Efficient Feature Extraction for Shape Analysis, Object Detection and Tracking

by

Andrés Solís Montero

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the Doctorate in Philosophy degree in Computer Science

uOttawa

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

Abstract

During the course of this thesis, two scenarios are considered. In the first one, we contribute to feature extraction algorithms. In the second one, we use features to improve object detection solutions and localization. The two scenarios give rise to into four thesis sub-goals. First, we present a new shape skeleton pruning algorithm based on contour approximation and the integer medial axis. The algorithm effectively removes unwanted branches, conserves the connectivity of the skeleton and respects the topological properties of the shape. The algorithm is robust to significant boundary noise and to rigid shape transformations. It is fast and easy to implement. While shape-based solutions via boundary and skeleton analysis are viable solutions to object detection, keypoint features are important for textured object detection. Therefore, we present a keypoint feature-based planar object detection framework for vision-based localization. We demonstrate that our framework is robust against illumination changes, perspective distortion, motion blur, and occlusions. We increase robustness of the localization scheme in cluttered environments and decrease false detection of targets. We present an off-line target evaluation strategy and a scheme to improve pose. Third, we extend planar object detection to a real-time approach for 3D object detection using a mobile and uncalibrated camera. We develop our algorithm based on two novel naive Bayes classifiers for viewpoint and feature matching that improve performance and decrease memory usage. Our algorithm exploits the specific structure of various binary descriptors in order to boost feature matching by conserving descriptor properties. Our novel naive classifiers require a database with a small memory footprint because we only store efficiently encoded features. We improve the feature-indexing scheme to speed up the matching process creating a highly efficient database for objects. Finally, we present a model-free long-term tracking algorithm based on the Kernelized Correlation Filter. The proposed solution improves the correlation tracker based on precision, success, accuracy and robustness while increasing frame rates. We integrate adjustable Gaussian window and sparse features for robust scale estimation creating a better separation of the target and the background. Furthermore, we include fast descriptors and Fourier spectrum packed format to boost performance while decreasing the memory footprint. We compare our algorithm with state-of-the-art techniques to validate the results.

Acknowledgements

I would like to thank my supervisors Jochen Lang and Robert Laganière,
for their patient guidance, encouragement and advice provided throughout
my time as their student.

.

I would also like to thank my committee members, professor Eric Dubois,
professor Wail Gueaieb, professor Prosejit Bose and professor Faisal Qureshi
for serving as my committee members and
for your brilliant comments and suggestions, thanks to you.

.

A special thanks to Sandra Cocea, who supported me in writing, and
incented me to strive towards my goal.

.

This work was partially supported by
the Natural Sciences and Engineering Research Council (NSERC)
Pulse Microsystems Ltd, Cohort System Inc, YOUi Labs,
Habitat Seven and Thales Canada.

Dedication

A mis padres,
María Elena Montero Pérez y Francisco Luis Solís Corvo.
Para ustedes que han sido mi guía e inspiración en todo momento.
Ojalá hubiesen tenido la oportunidad de ver todo lo que
he alcanzado gracias a ustedes.
Los quiero y extraño cada día.

Table of contents

List of Figures

Glossary

Alien Long-term object tracking algorithm that uses scale-invariant feature transform(SIFT) in a combined matching-and-tracking framework [108]. 25

BRIEF (B)inary (R)obust (I)ndependent (E)lementary (F)eatures is a general-purpose binary feature point descriptor and feature extractor [23]. 21, 77, 83–85, 88

BRISK (B)inary (R)obust (I)nvariant (S)calable (K)eypoints is a general-purpose binary descriptor and keypoint extractor with low computational cost [72]. 21, 25, 77, 78, 83–85

CCS Intel's Complex Conjugate Symmetric packed format to represent Fourier Spectrums. 26, 99, 103, 107

CMT (C)onsensus-based (M)atching and (T)racking of Keypoints for Object Tracking is a keypoint-based (i.e., uses BRISK binary descriptors) method for long-term object tracking in a combined matching-and-tracking framework [95]. 23, 25

CSK Long-term object tracking algorithm that uses tracking-by-detection strategy. The initials stand for (C)irculant (S)tructure of Tracking-by-detection with (K)ernels [51]. 23, 25, 26

DCE Skeleton pruning method based on contour partitioning obtained by (D)iscrete (C)urve (E)volution. The algorithm creates polygonal approximation of the binary shapes to aid the pruning of digital images. 18

DFT (D)iscrete (F)ourier (T)ransform converts a finite list of equally spaced samples of a function into the list of coefficients of a finite combination of complex sinusoids, ordered by their frequencies, that has those same sample values. 92, 94, 98, 99

distance transform Also known as distance map or distance field, is a representation of a digital image. The map labels each pixel of the image with the distance to the nearest boundary pixel in a binary image. 16, 17

FAST (F)eatures from (A)ccelerated (S)egment (T)est is a corner detection method, which could be used to extract feature points and later used to track and map objects in computer vision. 21

feature transform Distance map for every single pixel of a binary shape with information about the closest point in the boundary and its distance to it. Feature transform differs from the distance transform in that not only the distance is computed, but also the closest boundary point. 16, 17

Ferns Ferns is a randomized based classifier to match feature keypoints. Each fern in the classifier is composed by a group of binary features. xii, 22, 23, 28, 57, 60, 64, 74, 77, 78, 83–87, 89, 90

FREAK (F)ast (Re)tin(a) (K)eypoint is a general-purpose binary descriptor inspired by the human visual system [146]. 21, 77, 78, 83, 85, 88

HIP Binary (H)istogrammed (I)ntesity (P)atches is a method for feature-based matching which offers fast runtime performance [141]. 21

HOG (H)istogram of (O)riented (G)radients is a feature descriptor used to detect objects in computer vision and image processing. The HOG descriptor technique counts occurrences of gradient orientation in localized portions of an image [29]. 21, 26, 28, 92, 103, 106

IDFT Inverse (D)iscrete (F)ourier (T)ransform. 98, 99

IMA (I)nteger (M)edial (A)xis is an algorithm that computes a discrete representation of the medial axis for digital images. 19

KCF (K)ernelized (C)orrelation (F)ilter is an extension of CSK framework by integrating multiple channel data using HOG features instead of raw pixel values [52]. 12, 23, 25, 26, 92–95, 98, 103, 106, 107

KLT (K)anade-(L)ucas-(T)omasi feature tracke is an approach to feature matching and optical flow estimation. 24, 25

long-term tracking Refers to object tracking algorithms that follows the location of an object in video sequences. The algorithm should track the object even if it

is occluded or if it exits and re-enters the video frame. The algorithm is usually a combination of multiple computer vision techniques for tracking, learning and detection. The term does not specify a time limit itself. It just refers to the ability of continue tracking the object after occlusion or when it leaves the image frame and returns. ii, 1, 5, 7, 15, 23, 24, 28, 103, 104, 108, 109

medial axis The medial axis of an object is the set of all points having more than one closest point on the object's boundary. Originally referred to as the topological skeleton, it was introduced by Blum as a tool for biological shape recognition. 2, 3, 7, 10, 12, 16–19, 27

model-free Refers to object tracking algorithms that do not requiere a prior knowledge of object to track. Model-free tracking consist in the online learning and adaptation of the representation of a given target. ii, 5, 6

NBCD (N)aive (B)ayes (C)lassified (D)escriptors is our classifier for binary keypoint matching. The classifier is a combination between a Naive Bayes model with binary descriptors. xii, 77, 84–86, 88, 90

NCC (N)ormalized (C)ross (C)orrelation is one of the methods used for template matching, a process used for finding instances of a pattern or object within an image. Cross correlation is a measure of similarity of two waveforms as a function of a time-lag applied to one of them. For image-processing applications in which brightness of the image and template can vary due to lighting and exposure conditions, the images are first normalized. 24

ORB (O)riented FAST and (R)otated (B)RIEF is a fast robust local feature detector and binary descriptor [124]. xii, 21, 77, 78, 83–86, 88

RANSAC (Ran)dom (Sa)mple (C)onsensus is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers. It is a non-deterministic algorithm that produces a reasonable result with a certain probability by separating inliers from outliers. 68, 69, 77, 80, 82

SFM (S)tructure (F)rom (M)otion is a range imaging technique; it refers to the process of estimating three-dimensional structures from two-dimensional image sequences which may be coupled with local motion signals. 20, 21

SIFT (S)cale-(I)nvariant (F)eature (T)ransform is an algorithm in computer vision to detect and describe local features in images [77]. 21, 25, 28, 57

skeleton In shape analysis, the skeleton (or topological skeleton) of a shape is a thin version of that shape, which is equidistant to its boundaries. The skeleton usually emphasizes the geometrical and topological properties of the shape such as its connectivity, topology, length, direction and width. Combined with the distance from its points to the shape boundary, the skeleton can serve as a representation of the shape. 2, 3, 7, 10, 12, 14, 16–19, 27

skeletonization Skeletonization is a process for reducing foreground regions in a binary image to a skeletal remnant that largely preserves the extent and connectivity of the original region while throwing away most of the original foreground pixels. 3, 14, 16, 17, 19

SLAM (S)imultaneous (L)ocalization (A)nd (M)apping is a technique used by digital machines to construct a map of an unknown environment (or to update a map within a known environment) while simultaneously keeping track of the machine's location in the physical environment. 21, 57

SSE Streaming SIMD Extensions (SSE) is a SIMD instruction set extension to the x86 architecture, designed by Intel and introduced in 1999. SIMD instructions can greatly increase performance when exactly the same operations are to be performed on multiple data objects. 99

Struck Framework for long-term object tracking using (Struc)tured Output Tracking with (K)ernels [48]. 23

SURF (S)peeded-(U)p (R)obust (F)eatures is a faster version of SIFT algorithm [9]. 21, 57

SVM (S)upport (V)ector (M)achines are supervised learning models with associated learning algorithms that recognize patterns, used for classification and regression analysis. 23, 24

TLD Refers to the long-term object tracking algorithm OpenTLD [57]. The initials stand for (T)racking, (L)earning and (D)etection. 23–26

Chapter 1

Introduction

Feature extraction and object detection are increasingly important in computer vision applications. We discuss the potential benefits of simple yet powerful methods that allow us to describe and detect objects in a robust manner, a topic that stands as the major driving force of this doctoral study.

We begin by presenting the motivation, the problem statement and its constituent sub-goals, the thesis statement and the contributions that support the novel aspects of this thesis. Finally, we conclude with an outline of the manuscript structure.

1.1 Motivation of the Problem

Research towards feature extraction and object detection is primarily motivated by a range of applications; they sit at the core of many "computer vision" applications, such as panorama stitching [21], augmented reality [151, 144], monitoring and surveillance [123, 122, 93], long-term tracking object tracking [58, 96, 94, 158], image retrieval [41, 148, 126], and shape analysis [11, 37, 137, 161], among many others. Both topics are extensive, and are growing in the field of computer vision and image processing. Given some knowledge of certain objects' appearance and an image of a scene

possibly containing those objects, object detection reports which objects are present in the scene and their whereabouts. The objects are detected and tracked by means of important invariant features, which in turn raises the challenge of developing feature descriptors with strong characteristics (e.g., rotational invariant and scale invariant). The term feature is considered an abstract representation and may be represented by a curve (e.g., edges [24, 75, 140], ridges [85, 88]), shapes (e.g., blobs [140, 143]), points (e.g., corners [140, 77, 9, 124, 23, 146, 90]), etc. In general, it can be used to represent and describe image content which is either a simple shape or a complex 3D object. Features allow computers to learn and recognize objects from different viewpoints, under transformations and following changes in shape, or even if they are cluttered by other objects or are partially occluded. Several different strategies have been proposed for object detection including divide-and-conquer search [67, 68], edge matching [12, 107], gradient matching [91, 115, 29, 97], recognition by parts [39], and feature-based methods [121, 141, 104, 152]. Approaches differ in their respective restrictions on the form of the objects analyzed (e.g., 2D or 3D objects), the matching scheme (e.g., geometry-based, appearance-based), and the knowledge they employ to solve the problem: appearance-based (i.e., based on the object's shape or appearance) [120, 28], or feature-based (i.e., based on the match of object and image features) [149, 105, 141]. In this thesis, we start with shapes represented by 2D binary raster images, continue with textured planar objects, and move to more complex 3D objects.

Since Blum's [13] introduction of the medial axis, medial representations of shapes have been studied extensively [136]. Several authors have offered definitions of the medial axis, their mathematical properties, their relationships to the object boundaries, and their representation in 2D and 3D [136, 31]. The medial axis can be seen as the set of points with more than one closest equidistant point to the shape's boundary. The medial axis of a shape gives a simple and compact representation of the shape. The medial axis with the corresponding distance from the boundary makes it possible to reconstruct the original shape. Geometrically, complex shape boundaries have a complex medial axis while simplified shape boundaries have a simpler medial axis. The medial axis of the simplified shape can be considered a simplified skeleton representation of the complex shape. Often, a simplified version of the shape is useful for shape matching [37, 137], shape segmentation [161], optical character recognition [59, 130], and fingerprint recognition [83], among others. One approach to shape simplification is to prune branches from a complex skeleton. Skeleton branches can be pruned by different means, e.g., by

distance or angle thresholds [53], by shape approximations [7], by smoothing the boundary, or by growing and shrinking the shape's boundary [45, 81, 70]. The definition of a desirable pruning outcome is not universal and hence an optimal automatic pruning solution does not exist to the best of our knowledge. Some of the existing algorithms create critical geometrical and topological changes to the skeleton's curves, creating more noise points, shrinking the final skeleton, disconnecting a skeleton's curves or removing some important branches of the skeleton. All these drawbacks decrease the effectiveness of the shape analysis in several applications [70]. Unfortunately, skeletonization algorithms of discrete shapes are particularly sensitive to noise and topological changes [7] because the quantized pixel locations cause excessive branching. The desired outcome of skeleton pruning is often application dependent, although all pruning techniques of shape skeletons aim at the simplification of the medial axis. Some output skeletons are shifted far from the medial axis to simplify their representation [45, 81, 66, 76], while others are discrete approximations of the medial axis [53, 4, 8, 76]. Connectivity [7, 135], topology [8], complexity [53] and automation [66, 164, 134] are other desirable outcomes. Our objective is a skeletonization and pruning solution that preserves the main features of the shape and its topological properties.

Besides shape, we also analyze the texture of 2D objects. The texture of an object contains information that can be used to identify a target of interest. The different changes of image intensities and contrast inside an object allow for different feature representations [140, 61, 77, 124]. In particular, keypoint features have many possible applications [21, 58, 104]. Their repeatability and efficiency determines how likely they are to be useful in a real-world scenario. Repeatability is important because the same scene viewed from different positions should yield features which correspond to the same real-world 3D locations. Efficiency is important as it determines whether the detector combined with further processing can operate at an acceptable frame rate. Also, the planarity assumption of the object allows for a homographic geometric transformation constraint between the matched keypoints. Several planar detection solutions have been studied [49, 104, 141, 152]; although they agree on the necessary steps for planar target detection (i.e., feature extraction, matching and robust computation of the geometric transformation), they differ in their application requirements, keypoint feature selection, matching scheme and the transformation estimation to account for mismatched keypoints. The application that motivates our textured planar object detection is localization. Localization is of key importance in mobile robot navigation. It is the process

by which the pose of the robot is determined from sensor data. It is well known that odometry and inertial navigation are not sufficient to maintain an accurate estimate of the robot's localization. As a consequence, many exteroceptive sensors, including sonar, laser, GPS, and vision systems, have been used for more accurate localization [98, 71]. There has been considerable research on the development of various localization methods. Many early successful approaches use artificial landmarks [74, 139, 20, 125], such as infrared light reflectors, ultrasonic beacons and visual patterns. Those methods provide a robust and stable solution for controlled environments, but are not appropriate for large spaces because of the need for fiducial markers. By contrast, vision-based schemes using natural landmarks are more appropriate for unmodified environments. They are generally low cost and require only a single camera to acquire images. Vision-based schemes extract invariant features from a scene and identify them within a map. In our application, we have access to maps, but we annotate the map with natural landmark locations and hence focus on localization based on the natural landmarks. The choice of planar targets is justified by the possibility to generate different synthetic views of the target during the training phase, which in turn allows for an evaluation of the planar target for accurate localization. This enables rejection of targets that are not expected to work well. While any 3D targets can be used, an accurate 3D model is required to calculate the perspective projection into the image plane. In indoor environments, planar targets abound and can be easily measured with a ruler for metric localization. Changes in the environment that clutter the scene when the robot tries to localize a previously seen landmark are another concern in localization. Viewpoint variance (i.e., changes in translation, rotation, scale), noise sensitivity, partial occlusion and illumination changes are major challenges for pose estimation and accuracy of localization. Most existing planar object detection solutions do not evaluate their outputs as they relate to these challenges [104, 141, 82].

The use of fast invariant keypoint features for planar object detection is a good balance between accuracy, speed and memory usage during our planar object work. They provide a set of strong characteristics that benefit the object detection solution. Therefore, we find it natural to develop them to detect more than just planar objects. Detecting and localizing 3D objects is a logical next step in our research. The 3D object constraint adds complexity to the learning and detection steps. The geometrical model can no longer be represented by a homography, and a single image view is not enough information to learn the geometry of the object. Memory usage and feature

descriptor representation become a challenge; more information needs to be stored to account for different views of an object. A larger database consequently starts to affect the time performance of the detection. We found that the existing feature-based 3D object detection solutions [92, 131, 121] are slow with high memory usage. In prior work, the representation of the descriptions grows exponentially with the amount of features [105, 17]. Most of these solutions need prior knowledge of the object's 3D model or an enumeration of all the object's possible views to account for camera viewpoint changes [105, 121]. This makes for a longer learning process in order to account for the object's transformations(i.e., rotation, scale, perspective).

Finally, we extend our work from 3D object detection to long-term tracking object tracking of 3D objects. Object tracking is the process of locating a moving object over time. The algorithm analyzes sequential video frames and outputs the movement of targets between the frames after the object is initially detected. Visual tracking is one of the fundamental research areas in computer vision. Tracking has several applications, such as video compression [35, 116], augmented reality [106], traffic control [55], surveillance and security [101]. Although some settings allow for strong assumptions about the target [87, 50, 118], most often tracking an object with no prior knowledge is desirable. Model-free tracking consist in the online learning and adapting of a representation of a given target. The tracker analyzes sequential video frames and after the initial detection of a target, outputs the successive positions of the object from frame to frame. Even though progress has been made in model-free trackers, fast and robust model-free tracking is still a challenging problem due to geometric transformations, changes in illumination, fast motions, noise, occlusions and background clutter. We will now discuss the differences between planar, 3D object and long-term tracking tracking. Long-term object tracking is a challenge because we need to learn a motion model for 3D objects. Planar object tracking simplifies the process because an affine transformation or homography can define the motion model between consecutive frames. Also, only one image of the object is needed, which is in fact the initial detection because its appearance will always be the same. On the other hand, 3D objects increase the complexity of the problem because the change in orientation and position alters the appearance of the object. In particular, it makes the learning process difficult because there is only an initial view of one angle of the object. Additionally, the object needs to be separated from the background during the learning process. Occlusion and object in-plane rotations make it more complicated to specify the motion model, and cause tracking failure and shifting

of the real position. Note that although tracking is also composed of object detection, there is no prior knowledge of the object model other than an initial selected area in the frame, the object model is learned live.

A considerable amount of work has been dedicated to monocular single-target visual tracking. Among the relevant work are model-free tracking solutions that use sparse object representations [58, 56, 96, 108], histogram-based representations [6, 51, 52, 48] and combinations of them [58, 57]. Performance measures have also been proposed and different benchmark datasets to evaluate visual trackers are available [158, 63, 26].

Recently, the tracking-by-detection paradigm has obtained excellent performance in the robust tracking of targets [46, 6, 48, 51, 96, 52, 57]. This tracking framework integrates an online learning component that acquires new information from each successful target detection. Recent correlation filter-based trackers [14, 51, 42, 52] replace the time-consuming convolution operation in the time domain by pointwise multiplication in the frequency domain. Henriques et al. [51, 52] extended the correlation filter by exploiting the circulant structure of the tracking object to quickly incorporate information from all cyclically shifted samples into the Fourier analysis without iterating over all possible samples in the target's neighbourhood in the time domain. Furthermore, Henriques et al. incorporated multiple feature channels instead of raw pixel into their correlation filter framework to improve the accuracy and robustness of the tracker [52]. However, the correlation filter tracker uses a fixed size template and hence the tracker is not able to handle scale changes of a target occurring during motion.

1.2 Problem Statement

The previous section highlighted two core issues: the process of selecting certain features with strong properties that describe an object and the process of using such features to detect the object. These two issues are intertwined in the formulation of the overall thesis goal.

We define our problem as twofold: first, we would like to extract a certain feature that best characterizes a given object. The feature should be invariant to object transformations and robust to noise, which will allow the use of the feature for detecting the object, describing it, or categorizing it to a certain class. Second, with some knowledge of how certain objects may appear, we would like to identify which objects are present in a given image that possibly contains those objects and their location, all the while keeping track of them. The recognition should be invariant to view point changes and object

transformations. The solution should be robust to noise, occlusions and illumination changes.

The two problems of the general thesis goal have been widely worked on. Both can be found in different ways, depending on the assumptions bound to the applications they are being developed for. This thesis provides a set of algorithms that address both problems. It centers on a selected group of applications with practical relevance to computer vision. The thesis is not meant to serve as a complete, generalizable form of theoretical and pragmatic solutions to the problem of feature extraction and object detection. Rather, it aims to promote future research in both topics and push forward state of the art solutions for this topic.

For a better understanding of the pursued research avenues, the goal of our thesis consists of the following four sub-goals: the first sub-goal is skeleton pruning for shape analysis, the second is the use of planar object detection for localization, the third is 3D object detection using an uncalibrated camera, and, finally, the fourth sub-goal is long-term tracking 3D object tracking.

1.2.1 Skeleton Pruning for Shape Analysis

Shape analysis is the process by which a computer analyzes and processes geometrical shapes. In digital form, the shapes are most commonly described using a boundary representation (i.e., a circular list of concatenated points). One of the problems in shape analysis is the creation of shape descriptors. Among the most important shape descriptors, we find the medial axis or skeleton of the shape. The medial axis is the set of all points with more than one closest point on the object's boundary. The shape's digital representation as well as boundary noise pose challenges to computing the shape's skeleton. Many noise branches can be generated during its computation due to boundary noise and rigid shape transformation. Our purpose is to compute a clean representation, which conserves the topology of the shape, and which is invariant to noise. Connectivity is also a main concern; a shape can only be represented by a single skeleton. The topology of the shape and skeleton must stay unaffected when rigid transformations or significant boundary noise are applied to the original shape. The algorithm must handle complex shapes with holes as part of their morphology. An accurate vector representation of the skeleton is needed for compact shape description.

1.2.2 Planar Object Detection for Localization

In this sub-goal we address the problem of developing keypoint-based visual localization for planar natural landmarks. Mobile navigation systems are often equipped with different sensors such as GPS, sonar, laser range finders and vision. Unfortunately, some of these devices do not provide an accurate estimate of the robot's localization, which is needed for different environments (i.e., indoors). Displacement errors are also a known problem in mobile robots that travel long distances without auto correction (i.e., drift can occur). The selection of the correct invariant keypoint features and proper learning techniques is the main concern to generate a stable planar detector that is robust against illumination changes, perspective distortions, motion blur and occlusions. The solution must account for unmodified environments because of the deployment cost of artificial landmarks such as bar codes, infrared reflectors and visual patterns. Schemes that use natural landmarks are more appropriate for places that cannot be modified and generate a lower cost (i.e., only a single camera is needed). An evaluation of a chosen landmark is also important to access the accuracy and viability of its usage for localization. Planar object detection by itself is a widely studied topic. However, the localization application adds certain constraints to the problem (i.e., wide changes in view, motion blur, occlusion and cluttered environments). More complex solutions address the problem by mapping the environment while the robots navigate, creating a 3D model of the space. Mapping solutions in embedded systems pose a challenge because of their memory and energy requirements due to the large models and big computations.

1.2.3 Rigid 3D Object Detection and Localization Using an Uncalibrated Camera

This sub-goal consists of a real-time application for 3D object detection. The solution needs to be independent of camera optics (i.e., no calibration process needed) and to account for object mobility. The camera is intended to pan the real world in search of instances of previously learned objects. The 3D objects, once detected within the frame, should be tracked and followed, even if they are subject to perspective transformations. An important additional challenge to the problem is the memory usage; the solution should be able to run in embedded systems where memory usage is a concern. The geometry of non-planar objects is to be learned from a set of images. The object at hand can be mobile and change appearance while the camera is also moving. This potentially means larger databases, more memory usage and slower search speed while classifying

features. Also, the planarity assumption does not hold anymore between the different views of the object. Memory usage and speed are two dominant challenges in detecting 3D objects among existing frameworks. Therefore, our problem statement is to create a real-time single camera video-based application capable of detecting moving objects from an uncalibrated mobile camera. The goal is to create compact feature descriptions, and therefore generate less memory usage without affecting performance. The detection should be invariant to viewpoint changes, and robust to noise and image illumination changes, while accounting for partial occlusions and cluttered environments.

1.2.4 Long-Term 3D Object Tracking

For this final sub-goal, we focus our interest on the implementation of a real-time long-term 3D object tracking algorithm. The idea is that after the object is initially selected, the learning process begins while the algorithm keeps track of the object while it is in the frame and not occluded. The algorithm must detect the object in question even if it disappears from the frame and reappears. 3D objects increase the complexity of the problem because of the change of their appearance, orientation and position. Our solution must be able to learn the different appearances of the object live, after an initial view of one angle of the object is selected in the video sequence. There is no prior knowledge of the object model other than initial frame selection. The object tracking solution must handle occlusion and object in-plane rotations that could cause tracking failure and shifting from the real position. An important additional challenge in the problem is to adapt to different scale transformations while separating the target from the background. Although a variety of tracking algorithms exist in the current literature, they have different strengths and weaknesses in different scenarios. Experimental comparisons between available state-of-the-art solutions and our solution are presented to validate accuracy and performance.

1.3 Thesis Statement

The goal of this thesis is to introduce and promote algorithms for feature extraction and object detection. This goal is achieved by means of the creation of fast and invariant feature extraction solutions and experimental verifications. Fast feature extraction and small representations promote real-time applications with small memory footprints. Invariant features reduce learning time and allow the tracking and detection of objects from different viewpoints and under different orientations, even if they are partially occluded

or in cluttered environments.

1.4 Contributions

The thesis makes the following contributions:

1.4.1 Skeleton Pruning by Contour Approximation and the Integer Medial Axis Transform

A novel skeleton pruning solution is created by means of the integer medial axis transform and contour approximation of the binary shape. The shape's boundary is approximated by a piecewise curve, which helps the process of removing unwanted branches from the integer medial axis. The novel pruning algorithm effectively removes unwanted branches and conserves the connectivity of the skeleton. It is suitable for automatic real-time applications and outputs a subset of the integer medial axis. The solution respects the geometric and topological properties of the skeleton's curve. Therefore, similar shapes have similar skeleton representations. The algorithm is robust to significant boundary noise and rigid shape transformations, and is easy to implement (i.e., significant changes to the shape do not alter the final skeleton result). High accuracy reconstruction of the shape is possible from the pruned skeleton and the distance map created while computing the medial axis of the shape. In our experimental validation, our pruning solution outperforms existing state of the art solutions in terms of stability (i.e., the result is more stable even if the shape undergoes a rigid transformation or boundary noise is added). Finally, an automatic vector representation of the skeleton is created ensuring that the curve is preserved inside the shape's boundary. The vector approximation of the skeleton uses automatic threshold value selection thanks to information from the distance map to automatically select the correct thresholds for the curve approximation. The research endeavours in this area led to a publication [85].

1.4.2 Framework for Natural Landmark-Based Localization

The proposed vision-based framework integrates and evaluates the use of invariant keypoint features and of the Fern classifier for localization applications. For the planar object detection, we include stratified sampling, off-line target evaluation, planarity check, and pose estimation of matched points to improve the accuracy of the localization. The framework assumes the planarity of the natural landmark. An off-line target evaluation

is capable of determining if the planar object can be used as a natural landmark for the localization and evaluating the precision to expect. The use of the Fern classifier proves to be robust against illumination changes, perspective distortion, motion blur and occlusions. Our experimental results validate the accuracy and speed of localization (i.e., real-time application). Although the experimental evaluation is realized with the Fern classifier, the framework is independent of the classifier (i.e., different techniques could replace Fern). The framework improves the detection rate by up to 20 percent, compared to using only basic Ferns. Also, analyzing the planarity constraint decreases the false detection rate by up to 10 percent. Stratified sampling in the image increases the localization scheme while searching for objects in cluttered environments; it doubles the detection rate of the target. The framework also improves pose estimation by up to 50 percent with the use of the invariant feature matching. We demonstrate that visual localization is feasible when using planar object detection, and we show how our contributions can benefit the final results. The framework recorded an average 3 ± 0.66 (95% Confidence Interval) centimetre translation error and an average 4 ± 0.52 (95% Confidence Interval) degree rotation error in ranges up to 3 meters using target dimensions of 27x20 centimetres. The research endeavours in this area led to a publication [89].

1.4.3 A General Framework for Fast 3D Object Detection and Localization Using an Uncalibrated Camera

We present real-time 3D object detection using a single, mobile and uncalibrated camera. The algorithm uses keypoint feature-based methods based on two novel naïve Bayes classifiers for viewpoint and feature matching. A new feature descriptor combines binary descriptors with a Bayes classifier. The new classifier exploits the specific structure of various binary descriptors in order to increase feature matching while conserving descriptor properties (e.g., rotational and scale invariance, robustness to illumination and real-time performance). Thus, the performance we achieved with the novel classifier is equivalent to that of the Fern classifier. At the same time, we boosted the speed thanks to the use of binary descriptors. Efficiently encoded features considerably reduce the memory footprint of the new classifier. An improved indexing scheme to speed up the matching process is also included among the contributions of this framework. The framework only needs a few images of the target object to learn its geometry. It demonstrates a better performance than the binary descriptors alone, and reduces memory footprint in comparison to the Fern classifier. Experimental results confirm the effectiveness of the

proposed solution. The research endeavours in this area have been published in WACV 2015 [87].

1.4.4 Scalable Kernel Correlation Filter with Sparse Feature Integration for Long-term Tracking

We extend the Kernelized Correlation Filter (KCF) tracker with the capability of handling scale changes by introducing adjustable Gaussian window functions leading to increased accuracy and robustness. The window filtering does not alter the pipeline of the correlation filters allowing it to use any variation of the original kernels, i.e., Gaussian, polynomial or linear. Our solution achieved up to 4x speedup over the original solution while producing superior results and reducing the memory footprint by half. The boost in achievable frame rates are realized by introducing Intel's Complex Conjugate Symmetric (CCS) packed format into the correlation filter framework and fast HOG descriptors using SMID (SSE) instructions to perform operations on multiple data points simultaneously. Finally, we conducted experiments to compare the original KCF algorithm against our solution using the dataset from the original paper [158]. These experiments demonstrate the gain in performance in terms of success rate, precision rate and speed. Also, the empirical experimental evaluations are extended with the VOT challenges for a more accurate representation of the performance gain. Our proposed solution achieves performance gains in accuracy, robustness, and speed compared to state-of-the-art trackers. The research endeavours in this area have been published in ICCV Workshop, VOT-Challenge 2015 [86, 64, 38].

1.5 Outline

The remainder of this thesis is structured as follows. Chapter 2 takes the reader through previously published research that are relevant to this thesis, starting with approaches on skeleton pruning, then taking a closer look at planar object detection and localization, and continuing with 3D object detection and localization. Finally, we examine long-term object tracking solutions.

Chapter 3 is devoted to skeleton pruning by contour approximation and the integer medial axis transform. Chapter 4 addresses planar natural landmark-based localization. Chapter 5 deals with fast 3D object detection and localization using an uncalibrated camera, and Chapter 6 with scalable kernel correlation filter with sparse feature integration for long-term object tracking. Chapter 7 wraps up the thesis and explores potential

future research directions.

Chapter 2

Related Work

We begin by reviewing existing work that relates to the sub-goals mentioned in our introductory chapter. We touch upon different areas of research, which we attempt to group appropriately into four sections: shape analysis, localization, keypoint descriptors and object detection, and object tracking.

In Section 2.1, we review the current state of algorithms that compute and prune the skeleton of a binary shape. We cover the main categories of skeletonization algorithms and their current state while discussing their performance. We expose their drawbacks and the problems of computing the skeleton of a binary shape. While reviewing the related work, we found that there was a need for a method to prune the skeleton of noisy images. Current solutions do not preserve the geometrical and topological properties of the shape; this behaviour is not acceptable in several applications. Unfortunately, skeletonization algorithms of discrete shapes are sensitive to noise and topological changes because of the quantitative pixel locations and excessive branching. Our goal is to create a pruning solution that preserves the main features of the shape and its topological properties. It needs to be fast to be used in real-time applications, as well as capable of being used in an automated manner. Automatic solutions remove the need for tuning

input parameter configurations and do not require previous knowledge of the shapes in the image dataset.

For our second sub-goal in this thesis, we move from binary shapes to textured planar objects. We envisioned using natural planar targets to maintain an accurate estimate of the robot's localization in map-based navigation systems. The odometry and inertial navigation of the robot are not sufficient. Many early approaches to solving the localization problem use artificial landmarks such as infrared targets, light reflectors and ultrasonic beacons. By contrast, vision-based schemes that use natural landmarks are more appropriate for unmodified environments, and are also low cost as they only require a single camera to acquire images. We identified that the use of planar targets is an asset for robot localization. The possibility to generate different synthetic views during the training phase allows for accurate detection and evaluation of the target. In Section 2.2, we review the related work for robot localization. We study map-based navigation systems that relate to our research and examine how they differ from our work. We develop a framework for planar object detection for localization.

Section 2.3 reviews work related to feature-based recognition and object detection systems for planar and 3D objects. We review the main keypoint feature extraction techniques currently available in object detection algorithms. Working with 3D objects adds complexity to the learning and detection steps. Memory usage and representation of invariant descriptors become a challenge, because more information needs to be stored to account from the object's different views. Current solutions have limitations in terms of memory usage and performance (i.e., detection speed). As a result, we developed a framework for 3D object detection that addresses these limitations.

Finally, Section 2.4 reviews work related to the fourth and final sub-goal of our thesis. We review the state of the art solutions for long-term tracking object tracking using a single camera. In this section, we discuss the limitations of current object tracking solutions. Our goal is to create a fast long-term tracking tracking solution that can be used in real-time applications while accurately tracking the target (i.e., changes in position and scale). We compare the existing solutions using the two most well known annotated datasets available to date: the Visual Tracker Benchmark and the VOT Challenge datasets. Although current long-term tracking tracking solutions present competitive performance results, there is still a need for improving their tracking capabilities. We found that correlation filters for long-term tracking visual object tracking provide a good platform due to their performance. However, the correlation filter uses a fixed size tem-

plate and hence, the tracker is not able to handle scale changes occurring during motion. As a result, we implement a Scalable Kernel Correlation Filter with Sparse Feature Integration to remove the fixed size limitation while improving accuracy and speed.

2.1 Shape Analysis

Many algorithms exist to compute and prune the skeleton of a binary shape. The main categories of skeletonization algorithms are shape thinning [66, 164, 134, 47, 76], Voronoi-based analysis [7, 70, 78, 103, 99], and algorithms based on the feature transform [53, 7] or on the distance transform [4, 8, 47]. In shape thinning [66, 164, 134], pixels are peeled from the boundary shape according to a set of rules relating to a pixel's neighbourhood. The process is repeated until there are no more pixels to remove from the shape and the shape skeleton remains. This solution conserves the connectivity of the final skeleton representation and the topological properties of the shape. It also works automatically, because there is no input parameter selection. On the other hand, although its output is a skeleton with many desirable properties, it fails to compute the actual medial axis. As a result, most thinning algorithms don't save the distance to the boundary [132] and the shape reconstruction from the skeleton is not possible [7, 66]. Also, shape thinning algorithms are often slow because of the re-iterative process of contour detection and pixel removal.

Another type of algorithm is based on Voronoi diagrams [100, 80, 76, 79, 154]. This solution samples points on the boundary and computes the Voronoi regions of these points. The boundary of the regions inside the binary shape is an approximation of the skeleton of the shape. The skeleton approximation becomes closer to the exact solution as the number of boundary points goes to infinity. Voronoi-based algorithms are typically quite complex due to the sampling of the boundary pixels. As a result, many algorithms in this category are slow [7, 100]. Martinez et al. [78] propose two variants to compute the Voronoi regions that simplify the boundary sampling problem, but they still need a post-processing step to prune their skeleton. Liu et al. [76] propose the use of a Voronoi diagram to create their Extened-Grassfire Transform that keeps track of the distance from each Voronoi vertex to the boundary. Jonás Martínez et al. [79] introduce a stable skeletal representation applied to orthogonal shapes only. Wekel et al. [154] consider the Voronoi diagram of the boundary vertices to extract the skeleton as a subset of the Voronoi edges using a classification scheme. A parameter allows them to control the skeleton's sensitivity to perturbations in the boundary curve. Aside from the desirable theoretical

properties of their solution, it is an interesting mixture between thinning, Voronoi-based, and feature transform based algorithms. However, initial sample selection influences the outcome.

The third type is based on the mapping of the feature transform [53, 44] or distance transform [4, 8, 47]. The feature transform computes the closest boundary point for each pixel inside a shape, and the distance transform computes the distance to the nearest boundary pixel. On one hand, these algorithms can be fast. They detect the medial axis of the binary raster shape accurately, and shape reconstruction can be easily achieved. On the other hand, noise sensitivity and pruning techniques that conserve connectivity and skeleton topology are challenging with these types of algorithms. Sanniti di Baja and Thiel [8] present a topology preserving pruning solution to this problem. They prune peripheral skeleton branches based on a comparison between the shape reconstructed with the inverse distance transform applied to the skeleton before and after removal of a branch. They only remove complete peripheral branches if the change in shape is below a threshold related to the length of the branch.

Pruning is an essential part of most skeletonization algorithms because of the excessive branching in noisy binary raster shapes. Different pruning techniques have been proposed, which fall mainly into two categories: first, those that pre-process the shape's boundary [45, 81, 132, 84, 110] (e.g., boundary smoothing), and second, those that aim to remove unwanted skeleton points according to some classification criteria [53, 7, 132, 76]. Shaked and Bruckstein [132] give a complete analysis of pruning methods and explain their drawbacks. With the algorithms that fall into the first category, the smoothing of boundaries may cause some undesirable effects on the skeleton, e.g., skeleton shift, and may violate the topological properties, e.g., add new branches to the skeleton [44]. In particular, this category has significant problems in distinguishing between noise and low frequency shape information on boundaries [7]. The scale-axis-transform [45, 81] is a different kind of smoothing method that uses a noisy pre-computed medial axis. In 2D, it grows the original shape by linearly scaling the radii of the medial disks and then removing branches of the medial axis that are covered by the larger disks. It preserves the main features of the shape and it reduces low frequency boundary noise. However, it also simplifies the topology of the shape, i.e., it succeeds in creating a new simpler skeleton representation of the shape by simplifying the topology of the shape (e.g., shape holes are occluded when the shape is grown). As a consequence, the skeleton may be shifted far from the medial axis of the original shape. This behavior of the scale-axis-transform

may be desirable for some applications, notably level-of-detail tasks, but does not meet our goals.

Liu et al. [76] extend the original idea of the medial axis transform and add the notion of the center of a shape in their Extended Grassfire Transform. They introduce novel significance criteria for pruning the extended medial axis with possible applications in shape alignment and shape matching. Their experiments show excellent results for shape alignment and matching. However, their pruning technique does not preserve connectivity of the extended medial axis and their solution does not preserve topology [76].

Bai et al. [7] introduce a new concept similar to our solution. They pre-process the shape boundary using a discrete curve evolution to prune the final skeleton. We approximate the shape with a simpler method (i.e., piecewise curve approximation) and introduce a more flexible pruning technique, which is also more stable for noisy shapes and shape rotations. Like ours, their solution is capable of conserving the topological properties without shifting the medial axis of the discrete shape or shortening branches. However, theirs relies on "a priori" knowledge of the shape to prune it, i.e., the number of sides of the polygon to approximate the original shape is required as an input parameter for their DCE method (see Figure 2.1). Shen et al. [135] introduce a different significance measure for skeleton pruning without boundary approximation that outperforms the DCE algorithm. The algorithm does not rely on "a priori" knowledge but its time complexity is higher than that of our solution. The authors report that their algorithm takes around 7 seconds on average to extract a pruned skeleton from [512 x 512] images [135] from the same MPEG-7 dataset. The MPEG-7 dataset consists of 70 classes, each having 20 members of binary shapes. Our solution extracts the pruned skeleton of the 1400 images with resolutions from [256 x 256] to [950 x 800] in 0.1 seconds on average (see Section 3.2). Images with a resolution of [512 x 512] take about 0.12 seconds on average.

Similarly to our work, Philip Buchanan [22] presents an automatic algorithm for extracting vector information from raster images. His algorithm proposes a heuristic for the creation of 3D mesh from a single piece of non-occluding 2D art by extracting its skeleton structure. The topology generated by this algorithm differs from the topology of algorithms that produce geometric centres such as our work. Buchanan's goal is to enhance content manipulation and develop content creation from the extracted topology rather than creating a subset of the medial axis.

Laura Pinilla et al. [109] create a new skeleton pruning solution based on our work.

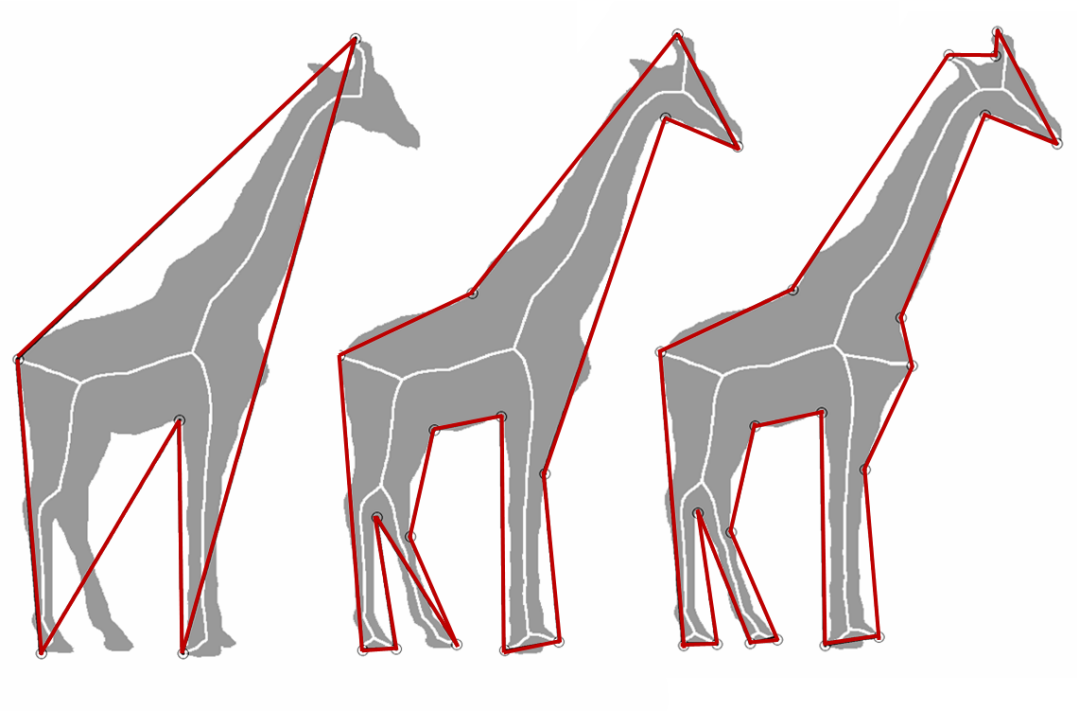


Figure 2.1: Pruning using Discrete Curve Evolution DCE. Left: 5 vertices, Middle: 15 vertices, Right: 20 vertices.

They introduce a method for dominant point detection during the contour approximation and add additional conditions on the pruning step. The authors claim to hinder the speed performance in order to obtain a more stable skeleton for a wide range of shapes using a single threshold instead of two, as per our solution. Ke Wang et al. [153] use our pruning solution to analyze the shape of tropical cyclone eye areas. The authors claim to achieve 92% agreement with the reference data when detecting cyclone eye areas. Furthermore, Pérez Rocha [117] uses our skeleton pruning solution in her work on 2D shape segmentation and line filling to automatically generate embroidery design patterns for machine embroidery.

The main challenge in pruning unwanted skeletal points is selecting a good removal threshold. The removal of a point on the medial axis needs to be based only on local information of the surrounding medial axis points [7]. The global information of the shape is usually discarded during skeletonization, causing topological changes and shortening of the branches [132]. Discerning noise in the data of a digital skeleton is still an open problem. Hesselink and Roerdink [53] introduce three pruning techniques with their Integer Medial Axis (IMA): constant, linear, and square-root pruning. However, all of them may disconnect the skeleton and may shorten branches (see Figure 2.2).

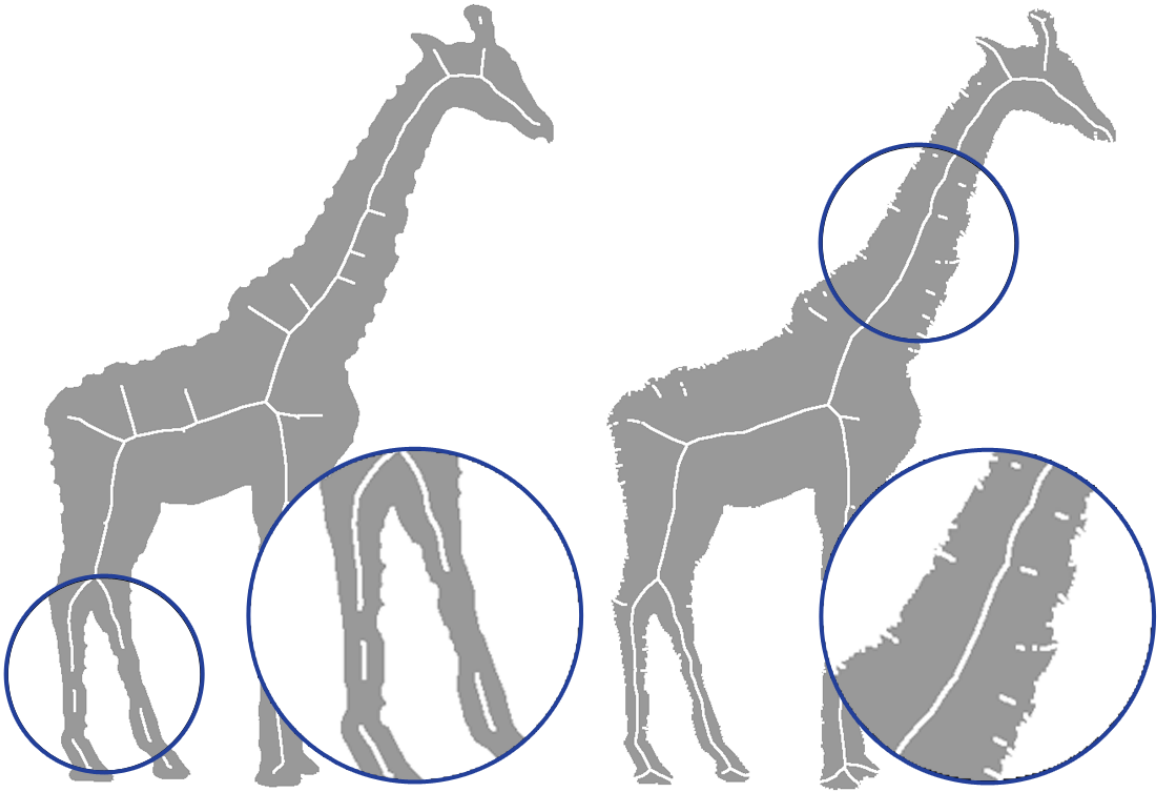


Figure 2.2: Distance and angular pruning, both are sensitive to input parameter selection creating disconnected skeletons.

2.2 Localization

Navigation and localization systems can be roughly divided into those that rely on *Map-based navigation* and provide the robot with models of the environment with different degrees of detail, and those that rely *Mapless navigation* and perceive the environment as they navigate through it. The method presented in this section falls within the first category. We refer the reader to surveys on robot navigation and localization systems found in [16, 34, 15] for details. Vision-based localization systems have made significant progress among other mobile robot localization techniques, particularly for indoor applications. In many cases, the localization problem is two-dimensional, i.e., the robot position on a plane and its heading [65, 5, 138, 32, 142]. Other applications require vision-based methods that provide a six degrees-of-freedom camera pose. The localization problem is related to the problem of structure-from-motion (SFM), which consists of inferring the environment structure and camera motion under the assumption of *small* baseline motion.

SFM-based localization for SLAM has been addressed, e.g., in [30, 113, 36, 27, 157, 60]. In general, SFM-based localization does not consider the kidnapping problem, i.e., that the robot may be picked up and placed in a different location.

Se et al. [128, 129] present a trinocular SLAM system using SIFT descriptors as landmarks. The landmarks are matched between the three views to reconstruct the 3D points. Their system also uses odometry-based initialization to estimate the ego-motion of the camera. Wuest et al. [159] use the FAST features detector [119] and model the probability distribution of each feature. Each feature is then tracked successfully by means of a finite set of Gaussian mixtures. An extension of this method for a large number of features proves difficult because of the use of a Gaussian kernel. Also, their visibility modeling only considered camera translation, while orientation is important for localization as well. Closest to our localization method is the system presented by Alcantarilla et al. [3] who use a non-parametric learning framework to predict each feature's visibility with respect to the varying camera poses.

Our work evaluates the use of textured planar objects as natural landmarks for the problem of localization. Through our experimental results, we show that our framework, with the use of planar objects, obtains strong robot location accuracy. We show how our framework reacts under viewpoint changes, noise sensitivity, partial occlusion, and illumination changes while reporting the accuracy of pose estimation and localization.

2.3 Keypoint Descriptors and Object Detection

Feature-based object recognition systems have been extensively studied in the literature [121, 141, 104, 152]. They strongly rely on feature matching schemes to recognize and localize objects in different views. Different matching schemes have been created to associate features and compute their similarities. The two main approaches in the literature are keypoint descriptors and classifiers.

A keypoint descriptor is a function that is applied to the neighbourhood of a keypoint location, creating a compact signature that describes the keypoint in an image. The compact signature is meant to be an 'unique' representation of a keypoint that is later used to match with another keypoint in a different image based on their similarity (as measured by, e.g., L1-norm, Euclidian norm, Hamming distance, etc.). Descriptors of a keypoint may be invariant to changes such as rotation, illumination, noise, scale, etc. They can be classified as binary (i.e., BRIEF [23], ORB [124], FREAK [146], BRISK [72], HIP [141], etc.), or gradient-based (i.e., SIFT [77], SURF [9], HOG [29], etc.). Unfor-

tunately, gradient-based descriptors, although considered to be the gold standard, are computationally expensive and less appropriate for low-powered embedded platforms in their original form. Conversely, binary descriptors show accuracy while achieving fast performance, have low memory requirements and are simple to implement. They can take advantage of CPU and GPU specific instructions for their implementation and matching [23], which makes them even faster. They have been effectively used in mobile platforms for planar object detection [141, 10, 1].

Classifiers treat the matching/similarity problem as one of classification [19, 17, 104]. They improve classification rates while reducing computation after the feature database is learned in an off-line training phase. Some classifiers, such as Random Forrest and Ferns, learn the feature distribution of a database by means of randomization [17]. The off-line learning step ensures a better encoding of the most relevant features and their structure from very significant perspective changes. One of the leading approaches in feature classifiers is Ferns [104, 152, 150].

Fern classifiers learn the distribution of binary features around a relevant location using multiple views of the same keypoint. Each binary feature represents the comparison between an image's intensity from two random positions around the keypoint. Ferns solve the classifying problem by maximizing the posteriori probability, given a set of binary feature observations around the keypoint. Ozuysal et al. [104] propose a Semi-Naive Bayes Classifier as a trade-off between complexity and performance. They assume that groups of binary features are conditionally independent.

Ferns are created assuming dependency between binary features and by randomly grouping the features (by choosing the size of a Fern group and the number of groups to describe a keypoint). Finally, the conditional distributions for each group (Ferns) are learned in an off-line phase by warping the image patch around each keypoint using multiple random transformations. The learned distributions are used as a lookup table at run-time to speed up the classification problem. Ferns' drawbacks are a large memory footprint and the off-line learning phase. The memory requirement is exponential in the Fern size, i.e., $2^{fernSize} \times nFerns \times nKeypoints \times sizeof(data)$. For example, if we train for 1000 keypoints ($nKeypoints$) using 50 ferns ($nFerns$) of size 11 ($fernSize$) with 4 bytes floats ($sizeof(data)$), storing their conditional probabilities results in a 400 MB database. Because of the random selection of binary features, the training phase needs to generate all possible transformations to be able to classify from arbitrary viewpoints, which is time consuming.

Some modifications to the original Ferns have made possible their inclusion on mobile platforms [150, 152]. Most of these modifications target the memory consumption by tuning the Fern size, the number of Ferns and the data used to store the conditional probabilities, but Ferns are inherently exponential in size. Also, the improvements do not address the off-line learning phase. Daniel et al. [152] combine PhonySIFT and PhonyFerns for a mobile platform by executing both in the same pipeline for object detection. Other approaches to adapt Ferns for mobile platforms perform computations on a remote server. However, these approaches suffer from low performance due to restricted bandwidth and high power consumption given the required network communications [152, 147]. Ventura et al. [147] associate 2D locations in the video frame with 3D locations in the real world, which requires building and updating a 3D model.

Our work combines descriptors and classifiers in one single matching scheme. We show through our experimental results that a full naive Bayes classifier, combined with binary descriptors, reduces the training time and the memory usage, and improves the matching rate of the descriptors while performing as well as the original Ferns.

2.4 Long-term Object Tracking

Numerous tracking-by-detection algorithms can be found in the current literature. Correlation filters for long-term tracking visual object tracking have seen great interest recently. Although they present competitive performance results, there is still a need for improving their tracking capabilities. Many of them have been evaluated with the Visual Tracker Benchmark [158] and in the VOT Challenges [63]. We restrict our review here to those tracker that are close to our work, including TLD [57], CMT [96], Struck [48], SCM [163], Alien [108], CSK [51], KCF [52] and SAMF [73].

Struck [48] is a tracking-by-detection algorithm, which relies on a kernelized structured output Support Vector Machine (SVM) in order to distinguish between tracked object and background.The binary classifier employs a Haar-like vector representation of the target to solve the classification problem. It uses six normalized (i.e., [-1,1]) Haar-like features arranged on a 4x4 grid at two scales resulting in 192 features to represent the model. At each frame, the algorithm tries to detect the object using a structured output SVM. A loss function based on bounding box overlap is used to determine the similarity between transformations. A threshold radius defined by the user delimits the search area of the tracked object. The structured output learning approach allows the tracker learn a prediction function rather than a classifier. The function directly estimates the

object transformation between frames. The structured output SVM also holds a number of support vectors, which are images that are classified as either positive or negative. Struck is capable of detecting partial occlusions but, if the target object is occluded for too long, the SVM will decrease in performance as it is learning the occluded image.

SCM [163] combines a binary classifier with a generative model to achieve high accuracy and robustness. The binary classifier compute the confidence value by assigning more weights to the foreground than the background. The generative model uses a histogram-based method that takes the spatial information of each patch into consideration with an occlusion handing scheme. The algorithm considers the latest observations and the original template to deal with appearance change and alleviate the drift problem.

TLD [57] is a long-term tracking object tracking method that combines three components, as its initials state, i.e., tracking, learning and detection. The overall idea of TLD is to combine tracking and detection results while learning the different appearances of the object. The detection and tracking components are always running in the video sequence. The algorithm's output is obtained by merging tracking and detection outputs. The object location and size is estimated from the tracking points and the detection output. Location and scale estimation are selected as the median value transformation between all pairs of successfully tracked points. The output area is associated with a confidence value to evaluate the performance of the tracking at each frame. If the confidence value is lower than a specified threshold, the target is classified as lost or occluded. At this point, the tracking component is stopped and it is restarted when the object is detected again. The learning component is enabled when the output area is associated to a high confidence value. A high confidence value is generated when the tracking and detection outputs concur on the location of the object. The tracking component uses a forward-backwards strategy based on KLT. During the tracking phase, the optical flow of equally spaced grid points inside the bounding box is computed and back projected to the initial frame. The distance error between the original grid points and their back projection should be smaller than a specified threshold to account for a good match. Matching points from the consecutive frames are also checked with normalized cross correlation (NCC) using the patch around the points. The detection phase uses a cascade classifier search for template matching and Fern classifiers that account for random binary checks inside the tracked area. Each area with a high/low confidence value is considered a positive/negative template that is fed to the cascade classifier for future matching and detection of the object. A threshold is used to define the maximum

number of templates to accept by the learning process.

CMT [96] uses a similar tracking strategy to TLD to estimate position, scale and also includes rotation. Instead of equally spaced points a keypoint-based matching strategy using BRISK [72] features and descriptor is implemented. With each frame, BRISK keypoints are extracted from the entire frame and matched against the object model. They are matched based on the Hamming distance between their BRISK descriptors. CMT uses a keypoint object representation based on the geometric constellation of the keypoints around their centroid. Keypoints are also tracked from the previous to the current frame using a KLT optical flow. The algorithm learns the different appearance of the object by employing an adaptive tracking technique that increases the number of keypoints in the model database. The position, scale and orientation are estimated from every pair of matched keypoints between the frame and the model. The final transformation is stated as the median of all the estimated values of location, scale and orientation from each pair.

Alien [108] uses a similar keypoint-matching method to CMT, but it represents the objects by two set of keypoints i.e., object and context. The keypoints from the object set are features inside the initial bounding box. Context keypoints are features detected around the initial bounding box. Alien uses SIFT extractor and descriptors to represent the model keypoints. The algorithm assumes that the object will appear within a radius r of previous detected object's bounding box, narrowing the area for keypoint extraction. The model representation is updated by adding keypoints to the object and context sets every time the object is detected and not occluded. Occlusion is detected when model context keypoints match keypoints inside the object's bounding box.Alien, uses single value decomposition to estimate position, scale and orientation of the matches.

All the trackers mentioned in the last paragraph achieve good results in the Visual Tracker Benchmark [158]. However, their computational cost is high which makes them less appealing than the correlation-filters based trackers. In addition, the correlation filter-based trackers CSK and KCF outperform all of the above mentioned trackers in the Visual Tracker Benchmark and VOT challenges while being faster.

CSK explores the structure of the circulant patch, which employs kernel correlation filter to achieve high performance. The detection phase implementation is a template matching approach in the frequency domain. At each frame, the algorithm looks for the area with the maximum correlation value to the updated model. The model is updated by linear interpolating the detected areas from the previous frames. To update the model,

a weighted sum of kernelized images is used in the frequency domain. Based on CSK, KCF adds multichannel features to the correlation filter pipeline. It improves accuracy and robustness by adopting HOG features instead of the raw pixel values used in CSK.

SAMF [73] is the closest work to our solution because of its goal. It adds scale to the KCF framework by sampling the original target with different scales and learning the model at each scale. Moreover, it combines HOG descriptor with colour-naming [145] technique to boost the overall performance. However, the inclusion of colour-naming to the HOG features and computing the kernelized correlation filter for all possible scale comes at a substantial computational cost.

Our proposed algorithm is based on the KCF algorithm. The main difference between our proposed solution and SAMF is that we learn the scale variations online instead of learning the model for all possible scales. Furthermore, instead of adding computation time, we are improving the high-speed performance of KCF while boosting its accuracy and robustness. The Gaussian window filtering allows us to adjust to scale changes while creating a better separation of the object and the background without affecting the KCF framework. Because of the separate Gaussian filtering, scale estimation can be added to any of the polynomial and linear correlation filters presented in Henriques et al. [52]. In addition, our C++ implementation incorporates fast HOG descriptors using SSE instructions and CCS packed format to improve the performance while decreasing the memory footprint. For scale estimation, we use a similar approach as in TLD and combine keypoint tracking with detection. We implement the forward-backward Kanade-Lucas-Tomasi [160] keypoint tracking strategy initialized at the position estimated by KCF. However, we select good features to track instead of grid-points. Also, we include a different measure to estimate the scale. While TLD uses a median value to estimate scale we use a weighted arithmetic mean where points near the center of the tracker contribute more than those near the boundaries.

Due to recent advances in object tracking there is an increasing necessity to evaluate and compare their performance under different challenging tracking scenarios. Among the most relevant benchmarks available to date we find the Visual Tracker Benchmark [158] and the VOT Challenges [63]. The Visual Tracker Benchmark consists of 50 test sequences, where each sequence is annotated according the tracking challenges presented in the video sequence (i.e., change of scale, occlusion, change of illumination, etc). The performance of each tracker is evaluated in terms of precision and success. Precision is envisioned as how close is the tracker output location from the ground truth and success

as the overlap area. More recent work in visual tracking benchmark identify a set of potential measures that can be interpreted and visualized for tracker comparison [25]. Luka et al. [25] propose a ranking benchmark based in accuracy and reliability. The VOT Challenges use these performance measures to generate a more detailed assessment of the tracker evaluation. Till date, there have been three VOT Challenges: 2013, 2014 and recently 2015. For each challenge a different annotated video dataset has been used. The first challenge (i.e., VOT 2013) introduced a dataset with 16 videos. The videos are annotated frame by frame according the different challenges (i.e., change in scale, occlusion, change of illumination, etc). The VOT 2014 extended the video dataset to 25 sequences and introduce improvement in annotations and testing of statistical significance. Finally, the VOT 2015 created two challenges for visual evaluation (VOT 2015 and VOT TIR 2015). The first dataset consist of 60 annotated video sequences and the second consist of the first thermal imagery tracking challenge (i.e., VOT TIR 2015) composed of 20 video sequences with annotations.

2.5 Summary

As we have seen in the review of related work, many directions have been explored in pruning a skeleton of noisy shapes in images. Many of these approaches have been successful, but nothing so far has been capable of computing a fast automated skeleton while preserving the main features of the shape and its topological properties. Many of the current solutions cause critical changes to the skeleton's curves, creating more noise points, shrinking the final skeleton, or disconnecting or removing important branches of the skeleton. We choose to pursue an approach of pruning the integer medial axis transform using contour approximation. Chapter 3 describes our solution, experimental evaluation and comparison with related work.

To address the sub-goal of localization using natural landmarks, we exploit the benefits of using planar object that could be present in the environment. Current solutions exploit the use of artificial landmarks and exteroceptive sensors to address the localization problem. These methods provide a robust and stable solution for controlled environments, but natural landmarks are more appropriate for unmodified environments and are generally of lower cost as they require only a single camera to acquire images. Artificial landmarks and sensors are previously selected and positioned in the environment for better accuracy, which is not appropriate for large spaces because of the deployment and extra cost. We choose planar targets because of the possibility to generate differ-

ent synthetic views of the target during the training phase, which in turn allows for an evaluation of the target during the training phase. We present a run time framework for localization using natural landmarks in Chapter 4.

We assess the accuracy of pose estimation and localization through synthetic and real experimental evaluations. During the review in Section 2.3, we found that gradient-based keypoint descriptors, although considered the gold standard, are computationally expensive. Conversely, binary descriptors are easy to implement, and achieve fast performance by sacrificing accuracy but lowering the memory requirements. Some classifiers such as Random Forrest and Ferns describe keypoints using randomized binary comparisons to boost accuracy and speed matching, but they compromise the memory usage. The off-line learning step of classifiers ensures the encoding of features from multiple changes in perspective and have shown improved results compared to SIFT descriptors. We identify the need for keypoint descriptors that can compare with classifiers such as Ferns while having a smaller memory usage. The 3D object constraint adds complexity to the learning and detection steps. We combine descriptors and classifiers in one single matching scheme in Chapter 5.

Finally, we explored related work from our last sub-goal in Section 2.4 and exposed the different techniques and their drawbacks in long-term tracking tracking of 3D objects. We discussed current approaches and how they react to different problems in single-object tracking. We introduced the benchmark dataset and the performance measures used to evaluate the current solutions and that our future work will follow. While reviewing the existing and related work, we found that correlation filters provides a good platform for visual long-term tracking tracking because of their performance. However, the correlation filter uses a fixed size template and hence the tracker is not able to handle scale changes of a target occurring during motion. As a result, we improved correlation filter by implementing a Scalable Kernel Correlation Filter with Sparse Feature Integration to remove the fixed size limitation while improving accuracy and speed. We introduce an adjustable Gaussian window function and a keypoint-based model for scale estimation to deal with fixed size limitation in the Kernelized Correlation Filter in Chapter 1.2.4. Furthermore, we integrate fast HOG descriptors and Intel's Complex Conjugate Symmetric packed format to boost the achievable frame rates.

Chapter 3

Skeleton Pruning by Contour Approximation and the Integer Medial Axis Transform

We present a new shape skeleton pruning algorithm based on contour approximation and the integer medial axis. The algorithm effectively removes unwanted branches, conserves the connectivity of the skeleton and respects the topological properties of the shape. The algorithm is robust to significant boundary noise and to rigid shape transformations. It is fast and easy to implement. High accuracy reconstruction of the shape is possible from the generated skeleton by means of the integer medial axis transform. Our algorithm also produces a vector representation of the skeleton. We compare our algorithm with state-of-the-art techniques for computing stable skeleton representations of shapes including pruning. We test and compare our solution using the MPEG-7 CE Shape-1 Part B dataset looking for skeleton connectivity, complexity, parameter selection, and accuracy/quality of the outcome. The experimental results show that our solution outperforms existing solutions according to these criteria.

3.1 Algorithm Overview

Our proposed algorithm has four major steps which we detail in the following: contour approximation, skeleton computation, pruning and skeleton vectorization.

Our solution shares the concept of Bai et al. [7] that every skeleton point is the center of a maximal circle whose boundary tangent points belong to different 'sides' of the shape. Skeleton points are pruned if they have all their closest boundary points lie on the same contour segment. Every point in the pruned skeleton remains linked to a boundary point that is tangential to its maximal circle [13] after pruning.

For the first step of our solution, we propose to use a boundary partition that does not require to know the number of vertices *a priori*. This is in contrast to the Discrete Curve Evolution (DCE) introduced by Bai et al. [7] that requires one to specify the number of vertices of the polygonal approximation beforehand. Intuitively, we want to separate our shape's boundary into different "sides" which we will use in the next step to prune skeleton points associated to only one "side".

The second step of our algorithm computes the integer medial axis transform [53] and starts to prune the skeleton using the contour approximation from the first step. As in [7], there is a skeleton branch at every partition point that needs to be removed after the second step of our algorithm (see Figure 3.5). We introduce a novel significance measure based on Blum's medial axis definition of maximal circles, where instead of analyzing isolated skeleton points we check the leaf branches for removal. Leaf branches of the skeleton end at a boundary pixel. This second pruning step handles the elimination of branches that while they have boundary points on different contour approximations because of the knot selection, they are still on the same 'side' of the shape. In contrast, Bai et al. use a significance measure calculated with their DCE. Finally, we create a vector representation of the skeleton points using the contour approximation from the first step of the algorithm.

3.1.1 Contour Approximation

For the algorithm's first step, we approximate the binary shape by a piecewise curve. Piecewise curve approximations have been used in the computer graphics industry for a long time. Many methods trying to approximate a set of points by a piecewise curve have been published, e.g., [111, 54, 133]. The most challenging problem for these methods is the knot placement to create an optimal approximation. Some of these algorithms may select the knot placement automatically as the methods of Denison et al. [33] and of

Plass and Stone [111]. Because we are interested in a real-time algorithm, we need to take into consideration the trade-off between optimal knot placement versus time/space complexity of the contour approximation method. Therefore, we select a simple split-and-merge fitting strategy for knot placement following the algorithm by Ramer [114] as the curve approximation step. Ramer's algorithm splits the approximating curve at points of maximal error which tend to be extremal points. While choosing extremal points may result in suboptimal approximations, it often helps us to make good pruning decisions in later steps of our algorithm. Figure 3.1 shows an example where the additional knot required by Ramer's algorithm over optimal node placement produces a better skeleton after pruning.

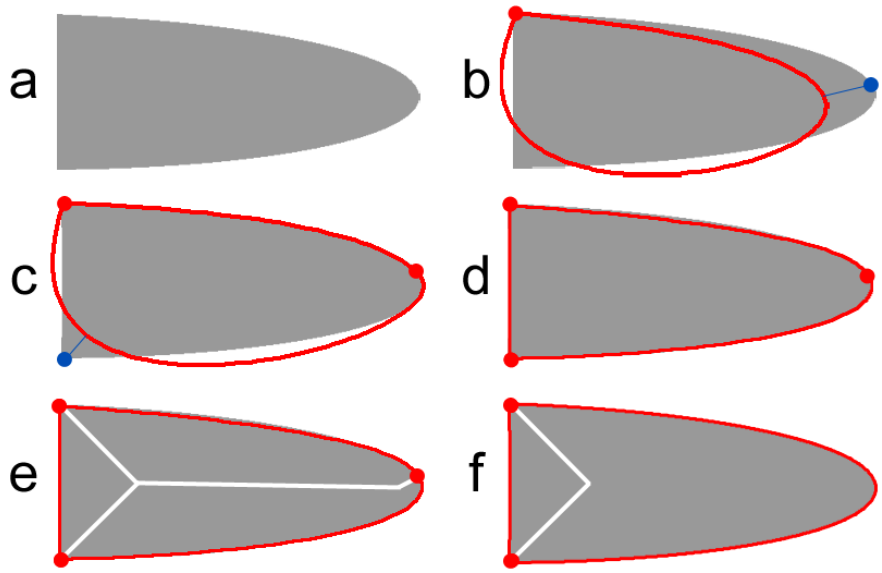


Figure 3.1: Example of significant branch removal because of optimal knot placement. Original shape (a) along with the contour approximation steps in (b) to (d) with thresholds $T = 10$ and $S = 1.2$. The suboptimal knots result in a better contour segmentation for final pruning as shown in (e) than the optimal knot placement in (f).

The approximation uses an input threshold T as the maximum distance error allowed between the fitted data and the approximated curve. The piecewise approximation uses the contours of the binary shape to generate a simpler and accurate representation. The fitting strategy can work with any curve but we demonstrate the use of line segments as the simplest curve approximation and cubic Bezier curves with least square fitting. Both, line segments and cubic Bezier curves lead to similar outcomes (see Figure 3.2),

although cubic Bezier curves require less knots for the same approximation error. The values of threshold T are connected to the image resolution because it is represented in pixel. The threshold value T determines how closely the approximation fits the original outline. Low values of T remove only higher frequency 'noise' from the skeleton without affecting significant parts of the shape while higher values will increase the number of branches removed but without creating a discontinuous skeleton. Our pruning does not shift the skeleton away from the location of the medial axis.

Given a shape contour with boundary pixels $C = \{p_1, p_2, \ldots, p_n\}$, we want to find a subset of those pixels and use them as endpoints of curve sections (e.g., lines or cubics) that approximate the boundary. The shape boundary may be formed by several contours, then the same fitting approach is applied for each of the contours. We select an input threshold error T that specifies the maximum allowed distance between the original points and the fitted curve. Initially, we take the start and endpoint of the contour as our initial knots and approximate them by a curve, e.g., a line segment with endpoints p_1 and p_n. Then we measure the minimum distance d_k of the center of each pixel to the fitted curve and look for the point p_k that is farthest from the curve with maximum distance d_k. If the distance from p_k to the curve exceeds the T value, we split the segment at p_k into two new segments p_1, p_k and p_k, p_n fit them using the same curve type (see Figure 3.3). This process is repeated for each segment until the T constraint is no longer violated. This produces an ordered set of knots that approximates the shape with a limit error T. The T threshold represents the level-of-detail of the approximation. We empirically selected a threshold of $T = 25$ in our experiments. The bigger the error threshold T, the less detail of the shape is taken into consideration and more likely to compute less knots. The smaller T the closer will be the piecewise approximation to the original data (see Figure 3.2). A value of $T < 0.5$ will make the piecewise curve approximation pixel accurate (i.e., knots will be a subset of the original pixel list). Consider a line segment aligned with the pixel grid: only two knots are needed.

For a cubic parametric curve approximation using Bezier curves we set the initial and last position at two consecutive knot points. Then, using the least square fitting technique we can approximate the best middle control points using the original points between the consecutive knots. This cubic approximation does not consider a continuity constraint on the first derivative of two consecutive segments but reduces the number of required segments. The knot set is typically smaller with cubics than using linear segments for the same T. Cubics have the advantage of fitting a larger amount of points with a single

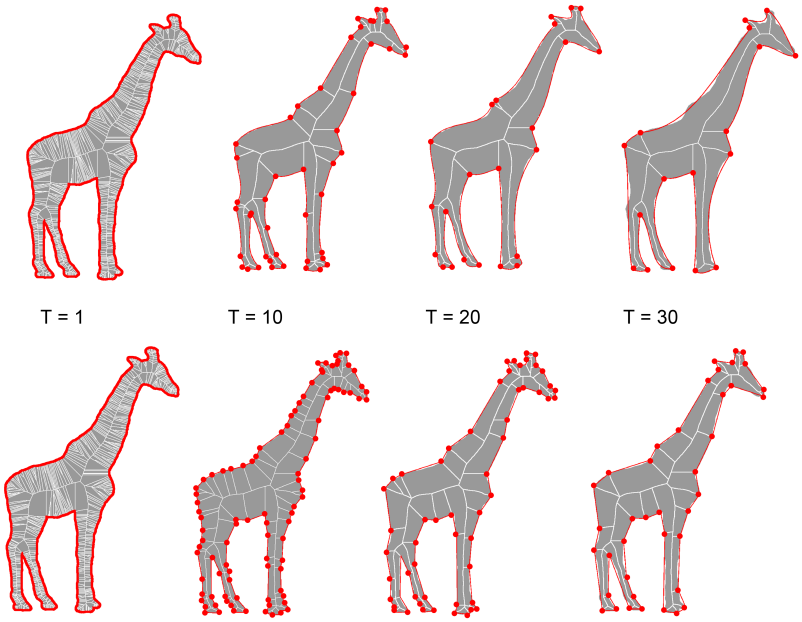


Figure 3.2: The effect of different selection of T while pruning the IMA without the final pruning (i.e., the value s is selected as 0). First row shows the results using cubic boundary approximation, second row results are generated with linear approximation.

curve section but more computation is required than with the linear approximation.

The least-squares Bezier fitting solves

$$S = \sum_{i=1}^{n} [p_i - q(t_i)]^2 \tag{3.1}$$

with n data points p_i and the parametric Bezier curve $q(t_i)$. We can write Equation 3.1 as

$$S = \sum_{i=1}^{n} \left(\begin{array}{r} p_i - (1-t_i)^3 P_0 + 3t_i(1-t_i)^2 P_1 \\ + 3t_i^2(1-t_i)P_2 + t_i^3 P_3 \end{array} \right)^2$$

where P_0 and P_3 are the first and last control points; the remaining control points P_1

and P_2 can be determined by setting $\frac{\delta S}{\delta P_1} = 0$ and $\frac{\delta S}{\delta P_2} = 0$. Solving for P_1 and P_2 gives

$$
\begin{aligned}
P_1 &= (A_2C_1 - A_1C_2)/(A_1A_2 - A_{12}A_{12}), \text{ and} \\
P_2 &= (A_1C_2 - A_{12}C_1)/(A_1A_2 - A_{12}A_{12}) \text{ where} \\
A_1 &= 9\sum_{i=1}^{n} t_i^2(1-t_i)^4, \quad A_2 = 9\sum_{i=1}^{n} t_i^4(1-t_i)^2, \\
A_{12} &= 9\sum_{i=1}^{n} t_i^3(1-t_i)^3, \text{ and} \\
C_1 &= \sum_{i=1}^{n} 3t_i(1-t_i)^2\left[p_i - (1-t_i)^3P_0 - t_i^3P_3\right], \\
C_2 &= \sum_{i=1}^{n} 3t_i^2(1-t_i)\left[p_i - (1-t_i)^3P_0 - t_i^3P_3\right].
\end{aligned}
$$

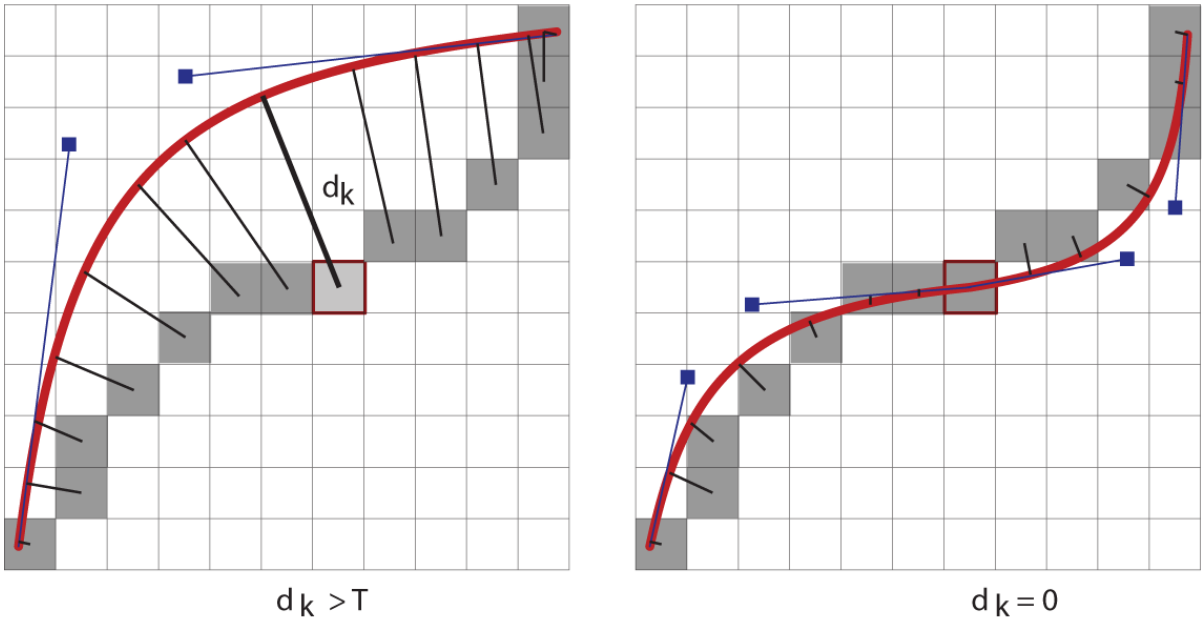


Figure 3.3: Piecewise contour approximation example. When we approximate the contour formed by n boundary points, we split the approximating curve (here, a cubic Bezier curve) if the maximum distance from the center of the pixel to the approximating curve is bigger than T threshold selection.

After the set of knots is computed, we assign the contour points between two consecutive knots $\{p_k \ldots, p_{k+1}\}$ to the same curve. We also assign an unique identifier for each

curve segment. All contours of the shape; exterior and interior (i.e., holes) are processed in the same manner.

3.1.2 Modified Integer Medial Axis

For the second step, we compute the skeleton by means of the Integer Medial Axis (IMA) without any of the pruning considered by Hesselink et al. [53]. In finding the skeleton we select pixels as skeleton points if they have two equidistant points belonging to two different boundary curves instead of simply two different equidistant points on the boundary as in the original IMA. Our goal is to select skeleton pixels that have equidistant boundary points on two "different" sides of the shape. This criterion will considerably reduce the number of skeleton points detected by the IMA but still produce a connected skeleton (see Section 3.1.5). Unfortunately, this criterion does not remove all the undesired skeleton points but leaves whole branches that should be removed in place. Branches that end at the intersection of two consecutive curves (see Figure 3.4) will be part of the skeleton after this modified IMA. Thus, we need to take extra care of these branches in the pruning step of our algorithm.

Listing 1: Program fragment for filtering the IMA skeleton using our pruning condition. The map C returns the curve associated to a boundary point and map ft2 returns the associated closest boundary point

```
procedure_compare(x,y)
xf:=ft2[x]; yf:=ft2[y];
C1:= C[xf]; C2:= C[yf];
if ||xf - yf||^2 > 1 and  C1 != C2 then
  crit:= inprod(xf-yf, xf+yf-x-y)
  if crit > 0 then skel[x]:=SKEL
  endif
  if crit < 0 then skel[y]:=SKEL
  endif
endif
```

3.1.3 Final Pruning

For the third step, we need to compute all of the end points and branch points of the skeleton output of the previous IMA step. Assuming a skeleton is one pixel wide [53] and always located at a pixel, then those pixels on the skeleton with more than two neighbouring pixels are branch points, while end points have only one neighbouring

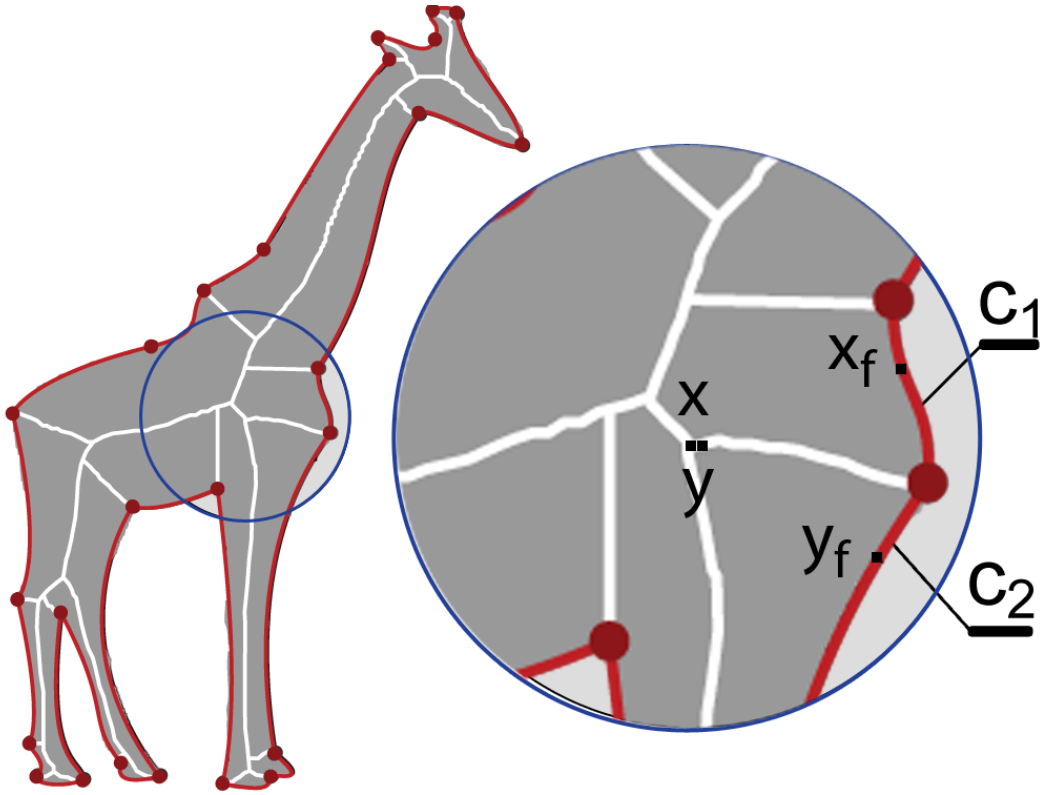


Figure 3.4: A Skeleton point X is selected if it has a neighbour point Y with a closest boundary point that belongs to a different curves(C_1 vs. C_2) at the same distance d, see Listing 1.

skeleton pixel. During branch removal, we consider all end points of a branch and their respective branch point as belonging to the same branch. Our pruning criterion is as follows: consider a branch $\{b_p, \ldots, e_p\}$ with branch point b_p and end point e_p such that the pixel chain formed by the skeleton points $\{b_p, \ldots, e_p\}$ does not contain other branch points or end points except b_p and e_p. Let f be the closest boundary point of b_p as extracted from the feature transform computed as a preprocessing step of the IMA. We mark a branch $\{b_p, \ldots, e_p\}$ for removal if the point e_p is inside of the circle with center b_p and a radius equal to the Euclidean distance $|f - b_p|_2$. We can make this pruning rule stronger if we introduce an input scale factor s for the radius and mark branches for removal if they are inside the scaled circle centered at the branch point (see Equation 3.2 and Figure 3.5), i.e.,

$$|e_p - b_p|_2 \leq s * |f - b_p|_2. \tag{3.2}$$

After analyzing all branches for removal, we remove those pixels that belong to the

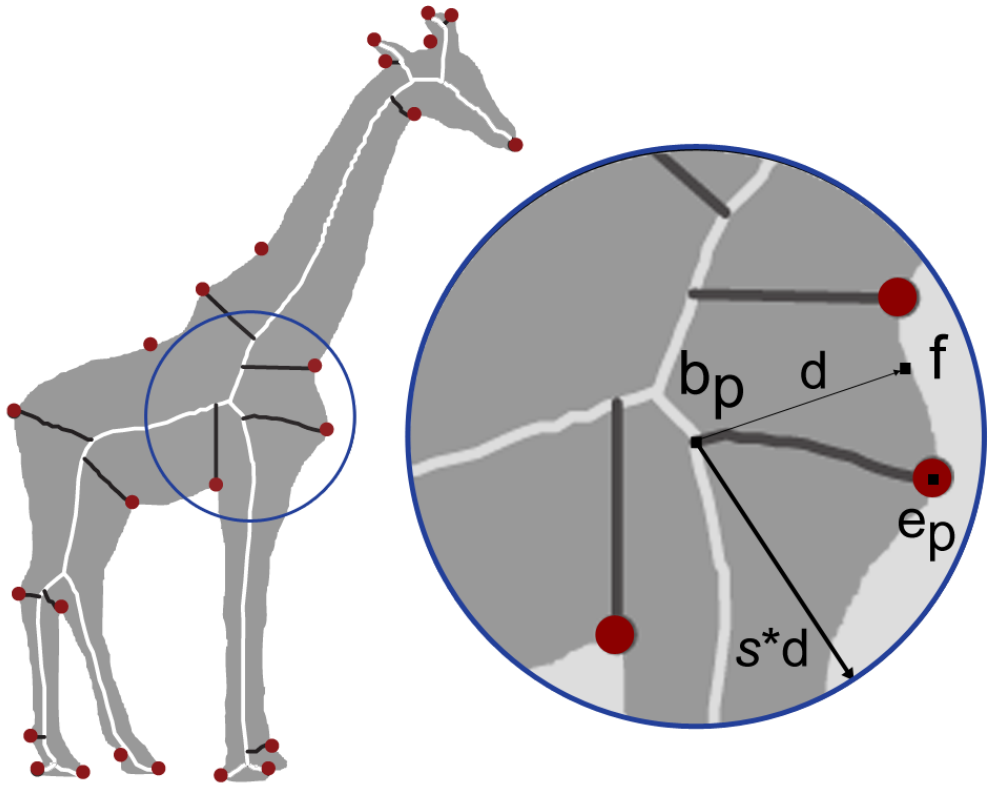


Figure 3.5: The unwanted branches (shown dark) introduced in the second step of the algorithm are pruned using Equation 3.2. The dark branch to endpoint e_p is removed by the test at branchpoint b_p.

marked branches of the skeleton, except for the branch points. We re-iterate this process until no more branch removals are possible. Note that after removing the pixel chain of a branch, we can update the status of the branch points and reduce the computation of searching over the full skeleton set for branch and end points. The final output of the algorithm is a clean skeleton representation (see Figure 3.6).

The scale factor s targets the branches that end at the approximation knots created while pruning the IMA. The selection of s will affect the final outcome of the algorithm, where larger values will generally remove more unwanted branches (See Figure 3.7).

3.1.4 Skeleton Vectorization

In the final step, we vectorize the skeleton points. Each branch $\{b_p, \ldots, e_p\}$ or $\{b_p, \ldots, b_p\}$ is processed separately and approximated with a piecewise cubic Bezier curve as explained in Section 3.1.1. We apply an automatic adaptive threshold using the feature transform

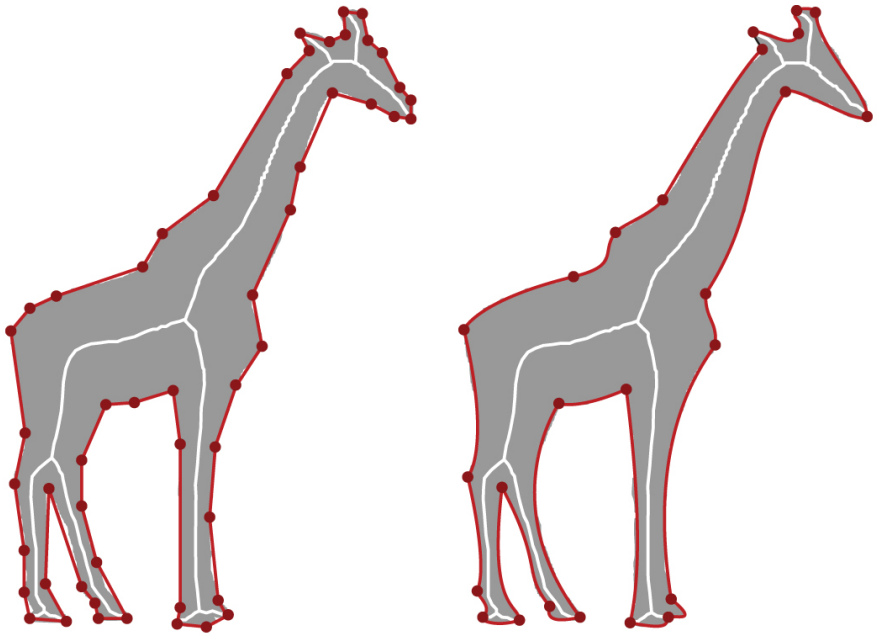


Figure 3.6: Final results of the proposed algorithm using linear and cubic approximation contours.

stored with the IMA from Section 3.1.2 during the skeleton curve approximation. While processing a branch, we need to check the distance e from every skeleton point x to the approximated curve. If the Euclidean distance e_{max} between the pixel with maximum distance to the approximated curve is bigger than the distance between the point x and its corresponding boundary point x_f stored in the feature transform or if it is bigger than threshold T from the boundary approximation of Section 3.1.1, we split at x (see Figure 3.8). Figure 3.9 shows the binary shape input of our algorithm and the final vector representation of the skeleton.

3.1.5 Correctness and Complexity

For showing the correctness of our algorithm, we first note that our skeleton output is a subset of the IMA skeleton pixels. This proves that every selected pixel is in fact on the medial axis of the discrete shape [7]. Additionally, we need to show that the pruning solution does not disconnect the final skeleton. Let us assume to the contrary that there is a branch that will become disconnected by pruning points $\{a, \ldots, b\}$ from the skeleton. Let the two boundary curves C_{a1} and C_{a2} be linked to skeleton point a, and the curves C_{b1} and C_{b2} be linked to skeleton point b (see Figure 3.10). Points $\{a, \ldots, b\}$ will be pruned if and only if the curve connecting C_{a1} and C_{b1} is the same that connects C_{a2}

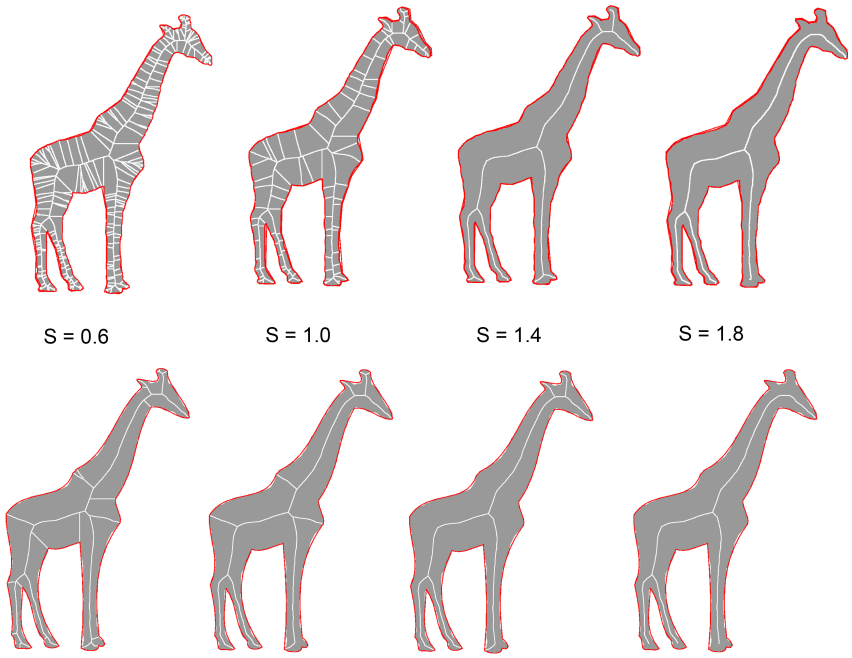


Figure 3.7: First row shows the effect of selection of s values for a cubic approximation with fixed value $T = 1$, and second row for a cubic approximation with fixed value of $T = 25$.

and C_{b2}. This requires that curve C_{a1} and C_{a2} or C_{b1} and C_{b2} be the same. If this is true, then the chains ending at a and b would also be pruned correspondingly, because they would have only one contour segment linked to them. But this contradicts our assumption of a disconnected skeleton. The connectivity of the skeleton is never broken in the final pruning step because only complete pixel chains between an end point and a branch point are removed. We note, however, that our algorithm may remove complete branches that can be considered significant depending on the boundary approximation.

The boundary approximation step of the algorithm has time complexity of $O(kn+N)$, where k is the number of knots detected during the contour approximation, n is the contour length and N the image size. Computing the contour of the shape is $O(N)$ [47] and the contour approximation has $O(kn)$ with a worst case of $O(n^2)$ for $k = n$ when $T = 0$ and all contour points are selected as knots. Assigning the contour pixels to a curve is $O(n)$. The IMA is computed in $O(N)$ [53]. The third step is $O(n)$, because each branch $\{b_p, \ldots, e_p\}$ is only checked once. The vectorization of the skeleton is $0(kn)$. Finally, the total time complexity is therefore $O(kn + N)$. A space complexity of $O(N)$

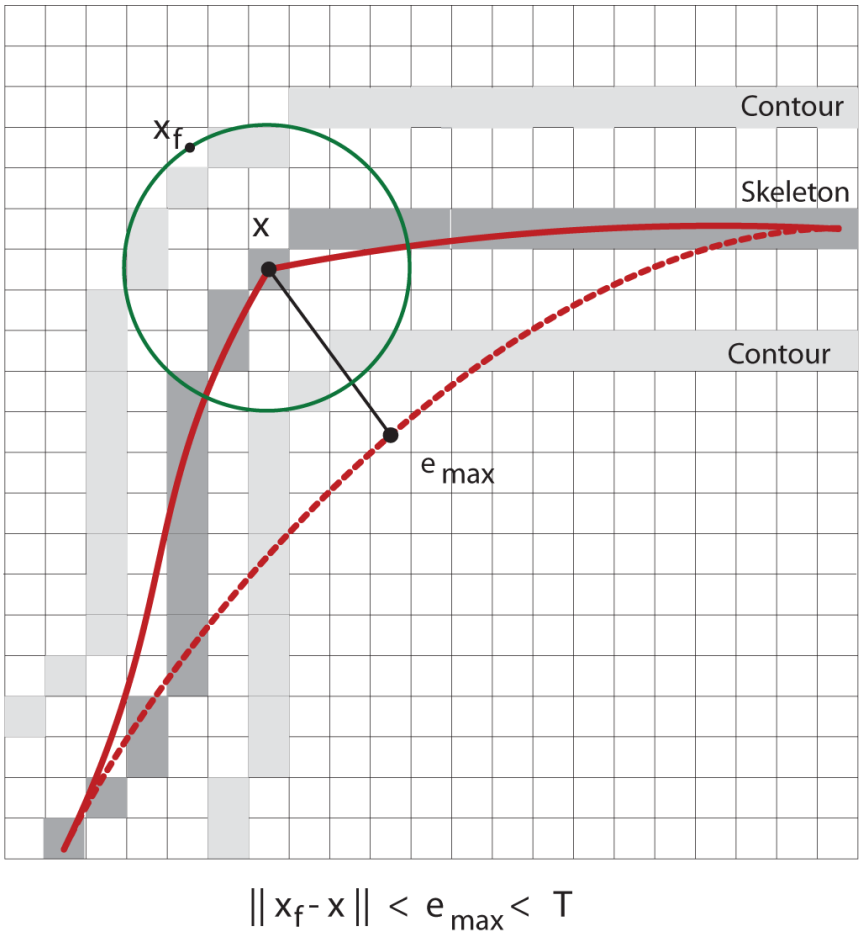


Figure 3.8: Skeleton vectorization using the feature transform to automatically adapt the threshold of the curve approximation. The point x has the maximum distance e_{max} to the approximated curve at the point x_f. We split the curve at x if e_{max} is bigger than $||x_f - x||$ or if it is bigger than T from Section 3.1.1.

can be easily achieved.

3.2 Experimental Results

We implemented three skeletonization algorithms for the comparison of our algorithms with the three categories: a thinning approach [66], a Voronoi-based algorithm [47], and a feature transform [53]. Also, five pruning techniques were implemented. Three pruning techniques as they were suggested by Hesselink and Roerdink [53]: constant pruning (DP), linear pruning (LP), and square-root pruning (SRP). A discrete variation of the scale-axis-transform [45, 81] and DCE pruning [7] were also implemented. The skeletonization and pruning solutions were tested over the 1400 image MPEG-7 CE Shape-1

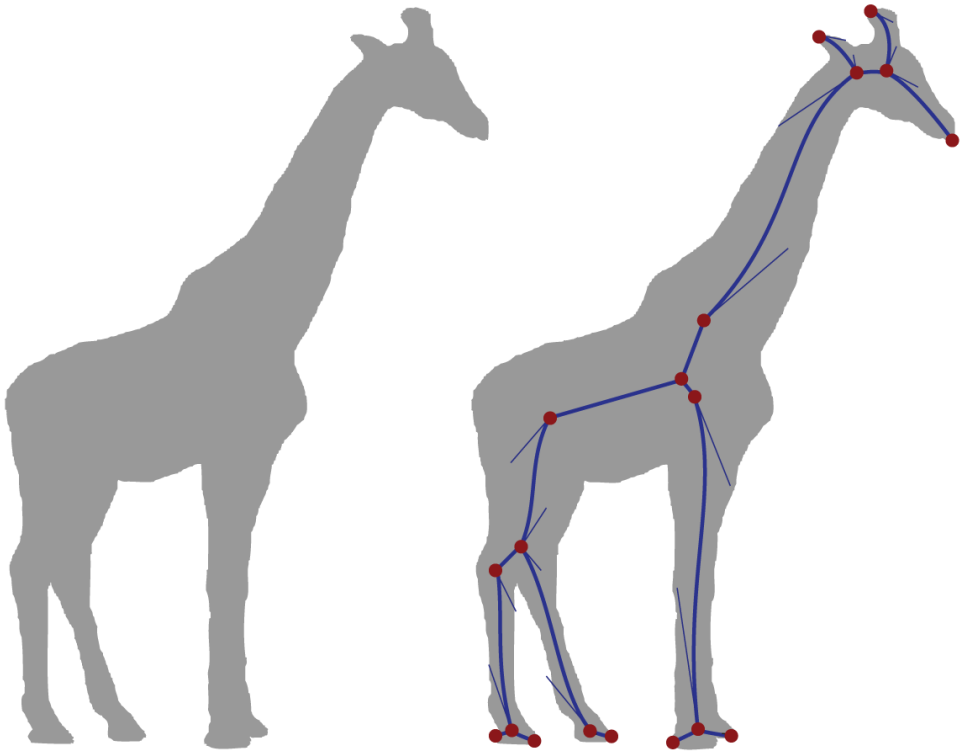


Figure 3.9: Input: 2D binary image. Output: Vector representation of the skeleton. Only 19 end points plus 19 control points are necessary to represent 4352 skeleton points using a piecewise cubic approximation.

Part B dataset, a dataset commonly used for shape analysis [69, 2] (e.g., shape matching, shape retrieval, shape skeletonization). The dataset is structured as twenty shape samples of seventy different classes (e.g., bats, planes, beetles, etc.). We account in our tests for running time, accuracy of the skeleton, rotation invariance, shape reconstruction from the skeleton, connectivity of the skeleton, and ease of parameter selection. The image resolution of the complete dataset varies from $[256 \times 256]$ to $[950 \times 800]$.

Pruning is a necessary step for the Voronoi-based and IMA algorithms. The thinning approach is more stable and it creates cleaner representations without input parameters or any pruning, i.e., it is fully automatic. However, the skeleton computed from the thinning solution is only an approximation of the medial axis and not the correct medial axis as in the other solutions. Also, the thinning approach is sensitive to noisy boundaries and shape rotations, introducing several unwanted branches for the same shape.

The selection of a correct pruning threshold for distance and angular pruning that is needed for the Voronoi-based algorithm and the IMA is problematic as different threshold

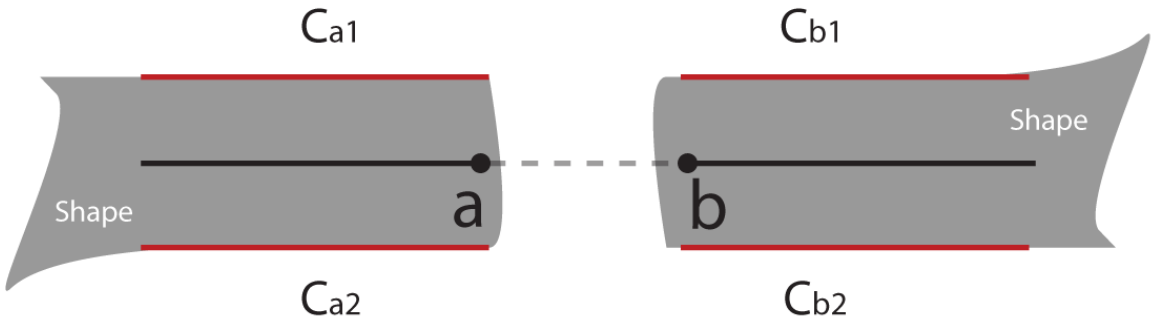


Figure 3.10: Pruning the IMA based on contour approximation does not disconnect a skeleton branch. For the skeleton points $\{a, \ldots, b\}$ to be pruned, the boundary contour between C_{a1} and C_{b1} and between C_{a2} and C_{b2} would have to be approximated with the same curve, which is impossible.

values disconnect the shape, e.g., when the leg width of the giraffe in Figure 2.1b is less than the selected threshold. In the case of angular pruning with the IMA, it introduced 'noise' in areas closer to the boundaries (see Figure 2.2). Also, the parameters for the pruning need to change for different shapes, in contrast to our solution where the selection of input parameters is quite stable and we used the same input parameters ($T = 25, s = 1.2$) for the complete dataset. We find the results of our approach also more visually appealing than with the other approaches, including the automatic thinning solution (see Figure 3.20).

To compare the running time of the implemented techniques, we recorded the computation times of each algorithm. Voronoi-based and thinning solutions were quite slow compared to the other solutions (see Table 1). The IMA is the fastest solution (i.e., IMA + {CP,LR,SRP} are grouped into one category because of their similar results in speed and accuracy), followed very closely by our proposed algorithm using linear and cubic contour approximations.

Shape reconstruction was not possible from the respective outputs of the thinning algorithm and the scale-axis-transform. Based on the skeleton calculated with our proposed algorithm, reconstruction of the original shape is possible with only small error for the complete dataset using the same input parameters ($T = 25, S = 1.2$). For example,

Table 1: Running time comparison.

Algorithm	Running Time (95 % Confidence Interval)
IMA + {CP,LP,SRP}	0.08 ± 0.01 sec
Linear	**0.09 ± 0.004 sec**
Cubic	**0.10 ± 0.009 sec**
Scale Axis Transform	0.17 ± 0.013 sec
Thinning	0.35 ± 0.018 sec
Voronoi	0.68 ± 0.021 sec
DCE	5.13 ± 0.518 sec

Table 2: Invariance to image rotations. The matching score gives the percentage of overlapping skeleton pixels detected in both the original image and in the rotated version.

Algorithm	Matching Score (95 % Confidence Interval)
Cubic	**0.95 ± 0.002**
Linear	**0.95 ± 0.004**
IMA + {CP,LP,SRP}	0.93 ± 0.005
Scale Axis Transform	0.90 ± 0.003
DCE	0.78 ± 0.003
Voronoi	0.71 ± 0.009
Thinning	0.49 ± 0.01

the reconstructed shape from the skeleton in Figure 3.12 showed 98% overlap with the original shape. The shape can also be fairly accurately reconstructed from the skeleton calculated with the Voronoi-based algorithm, with the IMA and with the DCE, but these three algorithms needed different input parameters for different shapes.

In general, all the algorithms were sensitive to noisy boundaries, but our solution outperformed the other implemented algorithms on the same image set (see Figure 3.13). To complete the comparison, we measured how stable the implementations were under image rotations. For each image in the dataset, we first computed their skeleton, then we rotated the original images in steps of 35, 45, 70, 85, 110, and 135, degrees using nearest-neighbour, bilinear, and bicubic interpolation. Each image was rotated around its center. We computed the skeleton of the rotated image using the same parameters as for the skeleton before rotation and compared the two skeletons. We calculated the percentage of matching pixels between the two skeletons after also rotating the pixels of the skeleton

Table 3: Invariance to image rotations. Inserted/Removed pixel score is the percentage of pixels introduced/removed in the skeletonization by the rotation.

Algorithm	Inserted Pixels (95 % Confidence Interval)	Removed Pixels (95 % Confidence Interval)
IMA + {CP,LP,SRP}	0.013 ± 0.008	0.057 ± 0.007
Cubic	**0.015 ± 0.002**	**0.035 ± 0.002**
Linear	**0.02 ± 0.0003**	**0.030 ± 0.003**
Scale Axis Transform	0.02 ± 0.00005	0.08 ± 0.00004
DCE	0.02 ± 0.0006	0.08 ± 0.0002
Voronoi	0.09 ± 0.0012	0.10 ± 0.0015
Thinning	0.19 ± 0.0006	0.32 ± 0.02

of the original image. Besides the percentage of matching pixels, we calculated how many pixels were inserted to the medial axis because of the image rotation and how many disappeared from the original skeleton representation (see Figure 3.11).

The best matching percentage was obtained by our proposed algorithm using linear and cubic boundary approximations with 95% overlap, followed by the IMA, the Scale-Axis-Transform, DCE and the thinning approach (see Table 2). The interpolation method had little influence on the results and the scores were quite similar for nearest-neighbour, bilinear and bicubic interpolation (see also Figures 3.15 to 3.18).

Finally, we compared the stability of the skeleton of our proposed algorithm against DCE when input parameters are varied. The parameter choice for DCE is specific to each shape (see Figure 3.14). Our proposed algorithm allows us to use the same parameters for the complete dataset, yielding excellent results for each of the 70 classes available including for noisy shapes (see Figure 3.19). The results obtained by DCE needs parameter tuning in search of an "optimal" skeleton representation. In general, the outcome of DCE with user supervision for each sample is similar to our results, but parameter selection has a big influence on the DCE outcome. It is usually not possible to find a choice that is optimal for all the samples in one class see, e.g., the beetle shape in the last column of Figure 3.14 where one of the beetle legs has been missed. Parameter tuning for the DCE is also a time consuming process.

3.3 Summary and Conclusion

We presented a new pruning algorithm that removes excessive branches from skeletons of noisy 2D shapes. The algorithm uses a simple contour approximation and a novel branch removal technique. The combination of our two pruning steps has been shown to be more robust and flexible than using a DCE-based algorithm. Because our algorithm is robust to noise and rigid transformations and it produces consistent high-quality results with the same input parameters, it can be used in semi-automatic or even automatic applications. The algorithm is fast and straightforward to implement, which makes the algorithm suitable for a range of applications. Because our algorithm produces a connected skeleton that is a subset of the integer medial axis of the noisy shape, the shape can be reconstructed from the skeleton using the feature transform.

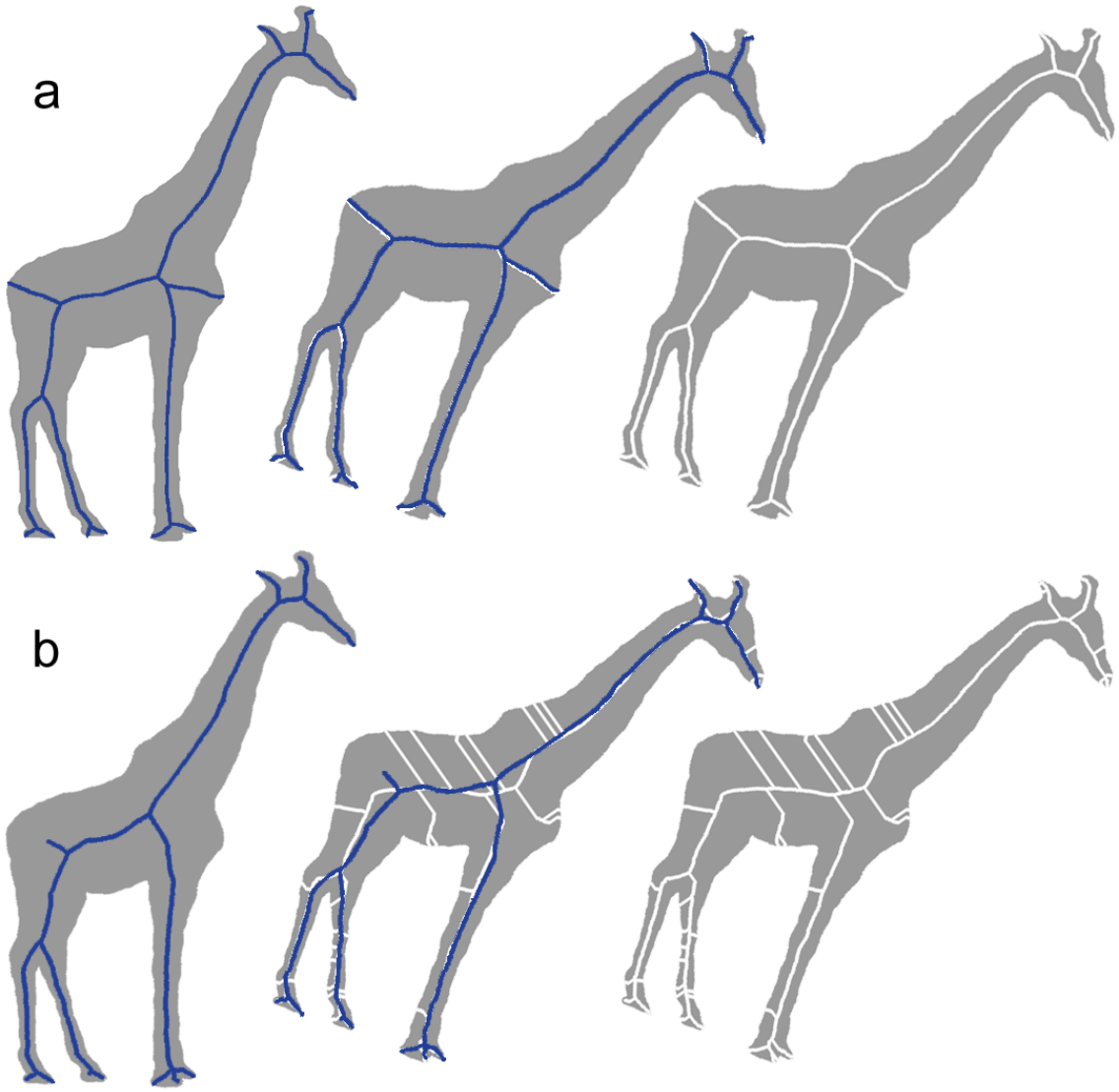


Figure 3.11: Example of the performance of DCE (a) and thinning (b) under image rotation. The blue skeleton is computed in the original image, then, the original image is rotated around the image center (using nearest neighbour and bilinear interpolation) and the white skeleton is computed. The blue skeleton is rotated using the same interpolation method and superimposed on the white skeleton to check for stability. The input parameters of the methods remained unchanged.

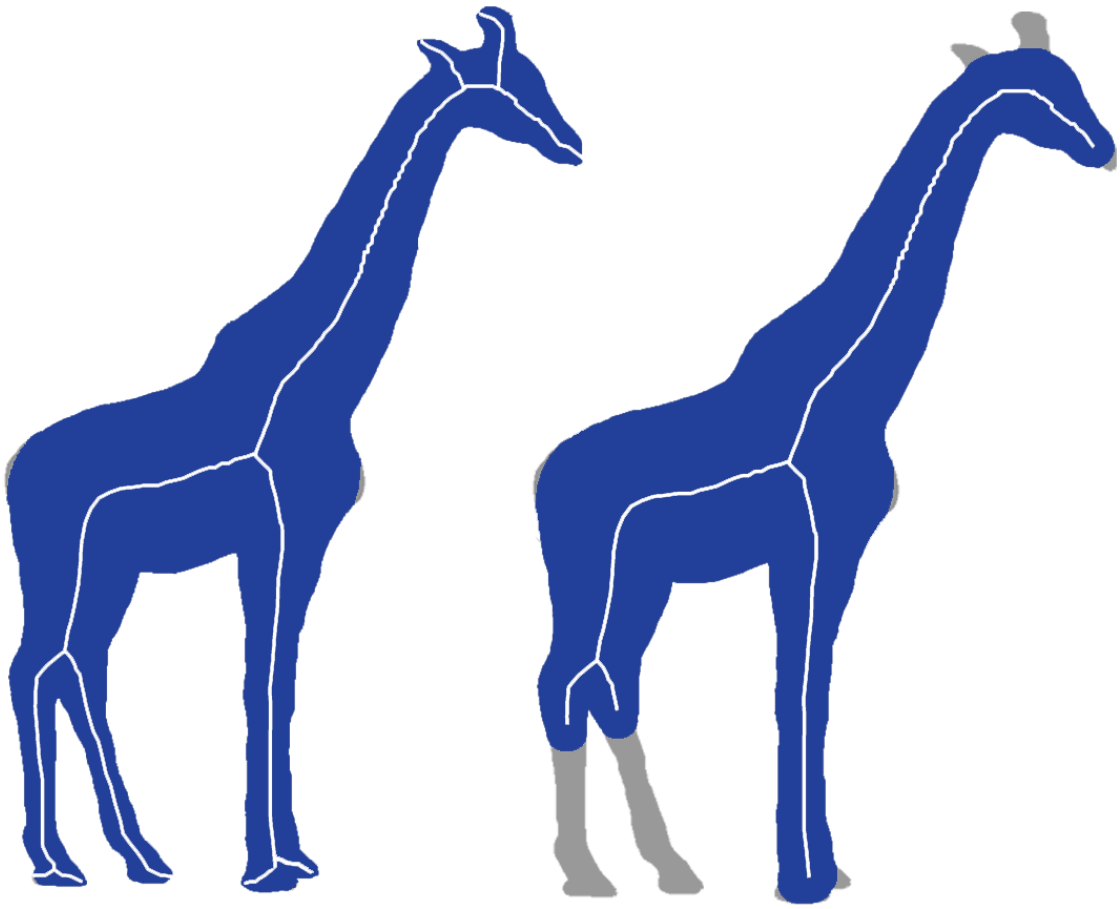


Figure 3.12: Example of shape reconstruction using the proposed solution (left) and IMA + constant pruning (right).

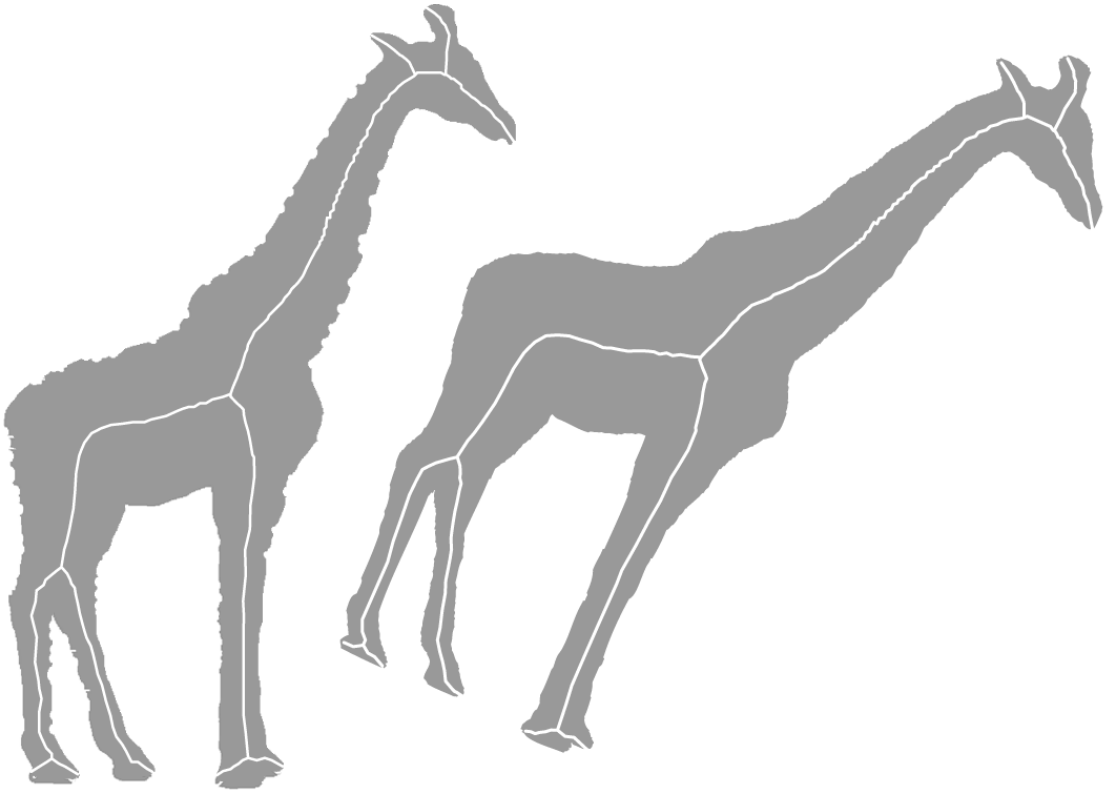


Figure 3.13: Output of the proposed algorithm with the same noisy and rotated example. Note the stability of the skeleton detection.

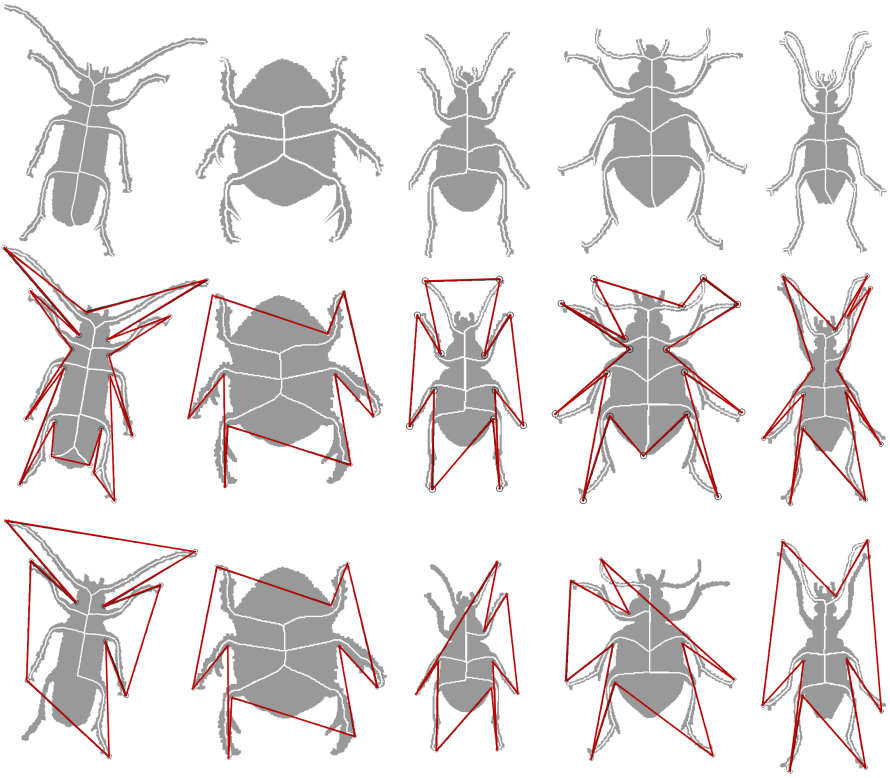


Figure 3.14: Parameter choice comparison between our solution and DCE, output examples are from the beetle class of the MPEG-7 dataset. First row is the output of our algorithm using the same configuration parameters ($T = 25, S = 1.2$). The number of cubic segments detected in the first step of our algorithm is 40, 21, 19, 32, and 27 respectively. Second row is the DCE using "optimal" user selected input parameters (i.e., number of sides to approximate the original shape, from left to right: 20, 10, 15, 17, 14). Third row is the DCE using constant number of sides equals to 10.

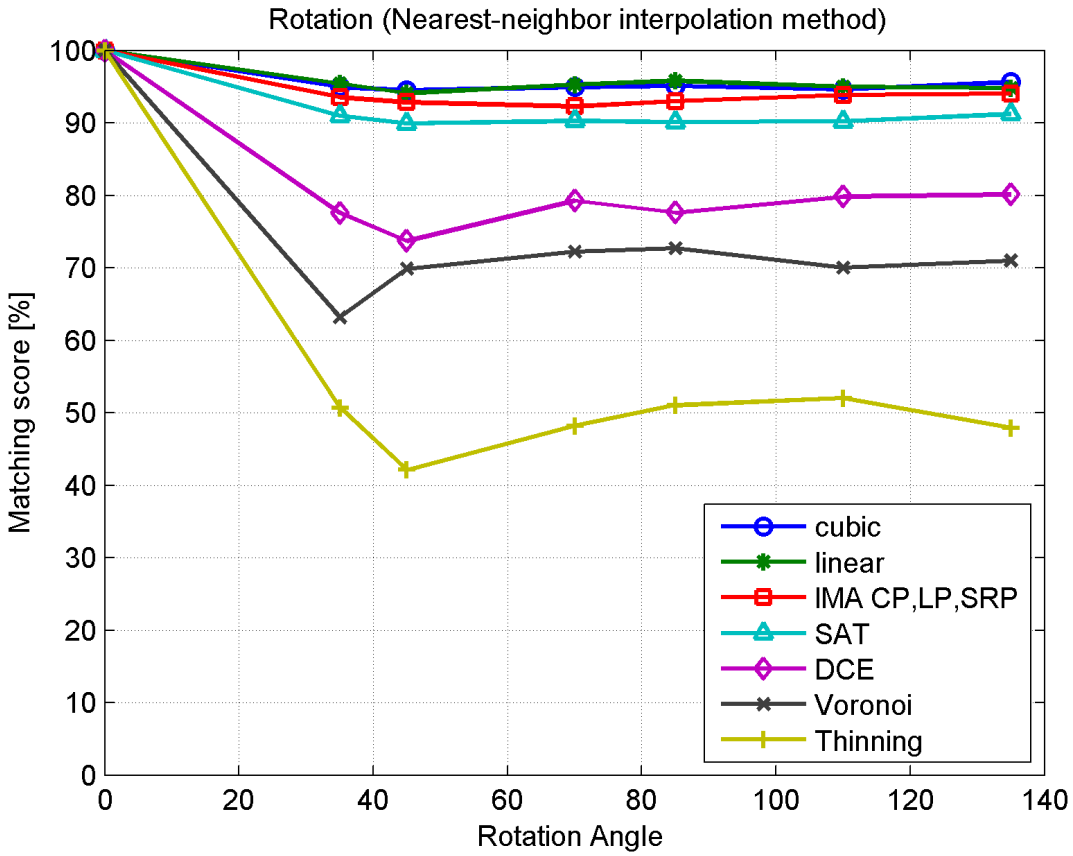


Figure 3.15: Image rotation test for angles of 35, 45, 70, 85, 110, and 135 degrees using nearest-neighbour interpolation.

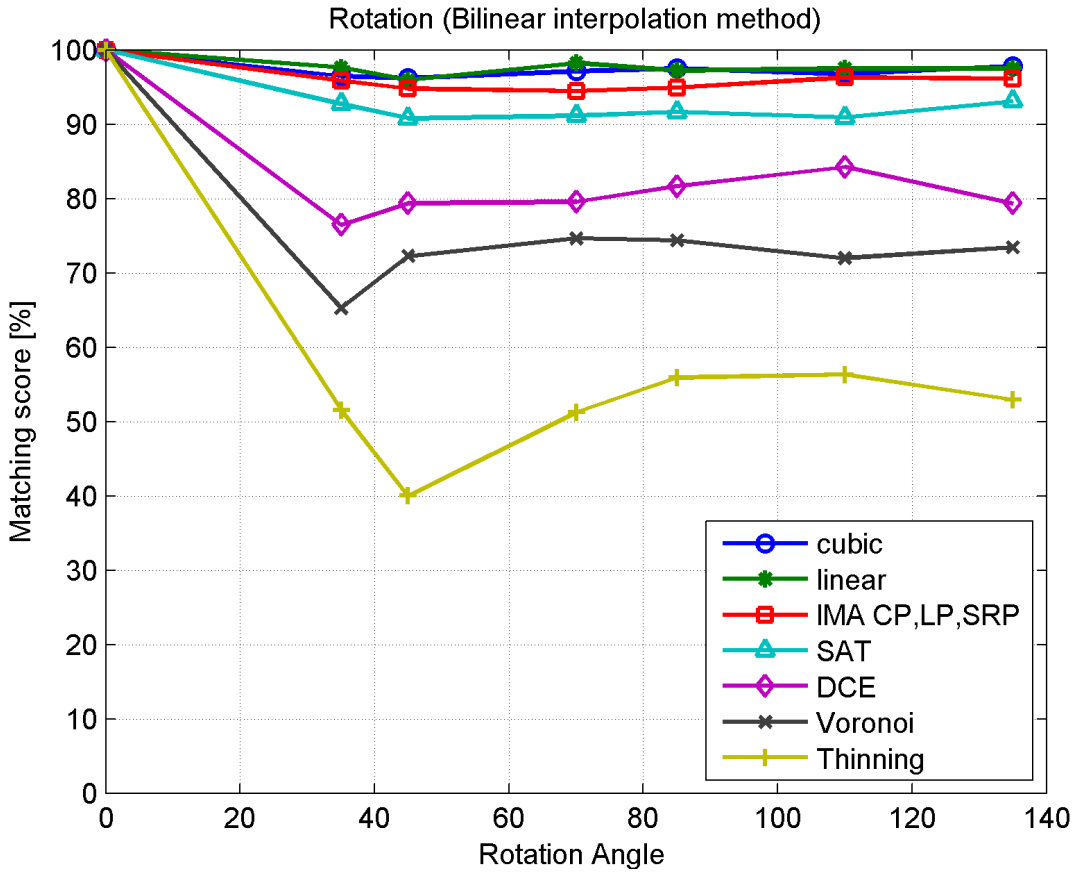


Figure 3.16: Image rotation test for angles of 35, 45, 70, 85, 110, and 135 degrees using bilinear interpolation.

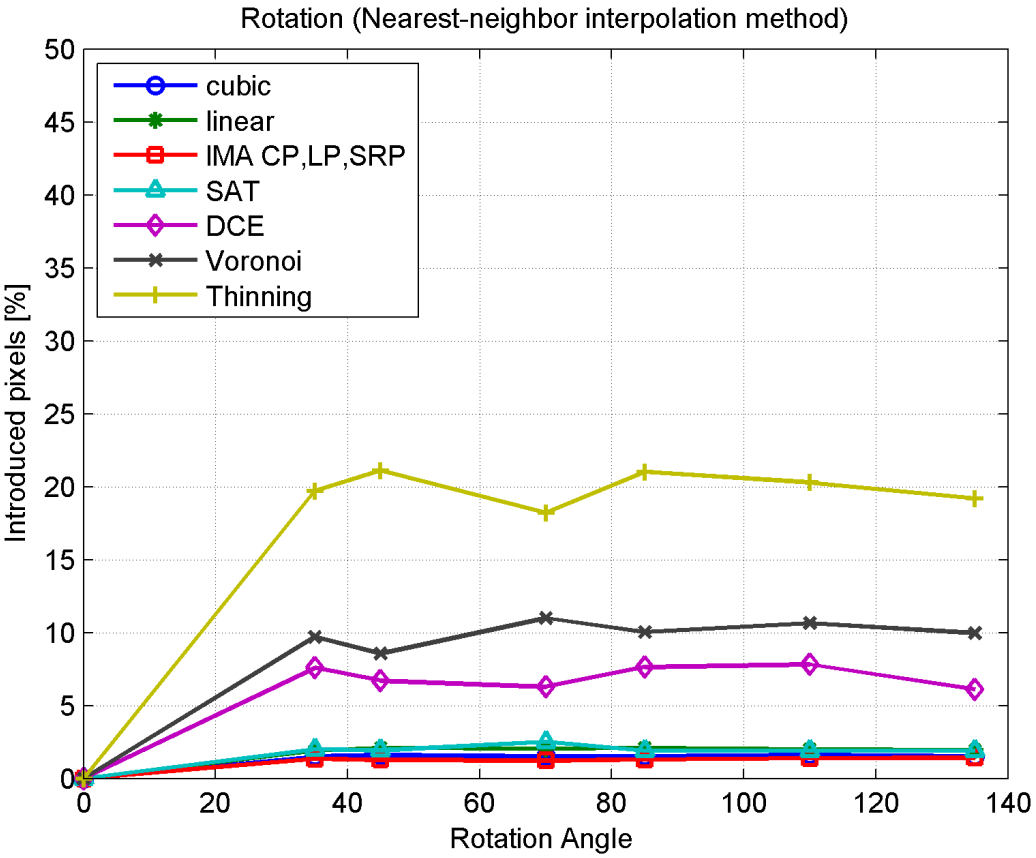


Figure 3.17: Percentage of skeleton pixels introduced after the skeletonization of the rotated image (i.e., skeleton pixels that were not detected in the original image but appeared in the rotated version.

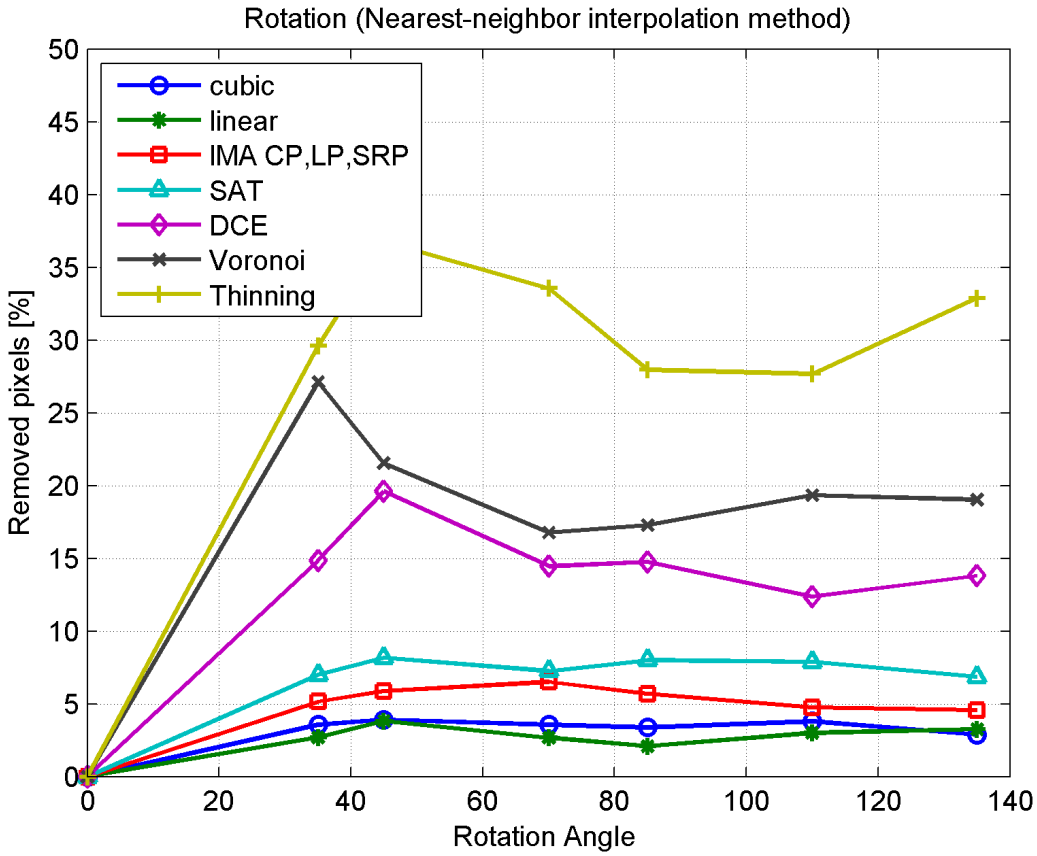


Figure 3.18: Percentage of skeleton pixels not present after the skeletonization of the rotated image (i.e., skeleton pixels that were detected in the original image but not in the rotated version)

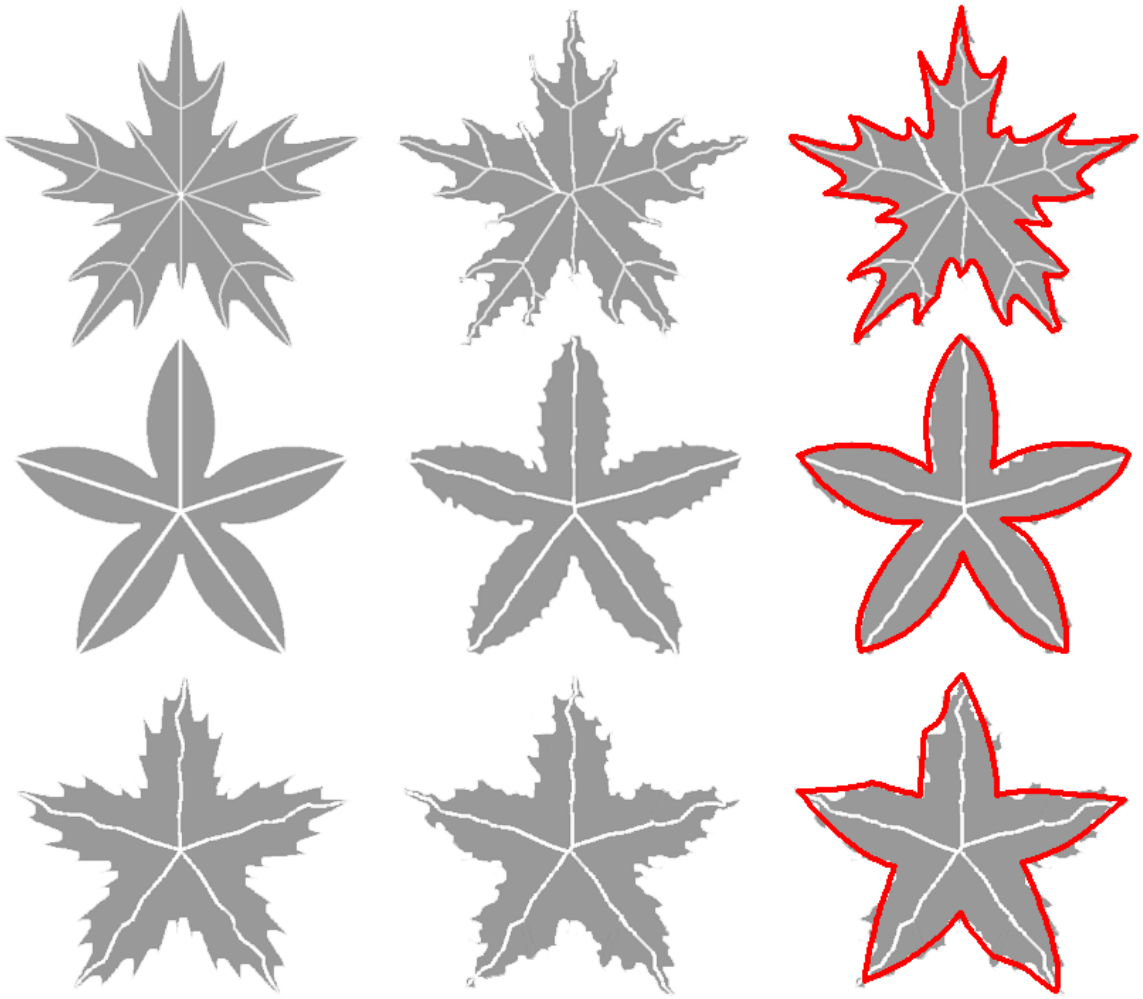


Figure 3.19: Output of our algorithm for regular and noisy images of same image class in the MPEG-7 dataset. Last is the same as middle column but with the contour approximation displayed. Parameters are unchanged from other experiments with $T = 25$ and $S = 1.2$.

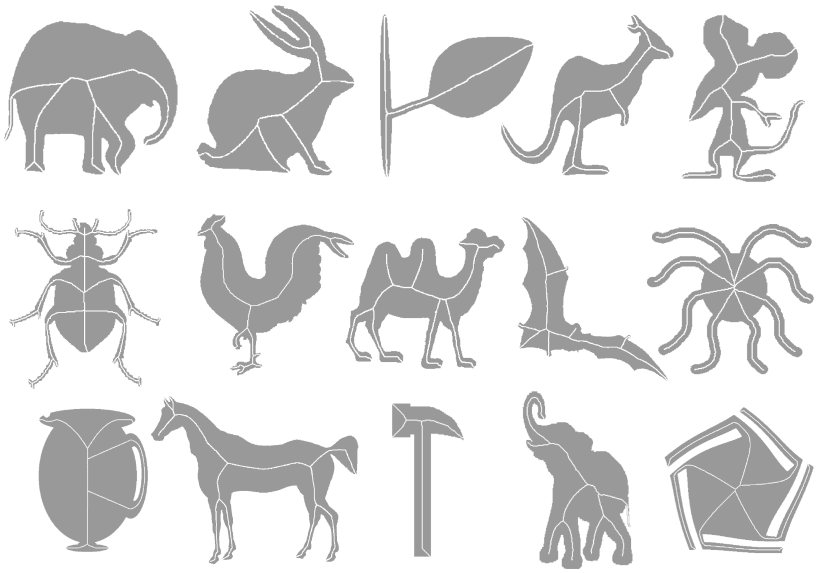


Figure 3.20: More output examples from proposed algorithm. The input parameter selection was the same for the whole dataset.

Figure 3.21: Some more examples of our proposed algorithm of shapes with holes using the same parameter selection.

Chapter 4

Framework for Natural Landmark-based Localization

In this chapter we present a framework for vision-based localization using natural planar landmarks. Specifically, we demonstrate our framework with planar targets using Fern classifiers that have been shown to be robust against illumination changes, perspective distortion, motion blur, and occlusions. We add stratified sampling in the image plane to increase robustness of the localization scheme in cluttered environments and on-line checking for false detection of targets to decrease false positives. We use all matching points to improve pose estimation and an off-line target evaluation strategy to improve *a priori* map building. We report experiments demonstrating the accuracy and speed of localization. Our experiments entail synthetic and real data. Our framework and our improvements are however more general and the Fern classifier could be replaced by other techniques.

4.1 Framework Overview

We review our framework, detailing the background for our evaluation of Ferns during localization. Ferns descriptors track an imaged object by recognizing patches of the object from different perspective views using a classification scheme [105]. Ferns rely on an off-line training phase during which multiple views of the image patches are used to train a naive Bayes classifier. Recognition is based on neighbourhood intensity comparisons. During training, features of the planar patches around a keypoint are synthesized by being randomly warped. A recent study shows the robustness of the Ferns as a feature tracker compared to other descriptors such as SURF, SIFT, Randomized Trees [43]. The Ferns classifier is geared towards tracking in real-time after a training stage. It copes well with dynamic lighting and motion blur due to camera motion. It also shows more robustness with respect to scale and perspective distortions compared to other descriptors. Ozuysal et al. [105] have shown the effectiveness of Ferns descriptors in a SLAM system, as well as their capability to recover from complete failure. However, they did not address performance during pose estimation, which is required by any localization system. The matches between the model target and tracked keypoints using the Ferns descriptors provide us with the 3D-to-2D correspondences needed to estimate the camera pose. The aim is to determine the position and orientation of a camera given its intrinsic parameters and a set of correspondences between 3D points and their 2D projections. The estimated camera pose is relative to the target position, which we assume to be centered at the origin of the world coordinates. We also assume that all the 3D points lie on the plane $Z = 0$. This simplifies our framework's input as we only need a fronto-parallel image of the target and its real dimensions (i.e., in world coordinates) to process it. The camera intrinsic parameters are obtained during an initial calibration step where we account for possible optical distortions. Finally, the target is evaluated in a synthetic environment to assess its quality and expected accuracy.

In our framework, the user manually specifies planar targets. These targets are entered in the system for evaluation and furthermore used for the localization problem. Camera intrinsic needs to be known to the framework for the evaluation and localization steps. Identifying potential planar landmarks could be an extension of our work, where the evaluation step could identify good planar targets. This case of identifying planar targets is out the scope of this thesis and not targeted here.

In the following sections, we provide an overview of the components of our framework for planar object detection. Our framework consist of four major components: camera

calibration, training phase, target evaluation, and pose estimation.

4.1.1 Camera Calibration

The initial step of our framework is to calibrate the camera by estimating its intrinsic parameters (i.e., focal length and optical center) and the distortion coefficients (i.e., radial and tangential factors). The calibration is only computed once for each camera as long as the focal length is fixed. During the calibration process, we determine the relation between the camera pixels (u, v) and the real world coordinate space (X, Y, Z). Assuming it is a pinhole camera model, this relation can be described as a perspective transformation

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{4.1}$$

where K is the 3×3 matrix of intrinsic parameters (i.e., the camera matrix) and $[R|t]$ is the 3×4 matrix of extrinsic parameters which translates a point in world coordinates (X, Y, Z) to the camera coordinate system (x, y, z). For the camera matrix K,

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

the unknown parameters are f_x and f_y (i.e., camera focal lengths) and (c_x, c_y), which are the optical center expressed in pixels. Because real lenses normally have some radial and tangential distortion, we need to take into account the radial and tangential factors. The radial distortion manifests itself in the form of the "barrel" or "fish-eye" effect. For the radial factor we use the following formulas

$$x' = \frac{x}{z}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{4.3}$$

$$y' = \frac{y}{z}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \tag{4.4}$$

where k_1, k_2, k_3 are the unknown radial distortion coefficients and $r^2 = (\frac{x}{z})^2 + (\frac{y}{z})^2$. The tangential distortion occurs because the lenses are not perfectly parallel to the imaging plane. It can be corrected via the formulas

$$x'' = x' + 2p_1\frac{xy}{z^2} + p_2\left(r^2 + 2\left(\frac{x}{z}\right)^2\right) \tag{4.5}$$

$$y'' = y' + p_1\left(r^2 + 2\left(\frac{y}{z}\right)^2\right) + 2p_2\frac{xy}{z^2}, \tag{4.6}$$

where p_1, p_2 are the unknown tangential distortion coefficients and (x, y, z) are the camera coordinates of a 3D point (X, Y, Z). The process of determining K and $D_c = (k_1, k_2, k_3, p_1, p_2)$ is the calibration process.

For our calibration process, we use Zhang's 2D plane calibration technique [162], available in OpenCV. The process only requires the camera to observe a planar pattern shown in a few (at least two) different orientations. Either the camera or the pattern can be freely moved. Knowledge of the plane motion is not necessary. This setup makes the calibration technique easy to integrate into our framework. We use a black-white 9×6 chessboard pattern where each squared box has a $3cm$ side length.

In our implementation, we check each frame of a video feed looking for the chessboard pattern and locate the internal chessboard corners. For example, a chessboard of 9×6 squares has 8×5 internal corners, that is, points where the black squares touch each other. We use OpenCV's *findChessboardCorners* function to determine whether a chessboard pattern with the specified number of internal corners appears in the frame. We further refine corner location by using OpenCV's *cornerSubPix* function. The use of *cornerSubPix* produces better calibration results. Sub-pixel corner location is based on the observation that every point within its neighbourhood is orthogonal to the image gradient. After a chessboard pattern is successfully detected, we stop analyzing the video feed until an input delay has passed (e.g., a second). This allows the user to move the pattern and get different views. Although in theory, only two views are necessary to use Zhang's technique [162], for good results, we accept more than 12 good views of the input pattern in different positions. This is because in practice, we could have noise in our input images and the corner detection process. From each accepted view, we establish the correspondence between the 3D corner locations and their image coordinates in pixels.

Once the desired number of accepted views has been reached, we continue to estimate the camera matrix and distortion coefficients using Zhang's technique. Zhang's algorithm assumes that the chessboard lies on the plane with $Z = 0$ and therefore, the 3D corners and their image points are related by a homography. This constraint is exploited to estimate the four intrinsic parameters and later the extrinsic parameters and distortion

coefficients. Finally, all the parameters are refined by Levenberg-Marquardt's optimization algorithm to minimize the re-projection error, that is, the sum of squared distances between the observed corner points in the views and the projected object points (using the estimated camera matrix and extrinsic parameters). A more detailed description of the technique and the mathematical proof can be found in Zhang's publication [162]. The re-projection error gives the precision of the parameter estimation. The error value should be as close to zero as possible. If the arithmetical mean of the re-projection errors for all the points exceeds a desired value $E_p > .5$ the calibration process is restarted.

4.1.2 Training Phase

In this subsection, we start by reviewing the Ferns basic model and our stratified technique for feature selection. We continue by reviewing the details of the training phase as used in our framework.

Ferns: Basic Model. Keypoints are detected using extrema of the Laplacian operator and binary features $f_1, f_2, \cdots, f_N$ are computed around each keypoint location. The set $(f_j, j = 1, \cdots, N)$ represents the binary features associated to keypoint p_k of size N. The binary features are computed by comparing image intensities in the neighbourhood of a keypoint, that is

$$f_j = \begin{cases} 1 & \text{if } I(p_l) < I(p_m) \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

where $(p_l, p_m) \in \mathcal{N}_k$, with $\mathcal{N}_k$ is the neighbourhood of the keypoint p_k. Given the set of features $(f_j, j = 1, \cdots, N)$, the problem of recognizing a target becomes identifying the class c_i to which a keypoint p_k belongs by maximizing the following joint conditional probability:

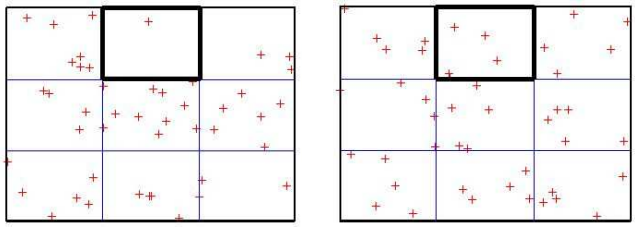
$$\underset{c_i}{\operatorname{argmax}}\, P(f_j, j = 1, \cdots, N \mid C = c_i) \tag{4.8}$$

With a high number of features N, the full joint distribution is too large to be fully represented. Instead, the features are grouped into M groups of size S, referred to as Ferns. By assuming independence between features, the joint probability can be approximated by the product of the individual conditional probabilities, that is

$$P(f_j, j = 1, \cdots, N \mid C = c_i) \approx \prod_{k=1}^{M} P(F_{k,s} \mid C = c_i), \tag{4.9}$$

where $F_{k,s} = \{f_{\sigma(k,1)}, \ldots, f_{\sigma(k,S)}\}$ represents the k^{th} Fern with a random permutation function $\sigma(k, s)$ with range $1, \ldots, N$. Independence between Ferns is crucial to successfully classifying keypoints. Lack of independence among sample points can bias estimates and measures of precision. This assumption holds when patches of different keypoints do not overlap. To reduce the chance of overlaps and to increase the probability that a keypoint lays on an interested area, we propose to use stratified sampling to reduce the variance of estimates.

Keypoint Selection by Stratified Sampling. Stratified sampling is a sampling technique that maintains some characteristics of the data set within its sampled subsets. It is achieved by first partitioning the data set into a number of mutually exclusive subsets of cases, each of which is representative of some aspect of the real-world process involved. In general, sampling from the population is then achieved by randomly sampling from the various subsets to achieve representative proportions. Examples of random sampling and stratified sampling are shown in Figure 4.1. Both use the same total number of samples in nine strata. In stratified sampling, the feature detector is applied on each subset to select the strongest features. Random sampling applies the same feature detection algorithm, but on the entire image. A detailed performance assessment comparing random and stratified sampling is given in Section 4.2.1.



(a) (b)

Figure 4.1: Illustration of random sampling (a) vs. stratified sampling (b). Both illustrations show the same number of samples and strata.

Training the model image. To train the model, we assume that one fronto-parallel image of the target is available. We call any such image a model image. The model image is warped using random transformations to create the training set. Training starts by selecting a subset of the keypoints detected on the training set. This is done by deforming the image many times, applying the keypoint detector, and keeping track of the number of times the same keypoint is detected. The keypoints that are found most

often are assumed to be the most stable. These stable keypoints are assigned a unique class number. The training set for each class is formed by generating thousands of warped images with randomly picked affine transformations. The deformation parameters of the transformation are sampled using a uniform distribution. We add Gaussian noise to each warped image, and smooth it with a Gaussian filter [105]. This ensures the robustness of the resulting classifier to run-time noise, especially when comparing two pixel values on a uniform area.

From the training set, we select the 100 most stable keypoints based on the number of times they are detected in the warped versions. We represent the affine transformations used to warp the model image as a 2×2 matrix of the form

$$R(\theta)R(-\phi)\begin{pmatrix}\lambda_x & 0\\ 0 & \lambda_y\end{pmatrix}R(\phi) \tag{4.10}$$

where $R(\theta)$ and $R(\phi)$ are 2×2 rotation matrices using angles θ and ϕ respectively. The values λ_x and λ_y represent the scaling factors. The deformations are computed by randomly choosing θ and ϕ in the $[0, 2\pi]$ range and λ_x and λ_y in the $[0.6, 1.5]$ range. As in the original publication [105], we use 30 random affine transformations per degree of rotation to produce 10800 images. We add Gaussian noise with zero mean and variance of 25 to gray images with levels in the $[0, 255]$ range, and Gaussian smoothing with a mask of 7×7 to the warped images.

During the training phase, we estimate the class conditional probabilities $P_{k,c_i} = P(F_{k,s} = k|C = c_i)$ for each Fern $F_{k,s}$ given a class c_i. The conditional probability considers when $F_{k,s}$ is to be equal to k; that is the decimal number formed by the binary features taken in sequence. The number of binary features in a Fern is determined by s, the size of the Fern. Finally, we estimate the conditional probabilities $P_{k,c_i}, k = 1, 2, 3, ..., K$, where $K = 2^s$ for each class c_i. The conditional probabilities follow the following constraint: $\sum_{k=1}^{K} P_{k,c_i} = 1$.

The conditional probabilities of a class c_i are computed using patches extracted from the training set centered at a keypoint location. The size of the patches used is of 32×32 pixels. We assign the maximum likelihood estimate to the conditional probabilities

$$P_{k,c_i} = \frac{N_{k,c_i} + N_r}{N_{c_i} + KxN_r}, \tag{4.11}$$

where N_r represents a regularization term. We use $N_r = 1$ for all our experimental data. The term N_{k,c_i} is the number of patches of class c_i where F_{k,c_i} is equal to k. N_{c_i} is the

total number of patches from class c_i.

4.1.3 Target Evaluation

After learning the target, we introduce an evaluation step to assess the quality of the learned model. In this step, we accept targets with high detection rates (i.e., 90% detection rate) that can produce good camera positions and orientations. The target evaluation exploits the planarity constraint of the target to randomly generate different views of the model image. Views are generated by using the estimated camera matrix, and randomly selected extrinsic parameters. Random noise and blur are added to simulate the real capture process. With each view, we try to identify the target and estimate the camera location and position. Finally, we evaluate the target according to detection rate, re-projection error, pose estimation and camera location, just as in our synthetic data simulation in Section 4.2.3. These values serve as a predictor of the *quality* of the target, i.e., an indication of the accuracy to expect when the target would be employed for localization.

The target views are obtained by randomly generating extrinsic parameters $\alpha, \beta, \gamma, \delta_z$ in

$$s \begin{bmatrix} u \\ w \\ 1 \end{bmatrix} = K \begin{bmatrix} & R(\alpha,\beta,\gamma) & & \vert & \begin{bmatrix} 0 \\ 0 \\ \delta_z \\ 1 \end{bmatrix} \\ 0 & 0 & 0 & & \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}, \tag{4.12}$$

where $R(\alpha, \beta, \gamma) = R_z(\alpha)R_y(\beta)R_x(\gamma)$ is the multiplication of the yaw, pitch, and roll rotation matrices. The $R_z(\alpha)$ (yaw), $R_y(\beta)$ (pitch), and $R_x(\gamma)$ (roll) rotation matrices are counterclockwise rotations of α, β, γ degrees about the z-axis, y-axis, and x-axis respectively. The translation in the Z axis δ_z represents the distance from the center of the target to the centre of the camera. The values of α, β, γ, and δ_z are generated randomly using a uniform distribution in the range of $[-\frac{2\pi}{3}, \frac{2\pi}{3}]$ degrees and $[80, 210]$ centimetres respectively. Each corner of the model target in world coordinates (i.e., using the target's dimensions) is projected to the image space to determine the new view. We estimate a homography H between the projected corners and the corners of the model image to remap the fronto-parallel image to the new view. We apply the perspective transformation to the model image by use of OpenCV's *warpPerspective* function. The transformation generates a $dst(x, y)$ image from the source $src(x, y)$ as

$$dst(x,y) = src\left(\frac{H_{1,1}x + H_{1,2}y + H_{1,3}}{H_{3,1}x + H_{3,2}y + H_{3,3}}, \frac{H_{2,1}x + H_{2,2}y + H_{2,3}}{H_{3,1}x + H_{3,2}y + H_{3,3}}\right). \tag{4.13}$$

where $H_{r,c}$ represents the element of the homograpy H at the row r and column c.

The target evaluation uses two metrics to declare a good planar target: true rate detection of the target (i.e., the number of frames from the synthetic generated views in which the target is correctly identified) and the accuracy of the camera pose estimation (i.e., the arithmetic mean of the re-projection errors of the camera location). Targets with a high detection rate and low camera location error are accepted by the framework. We use detection rates higher than 80% and location errors lower than 5 cm.

4.1.4 Pose estimation

For each video frame, we match extracted keypoints to their model target locations using the Fern classifier. We are able to obtain the 3D coordinates of the classified keypoints due to the fact that we have a planar object. Locations have Z coordinates equal to zero and the model target is centered at the origin of the world coordinates $(0, 0, 0)$. Furthermore, we solve the pose estimation problem of a calibrated camera using the 3D-to-2D correspondences created by the Ferns classifier. In our framework, we incorporate the pose estimation algorithm by Schweighofer and Pinz [127]. They propose a real-time pose estimation method to eliminate pose ambiguities of planar targets viewed by a perspective camera. They find the six exterior parameters (i.e., $R(\alpha, \beta, \gamma)$ and $t = [t_X, t_Y, t_Z]^T$) of a calibrated camera by minimizing the object-space error function

$$E_{os}(R(\alpha,\beta,\gamma),t) = \sum_{i=1}^{n} \left\| (I - \widehat{V}_i)(R(\alpha,\beta,\gamma) \begin{bmatrix} X \\ Z \\ Y \end{bmatrix} + t) \right\|^2 \tag{4.14}$$

from n correspondences of four or more coplanar and not collinear points ($4 \leq n \leq N$). The matrix $\widehat{V}_i$ is the observed projection matrix defined as $\widehat{V}_i = \frac{\widehat{v}_i \widehat{v}_i^T}{\widehat{v}_i^T \widehat{v}_i}$. The column vector $\widehat{v}_i = [u_i, v_i, 1]^T$ represents the homogenous coordinates of the matched keypoint i in the image space. The 3×3 matrix I is the identity matrix. The algorithm shows how to use a first guess of a pose R_1, t_1 to estimate a second pose, which also minimizes the error function E_{os}. The second estimation is done by transforming the E_{os} function and making it depend on a rotation about the y-axis $R_y(\beta)$ and t only. The algorithm estimates the local minima values of Equation 4.14 for the parameters β and t, and returns the pose with the lowest error E_{os}. A more detailed description of the technique and the

mathematical proof can be found in the publication [127]. For the first estimation, we use OpenCV's *solvePnPRANSAC* function to account for outliers.

4.2 Experimental Evaluation

We present our framework for the experimental evaluation of localization by planar natural landmarks in five subsections summarizing the performance for pose estimation and camera localization. In each subsection, we give the specific experimental framework and report results obtained for the different experiments. The five subsections are: Stratified Sampling versus Most Stable Keypoints selection (Section 4.2.1), Time Performance and Target Detection rate (Section 4.2.2), Pose Estimation Using Corner Points only versus use of all matched features (Section 4.2.3), Pose Estimation Using Real Data (Section 4.2.4) and finally Target Evaluation (Section 4.2.5). All experiments are executed on a MacBook Pro laptop (2.3 Ghz Intel Core i5 with 4GB of RAM) with an implementation utilizing OpenCV 2.3.1. The camera resolution is 1280×720 in all experiments.

4.2.1 Stratified Sampling vs. Most Stable Keypoints

One of the most important steps in planar object detection is keypoint selection. The selected keypoints are matched against the database of the training set in order to recognize the target in the video frame. Without a sufficient number of good quality keypoints detected on the target, the process of target detection and consequently the camera pose estimation will fail. Commonly, the strongest feature keypoints are selected in an image for processing. Selecting the strongest feature points is motivated by the fact that strong feature points are more likely to be stable under viewpoint changes. However, the strength of feature points in themselves is not always a sufficient criterion for success in target recognition. The feature points also have to be part of the target rather than part of the background scene. We study therefore a stratified sampling strategy where we tessellate the image plane and select strong features in each stratum.

We conducted experiments in order to compare the two strategies: a pre-selected number of the strongest feature points and the above stratified sampling strategy. We positioned a target in a cluttered environment and compared the detection results using different numbers of keypoints with the two approaches. The strongest feature points strategy is configured to select a maximum of 300, 700 and 1000 keypoints, respectively. Our stratified sampling approach uses a 7×7 grid where the most relevant points in each cell are selected. For comparison, the total number of keypoints in the image is kept the

same for both approaches.

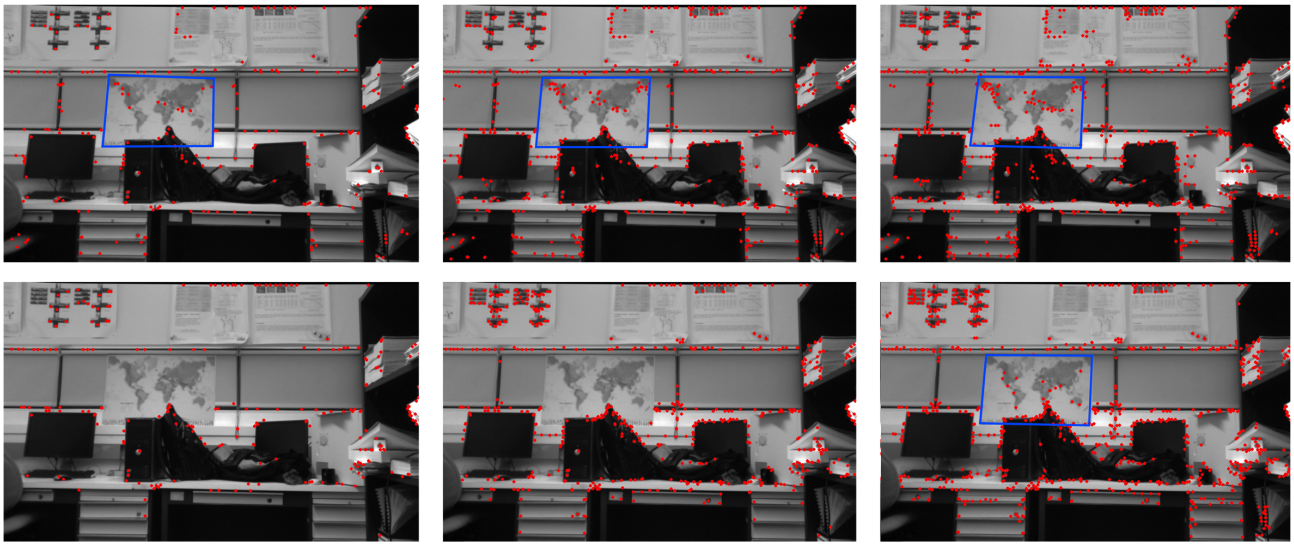


Figure 4.2: Top row shows the stratified sampling approach with a total of 300, 700 and 1000 keypoints detected. In the bottom row the top 300, 700 and 1000 keypoints in the image were detected.

In our experiments, the stratified sampling solution was more robust for target detection, detecting the target (here, a poster with a world map) for all the keypoints configurations in our experiment. This is because the sampling strategy inspects the whole image in search for the most representative points of each cell. When selecting the strongest points, we only were able to detect the target with 1000 keypoints while with a maximum of 700 and 300 no keypoint was detected in the target region. In cluttered scenes, the target is not necessarily the object giving rise to the most keypoints in the image. The surrounding objects may have more representative features and may therefore attract all of the keypoints. In the current test image, we needed to increment the number of keypoints to detect some in the target's area (see Figure 4.2). However, even increasing the number of keypoints is unlikely to work consistently because in dynamic scenes new clutter can appear in the image and may potentially attract more keypoints than the preset maximum. The stratified sampling solution is more stable in this scenario because adding objects to one region won't affect the keypoint selection in other regions. Figure 4.3 shows that the stratified strategy is able to detect the target despite the insertion of more clutter that partially occludes the target.

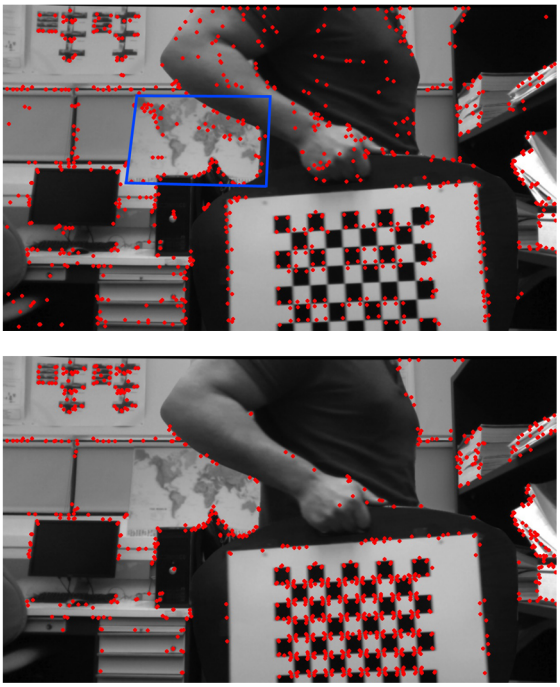


Figure 4.3: Stratified sampling (top) and top most stable keypoints (bottom) with a total of 1000 keypoints detected in both.

4.2.2 Time Performance and Target Detection

While the above example demonstrates the improvements achievable with stratified sampling, we are interested in the impact of the strategy on localization. Successful localization depends on the target detection success rate, the occurrence of false positives and the execution time. Execution time in a real-time application such as localization will have a direct impact on the utility of the method. Therefore, we evaluate the impact of using the two previous strategies in terms of processing time for each of the steps in detecting a planar target and computing the pose. We include target detection rate as a quality benchmark. For this experiment, we used a recorded video of 1627 frames containing the target seen from a moving camera directed towards the world map target on a random path. The camera has been calibrated with the aid of a printed checkerboard and the calibration procedure available in OpenCV [18]. The timed steps are: remove camera distortion (CD), image pre-processing (PRE), feature extraction (FE), feature matching (FM), and pose estimation (PE). In the experiment we use a cap of a total of 1000 keypoints for both strategies.

The FE and FM steps dominate the processing time of the algorithm using about 80% of the total execution time of the five steps (see Figure 4.4). In general, the total execution time is about the same for both approaches, in particular, stratified sampling does not increase the overall execution time of the localization. The execution time of FE depends on the method of feature extraction, i.e., the Laplacian of Gaussian as in the original Ferns approach [105]. The execution time of the FM step is affected by the number of features to match against in the database and the extraction of their descriptors. The time complexity of classifying a keypoint is $O(M)$, where M is the number of classes in the database (i.e., we use 100 classes). On the other hand, the detection rate improves greatly with the stratified sampling approach while the false detection reduces simultaneously (see Figure 4.5).

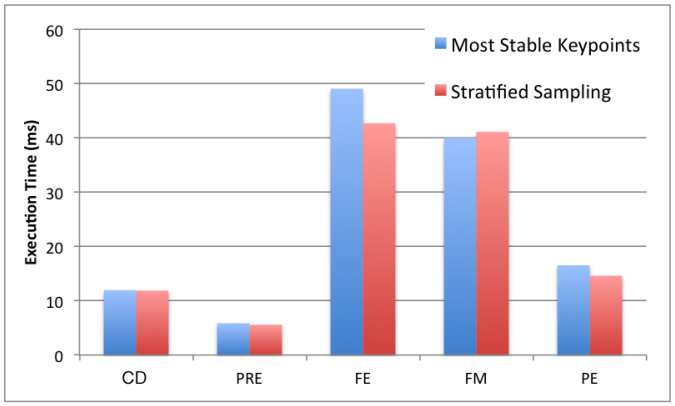


Figure 4.4: Execution time comparison (in milliseconds) between Most Stable Keypoints and Stratified Sampling. From left to right: Camera Distortion, Pre-processing, Feature Extraction, Feature Matching and Pose Estimation.

We propose to include an on-line check for detection of false positives after matching keypoints. Using prior-knowledge of the fact that we are looking for a planar object imposes constraints on the image of the keypoints belonging to the target. After removal of camera distortion, all the inliers must be imaged inside the area of the estimated target and this image must be planar. We check if the convex hull of the features for the estimated target are inside the corners of the estimated target pose projected into the image. A false detection is flagged when the obtained homography from the inliers computed using RANSAC [40] does not project to a planar target. The average processing time of the planarity check is about 0.015 milliseconds for each video frame. It has therefore negligible impact on the total execution time for pose estimation.

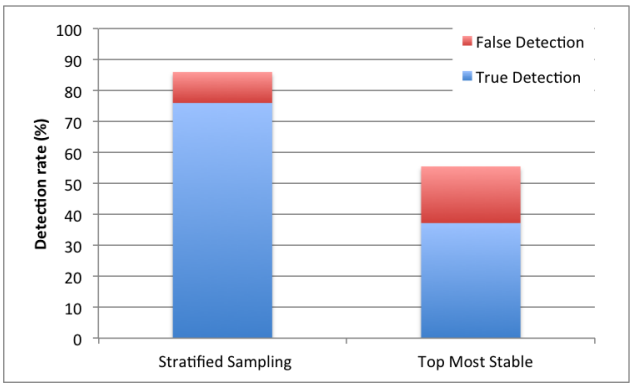


Figure 4.5: Detection rate (false detection, true detection) for stratified sampling and strongest keypoints.

4.2.3 Corner Points vs. Matched Points

In the next experiment we compare the impact of computing the pose, i.e., solving the PnP (perspective n-point) problem using two slightly different approaches.

For this experiment, we render images by projecting the target into the image plane from different angles and depths. We use the camera's intrinsic parameters obtained in the calibration of the real camera through the camera module of our implementation. We use the synthetic images to test for planar target detection accuracy and the subsequent pose estimation using the method by Schweighofer and Pinz [127]. In the synthetic images we know the exact projection of the target given the orientation and position of the camera. We generate 1000 images at three different distances from the target with orientation angles between 30 and 150 degrees and compute the projection, translation and rotation error. When generating the image using the camera information, we include random background and foreground noise, and random Gaussian Blur to simulate the real capture process (see Figure 4.6). The background is filled with uniform noise with parameters from 0 to 255 grayscale values. The foreground noise added is additive white Gaussian noise with zero mean and variance of 10 grayscale values. The Gaussian Blur uses uniformly distributed random kernel size (ksize), odd values from 3 to 9 and standard deviation $\sigma = 0.3 \times (ksize/2 - 1) + 0.8$.

The first approach, named here *Corner Points* (CP) uses only four points (i.e., corners of the target) to estimate the pose. The corners of the targets are acquired by computing the homography from the training image and the matching points in the video frame. The homography is obtained using RANSAC [40]. Transforming the four corners of the

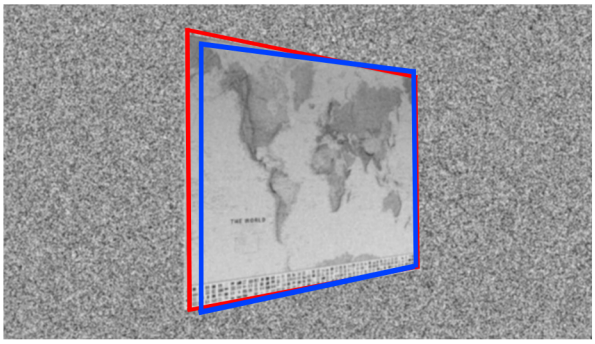


Figure 4.6: Example of generated target view by rotation around the image y-axis. Random noise and blur are added to simulate the real capture process. Red boundary is the ideal target projection, blue boundary an estimate.

target's training image we can get the position of the target in the video frame. The pose estimation is finally computed by the 3D correspondence of the four corners of the target and the translated corners positions (see Figure 4.7).

For the second approach named here "Matched Points" (MP), we skip the homography computation and the projection of the corners. Instead we use all the matching points of the target between its training image and the analyzed video frame to compute the extrinsic camera parameters. The 3D position of the matched points are obtained from the four 3D coordinates of the target's corners and the fact that it is a planar object and we know the image coordinates of the match in the training image (see Figure 4.7 right).

The projection error for CP is computed as the average distance of each corner between our approximation and the real projection (see Equation 4.15). For the MP the projection error is computed as the average distance of each matched point's real projection and its detected position by the Fern classifier (see Equation 4.16).

$$CP_{error} = \frac{\sum_{i=1}^{4} |C_i - \hat{C}_i|)}{4} \tag{4.15}$$

$$MP_{error} = \frac{\sum_{i=1}^{n} |M_i - \hat{M}_i|)}{n} \tag{4.16}$$

The rotation error is the angle in degrees between the normal of the projected and estimated target. Translation and camera location error are defined as the norm between the real and the estimated value.

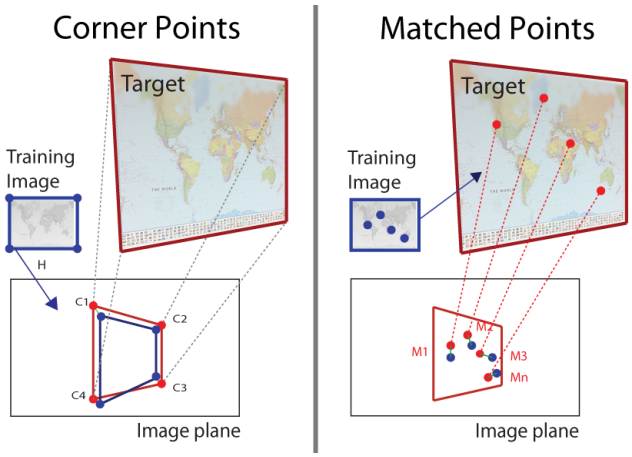


Figure 4.7: Outline of the Corner Points and Matched Point approaches. CP uses the matched points to obtain the homography and then translate the training image to the video frame. PnP is solved only with the four corners (i.e., $C_1...C_4$) of the target. MP translates the matching points in the training image to its 3D coordinates and solve PnP using their correspondence (i.e., $M_1...M_n$).

MP outperformed CP in all the classifications, increasing the detection rate and reducing false positives, re-projection, translation, rotation and camera location errors (see Table 4). MP is a more stable approach than only using the four target's corners directly from the homography estimation.

4.2.4 Pose Estimation Using Real Data

For this experiment we placed four targets and positioned the camera pointing towards their center at different ranges and view angles. Accurate ground truth for the pose of the camera is not available to us, therefore we evaluate the accuracy in terms of quantities that can be precisely measured. We place multiple targets at precisely known relative position and orientation from each other. Now, if we estimate the pose of the camera between two or more targets, we can calculate the difference between those poses. In a common coordinate system, the difference of the poses will be exactly the spatial transformation between the targets (see Figure 4.8). Suppose (R_a, T_a) and (R_b, T_b) are the pose of the camera with respect to a pair of targets (a, b), respectively. Then the relative pose between the two targets satisfies

$$\hat{T} = T_a - RT_b \text{ and } R = R_a R_b^{-1}. \tag{4.17}$$

Table 4: Comparison between MP and CP at different target's depths

Target Depth	200 cm		300 cm	
	MP	CP	MP	CP
True Detection (%)	92	90	86.1	82.9
False Detection (%)	3.8	5.8	3	6.2
Reproj. error (pixel)	7.2	11.4	7.46	16.3
Translation error (cm)	1.31	1.93	2.32	4.2
Rotation error (angle)	1.32	1.44	2.11	3.82
Location error (cm)	5.19	6.02	13.736	21.138

Figure 4.8: Detection output example of our implementation using the four target setup.

The errors of the relative translation and rotation are defined as the Euclidean distance between the real translation T and its estimation $\hat{T}$ and the angle between the two estimated normal vectors n_a and n_b of the planer targets using the dot product (see Equation 4.18).

$$e_T = |T - \hat{T}| \text{ and } e_R = arccos(n_a \cdot n_b). \tag{4.18}$$

The pose estimation is evaluated between four targets having different textures. Figure 4.9 shows those targets and their relative translations. The rotation between each pair of targets is the identity matrix. We placed the camera around the center of the four targets in nine equally spaced directions from $-60°$ to $60°$ degrees, and at different distances with respect to the center of the targets from 0.90 m to 2.10 m. We obtained

an average translation error of 2.74 cm over the whole dataset, and a rotation error of 2.6251°. These values are in line with our experiments with synthetic images in Section 4.2.3. Next, we will discuss our strategies to evaluate the feasibility of a target for localization based on an *a priori* test.

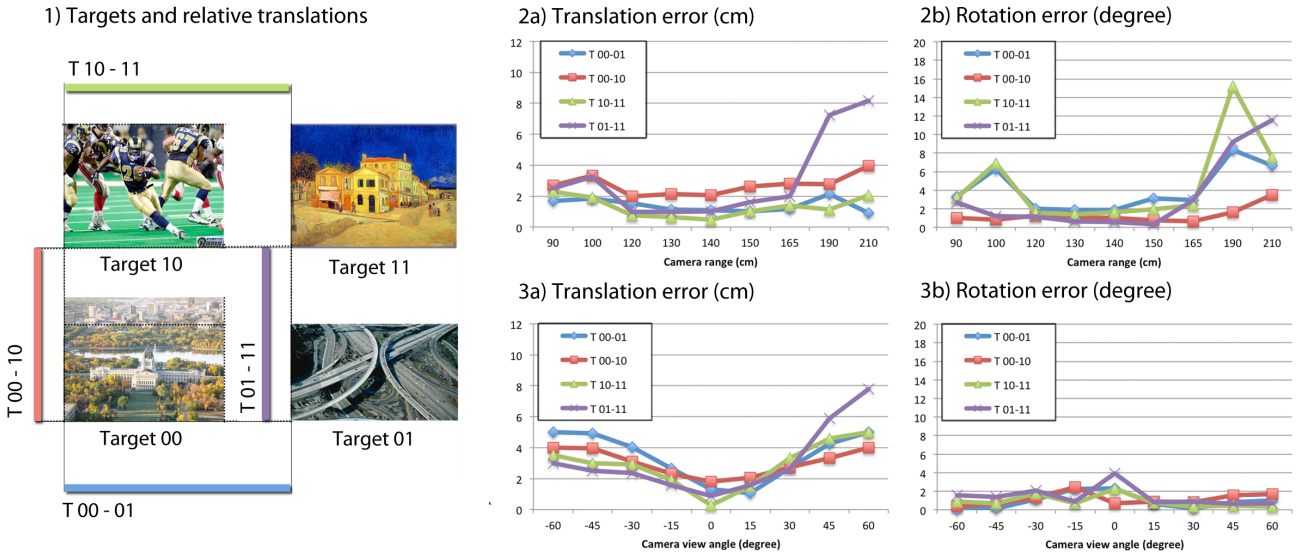


Figure 4.9: Image 1) shows the four targets $Target - \{00, 01, 10, 11\}$ and their positions. The translation $T00 - 01$ is the relative translation from $Target$ 00 to $Target$ 01. The same scheme applies for the other three relative translations shown in (1). The target dimensions are 27x20 cm, except for $Target - \{01\}$ which is 27x18 cm. Images 2a) and 2b) are the translation and rotation errors computed while moving the camera from 0.90 m to 2.10 m ranges. Images 3a) and 3b) are the translation and rotation errors with camera view angles from $-60°$ to $60°$ at 150cm range.

4.2.5 Target Evaluation

Pose estimation is highly dependent on the target detection process. The number of correct matches between the extracted keypoints and the training images of the target improve the accuracy. We use natural targets to solve the localization problem; therefore, the selection of a target with stable repeatable keypoints from different views and depths will aid the performance of the localization. We therefore like to evaluate the feasibility of a target before adding it to the map. To evaluate a target we use the results of the target's performance measured using synthetic data.

As an illustrative example, we take the four targets from the previous section and run the synthetic target evaluation on them. The results are highly related to the pose computed from real data (see Section 4.2.4). The target evaluation strategy confirms the instability of $Target$ 01 and $Target$ 11 compared to the other two (See Figure 4.10). The pose estimation inaccuracy of these two targets is therefore the likely cause of the translation and rotation errors in the real data experiment (i.e., translations T00-01, T01-11, and T10-11 are noisy if the camera distance is beyond 1.65 m). This gives us confidence to use the above evaluation process as a target selection strategy before adding targets to the system's database.

4.3 Summary and Conclusion

In this chapter we presented a framework and experimental evaluation of the Fern classifier for localization using natural landmarks along with practical improvements to increase the performance of the scheme. These improvements are stratified sampling, on-line checking of false target detection, use of all matching points to improve pose estimation and an off-line target evaluation strategy. Our stratified sampling technique is able to double the true rate of detection while reducing false target detection without affecting execution time. On-line checking of the detection of a planar target helps to reduce false positives. The use of all matching points vs. simply the corners of a rectangular target improves the accuracy of localization by up to 50%. Finally, our off-line target evaluation predicts the quality of the target and hence, the precision to expect when the particular target would be employed as part of a map. The experimental results illustrate that natural targets and Fern classifiers with our additions are a feasible solution for visual localization. All our extensions of the Ferns localization are not specific to Ferns and could be integrated with other robot localization schemes.

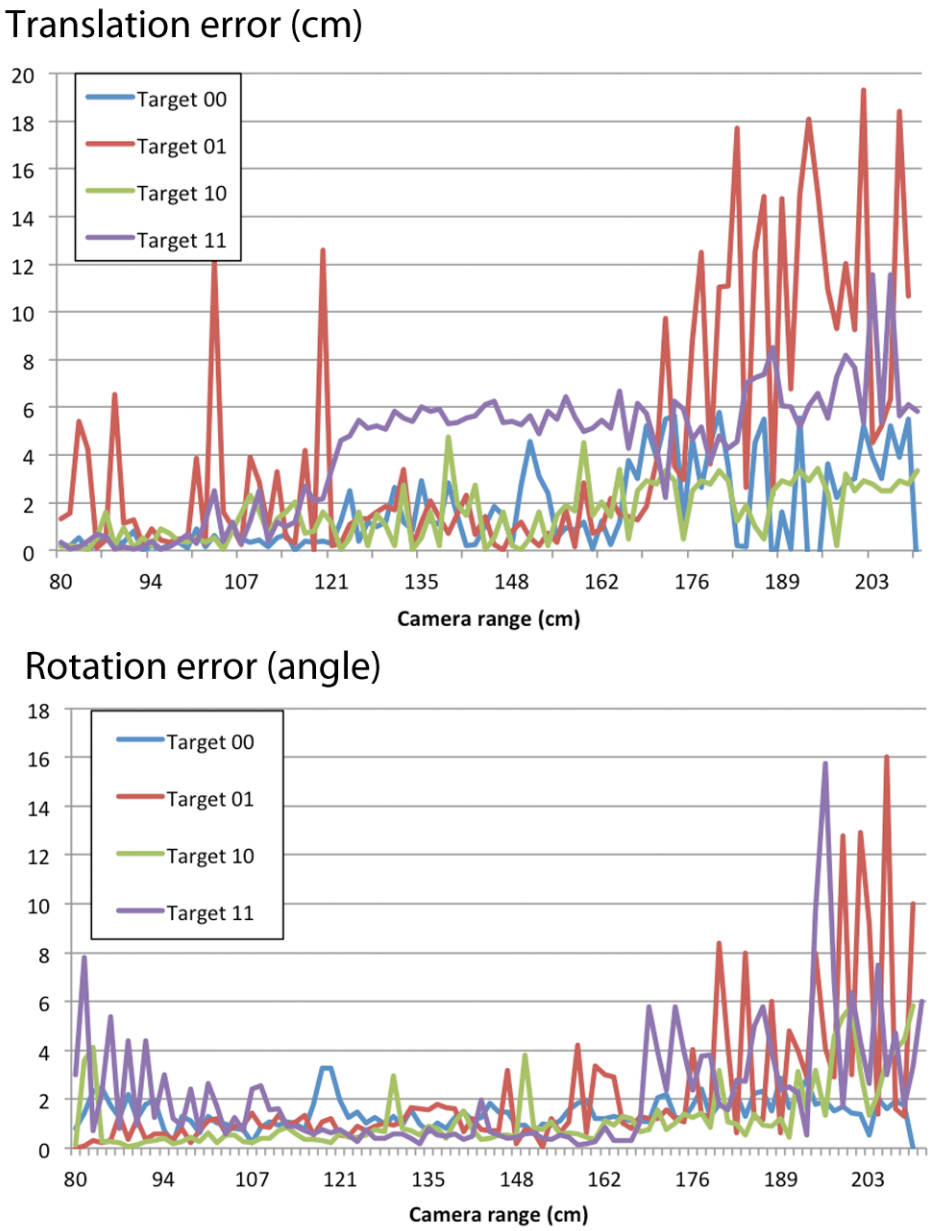


Figure 4.10: Target evaluation of the four targets. Translation and rotation error show the instability of *Target* 01 and *Target* 11 compared with the other two. In the experiment, we simulate the movement of the camera in front of the target from 0.8 m to 2.1 m. The average translation error for the complete set is 2.85 cm and 1.45° of average rotation error.

Chapter 5

A General Framework for Fast 3D Object Detection and Localization Using an Uncalibrated Camera

In this chapter, we present a real-time approach for 3D object detection using a single, mobile and uncalibrated camera. We develop our algorithm using a feature-based method based on two novel naive Bayes classifiers for viewpoint and feature matching. Our algorithm exploits the specific structure of various binary descriptors in order to boost feature matching by conserving descriptor properties (e.g., rotational and scale invariance, robustness to illumination variations and real-time performance). Unlike state-of-the-art methods, our novel naive classifiers only require a database with a small memory footprint because we store efficiently encoded features. In addition, we also improve the indexing scheme to speed up the matching process. Because our database is built from powerful descriptors, only a few images need to be "learned" and constructing a database for a new object is highly efficient.

5.1 Framework Overview

As in Ferns [104], Taylors and Drummond [141], and Akhoury [1] we create a two-phase framework: an off-line feature training and a runtime-matching scheme for object detection and localization. During the training phase (Section 5.2.1) we learn the model of the 3D object of interest based on a set of images of the object from different viewpoints. Each image is randomly warped several times for learning the most relevant keypoints and their binary descriptors. We then use the most stable keypoints for all views and their set of descriptors to train our Naive Bayes Classified Descriptors (NBCD) (Section 5.2.2). A tree-like index data structure from the classifiers is created to speed up classifications (Section 5.2.3). As the final steps, we compute the keypoint correspondences between the images using a RANSAC approach and a likelihood test to validate the geometrical constraints (Section 5.2.4). During the runtime object detection and localization phase, we extract the most relevant keypoints from a video stream. Once the match between the database and the video frame is created, we have sets of matching points for all close database views. These matched keypoints are used to select the closest view in the database with another Bayes classifier (Section 5.2.5). Finally, the closest view is projected back (Section 5.2.6) to the video input and tracked using a Kalman Filter (Section 5.2.7).

5.2 Implementation

5.2.1 Training Features

We use a training set of images covering the range of viewpoints of the entire 3D object of interest. Our approach warps each reference image using random transformations from different viewpoint bins and scales similarly to [1, 141] and [104]. Instead of applying the conventional affine transformation (Section 4.1.2), we apply a perspective transformation which allows us to sample the deformation space in a more controlled way requiring less images per viewset. We ensure that the plane will remain entirely visible under an arbitrary rotation around any of the three axes. Each warped keypoint selected for voting is back-projected to the original reference image. Stable keypoints $\{k_1, k_2, k_3, \ldots, k_n\}$ are selected as the points with more votes [104]. In our training stage we use binary descriptors instead of images patches. Also we do not have to create multiple viewpoint "bins" to train thanks to the invariant properties of the descriptors. For each artificially warped image, we extract a binary descriptor (e.g., BRIEF, ORB, BRISK, FREAK) of its

relevant keypoints. The descriptor is a binary string $[f_1, f_2, \ldots, f_L]$ of length L, denoted $D = f^{(1:L)}$ (we consider each bit of the string as a feature). Once the keypoints are back-projected and merged with the reference image keypoints, each keypoint contributes with a descriptor D extracted in the warped image. Typically, 1000 images are generated for each viewpoint bin. The use of binary features instead of image patches reduces considerably the amount of memory needed while creating the dataset of most stable keypoints before training the classifier. A grayscale image patch of 32x32 pixels with 8-bit depth has a memory usage of 1 Kb compared to 32 bytes (128 bits) in case of ORB, or 64 bytes (512 bits) in case of FREAK, or BRISK. The number of viewpoints bins used in training is related to the properties of the binary descriptor. In previous solutions [141, 104, 1], the reference images needed to be trained from all the possible views for detection. With rotation-invariant descriptors there is no need to train for the 360° in-plane rotations around the keypoint. The same idea applies when the descriptor is scale-invariant. In fact, all our examples in this Chapter use only one viewpoint bin with perspective transformations around the center of the image frame. Aside from the numbers of bins, we use the same perspective warping training as in [1] (i.e., we generate 1000 viewpoints inside a single bin with yaw and pitch of $[-45°, 45°]$ each, and a roll of $[-30°, 30°]$, and 3 scale factors of $\{0.6, 1, 1.6\}$.

5.2.2 Naive Bayes Classified Descriptors

We computed the set of most stable keypoints and their associated descriptors and learn the distribution of the binary features describing a keypoint. Formally, we want to be able to assign the most likely keypoint k_i given a binary descriptor D, $\arg\max_i P(K = k_i | f^{(1:L)})$. The event $f^{(i)} \epsilon \{0, 1\}$ is the binary value of the descriptor at i^{th} index.

Assuming independence between the descriptor features, a uniform prior $P(K)$, our problem reduces to the maximum likelihood classification problem. We assume all keypoints are equally likely for simplicity but it is possible to learn the prior distribution while training. In contrast to Ferns [104] we do not assume dependencies between the binary features combining them into groups. Our experiments demonstrate correct classification of binary descriptors into classes while removing the exponential memory use of Ferns. Each binary feature of the descriptor contributes a specific weight to the classification giving a strong *unique* representation. Repeated observations with invariant descriptors reinforce a feature. For example, a rotation invariant descriptor should have the same value for each index under any rotation but in practice the binary feature is

not completely stable. In our Bayes classifier, we describe the likelihood of every single bit inside the descriptor without knowing nor learning the relation with other features but at the same time benefiting from the descriptor's invariance.

Each binary feature's conditional probability is learned as the maximum likelihood estimation of the mean with a scale factor to ensure that the model is not invalidated when there is not enough training data or one of the features has zero probability [62].

$$P(f^{(i)} = 1|K = k_i) = \frac{\sum_{j=1}^{M} f_j^{(i)} + N_c}{M + C \times N_c} \tag{5.1}$$

Where $\sum_{j=1}^{M} f_j^{(i)}$ is the sum of the i^{th} bit values of all the descriptors associated with keypoint k_i and M is the total number of descriptors belonging to keypoint k_i. In practice, the value of N_c does not influence the results, we use the values of $N_c = 1$ and $C = 2$ because we have two possible classifications $f^{(i)} \epsilon \{0, 1\}$.

We reduce memory and computation by storing the logarithmic values $logP(f^{(i)} = 0|K = k_i)$ and $logP(f^{(i)} = 1|K = k_i)$ for bit i in a single byte. Each logarithmic probability is stored using 4 bits and hence values in the range of 0 to 15. The logarithm of a probability value is a negative number but we just save the absolute rounded integer value (see figure 5.1). Finally, for each keypoint we use a lookup table with L *bytes* where L is the size of the binary descriptor in bits. The i^{th} byte encodes the negative conditional log-probability of the feature $f^{(i)}$ given the keypoint. For each byte the 4 most significant bits encodes the probability of feature i being 0 given keypoint k_i and the least significant bits the probability of the binary feature being 1.

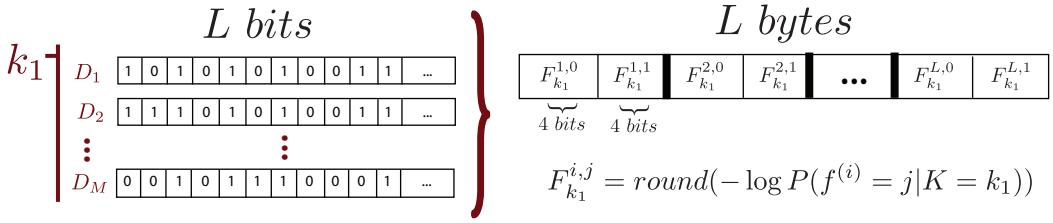


Figure 5.1: Given a point k_1 and its associated descriptors $\{D_j, j = 1, 2, ..., M\}$, we encode $F_{k_1}^{(i,j)} = round(-\log P(f^{(i)} = j|K = k_1))$ as the negative round value of the log conditional probability. We use only 4 bits (i.e., values from 0 to 15) to store each conditional probability, making a byte for each binary feature in the descriptor.

5.2.3 Indexing

We do not use standard pattern-based indices because they are not rotationally-invariant, e.g., the patterns used in [141] suffer deformations under different keypoint orientation which causes the same keypoint to generate different indices depending on orientation. Computationally expensive solution such as sub-pixel interpolation are not fast enough for indexing but high repeatability is nevertheless required. We propose a different index computation based on certain pattern positions around the keypoint, we select a list of binary features indices from the computed binary descriptor. This ensures that two keypoints with the same descriptors will have the same index value. Also, this scheme allows us to have a variable index size L according the database size and we select the proper index size (i.e., number of bits) accordingly.

As in [141], we assign an integer index value to each binary descriptor corresponding to the most relevant keypoint k_i by combining their binary values. Finally, we select the most repeated indices for each keypoint. Every index value will be pointing to a list of keypoint classifiers containing the same index (e.g., for a 13-bit index we will have a maximum of $2^{13} - 1$ index values). Each index will have approx. $\frac{n}{2^L} \times Z$ keypoint classifiers where Z is the amount of indices per keypoint added to the database, L is the number of bits of the index and n the total amount of keypoints in the database. This makes our indexing search order $O(\frac{n}{2^L})$, and in practice it is fast with an appropriate choice of L.

5.2.4 Probabilistic Model for Fundamental Matrix Verification

Our fundamental matrix verification step uses the likelihood-ratio test of Brown and Lowe [21]. The test denotes n_m and n_i being the number of matches and inliers defined by RANSAC. The event of a correct/incorrect Fundamental Matrix is represented by the binary variable $F_C \, \epsilon \, \{0, 1\}$. The event that the i^{th} keypoint match $K_M^{(i)} \epsilon \, \{0, 1\}$ is an inlier/outlier is assumed to be independent Bernoulli, so that the total number of inliers is Binomial.

$$P(K_M^{(1:n_m)} | F_C = 1) = B(n_i; n_m; p_1) \tag{5.2}$$

$$P(K_M^{(1:n_m)} | F_C = 0) = B(n_i; n_m; p_0) \tag{5.3}$$

Where p_1/p_0 is the probability a keypoint is an inlier given a correct/incorrect Fundamental Matrix. The set of keypoint matches is denoted $K_M^{(1:n_m)}$ and $B(.)$ is the Binomial distribution. The fundamental matrix is accepted if $P(F_C = 1 | K_M^{(1:n_m)}) > p_{min}$ which

leads to the likelihood-ratio test

$$\frac{B(n_i; n_m; p_1)P(F_C = 1)}{B(n_i; n_m; p_0)P(F_C = 0)} \gtrless_{reject}^{accept} \frac{p_{min}}{1 - p_{min}} \tag{5.4}$$

Following Brown and Love [21],we choose values of $P(F_C = 1) = 10^{-6}$, $p_{min} = 0.999$, $p_1 = 0.6$ and $p_0 = 0.1$ leads to $n_i > 8.0 + 3.0n_m$.

5.2.5 Naive Bayes Classifier for View Matching

Given an observation of keypoints, view matching finds the closest view in our database to a frame in the video sequence. Let $V = \{v_j, j = 1, 2, ..., v\}$ be the set of views in our database, each view $v_j = k^{(1:n)}$ is composed of a set of most stable keypoints from the training phase. The event that the k_i keypoint belongs to view v_j is represented by a binary variable $K_j^i \epsilon \{0, 1\}$ that we assumed to be independent Bernoulli.

Formally we are looking for the most likely view given an observation of keypoints in the video frame, $\arg\max_{v_j} P(V = v_j | K_j^{(1:n)})$.

Using Bayes Formula and assuming a uniform prior $P(V)$, neglecting the constant denominator, our problem reduces to finding the maximum likelihood $\arg\max_{v_j} \prod_{i=1}^{n} P(K_j^i | V = v_j)$. Because we assumed K_j^i to be independent Bernoulli given v_j, the total number of inliers becomes binomial.

$$\prod_{i=1}^{n} P(K_j^i | V = v_j) \;=\; B(n_i, n, \omega_j) \tag{5.5}$$

Where ω_j is the probability of a keypoint corresponding to view v_j, n is the total number of keypoints in a view and n_i is the amount of inliers and $B(.)$ is the Binomial distribution. Note that if we assume ω_j to be the same for every view v_j, the most likely view from our model database will be the one with the highest number of inliers.

5.2.6 Object Detection and Localization

Once we know the fundamental matrix between the closest view in the database and the video frame, we proceed to find the location of the object. However, we neither have 3D object information nor the camera intrinsic parameters. Instead of devising a computationally expensive camera auto-calibration process, we project the object's boundary in the closest view from the database to the video frame. The approximate object boundary has been previously identified by the user during the training stage. We simply assume that there is a plane that contains the object in the 3D space and compute

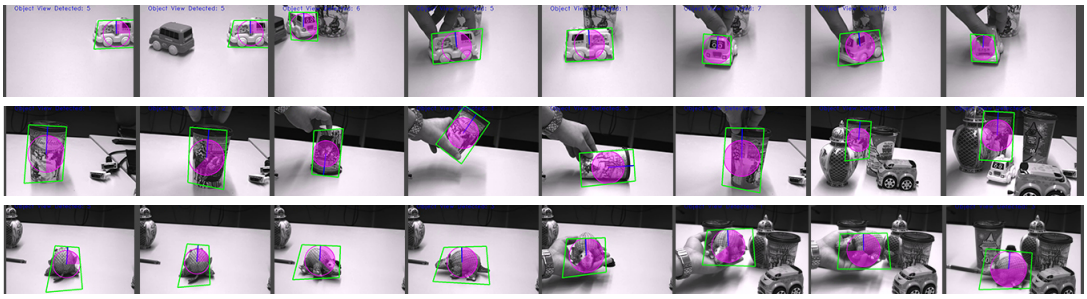


Figure 5.2: Object detection examples. The different objects in each row from top to bottom were trained with 8, 6 and 10 views, and the recognition rate was 91%, 93%, and 80% respectively. The circle inside the detection box represents the ratio of training views over the total training views used in the detection (full circle corresponds to all training views used).

a homography between the two views. The relationship between this homography H_ϕ and the Fundamental Matrix F is given as $F = [e']_X H_\phi$ where $[e']_X$ is a skew-symmetric matrix defined by Hartley and Zisserman [49]. The homography is computed using the RANSAC inliers of the Fundamental Matrix. The final projection of the objects boundary is forced to be on the epipolar lines by moving it to the closest point on its corresponding epipolar line.

5.2.7 Kalman Filter for Object Prediction

Each corner of the object bounding box is independently processed by a Kalman filter [155] at each video frame to produce a statistical estimate of the underlying system. The corner measurements are in pixel coordinates and passed through the filter each time we accept the likelihood-ratio 5.2.4. In case of a rejection we use the filter's predictions to narrow the search of the object in the next frame, which reduces the search between the keypoints in the video frame and database. If the object is not detected in the predicted area, the region is expanded to the frame size. The main reason of using a Kalman filter is to compensate for the noise associated with keypoint classifiers, for the noise in the Fundamental Matrix computation, and also to stabilize the detection results. We do not attempt to model the object's dynamics accurately and hence we simply use a constant velocity model for each corner separately.

5.3 Experimental Results

We compare our framework against Ferns and Binary Descriptors in terms of performance, memory usage, and running time. Our framework uses the binary descriptors' implementations of OpenCV (i.e., BRIEF, ORB, BRISK, FREAK) with their default parameter configurations. The Ferns implementation was the authors' [104] from their webpage. We use it unmodified except for adapting their training intervals to show the impact of training on their solution and ours.

5.3.1 Memory Usage

First, we compare the memory usage between binary descriptors without a training phase for Ferns and our framework. It is easy to see that binary descriptors are the most compact with a memory usage of $bits \times K$, where $bits$ is the size in bits of the descriptor and K the amount of the keypoints in the database. On the other hand, the memory footprint of the Ferns database grows exponentially with the Ferns' size S, i.e., $2^S \times M \times byteS \times K$, where M is the number of Ferns and $byteS$ is the size of bytes used to store the conditional probabilities. The original Ferns implementation uses $S = 11$, $M = 30$ and $byteS = 4$ (float). However, the memory needed for our database is $8 \times bits \times K$, i.e., it is $O(K)$ just as the binary descriptors. We use a byte to represent every bit of the descriptor. For example, 1000 keypoints using the BRIEF descriptor will require 31.25Kb for storage, Ferns will require 234Mb, and ours 250Kb. Or put differently, adding an extra keypoint to the Ferns database will require 240Kb, almost as much as another 1000 points in our representation.

Figure 5.2 shows three tracking results with our framework. The car object in the first row uses a database with 8 views for a total of 450Kb, the cup in the second row with 6 views uses 350Kb, and the toy in the third row uses 10 views and 550Kb. Each view contains ∼200 keypoints.

5.3.2 Performance

Next, we evaluated the performance of each algorithm under different images transformations. We synthetically generated perspective transformations of the planar object (see Figure 5.3) at different scales and different positions in the image with changes in contrast and brightness (see Table 5). The background is filled with white noise. See Figure 5.3 for some example transformations.

In considering Table 5, it is important to take training into account. The parameters for the transformation during the feature learning (Section 5.2.1) with our Naive Bayes Classified Descriptors (NBCD) are the same for all objects with rotations of roll of $[-30, 30]$, pitch and yaw of $[-45, 45]$, and with three scale levels $\{0.6, 1, 1.6\}$. The average training time for each view is $\sim$25 seconds. Although the range of training transformation is considerably less than the object transformation for the detection test, our NBCD with BRISK or ORB has the highest detection rate in all three tests in Table 5. Our NBCD exploits properties of the descriptor, e.g., BRIEF lacks rotation invariance and hence our NBCD with BRIEF also lacks this feature but our NBCD with ORB demonstrates rotational invariance because it is a feature of ORB. Ferns use affine transformations in their training. We compare to them with two set of transformations in training. The set Ferns in Table 5 uses the same training parameters than for our NBCD and the set Ferns* uses a full range of rotations and a wider scale range of 0.5 to 1.5.. The average training time for Ferns and Ferns* was $\sim$170 and $\sim$360 seconds, respectively. The results in Table 5 demonstrate the dependency of Ferns on training for all possible view transformations to successfully detect an object. In contrast, our NBCD with the corresponding descriptor is successful even with limited training.

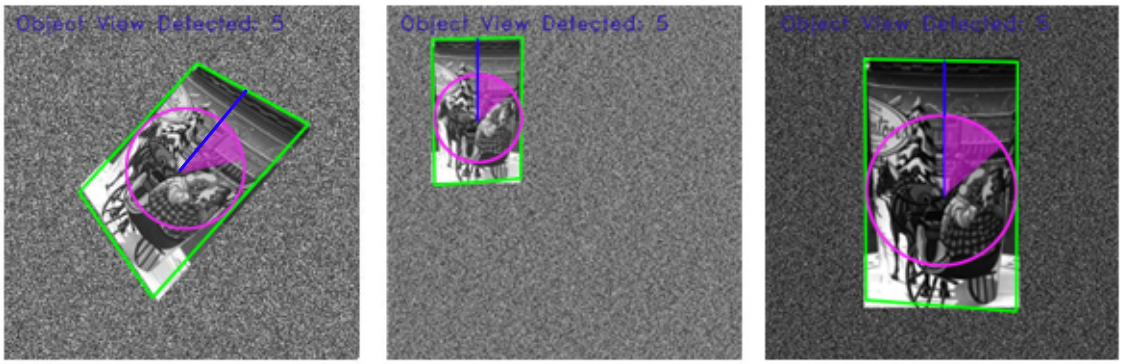


Figure 5.3: Example of an object viewed under different rotations, scales and illuminations.

We also compare Ferns and our NBCD with ORB on the mouse pad example of Ozuysal et al. [104]. Both methods achieve a similar detection rate but ours uses 200x less memory with training that was 14.5x faster (see Table 6). We also compare both approaches using 200 frames of a video of a more complex 3D object (see Figure 5.5). Our detection rate is 88% compared to 72% for Ferns because the planarity assumption in Ferns is violated by 3D objects. Running time for Ferns and NBCD are similar but

Algorithm	Rotation*	Scale×	Illumination+
BRIEF	9%	43%	62%
NBCD + BRIEF	15%	62%	64%
ORB	75%	74%	89%
NBCD + ORB	92%	91%	**90**%
BRISK	79%	75%	82%
NBCD + BRISK	**94**%	**92**%	84%
FREAK	69%	70%	79%
NBCD + FREAK	84%	89%	81%
Ferns	14%	46%	61%
Ferns*	89%	88%	68%
* roll $= [-180°, 180°]$, pitch and yaw $= [-70°, 70°]$			
× scale $= [0.2, 2]$			
+ $\alpha \times I(x, y) + \beta$, $\alpha = [.3, 3]$, $\beta = [-100, 100]$			

Table 5: Comparison of detection rates between matching with only binary descriptors, with Ferns and with our Naive Bayes Classified Descriptors (NBCD) when combined with binary descriptors. The detection rates were tested under random rotation, scale and illumination variations and averaged. The detection rates of Ferns are listed with similar training range to ours and in the line Ferns* with complete training.

Algorithm	Fps	Detection	Memory	Time
Ferns	25.1	**87.4**%	16Mb	348s
NBCD + ORB	**26.5**	86.9%	**80**Kb	**24**s

Table 6: Comparison between Ferns and our framework (i.e., NBCD + ORB) for the mousepad example of Ozuysal et al. [104]. We compare fps (i.e., frame per seconds), detection rate, memory usage and construction time of the model.

we find that our NBCD is independent of keypoint detection and feature extraction (see Figure 5.6). Our framework can therefore take advantage of improvement in speed of new binary descriptors or of hardware implementations.

Figure 5.4: Mousepad example of Ozuysal et al. [104]. Frames 131, 171, 219 are shown: Ferns in first row and our NBCD with ORB in the second row (see Table 6).

5.3.3 Running Time

Binary descriptors use the Hamming distance as a similarity measure because it can be computed efficiently counting bits in parallel [156]. Solutions exist to count the number of ones in a 32 bits integer with less than 15 operations in C. Ferns use a lookup table approach where the S bits (Ferns size) are packed together and used as the index for M lookup values (number of Ferns). As a result Ferns similarity is linear in the number of bits and the number of Ferns. Using S=8 or 16 creates better performance because

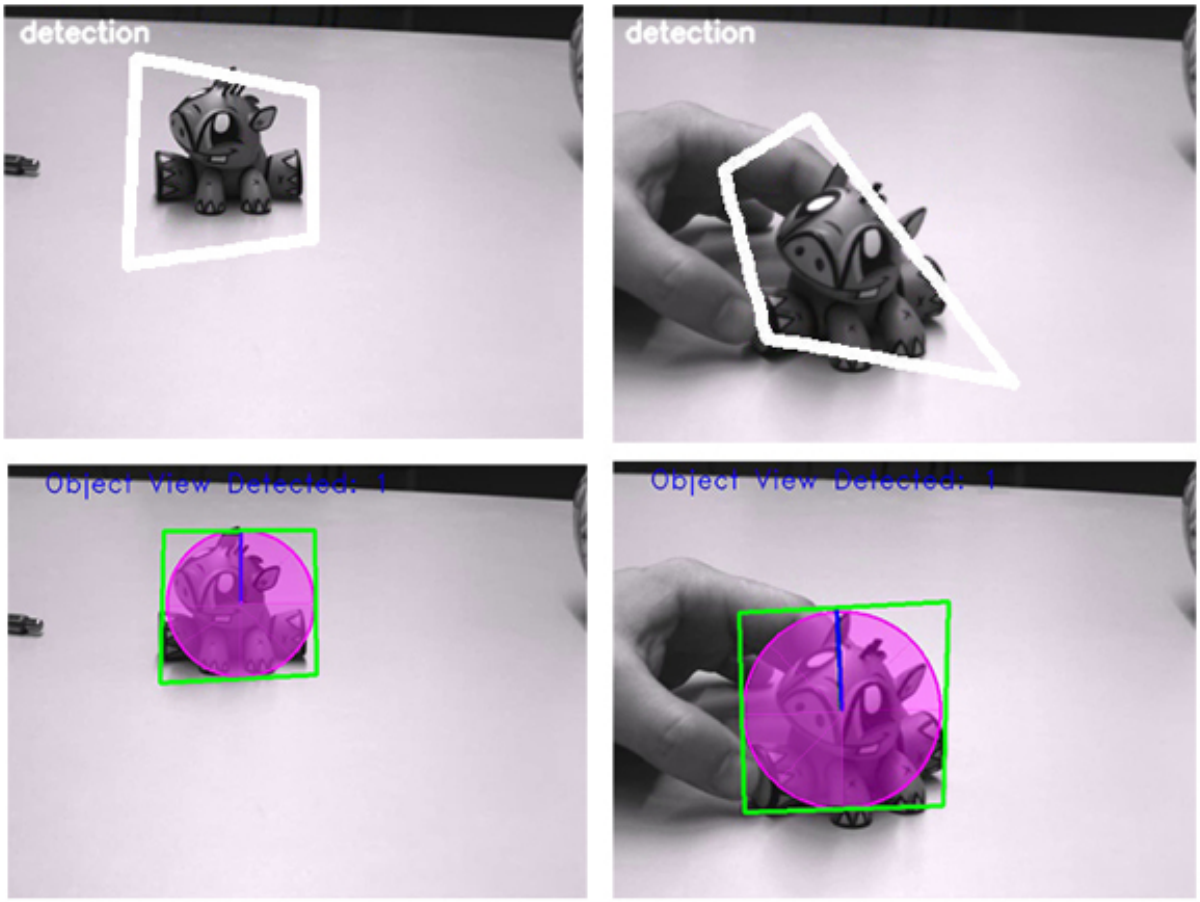


Figure 5.5: The planarity assumption in Ferns does not perform well with more complex non planar objects (e.g., top images). The use of the fundamental matrix is more stable and can be used with planar and non-planar object.

the bits are already grouped like primitive data types. Our keypoint classification is also a lookup table and it depends on the amount of bits of the descriptor, each descriptor bit is translated to a log probability value. Overall they are all linear in terms of the number of bits O(bits). However, our classifier and Ferns are capable of creating a better classification than the Hamming distance similarity measure (see Figure 5.7);

5.3.4 Indexing

Finally we compare pattern-based indexing [141] to our indexing scheme. We find that pattern-based indexing is highly dependent on the keypoint location and orientation. We check if the same keypoint from different viewpoints produce the same index value

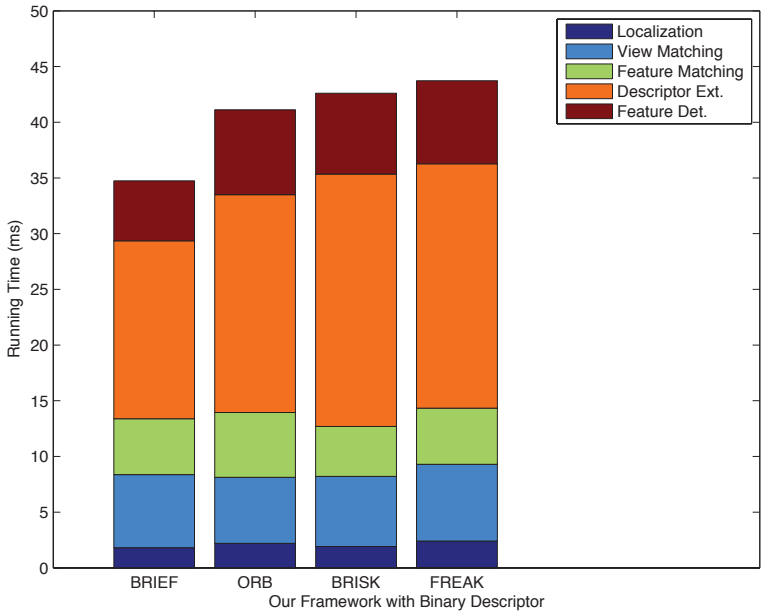


Figure 5.6: Running time of our framework using different binary descriptors. The running time is separated by Keypoint detection, Descriptor extraction, Keypoint Matching, View Matching, and Localization of the object. Our framework is independent of the keypoint extraction and descriptor extraction selection.

because this is indicative of rotation and scale invariance. In our experiments, we rotate (i.e., from 0 to 180) and scale (i.e., from 0.3 to 2) the object's views and compute the keypoint matches between the original and the transformed views. Matches are all those points that have more than 90% of descriptors bits with the same value. Finally, we compute their index value in both locations and compute the repetition rate out of all matches in Figure 5.8. In our index scheme (IS), we use a 13-bit index of randomly selected bits from the descriptor. For the pattern-based, we extracted a 5-bit (i.e., $P5$) and a 13-bit (i.e., $P13$) indices as described in [141]. In the case of FREAK which does not provide a feature detection; we use a combination of ORB feature detector plus FREAK descriptor to match the keypoints. BRIEF was removed from the comparison because it did not perform well under the transformations. See Figure 5.8.

5.4 Summary and Conclusion

In this chapter we presented a framework for 3D object detection and tracking from an uncalibrated mobile camera at frame rates higher than 24 fps. Our NBCD framework

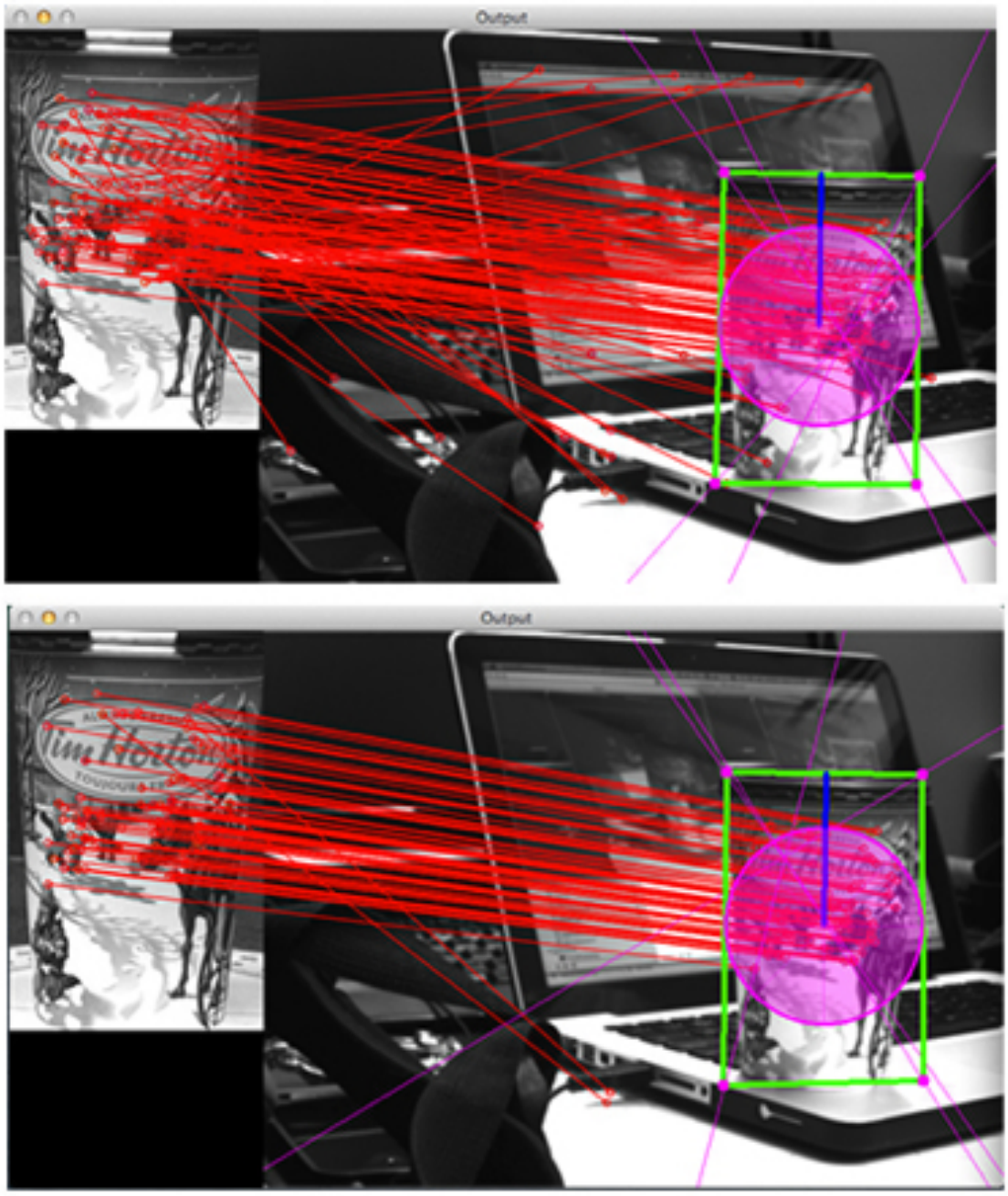


Figure 5.7: Matches between the database view and the video feed. The matches in the top image are using Hamming distance (118 matches: 92 inliers and 26 outliers). The bottom shows our classifier (84 matches: 82 inliers and 2 outliers). All matches in both images have the same Hamming distance. Ferns show similar results to ours.

achieves high recognition rates and can take advantage of the properties of the used keypoint detector and binary descriptor. We introduce an improved indexing scheme

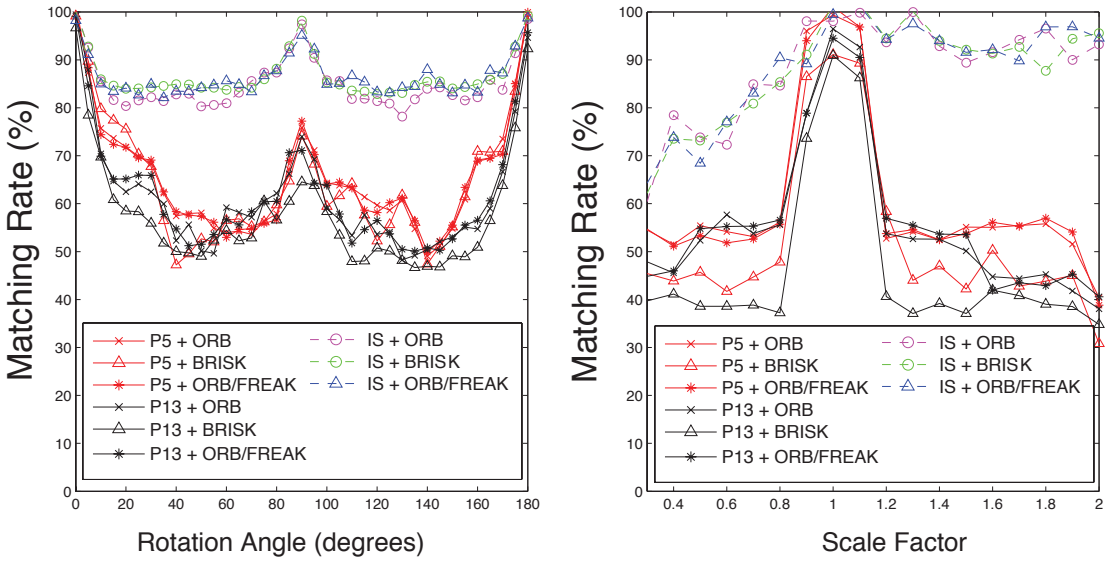


Figure 5.8: The diagrams show the index matching rates under different rotation and scale transformations. We compare pattern-based index schemes (i.e., $P5$ and $P13$) and our proposed index scheme (IS) using binary descriptors. The matching rate shows the repetitive rate indices from the matched keypoints.

for speeding up the keypoint classification that is rotation and scale invariant. NBCD successfully combines binary descriptors and classifiers for keypoint classification, leading to increased performance as compared to binary descriptors by themselves and to reduced memory footprint in comparison to Ferns classifiers. Unlike Ferns, the framework detects a non-planar 3D object from different views. As a result, 3D objects are detected even if they are partially occluded and in different orientations. Experimental results confirm the effectiveness of the proposed solution.

Chapter 6

Scalable Kernel Correlation Filter with Sparse Feature Integration for Long-term Tracking

In this chapter, we present a fast Scalable Kernel Correlation Filter with Sparse Feature Integration (sKCF) solution based on the KCF framework. We introduce an adjustable Gaussian window function and keypoint-based model for scale estimation to deal with the fixed size limitation in the Kernelized Correlation Filter. Furthermore, we integrate the fast HOG descriptors and Intel's Complex Conjugate Symmetric (CCS) packed format to boost achievable frame rates. We test our solution using the Visual Tracker Benchmark and the VOT Challenge datasets. We evaluate our tracker in terms of precision and success rate, accuracy, robustness and speed. The empirical evaluations demonstrate clear improvements by the proposed tracker over the KCF algorithm while ranking among the top state-of-the-art trackers.

6.1 KCF Algorithm

In this section, we first review the KCF[52] algorithm (See Alg. 1). Second, we discuss the introduction of Gaussian window filtering to allow the correlation tracker react to scale changes. Furthermore, we introduce the scale estimation used in our approach and how to integrate it into the KCF pipeline (See Alg. 2). Finally, we describe our additions to increase the performance of the original approach.

The key innovation of KCF is the use of the structure of circulant matrices to enhance the discriminative ability of the track-by-detection scheme. The algorithm proceed as follows: Given the initial selection of a target (i.e., center position and size), a tracked region is created. The tracked region is increased from the target size to provide some context. Features (either raw pixels or other feature channels) are extracted from the tracked region and each channel is weighted by a cosine window. A circulant matrix is used to learn all the possible shifts of the target from a base sample. The coefficient α_f encodes the training samples, consisting of all shifts of a base sample in the frequency domain. The fast learning equation is expressed as

$$\alpha_f = \frac{y_f}{k_f^{xx'} + \lambda} \tag{6.1}$$

where $y_f = Fy$ denotes the Discrete Fourier Transform (DFT) of y. The term $k_f^{xx'}$ denotes the DFT of $k^{xx'}$; the kernel correlation function between signals x and x'. The division represent element-wise division and the scalar λ is a regularization term. The training label matrix y is a Gaussian function that smoothly decays from the value of one for the centered target to zero for other shifts.

For multiple channel features, the vector x concatenates the individual vectors of c channels, e.g., the 31 gradient orientation bins for the HOG descriptor [52] as $x = [x_1, x_2, ..., x_c]$. The Gaussian kernel correlation function $k^{xx'}$ is defined as

$$k^{xx'} = exp\left(-\frac{1}{\sigma^2}\left(\|x\|^2 + \|x'\|^2 - 2F^{-1}\left(\sum_c x_f^c \odot (x_f'^c)^*\right)\right)\right) \tag{6.2}$$

where $\odot$ represent element-wise multiplication and $(x_f^c)^*$ the complex-conjugate of x_f^c (see lines 9, and 5).

The displacement $\delta = \underset{loc}{\text{argmax}}(r(k_f^{z\tilde{x}}))$ between the current and the next patch z is the spatial index with maximum response in $r(k_f^{z\tilde{x}})$. The response is computed as the element-wise multiplication between the learnt alphas $\tilde{\alpha}_f$ and the correlation of z_f with

the learnt model $\tilde{x}_f$. The detection response for each location is

$$r(k_f^{z\tilde{x}}) = F^{-1}(k_f^{z\tilde{x}} \odot \tilde{\alpha}_f) \tag{6.3}$$

The learnt model $\tilde{x}_f$ and alpha $\tilde{\alpha}_f$ are linear interpolations of x_f and α_f at each detection with the selection of an interpolation factor $factor \in [0, 1]$.

A detailed description of KCF algorithm pipeline in pseudocode follows. For the more detailed formulation, please refer to [52].

Algorithm 1 : KCF.

Variables with subscript f are in the frequency domain. Circled operators represent element-wise operations (i.e., $\odot$ and $\oslash$).

- w_sz: size of the tracked region, (W×H).
- pos: center location of the tracker in spatial domain.
- $patch$: region of img centered at pos with size w_sz, (W×H×C).
- $features(x)$: extracted features (e.g., HOG), (m×n×c).
- cos_window: cosine window weights each feature channel, (m×n×1).

Input: *sequence* of images and the tracker's initial position and size (i.e., pos and w_sz).

Ouput: list of positions pos for each frame in the *sequence*.

for each img **in** *sequence*:
  if not first image:
    $patch \leftarrow \text{region}(img, pos, w\_sz)$
    $z_f \leftarrow F(\text{features}(patch) \odot cos\_window)$
    $k_f^{z\tilde{x}} \leftarrow F(\text{correlation}(z_f, \tilde{x}_f))$ ▷ Eq. (6.2)
    $pos \leftarrow pos + \underset{loc}{\text{argmax}}(r(k_f^{z\tilde{x}}))$ ▷ Eq. (6.3)
  $patch \leftarrow \text{region}(img, pos, w\_sz)$
  $x_f \leftarrow F(\text{features}(patch) \odot cos\_window)$
  $k_f^{xx} \leftarrow F(\text{correlation}(x_f, x_f))$ ▷ Eq. (6.2)
  $\alpha_f \leftarrow y_f \oslash (k_f^{xx} + \lambda)$ ▷ Eq. (6.1)
  if first image: $f \leftarrow 1$ **else** $f \leftarrow factor$
  $\tilde{\alpha}_f \leftarrow f \times \alpha_f + (1 - f) \times \tilde{\alpha}_f$
  $\tilde{x}_f \leftarrow f \times x_f + (1 - f) \times \tilde{x}_f$

6.2 Adjustable Window Filtering

In image processing, the process of multiplying an image with a smoothly ending function that gradually reduces its values near the boundaries is called "windowing". The multiplication creates a windowed view of the input signal where the overlap is the signal of interest and reducing to zero the rest. The purpose of windowing is usually to isolate the signal of interest while reducing the frequency leakage in the DFT calculations. Separating the signal from the background is of vital importance for detection. Frequency leakage takes place when the frequency spectrum of a measured input has other frequencies than those of the original signal. Adjustable windows are functions that are capable of reducing the frequency leakage while controlling the bandwidth. They allow controlling how much information of the original signal we want to analyze. Gaussian windows have been widely used for window filtering because of their simple formulation and their benefits in reducing leakage and adjusting to different signal size [112, 102].

In KCF, the target size is strongly linked to the tracked region (i.e., the search area) because of the cosine window. Target scale changes will alter the signal to process and affect the learning process. If the target gets smaller, the cosine window will merge the target signal with the background. If the target gets bigger it discards information of the target and only processes a subset of it. Another issue linked to this behaviour is that we cannot change the tracked region (e.g., we would like to increase/decrease the search area) given the target size.

To overcome this limitation in KCF we replaced their cosine window with a Gaussian window G to allow for target changes of scale and a better separation from the background (See Figure 6.1). The Gaussian window allows us to control the bandwidth of the distribution while the cosine function is fixed to the region size. Furthermore, the Fourier transform of a Gaussian is also a Gaussian which ensures the separation between foreground and the background while reducing the frequency leakage. Scaling a cosine window has no such property.

$$G(m, n, \sigma_w, \sigma_h) = g(m, \sigma_w) * g(n, \sigma_h)' \tag{6.4}$$

The function $g(N, \sigma)$ returns a vector of size N computed as follows

$$g(N, \sigma) = \exp\left(-\frac{1}{2}\left(\frac{i - (N-1)/2}{\sigma(N-1)/2}\right)^2\right), \quad 0 \leq i \leq N. \tag{6.5}$$

The resulting two-dimensional Gaussian window is a matrix $gauss$ that has size $m \times n$.

Note that the $W \times H \times C$ tracked region, is filtered in feature space, $(m \times n \times c)$. For example, when using raw pixels as features their values match, i.e, $m = W, n = H$, and $c = C$. In the case of HOG features with a cell size of $4x4$ and 9 orientation bins, their correspondent values are $m = W/4, n = H/4$, and $c = 31$.

The bandwidth σ of the Gaussian function $g(N, \sigma)$ is computed independently for the horizontal and vertical orientations. The values of σ are selected as the ratio between the feature dimensions and the target dimensions for the horizontal and vertical orientations (i.e., $\sigma_w = \frac{m}{w}$ and $\sigma_h = \frac{n}{h}$). The replaced cosine window from KCF is computed as $cos\_window = h(m) * h(n)'$; $h(x)$ is the Hann windowing function

$$h(N) = \frac{1}{2}\left(1 - cos\left(2\pi\frac{i}{N}\right)\right), \quad 0 \leq i \leq N. \tag{6.6}$$

Figure 6.1: Gaussian and cosine window filtering raw pixel value example. The tracked region has a size of (175×113). First column: same regions with targets at three different scales (i.e., small (56×32), medium (110×61), large (164×83). Second column: Gaussian windows according the target size (i.e., small $(\sigma_w = .32, \sigma_h = .28)$, medium $(\sigma_w = .63, \sigma_h = .54)$, large $(\sigma_w = .94, \sigma_h = .73)$. Third column: cosine window for all the three examples (i.e, size (175×113)). Fourth and fifth columns: images filtered with Gaussian and cosine windows respectively. Figure shows how the fixed cosine window fails to represent the target compared to the Gaussian windows. The cosine window includes background for small targets and discards information for big targets.

6.3 Scale Estimation

With the adjustable window function the algorithm is capable of adjusting to changes in scale, we need to define how to estimate the new dimension of the target at each frame. The change of scale will be used to update the target size, which in turn, will update the σ values for the Gaussian window, the regression labels and the learnt model of the correlation filter. For the estimation, we implemented a keypoint-based strategy where the most interesting points inside the target area are extracted $K^{p1} = K_1^{p1}, K_2^{p1}, ..., K_N^{p1}$ and tracked to the next region (i.e., $K^{p2} = K_1^{p2}, K_2^{p2}, ...K_N^{p2}$). Pairs of tracked points are used to estimate the change of scale (i.e., $pairs = (K_1^{p1}, K_1^{p2}), (K_2^{p1}, K_2^{p2}), ..., (K_T^{p1}, K_T^{p2})$, with $T \leq N$). As in the TLD tracker, we use a forwards-backwards optical flow strategy where we keep the tracked points with high confidence value [57]. The differences between TLD and our approach are that we extract relevant keypoints instead of grid points and estimate the scale using a different metric. TLD uses the median value of all scale ratios between matched points to estimate the scale variation and pairs of points contribute equally to the final estimation. When TLD selects the grid-points inside the target area, they assume that the points belong to the object. During the tracking process this assumption is generally not correct. The rectangular representation of the target might not follow exactly the geometry of the target. To account for this fact, we assign different weights to the keypoints. We assume that points near the center of the target will be more likely to be part of the object (i.e., they should have greater weight in the scale estimation) than those near the boundaries (i.e., weight lower). This assumption is a generalization and does not hold all the time. Some keypoints might not be part of the tracking object but still be near the center. If that is the case, the scale estimation will introduce errors because of incorrectly selecting background points as target points. Nevertheless, the proposed method seems to be a good trade-off for different kind of objects. It is also possible for the user to incorporate a different scale estimation approach if the proposed one does not fit the selected target.

An extracted keypoint K_i^{p1} and its tracked position K_i^{p2} will have an associated weight w_i corresponding to the response of the extracted keypoint K_i^{p1} in a Gaussian window centered at the target area. The horizontal and vertical bandwidths of the Gaussian function are computed as in the previous section; relative to the target size. The variation of scale is computed as the weighted arithmetic mean of the ratio between all possible pairs of extracted and tracked points between next patch $p2$ and the current one $p1$.

$$scale(K^{p1}, K^{p2}) = \frac{\sum_i^T \sum_j^T w_i w_j \times \frac{\left\|K_i^{p2} - K_j^{p2}\right\|^2}{\left\|K_i^{p1} - K_j^{p1}\right\|^2}}{\sum_i^T \sum_j^T w_i w_j}. \tag{6.7}$$

where i and j are the indices of successfully tracked points T. The weights w_i and w_j are the responses of the extracted points.

The proposed sKCF algorithm can be seen in Alg. 2. Note that the scale estimation step is independent and does not affect the correlation filter. This allow us to select a different kernel correlation function (i.e., polynomial or linear) or a different scale estimation strategy without affecting the sKCF pipeline.

Algorithm 2 : sKCF.

Changes to the KCF pipeline are shown in different color.

- w_sz: size of the tracked region, (W×H).
- t_sz: size of the target, (w×h).
- $features(x)$: extracted features (e.g., HOG), (m×n×c).

Input: *sequence* of images and the tracker's initial position and size (i.e., *pos* and t_sz).

Ouput: list of positions *pos* and sizes t_sz for each frame in the *sequence*.

for each *img* **in** *sequence*:
  if not first image:
    $p2 \leftarrow \text{region}(img, pos, w\_sz)$
    $z_f \leftarrow F(\text{features}(p2) \odot cos\_window)$
    $k_f^{z\tilde{x}} \leftarrow F(\text{correlation}(z_f, \tilde{x}_f))$ ▷ Eq. (6.2)
    $pos \leftarrow pos + \underset{loc}{\text{argmax}}(r(k_f^{z\tilde{x}}))$ ▷ Eq. (6.3)
    $t\_sz \leftarrow t\_sz * \text{scale}(K^{p1}, K^{p2})$ ▷ Eq. (6.7)
  $p1 \leftarrow \text{region}(img, pos, w\_sz)$
  $gauss \leftarrow \text{G}(m, n, t\_sz, w\_sz)$ ▷ Eq. (6.4, 6.5)
  $x_f \leftarrow F(\text{features}(p1) \odot gauss)$
  $k_f^{xx} \leftarrow F(\text{correlation}(x_f, x_f))$ ▷ Eq. (6.2)
  $\alpha_f \leftarrow y_f \oslash (k_f^{xx} + \lambda)$ ▷ Eq. (6.1)
  if first image: $f \leftarrow 1$ **else** $f \leftarrow factor$
  $\tilde{\alpha}_f \leftarrow f \times \alpha_f + (1 - f) \times \tilde{\alpha}_f$
  $\tilde{x}_f \leftarrow f \times x_f + (1 - f) \times \tilde{x}_f$

6.4 Improving Algorithm Run-time

The overall complexity of the correlation filter is determined by the DFT/IDFT calls. As the algorithm only requires element-wise operations for the fast learning and the detection, the computational cost is $O(n \log n)$ where n is the number of pixels in the tracked region. Other parts of the code that can affect the performance of the algorithm are the scale estimation and the feature extraction, i.e., the use of raw pixels values demands no extra computation in the pipeline, but stronger and robust representations such as HOG descriptors can affect overall performance. Therefore, we include the following modifications to the original KCF algorithm to improve its run-time.

When performing the spectral analysis of an image, the original data is usually padded to get a bigger image that can be transformed much faster then the original. Images whose size is a power of two are the fastest to process. Nevertheless, data whose size can be represented as $S = 2^p \times 3^q \times 5^r$, for some integer values of p, q, r are also processed quite efficiently. Unfortunately, selecting the optimal input size for DFT/IDFT might be not desirable because the exponential growth of 2^p. For example, the spectral analysis of an image with size (300×300) could be efficiently computed (i.e., $300 = 5^2 \times 3 \times 2^2$) while using the optimal power of two selection, will end up computing the spectrum in an image of almost double its original size (i.e., 512×512). We, utilize OpenCV $getOptimalDFTSize$ who computes the minimum number S that is greater or equal to the original dimensions. We take advantage of this fact, because of our Gaussian window function. The filtered data is not affected by the size of the tracked region like in KCF. The bandwidth of Gaussian distribution is controlled by the target size. This small modification can not be adapted directly to the KCF pipeline because of its cosine window; increasing the search area will fail to correctly filter the target data.

Intel's CCS packed format exploits the symmetrical properties of the Fourier spectrum to efficiently encode the full spectrum of the frequency analysis. The complete spectrum is usually encoded into two matrices of floating precision with the same dimensions of the input data (i.e., for the real and imaginary part). CCS format interlaces the first half of the real and the imaginary spectrum data into one matrix of floating precision. This can be achieved because of its symmetrical representation. Introducing the CCS format into the DFT/IDFT calls effectively reduces the computation time and memory footprint by half. OpenCV only provides CCS data manipulation for their DFT/IDFT calls. We extended the OpenCV functionality to implement the element-wise division of the learning Formula 6.1 using the CCS packed format.

For the feature extraction, we introduce the fast HOG descriptor by Felzenszwalb et al. [39] in their work on discriminatively trained deformable part models. The implementation uses SIMD (SSE) to perform same operations on multiple data points simultaneously, exploiting data level parallelism for the HOG descriptor. This implementation gives nearly identical results to the original HOG descriptors while speeding up the algorithm 4x [39]. The algorithm is capable of processing a (640×480) image in $24ms$ on a Core i5 (2415M). The original algorithm was implemented for the Matlab environment and process data in column-major data format. OpenCV uses a row-major data format so we adapted the algorithm to be used in the correct data format.

Finally, in scale estimation the Lucas-Kanade [160] optical flow for a sparse set of features dominates run-time. To accelerate this process we use initial estimations by introducing the location results from the correlation filter to the tracked points. Also, we only need to compute the optical flow in the tracked region instead of the complete image frame. We drop the pyramidal approach under the assumption of small displacements. The overall complexity is then expressed as $O(kn)$, where k is the number of extracted features and n the size of the processed images. The number of keypoints k is considerable lower than n which in turn is bounded by the tracked region (i.e., search area).

6.5 Experiments

We conduct the evaluation of the proposed algorithm in three experiments. First, we compare the speed and accuracy performance using the 50 videos from the Visual Tracker Benchmark [158] as in the KCF paper [52]. Second, we evaluate and rank the trackers using the VOT Challenge [63]. Finally, we compare our tracker against state-of-the-art trackers to show the effectiveness of sKCF. We report the detailed evaluation on the VOT 2015 dataset.

6.5.1 Experimental Setup and Methodology

We developed three C++ implementations to compare against the original KCF algorithm (which is available as a Matlab implementation only). First, we implemented the KCF algorithm using Gaussian correlation plus integrating Gaussian windowing and fast HOG descriptors. This implementation uses the full Fourier spectrum for the DFT/IDFT calls (i.e., no CCS). We call it KCF_c. Second, we integrated CCS packed format to the previous implementation (i.e., KCF_{ccs}). Finally, we integrated scale estimation to the previous variation (i.e., sKCF). All experiments are conducted on an Intel i5 (2415M)

at 2.3GHz with 8GB of memory. We compared the three implementations to the original Matlab KCF implementation using HOG descriptors. We chosen the same parameter configurations for all implementations as described in [52]. The interpolation $factor = .02$, Gaussian kernel correlation $\sigma = .5$, regulation term $\lambda = 1e-4$, a HOG cell size of 4×4 and 9 orientations bins.

For the Visual Tracker Benchmark [158] dataset, two evaluation criteria are used (i.e., precision and success). Precision is the percentage of frames in a sequence the tracker output is under a certain location error value. The location error is defined as the distance between the center of tracked results and the ground truth. The overall precision is defined as the mean precision for all location errors; the higher the more accurate the result. Success is the percentage of frames the tracker output is under a certain overlap ratio. The overlap ratio is defined as $VOC = \frac{Area(B_T \cap B_G)}{Area(B_T \cup B_G)}$, where B_T is the tracked bounding box, and B_G is the ground truth bounding box. The overlap ratio values go from 0 to 1. The overall success is defined as the mean success for all overlap ratio values.

In the VOT Challenge, two evaluation criteria are used (i.e., accuracy and robustness). Accuracy is measured as the VOR score. The robustness indicates the number of failures to track an object in a sequence. A failure is determined when the VOR score goes to zero.

6.5.2 Experiment 1: Visual Tracker Benchmark

We evaluated the performance of our three implementations (i.e., KCF$_c$, KCF$_{ccs}$, and sKCF) against KCF using the Visual Tracker Benchmark [158] as in the KCF paper [52]. We computed the overall precision and success rates for the 51 video dataset presented by Wu et al. [158].

The introduction of the Gaussian window in KCF$_c$ and KCF$_{ccs}$ resulted in increased precision and success rates over the KCF tracker (see Figure 6.2). KCF$_c$ and KCF$_{ccs}$ displayed similar performance validating the idea that the CCS packed format does not affect the accuracy while speeding the computations. The sKCF tracker produces better results than all the trackers by introducing the scale estimation into the pipeline. Furthermore, sKCF displayed an average speed-up of 2.2x for all the sequences compared to the KCF implementation. For this experiment, we included the performances of the top three ranked trackers in Wu et al. [158] (i.e., SCM, Struck, and TLD) for completeness.

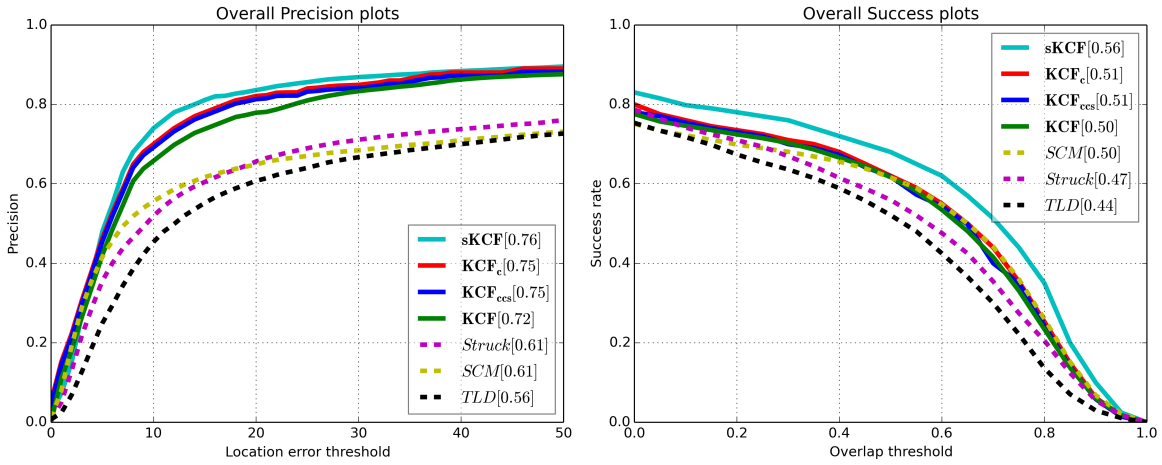


Figure 6.2: Precision and success plots for our three implementations (i.e., sKCF, KCF$_c$, KCF$_{ccs}$), KCF, and the top three ranked trackers in the Visual Tracker Benchmark [158] dataset.

6.5.3 Experiment 2: VOT Challenge

We continue to evaluate the trackers in terms of accuracy and robustness. For this experiment we used the VOT 2015 and VOT TIR 2015 datasets (i.e., 60 and 20 video sequences respectively). Along with sKCF, KCF$_c$, KCF$_{ccs}$ and KCF, we included the NCC (Fast Normalized Cross-Correlation) implementation provided in the VOT packages for evaluation purposes. Our three implementations produce superior results than KCF and NCC trackers in the overall ranking output for both challenges. In terms of speed, NCC was the faster tracker (i.e., higher frame per seconds) followed by KCF$_{ccs}$ and sKCF. The proposed sKCF algorithm showed the higher overall accuracy values (i.e., mean accuracy value from all the sequences) and a lower failure rate for the VOT 2015 dataset. In the VOT TIR dataset, NCC displayed the best accuracy values from all the implementations but with higher failure rate which translated into the lowest overall rank from all the implementations. KCF$_c$ and KCF$_{ccs}$ had the same results positioning it above the KCF tracker but at different speeds.

Table 7 shows the results obtained from the five trackers in the VOT and TIR 2015 datasets. The tables display the overall average accuracy and average number of failures in all the sequences, with their accuracy, robustness and overall ranks. The tables also display the average tracker speeds given in frame per seconds (i.e., fps).

Finally, we evaluate our proposed sKCF tracker on the VOT 2014 dataset (i.e., 25 sequences). Table 8 summarizes the top three ranked trackers. Our algorithm is the

Table 7: VOT and VOT TIR 2015 Results

VOT 2015						
	Overall		Rank			
	Acc.	Fail.	Acc.	Rob.	Overall	Speed(fps)
sKCF	**0.50**	**2.49**	**2.22**	**2.60**	**2.41**	64.5
KCF_c	0.49	2.58	3.19	2.64	2.92	49.8
KCF_{ccs}	0.49	2.58	3.19	2.64	2.92	71.2
KCF	0.47	2.61	3.29	2.68	2.99	24.4
NCC	0.48	11.34	3.18	4.43	3.81	**78.5**
VOT TIR 2015						
sKCF	0.58	**5.28**	2.92	**2.50**	**2.71**	215.0
KCF_c	0.57	5.40	3.32	2.52	2.92	180.8
KCF_{ccs}	0.57	5.40	3.32	2.52	2.92	216.3
KCF	0.56	5.66	3.40	2.65	3.02	94.8
NCC	**0.65**	9.52	**1.98**	4.80	3.39	**262.3**

fastest of the top trackers with competitive high accuracy and low failure rates. We achieved similar results to the two variations of the original KCF that incorporate scale support (i.e., SAMF and KCF*). The KCF* tracker improves the original KCF by adding a multi-scale support, sub-cell peak estimation and replacing the model update scheme. We couldn't find details of this implementation nor publication with details of the improvements. The KCF* entry on the VOT 2014 submission only refers to the original KCF paper [52]. The last entry in Table 8 shows the KCF Matlab's code [52] results on the VOT 2014 challenge.

Table 8: VOT 2014 Results

	Overall		Rank			
	Acc.	Fail.	Acc.	Rob.	Overall	Speed (fps)
DSST	0.65	**16.90**	5.44	**12.17**	**8.81**	5.8
SAMF	0.65	19.23	5.23	12.94	9.09	1.6
KCF*	**0.66**	19.79	**5.16**	13.55	9.36	24.2
sKCF	0.61	18.44	7.68	13.14	10.41	**65.4**
KCF	0.56	27.14	13.14	18.02	15.58	20.3

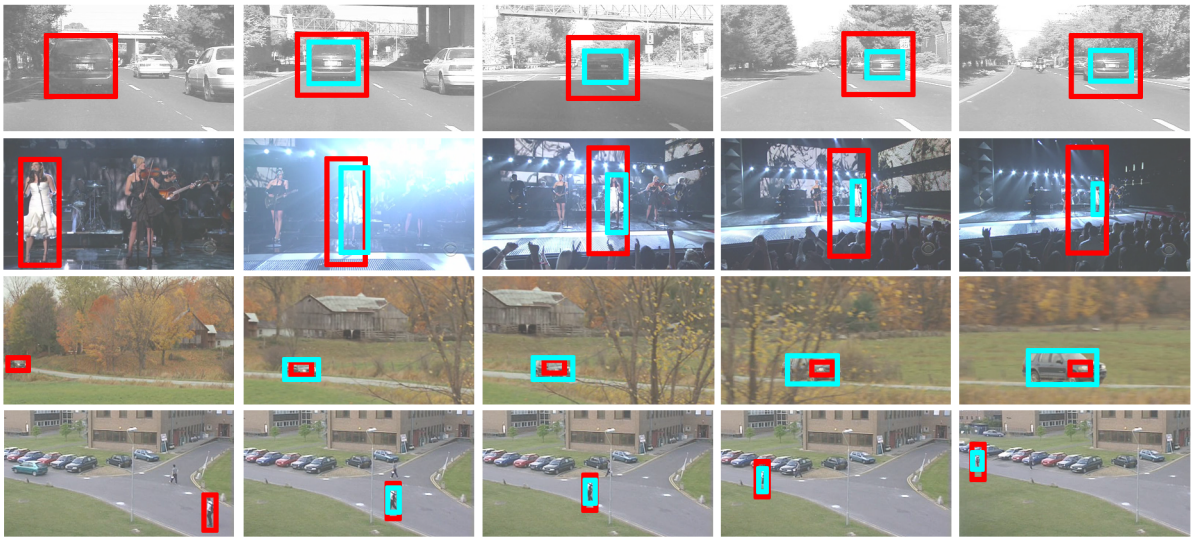


Figure 6.3: Some sequence examples with scale changes and the responses from KCF (i.e., red boxes) and sKCF (i.e., light blue boxes). The first column represents the initial selection and the size of the tracker at t he first frame in the sequence.

6.6 Summary and Conclusion

In this chapter we presented a scalable kernel correlation filter with sparse feature integration for long-term tracking object tracking. Our solution achieved up to 2x speedup of the original correlation filter while producing superior results in precision, success, accuracy and robustness.

We introduce Gaussian window filtering that overcomes the fixed-size limitation of the KCF tracker. The modifications allow the KCF framework to adapt to target changes in scale while creating a better separation between the target and the background. The keypoint-based scale estimation is independent of the correlation filter which can be integrated with any variation of Gaussian, polynomial, or linear correlations presented in KCF [52]. We boosted the speed performance of the tracker by introducing fast HOG descriptors and Intel's CCS packed format. Our solution ranked among the top state-of-the-art trackers under the different tracking challenges presented in the three tracking datasets. Experimental evaluation results under the Visual Tracker Benchmark and VOT Challenges confirm the effectiveness of the proposed solution.

Chapter 7

Conclusions

During the work presented in this thesis, we pursued two main goals. First, we extracted certain features that best characterized a given object. The features presented strong properties, which allowed us to detect the object and describe it. Second, given some knowledge of how a certain object may appear, we identified, localized and tracked the object given an image sequence. To pursue these goals, we created four solutions where we covered different kinds of objects. The solutions consisted of: skeleton pruning for shape analysis, planar object detection for localization, 3D object detection using an uncalibrated camera, and long-term tracking 3D object tracking. We started with binary shapes, followed by textured 2D planar objects, and finally 3D objects.

7.1 Summary

The purpose of the first sub-goal, presented in Section 1.2.1, is to compute a clean skeleton representation of a noisy image which conserves the geometrical and topological properties of the shape. The solution must handle simple and complex shapes (i.e., with holes as part of their morphology) without affecting connectivity. We present a novel skeleton pruning solution that uses the integer medial axis (IMA) transform and contour approx-

imation of the binary shape in Chapter 3. The shape's boundary is approximated by a piecewise curve, which helps the process of removing unwanted branches from the IMA. The novel pruning algorithm effectively removes unwanted branches without affecting the geometric and topological properties of the skeleton's curve. In our experimental validation, our pruning solution outperforms existing state of the art methods as discussed in Section 2.1. Our results are more stable even if the shape undergoes a rigid transformation or boundary noise is added (i.e., significant changes to the shape do no alter the final skeleton result). Our solution is suitable for automatic real-time applications and outputs a subset of the IMA (i.e., similar shapes have similar skeleton representations). Finally, an automatic vector representation of the skeleton is created for compact shape description. The vector representation is created ensuring that the curve is preserved inside the shape's boundary. Another highlight to our pruning solution is the possibility of high accuracy reconstruction of the original shape from the pruned skeleton and the distance map. In this sub-goal we accomplished the entire set of requirements for our pruning solution while outperforming the other solutions discussed in our related work. Our pruning solution stands among the fastest algorithms with robust skeleton computations [109].

For our second sub-goal 1.2.2, we addressed the problem of developing visual localization using planar natural landmarks. In Chapter 4, we presented a keypoint-based framework for localization using textured 2D planar objects. We utilized Fern classifiers to learn the planar objects because of the off-line learning mode and its fast computation. Along with the planarity object detection model using Ferns we introduced a set of improvements to the localization problem. These improvements are stratified sampling, on-line checking of false target detection, use of all matching points to improve pose estimation and an off-line target evaluation strategy as presented in Section 4.2. Our experimental results illustrate that natural planar targets and Fern classifiers with our additions are a feasible solution for visual localization. The stratified sampling technique is able to double the true rate of detection while reducing false target detection without affecting execution time. On-line checking of the detection of a planar target helped reducing false positives. The use of all matching points vs. simply the corners of a rectangular target improved the accuracy of localization by up to 50%. Finally, off-line target evaluation predicts the quality of the target and hence, the precision to expect when the particular target would be employed as part of a map. Although the experimental evaluation is realized with the Fern classifiers, the framework is independent of

the classifier (i.e., different techniques could replace Ferns). In this sub-goal we successfully integrated the use of textured natural planar landmarks to solve the localization problem. The presented framework was demonstrated to be robust against illumination changes, perspective distortions, motion blur and occlusions. Our framework recorded an average 3 ± 0.66 (95% Confidence Interval) centimetre translation error and an average 4 ± 0.52 (95% Confidence Interval) degree rotation error in ranges up to 3 meters using target dimensions of 27x20 centimetres. The framework includes an off-line target evaluation strategy that is capable of determining if the planar object can be used as a natural landmark for the localization problem and evaluates the precision to expect.

For our third sub-goal, we have successfully accomplished a real-time 3D object detection solution independent of camera optics. The solution accounts for a mobile object and it is capable of learning the object from a few images. Another achievement is the low memory consumption allowing the solution to possibly run in an embedded system. In Chapter 5, we presented our 3D object detection framework for uncalibrated cameras. We combined binary descriptors with a naive Bayes classifier to improve the learning, matching time and classification process. To our knowledge we are the first ones to combine both approaches (i.e., binary descriptors and random classifiers) for feature classification and matching. The new classifier (NBCD) exploits the specific structure of various binary descriptors in order to increase feature matching while conserving descriptor properties (e.g., rotational and scale invariance, robustness to illumination changes and real-time performance). Thus, we achieved equivalent performance to Fern classifiers; at the same time, we boosted the speed and reduced the memory usage from exponential to linear of the number of binary features. The framework is capable of frame rates higher than 24fps while using less than 1MB for the object model. We introduced an improved indexing scheme that is rotation and scale invariant for speeding up the keypoint classification. The objects are detected even if they are partially occluded and in different orientations. Thanks to the use of binary descriptor with invariant properties in our NBCD classifier, the time for the off-line learning phase is reduced considerably compared to state of the art schemes. Our method is currently among the 3D object detection solutions with the smallest memory footprint.

For the final sub-goal, we presented a very effective scale-adaptive tracker based on the Kernel Correlation Filter. We introduced a fast scheme to remove the KCF limitation of fixed template size. Moreover, our Gaussian window creates a better separation of the target and the background improving accuracy. Fast HOG descriptors, variable

template size selection, and the CCS packed format improved the overall run-time. The modifications to the KCF pipeline allows the usage of different kernel correlations as in the original paper [52]. The scale estimation is independent of the framework and a different approach could be adopted with in our framework. The empirical evaluations on the Visual Tracker Benchmark and VOT datasets demonstrate the validity of our algorithm. Our proposed solution ranked among the top state-of-the-art trackers.

7.2 Limitations

We have created robust feature descriptors for shape analysis, object detection and tracking solutions. We have succeeded in creating solutions that are fast and minimize the memory footprint while still achieving competitive results. Empirical evaluations demonstrate the results and benefits of these solutions. We included comparison with state-of-the art solutions to validate the superiority of our proposed solutions. Nevertheless, these research areas are still in need of improvements and are open to new challenges.

From the results described in Chapter 3, we can see that the parameter choice has little influence on the average performance for a tested dataset. However, there is still a correlation between the parameter selection and the pruned skeleton. The calculated skeleton for some shapes could be improved by tuning the selection, in particular, branches due to noisy boundaries may be accepted as part of the final skeleton with some parameters but removed under different ones. There are also scenarios of shapes where the control points from the automatic contour approximation could be improved. Also, the piecewise approximation step is limited to 2D binary shapes while the integer medial axis computation and pruning criteria could be easily adapted for 3D shapes too.

In Chapter 4, our planar landmark object detection uses manually selected targets to aid camera localization. This requires manually inserting multiple targets into the environment. A person needs to manually take a fronto-parallel picture and record the physical dimensions of the target to be used in the detection framework. Dynamic environments present another limitation. If the learned targets are moved from their initial position our system will not account for such changes. *Mapless navigation* systems deal with dynamic environments because they perceive the environment as they navigate through it [16, 34]. They map the environment and rely on the whole scene to account for changes. An automatic process could detect possible planar targets and evaluate them before inserting them into our framework. Online learning of new targets removes the human interaction and adapts to dynamic environments. Planar targets offer a simple

solution to aid robot navigation because of their simplicity and small memory footprint, as opposed to solutions that map the entire environment. However, the framework is limited to planar landmarks and could benefit from including 3D objects.

In Chapter 5, we extended the planar framework by detecting 3D objects. We reduced the memory footprint while increasing the detection rate and speed performance achieved in our planar detection framework. The fast response of the classifier and small memory usage allowed us to learn more views of a single target and efficiently encode them into our 3D detection framework. Only a few images from different views of the target were necessary to learn everyday rigid objects. The use of rotationally invariant descriptors in our classifier allowed us detects the target under different orientations and rotations without learning extra views. However, more geometrically complex objects would need more views to learn all the possible appearances. The framework performance will be affected by the complexity of the object and the amount of views needed to learn it. Our approach is not designed for objects undergoing change, e.g., due to articulation or deformation.

Another limitation is object representation in long-term tracking tracking. Our algorithm in Chapter 6 outputs a polygonal representation of the object while locating it in the frame view. This polygonal representation includes some background information, which is mixed with the target object. Although the representation's simplicity improves speed and simplifies the model of the target, there are applications where a more detailed representation of the object's boundaries is desirable. Complex model representations impact the detection speed, but benefit the overall detection and exact representation of the actual target.

7.3 Future Work

For future work, we are interested in the enhancement and extension of the implementations presented in this thesis. Although we have created fast and robust solutions to address our four sub-goals, there is still room for improvement in all four areas.

We would particularly like to extend our skeleton pruning solution (i.e., first sub-goal) to 3D surfaces. Our pruning solution could be adapted to the 3D scenario by approximating the model's boundary with simpler 3D representations. The integer medial axis can be computed for 3D objects, and our final pruning criteria can also be adapted by using spheres centered at the branch points instead of circles. We would encourage the creation of a benchmark and dataset of shapes and/or models in order to rank the dif-

ferent skeleton computations and pruning techniques. Such evaluation is available for the tracking problem, but no relevant work has been developed for benchmarking skeletonization algorithms. Although the 2D MPEG-7 dataset is available for evaluation, there is no annotation of what could be the expected skeleton representation for each shape.

Another improvement would be to our planar object detection framework such that new targets can be learned online. Currently, a manual selection and evaluation of planar objects is necessary in our framework to add relevant landmarks that can be identified from different views. An automated solution could detect planar surfaces around the robot trajectory and evaluate their suitability as planar landmarks for accurate camera pose estimation. This framework could also be extended to 3D objects by incorporating our research work in 3D object detection.

We would also like to study the creation of a 3D detection benchmark and dataset where multiple views of the initial target are provided a priori. For object tracking, datasets and benchmarks such as the VOT Challenges and the Visual Tracker Benchmark are available to the scientific community. However, there does not seem to have been any relevant work dedicated to 3D object detection and the study of performance measures. Another extension of this work is to learn features online and include them into the classifier when the object is successfully detected. Currently, only an off-line learning step is used to create the object model. Similar solutions such as our sKCF, learns different appearances starting from an initial detection.

Finally, we would like to improve the object output representation for our long-term tracking tracking solutions. Currently, we use bounding boxes to describe the object and output its location and size. Most of the available algorithms for long-term tracking tracking use simple box representation to simplify the process. We believe a more detailed representation that would follow the object edges could be computed by analyzing the background of the currently detected target area. A more detailed representation would benefit the learning process, which in turn can be expected to lead to more precise detection. Another issue related to the long-term tracking tracking is drift in some sequences. We find that scenes where the target is occluded affect the learning model. The continuous interpolation method for learning the frequency response of the target hinders the tracking by detection method. Currently, even if the tracker is occluded its frequency response is included into the model. A more stable learning solution should only include information while the object is not occluded. An approach that detects when

the object is occluded could be used to flag the learning process and avoid including the information into the model. Solutions, such as TLD [57] takes this scenario into account while learning the object.

References

[1] S. S. Akhoury and R. Laganière. Training binary descriptors for improved robustness and efficiency in real-time matching. In *In Proceedings of the International Conference on Image Analysis and Processing (ICIAP)*, volume 8157, pages 288–298. Springer, 2013. (Cited on pages 22, 77, and 78.)

[2] N. Alajlan, I. E. Rube, M. S. Kamel, and G. Freeman. Shape retrieval using triangle-area representation and dynamic space warping. *Pattern Recognition*, 40(7):1911–1920, 2007. (Cited on page 41.)

[3] P. F. Alcantarilla, S. M. Oh, G. L. Mariottini, L. M. Bergasa, and F. Dellaert. Learning visibility of landmarks for vision-based localization. In *IEEE Robotics and Automation Society (ICRA)*, pages 4881–4888. IEEE, 2010. (Cited on page 21.)

[4] C. Arcelli and G. S. di Baja. Euclidean skeleton via centre-of-maximal-disc extraction. *Image and Vision Computing*, 11(3):163 – 173, 1993. (Cited on pages 3, 16, and 17.)

[5] S. Atiya and G. D. Hager. Real-time vision-based robot localization. *IEEE Transactions on Robotics and Automation (TRA)*, 9:785–800, 1993. (Cited on page 20.)

[6] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, August 2011. (Cited on page 6.)

[7] X. Bai, L. Latecki, and W. Liu. Skeleton pruning by contour partitioning with discrete curve evolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(3):449–462, 2007. (Cited on pages 3, 16, 17, 18, 19, 30, 38, and 40.)

[8] G. S. D. Baja and E. Thiel. Skeletonization algorithm running on path-based distance maps. *Image and Vision Computing*, 14(1):47–57, 1996. (Cited on pages 3, 16, and 17.)

[9] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008. (Cited on pages xvii, 2, and 21.)

[10] D. Bekele, M. Teutsch, and T. Schuchert. Evaluation of binary keypoint descriptors. In *Proceedings IEEE International Conference on Image Processing (ICIP)*, pages 3652–3656, Melbourne, Australia, 2013. IEEE. (Cited on page 22.)

[11] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(4):509–522, Apr. 2002. (Cited on page 1.)

[12] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26–33, 2005. (Cited on page 2.)

[13] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the perception of speech and visual form*, pages 362–380. MIT Press, Cambridge, 1967. (Cited on pages 2 and 30.)

[14] V. N. Boddeti, T. Kanade, and B. V. K. V. Kumar. Correlation filters for object alignment. In *In Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 2291–2298. IEEE, 2013. (Cited on page 6.)

[15] F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, 53(3):263–296, 2008. (Cited on page 20.)

[16] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe. Mobile root positioning: Sensors and techniques. *Journal of Robotic Systems*, 14(4):231–249, 1997. (Cited on pages 20 and 107.)

[17] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE, 2007. (Cited on pages 5 and 22.)

[18] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. (Cited on page 67.)

[19] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. (Cited on page 22.)

[20] A. J. Briggs, D. Scharstein, D. Braziunas, C. Dima, and P. Wall. Mobile robot navigation using self-similar landmarks. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2000*, pages 1428–1434. IEEE Computer Society Press, 2000. (Cited on page 4.)

[21] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, Aug. 2007. (Cited on pages 1, 3, 80, and 81.)

[22] P. Buchanan. *Artistic Content Representation and Modelling based on Visual Style Features*. PhD thesis, 2014. Published. (Cited on page 18.)

[23] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: binary robust independent elementary features. In *In Proceedings of the European Conference on Computer Vision (ECCV)*, pages 778–792, Berlin, Heidelberg, 2010. Springer-Verlag. (Cited on pages xiv, 2, 21, and 22.)

[24] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 8(6):679–698, June 1986. (Cited on page 2.)

[25] L. Cehovin, M. Kristan, and A. Leonardis. Is my new tracker really better than yours? In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 540–547, 2014. (Cited on page 27.)

[26] L. Cehovin, A. Leonardis, and M. Kristan. Visual object tracking performance measures revisited. *CoRR*, abs/1502.05803, 2015. (Cited on page 6.)

[27] D. Chekhlov, M. L. Pupilli, W. W. M. Cuevas, and A. D. Calway. Real-time and robust monocular SLAM using predictive multi-resolution descriptors. In *Advances in Visual Computing*, pages II: 276–285, 2006. (Cited on page 21.)

[28] D. J. Crandall, P. F. Felzenszwalb, and D. P. Huttenlocher. Object recognition by combining appearance and geometry. In *Toward Category-Level Object Recognition*, pages 462–482, 2006. (Cited on page 2.)

[29] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 886–893, France, June 2005. IEEE. (Cited on pages xv, 2, and 21.)

[30] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1403–1410, 2003. (Cited on page 21.)

[31] L. de Floriani and M. Spagnuolo, editors. *Shape Analysis and Structuring*. Springer-Verlag New York, Inc., 2007. (Cited on page 2.)

[32] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *IEEE Robotics and Automation Society (ICRA)*, pages 1322–1328, 1999. (Cited on page 20.)

[33] D. G. T. Denison, B. K. Mallick, and A. F. M. Smith. Automatic Bayesian curve fitting. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(2):333–350, 1998. (Cited on page 30.)

[34] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(2):237–267, 2002. (Cited on pages 20 and 107.)

[35] L. Dong, I. Zoghlami, and S. Schwartz. Object tracking in compressed video with confidence measures. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 753–756, July 2006. (Cited on page 5.)

[36] T. W. Drummond and E. D. Eade. Edge landmarks in monocular SLAM. In *In Proceedings of the British Machine Vision Conference (BMVC)*, page I:7, 2006. (Cited on page 21.)

[37] M. V. Eede, D. Mancrini, A. Telea, C. Sminchisescu, and S. Dickinson. Canonical skeletons for shape matching. In *In Proceedings of the International Conference on Pattern Recognition (ICPR)*, volume 2, pages 66–69, 1997. (Cited on pages 1 and 2.)

[38] M. Felsberg, A. S. Montero, J. Lang, and R. L. et al. The thermal infrared visual object tracking vot-tir2015 challenge results. pages 76–88, December 2015. (Cited on page 12.)

[39] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(9):1627–1645, 2010. (Cited on pages 2 and 99.)

[40] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. (Cited on pages 68 and 69.)

[41] A. Frome, F. Sha, Y. Singer, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2007. (Cited on page 1.)

[42] H. K. Galoogahi, T. Sim, and S. Lucey. Multi-channel correlation filters. In *IEEE International Conference on Computer Vision, (ICCV) 2013, Sydney, Australia, December 1-8, 2013*, pages 3072–3079, 2013. (Cited on page 6.)

[43] S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision*, 94(3):335–360, 2011. (Cited on page 57.)

[44] Y. Ge and J. M. Fitzpatrick. On the generation of skeletons from discrete Euclidean distance maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18:1055–1066, 1996. (Cited on page 17.)

[45] J. Giesen, B. Miklos, M. Pauly, and C. Wormser. The scale axis transform. In *In Proceedings of the Symbolic Computation Group (SCG)*, pages 106–115, 2009. (Cited on pages 3, 17, and 40.)

[46] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In D. Forsyth, P. Torr, and A. Zisserman, editors, *European Conference on Computer Vision (ECCV)*, volume 5302 of *Lecture Notes in Computer Science*, pages 234–247. Springer Berlin Heidelberg, 2008. (Cited on page 6.)

[47] Haralick, R. M., and L. G. Shapiro. *Computer and Robot Vision*, volume 1. Addison-Wesley, 1992. (Cited on pages 16, 17, 39, and 40.)

[48] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *IEEE International Conference on Computer Vision (ICCV)*, pages 263–270, Nov 2011. (Cited on pages xvii, 6, and 23.)

[49] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. (Cited on pages 3 and 82.)

[50] J. F. Henriques, R. Caseiro, and J. Batista. Globally optimal solution to multi-object tracking with merged measurements. In D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, editors, *ICCV*, pages 2470–2477. IEEE, 2011. (Cited on page 5.)

[51] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *In Proceedings of the European Conference on Computer Vision - Volume Part IV*, ECCV'12, pages 702–715, Berlin, Heidelberg, 2012. Springer-Verlag. (Cited on pages xiv, 6, and 23.)

[52] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine*

Intelligence (TPAMI), 2015. (Cited on pages xv, 6, 23, 26, 92, 93, 99, 100, 102, 103, and 107.)

[53] W. H. Hesselink and J. B. Roerdink. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(12):2204–2217, 2008. (Cited on pages 3, 16, 17, 19, 30, 35, 39, and 40.)

[54] J. Horst and I. Beichel. A simple algorithm for efficient piecewise linear approximation of space curves. In *In Proceedings of the International Conference on Image Processing (ICIP)*, volume 2, pages 744–747, 1997. (Cited on page 30.)

[55] I. Hwang, H. Balakrishnan, K. Roy, and C. Tomlin. Multiple-target tracking and identity management in clutter for air traffic control. In *In Proceedings of the AACC American Control Conference*, 2004. (Cited on page 5.)

[56] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *IEEE International Conference Pattern Recognition (ICPR)*, pages 2756–2759, Aug 2010. (Cited on page 6.)

[57] Z. Kalal, K. Mikolajczyk, and J. Matas. P-n learning: Bootstrapping binary classifiers by structural constraints. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 49–56, June 2010. (Cited on pages xvii, 6, 23, 24, 96, and 110.)

[58] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(7):1409–1422, 2012. (Cited on pages 1, 3, and 6.)

[59] B. Kégl and A. Krzyźak. Piecewise linear skeletonization using principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(1):59–73, 2002. (Cited on page 2.)

[60] G. Klein and D. W. Murray. Improving the agility of keyframe-based SLAM. In *European Conference on Computer Vision*, pages II: 802–815, 2008. (Cited on page 21.)

[61] T. Kobayashi. Bfo meets hog: Feature extraction based on histograms of oriented p.d.f. gradients for image classification. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 0:747–754, 2013. (Cited on page 3.)

[62] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. (Cited on page 79.)

[63] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. P. Pflugfelder, G. Fernández, G. Nebehay, F. Porikli, and L. Cehovin. A novel performance evaluation methodology for single-target trackers. *CoRR*, abs/1503.01313, 2015. (Cited on pages 6, 23, 26, and 99.)

[64] M. Kristan, A. S. Montero, J. Lang, and R. L. et al. The visual object tracking vot2015 challenge results. pages 1–23, December 2015. (Cited on page 12.)

[65] E. Krotkov. Mobile robot localization using a single image. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 978 – 983, May 1989. (Cited on page 20.)

[66] L. Lam, L. Seong-Whan, and C. Y. Suen. Thinning methodologies - a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(9), 1992. (Cited on pages 3, 16, and 40.)

[67] C. H. Lampert. An efficient divide-and-conquer cascade for nonlinear object detection. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1022–1029. IEEE, 2010. (Cited on page 2.)

[68] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008. (Cited on page 2.)

[69] L. J. Latecki, R. Lakämper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 424–429, 2000. (Cited on page 41.)

[70] L. J. Latecki, Q. nam Li, X. Bai, and W. yu Liu. Skeletonization using SSM of the distance transform. In *Proceedings ICIP*, volume 5, pages 349–352, 2007. (Cited on pages 3 and 16.)

[71] E.-K. Lee, S. Yang, S.-Y. Oh, and M. Gerla. Rf-gps: Rfid assisted localization in vanets. In *IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 621–626. IEEE, 2009. (Cited on page 4.)

[72] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. *IEEE International Conference on Computer Vision (ICCV)*, 0:2548–2555, 2011. (Cited on pages xiv, 21, and 25.)

[73] Y. Li and J. Zhu. A scale adaptive kernel correlation filter tracker with feature integration. In *European Conference on Computer Vision (ECCV)*, volume 8926 of *Lecture Notes in Computer Science*, pages 254–265. Springer Publishing, 2015. (Cited on pages 23 and 26.)

[74] C.-C. Lin and R. L. Tummala. Mobile robot navigation using artificial landmarks. *Journal of Robotic Systems*, 14(2):93–106, 1997. (Cited on page 4.)

[75] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30:465–470, 1996. (Cited on page 2.)

[76] L. Liu, E. W. Chambers, D. Letscher, and T. Ju. Extended grassfire transform on medial axes of 2d shapes. *Computer-Aided Design*, 43(11):1496 – 1505, 2011. (Cited on pages 3, 16, 17, and 18.)

[77] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004. (Cited on pages xvii, 2, 3, and 21.)

[78] J. Martínez, M. Vigo, N. Pla-Garcia, and D. Ayala. Skeleton computation of an image using a geometric approach. In H. P. A. Lensch and S. Seipel, editors, *Eurographics Association*, pages 13 – 16, 2010. (Cited on page 16.)

[79] J. Martínez, N. Pla, and M. Vigo. Skeletal representations of orthogonal shapes. *Graphical Models*, 75(4):189 – 207, 2013. (Cited on page 16.)

[80] N. Mayya and V. Rajan. Voronoi diagrams of polygons: A framework for shape representation. In *Journal of Mathematical Imaging and Vision*, pages 638–643, 1994. (Cited on page 16.)

[81] B. Miklos, J. Giesen, and M. Pauly. Discrete scale axis representations for 3d geometry. *ACM Transactions of Graphics (SIGGRAPH)*, 29:101:1–101:10, July 2010. (Cited on pages 3, 17, and 40.)

[82] K. Mikolajczyk, B. Leibe, and B. Schiele. Local features for object class recognition. In *In Proceedings of the IEEE International Conference on Computer Vision*, volume 2 of *ICCV '05*, pages 1792–1799, Washington, DC, USA, 2005. IEEE Computer Society. (Cited on page 4.)

[83] B. Moayer and K. Fu. A syntactic approach to fingerprint pattern recognition. *Pattern Recognition*, 7(1-2):1–23, 1975. (Cited on page 2.)

[84] F. Mokhtarian and A. K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 14:789–805, 1992. (Cited on page 17.)

[85] A. S. Montero and J. Lang. Skeleton pruning by contour approximation and the integer medial axis transform. *Computers & Graphics*, 36(5):477–487, 2012. (Cited on pages 2, 10, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, and 55.)

[86] A. S. Montero, J. Lang, and R. Laganiere. Scalable kernel correlation filter with sparse feature integration. pages 24–31, December. (Cited on page 12.)

[87] A. S. Montero, J. Lang, and R. Laganière. A general framework for fast 3d object detection and localization using an uncalibrated camera. In *2015 IEEE Winter Conference on Applications of Computer Vision, (WACV) 2014, Waikoloa, HI, USA, January 5-9, 2015*, pages 884–891, 2015. (Cited on pages 5, 12, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, and 90.)

[88] A. S. Montero, A. Nayak, M. Stojmenovic, and N. Zaguia. Robust line extraction based on repeated segment directions on image contours. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pages 1–7, July 2009. (Cited on page 2.)

[89] A. S. Montero, H. Sekkati, J. Lang, R. Laganière, and J. James. Framework for natural landmark-based robot localization. In *Conference on Computer and Robot Vision (CRV)*, pages 131–138, 2012. (Cited on pages 11, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, and 75.)

[90] A. S. Montero, M. Stojmenovic, and A. Nayak. Robust detection of corners and corner-line links in images. In *IEEE International Conference on Computer and Information Technology (CIT)*, pages 495–502, June 2010. (Cited on page 2.)

[91] H. Murase and S. Nayar. Visual Learning and Recognition of 3D Objects from Appearance. *International Journal on Computer Vision*, 14(1):5–24, Jan 1995. (Cited on page 2.)

[92] H. Murase and S. K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, Jan. 1995. (Cited on page 5.)

[93] J. C. Nascimento and J. S. Marques. Performance evaluation of object detection algorithms for video surveillance. *IEEE Transactions on Multimedia*, 8(4):761–774, Aug. 2006. (Cited on page 1.)

[94] G. Nebehay and R. Pflugfelder. TLM: Tracking-Learning-Matching of keypoints. In *International Conference on Distributed Smart Cameras*, pages 21–26. IEEE, Oct. 2013. (Cited on page 1.)

[95] G. Nebehay and R. Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2014. (Cited on page xiv.)

[96] G. Nebehay and R. Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *Winter Conference on Applications of Computer Vision*. IEEE, Mar. 2014. (Cited on pages 1, 6, 23, and 25.)

[97] D. T. Nguyen, Z. Zong, P. Ogunbona, and W. Li. Object detection using non-redundant local binary patterns. In *Proceedings IEEE International Conference on Image Processing (ICIP)*, pages 4609–4612. IEEE, 2010. (Cited on page 2.)

[98] S. Nirjon, J. Liu, G. DeJean, B. Priyantha, Y. Jin, and T. Hart. Coin-gps: Indoor localization from direct gps receiving. In *In Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys)*. ACM – Association for Computing Machinery, June 2014. (Cited on page 4.)

[99] R. Ogniewicz and M. Ilg. Voronoi skeletons: Theory and applications. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 63–69, 1992. (Cited on page 16.)

[100] R. L. Ogniewicz and O. Kübler. Hierarchic Voronoi skeletons. *Pattern Recognition*, 28:343–359, 1995. (Cited on page 16.)

[101] J. Omar and M. Shah. Tracking and object classification for automated surveillance. In *In Proceedings of the European Conference on Computer Vision-Part IV*, ECCV '02, pages 343–357, London, UK, UK, 2002. Springer-Verlag. (Cited on page 5.)

[102] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-time Signal Processing (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999. (Cited on page 94.)

[103] C. Orrite-Urunuela, J. M. del Rincon, J. E. Herrero-Jaraba, and G. Rogez. 2D silhouette and 3D skeletal models for human detection and tracking. In *In Pro-*

ceedings of the IEEE International Conference on Image Processing (ICPR), pages 244–247, 2004. (Cited on page 16.)

[104] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions Pattern Analysis and Machine Intelligence (TPAMI)*, 32(3):448–461, Mar. 2010. (Cited on pages xii, 2, 3, 4, 21, 22, 77, 78, 83, 84, and 86.)

[105] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(3):448–461, 2010. (Cited on pages 2, 5, 57, 62, and 68.)

[106] Y. Park, V. Lepetit, E. Cvlab, and W. Woo. Multiple 3d object tracking for augmented reality. In *Proceedings International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 117–120, 2008. (Cited on page 5.)

[107] N. Payet and S. Todorovic. From contours to 3d object detection and pose estimation. *IEEE International Conference on Computer Vision (ICCV)*, 0:983–990, 2011. (Cited on page 2.)

[108] F. Pernici and A. D. Bimbo. Object tracking by oversampling local features. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 99(PrePrints):1, 2013. (Cited on pages xiv, 6, 23, and 25.)

[109] L. A. Pinilla-Buitrago, J. F. Martínez-Trinidad, and J. Carrasco-Ochoa. A new method for skeleton pruning. In *Pattern Recognition*, pages 301–310. Springer, 2014. (Cited on pages 18 and 105.)

[110] S. M. Pizer, W. R. Oliver, and S. H. Bloomberg. Hierarchical shape description via the multiresolution symmetric axis transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 9:505–511, 1987. (Cited on page 17.)

[111] M. Plass and M. Stone. Curve-fitting with piecewise parametric cubics. *SIGGRAPH Computer Graphics*, 17:229–239, 1983. (Cited on pages 30 and 31.)

[112] K. M. M. Prabhu. *Window Functions and their Applications in Signal Processing*. CRC Press, 2013. (Cited on page 94.)

[113] M. L. Pupilli and A. D. Calway. Real-time camera tracking using a particle filter. In *In Proceedings of the British Machine Vision Conference (BMVC)*, pages xx–yy, 2005. (Cited on page 21.)

[114] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244 – 256, 1972. (Cited on page 31.)

[115] X. Ren and D. Ramanan. Histograms of sparse codes for object detection. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 0:3246–3253, 2013. (Cited on page 2.)

[116] M. Ritch and N. Canagarajah. Motion-based video object tracking in the compressed domain. In *IEEE International Conference on Image Processing (ICIP)*, volume 6, pages VI – 301–VI – 304, Sept 2007. (Cited on page 5.)

[117] A. L. P. Rocha. *Segmentation and Line Filling of 2D Shapes*. PhD thesis, University of Ottawa, 2013. (Cited on page 19.)

[118] A. Roshan Zamir, A. Dehghan, and M. Shah. Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs. In *In Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. (Cited on page 5.)

[119] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, 2006. (Cited on page 21.)

[120] P. M. Roth and M. Winter. Survey of Appearance-Based Methods for Object Recognition. Technical report, Institute for Computer Graphics and Vision, Graz University of Technology, 2008. (Cited on page 2.)

[121] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66(3):231–259, Mar. 2006. (Cited on pages 2, 5, and 21.)

[122] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau. Fall detection from human shape and motion history using video surveillance. *Advanced Information Networking and Applications Workshops, International Conference on*, 2:875–880, 2007. (Cited on page 1.)

[123] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau. Robust video surveillance for fall detection based on human shape deformation. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(5):611–622, 2011. (Cited on page 1.)

[124] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE Computer Society, 2011. (Cited on pages xvi, 2, 3, and 21.)

[125] D. Scharstein and A. J. Briggs. Real-time recognition of self-similar landmarks. *Image and Vision Computing*, 19:763–772, 1999. (Cited on page 4.)

[126] J.-P. Schober, T. Hermes, and O. Herzog. Content-based image retrieval by ontology-based object recognition. In V. Haarslev, C. Lutz, and R. Möller, editors, *In Proceedings of the Workshop on Applications of Description Logics (ADL)*, September 2004. (Cited on page 1.)

[127] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(12):2024–2030, 2006. (Cited on pages 64, 65, and 69.)

[128] S. Se, D. G. Lowe, and J. J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotics Research*, 21(8):735–760, 2002. (Cited on page 21.)

[129] S. Se, D. G. Lowe, and J. J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3):364–375, 2005. (Cited on page 21.)

[130] T. B. Sebastian and B. B. Kimia. Curves vs. skeletons in object recognition. *Signal Processing*, 85:247–263, 2005. (Cited on page 2.)

[131] A. Selinger and R. C. Nelson. A perceptual grouping hierarchy for appearance-based 3d object recognition. *Computer Vision and Image Understanding*, 76(1):83–92, 1999. (Cited on page 5.)

[132] D. Shaked and A. M. Bruckstein. Pruning medial axes. *Computer Vision and Image Understanding*, 69(2):156–169, 1998. (Cited on pages 16, 17, and 19.)

[133] L. Shao and H. Zhou. Curve fitting with Bezier cubics. *Graphical Models and Image Processing*, 58:223–232, 1996. (Cited on page 30.)

[134] F. She, R. Chen, W. Gao, P. Hodgson, L. Kong, and H. Hong. Improved 3d thinning algorithms for skeleton extraction. In *In Proceedings of the International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, volume 1, pages 14–18, 2009. (Cited on pages 3 and 16.)

[135] W. Shen, X. Bai, R. Hu, H. Wang, and L. J. Latecki. Skeleton growing and pruning with bending potential ratio. *Pattern Recognition*, pages 196–209, 2011. (Cited on pages 3 and 18.)

[136] K. Siddiqi and S. Pizer, editors. *Medial Representations: Mathematics, Algorithms and Applications*. Springer-Verlag New York, Inc., 1st ed. edition, 2008. (Cited on page 2.)

[137] K. Siddiqi, A. Shokoufandeh, S. J. Dickenson, and S. W. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1):13–32, 1999. (Cited on pages 1 and 2.)

[138] R. Sim and G. Dudek. Learning visual landmarks for pose estimation. In *IEEE Robotics and Automation Society (ICRA)*, pages 1972–1978, 1999. (Cited on page 20.)

[139] B. Sohn, J. Lee, H. Chae, and W. Yu. Localization system for mobile robot using wireless communication with ir landmark. In *In Proceedings of the International Conference on Robot Communication and Coordination*, RoboComm '07, pages 6:1–6:6, Piscataway, NJ, USA, 2007. IEEE Press. (Cited on page 4.)

[140] G. Stockman and L. G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001. (Cited on pages 2 and 3.)

[141] S. Taylor and T. Drummond. Multiple target localisation at over 100 fps. In *In Proceedings of the British Machine Vision Conference (BMVC)*. British Machine Vision Association, 2009. (Cited on pages xv, 2, 3, 4, 21, 22, 77, 78, 80, 87, and 88.)

[142] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000. (Cited on page 20.)

[143] Y.-L. Tian, R. Feris, and A. Hampapur. Real-Time Detection of Abandoned and Removed Objects in Complex Environments. In *International Workshop on Visual Surveillance*, Marseille, France, 2008. Graeme Jones and Tieniu Tan and Steve Maybank and Dimitrios Makris. (Cited on page 2.)

[144] H. Uchiyama and E. Marchand. Object Detection and Pose Tracking for Augmented Reality: Recent Approaches. In *Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, Kawasaki, Japon, Feb. 2012. (Cited on page 1.)

[145] J. van de Weijer, C. Schmid, J. Verbeek, and D. Larlus. Learning color names for real-world applications. *Transactions Image Processing*, 18(7):1512–1523, July 2009. (Cited on page 26.)

[146] P. Vandergheynst, R. Ortiz, and A. Alahi. Freak: Fast retina keypoint. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 0:510–517, 2012. (Cited on pages xv, 2, and 21.)

[147] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. Global localization from monocular slam on a mobile phone. In *In Proceedings of the IEEE Virtual Reality (VR)*. IEEE, April 2014. (Cited on page 23.)

[148] J. Ventura and T. Höllerer. Fast and scalable keypoint recognition and image retrieval using binary codes. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 697–702. IEEE Computer Society, 2011. (Cited on page 1.)

[149] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 511–518. IEEE Computer Society, 2001. (Cited on page 2.)

[150] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *In Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 125–134. IEEE Computer Society, Sept. 2008. (Cited on pages 22 and 23.)

[151] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):355–368, 2010. (Cited on page 1.)

[152] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 16(3):355–368, 2010. (Cited on pages 2, 3, 21, 22, and 23.)

[153] K. Wang, X. Li, and J. C. Trinder. Mathematical morphological analysis of typical cyclone eyes on ocean sar. In *Geoscience and Remote Sensing Symposium (IGARSS), 2014 IEEE International*, pages 4393–4396. IEEE, 2014. (Cited on page 19.)

[154] T. Wekel and O. Hellwich. Simultaneous skeletonization and topologic decomposition for digital shape reconstruction. 2013. (Cited on page 16.)

[155] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, University of North Carolina, Chapel Hill, NC, USA, 1995. (Cited on page 82.)

[156] M. V. Wilkes, D. J. Wheeler, and S. Gill. *The Preparation of Programs for an Electronic Digital Computer (Charles Babbage Institute Reprint)*. The MIT Press, 1984. (Cited on page 86.)

[157] B. Williams, G. Klein, and I. D. Reid. Real-time SLAM relocalisation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007. (Cited on page 21.)

[158] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 0:2411–2418, 2013. (Cited on pages xii, 1, 6, 12, 23, 25, 26, 99, 100, and 101.)

[159] H. Wuest, A. Pagani, and D. Stricker. Feature management for efficient camera tracking. In *Asian Conference on Computer Vision*, pages I: 769–778, 2007. (Cited on page 21.)

[160] J. yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000. (Cited on pages 26 and 99.)

[161] J. Zeng, R. Lakaemper, X. Yang, and X. Li. 2d shape decomposition based on combined skeleton-boundary features. In *Proceedings International Symposium on Visual Computing (ISVC)*, pages 682–691, 2008. (Cited on pages 1 and 2.)

[162] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, Nov. 2000. (Cited on pages 59 and 60.)

[163] W. Zhong. Robust object tracking via sparsity-based collaborative model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1838–1845, Washington, DC, USA, 2012. IEEE Computer Society. (Cited on pages 23 and 24.)

[164] R. Zhou, C. Quek, and G. Ng. A novel single-pass thinning algorithm and an effective set of peforming criteria. *Pattern Recognition Letters*, 16(12):1267–1275, 1995. (Cited on pages 3 and 16.)