparative research work for 2d vision only hand gesture recognition because current datasets have too few samples or a publically unavailable or lack pose annotations for the hand samples which make them only suitable for hand detection and not multi class hand gesture recognition.

Validation functions were implemented which validate the hand gesture detections in the Gesture Recognition system. The validation functions of the Gesture Recognition system validate gestures using the Open Hand model, the General Hand model, and Shape Signature Analysis. The General Hand model is also used by the hand tracking in the Gesture Recognition with Hand Tracking system. Both hand models and the Shape Signature Analysis are also used by the Gesture Recognition with Colour Clustering system.

Numerical testing of the performance of the Gesture Recognition system has been done. The testing was performed on both the best HOG Cascade configuration and the best HOG SVM configuration. The Gesture Recognition systems were tested on their recognition performance, with and without their validation functions. Testing was also done to determine the effect on performance of unimodal histogram filtering in both systems. The results show that validation functions are important for reducing the false positive rate for the Cascade and SVM based gesture recognition system. The functions keep the false positive rate low while maintaining a sufficiently high true positive rate for

real-time hand gesture recognition. These functions are crucial for making the Gesture Recognition system viable for user interface applications.

A performance comparison was made of the 2 Gesture Recognition system configurations to MobileNets, which is a competing Convoluted Neural Network model for vision applications. The MobileNets model was successfully trained and implemented with a sliding window architecture allowing for comparison to the Cascade and SVM systems which also use sliding windows. Results show that MobileNets models perform very well, however this performance comes at a much heavier computational cost, and there is an obvious trade off between speed and performance. MobileNets models can run in real-time however this requires additional proposal generation architecture. Proposal generation can also be used to speed up the Cascade and SVM based Gesture Recognition configurations. The comparison is important because it shows the performance of the different classification methods when using the same sliding window proposal strategy. The computationally lighter Cascade and SVM models overcome the short comings of false positive detections with the validation functions and perform well compared to MobileNets models, even able to outperform them in some cases. Cascade and SVM configuration false positive rate performance is consistent with MobileNets models using very high confidence thresholds and the much lighter computational cost give these configurations a significant advantage in real-time hand gesture recognition where resources may be limited, and having a lower true positive rate with real-time speeds still results in a responsive system very desirable for user interfaces.

Chapter 5 Gesture Recognition with Hand Tracking

5.1 Overview

In the previous chapter a Hand Gesture Recognition system was presented that was capable of recognizing static hand gestures in live real-time situations. This chapter will show how the Hand Gesture Recognition system can be combined with hand tracking. The Gesture Recognition system provides a good initial registration of the user's hand which is then used to initialize robust and rabid hand tracking. This combined Hand Gesture Recognition with Tracking system achieves automatic real-time tracking of a user's hand which is suitable for user interface applications. The tracking uses adaptive colour segmentation as well as contour processing and template matching. The tracking performance of this system is evaluated using a dataset of videos of users going through a series of motions with their hands.

The Gesture Recognition with Hand Tracking Algorithm is a complete recognition and tracking system for human hands. This system will be able to recognize and track a user's hand anywhere in a video frame in real-time using a web camera. It offers many options for hand gesture based controls for user interfaces. Although this system has yet to be implemented, its two major components, which are hand gesture recognition and hand tracking, have been successfully implemented and are performing well in real-time.

The Gesture Recognition with Hand Tracking algorithm is an extension of the Gesture Recognition system presented in the previous chapter. The algorithm uses the Gesture Recognition algorithm to find a user's hand in a frame from a video input feed, and then the colour and shape information of the hand is used to initialize the hand tracking portion of the algorithm.

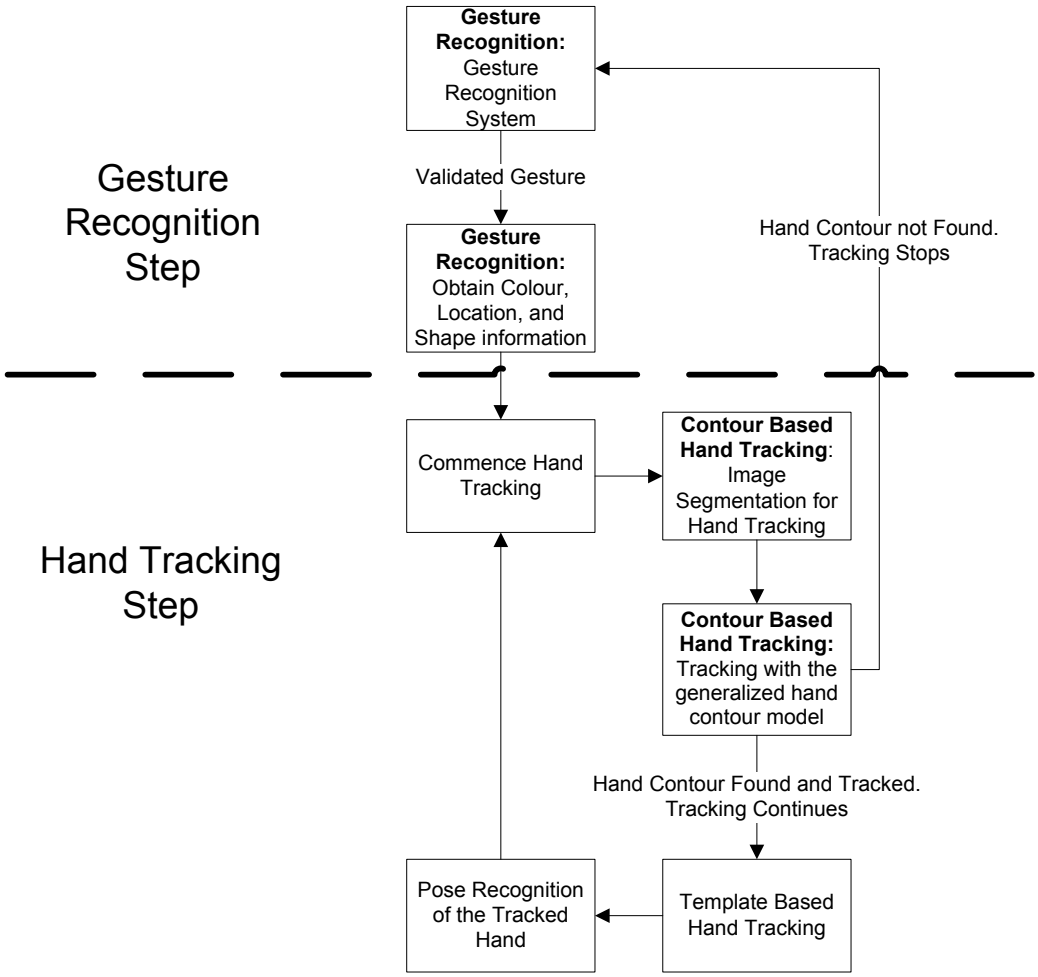


Figure 41 Gesture Recognition and Hand Tracking Flow Chart

The hand tracking is done using two different and complementary tracking strategies. The primary way to track the user's hand is by using the hand colour information to track the hand's contour in subsequent frames of the video feed. This is called contour based hand tracking. It is done by using the same general hand model presented in the Gesture Recognition algorithm chapter, and imposing a distance constraint to allow a hand contour to move within a certain range between successive frames. The contour based hand tracking controls the hand tracking part of the Gesture Recognition with Hand

Tracking system. Tracking of a user's hand continues so long as the contour based hand tracking is able to track the hand, otherwise the system reverts back to recognizing gestures with the Gesture Recognition part of the system.

The secondary way to track the user's hand is through its fingertips. Template based tracking of the fingertips of the extended visible fingers of the user's hand is used in order to provide stable tracking of the fingertips. While robust and fast, the contour based hand tracking provides fingertip locations that tend to jitter between frames which is due to fluctuations in the border of the fingertip contours. Template based tracking provides stable fingertip locations between successive frames which enables the Gesture Recognition with Hand Tracking system to be used as a user interface. Template matching used by the template based tracking is a computationally costly affair, but this problem is overcome by using the less stable contour based tracking fingertip locations to provide the template based tracking with small search regions. By refining the search regions in this way the template based tracking strategy is able to work in real-time providing stable tracking of the user's fingertips.

The last part of the hand tracking is pose recognition of the tracked hand. While this is not a hand tracking strategy it provides useful information on the static pose of the tracked user's hand if one is recognized. Using a series of HOG Cascades this part of the hand tracking analyzes the static pose of the tracked user's hand and returns one of 5 poses if the pose has been successfully recognized. Pose recognition is a useful capacity of the hand tracking part of the Gesture Recognition with Hand Tracking system, allowing for user interface commands to be mapped to static poses of a tracked user's hand.

It is useful to examine the relationships between the different components of the Gesture Recognition and Hand Tracking system. The dependencies between the components make the role of each component more defined. The relationships are described in the chart in Figure 42.

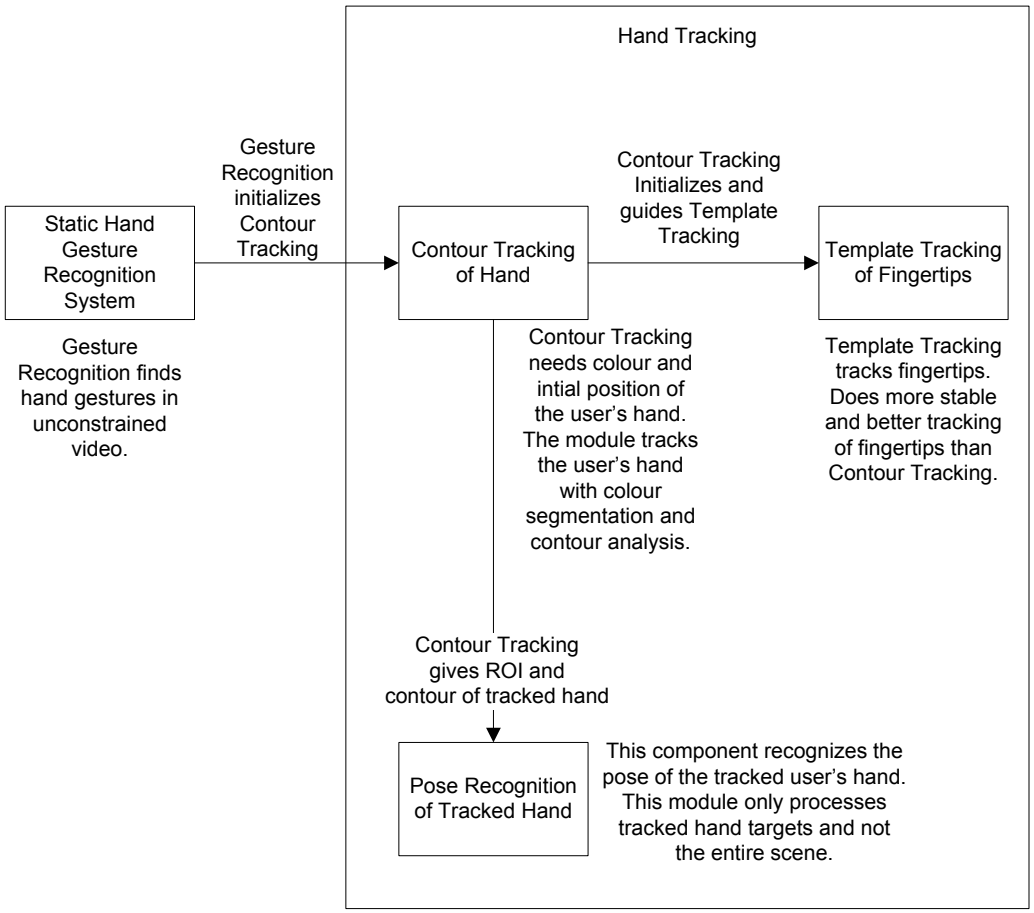


Figure 42 Gesture Recognition with Hand Tracking system component relationship diagram

The gesture recognition component finds hand gestures in unconstrained video. This initializes the hand tracking. Specifically the gesture recognition system initializes the contour tracking of the user's hand. It does so by providing initial location of the hand and the hand's colour signature which is used for subsequent segmentation and tracking. Contour tracking requires this initialization in order to track the user's hand. The colour signature allows it to segment the user's hand from the scene. The initial hand location allows it to discriminate between multiple contours if multiple contours are produced and the initial hand location also allows contour tracking to enforce realistic movement between successive frames. At the end of each tracking iteration this initial hand location is updated.

Contour tracking governs the entire hand tracking algorithm. Contour tracking initializes and guides the template tracking of the user's fingertips. Initialization is done by signaling how many user fingertips there are and where they are approximately located. The template tracking responds by providing untracked fingertips (untracked by template tracking) with a template tracker. The template trackers are capable of better tracking the locations of fingertips than contour based tracking. Noise along contour boundaries result in jittery fingertip locations in successive frames. Template tracking on the other hand provides a more stable sum of least squares tracking location which produces smooth and stable tracking of the fingertips from frame to frame. This template matching can be computationally costly, so the fingertip locations of tracked fingers from the contour tracking are used to delimit smaller search regions for the template trackers. This allows the template tracking to run in real-time.

Contour tracking also calls the pose recognition of the tracked hand. It provides the pose recognition with an ROI of the user's hand in the video frame, as well as the contour of the user's hand. The user's hand while it is tracked by the contour tracking, can have a specific static pose. Knowing this pose is of interest because it can be used to issue commands in user interfaces. The pose recognition component takes the hand image obtained from the ROI, and rectifies it using the hand contour. This rectified image is then run through a series of HOG Cascades in order to determine if it represents a static hand pose. While the pose recognition component has a similar function to the gesture recognition component there is a significant difference in the situations where these are used. Gesture recognition is used in unconstrained video in order to find a hand in a specific static pose. These gestures can be found anywhere in the video and prior hand location information is not available. Pose recognition on the other hand is used when hand location information is already available. This simplifies the classification problem and allows a greater variety of static poses to be recognized. This means that under typical operation negative background images do not have to be processed by the pose recognition. The pose recognition classification generally operates only on hand images. Additionally the hand gestures that appear in both the gesture recognition system and the pose recognition component have better recognition results in the pose recognition

component because of the reduction of the problem due to hand location information being available. This is why running classification of the pose on the tracked hand is of interest, because of the UI options that come from recognizing the pose of the hand, and because of the better recognition performance.

5.2 Gesture Recognition

The Gesture Recognition step is done with the Gesture Recognition system described in the previous chapter. The Gesture Recognition step is run on the input frames when the Gesture Recognition with Hand Tracking algorithm is run, until a hand gesture is recognized. This step recognizes hand gestures anywhere in the frame. The information gathered from the validated gesture detections is used to initialize the hand tracking.

If a specific hand gesture is recognized, that is to say detected and then validated, then the colour histogram, the hand contour, the extended visible fingertips, and the palm location are provided to the hand tracking. All of this information can already be requested from the Gesture Recognition system.

After a successful gesture recognition the hand tracking initializes and takes over. Gestures are not recognized by the Gesture Recognition system when the hand tracking is active. The hand tracking tracks the user's hand for as long as it appears, and tracking is successful, in subsequent frames from the video feed. If the hand tracking loses track of the hand then the Gesture Recognition system resumes scanning input frames for hand gestures.

5.3 Contour Based Hand Tracking: Image Segmentation for Hand Tracking

When the hand tracking takes over from the Gesture Recognition system it tracks the user's hand in subsequent frames so long as it continues to be successfully found by the Contour Based Hand Tracking in each frame. Otherwise control reverts back to the Gesture Recognition system to find any new user hand gestures. Control can revert back to the Gesture Recognition system if there is no hand contour found within a reasonable

distance of the previous tracked hand location or if the user's palm touches the border of the frame. These exit conditions will be explained later.

When hand tracking is performed on a frame, it must segment the input frame with the 2d colour histogram provided by the hand gesture validation that initiated the hand tracking. This segmentation is used to then track the user's hand using the generalized hand contour model. This is explained in the next section.

Segmentation is the first step in the hand tracking portion of the algorithm. It is done for every frame as long as tracking continues. It always uses the histogram provided by the initial gesture detection.

5.4 Contour Based Hand Tracking: Tracking with the generalized hand contour model

Tracking is an important problem in hand gesture recognition. It is different from the gesture detection or hand detection problem because once tracking is initialized it has a previous location that needs to be updated in the current frame of a video input. More generally it is different from the detection problem because there is continuity from one frame to the next. Successful detections mean detecting hands in a frame. Knowing which hand object is the one that was tracked in the previous frames among potentially several hand detections is the challenge with tracking. The tracking problem in this system is solved primarily by contour based hand tracking, which applies a nearest neighbour type strategy with a geometric hand contour model.

The tracking algorithm calls the generalized hand contour model when tracking the contour in the current frame. As explained earlier the generalized hand contour model is a series of geometric tests that is applied to a contour to determine its palm location and count any visible extended fingers. The tracking calls the model with a binary segmentation image and the palm center of the tracked contour from the previous frame.

The model is called with the constraint that a given contour is only valid if its palm center does not exceed a certain distance from the previous palm center point.

The constraint is as follows:

distance from previous palm point < *reference radius range* * original radius (5)

The previous palm point is the palm center point of the tracked contour in the previous frame. It is the saved location of the palm center point of the tracked contour either from the previous frame, or from the validated gesture detection which initiated tracking. The distance from previous palm point is the distance between the previous palm point and the palm point of the given analyzed contour in the current frame. The original radius is the saved palm radius of the validated gesture detection which initiated tracking. Finally the *reference radius range* is a value defined in the general hand contour model which enlarges the range of valid distance values relative to the original radius.

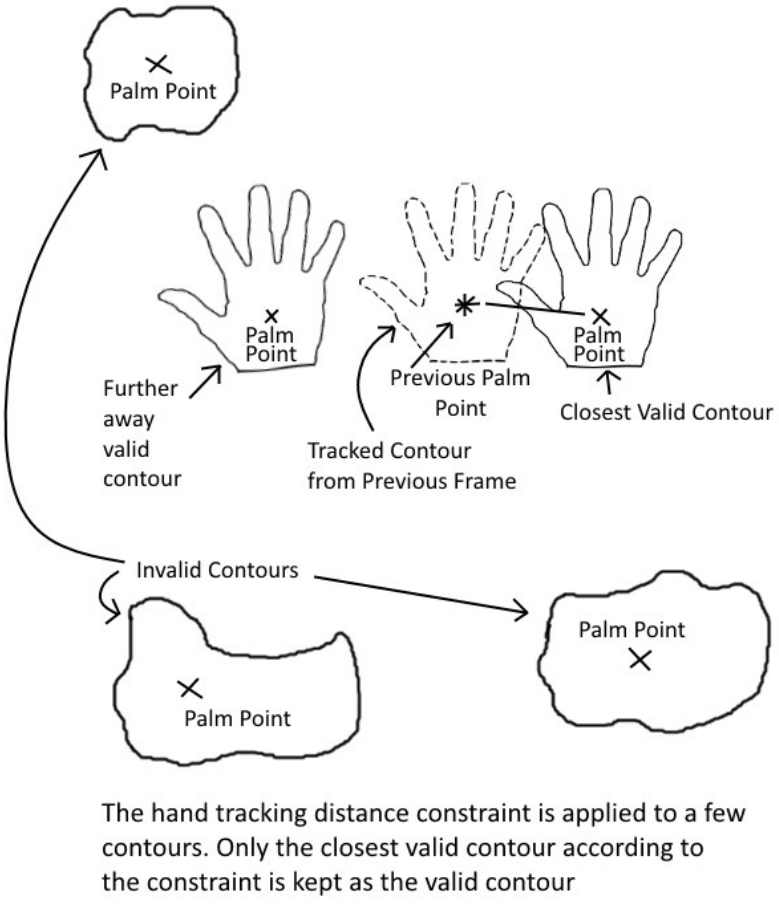


Figure 43 Finding the closest valid contour.

This constraint is imposed so that a tracked hand and its contour can move from frame to frame and it will continue to be tracked. The tracked contour in the current frame does not have to overlap with the area covered by the contour in the previous frame. It just has to meet the condition imposed by the distance constraint. Figure 43 shows an example of applying the distant constraint from equation (5). The distance between the palm points found with the general hand model is found between each respective palm point and the previous palm point. The distant constraint is applied and only some contours pass its criteria. Among those contours only the closest one is kept according to its distance.

Thus every contour found in the binary segmentation image is either validated or invalidated by the general hand contour model based on this constraint and the tracking keeps the closest valid contour. The distance between a given valid contour's palm center and the previous palm center is used to determine which is the closest. The previous tracked palm center location information is updated with the palm center location information of the closest valid contour for use in the next frame and the closest valid contour becomes the tracked contour. If no valid contour is found the tracking stops. If a tracked contour palm touches the frame border the tracking also stops because its assumed that the user voluntarily ended the hand tracking by taking their hand out of view. As is explained in the General Hand Contour model a contour may also be invalidated if it's too small. Tracking continues as long as the hand contour target is found in the frame, otherwise control reverts back to the Gesture Recognition system.

When a contour is successfully tracked in a frame, the contour based tracking gets the fingertip locations of the visible extended fingers from the general hand model, in addition to the palm location, and the contour itself. The locations of the fingertips of the extended fingers are used to initialize and then guide the template trackers in the template tracking phase of the tracking algorithm. The contour based tracking also feeds the hand contour and the original input frame to the Pose Recognition component of the tracking algorithm.

5.5 Template Based Hand Tracking

This algorithm takes as inputs the original input image, the CrCb histogram of the tracked hand contour, the palm position, and the locations of the fingertips of the tracked hand contour. The template tracking outputs the locations of the tracked fingertips which have been tracked with template trackers. These template tracked fingertip locations are significantly more stable than their contour based counterparts. In general these fingertip locations do not jitter or move when the user's hand stays reasonably still, and this increased stability, is why these template tracked fingertips are desirable for gesture recognition user interfaces. This template tracking algorithm is only called when there is a tracked hand contour in the current frame.

5.5.1 High Level View:

Figure 44 Template Tracking Flow Chartshows the steps of the template tracking of the user’s fingertips.

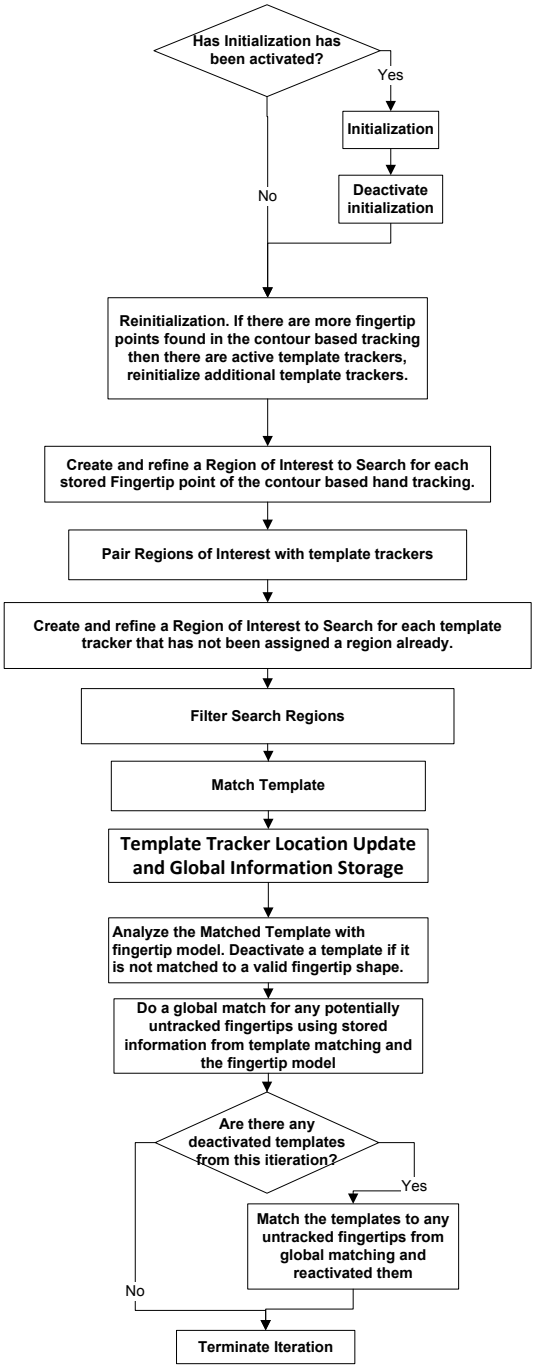


Figure 44 Template Tracking Flow Chart

5.5.2 Initialization

When the algorithm is run for the first frame of hand tracking it starts with the CrCb histogram of the hand and the initial position of all of the hand's detected fingers, as well as the position of the palm. The initialization runs in the first frame and then that phase of

the algorithm is deactivated. The contour based tracking can reactivate the initialization when there are more contour tracked fingertips than templates tracking them.

In the initialization phase the following should be done for each finger to be tracked:

Select the area in the input image around each initial position. This becomes the finger template. Save the initial position as the finger template start point. The size of the area around each finger tip is determined by the distance between the first fingertip and the palm center divided by the *palm finger divisor*, this is referred to as the saved template width, and it is used as the height and the width of the area. The first fingertip's distance to the center of the palm is sufficient to define the saved template width, however the largest fingertip to palm center distance can also be used for a more rigorous approach. Values in italicized font can be found in the table at the end of this section.

Convert the finger template image selected into greyscale. Take the original color finger template and convert it to YCrCb. This will be the YCrCb finger template. Take the back projection of the YCrCb template. Take the biggest contour found in the back projection and put it in a separate image. This becomes the back projection finger template. Dilate the back projection 2 times to give a wider border. Calculate the size of the aforementioned biggest contour and calculate the percent area coverage of the finger template.

Save the finger contour size and the percent area coverage for use in the subsequent tracking.

a) b) c)

Figure 45 Selection of a template to track: a) Initial Selected Template, b) Mask created from the back projection of the CrCb histogram unto the YCrCb colour version of a), c) Cropped finger template used for tracking

Mask the greyscale and YCrCb finger templates using the back projection to produce the cropped greyscale and cropped YCrCb finger templates respectively. Save the cropped greyscale finger template as the finger template to search for in subsequent frames.

At the end of the initialization, the cropped greyscale finger template, the finger template start point,

the finger contour area, and the percent area coverage should be saved for each finger. The given tracked hand's CrCb histogram used for back projection is also saved, it will be used for narrowing the search of the greyscale finger templates in subsequent frames.

5.5.3 Reinitialization

If the initialization has already happened in a previous frame, and there are more contour fingertip points than active trackers, then additional trackers are initialized and added to the active trackers. This process is called reinitialization, and counts as a special case of the initialization phase.

The given contour fingertip points are put in a collection. The points by the nature of the algorithms that produced them are ordered either clockwise or counter clockwise relative to the center of the tracked hand contour. The active trackers are also put into a collection, which is unordered. Then the first active tracker is taken and the closest ordered point is found, and then both are removed from their collections. This continues until the active tracker collection is empty.

The remaining contour fingertip points are used to initialize new template trackers in exactly the same way as in the regular initialization phase.

5.5.4 Tracking

Everything past the initialization phase is considered to be part of the tracking. The tracking phase creates a copy of the input frame and converts it to greyscale. This greyscale input image is used for the search regions and the template matching. The frame is also converted to the YCrCb colour space and this YCrCb input image is used in the global matching step.

Create and refine a Region of Interest to Search using the provided fingertip points of the contour based tracking

From each contour tracking finger point in the current image, select a square region of interest (ROI) centered on the contour finger point and its dimensions should be the width of the saved greyscale finger template multiplied by the *finger template ROI multiplier*.

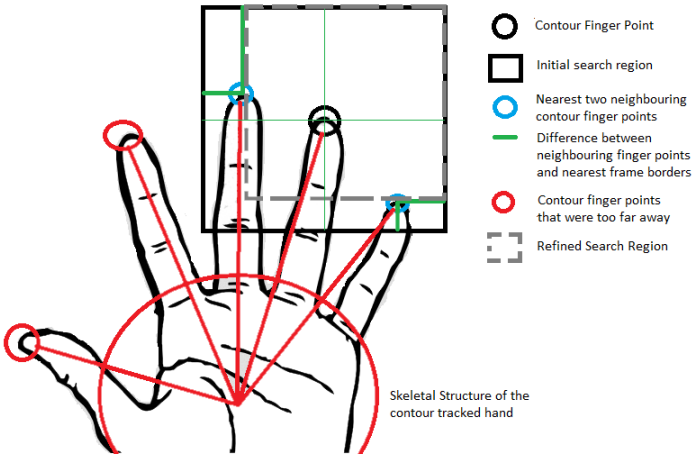


Figure 46 Region of Interest Resizing using contour tracked hand

The next step is to resize the ROI. Take the two nearest adjacent contour finger points and determine whether they fall inside the initial ROI. For each point that does fall in this region find the nearest border for the X dimension and the nearest border for the Y dimension. Calculate the X and Y dimension differences.

The smaller of the 2 differences for each neighbouring point within the initial ROI will be used to trim the initial ROI to remove that point. For example if a neighbouring point is in the top left quadrant of the initial ROI and its X difference is the smallest, then the X dimensions of the initial ROI will be modified to remove the range of values that the X difference represents. This can also be seen as shifting the nearest border to a neighbouring point over, such that it overlaps with that particular point.

This resizing is done to create search regions guided by the contour tracked hand from the contour based tracking. These search regions are centered on one of the fingertips of the currently tracked hand and they do not overlap with neighbouring fingertips. This aids

the template trackers to find the best match by providing them with good search regions. The more stable template trackers can find and match each finger to its finger template without jittering between frames, which is a drawback of the contour based tracking.

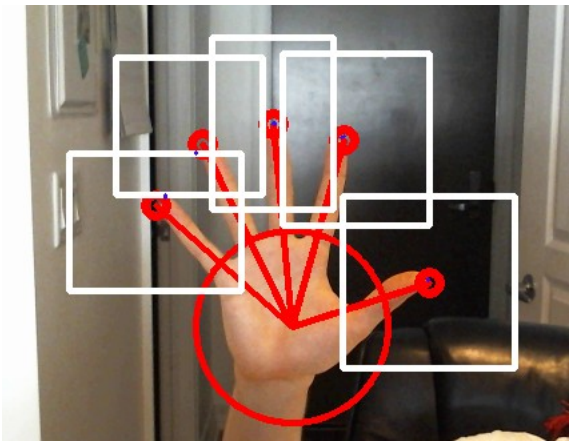


Figure 47 Results of search region trimming using contour tracked hand

The above figure shows the results of creating search regions using the contour tracked hand while ensuring that a given search region does not contain more than 1 finger point. Some of the search regions are resized from their default size, such as the middle finger's region, to exclude their neighbours. Only one clear finger is visible within each region which improves the tracking results.

Match the Regions of Interest to the fingers

Once the Regions of Interest have been created from the contour tracked hand these regions must be matched to the tracking templates using the positions of the tracking templates in the previous frame and the contour finger points. A line of best fit must be calculated for the contour finger points using linear regression. This line is then used to find its intersections with the edge of the frame. One of these intersections is taken as the anchor point. Although it doesn't matter which intersection is selected the implementations of the tracking algorithm for this system use the leftmost point. Figure 48 shows an example of finding the anchor point.

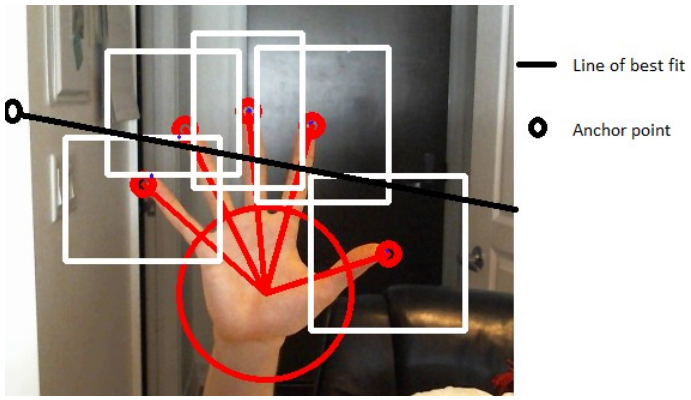


Figure 48 Finding an anchor point for a set of contour finger points and their respective search regions.

Due to the template tracker reinitialization step called by the contour based tracking, there are never less active template trackers than contour finger points at the start of a tracking iteration of the template tracking algorithm. This means there are two cases where template trackers must be matched to the search regions of contour finger points, one case where there are an equal number of contour finger points to active template trackers, and one case where there are less contour finger points.

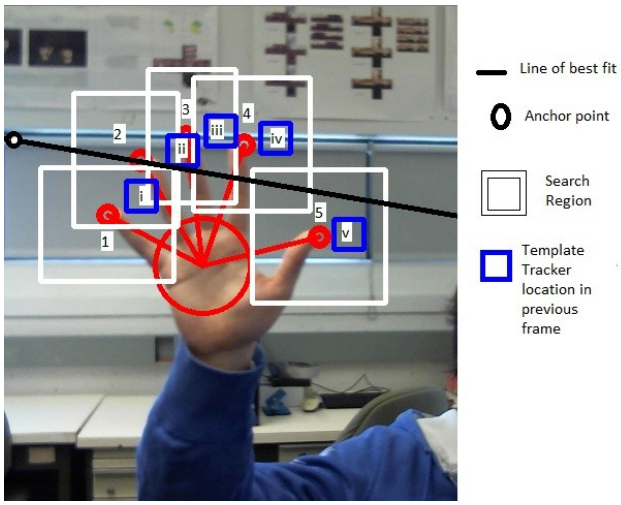


Figure 49 Matching case where there are an equal number of contour finger points to active template trackers. Matches are labeled with decimal numbered search regions matched to the template tracker with the same value roman numeral.

When there are an equal number of contour finger points to active template trackers both are ordered according to the anchor point with the closest member at the front of the queue. The template trackers use their location in the previous frame, or their initialized

location if they where inactive in the previous frame. Each active template tracker is assigned a search region according to their place in the queue.

The closest template tracker is assigned the search region of the closest contour finger point, the second closest is assigned the second closest contour finger point's search region and so on. Figure 49 illustrates the process. Each template tracker is assigned the search region of the finger contour point whose decimal number matches the template tracker's roman numeral.

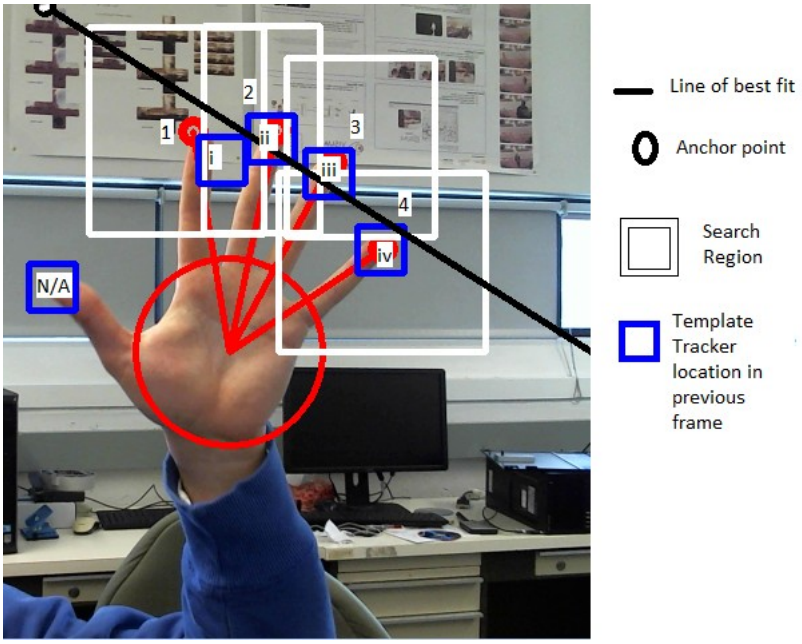


Figure 50 Matching case where there are fewer contour finger points to active template trackers. Matches are labeled with decimal numbered search regions matched to the template tracker with the same value roman numeral.

When there are fewer contour finger points to active template trackers a decision must be made as to which of the active template trackers will be matched to a search region and which will not. The contour finger points are again ordered by proximity to the anchor point, and the template trackers are all put into an unordered collection. Then in order of nearest proximity with the closest contour finger point first, the corresponding search region is assigned to the active template tracker which is closest to the given contour finger point, and then that tracker is removed from the unordered collection of active

template trackers. Thus only the template trackers closest to the contour finger points are assigned search regions.

Create and refine a Region of Interest to Search For each finger template tracker that does not already have a search region

The active template trackers that do not already have a search region assigned to them are each processed with a different set of steps to create their search regions. From the finger template start point in the current image, select a square region of interest (ROI) centered on the finger template start point and its dimensions should be the width of the corresponding saved greyscale finger template multiplied by the *finger template ROI multiplier*.

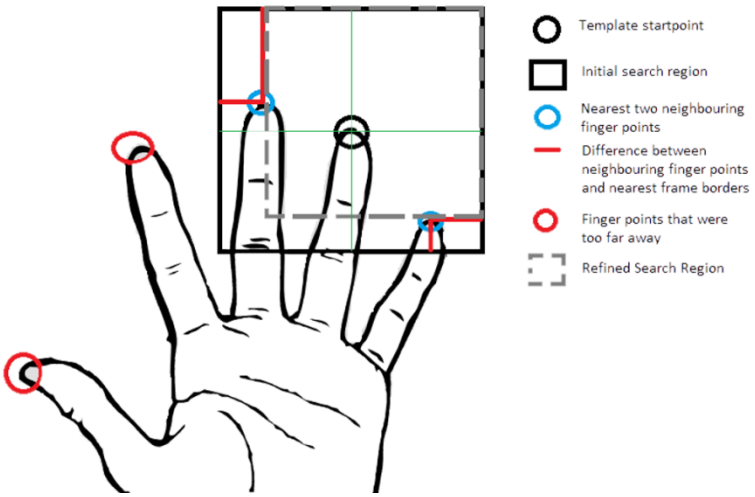


Figure 51 Region of Interest Resizing

The next step is to resize the ROI. Take the two nearest adjacent finger template start points and determine whether they fall inside the initial ROI. For each point that does fall in this region find the nearest border for the X dimension and the nearest border for the Y dimension. Calculate the X and Y dimension differences. The smaller of the 2 differences for each neighbouring point within the initial ROI will be used to trim the initial ROI to remove that point.

For example if a neighbouring point is in the top left quadrant of the initial ROI and its X difference is the smallest, then the X dimensions of the initial ROI will be modified to

remove the range of values that the X difference represents. This can also be seen as shifting the nearest border to a neighbouring point over, such that it overlaps with that particular point. This resizing is done to remove as much as possible the presence of neighbouring fingertips. The resizing helps the template matching algorithm make matches only to the desired fingertip. Without the refinement step the templates tend to "jump" from one fingertip to another in between frames.

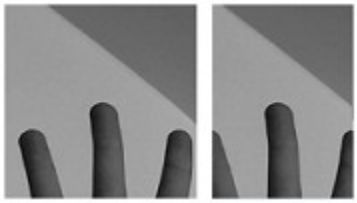


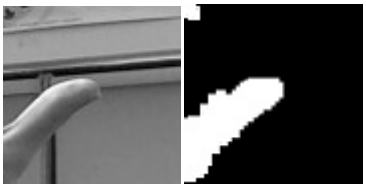
Figure 52 Results of search region trimming

The above figure shows the effect of trimming the initial ROI. Only one clear finger is visible which improves the tracking results.

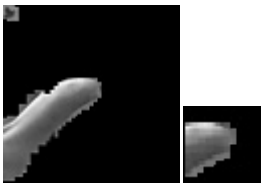
Filtering the search regions:

Once a region of interest has been created using either of the two preceding steps it must then be filtered to prepare for the template matching.

Take the ROI, convert one copy of it to YCrCb, and convert another to greyscale. From the YCrCb ROI create a back projection using the tracked hand's CrCb histogram. Dilate the back projection twice; this will be called the ROI mask. Mask the greyscale ROI with the ROI mask to create the cropped ROI.



a) b)



c) d)

Figure 53 Template Matching: a) Initial Search Region (After Resizing), b) Mask created from the back projection of the fingertip's CrCb histogram with the YCrCb colour version of a), c) Cropped Search Region, d) Template to match

Match Template for each finger tip and store global information about the match locations and candidates:

This part of the template based hand tracking algorithm uses the normalized squared difference template matching algorithm in the OpenCV library with the greyscale template and the cropped ROI.

The way that the normalized squared difference template matching algorithm works is that it compares pixel values between a mono channel, in this case grey scale, template image and a mono channel search image. It runs a window of the same size as the template across the entire search image. At each coordinate of the search image it places the search window with its top left corner on that particular coordinate. It then takes the pixels located within the window and subtracts them from the pixels located within the template image respective of the position. These differences are squared and then summed, and then recorded in a result image at same pixel coordinate where the window was placed.

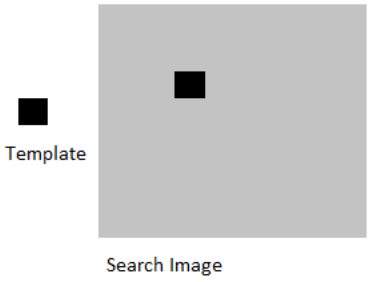


Figure 54 Template and Search Image Example

Once all the squared differences have been calculated the algorithm finds the largest squared difference and normalizes all the values in the result image by this difference. The result is an image like the one seen in Figure 55, where the smallest value indicated by the black dot represents the best candidate location for the template to be located in the search image.

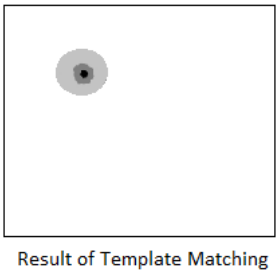


Figure 55 Result of Template Matching

Running template matching on large images is very costly computationally. Therefore the search regions are kept small, and one small region is made for each tracked template rather than one large region for all templates. This allows the algorithm to run in real time.

Template Tracker Location Update and Global Information Storage:

When the normalized squared difference match template algorithm is complete find the matched point in the resulting image, which under the conditions described is the

minimum intensity location of the result image produced by the algorithm. Use the matched point location from the result image to update the finger template start point.

Sometimes a search region can yield more than one good candidate point. That is why the resulting image from every search region is pulled together to form a template matching result image for the entire frame. This will allow the template tracking algorithm to later perform a match of any unused good candidate points to templates that were deactivated in the analysis of the matched template step. This means that if one template's search region yields a poor match and the template is deactivated, it can be re activated using good results from another search region that yielded two or more good fingertip candidate locations.

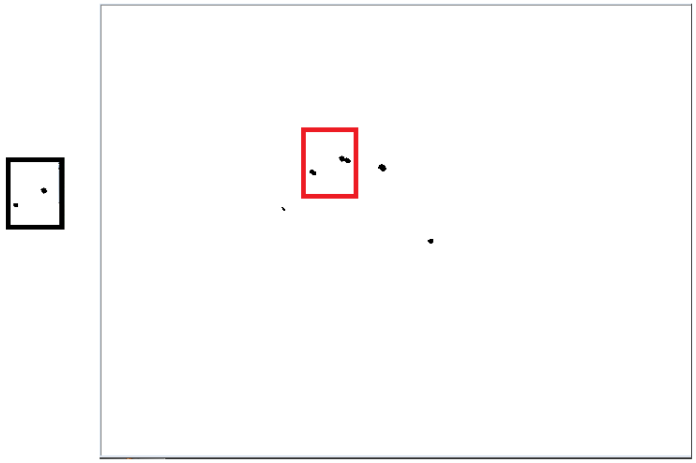


Figure 56 Example of combining multiple template matching result images into one image. Multiple result images such as the one outlined in the black box are combined using a minimum value operator. The red box indicates the location of the black boxed result image in the combined result image for the frame.

The result image for the entire frame is created by taking a white background and performing a minimum value operation with the white background and each of the positional referenced template matching result images. All of the result images are floating point values and they are normalized. Therefore 0.0 represents black and 1.0 represents white. This becomes the global match result image.

Analyzing the Matched Template and applying the fingertip model:

For each template and its result image from the template matching algorithm it is important to determine if the matched point corresponds to a good fingertip shape in the corresponding search ROI.

Going back to the ROI mask create a new image by selecting from the ROI mask the region centered around the updated finger template start point (the matched point that was used to update after the template matching), and let it be the same size as the finger template. This image will be called the matched template mass mask. Take the biggest contour found in the matched template mass mask and find its area (current contour area). From this calculate the percentage of the mass mask image that it covers, this percent coverage variable will be used to determine if a finger shape is valid.

For a given template the biggest contour found in the matched template mass mask is also analyzed with a fingertip shape model. If the contour passes the criteria established by the model, the template tracker and its tracked location in the frame will remain active, otherwise it will be deactivated.

The fingertip shape model loosely defines a valid full fingertip contour as having between 0 and 12 flat sections of points and between 0 and 4 long flat sections of points. A flat section is defined as 3 connected contour points that make an angle between the *min flat section angle* and the *max flat section angle*. And a long flat section is defined as 4 or more connected contour points where each of the points with the exception of the end points make an angle with their neighbours that is between the *min flat section angle* and the *max flat section angle*.

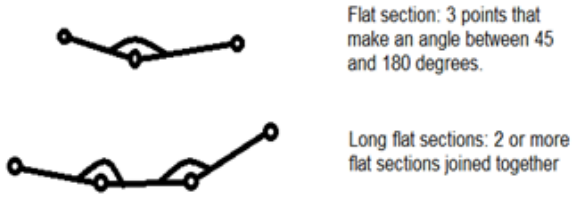


Figure 57 Definitions used in the determining of the presence of a full fingertip in a matched template's potential finger contour.

It is important to note that "flat" is a loosely used term in the aforementioned definitions. A potential fingertip contour is evaluated by going through each point along the contour as well as its immediate neighbours and measuring their angles and recording the number of flat sections and long flat sections.

A potential fingertip contour is also analyzed for the amount of times and the way that it touches the borders of the matched template mass mask. A potential fingertip is only allowed to touch a maximum of two different borders of the matched template mass mask and if two borders are touched they cannot be located on opposite sides of one another. Here are some examples of valid and invalid border touching conditions:

Figure 58 Valid Border touches for a fingertip contour

Figure 59 Invalid border touches for a fingertip contour

Notice how the contours with invalid border contact have very little or strange curvature and generally do not resemble fingertips. A fingertip contour also must have a percent coverage between *min percent coverage* and *max percent coverage* in order to be considered valid.

Global Matching

The last step of the template tracking algorithm is to take the global matching information on all of the remaining good candidate locations for fingertips and match those locations to any of the templates that were deactivated during thus far in the individual search region fingertip tracking in the current frame. This step does not initialize template trackers that were not active at the start of a given tracking iteration of the template tracking algorithm. Because fingertips in greyscale images often look very similar it is acceptable to reactivate a deactivated template tracker to a different finger than the one it was originally tracking, just as long as only one template tracker tracks one fingertip.

All of the good fingertip locations are found from the global matching information that was gathered by combining the individual search regions. This is done by iteratively finding the minimum value in the global match result image, and then once it is found flood fill that location in the result image with white (float value 1.0), and then repeat the process until the minimum value found is greater than 0.5. This ensures that all of the black dots representing the good candidate locations for fingertips are found.

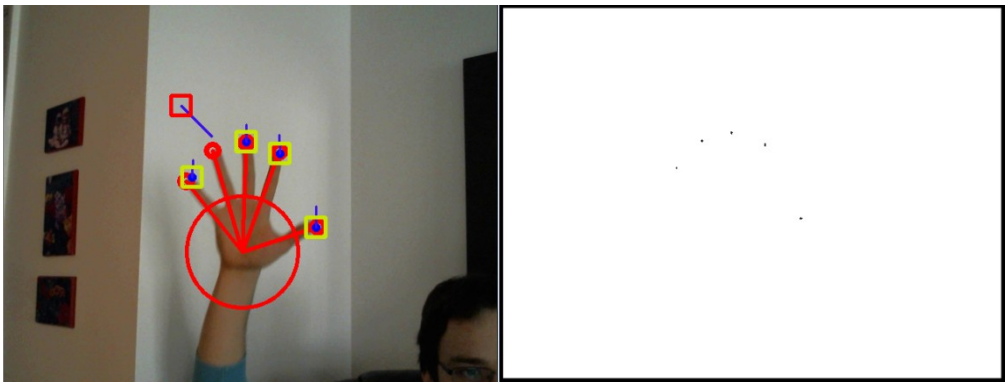


Figure 60 An example of global matching. The deactivate template in red on the left will be matched to a location found in the right image that is not already tracked.

Once all of the good candidate points are found, the points that fall within active template trackers are removed. The locations of the remaining points are recorded, and are each evaluated to determine if they actually represent the location of a fingertip. This is done by taking the YCrCb input image and creating a back projection from it and then using the back projection to find potential fingertip shapes at the remaining candidate point locations. The immediate areas of the back projection around the candidate points are taken as fingertip masks. These areas are squares whose dimensions are equal to the widths of the saved templates of the template trackers defined in the initialization phase. The biggest contours within the fingertip masks are found. These contours are analyzed using the same fingertip model as in the previous step.

If the fingertip model analysis validates any remaining candidate points as valid fingertip points then those locations are used to re activate any deactivated template trackers that were deactivated during the individual matches. The greyscale input image and the

fingertip masks of the valid fingertip points are used to create new greyscale fingertip templates for the template trackers to track in the next frame. The start locations of these trackers are also updated with the appropriate validated remaining candidate points. If any deactivated trackers remain after the global matching then their data is discarded and they are marked as uninitialized, ready to be initialized again by data received from the contour based tracking in future iterations.

Once the global matching step is completed the successfully tracked fingertips are outlined in a copy of the input frame in dark yellow squares. Red squares are used to indicate the last location of deactivated templates. These red squares only remain for one frame.

Once completed for the given input frame, the template tracking algorithm returns the locations of the template tracked fingertip points. The template squares can also be returned if needed.

Table 32 Template Based Hand Tracking Algorithm Values

Algorithm parameter	**Value**
palm finger divisor	4.0
finger template ROI multiplier	3.0
min flat section angle	180°/4.0
max flat section angle	180°
min percent coverage	1.5
max percent coverage	25.0

5.6 Pose Recognition of the tracked hand

Pose Recognition takes the tracked hand contour and the original input frame as input and outputs the name of the recognized pose of the tracked hand if one is recognized. There are 5 recognized hand gesture poses, hand-5, hand-L, hand-I, thumb out, and fist.

1. The first step is to get the tracked hand contour which was provided as input.

Figure 61 Hand Contour of Tracked Hand

2. The wrist of the tracked hand contour is filtered out in the same way as is done in the Shape Signature Analysis. Find the distance transform of the contour. This requires the filled in contour to be drawn upon a binary image, and then that binary image is used to find the distance transform. Then find the brightest point in the result image. This point is otherwise known as the center of mass.

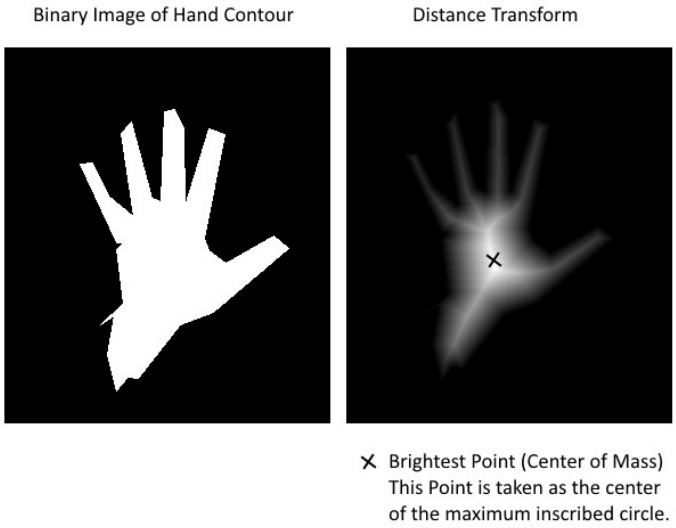


Figure 62 Finding the Center of Mass of Tracked Hand

Then filter out the wrist points of the hand contour. Measure the angle of each point along the contour relative to this center of mass. Then the points that fall within 15 degrees of the downward facing normal are filtered out of the contour. This is done to filter out the hand contour's wrist so that it doesn't impact the hand contour rectification.

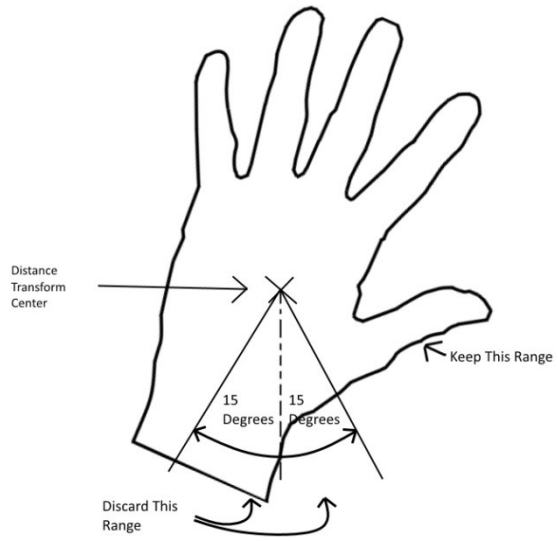


Figure 63 Wrist filtering of tracked hand contour

It is important to note that this 15 degree range within the downward facing normal is an approximation of the wrist location. Other parts of the hand contour, specifically parts of its palm can also be filtered out with this process and that is acceptable.

3. Find the average of all the filtered contour points. The axis formed by this point and the distance transform center will be referred to as the principle axis of the hand contour in this section. The principle axis will be used to rotate the filtered hand contour later.

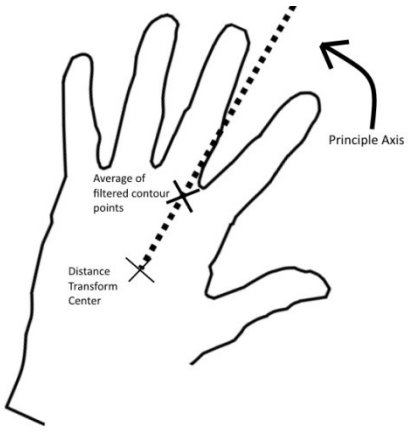


Figure 64 PCA of Tracked Hand Contour

4. Now that the hand contour has had its wrist filtered out and its principle axis orientation is known, find bounding box of the filtered hand contour.

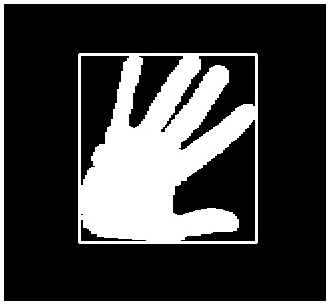


Figure 65 Bounding box of contour

5. Then expand it by multiplying all dimensions by the square root of 2.

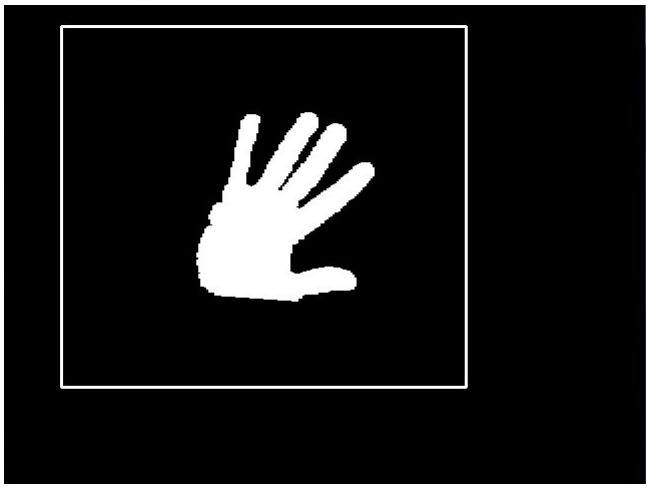


Figure 66 Expanded bounding box

6. Take a crop of the input image defined by this expanded bounding box. (Note: The HOG cascades take greyscale image as input so the input image can already be converted to greyscale at this point.)

Figure 67 Expanded bounding box on top of greyscale input image

7. Draw the filtered hand contour on a binary map the size of the expanded bounding box.

8. Rotate both the expanded bounding box binary map and expanded bounding box image crop in accordance with the angle of the principle axis, such that both are rectified.

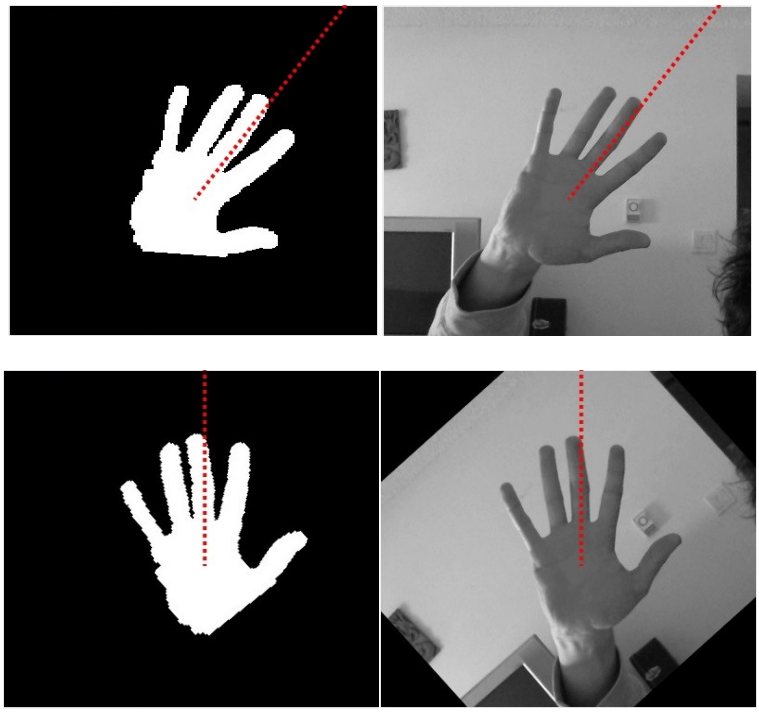


Figure 68 Rotation of the region of interest defined by expanded bounding box of segmentation and input image

9. Find the bounding box of the now rectified hand contour in the rotated expanded bounding box binary map. Use it to crop out the rectified hand image from the rotated expanded bounding box image crop. In practice the bounding box of the rectified hand contour should be enlarged by a few pixels in each direction to ensure a good crop of the hand with a safety margin.

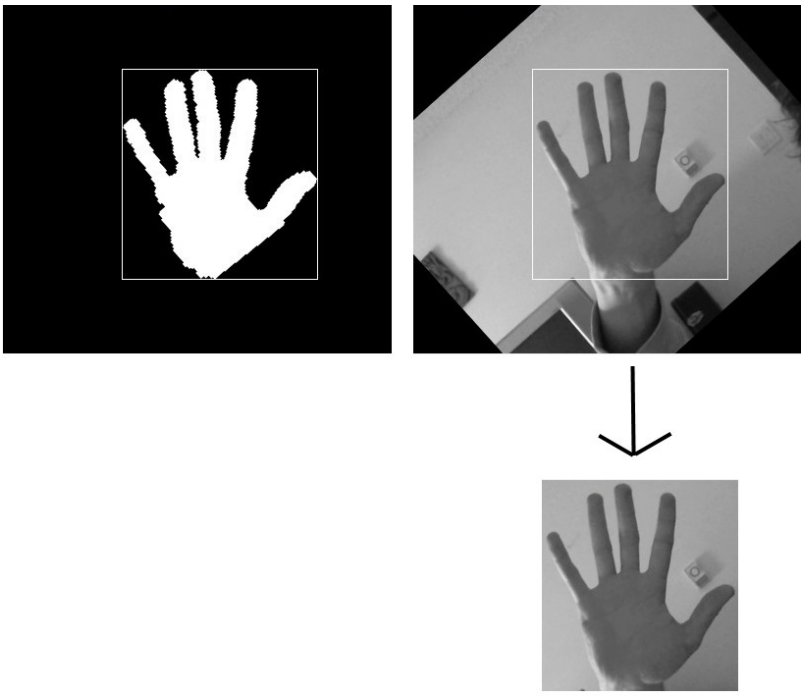


Figure 69 Finding bounding box of upright segmentation and using it to crop upright hand

10. The last step is the pose classification step. Take the rectified hand image and feed it through the Pose Recognition HOG cascades. These are a series of 5 HOG cascades trained to recognize the static pose of the tracked hand if it's one of the 5 trained poses.

The 5 trained poses are Hand-5, Hand-L, Hand-I, Thumb Out, and Fist.

Figure 70 5 hand gestures used by the pose recognition

The cascades are all trained using the following structure:

Given Hand Gesture positive samples vs. the other 4 Hand Gestures and the negative backgrounds as negative samples for all 5 of the hand gestures.

The rectified hand image is run through each of these 5 Pose Recognition HOG cascades. If only one of the cascades produces detections then the tracked hand is declared to be of that pose. If none, or more than one cascade produce detections then the pose of the tracked hand is not classified because it is not successfully recognized.

Figure 71 shows a visual representation of the pose classification step using the Pose Recognition HOG cascades.

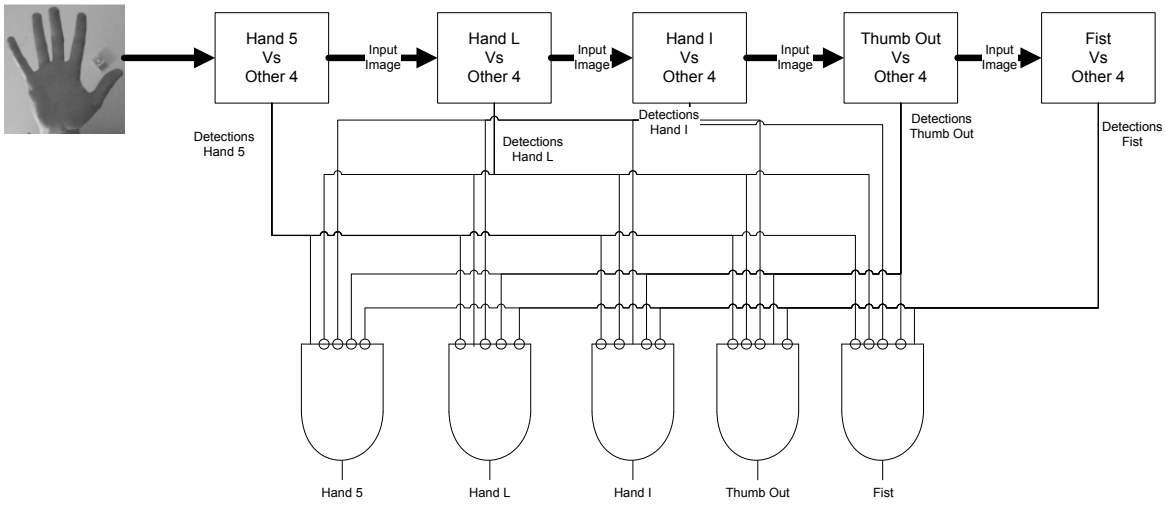


Figure 71 Pose Recognition Diagram

If the Pose Recognition cascades successfully recognize the pose of the tracked hand then the name of that pose is returned by the Pose Recognition algorithm.

It is interesting to note that thumb out and fist were not used as recognized gestures in the Gesture Recognition system presented in chapter 3. This is because they were deemed not distinct enough to be reliably distinguished from the other 3 gestures in the more unconstraint problem of detecting hand gestures in an input video frame which requires gestures to be distinguished from background. The problem for Pose Recognition is simpler because it only requires one gesture to be distinguished from the others, so all 5 are used.

The training of the pose recognition HOG Cascades was done with 7813 Hand-5 images, 6401 Hand-L Images, 6958 Hand-I images, 6650 Hand fist images, and 6834 thumb out images, and 35482 negative backgrounds. This was done with the same training dataset that was used to train the Hand Gesture Recognition system presented in Chapter 3.

5.7 Results

5.7.1 Qualitative Results

The Gesture Recognition system has been implemented as explained in the previous chapter. Here is a video of an earlier HOG Cascade configuration [193]. As mentioned earlier it can detect and validate hand gestures anywhere in a video frame. This system was connected to the Hand Tracking algorithm to create the Gesture Recognition with Hand Tracking system. Only the hand-5 gesture was used to recognize and register the user's hand and to commence hand tracking. This is because it had a lower false positive rate and because colour signatures generated from that gesture as a consequence were better suited for segmentation in subsequent tracking. A video demonstration of the Hand Gesture Recognition with Hand Tracking system can be found here [185] which includes a demonstration of the Pose Recognition component. The video shows the successfully implemented system running in real-time.

Here are some images showing the operation of the Gesture Recognition with Hand Tracking system.

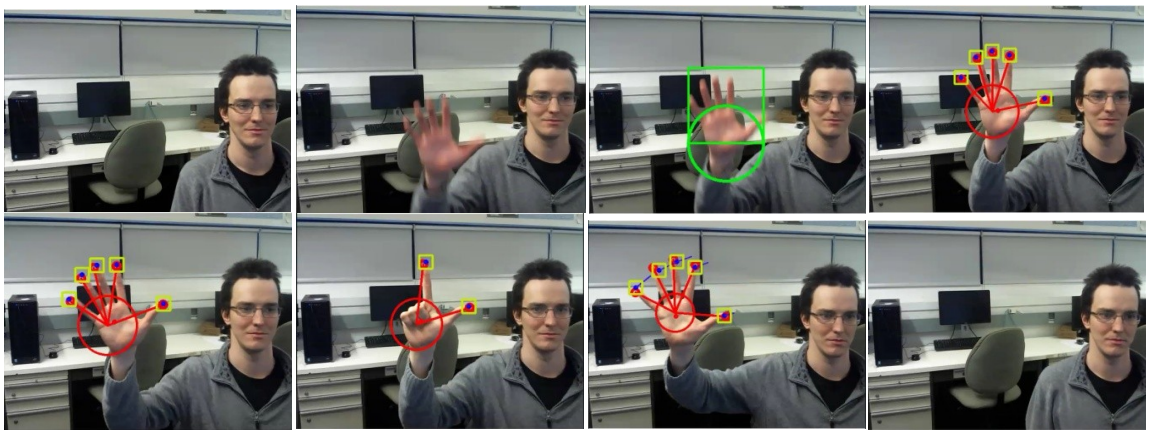


Figure 72 Examples of hand tracking results using the Gesture Recognition and Hand Tracking system

The Hand Tracking algorithm works well in real-time with a web camera. The algorithm works with manual initialization from the user. Manual initialization means putting the palm of a hand-5 gesture in the colour sampling area, define by a green square, in the center of the frame. Here is a video showing the performance of the hand tracking [186].

Here are some images showing the results of the hand tracking.

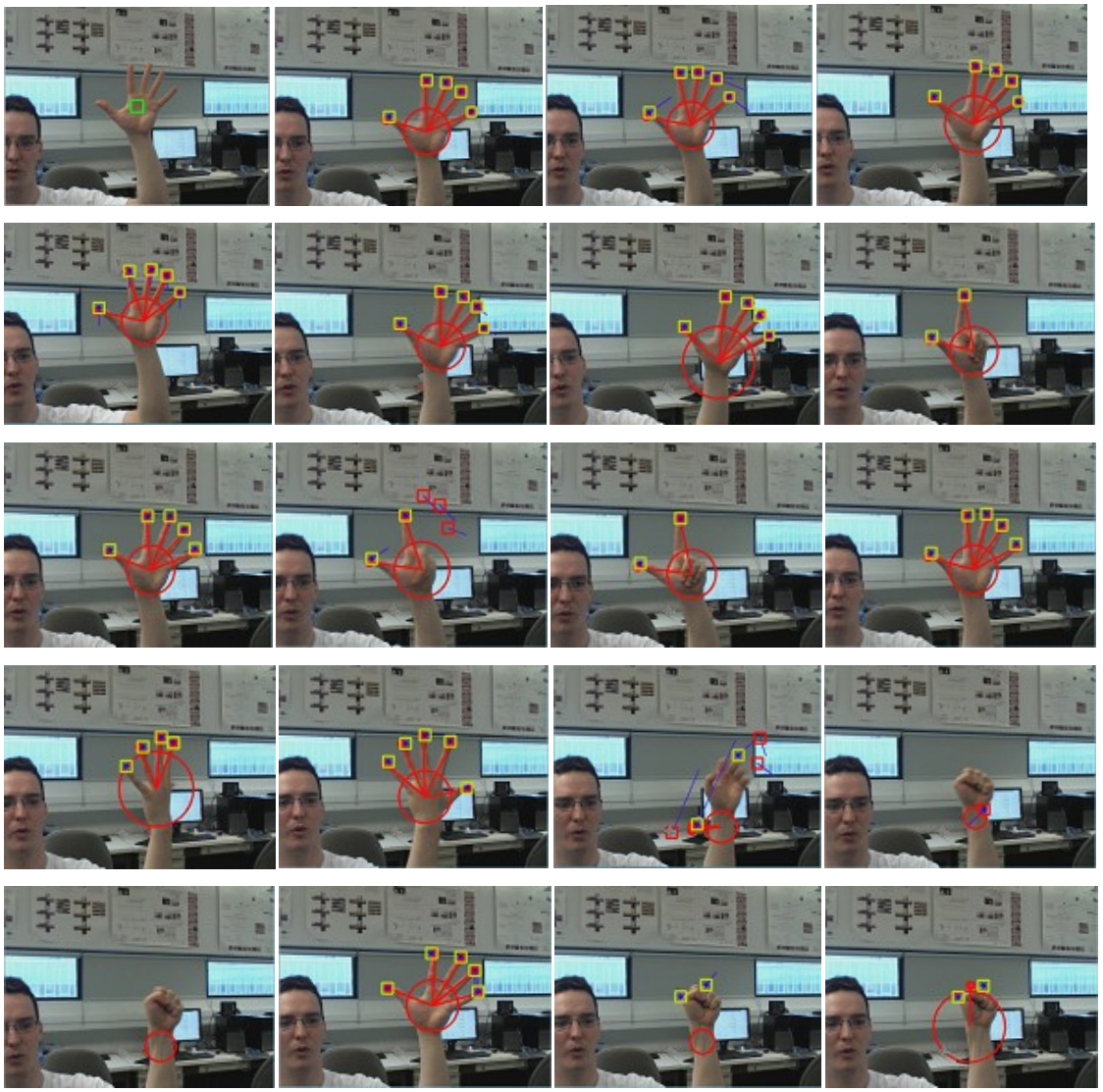


Figure 73 Examples of hand tracking results using the tracking algorithm

The tracking responds well to changes in position and hand pose of the tracked hand. Template trackers are added and removed as necessary to stably track the visible extended fingertips. Removed template trackers are marked in red and remain visible for

one frame in order to show that they have been terminated. Occasionally template trackers linger after their tracked fingertip is no longer visible and extended because that finger has been folded. But these lingering trackers are in most cases removed after a couple of subsequent frames. In addition sometimes the contour based tracking makes an error on how many visible fingertips there are. However these drawbacks seldom have any impact on the otherwise very robust and rapid hand tracking. The algorithm even works reasonably well in non ideal lighting conditions:

Figure 74 Examples of hand tracking results using the tracking algorithm in non ideal illumination

In the above figures the illumination is coming mostly from one side casting many shadows. This does deteriorate the tracking slightly, but it still maintains good tracking of the user's hand and in most cases corrects any poorly performing fingertip template tracker after a couple of frames.

It should be noted that occlusion with colours similar to the user's hand can deteriorate tracking results. Wood coloured objects and bright red colours can blend with the user's hand and cause the algorithm to lose track of the hand, these should be avoided for best tracking results.

The Pose Recognition component has also been successfully implemented. It works on top of the hand tracking algorithm, recognizing the static pose of the user's tracked hand. A video demonstration is available here [194].

Here are some results from the current configuration showing the Pose Recognition component's ability to recognize the static poses of the user's tracked hand.

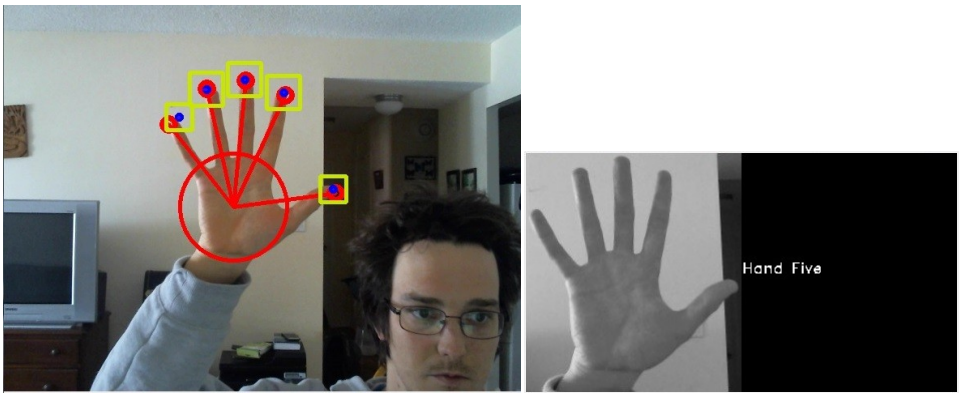


Figure 75 Pose recognition of hand-5 gesture

The above figures show the recognition of the Hand-5 pose of the tracked hand.

Figure 76 Pose Recognition of Fist

The above figures show the recognition of the Fist pose of the tracked hand.

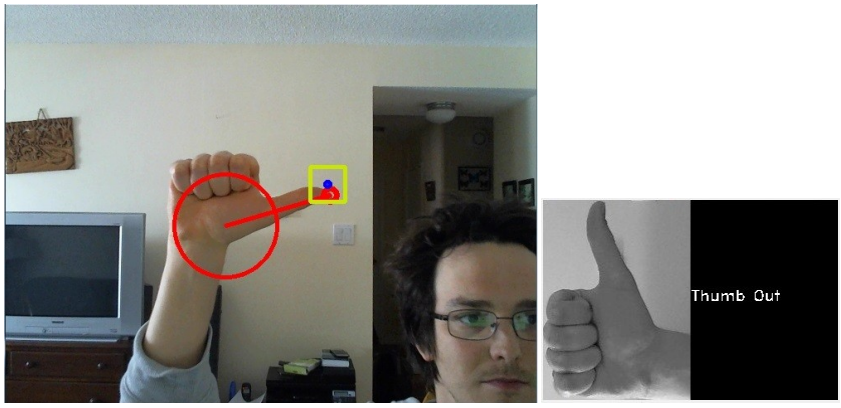


Figure 77 Pose Recognition of Thumb Out

The above figures show the recognition of the Thumb Out pose of the tracked hand.

The Hand Tracking algorithm has shown its potential as a real-time control input method for user interfaces. The algorithm was used to make a user interface on PC for a video game called Halo: Combat Evolved, which is a first person shooter requiring fast user response. The interface is called the Long Hands Gesture Recognition Controller for Halo. It uses two web cameras, with each web camera tracking one of the user's hands, so that both user's hands can be used to input commands in real-time. The tracking of the right hand is used to control the computer mouse which controls the aiming and the weapon firing. The tracking of the left hand is used to control keyboard commands which control character movement. A video demonstration of the Long Hands Controller for Halo is available here [187]. The Long Hands Controller proves that the Hand Tracking algorithm is rapid and robust enough to be used for user interfaces requiring fast and accurate user response. This controller also shows that the Hand Tracking algorithm can be used as though it were a keyboard or mouse input device. An earlier version of the hand tracking algorithm was also successfully used to control a computer mouse cursor in a general Microsoft Windows operation system setting. A demonstration is available here [189].

The Hand Tracking algorithm was also used to make an Android application called GR Paint. The application was built for use with Android Tablets, and it allows a user to draw pictures using the Hand Tracking algorithm. The application tracks the user's hand using the Android Tablet device's front facing camera, and it lets the user draw pictures on the Android device using their fingers in real-time. The drawn picture is displayed on the tablet screen. A video demonstration of GR Paint can be seen here. [188]. The GR Paint application proves that the Hand Tracking algorithm is light enough to be used on mobile devices in real-time.

5.7.2 Testing

The hand tracking performance of the system is tested using an annotated dataset of hand videos that was produced as part of this thesis to evaluate the performance of the hand tracking component of the Gesture Recognition and Hand Tracking system. The dataset is a series of videos where users perform a sequence of hand gestures in an indoor office environment. A user in each video displays a certain sequence of visible fingers. The

dataset is annotated frame by frame indicating where the user's hand is with a bounding box, and the number of fingers that the user is displaying. The dataset is available at [55] This dataset represents the types of environments where the system is intended to work, namely indoor environments with sufficient and consistent light. Wood coloured objects and bright red colours in the background of the scenes were avoided when possible to minimize the deterioration of the tracking that can occur when occluding with objects of these colours. This is due to these colours being quite similar to users' hands.

The hand tracking performance of the Gesture Recognition with Hand Tracking system will be evaluated using this dataset. The results will display for what percentage of frames were the users' hands successfully tracked and what percentage of frames had correct finger counting. If a hand contour is tracked within the annotated bounding box that frame is considered to be a successfully tracked frame. Likewise when no hand is tracked in frames where there is no bounding box that is also counted as a successfully tracked frame. When the number of fingers of the tracked hand matches the amount indicated in the annotation that frame is counted as a successful finger tracked frame. This is done for both contour tracking and template tracking.

There is a stochastic case where a bad tracking iteration could still produce some "correct" finger counting but this would not produce a good overall result when the tracking is bad throughout. Furthermore the hand tracking and pose recognition in its current state perform well in real-time with a webcam. Both components have been tested qualitatively by using the system in real-time and the hand tracking has been tested by making user interface applications.

The results of testing the system with the hand tracking dataset is tallied in a table. For each of the 19 hand tracking dataset videos the amount of frames with successful hand tracking, successful contour finger counting, and successful template finger counting, is expressed as a percentage. These show the hand tracking performance of the system.

The Pose Recognition component of the system is tested using the testing hand gesture dataset. Specifically the pose classification step which uses HOG cascades to recognize the pose of the tracked and rectified hand is tested. The pose recognition performance of

the Pose Recognition HOG cascades is evaluated for all 5 of the recognized gestures. The results are recorded in a confusion matrix. The testing is recorded in terms of sample numbers and percents.

Therefore numeric performance is measured for the Pose Recognition HOG cascades using the testing hand gesture dataset, and the tracking performance of the system is measured on the hand tracking video dataset with localization testing and finger counting. The series of Pose Recognition HOG cascades is evaluated on their collective ability to recognize each of the 5 poses by using the testing images, and the hand tracking is measured on its ability to count fingers and its ability to track the user's hand.

Only the best Gesture Recognition system configuration between HOG Cascades and HOG SVMs was used for the Gesture Recognition with Hand Tracking system, and this choice did not affect the Pose Recognition component or the Hand Tracking. Therefore the testing was only done on 1 system.

5.7.3 Quantitative Results

Testing was done of the Gesture Recognition and Hand Tracking system to determine to important aspects of the system. The first was its ability to recognize and track a user's hand when it appears in a video. The second aspect was its capacity to recognize the static pose of the user's tracked hand once tracking had already commenced. Each aspect was tested using a different dataset.

The dataset used to test the tracking performance is a set off 19 videos which contain footage of users displaying their hand in a hand-5 gesture for 30 seconds before displaying a pre determined sequence of numbers with their fingers. The users are free to extend whichever fingers they wish in order to display a given number with the only restriction is that the hand has to be front facing relative to its palm and the correct number of fingers is visibly extended. There is significant movement of general hand location from some users, while others keep their hands mostly stationary. An excel file accompanies each video providing frame by frame annotation. For each frame there is an indication if a user's hand is present, and if it is present there is a bounding box labeled

indicating the hand's location. The number of extended visible fingers that are being displayed by a user is also labeled in each frame.

The pose recognition aspect of the Gesture Recognition with Hand tracking system is tested using the testing portion of the dataset used to train the Gesture Recognition system presented in Chapter 3 as well as the pose recognition component itself. The test portion is comprised of images of 5 hand gesture poses which are the same gesture pose classes used to train the pose recognition component. This test dataset is made up of 2024 hand-5 samples,1705 fist samples, 1791 hand-I samples, 1760 thumb out samples, and 1638 hand-L samples.

The hand tracking performance of the Gesture Recognition and Hand Tracking system was tested using the 19 video hand tracking dataset. 3 things were tested to determine the performance, which were correct hand tracking, correct contour based finger counting, and correct template based finger counting. Correct hand tracking was evaluated by seeing whether or not the contour tracking algorithm, which governs the operation of the hand tracking, could track the user's hand within the annotated bounding boxes within the videos. This was done by determining the percentage of frames where a hand is present and a contour is tracked which has a center of mass within the hand location bounding box provided by the annotation of each frame. If there is no hand present, which is also indicated by the annotation of a given frame, and there is no hand tracked by the algorithm then this situation also counts as a tracking success. The other two elements tested were correct finger counting with both the contour and the template based portions of the tracking algorithm. Successful finger counting means that the output number of tracked fingers for a given frame is the same as the number in the frame's annotation.

The Gesture Recognition with Hand Tracking system was tested using this 19 video dataset to determine its ability to locate and track a user's hand in an unconstrained video only setting with a complex background. The results can be seen for each video in Table 33.

Table 33 Performance of the Gesture Recognition and Hand Tracking System

Video Number	Successful Hand Tracking	Successful Finger Counting with Contour Tracking	Successful Finger Counting with Template Tracking
1	0.9663	0.7243	0.6753
2	0.7527	0.6081	0.5194
3	0.9880	0.7954	0.5797
4	0.9777	0.7984	0.6706
5	0.9835	0.7102	0.5741
6	0.9753	0.8571	0.8519
7	0.9819	0.7418	0.6698
8	0.9795	0.7855	0.6188
9	0.9949	0.7237	0.6822
10	0.7417	0.4466	0.3883
11	0.4583	0.3031	0.3360
12	0.8824	0.7093	0.5824
13	0.9020	0.6544	0.7389
14	0.8874	0.6941	0.6558
15	0.9531	0.7685	0.6960
16	0.9500	0.4406	0.3828
17	0.7718	0.6328	0.6025
18	0.8003	0.3494	0.4138
19	0.9777	0.6610	0.6918
Average	0.8908	0.6529	0.5963

The system performs very well in an unconstrained setting in terms of tracking the user's hand. It achieves an 89% success rate. It is worth noting that this 19 video dataset is challenging with other people as well as their hands sometimes being visible in the back ground. It also has a several different users with different hand proportions.

On the other hand the 65% finger contour counting accuracy may seem low, but it must be remembered that noise on the contour boundary or in the segmentation can be a significant factor. Noise can add finger like shapes, or remove fingers from a tracked contour. This can cause a frame to register as false when the tracking is otherwise good for all of the successfully tracked fingertips. This scenario is demonstrated in Figure 78.

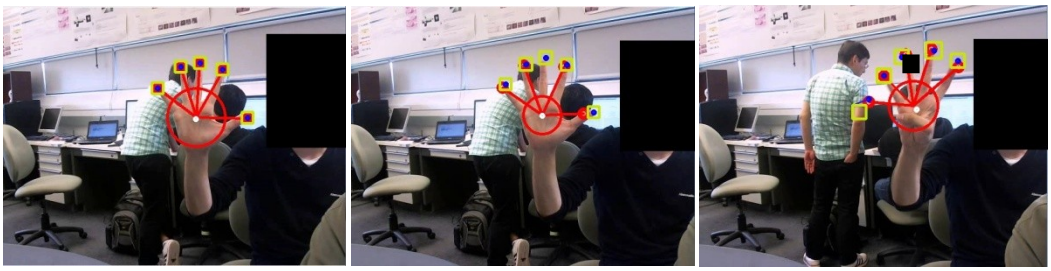


Figure 78 Tracking results examples

The frame on the left shows the system correctly tracking all 5 fingertips. In the middle frame 4 out of 5 fingertips are correctly tracked. In the right frame there are 4 correctly tracked fingertips while there is one is incorrectly added. In the cases displayed in the middle and the right frames a fingertip has been either removed or added due to noise in the segmentation. This causes the middle and the right frames to be recorded as false in the testing. Despite being false results, it is evident that in the middle and the right frames the tracking does successfully track fingertips. In a user interface situation an added finger or a removed finger due to noise can be ignored. This is significant because a robust experience can continue to be delivered to the user using the other tracking results.

For each of the 19 videos of the hand tracking dataset an set of coordinates which define the initial location of the user's first 30 second long hand-5 gesture is provided. This initial location allows the testing of hand tracking strategies that require a manual initialization. The initial location is defined within the user's presented palm in such a way that a good segmentation of the user's hand can be achieved in YCrCb colour space. These locations were used to test only the Hand Tracking algorithm of the Gesture Recognition with Hand Tracking system. This tracking strategy requires these initial locations to register the user's hand and initialize the tracking. This is handled automatically by the Gesture Recognition component in the complete Gesture Recognition with Hand Tracking system. The testing of just the hand tracking algorithm is useful in order to make a performance comparison. Testing the hand tracking algorithm involves only using the fixed initial hand location to register the user's hand and commence hand tracking for subsequent frames in each video. The performance of the hand tracking algorithm can be seen in Table 34.

Table 34 Performance of the Hand Tracking Algorithm using define coordinates for initial hand registration

Video Number	Successful Hand Tracking	Successfull Finger Counting with Contour Tracking	Successfull Finger Counting with Template Tracking
1	0.9663	0.7259	0.6876
2	0.8802	0.7885	0.7621
3	0.9639	0.8024	0.6489
4	0.9308	0.8453	0.8054
5	0.9824	0.7980	0.7113
6	0.9727	0.8636	0.6792
7	0.9456	0.7242	0.6124
8	0.5410	0.4949	0.4650
9	0.9881	0.8051	0.7500
10	0.7309	0.6018	0.5848
11	0.4627	0.3338	0.3419
12	0.8713	0.6074	0.6324
13	0.8169	0.7350	0.6995
14	0.9635	0.7413	0.5840
15	0.9552	0.7312	0.6539
16	0.9359	0.5078	0.6453
17	0.7718	0.6613	0.5544
18	0.7955	0.3591	0.3688
19	0.9264	0.6918	0.7209
Average	0.8632	0.6747	0.6267

The tracking performance diminishes when using just the Hand Tracking algorithm. The reason for this is that the Hand Tracking algorithm can lose track of the user's hand and this drawback is overcome by the Gesture Recognition step which provides a re-initiation and recovery mechanism to the system. The Gesture Recognition system clearly aids the hand tracking by allowing it to recover. This makes the Gesture Recognition with Hand Tracking system significantly more robust than by using the hand tracking alone. However the hand tracking algorithm is able to achieve slightly more accurate finger counting when operating on its own. This is explained by the fact that the defined locations of the initial location of the user's hand allow for more stable initial hand registrations. These more stable registrations result in reduced noise in subsequent frames.

Motion blur can affect the initial registration of the user's hand which can have an impact on subsequent hand tracking. When a user's hand is initially presented in the video dataset for many of the videos it moves into view from off camera. This means that as these hands are presenting the hand-5 gesture there is significant motion than can cause

significant blur in the video which can deteriorate the quality of the initial registration. The quality of the extracted colour signature can be reduced by motion blur which in turn deteriorates the quality of subsequent hand tracking and it can make it unstable. The system was tested again in order to see the effects of motion blur. The Gesture Recognition with Hand Tracking, and the Hand Tracking component were tested a second time with each video with instructions to run only after 15 frames of the initial hand-5 gesture had been present. It was assumed that after 15 frames of being displayed the hand-5 gesture would be either stationary or almost stationary and motion blur would no longer be present. This would display a clearer and sharper image of the hand-5 gesture allowing for better registration of the user's hand. The results of the previous two experiments from Table 33 and Table 34 are summarized in Table 35 and compared to the results of running both systems after the 15 frame hand-5 gesture delay.

Table 35 Testing the effects of motion blur

Experiment	Description	Successful Hand Tracking	Successful Finger Counting with Contour Tracking	Successful Finger Counting with Template Tracking
1	GR with Hand Tracking System	0.8908	0.6529	0.5963
2	GR with Hand Tracking System after 15 frames of hand-5 gesture	0.9001	0.6637	0.6144
3	Hand Tracking Component with Defined Coordinates for initialization	0.8632	0.6747	0.6267
4	Hand Tracking Component with Defined Coordinates for initialization after 15 frames of hand-5 gesture	0.8138	0.5956	0.5359

The Gesture Recognition with Hand Tracking system benefits from the reduced motion blur resulting from the 15 frame delay. The reduced motion blur results in a better registration of the users' hands from the initial gesture recognition. In all 3 of the criteria small improvements in performance can be seen. It allows the system to obtain 90% successful hand tracking.

Contrary to the expected result the 15 frame delay reduces the performance of the Hand Tracking algorithm when it is used on its own. All 3 of the evaluated criteria are reduced 5-9%. This drop in performance suggest that the delay reduces the quality of the initial registration. The hand tracking algorithm when used by itself uses the defined coordinates provided to obtain a good user hand registration and it is evident that these locations are time sensitive.

The initial locations of the user's hand provided in each video were set in order to provide a good hand location for segmentation with the assumption that no delay would be used. They were chosen manually based on visual interpretation and quality evaluation of the subsequent segmentation and tracking with the hand tracking algorithm. When the locations were being chosen the hand tracking algorithm running from frame zero. It was attempting user registration using the defined coordinates from frame zero without waiting for any delay. After a 15 frame delay the defined hand locations are no longer as optimal as they once were because a user's hand is never truly stationary. The delay, while it reduces the impact of motion blur, allows for small shifts in the user's hand which result in less optimal user hand registration when using the defined coordinates. The sampling locations relative to the user's hands are no longer the same as they were when they were initially set which results in the registrations being deteriorated which affects subsequent tracking. This further builds the case that using Gesture Recognition with Hand Tracking in a combined strategy is a good idea. The Gesture Recognition step allows the Hand Tracking to recover from bad tracking by re initializing.

The results can also be examined on a per user basis to get a sense of the generality of the algorithm. The video dataset was produced by 5 different users, with the details on which videos were produced by which user being listed in Table 36. User 2 being the author of this thesis.

Table 36 Hand Tracking Video Dataset user identification

User ID	Name	Videos	Number of Videos
User 1	Hamid	1,2	2
User 2	Pavel	3,4,5,6,13,14,15	7
User 3	Ashkan	7,8,9	3
User 4	Ishmael	10,11,12	3
User 5	Emilienne	16,17,18,19	4

The results of Table 35 can be separated into per user results which can be seen in Table 37.

Table 37 Tracking Results divided per user

Experiment	Description	User ID	Successful Hand Tracking Per User	Successful Finger Counting with Contour Tracking Per User	Successful Finger Counting with Template Tracking Per User
1	GR with Hand Tracking System	User 1	0.8595	0.6662	0.5974
		User 2	0.9524	0.7540	0.6810
		User 3	0.9854	0.7503	0.6569
		User 4	0.6941	0.4863	0.4356
		User 5	0.8750	0.5210	0.5227
2	GR with Hand Tracking System after 15 frames of hand-5 gesture	User 1	0.9904	0.8107	0.7356
		User 2	0.9591	0.7197	0.6211
		User 3	0.9858	0.7676	0.7184
		User 4	0.7166	0.5018	0.4810
		User 5	0.8252	0.5358	0.5642
3	Hand Tracking Component with Defined Coordinates for initialization	User 1	0.9233	0.7572	0.7248
		User 2	0.9408	0.7881	0.6832
		User 3	0.8249	0.6747	0.6091
		User 4	0.6883	0.5143	0.5197
		User 5	0.8574	0.5550	0.5723
4	Hand Tracking Component with Defined Coordinates for initialization after 15 frames of hand-5 gesture	User 1	0.9424	0.7368	0.7420
		User 2	0.9186	0.6806	0.5672
		User 3	0.8369	0.6895	0.6328
		User 4	0.3691	0.2742	0.2573
		User 5	0.8824	0.5468	0.5142

The first thing to note is that experiment 4 has worst tracking results for user 4 validating that the defined coordinates for initial registration are indeed very time sensitive and as

such experiment 4 is the least informative on the performance of the system. Looking at the tracking results of experiments 1 and 2 performance gains can be seen from the added 15 frame delay in 4 out of 5 users. This demonstrates that reducing the initial motion blur of a presented user's hand can have significant impact on the quality of the initial hand registration. The tracking results of all the experiments show that different users can have success using the system. All 5 users have good tracking results for experiments 1, 2, and 3, which reflect more closely the real world conditions in which users would use this system.

Looking at the contour finger counting results shows additional traits of the system. The videos of users 4 and 5 are consistently the lowest performers when it comes to contour finger counting. User 5's videos having poor finger counting can be explained in part due to self occlusions and it part due to the colour variability of the user's hand which leads to stable tracking results however not all fingers are correctly identified.

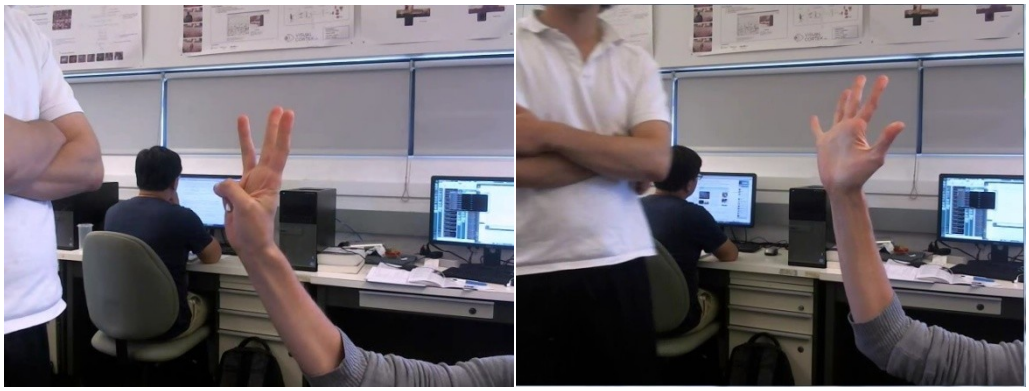


Figure 79 Examples of self occlusion in User 5's videos.

User 4's videos are the most difficult for hand tracking in the dataset. These videos have the most objects in the background of similar colour to User 4's hands. These include bright red hues and other hands. This serves to highlight the limitation of the system with regard to occlusion of similar coloured objects. These issues would be interesting to address in future research which would most likely require data driven methods to train models that can discriminate between foreground and background.

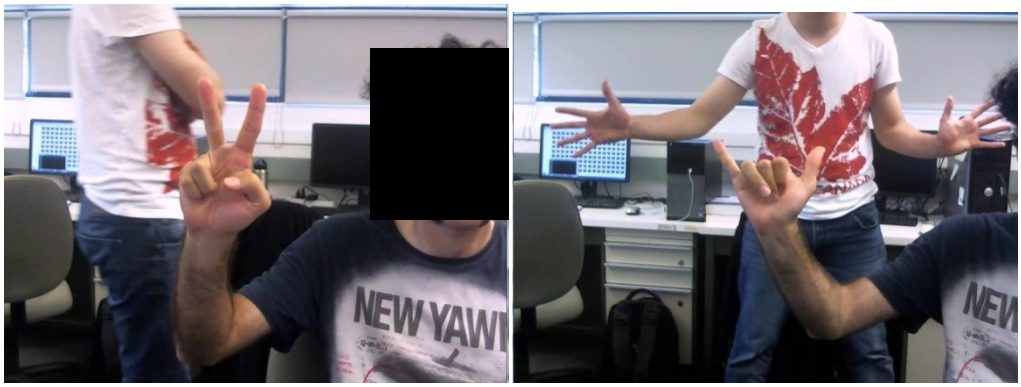


Figure 80 Similar coloured objects to the user's hand in the background of User 4's videos.

The performance of the template tracking for fingertip counting is predictable less than the contour based tracking fingertip counting for the majority of the videos. Template tracking being driven by the contour tracking is therefore less robust to errors because its focus was to provide stable position tracking of fingertips not to identify bad targets, although it still does a reasonably good job compared to contour based tracking fingertip counting.

The distribution of videos does favor User 2, however as it can be seen in the results more samples does not mean more favourable testing performance in all cases. Other users outperform User 2 in experiment 1 and 2. They also outperform User 2 in experiment 4 but as stated earlier this experiment is less informative than the others because the defined coordinates for initial registration of the user's hand are time sensitive. More videos with a larger variety of users would be very informative, however even this 5 user dataset proves that the algorithm can be used by users other than the thesis author with comparable performance.

The Gesture Recognition with Hand Tracking system has shown good performance capable of automatically tracking the user's hand in an unconstraint video setting and recovering from bad tracking. In several of the experiments when the hand tracking loses track of the hand the gesture recognition step recognizes a user's hand-5 gesture after tracking has terminated and re initializes tracking. This allows the Gesture Recognition with Hand Tracking system to have better overall hand tracking performance than the Hand Tracking component can achieve on its own. This demonstrates the robustness of

the Gesture Recognition with Hand Tracking system. The testing has also shown how the quality of the initial user hand registration impacts subsequent tracking. Good quality registrations result in better subsequent tracking whereas a bad registration can make the system lose track of the user's hand. Tracking a user's hand is a difficult endeavor and tracking can fail, which makes a good Gesture Recognition step a necessity for making the tracking robust, automated, and able to recover from setbacks.

The Pose Recognition component was tested using the testing dataset described earlier. This dataset allows for the Pose Recognition cascades to be tested on a set of images of the 5 different classes of recognized poses. However this testing does not represent a live user scenario which commences with gesture recognition in an unconstrained setting. The unconstrained scenario would require a collection of a new dataset with users displaying their hand for tracking and subsequently making a series of determined hand poses. This would require significant labor and is beyond the scope of this thesis because the pose recognition of the Gesture Recognition with Hand Tracking system can be tested with the 5 gesture test set previously described. As such the Pose Recognition was tested separately from the Gesture Recognition with Hand Tracking system in a scenario which the testing dataset allows. This scenario assumes successful tracking has provided the necessary demarcation of the user's hand. The test images of the 5 hand gestures of the testing dataset are therefore classified by using the Pose Recognition component directly.

Table 38 the confusion matrix of the Pose Recognition component.

Table 38 Confusion Matrix for Pose Recognition

Sample\Result	Hand-5	Hand - Fist	Hand-I	Thumb-Out	Hand-L	No Detection	**Total**
Hand-5	**1341**	3	6	7	18	649	2024
Hand-Fist	2	**173**	0	1	11	1518	1705
Hand-I	21	20	**679**	16	16	1039	1791
Thumb-Out	1	20	5	**439**	6	1289	1760
Hand-L	15	10	43	35	**723**	812	1638

The performance of the different classes of the Pose Recognition component is summarized in Table 39.

Table 39 Pose Recognition Overall Results

Gesture	True Positive	False Positive
Hand-5	0.6625	0.005657
Hand-Fist	0.1015	0.007348
Hand-I	0.3786	0.007577
Thumb-Out	0.2494	0.008243
Hand-L	0.4414	0.007005
Average	0.3667	0.007166

The best performing gesture is the hand-5 gesture. This is because of its shape which is the most unique among the 5 tested gestures with 5 visible extended fingers. The second best performing gesture is the Hand-L gesture which has 2 extended visible fingers which makes it more distinct than the Hand Fist, Hand-I, and Thumb Out gestures. An interesting cases is the performance difference between the Hand-I and Thumb Out gestures. While both have 1 extended visible finger the Hand-I gesture outperforms the Thumb Out gesture suggesting that the Hand-I is more distinct and easily recognized. This can be explained by user hand proportions, hand flexibility and user habits. Most users have an index finger that they can extend a farther distance from their palm than their thumb. Additionally users sometimes do not stretch their thumb to its full range of motion and keep it closer to their palm. Sometimes this can be due to flexibility, other times its just more comfortable for a user even when their flexibility allows them to extend their thumb further. This is demonstrated by a couple of samples from the dataset.

Figure 81 Samples of the Hand-L gesture

When the thumb is closer to the palm it becomes less distinct and starts to look like a fist gesture. Proportions, flexibility, and habits make the thumb out gesture less distinct than

the hand-I gesture and the hand-I gesture consequently performs better. The least distinct gesture is the fist gesture. It has no extended fingers and also its boundary is a square or circular structure. The results show that the fist gesture is the least well performing gesture with the lowest true positive rate.

All of the 5 gestures perform well however. The results are at least 10% for the true positive rate for all gestures, and all gestures have very low false positive rates. All of the gestures are suitable for real-time performance because of the low false positive rates of less than %1 for each gesture. Even the least distinct hand fist gesture is suitable. In theory its 10% true positive rate would produce in a real-time system 30 fps system 3 detections in 1 second. Which is sufficient according to the criteria discussed in Chapter 3 for real-time hand gesture recognition. This makes the Pose Recognition able to process the user's hand and recognize its pose in real-time.

The Pose Recognition component is successfully implemented in the Gesture Recognition with Hand Tracking system. A video demonstration of the system include the Pose Recognition component can be seen at [185]. When the Pose Recognition is run in the Gesture Recognition with Hand Tracking system in a live real-time scenario, the component also validates the recognized gesture against the finger count produced by the contour based tracking. For a hand-5 gesture the contour tracking should produce 4 or more tracked fingers. A fist gesture should have 0 fingers. There should only by one finger for the Hand-I and Thumb Out Gesture. The contour tracking should only have 2 fingers for a Hand-L gesture. Using the contour tracking finger count as a second validation test allows the pose recognition HOG cascades to be used with increased sensitivity. This increased sensitivity results in better live real-time performance and the added chance of having a false positive is mitigated by the contour tracking finger count test. Furthermore the ability of the Pose Recognition component to distinguish between 2 hand poses that each have 1 extended visible finger is an important milestone for the system. Distinguishing between the hand-I and thumb out gestures which both only have 1 extended finger means that the pose recognition of the system has advanced beyond simple finger counting. The machine learning trained Pose Recognition component

allows the system to better perceive what the user's hand is doing by understanding the structure of the user's hand.

5.8 Summary

The Gesture Recognition system presented in chapter 3, which is capable of multi scale gesture recognition, was successfully combined with a hand tracking algorithm based on the general hand model. This chapter presented this combined system which automatically detects a user's hand gesture and then proceeds to track the user's hand as long as it stays in view. For simplicity, and because it was the best performing gesture, only the hand-5 gesture is used to initialize the hand tracking.

The Tracking has two main components: Contour based tracking with the general hand model, and the Template tracking. The Contour based tracking partially guides the Template tracking to improve the results and the template tracker stability.

A test video dataset was specifically made for the evaluation of the hand tracking capabilities of the Gesture Recognition with Hand Tracking system, and the Hand Tracking algorithm separately. The Hand Tracking was evaluated on its ability to track the user's hand within the ground truth bounding box of a given video frame and also on its ability to correctly count the fingers of a user's tracked hand. Testing shows that the hand tracking was able to track the location of the location of the user's hand well in the video dataset. The drawback of the method comes from the fact that sometimes fingers are eliminated from tracking, or false finger tip detections are tracked. Both of these phenomena are due to noise in segmentation, which deteriorates the algorithm's finger counting performance. This can be improved in future work. Never the less, the successfully tracked fingertips provide very stable output locations, which allowed the hand tracking system to be used as a control interface for a variety of interesting and challenging applications for human computer interaction.

The combined Recognition and Tracking system is also capable of Pose Recognition of the user's tracked hand. Pose recognition occurs after hand tracking. It uses the hand location to narrow the field of view analyzed within a frame. This makes it different from hand gesture recognition which processes the entire input frame. The Pose Recognition is

capable of identifying the 5 gesture poses for which it was trained. It was trained using all of the samples of the Hand-5, Hand-L, Hand-I, Thumb Out, and Fist gestures of the training dataset. The narrow field of view makes the classification perform better than un constrained hand gesture recognition because less of the background objects have to be processed by the classifier.

The combined Hand Gesture Recognition and Tracking system is a robust and powerful hand tracking solution, which works in a significantly constrained 2D video only medium without tuning from a user. It is a promising system for user interfaces with several interesting proven applications.

Chapter 6 Gesture Recognition with Colour Clustering Segmentation

6.1 Overview

In chapter 4 a real-time hand gesture recognition system was presented. This chapter will examine possible ways to improve the gesture recognition performance. Testing will be done using the same methodology as chapter 4.

Gesture Recognition with Colour Clustering Segmentation is a real-time hand gesture recognition system that aims to have faster hand gesture recognition than the system presented in chapter 4.

The Gesture Recognition with Colour Clustering system is actually a modification of the Gesture Recognition system presented in chapter 3. It is designed to improve upon that system by using all of the colour information available in the hand gesture detection boxes. The Gesture Recognition system uses machine learning trained multi scale detectors to detect potential hand gestures and then colour segmentation and contour shape analysis to find the hand shapes in the bounding boxes to either validate or reject the detections.

The segmentation is achieved by sampling the input frame at several locations determined by the location of the bounding box. For more specifics on the types of sampling patterns used consult the Detected Gesture Sampling section in chapter 4. The sampling approach however relies significantly on well placed detection boxes, and it can miss important colour information and result in bad segmentations. Colour clustering is an idea proposed to overcome this problem.

6.2 Rationale for Colour Clustering

It is observed that distinct groups of similar colours such as human skin tones, clothing textiles, shades of wall paint, and the colours of home and office furniture materials produce distinct disjoint and separable contours on 2d CrCb histograms. These separable

contours, or colour clusters, group similar colours within their boundaries. 2d histograms are otherwise known as colour signatures.

The following example graphically illustrates the idea.

When a hand image is sampled in the Detected Gesture Sampling, the surrounding area (sampling zone) around a sample point is used to produce a CrCb histogram by the Shape Extraction with CrCb Histogram section, which is in turn used to segment the hand image. When a sample point is poorly located then the corresponding sampling zone is also poorly located. It often looks like Figure 82.

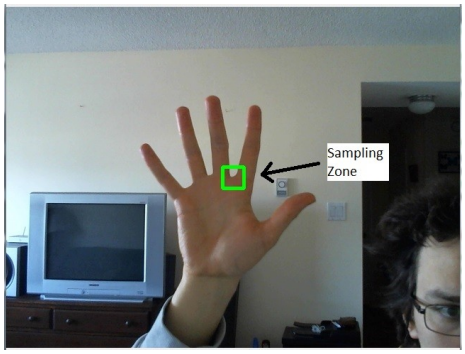


Figure 82 Poorly placed sampling zone

The sample zone is poorly located because it is using both background and hand pixels to form the 2d CrCb histogram. This sort of sampling zone produces a 2d CrCb histogram and backprojection like the one in Figure 83.

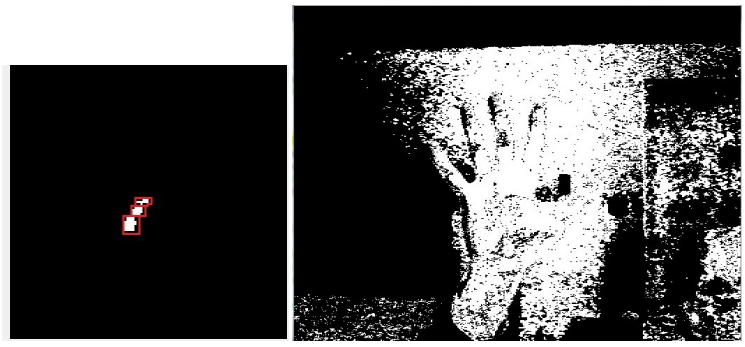


Figure 83 Segmentation resulting from a CrCb histogram with multiple modes

There are multiple contours or colour modes present in the 2d CrCb histogram. They have been highlighted in red to be easily seen.

It can be seen that the hand has been segmented but there are additional noise artifacts attached to the hand contour. This is due to the fact that the 2d CrCb histogram contained background pixels.

Likewise when a sampling zone is well located, and it is completely within the hand in the hand image, it looks like this:

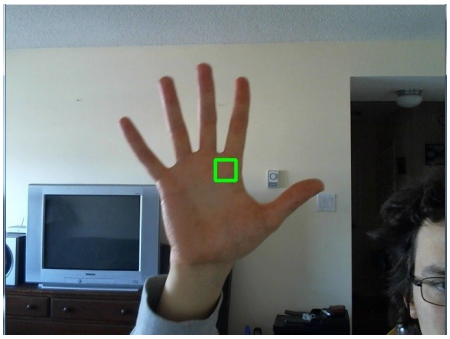


Figure 84 Well placed sampling zone

A well located sampling zone produces a 2d CrCb histogram and back projection like the ones in Figure 85.

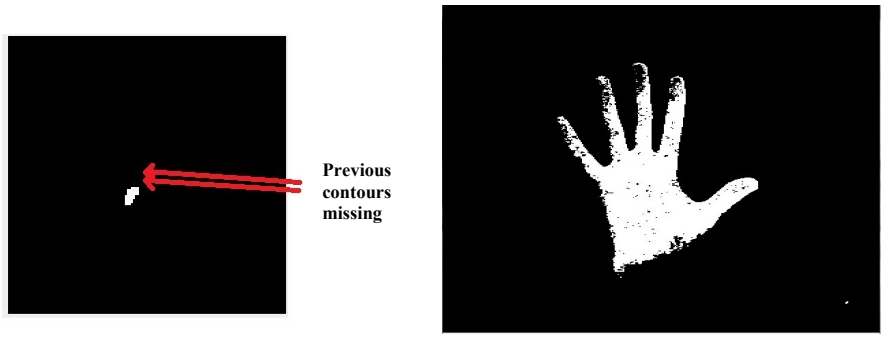


Figure 85 Segmentation results using a CrCb histogram with only 1 colour mode

In this case the back projection has produced a good segmentation of the hand. What is important to note is that the 2d histogram only has 1 distinct contour shape. This example reflects the experience gained working with colour segmentation. The best hand segmentations were often produced by only a single colour contour. Sampling the center of a hand will typically only produce 1 contour in the 2d histogram which usually results in very good segmentations of the user's hand. These sort of situations have happened numerous times during the experimenting work done for this thesis.

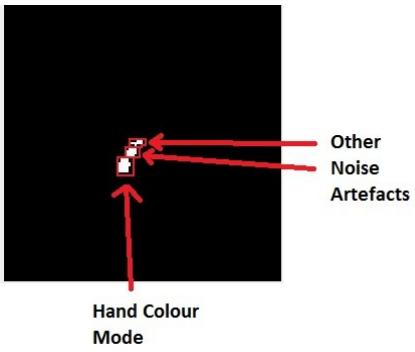


Figure 86 CrCb histogram with multiple modes

It is hypothesized that hand colours often times produce only a single colour contour in a 2d CrCb histogram. In situations where colour sampling of a hand produces more than 1 colour contour on the 2d histogram the additional contours or clusters are hypothesized to be due to non hand colours. These contours in the 2d histogram produce noise artifacts in the back projection deteriorating the quality of the hand segmentation. If these additional contours can be somehow removed the segmentations can be improved.

The idea behind colour clustering is to resolve the noise problem in hand segmentation which occurs from multiple contours in a 2d histogram. The general idea is to separate the 2d histogram into component histograms, one for each distinct colour contour. These 2d component histograms will be used to produce multiple segmentations and multiple contours. That way the hand colour contour or cluster will be used on its own to produce a hand segmentation. All the contours produced by the multiple back projections will be evaluated by the gesture validation procedure of the Gesture Recognition system presented in chapter 4 to find the valid hand contour for the given gesture detection box. The hand segmentation produced by the hand colour contour will produce better quality hand contours more often, which will increase the speed of the gesture recognition.

Colour clustering is also designed to use all of the colour information in the hand gesture detection bounding boxes. The sampling patterns used to produce the 2d histograms from the gesture detection bounding boxes will be replaced with just one histogram of the whole gesture detection bounding box. That way the hand, if it's there, will never be missed because its colour information will always contribute to the 2d CrCb histogram. This, combined with separation of the 2d CrCb histogram contours, will result in better

quality segmentations which will produce clear hand shape contours in more input frames. This will result in more frequent hand gesture validation which will increase the gesture recognition speed of the system.

6.3 System Layout

The flowchart in Figure 87 shows the steps of the Gesture Recognition with Colour Clustering Segmentation system. The boxes marked by "*" indicate modifications introduced by the Colour Clustering segmentation method.

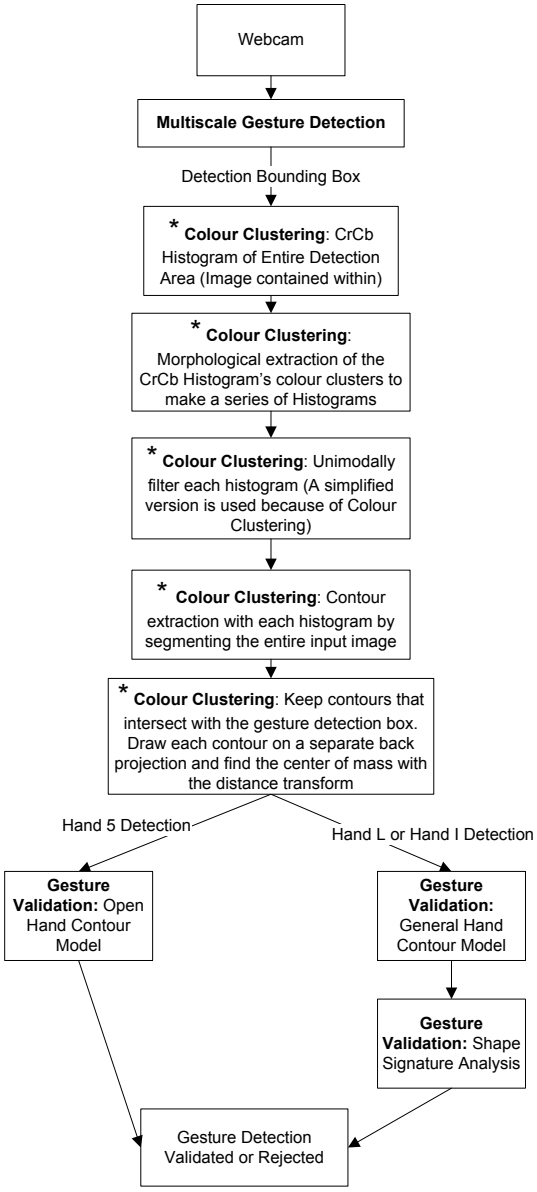


Figure 87 Gesture Recognition with Colour Clustering Segmentation System Flowchart

As it can be seen from the flow chart several things have been changed from the Gesture Recognition system from chapter 3. The multi scale hand gesture detections are done in the same way as before. Only the best detector configuration, determined by the Gesture Recognition system testing, will be used for the multi scale detectors in this system. The shape extraction with unimodal histogram filtering has been removed. The shape extraction has been replaced by the colour clustering which does a simplified version of unimodal histogram filtering as part of its process.

The Colour Clustering is the biggest addition in this algorithm. It is represented by several steps in the flowchart: *Take the CrCb Histogram of the entire detection image*, *Morphologically extract the histogram's colour clusters and make them into separate histograms*, *Unimodally filter each histogram* (this is the simplified unimodal filtering mentioned earlier), *Contour extraction with each histogram using the entire input image*, and *Keep contours that intersect with the detection bounding box*. All of these steps will be explained in detail in the Colour Clustering Segmentation section of this chapter.

The validation procedure is used in the same way as before in the Gesture Recognition system of chapter 3. The colour clustering provides back projections and sample points to the validation procedure in order to process the contours and determine if they are valid. The details of how the back projections and sample points are generated are also in the Colour Clustering section.

The final change is that there is now step at the end of the validation procedure to determine the best valid hand gesture contour if there is more than one valid contour found. This is done with Support Vector Machines which choose which contour is the best or "most like sought hand gesture" from the multiple valid candidates.

6.4 Multi scale Gesture Detection

The hand gesture detection is done in the same way as in the Gesture Recognition system presented in Chapter 3. A machine learning trained multi scale detector is run for each gesture on the input image from a video feed. The detectors produce bounding boxes over the hand gestures in the input image when they appear, and are detected. The bounding boxes, when they are generated, are passed to the Colour Clustering Segmentation.

6.5 Colour Clustering Segmentation

The Colour Clustering Segmentation is responsible for processing the bounding box gesture detections and producing contour shapes from the original input image. The contour shapes are used to make synthetic back projection binary images, which only contain a single contour to be analyzed. These back projections are passed on to the Gesture Validation procedure which is identical to the one described in the Gesture Recognition system of chapter 3. The Gesture Validation procedure uses back projections as inputs and validates or invalidates the contours and thus the gesture detections.

The Colour Clustering algorithm's job is to process the bounding box and the input image to extract the back projections necessary to call the Gesture Validation procedure, and to output the result of the hand gesture validation. The Colour Clustering Segmentation algorithm is called for every gesture detection bounding box produced by performing Gesture Detection on the input image.

The Colour Clustering algorithm seeks to separate the different colour modes or clusters that make up the objections located within the gesture detection boxes. These colour clusters are then used to perform segmentations on the input image, and only contours that intersect with the hand gesture detection boxes are kept. In this way every object including the user's hand if its present, will get separated with its own distinct colour cluster which should result in better segmentations and therefore more frequent validations by the Gesture Validation procedure.

The details of how the Colour Clustering Segmentation algorithm works are found in the following steps. The details will be explained with a crafted example to show the idea of Colour Clustering Segmentation.

1. The first step in the Colour Clustering Segmentation algorithm is to obtain the input video frame and the hand gesture detection bounding box from the input.

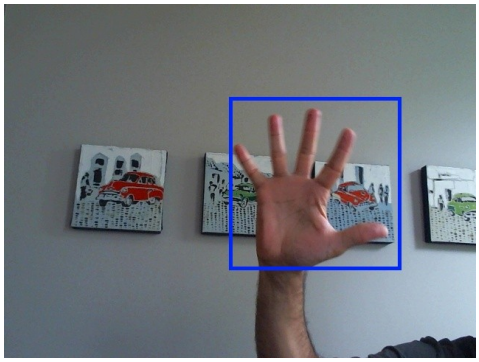


Figure 88 Input detection bounding box

In practice for the implementation the whole clustering procedure is repeated with a second focused ROI located in the center. The focused ROI is of size 60 by 60 pixels. This gives the algorithm an additional chance to obtain a good segmentation of the hand.

2. Take the CrCb histogram of the entire image contained within the bounding box.

Figure 89 Example of multimodal histogram that could be generated

It will probably contain, multiple, potentially attached, contours representing groups of colours otherwise referred to as colour clusters.

5. The colour cluster contours should be separated with morphological operations. This is done using a combination of erosion and dilation. The resulting separated colour cluster contours should then be masked with the original 2d CrCb histogram generated using the image within the bounding box.

Figure 90 Separated colour modes

8. Unimodally filter each of the histograms in the colour cluster histogram collection in order to fill in any missing colours of the modes. Unimodal filtering is simpler in this algorithm than it was in the Gesture Recognition system of chapter 3. Filtering is done by finding the bounding box of each contour in each colour cluster histogram, and then replacing the contours with their filled bounding boxes. Figure 91 illustrates the idea.

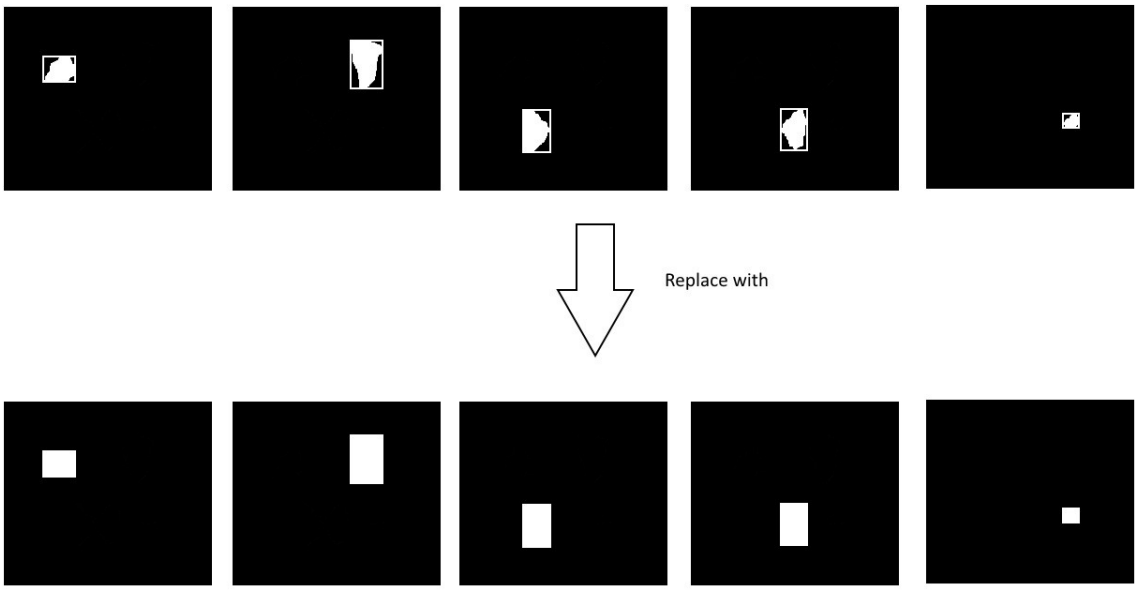


Figure 91 Using unimodal filtering on each mode

9. Use each histogram in the filtered colour cluster histograms collection to make a back projection of the input frame. For this example, in order to keep things simple, only 3 back projections will be made even though 5 histograms were generated.

Figure 92 Sample back projections that could be generated from the input image in Figure 88.

10. For each back projection find the contours that intersect with the input gesture detection box, or contain the bounding box, or are contained within the bounding box. Draw each contour on a separate blank back projection binary image.

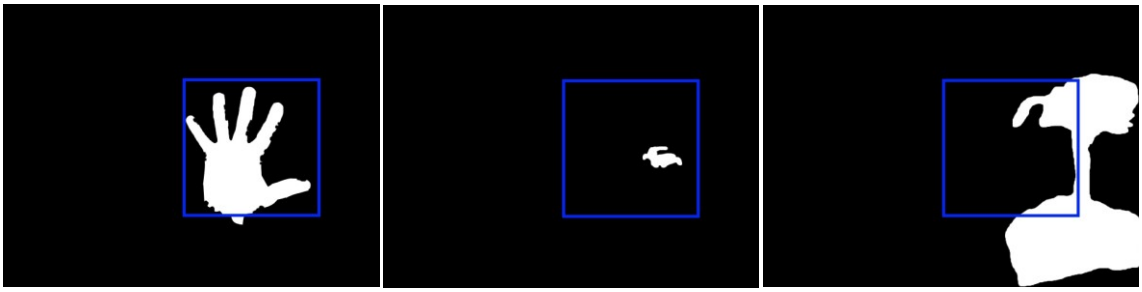


Figure 93 Contours that intersect with the input bounding box

The initial bounding box for the detected gesture is shown in blue for the figures above.

11. For each individual contour drawn on a separate back projection call the Gesture Validation procedure.

12. The final result should be 1 or more valid hand gesture contours that intersect with the gesture detection box if the gesture detection is valid. 0 validated contours means that the gesture detection is invalid.

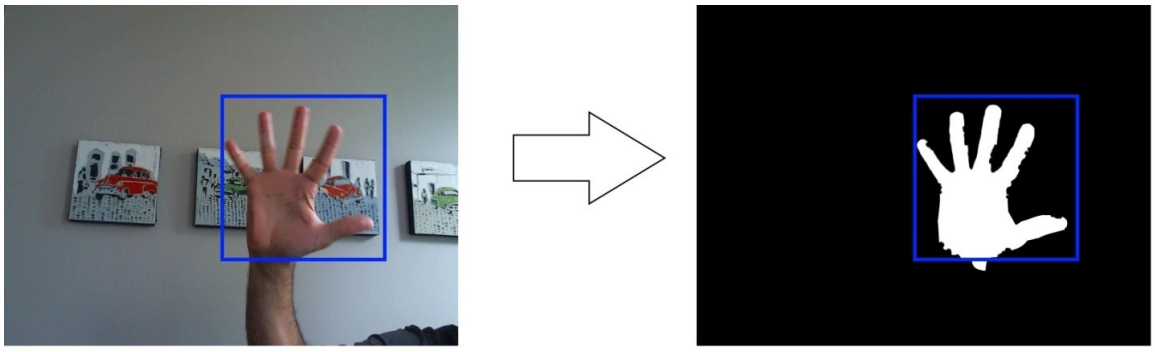


Figure 94 Positive Gesture Validation Result obtained from Gesture Detection

Thus Colour Clustering Segmentation will be able to validate or invalidate a gesture detection by finding all of the object contours, including hand contours if hands are present, that intersect with the gesture detection bounding box.

Upon termination the output of the Colour Clustering algorithm outputs whether or not a gesture detection is valid, and any valid contours that were found by the algorithm of that hand gesture.

6.6 Results

6.6.1 How the Gesture Recognition with Colour Clustering Segmentation system was tested

The Gesture Recognition with Colour Clustering Segmentation was tested in the same way as the Gesture Recognition system presented in Chapter 4. Performance comparisons were made to both configurations of the Gesture Recognition system described in Chapter 4. The Gesture Recognition system performance with gesture validation and without gesture validation was also considered when comparing it to the Gesture Recognition with Colour Clustering Segmentation system.

6.6.2 Quantitative Results for Colour Clustering

Colour clustering was implemented with on both the HOG Cascade and the HOG SVM configurations of the Gesture Recognition system. However due to poor performance with the Hand-L and Hand-I gestures the colour clustering was only retained for the Hand-5 gesture in both of the configurations. Table 40 shows the results of best

configurations of both of the systems with the new colour clustering algorithm used for segmentation of the hand-5 gestures.

Cascade Configuration Original

Gesture	% True Positives	% False Positives
hand-5	31.67	0.04854
hand-L	25.58	0.8943
hand-I	16.19	1.008

Cascade Configuration with Colour Clustering Best Configuration

Gesture	% True Positives	% False Positives
hand-5	39.38	0.04854
hand-L	25.64	0.8473
hand-I	16.19	1.008

SVM Configuration Original

Gesture	% True Positives	% False Positives
hand-5	26.33	0.07281
hand-L	20.51	1.828
hand-I	10.11	2.898

SVM Configuration with Colour Clustering Best Configuration

Gesture	% True Positives	% False Positives
hand-5	30.53	0.4288
hand-L	20.51	1.797
hand-I	10.16	2.866

Table 40 Summary of Colour Clustering Results for Best Configurations

These best colour clustering configurations are placed side by side with the originals in order to see what the effects are on performance. The colour clustering performs well with the hand-5 gestures for both Cascade and SVM configurations. This results in a very good true positive increase of 8% for the Cascade system with a 0% increase in false positive rate. The SVM system enjoys a respectable 4% increase in true positive rate at the cost of a marginal 0.35% increase in false positive rate. It can be seen that the colour clustering algorithm significantly improves the recognition performance of the hand-5 gestures. The runtimes for the testing of both systems with the colour clustering are not significantly greater than the original systems.

Complete results for the effects of colour clustering on both Cascade and SVM configurations of the Gesture Recognition System can be found in appendix B.1. These results motivated the retention of colour clustering only for the hand-5 gesture.

Colour Clustering was successfully implement in both systems. It was only found to be beneficial for the Hand-5 gesture in both systems. It significantly increases true positive performance with only marginal increases in false positive rates. In both cases the false positive rates remain below 0.5% for the Hand-5 gesture. As such it was only retained for the Hand-5 gesture in both systems. The benefits of recognition of the hand-5 gesture are

clear. Further work can perhaps bring the benefits of colour clustering to the other gestures. For example strong edges can be used to cut hand contours from background contours. Colour clustering successfully improved the performance of the Hand-5 gesture and is an interesting direction for future work that may continue to improve the segmentation of user hand contours.

6.6.3 Quantitative Results for K-Means Clustering

In the motif of using clustering, k-means clustering was investigated as a way of improving the SVM detector performance of the Gesture Recognition system. In the SVM configuration of the Gesture Recognition system described in chapter 3, a 2-stage SVM architecture was used. A 2-stage SVM was trained for each of the 3 recognized hand gestures.

Each stage of the SVM was trained with positive samples of a given gesture class as well as negative samples of a specific partion of the negative background patches dataset. The SVMs used will be summarized below, for further details please consult chapter 3.

Table 41 Summary of SVMs used in Gesture Recognition System

SVM	Stage	Positive Samples	Negative Samples
Hand 5	Stage 1	Hand-5 Samples	High High Low Gradient Samples
	Stage 2	Hand-5 Samples	High High High Gradient Samples
Hand L	Stage 1	Hand-L Samples	High High Low Low Low Gradient Samples
	Stage 2	Hand-L Samples	High High Low Low High Gradient Samples
Hand I	Stage 1	Hand-I Samples	High High Low Low Low High Gradient Samples
	Stage 2	Hand-I Samples	High High Low Low High Gradient Samples

These SVM stages were trained by randomly selecting around 1500 samples from the given gesture class and roughly 4000 samples from the given negative patch partition. These SVMs were retrained using k-means clustering to select the samples instead. 1500 k-means clusters were used on each gesture class, and 4000 k-means clusters were used

for each given negative sample partition. Then the closest sample to each cluster is selected. This results in 1500 samples from each gesture class and 4000 samples from each of the negative patch partitions that were used. These k-means selected samples were used to retrain the SVMs. This was done with the hope that k-means sampling would result in smaller better performing SVMs. Table 42 shows the size comparison of the k-means trained SVMs with the originals in terms of the number of SVM support vectors.

Table 42 K-means clustering SVM training size comparison

Gesture	SVM stage	Original Size	k-means Size
Hand-5	Stage 1	445	560
Hand-5	Stage 2	388	480
Hand-L	Stage 1	565	723
Hand-L	Stage 2	523	717
Hand-I	Stage 1	513	2000
Hand-I	Stage 2	526	2000

The sizes of the SVMs were actually bigger which is contrary to the expectation. The expectation was that by using k-means a more distributed sample set is achieved than by using random sampling. This means that there should theoretically be fewer outliers and therefore fewer support vectors. Due to k-means sampling resulting in bigger SVMs, a reduced k-means configuration was investigated. A fraction of the original k-means samples were selected using k-means clustering. The fraction was dependent on the size of the k-means SVM. The desired size of the SVM was 300-500 support vectors. Selecting a fraction of the k-means samples using k-means made sense because k-means sampling is uniformly distributed in the sample set.

Table 43 Reduced K-means SVM training size comparison

Gesture	SVM stage	Original Size	k-means Size	Reduced sample size positive (down from 1500 positive)	Reduced sample size negative (down from 4000 negative)	Percentage of original samples size	Desired reduced k-means SVM size	Actual reduced k-means SVM size
Hand-5	Stage 1	445	560	900	2400	60%	336	501
Hand-5	Stage 2	388	480	900	2400	60%	288	426
Hand-L	Stage 1	565	723	750	2000	50%	362	595
Hand-L	Stage 2	523	717	750	2000	50%	359	600
Hand-I	Stage 1	513	2000	375	1000	25%	500	388
Hand-I	Stage 2	526	2000	375	1000	25%	500	399

The results of the performance of the Gesture Recognition system are as follows:

Table 44 K-means SVM Gesture Recognition performance comparison with Validation Functions

	Original	k-means	reduced k-means
hand-5 TP%	26.33	13.34	1.235
hand-5 FP%	0.07281	0.03236	0.000
hand-L TP%	20.51	15.63	16.97
hand-L FP%	1.828	1.773	1.820
hand-I TP%	10.11	8.152	7.817
hand-I FP%	2.898	2.954	2.509

Table 45 K-means SVM Gesture Recognition performance comparison without Validation Functions

	Original	k-means	reduced k-means
hand-5 TP%	97.13	60.42	7.016
hand-5 FP%	81.89	43.80	21.55
hand-L TP%	91.82	100	63.13
hand-L FP%	81.13	97.46	69.30
hand-I TP%	100	100	100
hand-I FP%	96.26	98.09	96.03

The performance deteriorated significantly. However k-means allowed the entire negative sample partitions and all the positive class samples to influence the training of the SVMs. Additionally k-means offered a somewhat reliable way to control SVM size. However using random sampling clearly resulted in better detectors according to the experiments performed. Further work will have to be done to determine what benefits can be obtained from using k-means for SVM training. An interesting idea is to use k-means to partition

the negative background patches as opposed to using the magnitude of the HOG. This, and any other further k-means related ideas, are however beyond the scope of the thesis.

6.7 Summary

The Gesture Recognition system presented in Chapter 4 was used as the foundation of the Gesture Recognition with Colour Clustering Segmentation system that was presented in this chapter. The Colour Clustering Segmentation algorithm was connected to the Gesture Recognition system, replacing several modules from the original system. The performance of the colour clustering algorithm was tested in the same manner as the Gesture Recognition system presented in Chapter 4. It showed significant improvement for the hand-5 gesture performance, however it did not improve the performance of the other gesture classes. As such, colour clustering was retained only for the hand-5 gesture. In the vein of using clustering to improve gesture recognition performance K-means clustering was briefly investigated as a means to improve the training of the SVM detectors used by the gesture recognition system, with the goal of making smaller and better performing models.

The performance of the system deteriorated when k-means was used with the SVM training. K-means has some advantages such as allowing the entire dataset to make more of an impact in training, and also offering a way to somewhat control SVM size. The deteriorated performance means that further experiments will have to be performed to determine if any performance benefit can be attained by using k-means with SVMs for this problem. This can be an avenue for future research. Clustering did however show a clear benefit, although not with k-means, but in the form of the colour clustering segmentation algorithm which when used for the hand-5 gesture improved its recognition performance, and thus the overall recognition performance of the Gesture Recognition system.

Chapter 7 Conclusion

3 systems have been presented this thesis. They address problems of hand gesture recognition and hand tracking in video only vision based systems.

The first system is a Hand Gesture Recognition system that is capable of recognizing static hand gestures using a 2-stage approach that combines machine learning and contour shape processing. This system is designed to fulfill the thesis goals of real-time multi scale hand gesture recognition from a video or webcam input, and of a very low false positive rate for gesture recognition. These goals have been evaluated qualitatively and quantitatively with good results.

The second system is a Hand Gesture Recognition and Hand Tracking system that extends the first system and adds robust multimember hand tracking. This system is designed to fulfill the thesis goal of robust, articulated and steady hand tracking, that tracks the general hand location, the palm, and its extended visible fingers and their fingertips. This goal has been evaluated qualitatively and quantitatively. The hand tracking is quite good and has been successfully implemented as a user interface in applications for PCs and mobile devices. The quantitative testing has evaluated the ability of the tracking to adapt to changes in the user's hand, which it is successfully able to do. This system fulfills the thesis goal of combining both hand gesture recognition and hand tracking into a robust and highly usable system.

The third system is a Hand Gesture Recognition with Colour Clustering Segmentation system. This system extends the first system using a novel segmentation technique. This system is designed to improve upon the goal of real-time multi scale hand gesture recognition from a video or webcam input, specifically improving the hand gesture recognition rate. It significantly improves the recognition rate for the hand-5 gesture and is retained for use with the hand-5 gesture.

The first two systems have been fully described in academic papers and are ready to submit for publication. The third system is also a significant contribution that can be published later.

7.1 Future Research directions

Future work in this direction would include optimizing the Open Hand and General Hand models and exploring the power of 3D modeling solutions to solve gesture recognition and tracking problems. Additional areas of research include building a complete Gesture Recognition system with MobileNets, improving the training of the SVM detectors, exploring the implications of good hand detection and segmentation on human action recognition, and improving hand tracking with a neural network model.

The Open and General Hand models have been proposed as a result of a lot of trial and error experimentation. Their formulation, specifically the tests that they apply to understand the hand contour shapes and their finger and palm characteristics, works well to identify the fingers and the palm and has already been shown to aid in gesture recognition and hand tracking. However because of their hand crafted nature they are not optimal. Optimization of the proposed models, specifically finding the best value for each parameter of the model, can be done with either machine learning, or numerical optimization, as well as a large hand contour dataset. This would likely improve the performance of the model and would serve to mathematically converge the model. These models provide the discrimination power of the hand gesture recognition system. Without them the false positive detections become too numerous for the system to be usable. Having an optimized version of these models would allow even greater confidence in finger and palm localization, and this would allow the gesture recognition system to become more easily scalable to a larger variety of static hand gestures.

3D model fitting techniques present interesting new ways to solve hand tracking and pose estimation problems. With some machine learning detection 3D modeling can also be used to solve the gesture recognition problem as well. Extending the gesture recognition methods presented in this thesis to use 3D modeling techniques and depth sensors, or exploring new 3D methods on their own, is an interesting direction for future research which would improve the control and responsiveness that can be achieved in a gesture based user interface.

The MobileNets comparison system provided an interesting comparison between the benefits and tradeoffs of using CNNs compared to other models. It would be interesting to investigate how well MobileNets would work for recognizing hand gestures using a proposal generator architecture to allow it to run as a real-time system. This would allow for further analysis of benefits and tradeoffs of CNN models.

The best way of training the SVM model for gesture recognition is not entirely certain at this point. There are further investigations that can be done by partitioning the dataset using k-means clustering as opposed to the magnitude of the HOG. This k-means dataset partitioning will produce new data partitions which would have to be used to train SVMs with both random sampling and k-means sampling to determine the best way to train multi-stage hand gesture detection SVMs. Also the effect of only using a subset of a given SVM's support vectors can be investigated to determine what the tradeoffs are The in terms of accuracy vs. computational load at run time. The overall goal is to determine how to train smaller and more accurate SVM models and other techniques to accomplish this will no doubt present themselves with further investigation.

Hand and gesture detection is a very important part of being able to understand human behavior. Almost every human action involves the hands in some way. Even a runner moves their hands in a particular way while running and in a different way while walking. Therefore having good hand and gesture detection becomes an important component of human action recognition. An interesting paper by Bambach et al.[128] achieves human activity recognition using segmented hands and a CNN model. Applying hand detection and hand gesture recognition approaches to human action recognition will result in better action recognition approaches which are of particular interest in video summarization and video surveillance fields. The work of Bambach et al[128] shows that accurate hand segmentation and hand gesture recognition is part of solving the human action recognition problem.

The current hand tracking method is robust and flexible, however there is some room for improvement using machine learning. The contour based tracking relies on the segmentation provided by the registration of the user's hand. The nearest contour in successive frames is selected with a constraint metric and this is how contour based

tracking is accomplished. While the segmentation is generally good there are situations where skin colour like objects produce contours near the hand location in the previous frame which can confuse the algorithm and cause tracking to fail. The same situation can occur with occlusions with skin colour like objects. Using a machine learning model, CNN or otherwise, can alleviate this problem. A model can be trained to distinguish between hand contours and the skin colour like object contours using a combination of segmentation data and raw input images. This can then select the best contour and thus resolve the problems of occlusion and tracking non-hand objects. This research direction would further improve the quality of the hand tracking and make it even more robust.

7.2 Concluding thoughts

This thesis research has advanced what is possible with hand gesture recognition and the presented systems allow for the creation of powerful gesture based user interfaces for a variety of applications. The research has been implemented for user interfaces for several applications. Earlier work in finger mouse controls showed that the hand tracking was capable of providing a steady enough performance for mouse input. GR Paint showed a paint application for mobile devices which used the hand tracking algorithm presented in this thesis as input. A controller for the first person video game HALO was made. The game demands quick user response and stable aiming controls which was achieved by the robust contour based tracking and the template fingertip tracking respectively. A static hand gesture based user interface for mobile devices for the car market was created as part of a collaboration with a company called Klashwerks. The interface performed well on PC proving the concept of the technology. Further work will still be required to port it to their mobile device. The interest and the potential are both there. The systems presented in this thesis will contribute to making hand gesture recognition interfaces a robust and viable option for computer applications and human computer interaction.

Hand gesture recognition and hand tracking in 2D video is a very important research direction. It is a viable option for cheap and versatile user interfaces. Using only video not only reduces cost compared to RGB-D sensors, but it also allows gesture recognition and hand tracking to be applied in situations where depth information is unavailable. Furthermore hand detection and gesture recognition are important elements in solving the

human action recognition problem. The locations of hands and what they are doing is an intrinsic part of many human activities and human activity is a primary interest in video surveillance. Far from being a fad or a novelty the commercial applications for user interfaces and surveillance will continue to drive research interest in this field for many years to come, and research contributions from a large variety of fields will continue to lend a hand.

Appendix A.1 The Open Hand Contour Model

The open hand model is a series of geometric tests involving point calculations and proportionality ratios that are applied to a given contour shape and its corresponding convexity defects. The model used for hand-5 gesture recognition. The model takes a sample point and a back projection as input and outputs whether a contour was found that has a valid hand-5 gesture shape. The hand-5 gesture is an open hand with 5 visible extend fingers.

Applying the open hand model

A potential hand-5 shape must first have its convex hull calculated in order to then determine its convexity defects. The inner most point of each of the convexity defects is found and kept as the set of inner points. The depth lines are also calculated by finding the midpoint of the appropriate convex hull lines and drawing a line all the way to the corresponding inner point. The convexity defect with the longest depth line is found and then every other convexity defect is compared to it and a given convexity defect is discarded along with its calculated attributes if it's depth line is smaller than the largest convexity defect depth line divided by the *big line threshold*. The *big line threshold* is a ratio that the hand model uses and all ratios and set values of the open hand model indicated by their *italicized* font can be found in a table at the end of this section. A potential hand shape must have between 3 and 8 kept convexity defects which correspond to the *min number of big lines* and *max number of big lines* values of the hand model.

The next step is to find a minimum enclosing circle around the inner points. The center of the circle is pCenter, and the corresponding radius is pRadius. pCenter represents the center point of the potential palm of the contour, and pRadius represents its radius. The sample point provided must fall within the minimum enclosing circle. This speeds up computation because contours that do not overlap with the sample point are not considered.

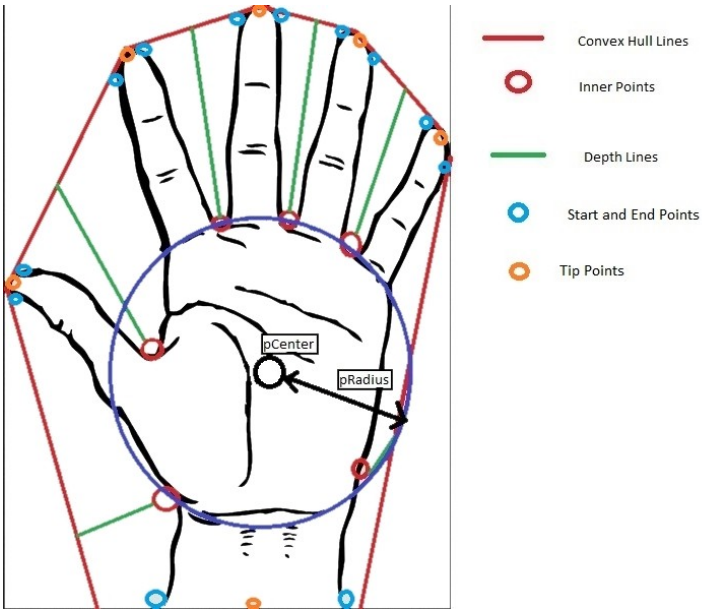


Figure 95 Finding the convexity defects and the palm of the contour

Each of the kept convexity defects has a start point and an end point describing where the shape of the contour begins to depart from the convex hull. These start and end points are paired off such that an end point of one kept convexity defect is paired with the start point of the next kept convexity defect. The midpoints between these pairs are taken as the tip points. The distances between each pair of start and end points and their corresponding inner points are taken as the tip distance and the base distance respectively. The angle between each tip point and its adjacent inner points is calculated and recorded as the tip angle.

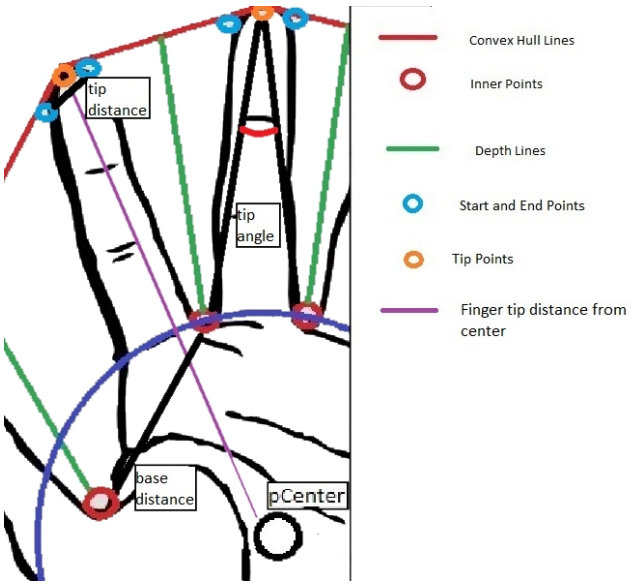


Figure 96 Finding the valid tip points of the contour

The finger base point of a given tip point is defined as the midpoint between its corresponding inner points. The shifted finger base point is found by finding the point along the minimum enclosing circle around the inner points that has the same angle as the finger base point relative to the point pCenter. The finger length of the potential finger is the distance between a tip point and its shifted finger base point, and the point halfway between them is taken to be the knuckle point. The distance between the knuckle point and pCenter is also calculated.

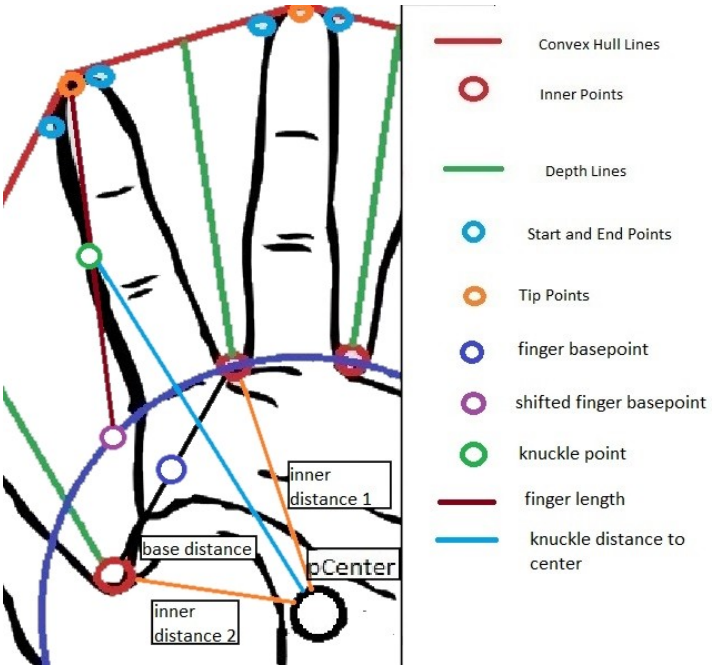


Figure 97 More valid finger point tests to evaluate finger orientation

Once all of these variables and points have been found the hand model evaluates the following set of relationships. If these relationships hold for a given tip point then it is declared a valid finger point. The ratios defined in this set of relations can be found in the table at the end of this section.

$$\text{tip angle} < 180° / \textit{angle divisor} \quad (1)$$

$$\text{tip distance} * \textit{tip distance factor} < \text{base distance} \quad (2)$$

$$\text{finger tip distance from center} < \textit{max finger distance factor} * \text{pRadius} \quad (3)$$

$$\text{inner distance 1} < \textit{inner distance radius factor} * \text{pRadius} \quad (4a)$$

$$\text{inner distance 2} < \textit{inner distance radius factor} * \text{pRadius} \quad (4b)$$

$$\text{finger length} > \textit{min finger radius factor} * \text{pRadius} \quad (5a)$$

$$\text{finger length} <= \textit{max finger radius factor} * \text{pRadius} \quad (5b)$$

knuckle distance to center > *min knuckle radius factor* * pRadius (6)

Once the valid finger points of a contour have been found the last element to consider is the number of consecutive finger points. Valid finger points will not always occur in consecutive pairs of convexity defects and pairs of adjacent valid finger points will not always have realistic positions relative to each other. The angle between pairs of adjacent valid finger points and their intermediary inner point is calculated and this consecutive finger angle is evaluated with the following relation.

consecutive finger angle < 180° / *consecutive finger angle divisor* (7)

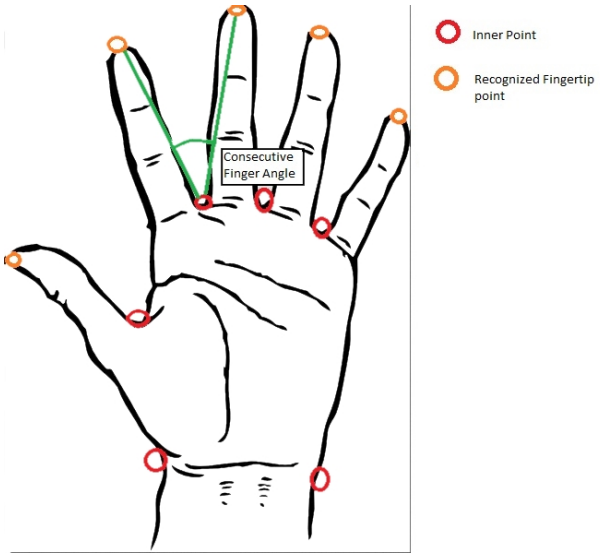


Figure 98 Consecutive finger angle tests

If it holds then both points are recognized as consecutive finger points. Although a given point can be consecutive with both of its neighbouring adjacent points a twice recognized point is treated the same way as a once recognized point by the hand model.

A contour shape is declared to be a hand-5 shape if it has either 4 or 5 valid finger points or at least 3 consecutive finger points. These conditions are designed to ensure quick recognition in the presence of noise. If 1 or 2 fingers of a contour shape are cut off during segmentation due to varying lighting conditions then the hand-5 shape and its

corresponding CrCb histogram can still be declared usable under the flexible conditions of the open hand model. This makes the hand model more adaptive, faster and results in the Gesture Recognition system being more user friendly.

If a contour shape is found to be a hand-5 shape then the open hand model returns true, and the contour shape, its palm, and its fingertip location information can be saved for later use.

Table 46 Open Hand Contour Model Ratios and Values

Ratio	Ratio value
big line threshold	15.0
min number of big lines	3
max number of big lines	8
angle divisor	4.0
tip distance factor	1.0
max finger distance factor	4.0
inner distance radius factor	1.5
min finger radius factor	0.8
max finger radius factor	4.0
min knuckle radius factor	1.1
consecutive finger angle divisor	4.0

Appendix A.2 The Generalized Hand Contour Model

The generalized hand contour model is used for validating the hand-L and hand-I gestures and it is also used for hand tracking. The generalized hand contour model was determined experimentally with online testing with video input from a web camera. The hand gesture shape validation and finger counting that is provided by the general hand contour model helps to validate the hand-L and hand-I gestures, and discard false positives. The hand tracking algorithm calls the generalized hand contour model when tracking the contour in the current frame. By calling the model at each frame the tracking algorithm can achieve real-time and robust tracking of a user's hand.

The model takes a sample point and a back projection as input and outputs the visible extended finger count of a valid hand contour if one is found. It can also give the valid hand contour itself, the visible fingertip locations and palm information.

Applying the generalized hand model

The general hand contour model takes the input back projection and finds and extracts all of the contours. It treats each of these contours as potential hand shapes.

A potential hand contour shape is only evaluated with the generalized hand model if it has at least as many points as the *minimum contour size*. A potential hand shape has its convex hull calculated and its convexity defects found. The inner most point of each of the convexity defects is found and kept as the set of inner points. The depth lines are also calculated by finding the midpoint of the appropriate convex hull lines and drawing a line all the way to the corresponding inner point.

The convexity defect with the longest depth line is found and then every other convexity defect is compared to it. A given convexity defect is discarded along with its calculated attributes if its depth line is smaller than the largest convexity defect's depth line divided by the *big line threshold*. The *big line threshold* is a ratio that the generalized hand model uses in its analysis of the hand. Unlike the open hand model any number of retained convexity defects is acceptable. All ratios and set values of the generalized hand model indicated by their *italicized* font can be found in a table at the end of this section.

In the generalized hand model the palm must be determined more precisely than in the open hand model. Therefore calculations are performed to determine which inner points are to be kept as the palm points of a hand contour. Hands often share colours with other aspects of a user's body. Filtering the inner points of a hand shape is done in order to avoid confusing the palm with attached body parts such as the forearm, elbow and bicep. The inner points of the kept convexity defects are used to find a minimal enclosing circle. The radius of this circle is referred to as the current radius. The center of this circle is found and it is kept as the minimal enclosing circle center.

The average of all the inner points is also calculated. Each inner point has a distance to the average of all inner points and a distance to the center of the minimal enclosing circle (test distance). The general idea is that an inner point is declared to be a palm point if it is closer to the average of all the inner points than the center of the minimal enclosing circle; however the model applies a slightly more elaborate set of conditions.

For each inner point this condition is evaluated:

test distance > *current radius factor* * current radius (1)

If (1) is true then an inner point must satisfy these conditions to be declared a palm point:

test distance factor * test distance > distance to average of inner points (2)

test distance < *depth line factor* * depth line (3)

If (1) is false then an inner point must only satisfy condition (3) in order to be a palm point.

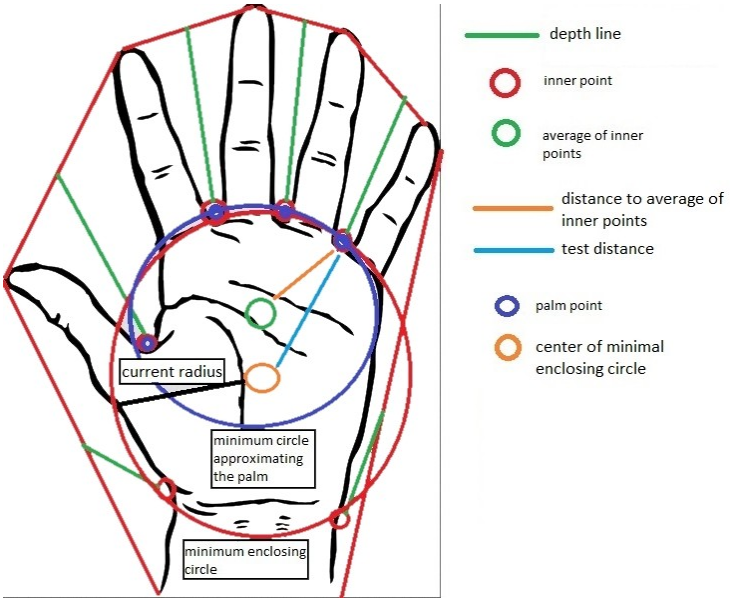


Figure 99 Finding the palm of the contour with the general hand model

Additional filtering occurs of the palm points in order to remove outliers that do not form a general circular shape. The angle between 3 consecutive palm points must obey the following relation otherwise the palm point located at the angle is discarded.

angle between 3 consecutive palm points > 180° / *palm filter angle divisor* (4)

The results of the filtering are only used as the palm points for the subsequent sections of the generalized hand model if there are at least 5 palm points left after the filtering, otherwise the removed points are added back to the set of palm points. Then the minimum enclosing circle of the palm points can be found and this is the contour's palm. The circle center is the contour's palm center point and the radius is the contour's palm radius.

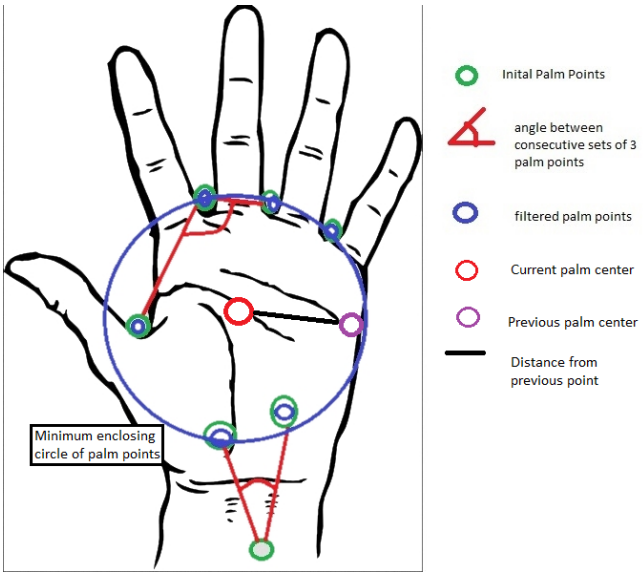
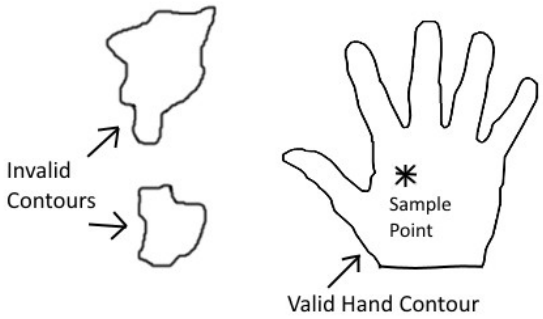


Figure 100 Filtering the palm

Once the palm of a given hand contour has been found it is important to determine if that hand contour is in a realistic place relative to the sample point, or relative to the tracked hand contour from the previous frame. This is where a distance constraint can be imposed on the general hand model.

It can be set to "the sample point must be inside a contour for it to be valid" when the general hand model is used to count the number of visible extended fingers of a potential hand contour at a given sample point. This constraint is referred to as the finger counting distance constraint. This is the case when the Gesture Recognition system calls the general hand model to count the number of fingers in a potential hand gesture detection. Only the contour with the sample point inside it is kept as valid, if such a contour exists.



The finger counting distance constraint is applied to some contours. Only the contour with the sample point inside it is valid.

Figure 101 Finding a valid contour for the validation functions in the Gesture Recognition system

The distance constraint can also be set to the following condition, which is the case when the hand tracking calls the general hand model:

distance from previous palm point < *reference radius range* * original radius (5)

This is the distance constraint used for hand tracking.

In this case the general hand model is called with the previous palm center point of the tracked hand contour as the sample point, which is referred to as the previous palm point when the general hand model is called by the tracking. This previous palm point is the palm center from tracked hand contour in the previous frame, or from the contour of the validated hand gesture detection which initiated the tracking. When the general hand model is used for hand tracking the tracking must also provide the palm radius of the hand contour of the validated gesture detection which initiated the tracking. This palm radius is referred to as the original radius and is another component of the hand tracking distance constraint.

The general hand model uses the palm center of a given potential hand contour and calculates the distance from the previous palm point. It evaluates the distance constraint for hand tracking condition to determine if the contour that is being analyzed is valid. The *reference radius range* value used in the constraint is defined in the table at the end of the general hand model section.

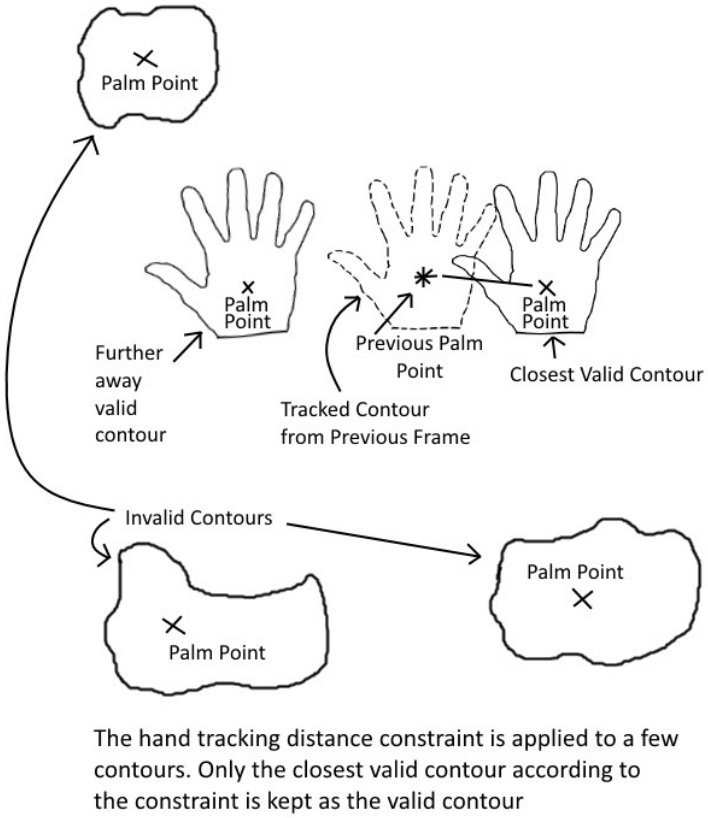


Figure 102 Finding a valid contour for the Hand Tracking Algorithm

If this condition is met the general hand model continues the analysis of the given contour hand shape. If the condition is not met then the current hand contour shape is declared invalid and the algorithm proceeds to the next contour if one is available, otherwise it terminates. When there are multiple valid contours the closest one is kept. Whichever distance constraint is used, if a valid contour is found the algorithm proceeds to the next step. Otherwise it terminates returning a flag that no valid contours were found.

The next part of the generalized hand contour model is applied to determine if any fingers are visible for the given hand shape. Each of the retained inner points has its corresponding convexity defect analyzed to find the start and end points, and these are paired with the start and end points of the adjacent convexity defects and then the average of these pairs are found and taken as the tip points in the same way that was done in the open hand contour model of the algorithm. In order to evaluate the conditions that

determine whether a given tip point represents a fingertip the angle between the tip point and its adjacent inner points must be found.

Note that the inner points are used in the fingertip angle calculations and not the palm points which are a subset. This means that a finger shape does not have to explicitly touch the palm point circle with both of its inner points. The reason for this is to ensure that inner points that were discarded in order to find the best approximation of the palm can still yield fingertips. Because the location of the palm is an approximation good data could potentially be discarded if the inner points are not used in the finger tip calculations.

Two position dependant angles are calculated. The finger orientation angle is the angle of the given tip point, the finger base, which is the average of the adjacent inner points, and the bottom of the frame. The position angle is the angle between the finger base, the center of the palm circle and the top of the screen.

The following condition must be met by each tip point:

$$\text{Fingertip angle} < 180°/ \textit{finger angle divisor} \quad (6)$$

The next two conditions for each tip point are related to each other:

$$\text{If: position angle} < 60° \quad (7a)$$
$$\text{then: finger orientation angle} > 120° \quad (7b)$$

A tip point must satisfy condition (6) and if it satisfies condition (7a) it must also satisfy condition (7b) otherwise it is discarded.

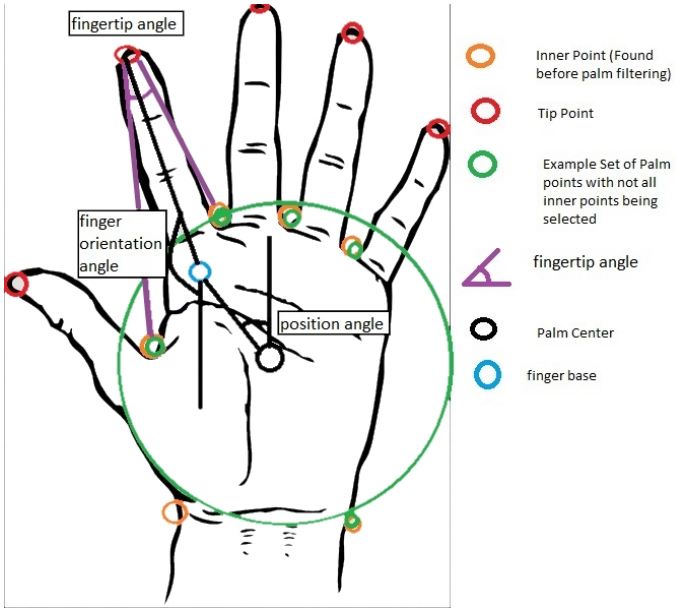


Figure 103 Finger orientation tests

For a tip point to be considered a valid finger point it must have at least 1 of its inner points within the minimal enclosing circle of the palm points. The distances between a tip point's inner points and the center of the palm circle are calculated and recorded as inner point distance 1 and 2, and either one of the following conditions must be upheld.

$$\text{inner point distance 1} < \text{current radius} \quad (8a)$$

$$\text{inner point distance 2} < \text{current radius} \quad (8b)$$

The distance between the inner points of a tip point is the base distance. The distance between the start and the end points of the convexity defects that were used to find the tip point is referred to as the tip width. The tip width must obey both of the following conditions otherwise the tip point is invalid. This is done to eliminate wrist and forearms being considered fingers because they are attached to the palm.

$$\text{tip width} < \text{base distance} * \textit{tip width range} \quad (9)$$

$$\text{tip width} < \text{current radius} * \textit{tip width palm radius range} \quad (10)$$

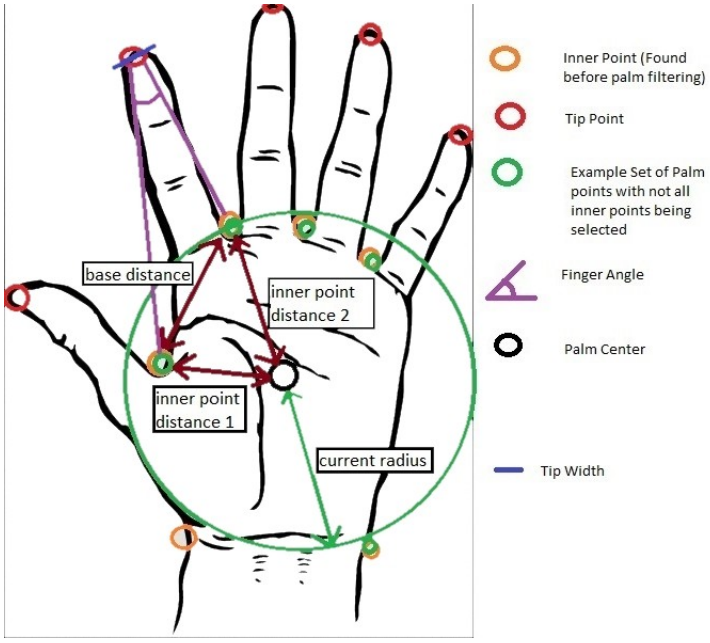


Figure 104 Finger base structure tests

Several more proportions are evaluated using the finger length, which is the distance between the tip point and the finger base, the finger to center distance, which is the distance between the tip point and the palm center, and the finger base to center distance (finger base to palm center). The proportions are evaluated for each retained tip point using the following three conditions which must all hold, otherwise the tip point is discarded from the evaluation.

$$\text{finger base to center} < \text{current radius} * \text{finger base palm radius range} \quad (11)$$

$$\text{finger length} > \text{current radius} / \text{finger length palm radius divisor} \quad (12)$$

$$\text{finger to center} > \text{finger to center minimum range} * \text{current radius} \quad (13)$$

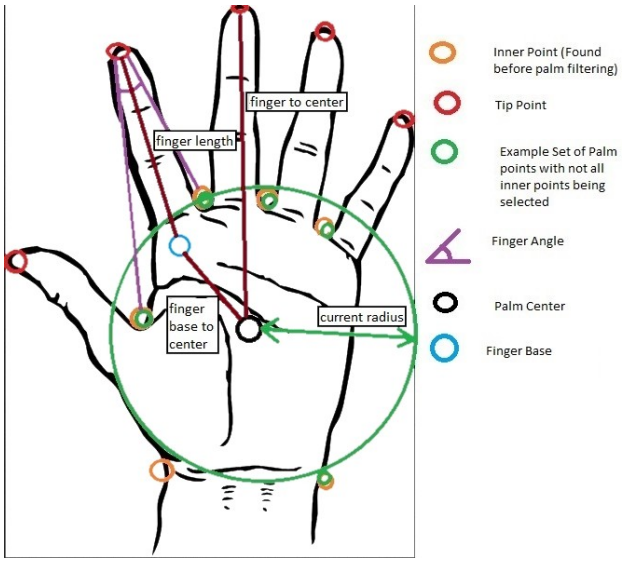


Figure 105 More finger orientation tests

The final set of conditions is evaluated to determine whether or not a retained tip point has a realistic orientation and whether it is far enough from its neighbours. The angle between a given retained tip point, the palm center and the top of the frame is taken as the orientation angle. The following condition must hold for a tip point to be retained. This was done to ensure that the hand remains upright within a reasonable range.

$$\text{orientation angle} < 105° \quad (14)$$

If the evaluated tip point passes this condition and there are no other declared valid fingerpoints then it becomes the first valid finger point. Otherwise if the tip point is a valid fingerpoint under all prior conditions it must be evaluated against the previously declared valid finger point to ensure that it is not too close to the existing set of valid finger points. It will be added to the valid finger point set if its proxy distance, which is the distance between it and the previously declared valid finger point holds in the following condition:

$$\text{proxy distance} < \text{current radius} / \textit{proxy distance palm radius divisor} \quad (15)$$

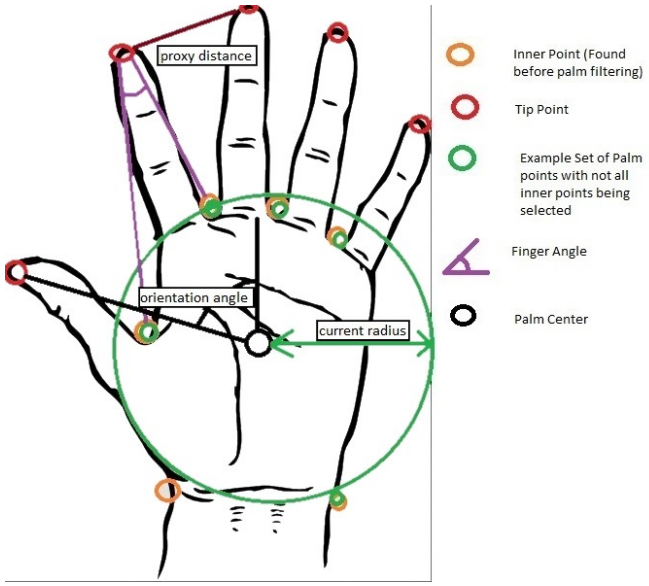


Figure 106 Orientation angle tests relative to absolute hand position in frame

If the generalized hand model is successful in finding a valid contour it will output the visible extended finger count of the contour. It can also give the valid hand contour itself, the visible fingertip locations and palm information. If no valid contour is found the algorithm will return that no valid contour was found.

Table 47 Generalized Hand Contour Model Ratios and Values

Ratio	Value
minimum contour size	120.0
big line threshold	10.0
inner distance factor	5.5
palm filter angle divisor	6.0
current radius factor	0.6
test distance factor	2.5
depth line factor	5.5
reference radius range	2.0
finger angle divisor	3.5
tip width range	3.5
tip width palm radius range	2.5
proxy distance palm radius divisor	4.0

Appendix B.1 Configuration Results for the Gesture Recognition with Colour Clustering Segmentation System

In order to investigate the effects of colour clustering, on the performance of all 3 gestures for both Cascade and SVM Gesture Recognition systems, a series of experiments were conducted under different configurations. The results of these experiments drove the decisions taken to only retain colour clustering for the hand-5 gesture in both configurations.

The results for the Cascade system are as presented in Table 48.

Table 48 Results of Colour Clustering Testing for Cascade System

Colour Clustering Testing Cascade System (Y = Yes, N = No)

		hand-5			hand-L			hand-I			Hand-5		Hand-L		Hand-I	
test ID	test number	unimodal	ROI	focused ROI	unimodal	ROI	focused ROI	unimodal	ROI	focused ROI	TP %	FP %	TP %	FP %	TP %	FP %
old	0	N/A old sampling			N/A old sampling			N/A old sampling			31.67	0.04854	25.58	0.8943	16.19	1.008
t11	1	Y	Y	Y	Y	Y	Y	Y	Y	Y	39.38	0.04854	34.37	5.358	22.05	3.653
t12 (Conf 4)	2	Y	Y	Y	Y	Y	Y	Y	Y	Y	39.48	0.04854	34.37	5.358	22.05	3.653
t13	3	Y	Y	Y	N	Y	N	N	Y	N	39.38	0.04854	21.06	3.130	14.46	2.525
t14	4	Y	Y	Y	Y	Y	N	Y	Y	N	39.38	0.04854	22.65	3.264	13.90	2.668
t15	5	Y	Y	Y	N	N	Y	N	N	Y	39.38	0.04854	20.82	3.154	15.30	2.096
t16	6	Y	Y	Y	Y	N	Y	Y	N	Y	39.38	0.04854	19.84	2.958	15.08	2.041
t17	7	Y	Y	Y	N	Y	Y	N	Y	Y	39.38	0.04854	35.29	5.546	23.45	3.645
t19	8	N	Y	Y	N	Y	Y	N	Y	Y	36.51	0.08090	35.10	5.578	23.45	3.637
best	9	Y	Y	Y	N/A old sampling			N/A old sampling			39.38	0.04854	25.64	0.8473	16.19	1.008

The Colour Clustering performed well for the hand-5 gesture from test 1. Test 2 was an attempt to improve the hand-5 performance by adding another structural element to the erosion and dilation which was a 3x3 cross with 1 iteration. It did not significantly improve the performance of the hand-5 gesture and so the element was not used any further and was not adopted for the final colour clustering algorithm. The remaining tests attempted to improve the performance of the Hand-L and Hand-I gestures which despite having increased true positive rates had significant increases in false positive rates in test 1 of roughly 4.5% for hand-L and roughly 2.7% for hand-I. Colour clustering took them above 3% false positive rate which was deemed to be too high especially since the worst false positive rate for the SVM configuration of the Gesture Recognition system, which was the less performing of the two configurations, was roughly 2.9%. Test 3 was done to see if removing unimodal filtering and the focused ROI for the 2 gestures would improve performance. It reduced true positive rates and increased false positive rates. Test 4 added unimodal filtering back to the Hand-L and Hand-I gestures with negligible improved in the hand-L true positive rate compared to test 3. However it deteriorated all the other quantities and the true positive rate of hand-L in the original configuration was still better. Using the focused ROI with and without unimodal filtering for the 2 gestures in tests 5 and 6 also did not improve upon the original configuration. Test 8 was a verification of unimodal filtering for colour clustering with the hand-5 gesture. The test shows that deactivating the unimodal filtering deteriorates both true and false positive performance. The best configuration was tested in test 9. It uses colour clustering with unimodal filtering and both the full ROI and the focused ROI for the hand-5 gesture, and it uses the old sampling method described in Chapter 3 for the hand-L and hand-I gestures. This configuration yields the same performance for hand-L and hand-I gestures as the old Cascade Gesture Recognition configuration while significantly improving the performance of the hand-5 gesture.

The results for the SVM system are presented in Table 49.

Table 49 Results of Colour Clustering Testing for SVM System

Colour Clustering Testing SVM System (Y= Yes, N = No)

		hand-5			hand-L			hand-I			Hand-5		Hand-L		Hand-I	
test ID	test number	unimodal	ROI	focused ROI	unimodal	ROI	focused ROI	unimodal	ROI	focused ROI	TP %	FP %	TP %	FP %	TP %	FP %
old	0	N/A old sampling			N/A old sampling			N/A old sampling			26.33	0.07281	20.51	1.828	10.11	2.898
t_s1	1	Y	Y	Y	Y	Y	Y	Y	Y	Y	30.53	0.4288	35.90	20.48	22.89	17.05
t_s2	2	N	Y	Y	N	Y	Y	N	Y	Y	28.01	0.2427	32.17	12.46	68.62	26.28
Best	3	Y	Y	Y	N/A old sampling			N/A old sampling			30.53	0.4288	20.51	1.797	10.16	2.866

The colour clustering performed well in the SVM system for the Hand-5 gesture in test 1. It however did not perform well for the Hand-L or Hand-I gestures adding an unacceptable number of false positives. In test 2 the effects of removing the unimodal filtering were investigated. It somewhat diminished the true positive rate of the Hand-5 gesture but did slightly reduce the false positive rate as well. It decreased the false positive rate of the Hand-L gesture but it still remained at unacceptable levels. It did however increase the true positive rate. The Hand-I gesture performance in test 2 suffered dramatically. While removing unimodal filtering significantly increased the true positive rate, the false positive rate increased even further than in test 1. After reviewing these results it was decided that Colour Clustering should only be retained for the Hand-5 Gesture. Unimodal filtering increased the true positive rate while retaining a low false positive rate with only a marginal 0.18% increase when compared to the non unimodal

filtering case, so it was also retained. The colour clustering increased the false positive rate too much for the Hand-L and Hand-I gesture, and that is why it was not retained for these gestures in the best Colour Clustering SVM Gesture Recognition system. Comparing tests 1 and 2 of the SVM Colour Clustering system to tests 1 and 8 of the Cascade Colour Clustering system reveals why further SVM colour clustering configurations were not investigated. Tests 1 and 8 of the Cascade system have the same configurations of colour clustering as tests 1 and 2 of the SVM system respectively. In test 1 and test 8 of the Cascade system colour clustering adds approximately 4.4% to the false positive rate of the Hand-L gesture in both cases when compared to the original Cascade system. The Hand-I false positive rate also increases by approximately 2.6% in both cases when compared to the original system.

For the SVM system testing colour clustering adds between approximately 11% and 18% for the Hand-L gesture and between approximately 14% and 23% for the Hand-I gesture. In the Cascade system investigating further sub configurations for the Hand-L and Hand-I gestures, such as only using the focused ROI or only the ROI, continued to result in false positive rates that were deemed too high. Investigating sub configurations did not yield in good results. In comparison, the SVM system with complete colour clustering and colour clustering without unimodal filtering increase the false positive rates of the Hand-L and Hand-I gestures significantly more than these configurations did for the Cascade system. Investigating sub configurations did not improve the performance of colour clustering for the Hand-L and Hand-I gestures in the Cascade system where colour clustering performed better for those gestures than in the SVM system. Therefore sub configurations of Hand-L and Hand-I were not investigated in the SVM system where colour clustering performed worse for those gestures than in the Cascade system.

It should be noted that for both Cascade and SVM systems while the effects of sub configurations of ROI were not explicitly tested for the hand-5 gesture, using both ROI and focused ROI did not result in high false positive rates. It was deemed useful for the algorithm to acquire hand shape contours with both ROIs for hand-5 because both ROIs results in more contours that the hand model can use to find the hand-5 hand shape.

Glossary of Terms and Acronyms

Adaboost - A learning algorithm proposed by Viola and Jones [70] in which a series of classifiers are linked together to work consecutively. Also known as a Cascade. Sometimes the term *Cascade of <classifier>* is used to refer to other classifiers that mimic the structure of Adaboost. For example Cascade of SVMs refers to a Support Vector Machine Classifier where several SVMs are run in sequence mimicking the Adaboost structure.

Back projection - Another term for image segmentation with a colour signature. See *Segmentation.* Back projection in the context of this thesis refers to the fact that a colour signature was produced from a specific are in an input image and then the extracted colour information is projected back onto the input image to segment it into white areas representing colours within the information range, and black areas representing colours not present in the colour signature.

Cascade - Can refer to the Adaboost algorithm, or if it appears as *Cascade of <classifier>* refers to a classifier that is mimicking the Adaboost structure. See *Adaboost.*

Classifier - A model or function that takes input data, in the case of this thesis usually an image, and outputs a classification of said data. It can be created or "trained" with statistical learning and a dataset. It can also be set manually by a designer.

CNN - Convolutional Neural Network. See *Convolutional Neural Network.*

Colour Signature - See *Histogram,* specifically the 2D histogram sub section.

Contour - A series of linked points that define the boundary of a shape in an image. In thesis thesis contours are taken of the segmented white areas from binary image segmentations. See *Segmentation.*

Convolutional Neural Network - A class of deep learning models that use convolutional filters in order to extract features from images. These features are further processed to classify the input images. The convolutional filters define the category of models and set it apart because it allows the networks to work with images directly.

Deep Learning - A term used to describe neural networks that have 1 or more middle layers. These layers allow the learning of abstract features extracted from several layers of transformations. These features are not as intuitive as features extracted with just 1 transformation. The abstract features result from these so called deep layers.

Feature - A feature is any type of quantifiable information that can be systematically stored about an image. These generally capture some sort of characteristic about a given image. Features can include colour information, shape information, texture information, or any other metric derived from a given input image. Feature is a general term that is used to encompass all of the different types of information, usually numeric, that are extracted from images.

Histogram - This term has 2 different meaning depending on whether or not it is stated as a 1 dimensional (1D) histogram, or a 2 dimensional (2D) histogram.

1D histogram - This is a traditional interpretation of a histogram. The x axis represents the attribute that has chosen to be logged, for example red in an RGB image. The y axis represents how often a particular value occurs. A histogram is essentially a tally of the frequency of occurrence of a attribute at specific values.

2D histogram - This is a more advanced type of histogram which has 2 axis that represent attributes that have been chosen to be logged. If a 1D histogram is a tally of the frequency of occurrence of an attribute, then a 2D histogram is a tally of the frequency of joint occurrences of both attributes at the values specified by the 2 axis space of these attributes' value ranges. There is a 3rd axis which in theory logs how many times the joint occurrences of the attributes occur. This is not needed for the methods described in this thesis and the 3rd axis is reduced to a binary value for each joint set of attribute values. 1 indicates that it occurs, and 0 indicates that it does not. This is also called a colour signature because it is used to log which joint occurrences of colour value pairs are present in an image. Typically colour images have 3 colour channels. In practice only 2 of these channels are used. However this logging technique can be extended to have 3 axis that represent attributes.

HOG - Histogram of Oriented Gradients. This is a type of feature that is generated from an input image. It is done by dividing it into a grid of sections and then calculating the general pixel difference and direction of the difference in each section. This is called the gradient and the magnitude and direction of these gradients are logged in a vector by frequency of occurrence. This vector is the output feature of the HOG calculation.

Neural Network - A context sensitive term. It could refer to all types of neural networks, or more specifically just CNNs.

Qualitative - Some things are difficult to be evaluated numerically. This is either because they are difficult to describe in terms that can be evaluated to produced a numeric result, or because doing so would be a very laborious process. In these instances it is useful to evaluate the quality a given system in terms of demonstrating what it is capable of doing with a user interaction demonstration. Qualitative in this thesis refers to evaluation of the algorithms developed in a live online context with a user using the system. This offers visual examples of how the system works which facilitates understanding. It also demonstrates what can be accomplished with the system by a user. It is quite difficult to objectively evaluate user appeal, and usability of a system, and visual qualitative examples show the performance of the system in an intuitive way. Qualitative analysis is also complemented with quantitative numerical testing.

Quantitative - This references to attributes and results that can be evaluated to produce a numeric result. Quantitative testing is used to give an objective measure of performance of the systems presented in this thesis.

RGB - Basic colour space for computer images. It has 3 channels, R stands for red, G stands for green, and B stands for blue. In practice input images can also be coded in BGR which is the reverse of RGB. Care should be taken when reading images from a camera not to assume that they are automatically coded in RGB.

Segmentation - A binary image created by taking an input colour image and processing it with a given colour signature or 2D histogram. Colour value pairs which occur in the image which are also present in the colour signature are retained and others are discarded. The result is that pixels with retained colour value pairs are marked as 1 and all other

pixels are marked as 0. Thus dividing the input image into black sections and white sections.

Support Vector Machines - A learning algorithm that is used in this thesis work. Proposed by Vladimir Vapnik and his colleagues at AT&T Bell Labs. [100], [101]

SVM - Support Vector Machine.

Training - The process of making a classifier with statistical learning algorithms and a dataset. The degree of complexity and inputs required varies with the algorithm.

Testing - The processes of evaluating a system either qualitatively or quantitatively. It varies with the context. See *Qualitative,* and *Quantitative.*

UI - User Interface

User Interface - Any hardware controls or software applications that allow a user to interact with, and issue commands to, a computer or machine. In the context of this thesis a user interface or UI usually refers to a software application that uses a camera as input to allow a user to interact with a computer.

YCbCr - A colour space with 3 channels. Y is the Luminance, Cr is the chrominance of red, and Cb is the chrominance of blue. In this thesis Cr and Cb channels are used to calculate colour signatures of images.

References

[1] G. Simion, V. Gui, and M. Otesteanu, “Vision based hand gesture recognition: A review,” *Int. J. Circuits, Syst. Signal Process.*, vol. 6, no. 4, pp. 275–282, 2012.

[2] J. Galván-Ruiz, C. M. Travieso-González, A. Tejera-Fettmilch, A. Pinan-Roescher, L. Esteban-Hernández, and L. Domínguez-Quintana, “Perspective and evolution of gesture recognition for sign language: A review,” *Sensors (Switzerland)*, vol. 20, no. 12, pp. 1–31, 2020.

[3] S. Goldin-Meadow, “The role of gesture in communication and thinking,” *Trends Cogn. Sci.*, vol. 3, no. 11, pp. 419–429, 1999.

[4] R. M. Krauss, Y. Chen, and P. Chawla, “Nonverbal Behavior and Nonverbal Communication: What do Conversational Hand Gestures Tell Us?,” *Adv. Exp. Soc. Psychol.*, vol. 28, no. C, pp. 389–450, 1996.

[5] P. Bernardis and M. Gentilucci, “Speech and gesture share the same communication system,” *Neuropsychologia*, vol. 44, no. 2, pp. 178–190, 2006.

[6] B. M. Ciuffani, “Non-verbal Communication and Leadership The impact of hand gestures used by leaders on follower job satisfaction,” 2017.

[7] P. K. Pisharady, P. Vadakkepat, and A. P. Loh, “Attention Based Detection and Recognition of Hand Postures Against Complex Backgrounds,” *Int. J. Comput. Vis.*, vol. 101, no. 3, pp. 403–419, 2013.

[8] V. Inmoonnoy and M. Ketcham, “The message notification for patients care system using hand gestures recognition,” no. 2, 2017.

[9] C. L. Hwang and H. W. Lee, “The command control by hand gesture with Hu and contour sequence moments and probability neural network,” *Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern.*, pp. 2056–2061, 2011.

[10] H. A. Jalab and H. K. Omer, “Human computer interface using hand gesture recognition based on neural network,” *Inf. Technol. Towar. New Smart World (NSITNSW), 2015 5th Natl. Symp.*, pp. 1–6, 2015.

[11] O. Koller, H. Ney, and R. Bowden, “Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Data is Continuous and Weakly Labelled,” *2016 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 3793–3802, 2016.

[12] A. K. Lekova and M. I. Dimitrova, “Hand gestures recognition based on lightweight evolving fuzzy clustering method,” in *Image Information Processing (ICIIP), 2013 IEEE Second International Conference on*, 2013, pp. 505–510.

[13] M. Panwar, “Hand gesture recognition based on shape parameters,” in *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, 2012, pp. 1–6.

[14] V. T. Nguyen, T. L. Le, T. H. Tran, R. Mullot, and V. Courboulay, “Hand posture

recognition using Kernel Descriptor," *Procedia Comput. Sci.*, vol. 39, no. C, pp. 154–157, 2014.

[15] H. Li, L. Yang, X. Wu, S. Xu, and Y. Wang, "Static hand gesture recognition based on HOG with kinect," *Proc. 2012 4th Int. Conf. Intell. Human-Machine Syst. Cybern. IHMSC 2012*, vol. 1, pp. 271–273, 2012.

[16] Q. Munib, M. Habeeb, B. Takruri, and H. A. Al-Malik, "American sign language (ASL) recognition based on Hough transform and neural networks," *Expert Syst. Appl.*, vol. 32, no. 1, pp. 24–37, 2007.

[17] X. Meng, J. Lin, and Y. Ding, "An extended HOG model: SCHOG for human hand detection," *2012 Int. Conf. Syst. Informatics, ICSAI 2012*, no. Icsai, pp. 2593–2596, 2012.

[18] M. S. Sefat and M. Shahjahan, "A hand gesture recognition technique from real time video," *2nd Int. Conf. Electr. Eng. Inf. Commun. Technol. iCEEiCT 2015*, no. May, pp. 21–23, 2015.

[19] F. Tian, Q. C. Hu, and T. N. Zhang, "A hand gesture detection for multi class cascade classifier based on gradient," *Proc. - 5th Int. Conf. Instrum. Meas. Comput. Commun. Control. IMCCC 2015*, no. 1, pp. 1364–1368, 2016.

[20] W. Wu, C. Li, Z. Cheng, X. Zhang, and L. Jin, "YOLSE : Egocentric Fingertip Detection from Single RGB Images," pp. 623–630.

[21] F. Wang, L. Zhou, Z. Cui, H. Li, and M. Li, "Gesture recognition based on BoF and its application in human-machine interaction of service robot," *6th Annu. IEEE Int. Conf. Cyber Technol. Autom. Control Intell. Syst. IEEE-CYBER 2016*, pp. 115–120, 2016.

[22] Y. Zhao, Z. Song, and X. Wu, "Hand detection using multi-resolution HOG features," *2012 IEEE Int. Conf. Robot. Biomimetics, ROBIO 2012 - Conf. Dig.*, pp. 1715–1720, 2012.

[23] Sheenu, G. Joshi, and R. Vig, "Histograms of orientation gradient investigation for static hand gestures," *Int. Conf. Comput. Commun. Autom. ICCCA 2015*, pp. 1100–1103, 2015.

[24] E. Stergiopoulou and N. Papamarkos, "Hand gesture recognition using a neural network shape fitting technique," *Eng. Appl. Artif. Intell.*, vol. 22, no. 8, pp. 1141–1158, 2009.

[25] N. H. Dardas and N. D. Georganas, "Real-Time Hand Gesture Detection and Recognition Using Bag-of-Features and Support Vector Machine Techniques," *Instrum. Meas. IEEE Trans.*, vol. 60, no. 11, pp. 3592–3607, 2011.

[26] Y. Liu, Z. Gan, and Y. Sun, "Static hand gesture recognition and its application based on Support Vector Machines," pp. 517–521, 2008.

[27] D. K. Ghosh and S. Ari, "Static Hand Gesture Recognition Using Mixture of

Features and SVM Classifier," *2015 Fifth Int. Conf. Commun. Syst. Netw. Technol.*, pp. 1094–1099, 2015.

[28] D. K. Ghosh and S. Ari, "A static hand gesture recognition algorithm using k-mean based radial basis function neural network," *2011 8th Int. Conf. Information, Commun. Signal Process.*, no. i, pp. 1–5, 2011.

[29] J. Guo, "REAL-TIME HAND DETECTION BASED ON MULTI-STAGE HOG-SVM CLASSIFIER Guangdong Provincial Key Laboratory of Robotics and Intelligent System , Shenzhen Institutes of Advanced Technology , Chinese Academy of Sciences The Chinese University of Hong Kong," pp. 4108–4111, 2013.

[30] M. Han, J. Chen, L. Li, and Y. Chang, "Visual hand gesture recognition with convolution neural network," *2016 IEEE/ACIS 17th Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2016*, pp. 287–291, 2016.

[31] D. A. Huang, M. Ma, W. C. Ma, and K. M. Kitani, "How do we use our hands? Discovering a diverse set of common grasps," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 666–675, 2015.

[32] O. Rumyantsev, M. Merati, and V. Ramachandran, "Hand Sign recognition through palm gesture and movement," *Image Process.*, 2012.

[33] H.-S. Yeo, B.-G. Lee, and H. Lim, "Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware," *Multimed. Tools Appl.*, vol. 74, no. 8, pp. 2687–2715, 2015.

[34] S. S. Rautaray and A. Agrawal, "Design of gesture recognition system for dynamic user interface," in *Technology Enhanced Education (ICTEE), 2012 IEEE International Conference on*, 2012, pp. 1–6.

[35] J. Alon, V. Athitsos, S. Q. Yuan, and S. Sclaroff, "A Unified Framework for Gesture Recognition and Spatiotemporal Gesture Segmentation," *Pattern Anal. Mach. Intell. IEEE Trans.*, vol. 31, no. 9, pp. 1685–1699, 2009.

[36] M. Oberweger, P. Wohlhart, and V. Lepetit, "Hands Deep in Deep Learning for Hand Pose Estimation," 2015.

[37] A. M. J. L. Raheja and A. Chaudhary, J. L. Raheja, A. Mishra, and A. Chaudhary, "Indian sign language recognition using SVM," *Pattern Recognit. Image Anal.*, vol. 26, no. 2, pp. 434–441, 2016.

[38] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1106–1113, 2014.

[39] S. S. Rautaray and A. Agrawal, "Interaction with virtual game through hand gesture recognition," *Multimedia, Signal Process. Commun. Technol. (IMPACT), 2011 Int. Conf.*, pp. 244–247, 2011.

[40] S. S. Rautaray and A. Agrawal, "Real Time Multiple Hand Gesture Recognition

System for Human Computer Interaction," *Int. J. Intell. Syst. Appl.*, vol. 4, no. 5, pp. 56–64, 2012.

[41] J. Lee, Y. Lee, E. Lee, and S. Hong, "Hand region extraction and Gesture recognition from video stream with Complex Background Through Entropy Analysis," *Hand, The*, pp. 1513–1516, 2004.

[42] Z. Liu, F. Hu, D. Luo, I. Member, X. Wu, and I. Senior, "Visual Gesture Recognition for Human Robot Interaction Using Dynamic Movement Primitives," 2014.

[43] C. Li, X. Zhang, and L. Jin, "LPSNet: A Novel Log Path Signature Feature based Hand Gesture Recognition Framework," *Openaccess.Thecvf.Com*, pp. 631–639.

[44] Q. Yuan, S. Sclaroff, and V. Athitsos, "Automatic 2D hand tracking in video sequences," *Proc. 7th IEEE Work. Appl. Comupter Vis.*, pp. 250–256, 2005.

[45] V. Spruyt, A. Ledda, and W. Philips, "Real-time hand tracking by invariant hough forest detection," *Proc. - Int. Conf. Image Process. ICIP*, pp. 149–152, 2012.

[46] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 824–832, 2015.

[47] V. Spruyt, A. Ledda, W. Philips, and G. University-telin-ipi-iminds, "Real-time, long-term hand tracking with unsupervised initialization," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*, 2013, pp. 3730–3734.

[48] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt, "Fast and robust hand tracking using detection-guided optimization," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 3213–3221, 2015.

[49] S. Bhowmick, S. Kumar, and A. Kumar, "Hand gesture recognition of English alphabets using artificial neural network," *2015 IEEE 2nd Int. Conf. Recent Trends Inf. Syst.*, pp. 405–410, 2015.

[50] P. Chochai, T. Mekrungroj, and T. Matsumaru, "Real-time gesture recognition with finger naming by RGB camera and IR depth sensor," *2014 IEEE Int. Conf. Robot. Biomimetics, IEEE ROBIO 2014*, pp. 931–936, 2014.

[51] L.-M. Chen, C. T. Hsieh, K.-M. Hung, C.-H. Yeh, and C.-Y. Ke, "A Real Time Hand Gesture Recognition System Based on DFT and SVM," *Inf. Sci. Digit. Content Technol. (ICIDT), 2012 8th Int. Conf.*, vol. 284–287, pp. 490–494, 2013.

[52] T. Grzejszczak, M. Kawulok, and A. Galuszka, "Hand Landmarks Detection and Localization in Color Images," *Multimed. Tools Appl.*, vol. 75, no. 23, pp. 16363–16387, Dec. 2016.

[53] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.

[54] P. A. Popov, "Long Hands Gesture Recognition: Gesture Recognition Dataset," 2020. [Online]. Available:

http://www.site.uottawa.ca/research/viva/projects/handgesturerecognition/index.html#dataset_gesture.

[55] P. A. Popov, “Long Hands Gesture Recognition and Hand Tracking: Hand Tracking Dataset,” 2020. [Online]. Available: http://www.site.uottawa.ca/research/viva/projects/handgesturerecognition/index.html#dataset_handtrack.

[56] C.-C. Hsieh and D.-H. Liou, “Novel Haar features for real- time hand gesture recognition using SVM,” *J. Real-Time Image Process.*, vol. 10, no. 2, pp. 357–370, 2015.

[57] S. K. Yewale and P. K. Bharne, “Hand gesture recognition using different algorithms based on artificial neural network,” *2011 Int. Conf. Emerg. Trends Networks Comput. Commun.*, no. 1998, pp. 287–292, 2011.

[58] K. Roy, A. Mohanty, and R. R. Sahay, “Deep Learning Based Hand Detection in Cluttered Environment Using Skin Segmentation,” pp. 640–649, 2017.

[59] C. Costanzo, G. Iannizzotto, and F. La Rosa, “VirtualBoard: real-time visual gesture recognition for natural human-computer interaction,” *Proc. Int. Parallel Distrib. Process. Symp.*, vol. 00, no. C, p. 8, 2003.

[60] Z. Ren, J. Yuan, J. Meng, and Z. Zhang, “Robust part-based hand gesture recognition using kinect sensor,” *IEEE Trans. Multimed.*, vol. 15, no. 5, pp. 1110–1120, 2013.

[61] Z. Ren, J. Yuan, and Z. Zhang, “Robust hand gesture recognition based on finger-earth mover’s distance with a commodity depth camera,” *Proc. 19th ACM Int. Conf. Multimed. - MM ’11*, p. 1093, 2011.

[62] Y. Kong and Y. Fu, “Bilinear Heterogeneous Information Machine for RGB-D Action Recognition,” *Cvpr*, 2015.

[63] J. Liu, K. Furusawa, T. Tateyama, Y. Iwamoto, and Y. W. Chen, “An Improved Hand Gesture Recognition with Two-Stage Convolution Neural Networks Using a Hand Color Image and its Pseudo-Depth Image,” *Proc. - Int. Conf. Image Process. ICIP*, vol. 2019-Septe, pp. 375–379, 2019.

[64] C. Kerdvibulvech, “Hand tracking by extending distance transform and hand model in real-time,” *Pattern Recognit. Image Anal.*, vol. 25, no. 3, pp. 437–441, 2015.

[65] T. Lu, L. Peng, and Y. Zhang, “Edge feature based approach for object recognition,” *Pattern Recognit. Image Anal.*, vol. 26, no. 2, pp. 350–353, 2016.

[66] D. K. Ghosh and S. Ari, “On an algorithm for Vision-based hand gesture recognition,” *Signal, Image Video Process.*, vol. 10, no. 4, pp. 655–662, 2016.

[67] X. Fu, T. Zhang, C. Bonair, M. L. Coats, and J. Lu, “Wavelet enhanced image preprocessing and neural networks for hand gesture recognition,” *Proc. - 2015*

IEEE Int. Conf. Smart City, SmartCity 2015, Held Jointly with 8th IEEE Int. Conf. Soc. Comput. Networking, Soc. 2015, 5th IEEE Int. Conf. Sustain. Comput. Communic, pp. 838–843, 2015.

[68] V. Chernyshov and L. Mestetskiy, "Real-time hand detection using continuous skeletons," *Pattern Recognit. Image Anal.*, vol. 26, no. 2, pp. 368–373, 2016.

[69] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proc. - 2005 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition, CVPR 2005*, vol. I, pp. 886–893, 2005.

[70] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, vol. 1, pp. I-511-I–518 vol.1.

[71] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3951 LNCS, pp. 404–417, 2006.

[72] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast Retina Keypoint," *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. pp. 510–517, 2012.

[73] D. G. Lowe, "Object recognition from local scale-invariant features," *Proc. Seventh IEEE Int. Conf. Comput. Vis.*, pp. 1150–1157 vol.2, 1999.

[74] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," *Proceedings. Int. Conf. Image Process.*, vol. 1, pp. I-900-I–903, 2002.

[75] C. H. Messom and A. L. Barczak, "Stream processing for fast and efficient rotated Haar-like features using rotated integral images," *Int. J. Intell. Syst. Technol. Appl.*, vol. 7, no. 1, p. 40, 2009.

[76] P. Dollár, P. Welinder, and P. Perona, "Cascaded pose regression," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1078–1085, 2010.

[77] J. Uwineza, H. Ma, B. Li, and Y. Jin, *Static Hand Gesture Recognition for Human Robot Interaction*, vol. 11741 LNAI. Springer International Publishing, 2019.

[78] T. Kadir and M. Brady, "Saliency, Scale and Image Description," *Int. J. Comput. Vis.*, vol. 45, no. 2, pp. 83–105, 2001.

[79] M. H. Alamatsaz, "On discrete a-unimodal distributions," *Stat. Neerl.*, vol. 47, no. 4, pp. 245–252, 1993.

[80] R. P. Larkin, "An algorithm for assessing bimodality vs. unimodality in a univariate distribution," *Behav. Res. Methods Instrum.*, vol. 11, no. 4, pp. 467–468, 1979.

[81] P. Medgyessy, "On the Unimodality of Discrete Distributions," vol. 2, pp. 245–257, 1972.

[82] P. L. Rosin, “Unimodal thresholding,” *Pattern Recognit.*, vol. 34, no. 11, pp. 2083–2096, 2001.

[83] J. D. Elon, “A non parametric theory for histogram segmentation,” pp. 1–8.

[84] S. Basu, “Bayesian Tests for Unimodality.” .

[85] N. Coudray *et al.*, “A robust thresholding algorithm for unimodal image histograms To cite this version : A robust thresholding algorithm for unimodal image histograms,” vol. 31, pp. 1010–1019, 2013.

[86] C. Nölker and H. Ritter, “Visual recognition of continuous hand postures,” *IEEE Trans. Neural Networks*, vol. 13, no. 4, pp. 983–994, 2002.

[87] K. Hu, L. Yin, and T. Wang, “Temporal Interframe Pattern Analysis for Static and Dynamic Hand Gesture Recognition,” *Proc. - Int. Conf. Image Process. ICIP*, vol. 2019-Septe, pp. 3422–3426, 2019.

[88] M. Z. Islam, M. S. Hossain, R. Ul Islam, and K. Andersson, “Static hand gesture recognition using convolutional neural network with data augmentation,” *2019 Jt. 8th Int. Conf. Informatics, Electron. Vision, ICIEV 2019 3rd Int. Conf. Imaging, Vis. Pattern Recognition, icIVPR 2019 with Int. Conf. Act. Behav. Comput. ABC 2019*, pp. 324–329, 2019.

[89] Y. Pang, S. Member, K. Zhang, Y. Yuan, S. Member, and K. Wang, “Distributed Object Detection With Linear SVMs,” vol. 44, no. 11, pp. 2122–2133, 2014.

[90] D. Wu *et al.*, “Deep Dynamic Neural Networks for Multimodal Gesture Segmentation and Recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1583–1597, 2016.

[91] Y. Chen and W. Zhang, “Research and Implementation of Sign Language Recognition Method Based on Kinect,” pp. 1947–1951, 2016.

[92] V. Adithya and R. Rajesh, “A Deep Convolutional Neural Network Approach for Static Hand Gesture Recognition,” *Procedia Comput. Sci.*, vol. 171, no. 2019, pp. 2353–2361, 2020.

[93] A. P. Nemirko, “Transformation of feature space based on Fisher’s linear discriminant,” *Pattern Recognit. Image Anal.*, vol. 26, no. 2, pp. 257–261, 2016.

[94] H. Wang, A. Kl, C. Schmid, and C. Liu, “Action Recognition by Dense Trajectories,” *Cvpr*, pp. 3169–3176, 2011.

[95] K. E. A. Van De Sande, C. G. M. Snoek, and A. W. M. Smeulders, “Fisher and VLAD with FLAIR,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2377–2384, 2014.

[96] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “Exploiting the circulant structure of tracking-by-detection with kernels,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7575 LNCS, no. PART 4, pp. 702–715, 2012.

[97] Y. Zhou, K. Liu, R. E. Carrillo, K. E. Barner, and F. Kiamilev, *Kernel-based sparse representation for gesture recognition*, vol. 46, no. 12. Elsevier, 2013.

[98] L. Bo, X. Ren, and D. Fox, "Kernel descriptors for visual recognition," *Nips*, pp. 1–9, 2010.

[99] S. Jagannathan *et al.*, "Efficient Object Detection and Classification on Low Power Embedded Systems," pp. 7–8, 2017.

[100] C. Cortes and V. N. Vapnik, "Support- vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[101] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classiiers," *Proc. fifth Annu. Work. Comput. Learn. theory*, pp. 144–152, 1992.

[102] and C.-J. L. Chih-Wei Hsu, Chih-Chung Chang, "A Practical Guide to Support Vector Classification," *BJU Int.*, vol. 101, no. 1, pp. 1396–400, 2008.

[103] J. Oh, T. Kim, and H. Hong, "Using binary decision tree and multiclass SVM for human gesture recognition," *2013 Int. Conf. Inf. Sci. Appl.*, pp. 1–4, 2013.

[104] D. D. Nguyen and H. S. Le, "Kinect Gesture Recognition: SVM vs. RVM," *Proc. - 2015 IEEE Int. Conf. Knowl. Syst. Eng. KSE 2015*, pp. 395–400, 2016.

[105] J. P. Sahoo, S. Ari, and S. K. Patra, "Hand gesture recognition using PCA based deep CNN reduced features and SVM classifier," *Proc. - 2019 IEEE Int. Symp. Smart Electron. Syst. iSES 2019*, pp. 221–224, 2019.

[106] Y. Zhiqi, "Gesture learning and recognition based on the Chebyshev polynomial neural network," *Proc. 2016 IEEE Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2016*, pp. 931–934, 2016.

[107] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand gesture recognition with 3D convolutional neural networks," *2015 IEEE Conf. Comput. Vis. Pattern Recognit. Work.*, pp. 1–7, 2015.

[108] P. Mekala, J. Fan, W.-C. Lai, and C.-W. Hsue, "Gesture recognition using neural networks based on HW/SW cosimulation platform," *Adv. Softw. Eng.*, vol. 2013, p. 2, 2013.

[109] S. Ji, M. Yang, and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition.," *Pami*, vol. 35, no. 1, pp. 221–31, 2013.

[110] H. Hasan and S. Abdul-Kareem, "Static hand gesture recognition using neural networks," *Artif. Intell. Rev.*, vol. 41, no. 2, pp. 147–181, 2014.

[111] M. Baccouche, F. Mamalet, and C. Wolf, "Sequential deep learning for human action recognition," *Proc. Int. Conf. Hum. Behav. Underst.*, pp. 29–39, 2011.

[112] M. Abadi *et al.*, "TensorFlow: A System for Large-Scale Machine Learning TensorFlow: A system for large-scale machine learning," *12th USENIX Symp. Oper.*

Syst. Des. Implement. (OSDI '16), pp. 265–284, 2016.

[113] T. Sheng, C. Feng, S. Zhuo, X. Zhang, L. Shen, and M. Aleksic, "A Quantization-Friendly Separable Convolution for MobileNets," 2018.

[114] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," pp. 7310–7319, 2016.

[115] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," 2017.

[116] Y. Rubner, C. Tomasi, and L. J. Guibas, "The Earth Mover ' s Distance as a Metric for Image Retrieval," *Work*, vol. 40, no. 2, pp. 1–20, 2000.

[117] Z. Ren, J. Yuan, C. Li, and W. Liu, "Minimum near-convex decomposition for robust shape representation," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 303–310, 2011.

[118] M. J. Jeon, S. E. Yang, and Z. Bien, "User adaptive hand gesture recognition using multivariate fuzzy decision tree and fuzzy garbage model," *IEEE Int. Conf. Fuzzy Syst.*, pp. 474–479, 2009.

[119] R. Xu, S. Zhou, and W. J. Li, "MEMS accelerometer based nonspecific-user hand gesture recognition," *IEEE Sens. J.*, vol. 12, no. 5, pp. 1166–1173, 2012.

[120] T. Cootes, "Active Appearance Models," *Pattern Anal. …*, vol. 23, no. 6, pp. 681–685, 2001.

[121] I. Matthews and S. Baker, "Active Appearance Models Revisited," *Ijcv*, vol. 60, no. 2, pp. 135–164, 2004.

[122] I. Matthews, T. Ishikawa, and S. Baker, "The template update problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 810–815, 2004.

[123] S. Khamis, J. Taylor, J. Shotton, C. Keskin, S. Izadi, and A. Fitzgibbon, "Learning an efficient model of hand shape variation from depth images," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 2540–2548, 2015.

[124] F. A. Kondori, S. Yousefi, J.-P. Kouma, L. Liu, and H. Li, "Direct Hand Pose Estimation for Immersive Gestural Interaction," *Pattern Recogn. Lett.*, vol. 66, pp. 91–99, Nov. 2015.

[125] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand Keypoint Detection in Single Images Using Multiview Bootstrapping," *2017 IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017.

[126] D. Tang, H. J. Chang, A. Tejani, and T.-K. Kim, "Latent Regression Forest: Structured Estimation of 3D Hand Poses," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 7, p. 1, 2017.

[127] H. Liang, J. Yuan, and D. Thalmann, "Parsing the Hand in Depth Images," *IEEE Trans. Multimed.*, vol. 16, pp. 1241–1253, 2014.

[128] S. Bambach, S. Lee, D. Crandall, and C. Yu, “Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions,” in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[129] J. Taylor *et al.*, “Efficient and Precise Interactive Hand Tracking Through Joint, Continuous Optimization of Pose and Correspondences,” *ACM Trans. Graph.*, vol. 35, no. 4, pp. 143:1--143:12, Jul. 2016.

[130] D. Tzionas, A. Srikantha, P. Aponte, and J. Gall, “Capturing Hand Motion with an RGB-D Sensor, Fusing a Generative Model with Salient Points,” *Pattern Recognit.*, pp. 277–289, 2014.

[131] D. Tang *et al.*, “Opening the Black Box: Hierarchical Sampling Optimization for Estimating Human Hand Pose,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.

[132] J. Tompson, M. Stein, Y. LeCun, and K. Perlin, “Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks,” *ACM Trans. Graph.*, vol. 33, pp. 169:1-169:10, 2014.

[133] S. Sridhar, H. Rhodin, H.-P. Seidel, A. Oulasvirta, and C. Theobalt, “Real-time hand tracking using a sum of anisotropic gaussians model,” in *2014 2nd International Conference on 3D Vision*, 2014, vol. 1, pp. 319–326.

[134] M. Oberweger, P. Wohlhart, and V. Lepetit, “Training a Feedback Loop for Hand Pose Estimation,” in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 3316–3324.

[135] J. Charles, T. Pfister, M. Everingham, and A. Zisserman, “Automatic and Efficient Human Pose Estimation for Sign Language Videos,” *Int. J. Comput. Vis.*, pp. 70–90, 2014.

[136] M. Elmezain, A. Al-hamadi, R. Niese, and B. Michaelis, “A robust method for hand tracking using mean-shift algorithm and Kalman filter in stereo color image sequences,” in *Proceedings of World Academy of Science, Engineering and Technology*, 2009, pp. 283–287.

[137] C. Wan, T. Probst, L. Van Gool, and A. Yao, “Crossing Nets: Combining GANs and VAEs with a Shared Latent Space for Hand Pose Estimation,” *2017 IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017.

[138] S. Sridhar, F. Mueller, M. Zollhoefer, D. Casas, A. Oulasvirta, and C. Theobalt, “Real-time Joint Tracking of a Hand Manipulating an Object from RGB-D Input,” in *Proceedings of European Conference on Computer Vision ({ECCV})*, 2016, pp. 1–17.

[139] E. Remelli, A. Tkach, A. Tagliasacchi, and M. Pauly, “Low-Dimensionality Calibration through Local Anisotropic Scaling for Robust Hand Model Personalization,” 2017, pp. 2554–2562.

[140] A. Tkach, A. Tagliasacchi, E. Remelli, M. Pauly, and A. Fitzgibbon, “Online Generative Model Personalization for Hand Tracking,” *ACM Trans. Graph.*, vol. 36,

no. 6, pp. 243:1--243:11, Nov. 2017.

[141] L. Ge, H. Liang, J. Yuan, and D. Thalmann, “Robust 3D Hand Pose Estimation in Single Depth Images: From Single-View CNN to Multi-View CNNs,” *2016 IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016.

[142] Y. Huang, X. Liu, X. Zhang, and L. Jin, “A Pointing Gesture Based Egocentric Interaction System: Dataset, Approach and Application,” 2016, pp. 370–377.

[143] A. Tagliasacchi, M. Schroeder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly, “Robust Articulated-ICP for Real-Time Hand Tracking,” *Comput. Graph. Forum (Symposium Geom. Process.*, vol. 34, no. 5, 2015.

[144] X. Suau, M. Alcoverro, A. López-méndez, J. Ruiz-hidalgo, and J. R. Casas, “Real-time fingertip localization conditioned on hand gesture classification,” *Image Vis. Comput.*, vol. 32, no. 8, pp. 522–532, 2014.

[145] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt, “Real-time hand tracking under occlusion from an egocentric rgb-d sensor,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1284–1293.

[146] A. Bulat and G. Tzimiropoulos, “Human Pose Estimation via Convolutional Part Heatmap Regression,” *Lect. Notes Comput. Sci.*, p. 717–732, 2016.

[147] I. Oikonomidis, M. I. A. Lourakis, and A. A. Argyros, “Evolutionary Quasi-random Search for Hand Articulations Tracking,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[148] M. Oberweger, G. Riegler, P. Wohlhart, and V. Lepetit, “Efficiently Creating 3D Training Data for Fine Hand Pose Estimation,” 2016.

[149] F. Mueller *et al.*, “GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 49–59, 2018.

[150] C. Liu, Z. Li, C. Zhang, Y. Yan, and R. Zhang, “An Improved hand tracking algorithm for Chinese sign language recognition,” *Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2019*, no. Itnec, pp. 229–232, 2019.

[151] P. N. Crisnapati, M. Setiawan, I. G. N. Wikranta Arsa, P. Devi Novayanti, M. S. Wibawa, and K. G. Oka Ciptahadi, “Real-Time Hand Palm Detection and Tracking Augmented Reality Game Using Lucas Kanade Optical Flow Combined with Color Blob Detection,” *2019 1st Int. Conf. Cybern. Intell. Syst. ICORIS 2019*, no. August, pp. 263–268, 2019.

[152] Y. Sun, X. Liang, H. Fan, M. Imran, and H. Heidari, “Visual Hand Tracking on Depth Image using 2-D Matched Filter,” *2019 UK/China Emerg. Technol. UCET 2019*, pp. 17–20, 2019.

[153] G. Park, A. Argyros, J. Lee, and W. Woo, “3D Hand Tracking in the Presence of Excessive Motion Blur,” *IEEE Trans. Vis. Comput. Graph.*, vol. 26, no. 5, pp. 1891–1901, 2020.

[154] Y. Zhang, C. Wang, J. Zhao, L. Zhang, and S.-C. Chan, “Template selection based superpixel earth mover’s distance algorithm for hand gesture recognition,” in *2016 IEEE 13th International Conference on Signal Processing (ICSP)*, 2016, pp. 1002–1005.

[155] A. I. Maqueda, C. R. Del-Blanco, F. Jaureguizar, and N. Garc\’\ia, “Human-computer Interaction Based on Visual Hand-gesture Recognition Using Volumetric Spatiograms of Local Binary Patterns,” *Comput. Vis. Image Underst.*, vol. 141, no. C, pp. 126–137, Dec. 2015.

[156] C.-H. Wu, W.-L. Chen, and C. H. Lin, “Depth-based Hand Gesture Recognition,” *Multimed. Tools Appl.*, vol. 75, no. 12, pp. 7065–7086, Jun. 2016.

[157] E. Ohn-Bar and M. M. Trivedi, “Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2368–2377, 2014.

[158] G. Plouffe and A. Cretu, “Static and Dynamic Hand Gesture Recognition in Depth Data Using Dynamic Time Warping,” *IEEE Trans. Instrum. Meas.*, vol. 65, no. 2, pp. 305–316, Feb. 2016.

[159] H. Cheng, Z. Dai, Z. Liu, and Y. Zhao, “An image-to-class dynamic time warping approach for both 3D static and trajectory hand gesture recognition,” *Pattern Recognit.*, vol. 55, pp. 137–147, 2016.

[160] M. K. Bhuyan, D. A. Kumar, K. F. MacDorman, and Y. Iwahori, “A novel set of features for continuous hand gesture recognition,” *J. Multimodal User Interfaces*, vol. 8, pp. 333–343, 2014.

[161] C. Linqin, C. Shuangjie, X. Min, Y. Jimin, and Z. Jianrong, “Dynamic hand gesture recognition using RGB-D data for natural human-computer interaction,” *J. Intell. Fuzzy Syst.*, vol. 32, pp. 3495–3507, 2017.

[162] D. Xu, X. Wu, Y.-L. Chen, and Y. Xu, “Online Dynamic Gesture Recognition for Human Robot Interaction,” *J. Intell. Robot. Syst.*, vol. 77, no. 3–4, pp. 583–596, Mar. 2015.

[163] W. Yang, J. Tao, and Z. Ye, “Continuous sign language recognition using level building based on fast hidden Markov model,” *Pattern Recognit. Lett.*, vol. 78, pp. 28–35, 2016.

[164] N. Neverova, C. Wolf, G. Taylor, and F. Nebout, “ModDrop: Adaptive Multi-Modal Gesture Recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1692–1706, Aug. 2016.

[165] O. K. Oyedotun and A. Khashman, “Deep Learning in Vision-based Static Hand Gesture Recognition,” *Neural Comput. Appl.*, vol. 28, no. 12, pp. 3941–3951, Dec.

2017.

[166] D.-L. Dinh, S. Lee, and T.-S. Kim, “Hand number gesture recognition using recognized hand parts in depth images,” *Multimed. Tools Appl.*, vol. 75, no. 2, pp. 1333–1348, 2016.

[167] C. Wang, Z. Liu, and S.-C. Chan, “Superpixel-based hand gesture recognition with kinect depth camera,” *IEEE Trans. Multimed.*, vol. 17, no. 1, pp. 29–39, 2015.

[168] D. Tang and Y. J. Zhang, “Combining mean-shift and particle filter for object tracking,” *Proc. - 6th Int. Conf. Image Graph. ICIG 2011*, pp. 771–776, 2011.

[169] S. Wang, H. Ji, and I. Engineering, “a New Appearance Model Based on Object Sub-Region for TRACKING,” *Proc. 2007 Int. Conf. Wavelet Anal. Pattern Recognit.*, pp. 2–4, 2007.

[170] B. Deutsch, C. Grassl, F. Bajramovic, and J. Denzler, “A comparative evaluation of template and histogram based 2D tracking algorithms,” *Pattern Recognition, Proc.*, vol. 3663, pp. 269–276, 2005.

[171] E. Emami and M. Fathy, “Object Tracking Using Improved CAMShift Algorithm Combined with Motion Segmentation,” *2011 7th Iran. Conf. Mach. Vis. Image Process.*, pp. 1–4, 2011.

[172] W. C. Hu, “Adaptive template block-based block matching for object tracking,” *Proc. - 8th Int. Conf. Intell. Syst. Des. Appl. ISDA 2008*, vol. 1, pp. 61–64, 2008.

[173] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-Learning-Detection,” *Pattern Anal. Mach. Intell. IEEE Trans.*, vol. 34, no. 7, pp. 1409–1422, 2012.

[174] G. R. Bradski, “Computer Vision Face Tracking For Use in a Perceptual User Interface (cité 1923 fois),” *Intel Technol. J.*, vol. 2, no. 2, pp. 12–21, 1998.

[175] Z. Kalal, K. Mikolajczyk, and J. Matas, “Forward-Backward Error : Automatic Detection of Tracking Failures,” *2010 20th Int. Conf. Pattern Recognit.*, pp. 2756–2759, 2010.

[176] C. R. Del-Blanco, F. Jaureguizar, and N. García, “An Efficient Multiple Object Detection and Tracking Framework for Automatic Counting and Video Surveillance Applications,” pp. 857–862, 2012.

[177] B. Lucas and T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision.” 1981.

[178] R. Y. Tsai, “A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses,” *IEEE J. Robot. Autom.*, vol. 3, no. 4, pp. 323–344, 1987.

[179] B. K. P. Horn, “Tsai’s Camera Calibration Method Revisited,” *Online: http://people.csail.mit.edu/bkph/articles/Tsai_Revisited.pdf*, vol. i, p. 123, 2000.

[180] Z. Zhang, “A Flexible New Technique for Camera Calibration (Technical Report),”

IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 11, pp. 1330–1334, 2002.

[181] I. I. Workshop, M. Learning, and S. P. September, "A PRACTICAL APPROACH FOR DEPTH ESTIMATION AND IMAGE RESTORATION USING DEFOCUS CUE Keyur R . Ranipa M . V . Joshi," pp. 0–5, 2011.

[182] J. Lalonde, R. Laganiere, and L. Martel, "A single-view based obstacle detection for smart back-up camera systems," *IEEE Int. Conf. Comput. Vis. Pattern Recognit. Work.*, p. 2012, 2012.

[183] G. Surya and M. Subbarao, "Depth from defocus by changing camera aperture: a spatial domain approach," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR 1993)*, pp. 61–67, 1993.

[184] S. Ruffieux, D. Lalanne, E. Mugellini, and O. A. Khaled, "Gesture recognition corpora and tools: A scripted ground truthing method," *Comput. Vis. Image Underst.*, vol. 131, pp. 72–87, 2015.

[185] P. A. Popov, "Hand Gesture Recognition and Tracking," 2019. [Online]. Available: https://www.youtube.com/watch?v=6ICK1ELZmfE.

[186] P. A. Popov, "Long Hands Gesture Recognition System," 2015. [Online]. Available: https://www.youtube.com/watch?v=OcgXYQFNkIU.

[187] P. A. Popov, "Long Hands Gesture Recognition Controller for Halo," 2015. [Online]. Available: https://www.youtube.com/watch?v=NuVAW6wihZ8.

[188] P. A. Popov, "GR Paint App Presentation HD," 2015. [Online]. Available: https://www.youtube.com/watch?v=8e2xZpFhcIY.

[189] P. A. Popov, *GestureVue Finger Mouse Improved Version 2*. 2013.

[190] G. Kapidis, R. Poppe, E. Van Dam, L. Noldus, and R. Veltkamp, "Egocentric hand track and object-based human action recognition," *Proc. - 2019 IEEE SmartWorld, Ubiquitous Intell. Comput. Adv. Trust. Comput. Scalable Comput. Commun. Internet People Smart City Innov. SmartWorld/UIC/ATC/SCALCOM/IOP/SCI 2019*, pp. 922–929, 2019.

[191] K. M. Jonathan Huang, Vivek Rathod, Derek Chow, Chen Sun, Menglong Zhu, Matthew Tang, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, Jasper Uijlings, Viacheslav Kovalevskyi, "Tensor Flow with Object Detection." 2017.

[192] A. Y. Khinchin, "On unimodal distributions," *Trams. Res. Inst. Math. Mech. (in Russ.*, vol. 2, no. 2, pp. 1–7, 1938.

[193] P. A. Popov, *Testing Multiscale Gesture Recognition with HOG Cascades Result Video*. 2017.

[194] P. A. Popov, *Demo of HOG Classifier Pose Recognition d1 dataset*. 2016.