

Immersive Dynamic Scenes for Virtual Reality from a Single RGB-D Camera

by

Po Kong Lai

Thesis submitted 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

© Po Kong Lai, Ottawa, Canada, 2019

Abstract

In this thesis we explore the concepts and components which can be used as individual building blocks for producing immersive virtual reality (VR) content from a single RGB-D sensor. We identify the properties of immersive VR videos and propose a system composed of a foreground/background separator, a dynamic scene re-constructor and a shape completer.

We initially explore the foreground/background separator component in the context of video summarization. More specifically, we examined how to extract trajectories of moving objects from video sequences captured with a static camera. We then present a new approach for video summarization via minimization of the spatial-temporal projections of the extracted object trajectories. New evaluation criterion are also presented for video summarization. These concepts of foreground/background separation can then be applied towards VR scene creation by extracting relative objects of interest.

We present an approach for the dynamic scene re-constructor component using a single moving RGB-D sensor. By tracking the foreground objects and removing them from the input RGB-D frames we can feed the background only data into existing RGB-D SLAM systems. The result is a static 3D background model where the foreground frames are then super-imposed to produce a coherent scene with dynamic moving foreground objects. We also present a specific method for extracting moving foreground objects from a moving RGB-D camera along with an evaluation dataset with benchmarks.

Lastly, the shape completer component takes in a single view depth map of an object as input and "fills in" the occluded portions to produce a complete 3D shape. We present an approach that utilizes a new data minimal representation, the *additive depth map*, which allows traditional 2D convolutional neural networks to accomplish the task. The additive depth map represents the amount of depth required to transform the input into the "back depth map" which would exist if there was a sensor exactly opposite of the input. We train and benchmark our approach using existing synthetic datasets and also show that it can perform shape completion on real world data without fine-tuning. Our experiments show that our data minimal representation can achieve comparable results to existing state-of-the-art 3D networks while also being able to produce higher resolution outputs.

Acknowledgements

Thank you to my supervisor Robert Laganière, for his guidance, encouragement and advice provided throughout my time as his student. Without his support, I would not have been able to freely explore so many different topics in computer vision, machine learning and virtual reality.

I would also like to thank professor Jochen Lang, professor Hussein Al Osman, professor Prosenjit K. Bose and professor Michael Greenspan for a slice of their valuable time to read my work, their insightful comments and for their service as my committee members.

A special thank you to my family, whom without their constant support I would not have been able to complete this thesis.

Finally, last but by no means least, also to everyone I've met at VIVALab... it was a pleasure sharing the laboratory with you all these past few years.

This work was partially supported by the National Sciences and Engineering Research Council (NSERC) of Canada, Thales Canada, Crosswing and BRINX Software.

Table of Contents

Abstract	ii
Acknowledgements	iii
Table of Contents	iv
List of Figures	vii
Glossary	xiii
1 Introduction	1
1.1 Anatomy of an Immersive HMD	2
1.1.1 Consumer Level HMDs	3
1.2 Capture Methods, Hardware and Formats	6
1.2.1 Visual Sensors	7
1.2.2 Capture Methods	9
1.2.3 Output Formats	11
1.3 Thesis Statement	12
1.4 Research Questions	12
1.5 Contributions	13
1.5.1 Video Summarization of Surveillance Cameras	13
1.5.2 Creating Immersive Virtual Reality Scenes Using a Single RGB-D Camera	13
1.5.3 Additive Depth Maps, a Compact Approach for Shape Completion of Single View Depth Maps	14
1.6 Thesis Outline	15
2 A System for Immersive Virtual Reality Videos	16
2.1 Properties of an Immersive VR Video	17
2.2 System Overview	19
2.2.1 Building a 3D Model of the Scene	20
2.2.2 Separating the Foreground and Background	21

2.2.3	Localizing Foreground Objects	22
2.2.4	Filling in Occlusions of Detected People	22
2.3	Database Solution	23
2.4	Generalized Solution	24
2.5	Summary	24
3	Background Subtraction	26
3.1	Literature Review - Background Subtraction	27
3.2	Application: Video Summarization of Surveillance Cameras	29
3.2.1	Problem Definition and Related Works	29
3.2.2	Overview of Proposed Approach	30
3.2.3	Detecting Foreground Objects	31
3.2.4	Building Spatio-Temporal Trajectories	32
3.2.5	Summarization	34
3.3	Dataset and Evaluation	34
3.3.1	Dataset	35
3.3.2	Evaluation Criteria	35
3.4	Results and Discussion	37
3.4.1	Summarization Perception	37
3.4.2	Summarization Evaluation	38
3.4.3	Spatio-Temporal Trajectory Building Performance Evaluation . .	40
3.5	Summary	41
4	Visual SLAM with RGB-D Sensors	45
4.1	Overview of Visual SLAM	46
4.2	RGB-D SLAM	47
4.3	Application: Change Detection in RGB-D Videos Captured with Pa- trolling Robots	49
4.3.1	Related Works	50
4.3.2	Proposed Approach	51
4.4	Summary	56
5	Single Camera Dynamic VR Scene Creation	57
5.1	Related Works in VR Scene Creation	58
5.2	Dynamic VR Scene Creation with a Single Camera	59
5.2.1	System Overview	60

5.2.2	Dataset and Evaluation	64
5.3	Discussion	65
5.4	Summary	67
6	Shape Completion of Single View Depth Maps	68
6.1	Overview of Neural Networks	70
6.1.1	Artificial Neural Networks	70
6.1.2	Convolutional Neural Networks	71
6.1.3	Network Layers	72
6.2	Related Works in 3D Shape Completion	77
6.3	Additive Depth Maps	79
6.4	Network Architecture	81
6.5	Data Generation	84
6.6	Experimental Setup	85
6.7	Experimental Results	87
6.7.1	Evaluation Measures	88
6.7.2	Multi-Task Learning	89
6.7.3	Depth Map Evaluation	89
6.7.4	Volumetric Evaluation	92
6.7.5	Application to Real World Data	94
6.7.6	Learning with Semantic Labels	99
6.8	Discussion	102
6.9	Summary	105
7	Conclusions	107
7.1	Summary	108
7.2	Review of Research Questions	111
7.3	Future Work	114
	References	115

List of Figures

1.1	Basic outline of an immersive HMD which displays stereoscopic images. .	4
1.2	(a) Raw image of the IR LEDs on the Oculus Rift Dev Kit 2 (DK2) and (b) the general LED positions extracted from a blob detector. Image source: http://doc-ok.org/?p=1095	6
2.1	Flowchart of our system for building immersive experiences for VR using a single RGB-D camera. The dashed arrows corresponds to the optional feature of occlusion filling for people trajectories as described in Section 2.2.4 and examined in Chapter 6.	21
2.2	Flowchart of the database solution obtained by augmenting Fig. 2.1 with a database containing the known background model.	23
2.3	Flowchart of the generalized solution obtained by removing the requirement of a background video that produces a 3D Model for camera pose estimation from Fig. 2.1.	25
3.1	Illustration of each major step required in our approach for video summarization of surveillance cameras. From the top to the bottom, we are first given a set of input video frames (first row). Next all of the moving objects, per frame, are detected through a combination of background subtraction, dense optical flow and pixel cluster (second row). With these objects detected, the spatio-temporal (s-t) trajectories are built incrementally by matching the objects with the set of active trajectories for the current frame (third row). Finally once we have all of the s-t trajectories, they can be re-arranged by examining their projections onto a video cube (last row).	42
3.2	Thumbnails of our dataset. Each column is a separate sequence. The top row is a single frame from a sequence and the bottom is a single frame from the corresponding manual binary mask.	43
3.3	A single frame in the video summary (bottom row) and original frames for each object in the summary (top row) for the Building sequence. . . .	43

3.4	A single frame from two different video summaries for the Transitway2 sequence. (a) Overlapping trajectories and no timestamps. (b) No overlapping trajectories and timestamps.	43
3.5	An AR - CR plot for each sequence. The horizontal axis represents the activity recall (AR) and the vertical axis represents the compression rate (CR). A blue line is drawn between the NAS and UniSamp-NAS methods which represents the different summaries, and where they would fall in the AR - CR plot, that could be produced by varying the N parameter for UniSamp-NAS.	44
4.1	Pipeline of our proposed approach. Steps for building and aligning 3D maps, change detection and object segmentation, and producing a video summary are contained in the grey, red and blue boxes, respectively. . . .	52
4.2	Both 4.2a and 4.2b show a planar object with several pictures on it. The differences are apparent in the color images (left) while the depth maps (right) are unable to pick up this difference.	54
4.3	The results of change detection without appearance constraints (left) and with appearance constraints (right) overlaid onto of the probe 3D map (top row) and video frame (bottom row). The appearance constraints allows us to detect the plane with pictures on it from Fig. 4.2a.	54
4.4	A example of projecting the view from a camera onto the original video frame so that the detected differences are overlaid. The detected differences were rendered as colored discs and the camera bounds as the green rectangle. (a) Projected points onto pixel buffer. (b) Crop to camera bounds and compute θ to align with the axis. (b) Rescale to original size and draw the differences.	55
5.1	Visualization of the pipeline for our proposed approach to solve the first three properties as defined in Section 2.1.	61
5.2	Example of the results produced from the input video in Fig. 5.1. A single frame is shown from multiple viewpoints coherently with the complete background model.	61

5.3	A sampling of frames from the KinectLoop sequence (see Table 5.1) with (a) color frames and the foreground binary masks from (b) just the Kinect SDK and (c) using Kinect SDK combined with sKCF and segmenting with flood fill.	63
5.4	Static background models produced with background masks from the (a) Kinect SDK only (note the unsegmented foreground fused into the background) and (b) Kinect SDK + sKCF [145].	64
5.5	Per frame TP (green lines) and FP (red lines) results for the sequences captured using a structured light sensor. Vertical and horizontal axis denotes the TP or FP value and frame number, respectively. Columns correspond to a different image sequence from our dataset. Note, since the structured light sensor did not capture body poses, we manually initialized sKCF on the first frame.	66
5.6	Per frame TP (green lines) and FP (red lines) results for the sequences captured using a Kinect V2. The left column contains results from using just the Kinect SDK to perform tracking and dynamic foreground object segmentation while the right column combines the sKCF tracker with the Kinect SDK to track and segment the dynamic foreground objects. . . .	67
6.1	Illustration of the differences between a fully connected layer with one neuron (left) and a convolutional layer with an output volume (right) when applied to an image. The single neuron for a fully connected layer has weights for each pixel while a neuron from the output volume of the convolutional layer only has weights for a region (magenta) of the image. The fully connected neuron (left) must be replicated many times in order for a fully connected layer to produce the same output volume as the convolutional layer (right).	73
6.2	Diagram of the depthwise separable convolution. The blue region shows the $n \times n$ convolutions performed on each channel of the input data volume separately. The orange region shows the pointwise convolution which transforms the results of the depthwise convolutions into the desired output data volume.	75
6.3	Diagram from He et al. [54] who originally introduced the residual block.	75

6.4	Illustration of the additive depth and its use in shape completion. On the left is the ground truth mesh viewed from the front and side. Next is a camera imaging the mesh to produce the training data. The additive depth, d_{ad} , is visualized as the (exaggerated) red lines from the camera facing mesh (in grey) to the back view of the mesh (in green). To the right is d_{ad} shown as an image as well as the depth maps d_f and d_b which correspond to the grey and green partial meshes, respectively. The union of d_f and d_d produces there mesh result shown in the far right. For all depth map images, brighter colors imply further distances.	81
6.5	General structure of the network. The numbers at the top/bottom of each block indicates the number of incoming/outgoing feature maps.	82
6.6	Detailed visualization of the contraction (top), expansion (bottom left) and dense (bottom right) blocks from Fig. 6.5. DWConv represents a separable depth-wise convolution, d_{in} is the number of input feature maps while d_{out} is the number of output feature maps. For the dense blocks, networks which do not have a particular output will have their dense branch removed. The variable L refers to the number of depth labels. . .	83
6.7	Examples of training data generated from mesh models. (a) Original mesh model, (b) front facing depth map d_f , (c) back facing depth map d_b , (d) the additive depth map $d_{ad} = d_b - d_f$. For both d_f and d_b , darker colors has lower depth value while for d_{ad} blue has lower depth value. (e) normal map of d_{ad} and (f) depth labels image of d_{ad} with $\lambda = 2.0$ and $L = 16$. . .	86
6.8	Example meshed outputs of a male (left) and female (right) from the clothed human dataset shown from two different views (top/bottom). The input is shown in grey, ground truth in green and the network prediction in blue. For both examples, from left to right, we have (a) the original mesh, (b) ground truth mesh from depth maps, (c) prediction mesh from depth maps, (d) applied Poisson surface reconstruction to results of (b) and (e) applied Poisson surface reconstruction to results of (c).	90

6.9	Example meshed outputs of lamp, motorbike, rocket, vase, chair and table categories from the SHREC'17 dataset. From left to right: original mesh model, ground truth, prediction, ground truth with Poisson surface reconstruction and prediction with Poisson surface reconstruction. As in Fig. 6.8, the input is shown in grey, ground truth in green and the network prediction in blue.	91
6.10	Voxelization results of a male (left) and female (right) from the clothed dataset. For both subjects, the input voxels are shown in red and the reconstruction in yellow. The top three rows visualize voxelizations of the input, ground truth and prediction. The bottom three rows visualize post processing with Poisson surface reconstruction prior to voxelization for the ground truth, prediction and the original mesh model.	95
6.11	Voxelization of our shape completion results on six SHREC'17 categories. The inputs and outputs for each category are the same as those show in Fig. 6.9.	96
6.12	Encoder-decoder network for predicting the back color images. The numbers for each block represent the number of feature maps. Each convolution and deconvolution is applied with a kernel size (3×3) . The orange, blue and green blocks also use a stride of (2×2) thus halving (orange, blue) and doubling (green) the image dimensions.	98
6.13	Mesh visualizations of completed human shapes from captured RGB-D video. Each row is a different frame from the video and each column presents an alternative view of the input and completed shape.	98
6.14	Modification of the network shown in Fig. 6.5 to allow for two inputs: the normalized depth map and the corresponding semantically labeled image. Like in Fig. 6.5, the numbers at the top/bottom of each block indicates the number of incoming/outgoing feature maps. The blue and green triangle shape connected by four black lines represents the contraction, expansion and dense portions of the original network.	101
6.15	Examples of semantically labeled mesh data. The colors of each label represent the following body parts: clothed main body (blue), head (magenta), cyan (left arm), red (right arm), left leg (yellow) and right leg (green).	102

6.16	Examples of semantic images generated from the manually labeled clothed human mesh models. (a) Input depth map, (b) ground truth additive depth map, (c) labeling from mesh data of the front view, (d) labeling from mesh data of the back view, (e) combined labeling of (c) and (d) with left and right arms/legs merged into two distinct pixel groups. Note that we place the limb labels on top of the body labels as they tend to cross the body often.	103
6.17	A visual comparison between the outputs and reconstructions from our best performing model (denoted as "Baseline") trained with just depth as the input and another model (denoted as "Semantics") trained with both depth and semantic labels as input. The input depth map from the two examples has one arm completely occluded which makes it difficult to perform reconstruction. Providing the network with a hint through the semantic labels allows it to reconstruct hidden body parts more accurately and completely.	104

Glossary

- SE_3 A mathematical term used to describe arbitrary combinations of translations and rotations (but not reflections) in 3D Euclidean space. 46
- $\mathcal{F}_{bg}$ Frames of $\mathcal{F}_{input}$ which were processed by a foreground background separator so that only the background pixel data remains. xii, 19, 20, 22, 60
- $\mathcal{F}_{fg}$ Frames of $\mathcal{F}_{input}$ which were processed by a foreground background separator so that only the foreground pixel data remains. xii, 19, 20, 22, 60
- $\mathcal{F}_{input}$ Set of input RGB-D frames captured from a sensor. xii, 19, 22, 24, 60, 63
- $\mathcal{M}_{static}$ Static background model produced from $\mathcal{F}_{bg}$. 20–24, 60, 63
- $\mathcal{V}_{input}$ Input RGB video for video summarization of surveillance camera algorithms. 29, 30
- $\mathcal{V}_{output}$ Output video from surveillance camera video summarization algorithms. 29, 30
- f^i The i^{th} frame of $\mathcal{F}_{input}$. 21, 22, 24, 60, 63
- f_{bg}^i The i^{th} frame of $\mathcal{F}_{bg}$. 22, 60
- f_{fg}^i The i^{th} frame of $\mathcal{F}_{fg}$. 22, 60, 63
- ANN** (A)rtificial (N)eural (N)etwork are computational systems which are inspired by, but are not identical to, the biological neural networks found in brains of animals in nature. 70–72, 76
- BG** Short-hand for the term "background". In the field of computer vision, the term background is context sensitive but generally represents all of the information which *is not* of interest. 27, 28
- BRISK** (B)inary (R)obust (I)nvariant (S)calable (K)eypoints is a general-purpose binary descriptor and keypoint extractor with low computational cost [87]. 47

CNN (C)onvolutional (N)eural (N)etwork is a class of deep neural networks most commonly utilized in visual image analysis. 15, 69, 72, 74, 76–80, 93, 106, 110, 111, 114

correspondence problem In computer vision, the correspondence problem deals with automatically ascertaining which parts of one image correspond to which parts of another image. A solution would yield a set of points from one image that map to another set of points on the other image. 7, 8

FG Short-hand for the term "foreground". In the field of computer vision, the term foreground is context sensitive but genenerally represents all of the information which *is* of interest. 27

HMD (H)ead (M)ounted (D)isplays are a type of display device that is worn on the head or as part of a helmet. A small digital display is used show to content to the wearer. A HMD is the primary component of consumer virtual reality headsets. vi, xiv, 1–9, 11, 12, 19, 108, 109, 112

ICP (I)terative (C)losest (P)oint is an algorithm that computes the spatial transformation which aligns two point sets. This algorithm is often used to reconstruct 3D surfaces by aligning multiple scans or frames from an RGB-D sensor. 47, 48

LiDAR (Li)ght (D)etection (A)nd (R)anging is a surveying method that measures distance to a target by illuminating the target with laser light and measuring the difference in laser return times with a sensor. 8–11

marching cubes An computer graphics algorithm that extracts a polygonal mesh of an isosurface from a three-dimensional discrete scalar field [96]. 48, 113

perspective-n-point problem Deals with the problem of estimating the pose (rotation and translation) of a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the image. 5

photogrammetry A field in computer vision which contains methods for obtaining reliable information (most often 3D data) about physical objects and environments through the process of recording, measuring and interpreting digital images taken from visual sensors. 2, 11, 20

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 approach which outputs a reasonable result with a certain probability via separation of inliers from outliers. 47, 48

RGB-D SLAM A variant of SLAM where only RGB-D sensors are used. 13, 14, 20, 47, 50, 51, 53, 56, 60, 63, 64, 109, 110, 113

SIFT (S)cale-(I)nvariant (F)eature (T)ransform is an algorithm in computer vision to detect and describe local features in images [98]. 47, 59

SLAM (S)imultaneous (L)ocalization (A)nd (M)apping is the computational problem of constructing a map of an unknown environment (or updating a map within a known environment) using sensors integrated in a machine while simultaneously keeping track of the machine’s location within it. xiv, 11, 15, 20, 24, 45–47, 53, 56, 69, 109

TSDF (T)runcated (S)igned (D)istance (F)unction is a volumetric scene representation that allows for integration of multiple depth images taken from different viewpoints [18]. 47, 48, 113

UVC The (U)SB (V)ideo (C)lass is a USB device class which describes devices that are capable of streaming video. 5

video summarization The act of producing a video summary. vi, xii, 13, 26, 29, 30, 34, 40–42, 55, 107, 109

video summary An abbreviated video which preserves the important elements of the original video while removing the spatio-temporal segments which are not of interest. vi, vii, xiv, 13, 26, 29–31, 34–38, 40, 41, 43, 50–52, 55, 109

VR (V)irtual (R)eality is a simulated experience that can be similar to or completely different from the real world. Currently, regular consumers can experience VR through the use of commodity HMDs which simulate binocular human vision so that the user can perceive depth in the content they are viewing. vi, 1, 2, 6–8, 10, 12–19, 21, 24, 26, 45, 57–60, 62, 65–69, 107–112, 114

Chapter 1

Introduction

With the recent introduction of immersive head mounted displays (HMD) at the consumer level, interest in producing virtual reality (VR) content has been renewed. Broadly speaking, content for VR falls between the spectrum of completely virtual to completely reality-based. An example of completely virtual content would be a video game whose assets were created through a 3D modelling program such as Blender [8] while an example of completely reality-based content would be a 360° video captured using multiple synchronized calibrated cameras. On the one hand, the completely virtual approach allows for complete control over the final product leading to a potentially more powerful experience. On the other hand, finer control over the details often results in an increased amount of manual labour with the additional risk of wandering into the uncanny valley [135]. In contrast, the completely reality-based approach uses visual sensors to capture a scene from the real world for a user to experience virtually. The goal is to take the

user's head movements as detected by the HMD and show the appropriate portion of the captured content. The final product of this approach often depends on the sensors used. An approach which falls between the virtual and reality-based methods would have elements of both: some sensor(s) to capture a scene (or object of interest) which is then displayed through a 3D model in virtual space. An example of this would be observing a 3D mesh model produced from photogrammetry. For all approaches, it is the use of the HMD which allows for immersion as, depending on the limitations of the content, the user can have 3 to 6 degrees of freedom (DoF) movement.

In this thesis we propose an approach for creating VR content which leans towards the reality-based methods. More specifically, we explore the problem of creating an immersive VR experience using data captured with a single RGB-D camera. Unlike other existing methods for VR content creation with visual sensors, the approach presented in this thesis produces a 3D video where each of the frames are true 3D objects that are consistently aligned with a static 3D background. The result is a more immersive experience where the viewer can have 6 DoF motion while observing their captured content. Furthermore, since we apply a single sensor restriction, the methods presented in this thesis lowers the barrier of entry for producing immersive VR experiences to consumer levels. Applications which have a single sensor restriction, such as journalists in environments which require impromptu filming, can also make use of our methods to produce more engaging VR content.

The rest of this chapter is organized as follows. Section 1.1 covers the basic principles of an immersive HMD while Section 1.2 details different methods and hardware for capturing VR content. Section 1.3 presents the thesis statement, Section 1.4 states the main research questions and Section 1.5 lists the main contributions of this thesis. Finally, an outline for the rest of this thesis is given in Section 1.6.

1.1 Anatomy of an Immersive HMD

In order for a HMD to be immersive it must satisfy the following criteria:

1. Encompass the peripheral vision of a user;
2. One-to-one tracking of user's head movement;
3. High resolution and fast refresh rate displays;
4. Lightweight and ergonomic.

The first criteria is required since moving objects or light in the peripheral vision can draw a user's attention away from the content they are viewing which breaks immersion. To reduce these outside distractions a large display screen and an outer shell covering the peripheral vision is typically utilized. Without one-to-one tracking of the user's head movement, the HMD would be no different than a standard computer monitor. Thus, the second criteria allows a HMD to immerse a user by replicating their own head movements in the content that they are viewing. A HMD which does not satisfy the third criteria again results in a loss of immersion. This is because a low resolution image does not provide the necessary amount of detail and a low refresh rate may induce "simulator sickness" since the display of content lags behind the user's head movement [70, 84]. Finally, the last criteria allows a user to wear the HMD for extended periods of time without undue stress to their neck or face.

1.1.1 Consumer Level HMDs

Recently, affordable consumer level HMDs which satisfy all of the above criteria have been released. The most prominent ones are the Oculus Rift ¹ and the HTC Vive ². Both HMDs work in a similar manner with a major difference in how the one-to-one head tracking is performed. We will first describe how both HMDs are able to display a stereoscopic image to a user before examining their respective head tracking methods.

Stereoscopic Image Viewing

The basic principle for displaying a stereoscopic image in an immersive HMD is to provide each eye with a different image taken from a viewpoint representing the physical location of the eye. This technique allows for binocular vision [7] of the presented content which in turn allows the viewer's brain to perceive depth. Depending on the HMD, the actual images may be displayed through one or two screens. In the case of one screen HMDs, the screen is split in half with the contents from the left eye viewpoint displayed in the left half and similarly for the right eye. In the case of two screens no screen splitting is required but the displays need to be synchronized. In both cases a separator is used so that the contents intended for each eye are not mixed together. Fig. 1.1 provides an illustration of binocular vision through an immersive HMD.

¹<https://www.oculus.com/>

²<https://www.htcvive.com/>

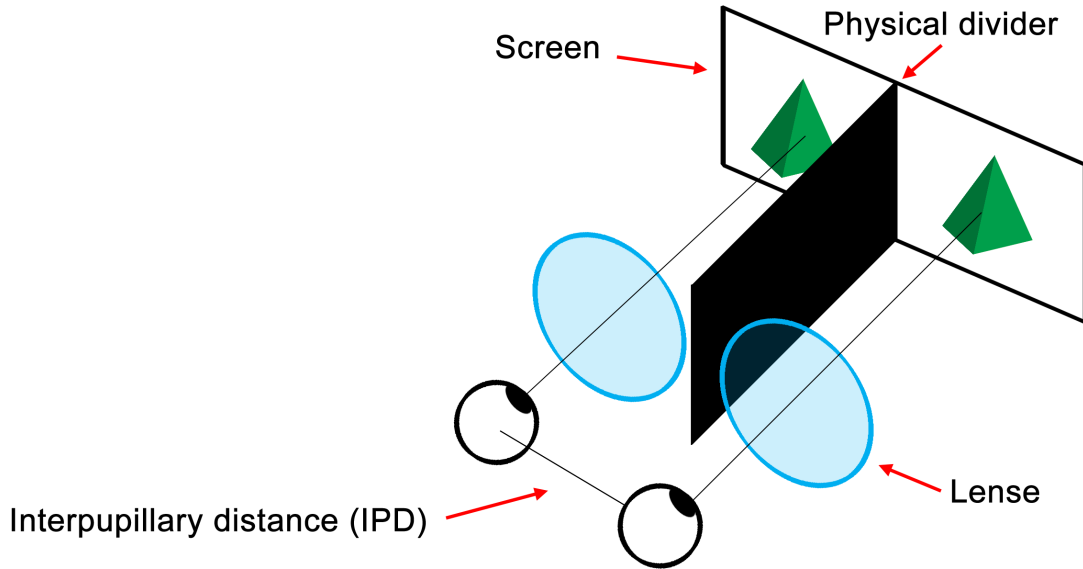


Figure 1.1: Basic outline of an immersive HMD which displays stereoscopic images.

Producing a viewpoint per eye is trivial in the case of purely virtual content since it is simply a virtual camera positioned at each of the eye positions. Determining the viewpoints of reality based content often depends on the output of the capture device. For example, in non-stereoscopic 360° videos the final stitched image is texture mapped onto a virtual object which contains the viewpoints of the viewer. Thus the solution reduces to the case of viewing purely virtual content. When the video to be displayed is captured using a standard single passive stereo camera (see Section 1.2) with a baseline approximately equal to the average distance between human eyes, the viewpoints would need to be positioned at the individual camera locations. Changing the viewing baseline would alter depth perception causing the viewer to feel as if they were a giant with eyes farther apart than a normal human (in the case of a longer baseline) which in turn miniaturizes the content shown in the video. While this can sometimes be an intended effect [164], it can also lead to discomfort [75]. More details on capture hardware and their outputs can be found in Section 1.2.

Head-Tracking

We will first describe the principles for rotational head-tracking before examining how positional head-tracking can be achieved. As the concepts for rotational head-tracking are virtually the same between the Oculus Rift and the HTC Vive, we refer readers interested in a detailed break down of rotational head-tracking to LaValle et al. [83] who

contributed to the head-tracking solution for the Oculus Rift.

An inertial measurement unit (IMU) equipped with a gyroscope, accelerometer and magnetometer is placed in the HMD to measure the angular velocity, linear acceleration and magnetic field strength. While the measurements of the gyroscope from the IMU allows for the estimation of pitch, roll and yaw through dead reckoning - error accumulates over time in the form of drift [169, 72]. The *tilt error* is all of the drift error except the rotation about the vertical axis in the physical world and can be corrected by averaging the accelerometer data over a long period of time to estimate the direction of gravity. If this error was not corrected then horizontal planes would not be perceived as being level. Since the gyroscope is reliable in the short-term and the accelerometer is reliable in the long-term, the two are combined with a complementary filter [38]. The remaining error, known as the *yaw error*, is concerned with the rotation about the vertical axis (i.e: axis which is parallel to the gravity vector) and is corrected using the magnetometer. In the approach by LaValle et al. [83] they first calibrate the magnetometer and then assign reference points in order to detect the yaw error. Once the error is detected they perform a similar procedure as the tilt error correction and use a complementary filter except it is only about the Y axis. With the tilt and yaw errors corrected the user can experience drift free rotational freedom while wearing a HMD.

The general concept of positional head-tracking for the Oculus Rift and the HTC Vive are similar in the sense that they both employ a tracking solution that leverages an external source with some IR markers on the HMD. Thus, in both HMDs, if the IR markers are not visible to the external source then positional head-tracking fails. The biggest difference between the Oculus Rift and the HTC Vive is that the former uses an "outside-in" approach while the latter uses an "inside-out" approach. As the hardware of the Oculus Rift and the HTC Vive are not open source, much of the information for how positional tracking works was reverse engineered by the enthusiast community^{3 4}.

The Oculus Rift utilizes a standard UVC web camera with an IR filter to detect the IR markers, represented as IR LEDs on the HMD, and a blob extractor [129, 24] to extract the position of the IR markers per frame (see Fig. 1.2). To associate individual IR LEDs across multiple frames as well as to identify where it is on the HMD, each of the IR LEDs blink a unique pattern. Once identified, the problem of estimating the position of the HMD in 3D space reduces to solving the well studied perspective-n-point problem [86] with known a-priori image point to 3D model point correspondence.

³<http://xinreality.com/wiki/Constellation>

⁴<http://xinreality.com/wiki/Lighthouse>

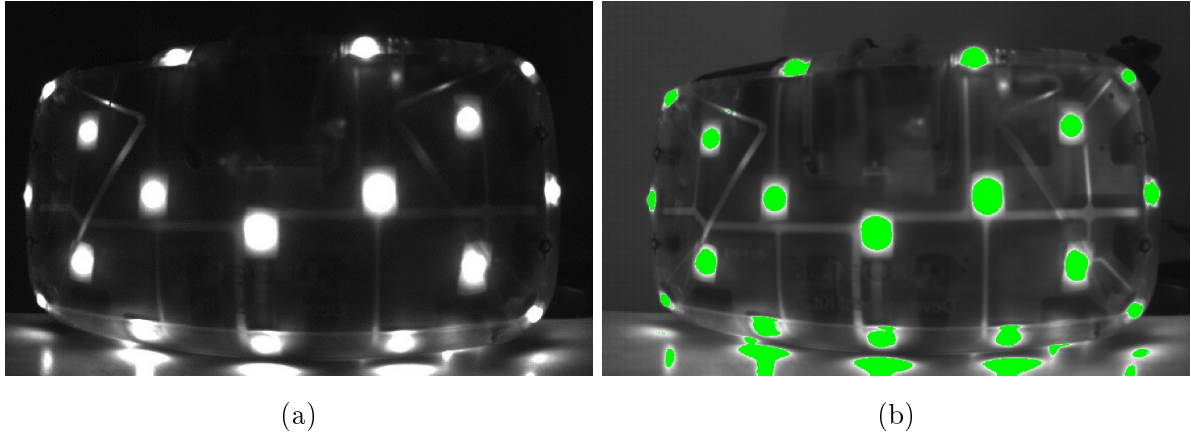


Figure 1.2: (a) Raw image of the IR LEDs on the Oculus Rift Dev Kit 2 (DK2) and (b) the general LED positions extracted from a blob detector. Image source: <http://doc-ok.org/?p=1095>

The HTC Vive utilizes two "base stations" each consisting of an IR beacon and two laser emitters which spin rapidly. The IR beacon floods the tracking volume with IR light and IR receptors on any tracked devices (in this case the HMD) intercept the light to determine their own position with respect to the base station. More precisely, each base station emits a synchronization pulse and 1 of the 2 spinning lasers sweeps a beam across the room. When an IR receptor detects the synchronization pulse it begins to count the amount of time it takes until the sweeping laser beam is detected. Once detected their system then knows when the IR receptor was hit which in turn allows them to compute the receptors angle with respect to the base station. Since each IR receptor has a fixed known position on the HMD, and the two base stations produces two angles per receptor, the exact position of the overall HMD can be computed.

1.2 Capture Methods, Hardware and Formats

There exists a large number of capture methods and hardware which can be used to record content for VR and each of these methods has their strengths and weaknesses. In Section 1.2.1 we provide a brief overview of the different types of hardware which can be used to capture content. In Section 1.2.2 the main capture methods, and how they relate to the capture hardware, are discussed. The different output formats which can be produced from the combination of different capture hardware and methods is covered in Section 1.2.3.

1.2.1 Visual Sensors

Broadly speaking, there are five main types of visual sensors which can be combined to produce 10 categories of capture hardware. The types of visual sensors and how they relate to producing VR content are described below. The various combinations of sensor types, with their relative capabilities, are listed in Table 1.1.

Passive Stereo In a passive stereo system there are two (or more) calibrated cameras which are at some distance apart. In terms of viewing through an immersive VR HMD, if the distance between two cameras is equal to the interpupillary distance (IPD) of the user, then just displaying the left camera image on the left eye and the right camera image on the right eye will allow the user's brain to perform the difficult task of fusing the two images into a depth map. If the camera distance is not equal to the IPD of the user then disparity effects, such as miniaturization of the scene, will occur. In order to view proper depth, any of the stereo vision algorithms in the survey paper by Tippetts et al. [157] will suffice. It should be noted that all stereo algorithms suffer from an inability to handle too little or too much texture both of which causes instability when matching correspondences. Multi-view passive stereo systems can be used to alleviate some of these limitations as well as providing some robustness with respect to occlusion. An overview of the stereo correspondence problem is provided in the book by Szeliski [152] and an extensive detailed look at multi-view geometry is covered in the book by Hartley and Zisserman [52].

Active Stereo In an active stereo system, the correspondence problem is alleviated by actively controlling the scene's illumination through the projection of known patterns onto the scene [133]. The projected patterns, known as structured light, are encoded such that each pixel has a unique label (often called "gray codes"). The correspondence problem is then reduced to finding the pixels between the image pairs which have the same unique code. Since there is no global optimization or exhaustive block matching, the structured light approach is fast and has been adopted for use in the first version of Microsoft's Kinect to produce RGB-D videos. The main weaknesses of this approach is the depth map quality, which reduces exponentially with distance [71], and the requirement of seeing the projected patterns resulting in poor performance for outdoor environments. Additionally the resolution of structured light sensors are limited by the resolution of the light projectors

which results in a low resolution depth map when compared to its corresponding color camera. A user can view content captured from an active stereo system through an immersive VR HMD in full 3D resulting in a full 6 DoF experience.

Time-of-Flight (ToF) These types of systems measure distance by illuminating a target with a beam of laser light. The main difference between ToF and structured light active stereo is the fact that ToF does not project a pattern to solve a reduced correspondence problem, but instead measures the time it takes for a laser pulse to hit the target and return.

The output of ToF sensor is a point cloud where each point represents a surface that the laser has illuminated. Accuracy is the biggest advantage a ToF sensor has over passive and active stereo as sub-millimeter measurements can be achieved at short and long ranges. For this reason, the most common ToF sensor, Light Detection and Ranging (LiDAR), is often used as a surveying tool [45]. While the point clouds produced from a LiDAR are large, accurate, dense, and more recently colored, a single complete scan can take a long time to complete. It is only recently that high frame rate LiDARs have been developed [109] at the cost of severely reduced resolution. Thus, in general, LiDAR is not a suitable tool for recording any dynamic content.

It should be noted that Microsoft’s new Kinect is a ToF sensor which is different than traditional LiDAR in the sense that it does not scan the scene with a laser beam. An IR projector is used to illuminate the scene with a light pulse and a specialised image sensor is used to measure time for the IR to travel to the target. Since no physical scanning is required, the new Kinect can acquire range data at acceptable frame rates and resolution for recording dynamic content.

Finally, the output of any ToF sensor is a point cloud, and since it is a fully 3D object, 6 DoF is available for viewing through an immersive VR HMD.

Full 360° Sensors Many different types of hardware can fall into this category as the only requirement is the output of a full 360° view of its surroundings. In the context of VR, these sensors can provide the user full rotational freedom. Note that since this category of sensor spans many different hardware configurations and implementations, positional freedom is not always available. For example a 360° video can be captured by stitching together the output of several synchronized standard RGB cameras [110, 41] but true depth information is typically not possible. It

can be achieved if each of the standard RGB cameras were replaced with a passive stereo camera [14, 52] or an alternate system [118]. A LiDAR system can also capture in full 360° but currently it is limited in capturing dynamic content.

Light Field This type of sensor captures what is known as the "light field" of a scene. A light field includes both the intensity of light and the direction of the light rays traveling in space [50, 89, 90, 113]. In essence, a light field sensor is composed of an array of "micro-lenses" each of which captures an image from a slightly different position. This results in the ability to capture light rays traveling along the direction of a particular lens.

Once a light field has been captured, novel views can be reconstructed through interpolation between the images captured from different micro-lenses. The main advantage of this sensor is the ability to capture the lighting information within the scene. The biggest drawback is the enormous amount of space required to store even a single light field image as it is composed of multiple images captured from each micro-lens. To combat this weakness there has been a number of techniques designed to compress and progressively transmit such representations [101, 140, 141].

A light field capture can theoretically allow for 6 DoF viewing of high resolution images (limited by FoV of the capture device) in an immersive HMD. To date there does not exist a commercial light field capture device which can capture dynamic content.

1.2.2 Capture Methods

Once the capture hardware has been selected, precisely how it is used to capture content results in different output formats. For example, a standard RGB camera can be used to capture single images, videos and through structure from motion [52, 152] a 3D model can be computed. There are two main categories of capture methods which depend on the number of sensors used. We first describe the different ways a single sensor can be used to capture different types of content before covering how adding more sensors changes the capture methodology.

Single Sensor When a single sensor is used it can be placed in a fixed location to capture still frames and videos. Still frames are the standard output of choice for

Sensor Type	Rotational	Positional	Resolution	Frame Rate
360° video	Yes	No	High	High
Stereo video	Limited	No	High	High
360° stereo video	Yes	No	High	High
Structured light	Yes	Yes	Medium	High
ToF depth video	Yes	Yes	Medium	High
Passive-active stereo fusion	Yes	Yes	High	High
Survey LiDAR	Yes	Yes	High	N/A
360° LiDAR	Yes	Yes	High	Low
Flash LiDAR	Yes	Yes	Low	Medium
Light field	Limited	Limited	High	N/A

Table 1.1: Common categories of capture hardware devices and their relative capabilities.

sensors such as a survey LiDAR but typically a video is preferred since it allows for dynamic content. When the sensor is allowed free motion, just displaying the video captured forces the user along the camera trajectory and results in a roller-coaster like experience. This is especially true for the sensors which are not capable of positional freedom. If there are no dynamic objects in the scene, then by exploiting the motion of the camera, a background model (see Section 4) or even a light field [113] can be captured and displayed. How to handle dynamic objects and still produce an immersive experience from single sensors is the main focus of this thesis.

Multi-Sensor When multiple sensors are utilized, they are often placed in fixed locations to capture alternative viewpoints within a fixed space [64, 134, 152, 13]. In the context of capturing VR content, these alternative viewpoints reduce occlusions providing the user with a more complete experience. A weakness of this type of multi-sensor configuration is the requirement of a fixed location and a setup and tear down procedure. Fusion between two different types of sensors, such as a passive stereo with a ToF depth camera, have also been explored [44, 179] with the goal of increasing the quality of the depth map by leveraging the strengths of each sensor type.

1.2.3 Output Formats

While there are a wide range of sensors (Section 1.2.1) and two main capture methods (Section 1.2.2) only four main output formats are possible. Listed below are brief descriptions of these formats.

Static Environments These are 3D models which can be created from a single or multi-sensor setup and are most commonly built using photogrammetry or dense visual simultaneous localization and mapping (SLAM) (see Chapter 4). The model is completely static with baked lighting and no dynamic elements. Handling a single dynamic object has recently been explored [111] but a general solution which incorporates the background environment still remains an open problem. Every sensor listed in Section 1.2.1 can produce a static environment with the methods for RGB-D sensors explored in Chapter 4.

Regular Video This format simply displays the recorded video directly on the HMD and the user is, at most, allowed rotational freedom. The passive stereo, 360° video and 360° stereo video hardware types from Table 1.1 output this content format. While relatively simple to implement, since most of the difficulty lies in calibrating the capture hardware and depth perception is handled naturally by the brain, the lack of freedom means the user is forced to follow the camera. It can be immersion breaking when subtle head movements such as moving to a more comfortable position is not reflected.

RGB-D Video This format displays the content as either a point cloud or triangulated mesh per frame. The user is allowed full rotational and position freedom since each frame is a 3D object in space. All of the hardware categories from Table 1.1 can produce this type of format with different degrees of accuracy, resolution and frame-rate. For example, a 360° non-stereo video capture device can produce a RGB-D video by exploiting the overlaps between adjacent RGB cameras but the result is a poor depth map estimation. Similarly, a survey LiDAR can be used to image single frames at a time to produce a video but the result would be of a similar quality to stop-motion video where each frame is imaged separately. The active and passive stereo hardware types are best suited for producing this content format at a reasonable resolution and frame-rate.

Light Field Video This format is unique to light field sensors as novel viewpoints

are found through image interpolation [152]. Since the result is an interpolated image, the user still views it in a similar manner as a regular image meaning depth perception is still handled by the brain. It does suffer from some limitations as the degree of rotation is linked to the field of view of the sensor and complete positional freedom does not exist as the images are not true 3D objects. Producing light field videos is still an open problem as discussed in Section 1.2.1.

1.3 Thesis Statement

As consumer level HMDs are now available (Section 1.1), methods for producing immersive VR content is therefore of great interest. Of the many different capture hardware and methods available today (Section 1.2), a single passive or active sensor can be implemented in a mobile device ⁵. Thus, the main goal of this thesis is to explore and develop methods which should be used as components within a larger system for creating immersive VR videos with data captured from a single RGB-D sensor. An immersive VR video should (1) surround the viewer with the background of the scene which contained the sensor performing the capture, (2) display the captured dynamic objects coherently within the reconstructed scene, (3) be as complete as possible and (4) allow the viewer 6 degrees of freedom motion within the reconstructed virtual environment.

1.4 Research Questions

We address the following questions in this thesis:

1. What properties does an immersive VR video have and how can it be displayed?
2. How can we remove the background and capture all moving foreground objects from a single RGB-D video stream?
3. How do we model and display the static portions of the input RGB-D video?
4. How can we merge the results of (2) and (3) coherently into one dynamic scene?
5. Is it possible to "fill in" the occluded portions of the moving foreground objects?

⁵Project Tango is a tablet with a depth sensor: <https://www.google.com/atap/project-tango/>

1.5 Contributions

This thesis makes the following contributions:

1.5.1 Video Summarization of Surveillance Cameras

A method for extracting foreground object trajectories for the purposes of video summarization is devised. A new approach for producing a video summary, an abbreviated video containing all of the “important” elements, is presented along with new evaluation measures and a benchmark dataset with labelled ground truth.

The foreground object trajectory extractor utilizes background subtraction, optical flow and pixel clustering to capture moving objects and also employs a static object detector for the cases where objects of interest remain stationary in the scene. The video summarization algorithm treats the extracted foreground object trajectories as 3D objects and first condenses the temporal component before optimizing their spatial arrangement in the final output. This allows multiple object trajectories to be played one after another even if the objects appeared in the original video at different times. Evaluation of video summaries is difficult as the quality of a summary is dependent on the application and target users. As the goal of our approach was to retain as much of the moving objects as possible, we introduce new evaluation measures aimed at evaluating how well a given video summary captures the relevant activities in relation to the summary length. We compared our approach against several baseline fast-forwarding approaches and found that our video summaries retained more activity compared to similar length summaries produced through traditional frame-skipping.

A more detailed analysis of this contribution can be found in Chapter 3. The research endeavors in this area led to the following publication:

P. Lai, M. Décombas, K. Moutet and R. Laganière. *Video Summarization of Surveillance Cameras*. IEEE Advanced Video and Signal-based Surveillance (AVSS), 2016.

1.5.2 Creating Immersive Virtual Reality Scenes Using a Single RGB-D Camera

The proposed approach utilizes a combination of visual object tracking, human pose estimation and RGB-D SLAM in order to create immersive virtual reality (VR) scenes

from RGB-D video captured with a single sensor. Human pose estimation augmented with visual object tracking is used to extract the trajectories of human actors from the scene so that the background only frames can be fed into a RGB-D SLAM system. In this way, a fully 3D static background is reconstructed and used as a reference point in order to re-project the previously extracted trajectories back into the scene. The result is a volumetric video such that a user can view the captured video from any viewpoint using a VR head mounted display and interact with the scene with motion controllers.

The quality of the final result largely depended on: (1) how accurately we separated the foreground (i.e: human actors) from the background (2) how well the static background has been reconstructed. Since there already exists studies on RGB-D SLAM benchmarking and evaluation, we focused our evaluation on the foreground/background separator. To that end, we measured the accuracy of the foreground/background separator via the overlap of extracted foreground masks and our manually labeled ground truth foreground masks. Our evaluation showed that using a visual object tracker to augment a human pose estimator allowed for the highest quality extraction of foreground masks as the tracker could take over in the cases where the pose estimator failed. This is important for the task of VR scene creation since failure to extract the foreground objects for even a single frame would result in those frames fusing the foreground into the static background.

A more detailed analysis of this contribution can be found in Chapter 5. The research endeavors in this area led to the following publication:

P. Lai and R. Laganière. *Creating Immersive Virtual Reality Scenes Using a Single RGB-D Camera*. 14th International Conference Image Analysis and Recognition (ICIAR), 2017.

1.5.3 Additive Depth Maps, a Compact Approach for Shape Completion of Single View Depth Maps

We present a new data representation for the task of 3D shape completion of single view depth maps. This new data representation, which we call the "additive depth map", is an image where each pixel represents the amount of depth needed to transform the input depth map into the "back" depth map which would exist if there was a sensor exactly opposite of the input camera. A union of the input and back depth map is then the completed 3D shape. The advantage of using our representation is that it is data

minimal and thus traditional 2D convolutional neural networks (CNNs) can be trained to predict the additive depth map. Additionally since it is data minimal, we can train with higher resolutions when compared to the existing 3D CNN approaches allowing us the potential of completing the shape of objects with more detail.

To accomplish our task of 3D shape completion using the additive depth map, we employ a residual encoder-decoder network with skip connections as the overall architecture. We train, and benchmark, our network using existing synthetic datasets as well as real world data captured from commodity depth sensors. Our experiments show that the additive depth map can produce comparable results to existing state-of-the-art 3D CNN approaches and can be applied to real world data without fine-tuning. We also experimented with including semantically labeled images into the training pipeline by making a small modification to our network. Our experiments in this direction showed that incorporating semantic information allows for stronger predictions - especially for more heavily occluded inputs.

A more detailed analysis of this contribution can be found in Chapter 6. The research endeavors in this area led to the following publication:

P. Lai, W. Liang and R. Laganière. *Additive depth maps, a compact approach for shape completion of single view depth maps*. Graphical Models. Volume 104, 2019.

1.6 Thesis Outline

The rest of this thesis is structured as follows. Chapter 2 describes the properties of an immersive VR scene and details the building blocks of a potential system (with two variants) which can produce these scenes from a single sensor. Chapters 3 and 4 provide a literature review and a concrete application for two of the main building blocks described in Chapter 2: separating the background from the foreground to track the foreground objects and visual SLAM for building a background model. In Chapter 5 we cover an implementation of the system described in Chapter 2 and in Chapter 6 we detail an approach for 3D shape completion to handle occlusions inherent to single sensors. A summary and potential future work is provided in Chapter 7.

Chapter 2

A System for Immersive Virtual Reality Videos

In this chapter we first identify the properties of an "immersive" virtual reality (VR) scene (Section 2.1). Next, we cover the proposed system and each of the core components required to produce an immersive VR experience using a single RGB-D camera. In Section 2.2, a general overview of the essential building blocks of our proposed system is provided. Next, we present two variants of our system: one which leverages a known database (Section 2.3) and another which does not make use of any prior knowledge of the scene (Section 2.4). The common concept shared between the two variants is the detection, segmentation and camera pose estimation of all moving foreground objects in order to place them into a captured 3D model of the scene. Since the background model and the RGB-D video frames are 3D objects, the user is allowed 6 DoF to observe the

scene from any viewpoint as well as interact with the reconstructed static and dynamic elements of the scene.

2.1 Properties of an Immersive VR Video

The terms "immersion" and "presence" in the context of VR are often used interchangeably with the goal being to quantify the combined degrees of realism, interactivity and subjective feelings from a virtual experience. In the late 90s, Slater and Wilbur [144] distinguished between immersion and presence for VR by separating the technological components from the subjective feeling and behavioral aspects of experiencing a virtual environment (VE). More precisely, they defined immersion as a description of technology, specifically computer displays, which were capable of delivering an inclusive, extensive, surrounding and vivid illusion of reality to the senses of a human participant. Presence was then defined as how strongly the participants felt that they "belonged" in the VE. Thus, a person who felt strong presence within a VE would consider it more engaging than the surrounding physical world. These subjective feelings can be observed physically as a person who feels strong presence would move and behave as though they were in reality. For example, if an object was moving quickly towards a user who felt strong presence in a VE, it would invoke the users instinctual physical reaction to move out of the way. If a user pushes an object, it should move within the VE in a manner which is consistent with the rules.

In order to more precisely define "immersive VR video", we take inspiration from the framework as described by Slater and Wilbur [144]. Immersion for VR videos would then describe the technical components which would allow for an inclusive, extensive, surrounding and vivid illusion of the real world environment as captured by visual sensors. As the framework by Slater and Wilbur was originally designed to describe computer displays, a direct application does not make sense. Instead, we take the spirit of each element in their framework to derive our own versions for immersive VR videos. Listed below are brief descriptions of the original elements:

Inclusive (I) The extent to which the computer display shuts out physical reality.

Extensive (E) The range of sensory modalities accommodated by the display.

Surrounding (S) The extent to which the viewable regions are panoramic rather than limited to a narrow field of view.

Vivid (V) The level of resolution, fidelity, and variety of energy simulated within a particular modality (for example, the visual and colour resolution). Vividness is concerned with the richness, information content, resolution and quality of the displays.

Based on the original descriptions as defined above, we derive our own definitions for VR videos. Listed below are brief descriptions and the intuitive reasoning behind our choice of definitions:

Inclusive (I) The extent to which the background environment, as captured by visual sensors, is visible within a VR video. The intention behind the original definition was to minimize distractions from physical reality in order to increase engagement with the user in the VE. For VR videos, we are already within a VE and thus the distracting elements would then be the empty space. This is akin to having portions of a standard 2D video missing/corrupted or censored with a black bar.

Extensive (E) The range of sensory modalities accommodated by the VR video. Our definition is not much different than the original as our intentions are the same - having all five senses engaged would lead to the most immersive VR experience.

Surrounding (S) The extent to which the viewable portions of the VR video allow for a panoramic (rather than a limited narrow field of view) experience. Similar to the original definition with the exception that we are already within a VE. Thus it is the reconstructed VR scene, rather than the computer displays, which should accommodate a range of head motion from the user by displaying the appropriate portions of the captured scene.

Vivid (V) The level of detail, responsiveness and completeness of the captured static and dynamic elements from the VR video. As we are already in a VE, responsiveness refers to how well the static and dynamic elements respond to external stimulus (i.e: input from motion controllers). A reconstructed scene which is highly responsive to touch in addition to being detailed and complete would then provide a highly vivid experience.

Using the above definitions as guidelines, we identify several properties which must hold for a scene captured using visual sensors to be considered "immersive" in VR:

1. The background of the scene must be displayed to the viewer. While it may be compelling to display just the dynamic objects, without the background there is no context.
2. The dynamic objects must be consistent with the captured background. If there is no correlation between the dynamic objects being displayed and the background, then the captured scene does not reflect the viewer’s expectations.
3. The viewer should retain full rotational and positional freedom while experiencing the captured scene. Without this freedom the viewer will experience an inconsistency between their movements and what they are seeing through the HMD.
4. The captured scene, background and dynamic objects, should be as complete as possible.

The first property addresses the inclusive (I) component of the immersive VR video framework - missing background elements would then be similar to blanking out portions of a 2D video. The second property addresses both the inclusive (I) and vivid (V) components of the framework. If dynamic elements are not consistently displayed with the background then the completeness of the scene is questionable as the viewer would not know where or why these elements come into play. Additionally any inconsistency would then be a distraction. The reduction of these distractions is the main intention behind the inclusive (I) component. The third property addresses the extensive (E) and surrounding (S) components of the framework. Having a full range of motion means that the visual sensory modality is well accommodated and it also allows the user to have a panoramic viewing experience. Finally the fourth property deals with the inclusive (I), surrounding (S) and vivid (V) components of the framework. Having both the background and dynamic objects be as complete as possible ensures that the distractions are minimal, the reconstructed environment surrounds the user which in turn provides a more vivid experience. Failure to achieve these properties decreases the quality of the immersive experience.

2.2 System Overview

The input to our proposed system is a set of RGB-D video frames, $\mathcal{F}_{input}$, all taken from a single RGB-D camera. Next, a foreground background separator splits $\mathcal{F}_{input}$ into $\mathcal{F}_{bg}$ and $\mathcal{F}_{fg}$ where the contents of $\mathcal{F}_{bg}$ only contain the background scene and the

contents of $\mathcal{F}_{fg}$ contains foreground objects moving within the environment. A static 3D background model of the scene, $\mathcal{M}_{static}$, is created from $\mathcal{F}_{bg}$ while the foreground objects in $\mathcal{F}_{fg}$ are superimposed onto $\mathcal{M}_{static}$. This gives us the following basic building blocks for our proposed system:

1. A 3D model of the scene.
2. A foreground and background separator.
3. Localize segmented foreground objects to the scene.

In subsections 2.2.1, 2.2.2 and 2.2.3 we describe each of the above building blocks in more detail in terms of methods which can be used and how it relates to the final product. Note that since we are only using a single RGB-D camera, many occlusions will occur and thus all of the foreground objects in $\mathcal{F}_{fg}$ will be incomplete. An optional building block can be added to the above in order to remedy this issue. An overview for this particular building block applied for people extracted from $\mathcal{F}_{fg}$ is described in 2.2.4 with the details in Chapter 6. Fig. 2.1 illustrates the proposed system through a flowchart.

2.2.1 Building a 3D Model of the Scene

The 3D model of the scene serves as the background in which a viewer can move through with six degrees of freedom. Creation of such a model using a single RGB-D sensor is typically accomplished through photogrammetry [42], volumetric fusion methods [112, 63], and general dense visual simultaneous localization and mapping (SLAM) [69, 34]. In the case of photogrammetry and some of the dense visual SLAM methods, using just the RGB data would suffice. An overview of visual SLAM as well as a brief survey of RGB-D SLAM methods is provided in Chapter 4.

While a 3D model of the scene provides the viewer with an explorable environment, it is not just for display. During the model building process important information such as feature correspondences and the camera poses of the input image sequence is computed. This additional information allows for the localization of images which were not part of the original image sequence used to build the 3D model. Thus, newly captured image sequences within the same scene could then be superimposed onto the 3D background model.

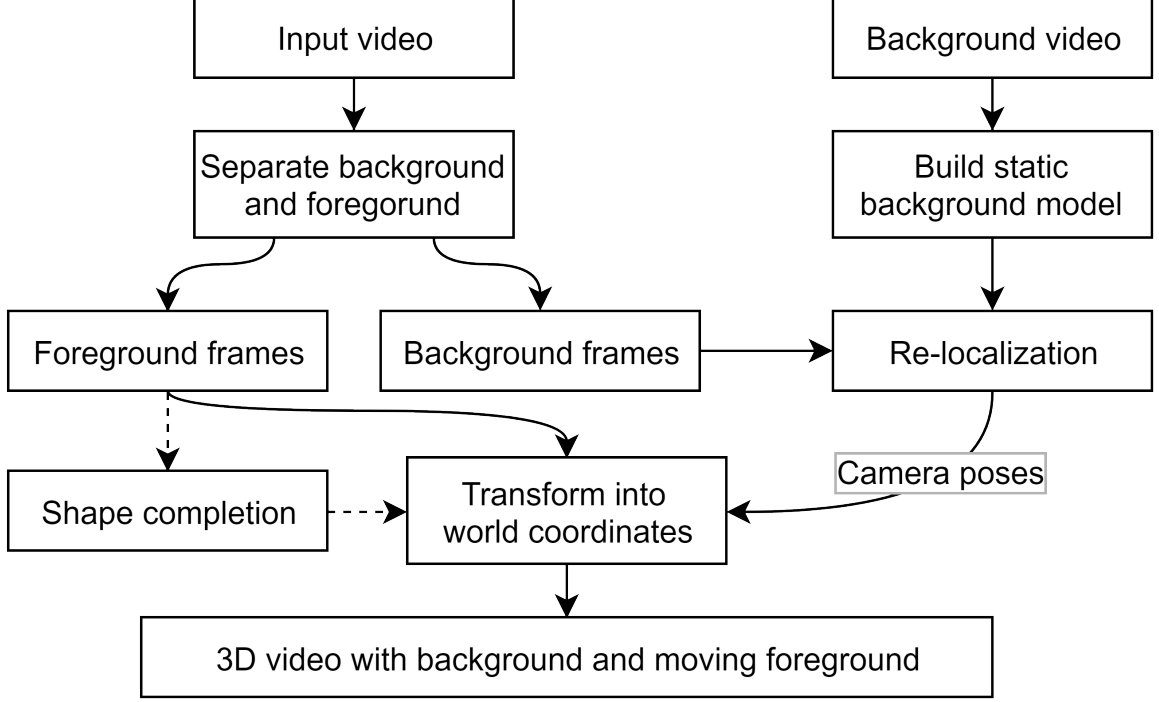


Figure 2.1: Flowchart of our system for building immersive experiences for VR using a single RGB-D camera. The dashed arrows corresponds to the optional feature of occlusion filling for people trajectories as described in Section 2.2.4 and examined in Chapter 6.

2.2.2 Separating the Foreground and Background

Simply localizing an input video frame f^i to the background model $\mathcal{M}_{static}$ has a side effect of superimposing the background image data onto $\mathcal{M}_{static}$. This is undesirable for display purposes as $\mathcal{M}_{static}$ already models all the static portions of the scene. Additionally, any foreground objects which are present in f^i may add some error to the localization step (Section 2.2.3). However, if enough of the background is visible then the fundamental matrix can still be estimated by using feature correspondences. This means that while a foreground/background separator is not strictly required so that only the background image data is used to localize f^i with respect to $\mathcal{M}_{static}$, it is still needed so that only the foreground image data is superimposed onto $\mathcal{M}_{static}$ for display.

The topic of background subtraction and foreground object segmentation is well studied and Chapter 3 provides a survey of the literature. It should be noted that most of the methods in Chapter 3 tackle the problem by assuming a static camera with the separator

operating online and in real-time. Since our goal of creating immersive experiences has no real-time component we can process the entire video sequence offline.

2.2.3 Localizing Foreground Objects

Once a background model, $\mathcal{M}_{static}$, is available and each of the RGB-D video frames $f^i \in \mathcal{F}_{input}$ have been separated into a background $f_{bg}^i \in \mathcal{F}_{bg}$ and foreground $f_{fg}^i \in \mathcal{F}_{fg}$, localization can be performed. Different localization approaches exist such as using a feature database for image matching [17, 9] and feature point based matching [131, 25]. Both of these approaches can then be followed up with sparse bundle adjustment [97] to optimize the camera poses. Precisely which method used for localization depends on how the model $\mathcal{M}_{static}$ was built as we would like to leverage any information already computed. Once $f_{bg}^i \in \mathcal{F}_{bg}$ has been localized with respect to $\mathcal{M}_{static}$ we can use the camera pose of $f_{bg}^i \in \mathcal{F}_{bg}$ to superimpose $f_{fg}^i \in \mathcal{F}_{fg}$ onto $\mathcal{M}_{static}$ resulting in an immersive experience.

2.2.4 Filling in Occlusions of Detected People

One of the most interesting and important foreground objects for an immersive viewing experience are people. Since our final output will be a fully 3D video which the viewer can observe from any viewpoint, it would be ideal to fill in the occlusions which occur due to the single RGB-D camera limitation of our system. In order to fill in these occlusions we must first separate the people from the other foreground objects that were detected in Section 2.2.2. This can be accomplished in just the RGB space through people detectors [117, 107] with the addition of depth in more recent methods [146]. Once the individual people have been separated their trajectories over time can be built through tracking or clustering. These trajectories would allow us to use all of the frames which may contain an alternative viewpoint of the person to help fill in the occlusions. Optionally we can also estimate a skeleton to aide with the people trajectory building phase. Having a skeleton estimated per person would also help with occlusion filling as it provides some geometric constraints on the person. More precise implementation details for the shape completion component is discussed in Chapter 6.

2.3 Database Solution

The basic building blocks described in Section 2.2 can be modified to take advantage of a known database. This database would contain a 3D background model, $\mathcal{M}_{static}$, which is the same scene that is visible in the background of the input RGB-D video. Thus a background video is not required and through $\mathcal{M}_{static}$ (which should be more complete and detailed) we can obtain camera poses for each video frame via re-localization. The separation of the foreground and background can now be accomplished by thresholding on the Hausdorff distance [106] between $\mathcal{M}_{static}$ and the localized RGB-D video frames. Transformation of the foreground frames remains unchanged from Section 2.2.3. The database solution can be thought of as the simplified version of the framework as $\mathcal{M}_{static}$ is already provided. Fig. 2.2 illustrates the complete workflow of the database solution.

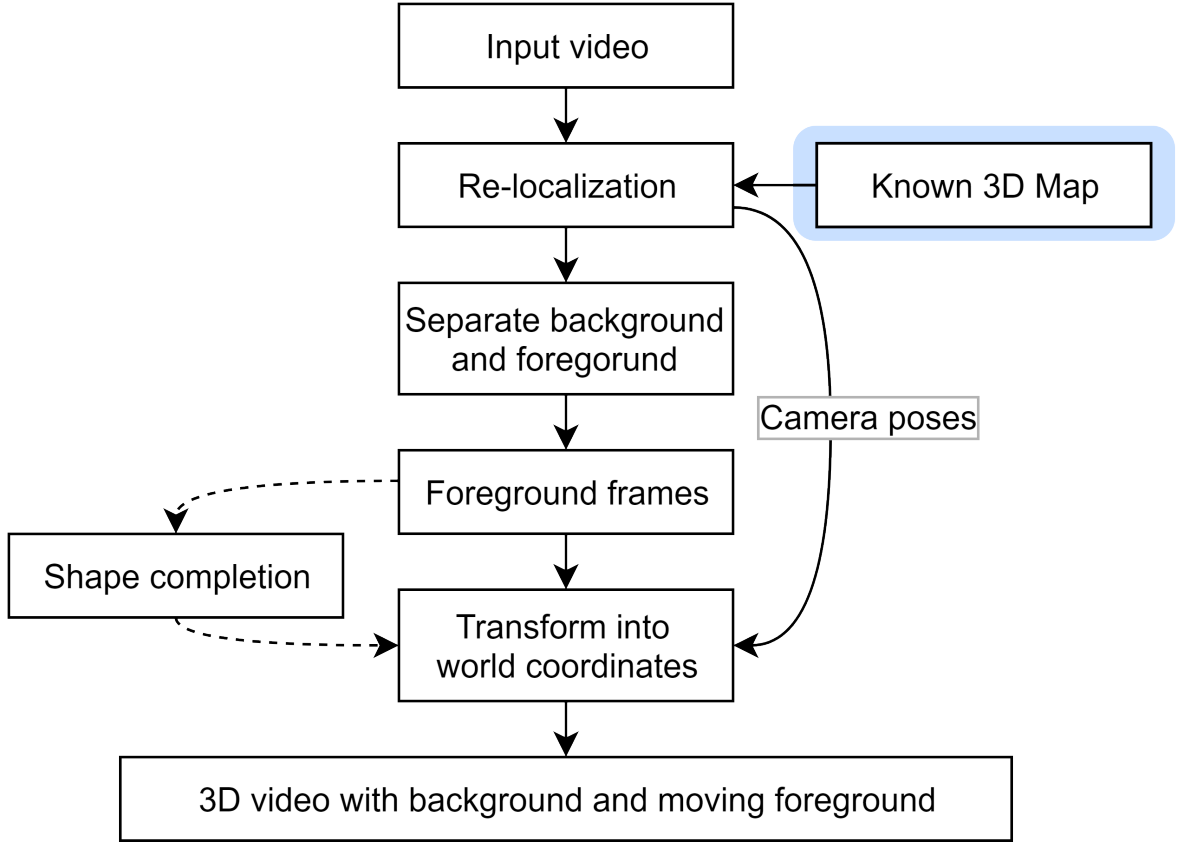


Figure 2.2: Flowchart of the database solution obtained by augmenting Fig. 2.1 with a database containing the known background model.

2.4 Generalized Solution

In this variant we are only given a single set of RGB-D video frames, $\mathcal{F}_{input}$, and thus we must both build the background model (Section 2.2.1) and localize the positions of all of the moving objects (Section 2.2.3) at the same time. This can be accomplished by first detecting and segmenting the moving foreground objects from the background (Section 2.2.2) and then using only the background image data to produce a model, $\mathcal{M}_{static}$, of the scene through dense visual SLAM. This results in a one-to-one mapping between the camera poses assigned to each background frame and its corresponding foreground portion. Using these camera poses we can then transform the segmented foreground objects such that they are consistent with $\mathcal{M}_{static}$.

The main challenge unique to the generalized solution is the requirement of segmenting foreground objects from a potentially moving camera. In the foreground and background separator building block as described in Section 2.2.2, segmentation of foreground objects can be performed after the RGB-D video frame, f^i , is localized to a known background $\mathcal{M}_{static}$. This allows the segmentation method to take advantage of the fact that the background of f^i is spatially "close" to $\mathcal{M}_{static}$. Thus a threshold on the Hausdorff distance [106] between f^i and $\mathcal{M}_{static}$ can be used to remove only the background image data from f^i leaving the foreground objects intact. The generalized solution does not have such a luxury as it must build $\mathcal{M}_{static}$ and segment the foreground objects from f^i at the same time. Fig. 2.3 modifies the right portion of Fig. 2.1 to illustrate the generalized solution workflow. Some methods for foreground object segmentation and background subtraction are covered in Chapter 3 and potential solutions specifically for this variant of our proposed system is discussed in Chapter 5.

2.5 Summary

In this chapter we first defined the "immersive VR video" by adapting an existing framework which was used to describe immersive computer displays. Next, an overall overview of the proposed system was provided. Then, a brief description of each main building block from our proposed system was given in terms of what methods can be used and how it will relate to the final product. Two variants, one which takes advantage of a database and one which does not assume any prior knowledge and, are introduced.

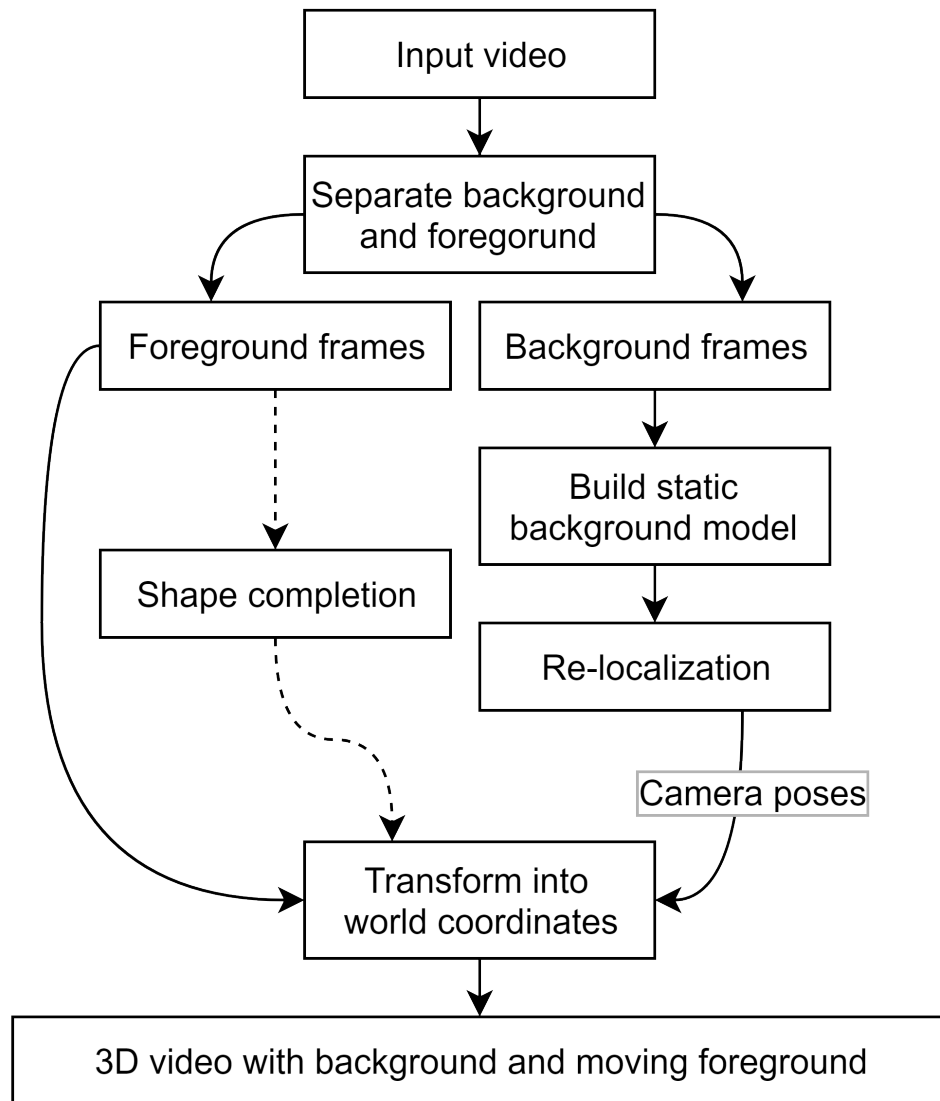


Figure 2.3: Flowchart of the generalized solution obtained by removing the requirement of a background video that produces a 3D Model for camera pose estimation from Fig. 2.1.

Chapter 3

Background Subtraction

Background subtraction is often used as a building block in larger computer vision problems. In relation to this thesis, we explore background subtraction as a method for extracting arbitrary foreground object trajectories. Such a method is of value for the immersive VR scene creation system as described in Chapter 2 because it would allow us to obtain a set of video frames which contain only the background data. These frames can then be feed into existing 3D reconstruction algorithms (Chapter 4) to produce the static 3D background model that serves as a basis of the VR scene. In this chapter we will first provide a brief literature review (Section 3.1) on background subtraction before describing an application which utilizes background subtraction algorithms to produce a video summary (Sections 3.2 to 3.4). The methods presented in the video summarization application can also be used to extract foreground trajectories for VR scene creation. Finally, we summarize in Section 3.5.

3.1 Literature Review - Background Subtraction

In general, background (BG) subtraction algorithms typically consists of two main steps: (1) the BG model initialization and (2) updating the model. An additional preprocessing step is sometimes applied before an image is considered for steps (1) and (2). This preprocessing step typically smooths, thresholds or applies some morphological operators to remove noise or condition the image for better results. For BG subtraction to be effective, it must be able to handle the following scenarios:

Changes in BG geometry: BG objects can be moved but afterwards should not be considered to be a foreground (FG) object forever. Thus the BG model should adapt with the understanding that the scene layout can be physically updated.

Gradual illumination: Slow changes which can occur when a video is recorded from day time to night time which gradually alters the appearance of the BG globally.

Sudden illumination: Changes such as a light switch turning on indoors or from street lamps outdoors and shadows cast from moving foreground objects. This scenario is different and more challenging than the gradual illumination scenario as the BG is changed suddenly which is a characteristic that is typical of a foreground entity.

False positives: In real life situations the BG is typically not completely static. For example, a tree which sways due to wind is an example of small movement that would be considered as a false positive.

False negatives: If a region is considered to be part of the BG but a FG object, with similar pixel characteristics as the BG, moves in front of the region then false negatives may occur. Another false negative scenario would be when an object is homogeneously colored. The interior pixels of the object may not be properly detected when the object moves producing a partial FG mask.

To tackle the challenging scenarios as described above, many different BG subtraction algorithms have been explored. Different categorizations for these algorithms have been proposed in the literature. For this brief survey we follow the taxonomy proposed by Cristani et al. [15] who separated BG subtraction algorithms into the following:

- *Per-pixel:* Each pixel signal is considered an independent process thus these approaches require minimal computation and are suited for real-time applications.

- *Per-region*: By dividing the input video frames into blocks more information can be gained allowing the background model to handle regions in images which locally oscillate or contain slight movements. In general, this means that a more robust visual description of the scene can be computed allowing for the minimization of false positives such as swaying trees.
- *Per-frame*: The more challenging problems, such as sudden illumination changes, are difficult to handle using the two previous approaches. Extending the local support provided by per-region methods to the entire frame global can alleviate some of those challenges.
- *Multi-stage*: Methods which combine the previous approaches together in order to refine the final result fall into this category.

In general, while per-pixel algorithms provide an excellent compromise between accuracy and speed, there are some major drawbacks [15, 119]. Since per-pixel methods do not take into account the global view of the scene, scenarios such as sudden illumination changes will cause false positives. Compared to per-pixel methods, per-region methods exhibit higher complexity in both time and space. Despite this, most approaches in the per-region claim real-time performance. Most of the methods in the per-frame category use an offline training step to model elements of the background model. A classification step is then deployed for real-time background subtraction.

One of the most commonly used per-pixel BG subtraction algorithms, which we utilize in Section 3.2, is the Mixture of Gaussians (MoG) [43, 149, 163, 180]. In these approaches, the state of a pixel is statistically modelled as a multi-modal signal described as a time-adaptive mixture of Gaussian components. Each Gaussian component of the mixture represents a gray level interval observed at a given pixel location. A weight, representing the confidence that it is part of the background, is assigned to each component. The higher the weight, the stronger the confidence and thus the longer a given gray level has been recently observed at that pixel location. The MoG model has been further improved by Wang and Suter [163] and Zivkovic [180] with the approach by Zivkovic [180] implemented in OpenCV [60].

3.2 Application: Video Summarization of Surveillance Cameras

In this section we present one of the contributions of this thesis which led to a publication [80]. The contribution is an approach for producing a *video summary* from static surveillance cameras (Section 3.2) and new evaluation criterion (Section 3.3) which aimed to quantify the summarization quality. A video summary is an abbreviated video which preserves the important elements of the original video while removing the spatio-temporal segments which are not of interest. In our work we assume the scene is composed of moving objects observed from a fixed camera and our goal is to produce video summaries that retain as much of the moving objects as coherently as possible. In this way, the interpretation of video events can be preserved. In order to be relevant and coherent, a video summary must adhere to a number of requirements. First, spatial locations and trajectories of objects must not be modified. Second, rules of occlusions must be applied (e.g: closer objects should occlude those which are farther away). Third, temporal ordering must remain unchanged. That is in a shortened version of a video, even if the timing between objects is changed, the order in which they appear should remain as in the original video. Fourth, the speed at which objects move should not be altered or, at least, not too greatly exaggerated. Failure to adhere to these guidelines may drastically change the way a video event is interpreted.

When a temporal sequence of images is viewed as a 3D data structure [68], the moving foreground objects form 3D spatio-temporal shapes, sometimes called *tublets*. These spatio-temporal objects can then be manipulated independently from the background sequence in order to produce an interpretable video summary. The approach presented in this section therefore aims at producing compact video summaries by extracting the trajectories of each object of interest and optimizing their arrangement in the summarized video. Section 3.3 covers the benchmarking dataset used in this work as well as our proposed evaluation measures for quantifying the quality of a video summary. Section 3.4 then provide a discussion on the merits of our approach.

3.2.1 Problem Definition and Related Works

Given an input video $\mathcal{V}_{input}$ the objective of a video summarization algorithm is to output another video $\mathcal{V}_{output}$ such that $|\mathcal{V}_{input}| < |\mathcal{V}_{output}|$, where $|x|$ represents the length of a

video x in frames. Additionally, the "interesting" content in $\mathcal{V}_{input}$ should be retained in $\mathcal{V}_{output}$. With the fixed camera prior, video summarization algorithms can be classified into three categories: fast forwarding, key frame selection and frame re-composition. We will focus on the frame re-composition methods since they are most related to our approach.

Related Works

Frame re-composition approaches construct video summaries by selecting relevant spatio-temporal segments and combining them into a new spatio-temporal scene. This process does not preserve the integrity of each individual frame. Pritch et al. [124, 120, 121] have published several different methods that follow this approach. In [124], they condense a video by showing several actions simultaneously, even if they originally occurred at different times. Multiple dynamic instances of the same moving object are shown simultaneously if the object moves slowly. In [120], they add an index into the original video by referencing the original time of each activity. An online conversion process takes the video stream and translates it into a database of objects and activities. Finally a video summary is generated based on the user's query. In [121], they combine action recognition with video summarization to cope with the fact that, while effective, the simultaneous display of multiple activities may produce confusing summaries. A methodology for the generation of short and coherent video summaries is presented, based on clustering of similar activities.

Other approaches exist based on an adaptive version of seam carving, a content aware image resizing algorithm. Instead of using this algorithm for spatial reduction, it is used to obtain a temporal reduction. Li et al. [93] present ribbon carving which iteratively suppressed continuous spatio-temporal seams and combine together widely separated dynamic events. This approach has some limitations when the speed of the objects becomes too high or when multiple objects are moving at different speeds and/or directions. Décombas et al. [22] propose an alternative to solve these problems. They first define the seams that can be suppressed and a spatio-temporal constraint to limit, but not eliminate, the temporal artifacts that could appear.

3.2.2 Overview of Proposed Approach

We follow the frame re-composition strategy used by Pritch et al. [124, 120, 121] and consider a video sequence with frame dimensions of (X, Y) and length T to be a "video

cube" with dimensions (X, Y, T) as in Keimel et al. [68]. Unlike the approaches by Pritch et al., who either assumed the object trajectories were given [124] or used only a background subtractor to build object trajectories [120, 121] (which often resulted in broken trajectories), we concentrated our efforts on building complete and coherent object trajectories which can include static objects (e.g: a loitering pedestrian). Additionally, we manage how these object trajectories are summarized to obtain a result that preserves instances when two objects cross each other in the original video. If these "crossing" events are not managed, then it's possible to have a video summary which shows occluded objects without the context that another object was passing in front of it.

To robustly and precisely extract the relevant spatio-temporal trajectories in the video we combined a background subtractor [62], dense optical flow [37] and a pixel clustering algorithm. These trajectories can be independently manipulated to produce a short video summary through minimizing the space between these 3D shapes. More precisely, we use projections of the video cube to condense the space between trajectories. This allows several objects from different time frames to be seen at the same time. As slow and static objects can increase the length of the video summary by reducing the free space for other objects, we also applied an adaptive motion acceleration for each trajectory. A timer was added on each object to highlight the original time frame of the object. Fig. 3.1 illustrates the video summary construction process and provides the global work flow for our approach. Sections 3.2.3 to 3.2.5 describe the details of each component as shown in Fig. 3.1.

3.2.3 Detecting Foreground Objects

Detection and segmentation of all moving foreground objects was achieved using a combination of background subtraction [62], dense optical flow [37] and a pixel clustering algorithm. The clustering of optical flow vectors separates pixels which are moving in different directions while background subtraction separates pixels spatially. The fusion of these two is then a set of pixel groups which represent foreground object detections.

To separate pixels using the dense optical flow output, we modified a general data clustering algorithm known as gravitational clustering [21]. This algorithm takes as input a set of n data points in d dimensions and initially treats each data point as a separate cluster. Two points, p_i and p_j , are randomly selected and moved closer together according to the force of gravity. If p_i and p_j are within a threshold Euclidean distance then they are

considered to be in the same cluster. This act of selecting two points and moving them closer is repeated until all n data points have been selected. When all n data points have moved, then one iteration of the clustering algorithm is completed. Over k iterations the final clusters are extracted. We modified the algorithm to use the angular distance rather than the Euclidean distance for moving data points and determining if two data points (pixels in our case) should be considered as being part of the same cluster. In this case each pixel can then be represented as a polar coordinate (θ, m) where θ is the flow direction and m is the flow magnitude. By plotting these points on the perimeter of a circle centered about the origin, the angular distance between two points is found through the inner product. Moving points towards each other can then be thought of as moving along the perimeter of the circle.

Once we have a set of pixel groups from the clustering of dense optical flow, these pixel groups are further subdivided by the output from background subtraction. Since background subtraction can provide us with groups of pixels which are spatially separated, it will divide a group of pixels which are all moving in the same direction spatially. The final result is a set of pixel groups each of which represents a moving foreground object in the current frame.

3.2.4 Building Spatio-Temporal Trajectories

To build spatio-temporal trajectories we first receive as input the detected foreground objects per frame via the method described in Section 3.2.3. Second, we associate each of these foreground object detections in consecutive frames through a sliding temporal window. Next, static objects are detected and associated with the currently tracked trajectories. Finally collision management is required to ensure that the trajectories are coherent and not occluded.

Foreground Object Association

A sliding temporal window of k frames was used to associate detected objects in consecutive frames with each other thus building a spatio-temporal trajectory for a single active object in the input video sequence. An association is made once a detected object has been matched against the last known detection from each trajectory which has a moving object (i.e.: trajectories with static objects are excluded) and the pixel overlap between them is greater than a threshold. Once a trajectory no longer has any active object detection associations within the sliding temporal window, it is marked for further

inspection to determine if the tracked object has become static or out of frame. In the latter case the trajectory is marked as completed if it is longer than a given number of frames. The former case is handled by the method described below.

Managing Static Objects

When foreground objects stop moving before exiting a scene they become “static objects”. That is, they are objects which are still of interest for summarization but do not move and thus our method of detecting foreground objects will not suffice. In order to detect these static objects a background is defined from the last frame with no foreground objects (static or otherwise). As noted in the foreground object association phase, a trajectory, t , with no additional foreground objects associated with it appears to have either become a static object or has exited the scene. To detect if t has become a static object we do a pixel-wise difference between the padded bounding box of the last known location of the foreground object and the background. If a large number of pixels differ from the background pixels, the object is defined as static. Furthermore, this static object must correlate to the trajectory t as this check is only performed when there are no longer any new active foreground objects matching the trajectory. The pixel-wise difference test is performed on every frame until the object moves outside the padded bounding box where the previous method applies once more.

Accelerating Trajectories

Adaptive motion acceleration of the output trajectories was applied to increase the speed of slow moving objects without much loss to intelligibility. This was accomplished by determining the mean speed of active objects and setting that as their summary playback speed. Since static objects have no motion, they are accelerated by keeping every N^{th} frame. In our experiments we have chosen $N = 10$. Note that fast forwarding is only applied to the frames when only the static objects are visible as we would like to minimize the loss of intelligibility. Thus if any active objects cross the static object or if the static object becomes active again, the previous method of acceleration applies.

Handling Collisions

One final consideration is required before the complete set of spatio-temporal trajectories can be used as the 3D shapes of the summarization algorithm. When two (or more) objects enter the scene, cross paths and exit the scene a decision must be made: treat

all these objects together as one spatio-temporal trajectory (i.e: merge these objects) or treat each object as a separate spatio-temporal trajectory. The former was chosen as the latter results in trajectories where the occluded object will only be partially visible. This decision results in large spatio-temporal trajectories in the case of a scene with a large number of crossing objects and thus the summary may not be as optimal. In essence, we trade improved visualization and ease of understanding at the cost of a potentially longer summary.

3.2.5 Summarization

After having obtained all the trajectories, we recombined them based on their spatio-temporal projection on the video cube planes (x, y) , (x, t) and (y, t) . Each trajectory is first projected on the (x, y) plane and brought as close as possible to the already integrated projections of the trajectories in the (x, y) plane. This allows multiple objects from different spatio-temporal locations whose trajectories never cross (in both space and time) to be seen at the same time. Multiple objects who follow similar trajectories at different time frames produce collisions on the (x, y) plane. To handle these collisions we project those trajectories onto the (x, t) plane and then bring them as close as possible temporally (see the last box in Fig. 3.1). The same is done for the (y, t) plane. This results in objects from different time frames appearing one after another. Textual information detailing the timestamp of the object in the original video can be added in the summary to help the user understand the final summary. Finally any trajectories which were accelerated can be identified through a different timestamp color to inform the user that acceleration of the object has taken place. To further decrease the length of a video summary overlaps between the spatio-temporal trajectories of objects are allowed. The precise amount of overlap is controlled through a threshold representing the percentage of overlap allowed between the projections of two trajectories.

3.3 Dataset and Evaluation

Evaluation of video summaries is difficult as the quality of a summary is dependent on the application and target users. To that end, user studies are often employed to measure how well a video summarization algorithm subjectively performs for a specific application and target user. As the goal of our video summarization approach was to retain as much of the moving objects as possible, we evaluate our approach in a more objective

manner. To this end we introduce new evaluation measures (Section 3.3.2) that can be used for estimating the quality of a video summary. These measures aim at evaluating how well a given video summary captures the relevant activities in a surveillance video. Our experimental evaluation is performed on a dataset of surveillance videos for which appropriate ground-truth annotations have been added (Section 3.3.1).

3.3.1 Dataset

To validate and compare the efficiency of our approach with traditional methods, a database of five sequences each accompanied with a manually created binary mask of all the activity pixels in the corresponding sequence is introduced ¹. Fig. 3.2 shows for each sequence an example of the original sequences and the binary mask associated with them. Table 3.1 summarizes the basic information (name, dimensions, frame rate, total length, number of foreground objects and brief descriptions of features present) of each sequence.

3.3.2 Evaluation Criteria

We define activity pixels as those which are important and thus should be included in a video summary. These activity pixels were identified through manual labeling of the dataset described Section 3.3.1. Listed below are our proposed metrics and how they are measured:

Compression rate

This is simply the amount of reduction in video length relative to the original video.

It is defined as:

$$CR = 1 - \frac{L_{summary}}{L_{original}} \quad (3.1)$$

where CR is the compression rate, $L_{summary}$ is the number of frames in the video summary and $L_{original}$ is the number of frames in the original video.

The CR is zero when the video summary has the exact same length as the original video.

Activity recall

The total percentage (relative to the original) of activity pixels which are visible in the final summary. Simply counting the number of pixels in each trajectory

¹<https://drive.google.com/open?id=0ByLzTb0oFmpvbFFHSG1Tc3dGQjA>

<i>ID</i>	<i>Sequence name</i>	<i>Dimensions</i>	<i>FPS</i>	<i>Length</i>	<i># of objects</i>	<i>Descriptive features</i>
A	Cesta4 [161]	360×288	25	0:49 min	12	Linear horizontal movement along a road in both directions.
B	Building	960×540	24	4:20 min	5	Objects enter from center of video frame. Movement is non-linear.
C	BuildingStatic	480×270	24	6:06 min	16	Same as Building sequence with addition of a static object with multiple objects crossing it.
D	Transitway1	480×270	24	3:15 min	14	Diagonal movement along a road in both directions. Background contains some subtle movement.
E	Transitway2	480×270	24	2:50 min	16	Same as Transitway1 with addition of overlapping objects.

Table 3.1: Basic information of each sequence in our dataset. A unique identifier is given for each sequence in the "ID" column which we refer to in Section 3.4 and Fig. 3.5 when discussing our results.

that intersect with the ground truth activity pixels is insufficient since it is possible to have overlapping trajectories in the video summary (Section 3.2.5). Thus the activity pixels from overlapping trajectories should only be counted once since a viewer cannot see two (or more) overlapping pixels at the same time. This gives us the following definition for activity recall (AR):

$$AR = \frac{P_{retained} - P_{overlap}}{P_{reference}} \quad (3.2)$$

where $P_{retained}$ is the number of activity pixels retained before summarization occurs, $P_{overlap}$ is the number of activity pixels which are overlapping with other activity pixels in the video summary and $P_{reference}$ is the total number of activity pixels in the ground truth.

The AR ranges from zero to one where the value zero implies that no activity pixels

from the ground truth were found in the video summary while the value one implies that all activity pixels from the ground truth were found in the video summary.

Activity density

The average percentage of a single frame that will be filled with activity pixels. For example if 10% of all the pixels in a video are activity pixels, then we would expect that, on average, any given frame should have 10% of its pixels be activity pixels. A video summary should have an increase in activity pixels per frame as we are removing irrelevant frames and filling empty locations within other frames which might already contain activity. We define activity density (AD) as follows:

$$AD = \frac{P_{retained}}{W \times H \times L} \quad (3.3)$$

where $P_{retained}$ is defined as in the activity recall metric, W and H are the width and height of the video and L is the number of frames.

3.4 Results and Discussion

In this section we present our results through three different perspectives: (1) summarization perception, (2) summarization evaluation and (3) spatio-temporal trajectory building performance. In Section 3.4.1, the visualizations of the final video summaries produced from our proposed approach are presented and examined. In Section 3.4.2, we utilize the evaluation measures defined in Section 3.3.2 to benchmark how well our proposed approach performs. Finally in Section 3.4.3 we examine the performance of our object tracker defined in Section 3.2.4 which was used to build the spatio-temporal trajectories.

3.4.1 Summarization Perception

As noted in Section 3.3, how well a video summary captures and compresses information as well as the ease of understanding all actions in the summary can be subjective as it depends on the application and target users. While we have proposed some metrics in Section 3.3.2 to reduce subjectivity, the analysis of the final result would not be complete without some subjective discussion on the results of our proposed approach. Figs. 3.3 and 3.4 visualize a single frame in the final video summary of two sequences.

Fig. 3.3 is from a summarized Building sequence where we can see multiple objects from different temporal locations in the original video (top row) that have been combined

in one frame (bottom row). Fig. 3.4 features two different video summaries of sequence E. In Fig. 3.4a trajectories are allowed to overlap and the timestamps are hidden while in Fig. 3.4b trajectories are not allowed to overlap and the timestamps are displayed. In Fig. 3.4a the overlapping trajectories results in all vehicle trajectories (14 in total) played at once. From this example it is clear that while allowing overlaps decreases the final length of the video summary, the result is undesirable as the summary becomes incomprehensible. While there are fewer objects displayed in Fig. 3.4b than in Fig. 3.4a, we can visualize their timestamps without damaging comprehension in addition to being able to see each object separately in rapid succession.

3.4.2 Summarization Evaluation

Our proposed approach is compared with three different fast forwarding approaches which will be denoted as uniform sampling (UniSamp), non-activity skipping (NAS), and uniform sampling with non-activity skipping (UniSamp-NAS). UniSamp simply keeps every N^{th} frame and for our experiments we set $N = 10$. NAS skips all frames without any activity pixels to summarize. We determined which frames contain activity pixels using the ground truth binary masks of each sequence. UniSamp-NAS takes the result of NAS as the input for UniSamp which effectively combines both approaches. It should be noted that for all sequences using UniSamp, with $N = 10$, results in $CR = 0.9$ and $AR = 0.1$ whereas NAS results in a variable CR whilst having $AR = 1.0$. UniSamp-NAS would thus yield $CR \geq 0.9$ with $AR = 0.1$. The AR and CR values were computed for several variants of our approach and visualized together with the AR and CR values for the fast forwarding methods in Fig. 3.5 as an AR - CR plot. For the rest of this section we refer to the sequences by their ID defined in Table 3.1.

From Fig. 3.5 we can see that in order for any video summary to be more effective than traditional fast-forwarding methods it must have AR and CR values that place it above the blue line defined by the NAS and UniSamp-NAS methods. If a summary is positioned below the blue line, then that means we can select the closest point on the blue line to obtain a variant of UniSamp-NAS which can produce a summary that improves upon the AR and/or CR . Using this observation we found that summaries which placed above and further away from the blue line were often most effective. Fig. 3.5 also shows that summaries produced from our approach out performed standard fast forwarding approaches by offering a good compromise between compression rate and activity recall.

Accelerating trajectories for moving objects had negligible effects for sequences B,

<i>Sequence name</i>	<i>True positives (TP)</i>		<i>False positives (FP)</i>		<i>True negatives (TN)</i>		<i>False negatives (FN)</i>	
	GT	Ours	GT	Ours	GT	Ours	GT	Ours
Cesta4	2.412	2.395	0.000	4.114	97.588	93.474	0.000	0.017
Building	0.118	0.116	0.000	0.314	99.882	99.568	0.000	0.002
BuildingStatic	0.395	0.385	0.000	2.289	99.605	97.319	0.000	0.009
Transitway1	2.061	2.015	0.000	2.645	97.939	95.294	0.000	0.046
Transitway2	2.216	2.183	0.000	2.762	97.784	95.022	0.000	0.033

Table 3.2: True/false positives and negatives represented as a percentage of the total pixels in each sequence for the ground truth (GT) and our spatio-temporal trajectory building method.

D and E as the objects all moved at roughly the same speed while sequence A and C had noticeable differences. Since sequence C also has a static object, three different summaries were produced: no acceleration, static object acceleration only and static object plus active object acceleration. Allowing overlaps had no effect on sequence C since all object trajectories which cross the static object were merged during the building of the spatio-temporal trajectories.

For all sequences, we achieved an $AR > 0.97$ when trajectories were not allowed to overlap and no trajectory acceleration was performed (green squares in Fig. 3.5). When compared to UniSamp, our method results in a large increase in AR and was able to achieve a $CR > 0.7$ for all sequences except for C which had a $CR \approx 0.01$ since the static object was visible for almost the entire duration of the sequence. If we allow acceleration of the static object, the CR jumps to ≈ 0.59 while AR drops to ≈ 0.58 (magenta square in Fig. 3.5). Since we only accelerate the static object when no other objects are visible, there was very little visual difference in terms of missing activity. If we further allow the active objects crossing the static object to be accelerated we achieved a $CR \approx 0.86$ and an $AR \approx 0.2$ (magenta diamond in Fig. 3.5). Despite the large drop in AR , there was little visual difference.

Increasing CR through overlapping trajectories can result in a large drop in AR which can increase the difficulty of interpretation (see Fig 3.4). This was especially noticeable in the sequences with vehicles since all of them move along the direction of the road, only at different times. Thus by overlapping several trajectories, the result was a summary that had several or all vehicle trajectories played on top of each other at the same time.

For example in sequences D and E, allowing trajectories to overlap effectively reduces the summary to the UniSamp-NAS in terms of AR - CR performance (see Fig. 3.5) with the additional difficulty in comprehension (see Fig. 3.4). Compare this to sequence B where allowing overlap increases the CR with a very minor drop in AR which is a much more acceptable trade off.

From the AD metric, and observations of the content in the original video sequences and summaries, we found that summaries which allowed overlapping trajectories had a significant increase in AD relative to the AD of the original video. A large increase implies that more activity pixels were packed per frame and thus the amount of overlaps that occurred must be proportional to the amount of additional AD . For example in sequence E we saw an increase in AD from 0.08 to 0.17 when trajectories overlap. The results of the additional 0.09 of AD , which is more than the AD of the summarized video without overlaps, can be seen in Fig. 3.4a. Since all of the vehicle trajectories overlap each other, the summary is difficult to comprehend. The issues in Fig. 3.4a are even more pronounced in sequence D where we observed an AD increase of over 0.50 when trajectories were allowed to overlap.

3.4.3 Spatio-Temporal Trajectory Building Performance Evaluation

To measure the performance of our spatio-temporal trajectory building method (Section 3.2.4) we examined the number of pixels which were true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) with respect to the ground truth binary masks of each video sequence. TP are pixels which are labelled as activity pixels by both our trajectories and ground truth while FP are activity pixels in our trajectories but not in the ground truth. TN and FN are defined similarly for pixels which are not labelled as activity pixels by our trajectories. Our method for building spatio-temporal trajectories was able to retain 97% of all ground truth activity pixels. Table 3.2 summarizes the results of this evaluation.

While ideally the spatio-temporal trajectories would be same as the ground truth, in the case of video summarization it is more important that the FN are minimized as they will not be included in the final video summary. This means that the greater the amount of FN, the lower the activity retained which leads to an increased difficulty in comprehension as portions of the objects will be missing in the spatio-temporal trajectories. From Table 3.2 we can see that our method produces under 4.2% FP and 0.05%

FN for any given sequence in our dataset. Thus while our spatio-temporal trajectory building method does produce a larger amount of FP, it also produces very little FN and as a result it is rare to see a partial object or broken trajectories in our video summaries.

3.5 Summary

In this chapter a brief survey on background subtraction was provided with a focus on the per-pixel Mixture of Gaussians (MoG) methods. A practical application, video summarization, which utilized background subtraction as a building block was also presented. The main contributions of this chapter is the overall approach for video summarization which includes spatio-temporal trajectory building plus management and the labeled benchmarking dataset along with proposed evaluation measures to quantify the performance of a summarized video in terms of compression rate (CR), activity recall (AR) and activity density (AD). Our experimental results show that the spatio-temporal trajectory building method described in this chapter was able to retain upwards of 97% of the activity in the original video. Being able to separate the foreground from background whilst retaining the majority of activity from input videos is an essential building block for our framework as described in Chapter 2.

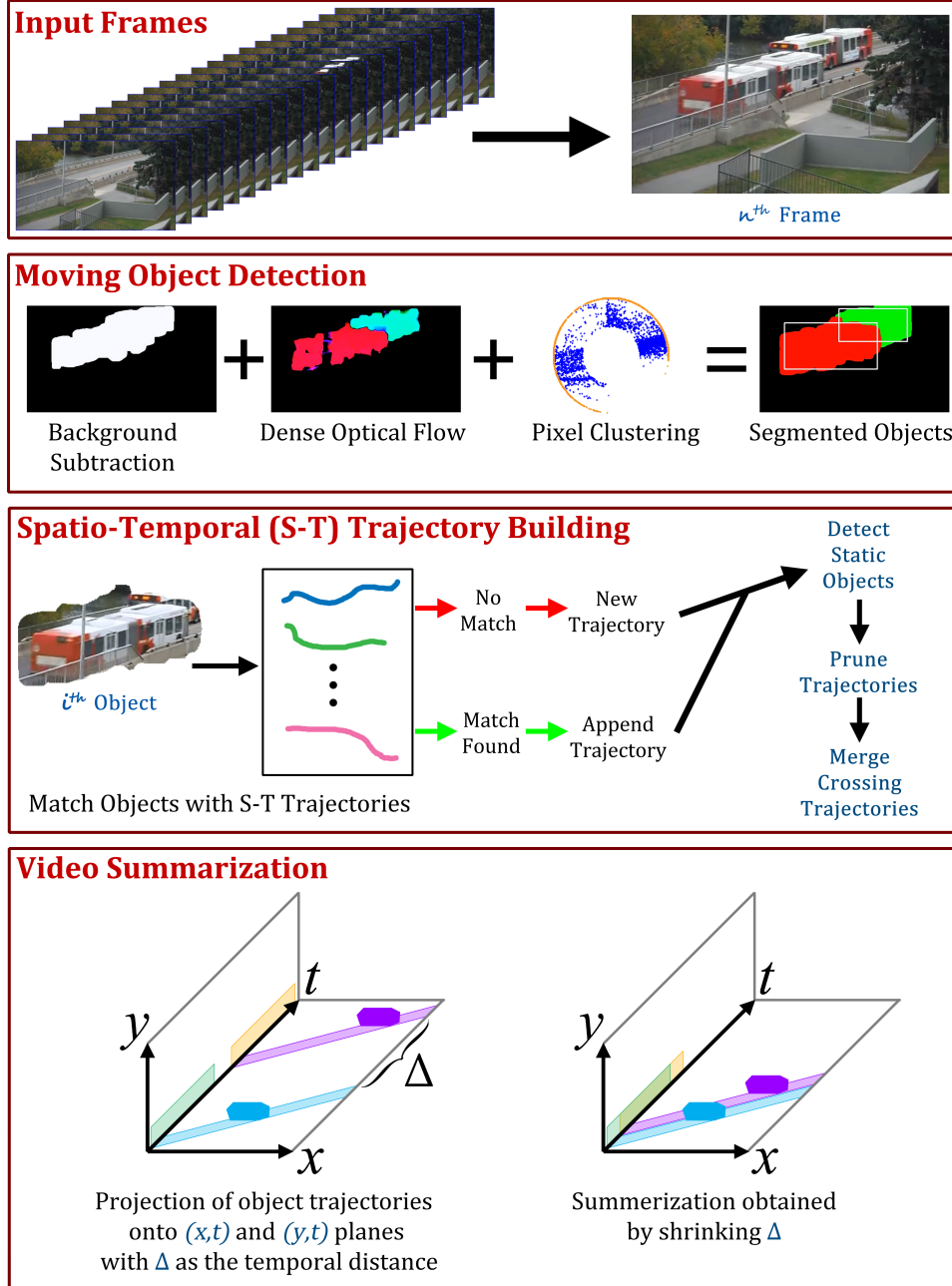


Figure 3.1: Illustration of each major step required in our approach for video summarization of surveillance cameras. From the top to the bottom, we are first given a set of input video frames (first row). Next all of the moving objects, per frame, are detected through a combination of background subtraction, dense optical flow and pixel cluster (second row). With these objects detected, the spatio-temporal (s-t) trajectories are built incrementally by matching the objects with the set of active trajectories for the current frame (third row). Finally once we have all of the s-t trajectories, they can be re-arranged by examining their projections onto a video cube (last row).

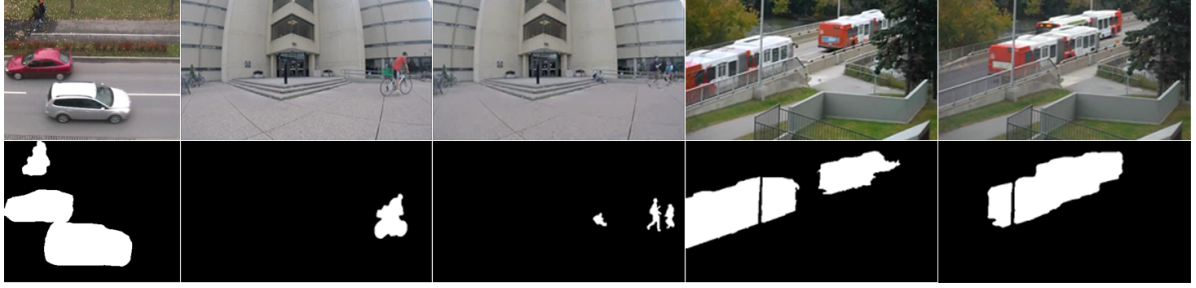


Figure 3.2: Thumbnails of our dataset. Each column is a separate sequence. The top row is a single frame from a sequence and the bottom is a single frame from the corresponding manual binary mask.

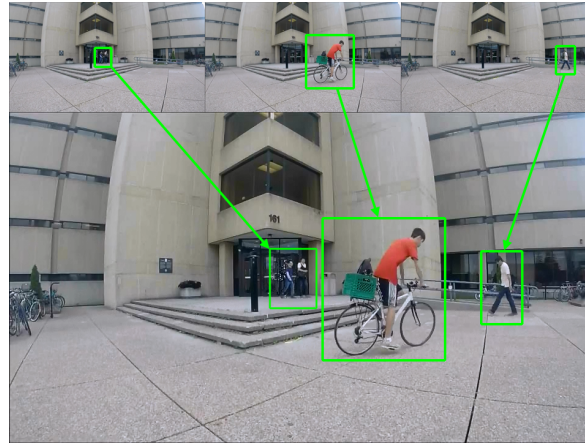


Figure 3.3: A single frame in the video summary (bottom row) and original frames for each object in the summary (top row) for the Building sequence.

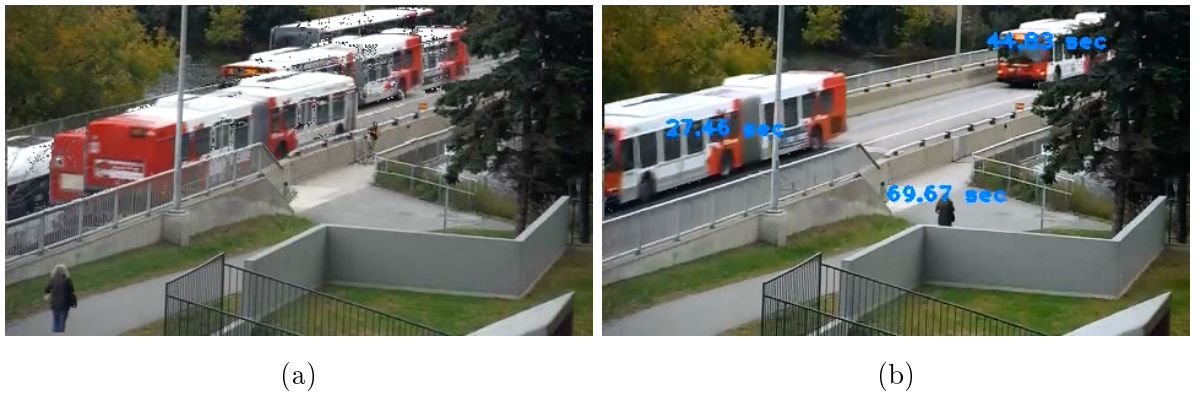


Figure 3.4: A single frame from two different video summaries for the Transitway2 sequence. (a) Overlapping trajectories and no timestamps. (b) No overlapping trajectories and timestamps.

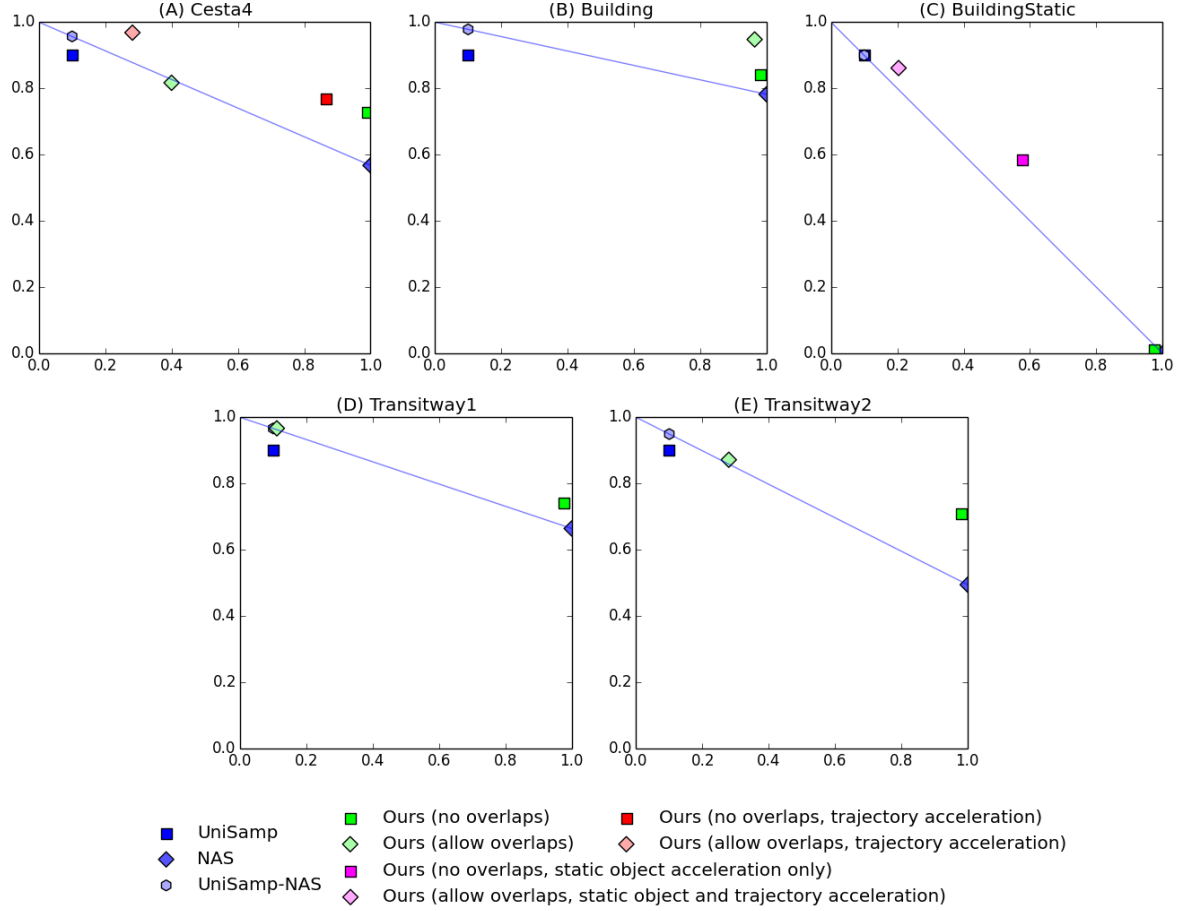


Figure 3.5: An $AR-CR$ plot for each sequence. The horizontal axis represents the activity recall (AR) and the vertical axis represents the compression rate (CR). A blue line is drawn between the NAS and UniSamp-NAS methods which represents the different summaries, and where they would fall in the $AR-CR$ plot, that could be produced by varying the N parameter for UniSamp-NAS.

Chapter 4

Visual SLAM with RGB-D Sensors

Accurate and drift-free pose estimates of a moving camera is integral in robotics applications which require navigation and mapping. In relation to this thesis, we explore the topic of dense visual simultaneous localization and mapping (SLAM) with consumer grade RGB-D sensors, such as Microsoft's Kinect ¹, in the context of 3D change detection. In Section 4.1 a general overview of the visual SLAM problem is given. Section 4.2 covers visual SLAM algorithms which make use of a RGB-D sensor. An application using dense visual SLAM for detecting changes between two RGB-D videos taken in the same environment is covered in Section 4.3. The methods developed for the application are beneficial to our thesis goals of producing immersive VR scenes as in addition to being able to perform 3D reconstruction of static scenes, we can detect changes (i.e: objects of interest) between two RGB-D videos viewing the same scene.

¹<https://developer.microsoft.com/windows/kinect>

4.1 Overview of Visual SLAM

Typically, SLAM for moving robots is solved using a variety of sensors (lasers, IMU, GPS, stereo cameras, etc...) [156, 30, 4, 88]. While the multi-sensor approaches have been successful in applications such as automated driving [88], there exist many situations and environments where some or most of the sensors are not reliable or even not available. Visual SLAM refers to a variant of the SLAM problem where the only sensors available are cameras.

Let $\mathcal{I} = \{i_1, i_2, \dots, i_n\}$ be a set of input images. A visual SLAM algorithm takes in $\mathcal{I}$ and produces two sets:

1. The camera poses $\mathcal{C} = \{c_1, c_2, \dots, c_k\}$ where $c_i \in SE_3$. These camera poses represent the trajectory the robot has taken within its environment.
2. The map, $\mathcal{M} = \{p_1, p_2, \dots, p_x\}$, of the environment where each $p \in \mathcal{M}$ represents a "landmark" position. Landmark positions are those which are used to help determine the camera poses and reconstruct the environment.

Most visual SLAM algorithms assume the sensors observe a static environment. Maintenance and optimization of the computed camera poses is typically achieved through a *pose graph* [99]. The nodes of a pose graph are the camera poses while an edge connects two poses if there exists some spatial constraint between them. A constraint consists of a probability distribution over the relative transformations between two poses. These transformations can come from odometry measurement data or from image alignment data. Some exceptions exist, such as ElasticFusion [166] which maintains the camera poses without a pose graph and DynamicFusion [111] which can handle a limited dynamic environment. However, for the vast majority of methods, the following building blocks outline the general pipeline used in visual SLAM algorithms:

Visual Odometry Incremental estimation of the camera trajectory and the 6-DoF cameras poses using the input images.

Map Building Reconstruction of the 3D environment using the estimated camera poses and 3D points found from the input list of images.

Loop-closures Detection for when the camera re-visits portions of the scene that have already been mapped. A positive detection results in a re-arrangement of the

current camera poses by essentially connecting two nodes in the pose graph and then optimizing the new arrangement of poses and spatial constraints.

Bundle Adjustment Optimize the camera poses by minimizing the 2D image re-projection errors of estimated 3D points from the scene.

Precisely how each of the building blocks are implemented has given rise to a multitude of visual SLAM algorithms. Many of these variants can be categorized as follows:

Sparse or Dense Sparse visual SLAM methods use only a small portion of the image (i.e: keypoint features such as SIFT [98] or BRISK [87]) for visual odometry and map building. Dense visual SLAM methods use the entire image.

Mono or RGB-D Mono-SLAM algorithms only use RGB or grayscale images while RGB-D SLAM algorithms can include images captured with a passive or active stereo cameras such as the Kinect.

Fusion-based or Point-based A fusion-based SLAM algorithm reconstructs the environment by incrementally accumulating a truncated signed distance function (TSDF) [18]. A 3D mesh model can be extracted from the TSDF induced by the input images. A point-based SLAM algorithm outputs a point cloud as the 3D reconstruction of the scene.

Given the large number of algorithms in the topic of visual SLAM, we will only concentrate on the relevant algorithms which make use of RGB-D sensors.

4.2 RGB-D SLAM

In this section we cover how various RGB-D SLAM algorithms satisfied each of the building blocks of a visual SLAM algorithm as defined in Section 4.1.

Visual Odometry Henry et al. [57] and Endres et al. [32] both combine visual features such as SIFT [98] or BRISK [87] with the associated depth values to first estimate an initial rigid transformation through RANSAC. This initial transformation is then refined through a joint optimization over the sparse feature matches and dense point cloud alignment through the Iterative Closest Point (ICP) algorithm [176] (or any of its variants [128]). The refined transformation is applied to the previous camera pose and the pose graph is updated with the new camera pose.

In KinectFusion [112] only the depth data is utilized for visual odometry. The camera is tracked using a version of ICP which computes correspondences through projective data association [128]. Real-time performance is achieved through a GPU-based implementation.

Map Building All methods can produce a point cloud reconstruction by accumulating all of the point clouds produced from each camera pose transformed RGB-D frame. Surface reconstruction algorithms (such as the Poisson surface reconstruction [66]) can be applied to the resulting point cloud to obtain a mesh based representation. KinectFusion-based [112] algorithms accumulate a truncated signed distance function (TSDF) [18] and a mesh surface can be extracted through a marching cubes algorithm [96].

Loop-closures Most approaches for loop-closures depend on fast image matching techniques. Henry et al. [57] and Endres et al. [32] both define keyframes based on the number of RANSAC inliers from their frame-to-frame alignment approach for visual odometry. A low number of inliers implies the current view contains too few 3D feature point matches with the previous keyframe and thus a new keyframe is required. This new keyframe is still localized relative to the previous keyframe through the frames in between. Each time a new keyframe is created, it is matched against a portion of the previous keyframes to detect if the camera is revisiting a previous location. A vocabulary tree [114] can be used to efficiently select n of the previous keyframes with similar appearance. These n keyframes are then examined in more detail to determine the best match geometrically. The vocabulary tree approach by Nister et al. [114] was inspired by the work of Sivic and Zisserman [143] which accumulated into the open source software OpenFABMAP [16, 47]. Lastly, since Whelan et al. [166] does not use a pose graph, loop closure is achieved through surface matching and deformation.

Bundle Adjustment Once the complete set of loop-closed camera poses has been estimated per key frame, a large bundle adjustment [97] over all camera poses can further optimize the entire pose graph. This step is entirely optional and is typically not used in real-time applications as it is computationally expensive.

4.3 Application: Change Detection in RGB-D Videos Captured with Patrolling Robots

The detection of changes within a scene is a fundamental pre-processing step in many higher level computer vision and video surveillance applications. Typically these applications deploy change detection to identify regions within an image or a video which are of interest for subsequent processing. In this section we examine and propose a solution to a variant of the change detection problem for patrolling robots. In this variant, the robot records a video while patrolling its route and the goal is to automatically detect changes, or "abnormalities", from any subsequent traversals of the route. Thus the input to the problem of change detection for patrolling robots is a reference video, which we consider to contain no abnormalities, and a set of probe videos for which changes or abnormalities might exist. The output would be a video highlighting any changes detected with respect to the reference video. More specifically, changes such as objects which were present in the reference video but missing in the probe video (or vice versa) should be highlighted while the frames without changes should be skipped. Such a video would be of value to human operators who wish to be able to quickly see all the abnormalities found during the robots patrol.

Performing change detection for patrolling robots presents three main challenges:

1. While the patrolling robot follows the same route, the exact path and the time it takes may vary.
2. Since the robot does not follow the exact same path, it is necessary to understand the route in terms of geometry and appearance to accurately detect changes.
3. As we wish to detect an "abnormality" in general, specific object detectors will not suffice.

With these challenges in mind, we need to solve several problems in order to detect changes for patrolling robots. Firstly, the reference and probe videos must be aligned spatial-temporally in order to perform any comparisons. Secondly, once the differences between the two videos have been detected, there needs to be some understanding of what the differences correspond to since the same object or event should be given the same labelling throughout the video. Finally, in the output video we would like to give focus to the frames where the differences exist.

In this section, a summary of the proposed framework solves all of above by examining the 3D maps, represented as point clouds, of the scene containing the robot routes produced from the reference and probe videos. The difference between these maps would then highlight all the abnormalities. Clustering these abnormalities in 3D space provides a consistent labelling. The camera poses estimated while building the 3D map allows for a re-projection from 3D back to 2D so that an easy to understand video highlighting the differences can be obtained. The rest of this section is organized as follows. Section 4.3.1 provides a survey of related works and Section 4.3.2 summarizes our proposed approach for detecting changes and producing a video summary of those changes.

4.3.1 Related Works

Andreasson et al. [1] and Finman et al. [39] both tackle the problem of change detection using patrolling robots in a similar manner - they move the problem into the domain of 3D space. Andreasson et al. [1] deployed a laser scanner combined with a CCD color camera to first build their reference model, represented as a point cloud. The probe model is generated similarly. In order to detect changes, Andreasson et al. [1] combined color image matching with the associated interpolated [2] laser scan data to more robustly estimate the camera pose of a scan from the probe model with respect to the reference model. Once the colored depth map has been registered and aligned to the reference model, changes are detected through a 3D Normal Distributive Transform (3D-NDT) [116, 102] point cloud difference.

More recently, Finman et al. [39] used a commodity RGB-D sensor and a RGB-D SLAM system (specifically, Kintinuous [165, 167]) to generate a map, represented as a point cloud, of the patrolling robots route. With a reference map defined, changes can be detected from probe maps using a 3D point cloud difference where the only points kept are those which are within some radius r of another point in the reference map. It should be noted Finman et al. [39] aligned their reference and probe maps manually as their main goal was not a complete framework for change detection but to use change detection in facilitating the learning of multiple object segmentation algorithms for the reference map. Additionally while their object segmentation learning approach does take into account color differences, their change detection method does not.

The major disadvantage of the work by Andreasson et al. [1] is the usage of a specific combination of hardware which results in data being captured only at discrete locations along the patrolling robots route rather than continuously. Thus motion which occurs

in between laser scans will not be captured resulting in a loss of information. RGB-D SLAM systems, such as the one used by Finman et al. [39], utilizes continuous data which provides some robustness to moving objects. However, despite using a RGB-D SLAM system, Finman et al. [39] aligned captured maps manually, their change detection method did not take into account color changes and they did not make full use of the captured RGB-D video. Thus, similar to the method by Andreasson et al. [1], any detected changes would only be viewable as static 3D objects leading to a potential loss of contextual information. For example, if the patrolling robot was able to capture a door which opened and closed, this would be visualized as a static object despite the fact that in the RGB-D video you can clearly see the motion of the door.

4.3.2 Proposed Approach

Our approach addresses the problems and draws from the strengths found from both methods. We use a RGB-D SLAM system, similar to Finman et al. [39], to build our maps but also perform automatic map alignment and account for color differences like in Andreasson et al. [1]. Furthermore, the output of our change detection not only includes the 3D objects but also a video with indicators per frame highlighting the detected differences. The indicators track each of the 3D objects throughout the video. As far as we know, there have been no approaches for change detection using video captured from patrolling robots which can do all of the following: (1) automatically align two maps, (2) detect changes in both 3D space and color space and (3) produce an easy to understand RGB video summary highlighting all of the detected changes.

The proposed framework leverage's the additional information which can be obtained from 3D maps to bypass the problems of temporal and spatial de-synchronization that can occur when several videos are taken by a patrolling robot at different times. Once a 3D map has been built for the reference and probe videos, any changes can be detected through the difference between the two maps. After filtering these differences for noise, individual objects are segmented through spatial and appearance based clustering. We use the segmented result in conjunction with the computed camera poses from the 3D map to perform a "3D walkthrough" of the partolling robots route. Whenever an abnormality is within a camera's field of view, its 3D coordinates are projected back onto the original RGB video frame. The frames where there are no changes in view can be safely skipped. Fig. 4.1 illustrates the pipeline for our proposed framework.

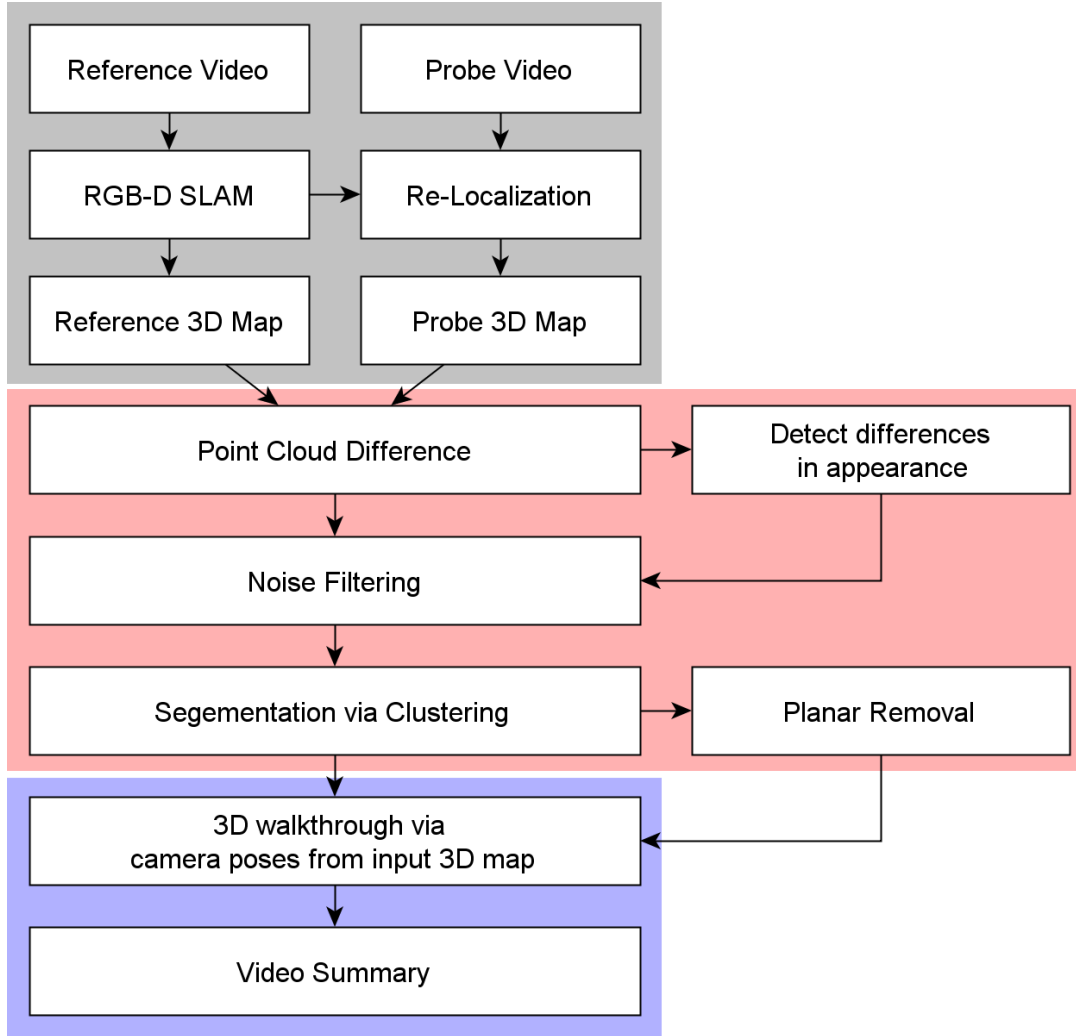


Figure 4.1: Pipeline of our proposed approach. Steps for building and aligning 3D maps, change detection and object segmentation, and producing a video summary are contained in the grey, red and blue boxes, respectively.

Building 3D Maps

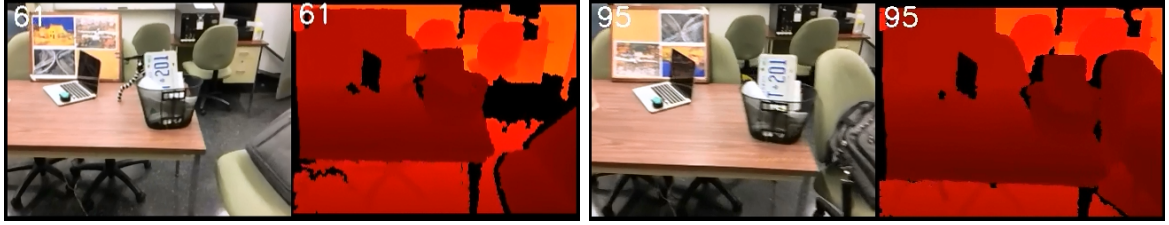
The RGB-D SLAM system from Labbé et al. [77, 76, 78] was used in conjunction with a commodity structured light sensor to build 3D maps from RGB-D videos. The output of the RGB-D SLAM system is a point cloud representing the scene that the robot traversed through as well as a list of camera pose estimates for the trajectory taken. Typically, in real-time SLAM systems, only key frames are used for map building and camera pose estimation. Camera poses for regular frames can be found through interpolation of neighbouring key frames. Once we have a 3D map of the reference video, the 3D map of the probe video can be built similarly with the additional aide of re-localization. By leveraging the information we already have from reference model we can automatically align two 3D maps to a common coordinate system to perform a point cloud difference.

Point Cloud Difference

The difference between two point clouds was found using the Hausdorff distance [3] which computes a closest point $a_i \in A$ for every point $b_j \in B$, where A and B are point clouds. For every $b_j \in B$, if $d(a_i, b_j) > \alpha$ or $c(a_i, b_j) > \beta$ then b_j is marked as a "difference" where $d(x, y)$ is the Euclidean distance and $c(x, y)$ is the color distance in CIELab color space [159]. We choose α to be a percentage of the largest side length of the 3D bounding box containing $A \cup B$. This allows us to detect geometric differences whose size is relative to region covered by $A \cup B$. We choose $\beta = 20$ so that the color differences are perceivable by the human eye and to account for the fact that RGB-D SLAM systems integrate images from many different viewpoints. Integration of multiple images means that illumination can have an effect on the colors of A and B . Fig. 4.3 provides a visualization of adding appearance difference detection to the images displayed in Fig. 4.2.

Change Detection

Using the result of a point cloud difference, individual objects are detected through spatial clustering. More specifically, our framework makes use of the method introduced by Ester et al. [35], called density-based spatial clustering of applications with noise (DBSCAN), as it does not require prior knowledge of the number of clusters beforehand and is robust to noise. DBSCAN requires two parameters: ε , which controls the search neighbourhood for each point, and $minPts$, the minimum number of points which forms a "dense region". The algorithm searches in the ε -neighbourhood of each point to find dense regions to merge. For more details on the algorithm see the original paper [35].



(a) Frame 61 in the "Reference" video from "Cluttered" dataset. (b) Frame 95 in the "Change01" video from "Cluttered" dataset.

Figure 4.2: Both 4.2a and 4.2b show a planar object with several pictures on it. The differences are apparent in the color images (left) while the depth maps (right) are unable to pick up this difference.

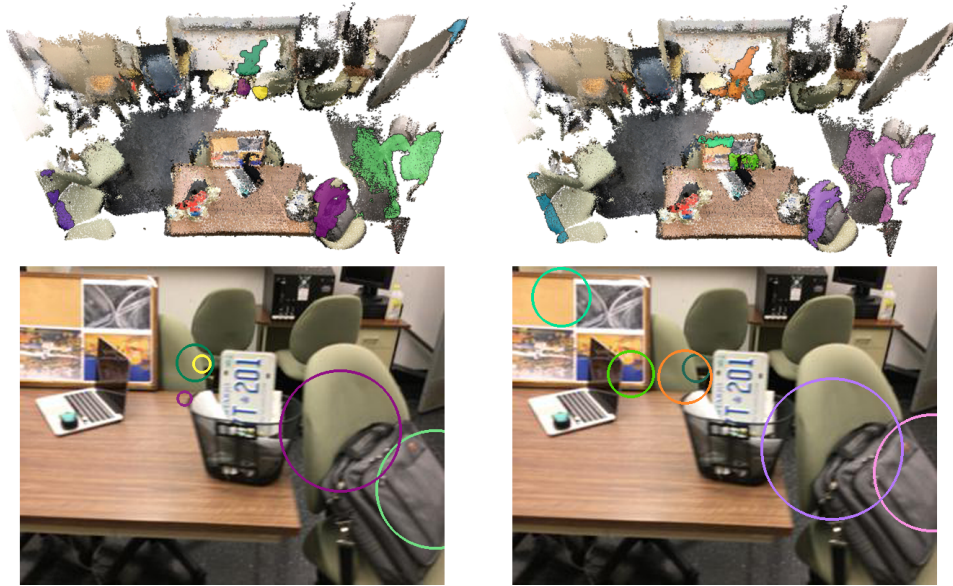


Figure 4.3: The results of change detection without appearance constraints (left) and with appearance constraints (right) overlaid onto of the probe 3D map (top row) and video frame (bottom row). The appearance constraints allows us to detect the plane with pictures on it from Fig. 4.2a.

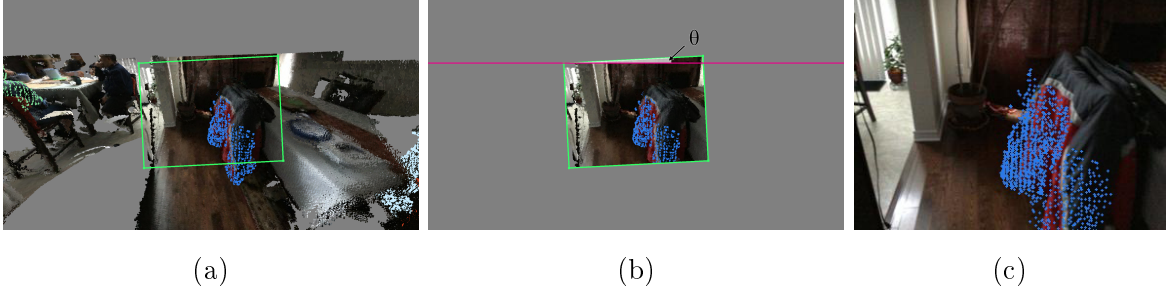


Figure 4.4: A example of projecting the view from a camera onto the original video frame so that the detected differences are overlaid. The detected differences were rendered as colored discs and the camera bounds as the green rectangle. (a) Projected points onto pixel buffer. (b) Crop to camera bounds and compute θ to align with the axis. (b) Rescale to original size and draw the differences.

Projecting Detected Changes to Video

Let $\mathcal{P}$ be the 3D map, represented as a point cloud, built from the video frames $\mathcal{F} = \{f_1, f_2, \dots, f_k\}$ and $\mathcal{C} = \{c_1, c_2, \dots, c_k\}$ be the camera poses obtained from the 3D map building process. The detected changes, $\tilde{\mathcal{P}} = \{p_1, p_2, \dots, p_n\}$, are then a set of point clouds such that each $p_j \in \tilde{\mathcal{P}}$ ideally corresponds to a 3D object of interest and $p_j \subseteq \mathcal{P}$.

To project $\tilde{\mathcal{P}}$ onto some frame $f_i \in \mathcal{F}$, we first render all the points of every point cloud $p_j \in \tilde{\mathcal{P}}$ which are visible from $c_i \in \mathcal{C}$ onto a pixel buffer B of dimensions $w \times h$ (Fig. 4.4a). Using the transformation matrix of c_i , and the video frame dimensions $x \times y$, we can determine the coordinates of the virtual camera bounds, v_i , projected onto B . We crop B according to v_i to isolate all of the points $p \in p_j \forall p_j \in \tilde{\mathcal{P}}$ which belong to f_i . The isolated points are rotated so that they are axis aligned (Fig. 4.4b). In order to render them back onto f_i we re-scale the cropped image back to the dimensions of f_i (Fig. 4.4c). To avoid cluttering we can: (1) render a bounding circle instead of the point clouds and (2) only project a point cloud $p_j \in \tilde{\mathcal{P}}$ onto f_i when p_j is large enough.

Video Summarization of Detected Changes

Since the detected changes are 3D objects, they can be difficult to interpret without the context of the original video. We introduce a video summary which takes the detected 3D objects and overlays it onto the input video. Such a video can quickly provide context to the detected changes and can be viewed on both desktop and mobile devices.

Our video summarization algorithm takes as input: (1) a set of camera poses, $\mathcal{C} =$

$\{c_1, c_2, \dots, c_k\}$, (2) a set of point clouds, $\tilde{\mathcal{P}} = \{p_1, p_2, \dots, p_n\}$, representing the detected changes and (3) the video $\mathcal{F} = \{f_1, f_2, \dots, f_k\}$ to be summarized. In essence, our algorithm performs a "3D walkthrough" using $\mathcal{C}$ and determines which points from every point cloud $p_j \in \tilde{\mathcal{P}}$ are visible from a camera $c_i \in \mathcal{C}$. If there are not enough points visible from c_i then the associated video frame, $f_i \in \mathcal{F}$, is marked for fast-forwarding. In addition to determining which $f_i \in \mathcal{F}$ are to be fast-forwarded we also determine which locations in f_i are of interest so that an indicator can be rendered onto f_i for improved comprehension. To do this for some frame $f_i \in \mathcal{F}$, we follow the steps outlined in Section 4.3.2

Finally, since the camera may linger on any detected change for an extended period of time, we can apply addition rules to fast-forward those frames. The goal is to simultaneously reduce the length of the final result while not skipping frames which contain "interesting" portions. As the latter is inherently subjective, we define "interesting" to be when new changes come into view or when all currently visible changes have been visible for a threshold amount of time. The steps we take for summarization with visible changes are as follows:

1. Record the amount of time that each change has been visible.
2. For any frame f_i , if all the projected changes have been visible for k frames, fast-forward the next T frames until a new change moves into view.
3. Reset the timer for each summarized change once the summarization stops.

The effect of applying these additional rules is that we fast-forward until either new changes or a new view of the current changes is found.

4.4 Summary

In this chapter a brief survey on visual SLAM was provided with a focus on the RGB-D SLAM methods. A practical application, change detection for patrolling robots, which utilized RGB-D SLAM as a building block was also presented. The methods for reconstruction of static 3D models using RGB-D SLAM systems and the detection of changes between two RGB-D videos can be used to satisfy the static 3D background reconstruction and separation of the foreground and background components for the system defined in Chapter 2.

Chapter 5

Single Camera Dynamic VR Scene Creation

Creating experiences for VR typically falls along a spectrum with one end containing computer generated imagery (CGI), such as 3D video games, and the other end containing reality based, such as stereo video. CGI often requires a team of specialists to create a virtual 3D world and provide rules for interactions, while reality based content utilizes a combination of visual sensors and algorithms to display or reconstruct a captured scene. The approach in this chapter is concerned with the latter, specifically the case of a single RGB-D sensor.

One motivation for focusing on the case of a single RGB-D sensor is to explore the possibility of creating VR content through a hand-held device. Realistically, an average consumer may not have the space and/or the resources to setup a multi-sensor

environment. Furthermore such an environment has poor mobility making it difficult to record content spontaneously at different locations. The aim of this chapter is to provide a framework to empower regular consumers to record their own content to be viewed or interacted with in VR at a later date. Being able to reproduce a dynamic scene through a single sensor is therefore of great interest.

The remainder of this chapter is organized as follows. Section 5.1 surveys related works and Section 5.2 details a contribution of this thesis: an approach for creating VR scenes with dynamic objects using a single RGB-D sensor. Section 5.3 provides a discussion of our approach based on our experimental results. Finally, Section 5.4 summarizes.

5.1 Related Works in VR Scene Creation

A large quantity of methods exist for producing reality based VR content through visual sensors. In this section a brief overview of the most dominant categories is provided before highlighting the works most relevant to this thesis. When discussing each category, the focus will be on the amount of motion accommodated for a viewer in terms of degrees of freedom (rotational and positional). The more degrees of freedom accommodated, the more immersive the VR experience will be.

Perhaps the oldest, and most studied, approach for producing VR content is stereo video. The concept is simple: a stereo camera is composed of two cameras which simulates human binocular vision. When the captured video is played back, each camera displays an image for one eye thus allowing a user to feel as if they were at the camera's position. The major drawback is the fact that the viewer's head movement, rotational and positional, will not be reflected in the content they are viewing. Multiple monocular or stereo cameras can be combined into one capture rig to produce 360° video [173] or 360° stereo video [158] allowing for rotational head movement. Regardless of the methodology used to produce a 360° video (stereo or otherwise), it still cannot provide the viewer's head with positional freedom.

A volumetric or RGB-D video provides viewers with rotational and positional freedom since each frame is a 3D object. Typically, a RGB-D video is captured through active stereo sensors such as Microsoft's Kinect [177]. Passive stereo systems can also produce a RGB-D video but require additional processing to obtain a depth map [157]. Both active and passive stereo systems have their weaknesses, which are often complementary, and

some have tried to combine both to overcome the weaknesses of each [174]. If a RGB-D video does not contain any dynamic objects, then static scene reconstruction algorithms, such as KinectFusion [112], can be used to accumulate data from multiple viewpoints resulting in a complete 3D model. More recently, DynamicFusion [111] demonstrated dynamic object reconstruction by non-rigidly warping a reference volume to each new RGB-D frame and fusing new depth samples into the model. It should be noted that this work was aimed at reconstructing objects rather than a complete scene and thus the background data is ignored.

Methods using multiple RGB-D sensors have been explored. The approach by Dou et al. [26] is composed of two stages: off-line scanning and on-line running. During the off-line stage a single Kinect sensor scans the target room beforehand to obtain a high quality background model. The on-line stage utilizes 10 Kinect sensors positioned around the target room to record dynamic content. Alignment of the background model and live data is achieved with through SIFT features and planes [27]. More recently, Dou et al. [28] introduced Fusion4D, a pipeline for real-time performance capture that is robust to large frame-to-frame motion and topology. They accomplish this through the use of volumetric key-frames allowing for a more recent model to incorporate new data during quick movement or topological changes. However, similar to DynamicFusion [111], their goal was to capture specific objects within the view of their multiple sensors and thus the background data is not considered.

The works most related to the approach presented in this chapter are due to Dou et al. [26] and Newcombe et al. [111]. While compelling, multi-view approaches require complex camera setups and thus are not practical in a mobile environment. In contrast, single sensor approaches for scene reconstruction have either focused on building dynamic objects through precise tracking of the target while ignoring the background [111], or focused on producing static models without dynamic objects [112]. As discussed in Section 2.1, in order to have an immersive VR scene, both the dynamic objects and their corresponding background must be present simultaneously.

5.2 Dynamic VR Scene Creation with a Single Camera

An initial approach based on the knowledge gained from Chapters 3 and 4 was devised with the aim to address the properties as defined in Section 2.1. This approach assumed a static camera like in Chapter 3 and was localized to a background model produced

through RGB-D SLAM as in Chapter 4. Foreground and background separation was applied as in Chapter 3 and the result refined with the 3D point cloud change detection measures as in Chapter 4. In essence, this approach was a restricted version of the database solution described in Section 2.3 where the camera is static.

While the above approach can produce VR scenes which technically address the first three properties as described in Section 2.1, the scenes do not feel very dynamic due to the fact that the interesting foreground objects must enter and leave the field of view of a fixed camera pose. A freely moving camera would allow for more interesting and dynamic scenes as the actors may move around the environment rather than be restricted to a sub-portion. To allow for a freely moving camera and to improve the dynamism of the scene, an approach based on the generalized solution of Section 2.4 was explored. This approach is one of the contributions of this thesis which led to a publication [79]. A technical overview of the system is described in Section 5.2.1 and the experimental results illustrating the effects of different components is detailed in Section 5.2.2.

5.2.1 System Overview

Given a set of RGB-D frames, $\mathcal{F}_{input}$, the foreground, $\mathcal{F}_{fg}$, is separated from the background, $\mathcal{F}_{bg}$, using a combination of human pose recognition [138] and visual object tracking [145]. Once the foreground and background have been separated, we feed the background only frames, $\mathcal{F}_{bg}$, into a RGB-D SLAM system to obtain a static background model, $\mathcal{M}_{static}$, and a set of camera poses $\mathcal{C}$ such that there is a one-to-one correspondence for each $c_i \in \mathcal{C}$ to each $f_{bg}^i \in \mathcal{F}_{bg}$ and $f_{fg}^i \in \mathcal{F}_{fg}$ where i is the frame number. Since $f_{bg}^i + f_{fg}^i = f^i$ where $f^i \in \mathcal{F}_{input}$, then the foreground objects $\mathcal{F}_{fg}$ can be superimposed onto $\mathcal{M}_{static}$ via $\mathcal{C}$ allowing for a dynamic scene to be displayed in VR. Figs. 5.1 and 5.2 visualize the proposed pipeline and results. The next subsections provide the technical details for achieving the individual building blocks as discussed in Chapter 2 and shown in Fig. 5.1.

Separating the Foreground and Background

Since people are often the most interesting subject of recorded videos, we first focus on segmenting people from the input RGB-D frames before discussing the case of more general foreground objects. Section 5.2.2 provides a comparison of several methods for segmenting the foreground objects.

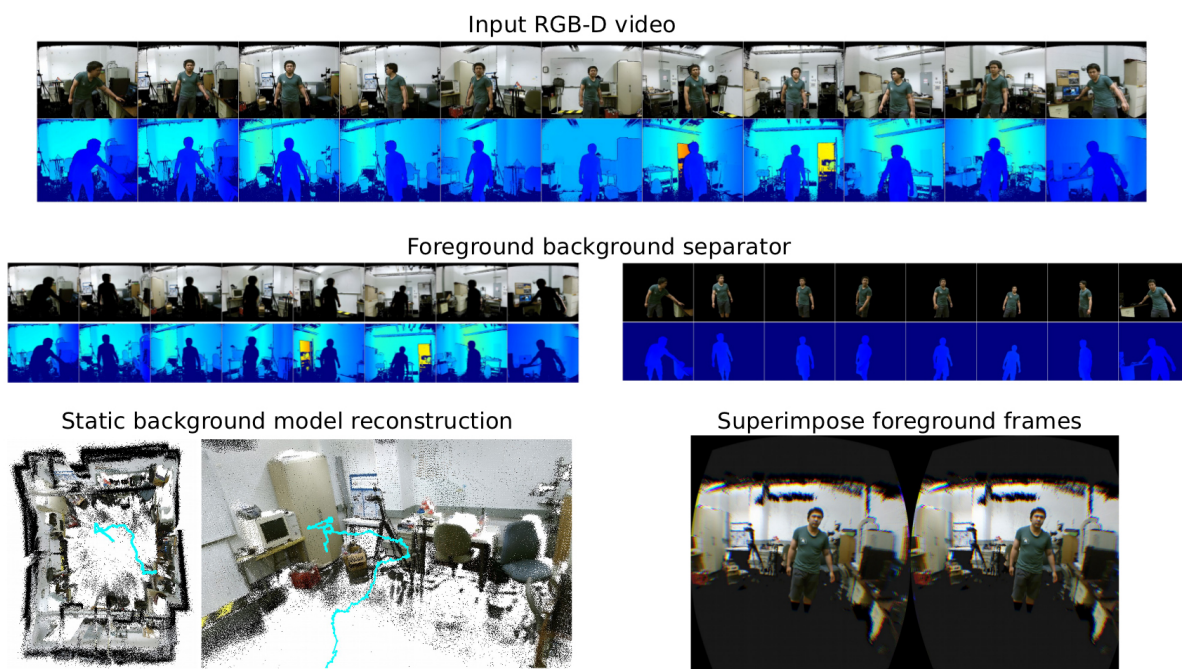


Figure 5.1: Visualization of the pipeline for our proposed approach to solve the first three properties as defined in Section 2.1.

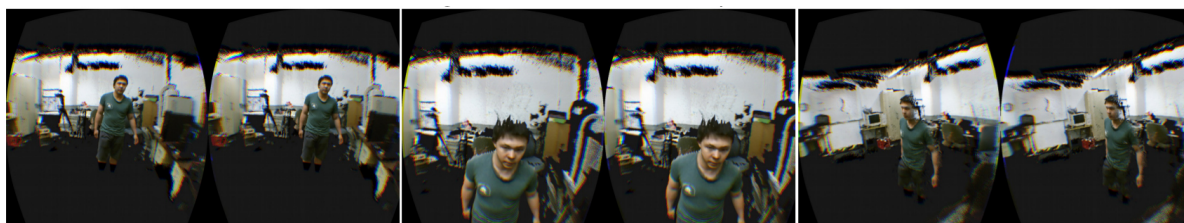


Figure 5.2: Example of the results produced from the input video in Fig. 5.1. A single frame is shown from multiple viewpoints coherently with the complete background model.

People Extraction

We utilize Microsoft’s Kinect SDK [138] to capture skeletal information and the foreground mask for each detected person. The Kinect SDK is not without its faults and does have skeleton tracking failures (see Fig. 5.3). These failures mostly occur when a person is not facing the sensor and/or when there is a limited/occluded view of the person. To overcome this problem, we deploy a general visual object tracker (VOT) to aid in tracking when the Kinect SDK produces a skeleton tracking failure. Specifically, we use the scalable kernel correlation filter (sKCF) [145] tracker as it is robust and capable of real-time performance. We combine the results of skeletal tracking and the sKCF by examining two events:

Skeletal Tracking Failure (STF) A potential STF is detected when the previous frame contains N skeletons but the current frame has $N - i$ skeletons, where $N \geq 1$ and $N \geq i \geq 1$. The sKCF tracker is initialized using the head positions of the previously detected skeletons and then run for the current frame. The classic flood fill algorithm [56] is applied only to the depth frame to extract a foreground mask. A large overlap between the previous and the current frames foreground masks implies that the person still exists and thus a STF has occurred. Once a STF is detected, the sKCF tracker is given tracking control until a second type of event occurs: tracked person merging.

Tracked Person Merging (TPM) A TPM is required when the previous frame contains N skeletons but the current frame has $N + i$ skeletons, where $N \geq 0$ and $i \geq 1$. In this case we have either rediscovered an existing person who had previously lost skeletal tracking or we have a new person entering the scene. A large overlap between the previous and current foreground masks implies we have rediscovered an existing person. Otherwise a new person has entered the scene and we begin tracking them separately.

General Object Extraction

Since we are interested in producing VR scenes from captured RGB-D video, online real-time processing is not strictly required. We present two approaches for general object extraction:

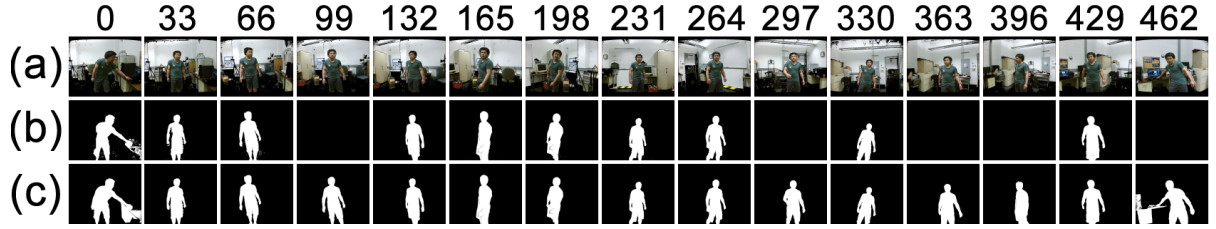


Figure 5.3: A sampling of frames from the KinectLoop sequence (see Table 5.1) with (a) color frames and the foreground binary masks from (b) just the Kinect SDK and (c) using Kinect SDK combined with sKCF and segmenting with flood fill.

Two-pass filter In the first pass, similar to Dou et al. [26], we scan the scene without any foreground objects to build a static background model $\mathcal{M}_{static}$. In the second pass, each RGB-D frame $f^i \in \mathcal{F}_{input}$ is localized against $\mathcal{M}_{static}$. Once localized, f^i is expanded into a 3D point cloud, P , through the camera intrinsics and the Hausdorff distance [3] is computed between P and $\mathcal{M}_{static}$. A threshold on the Hausdorff distance separates the points in P which are too close to $\mathcal{M}_{static}$ and the remaining points are considered the foreground.

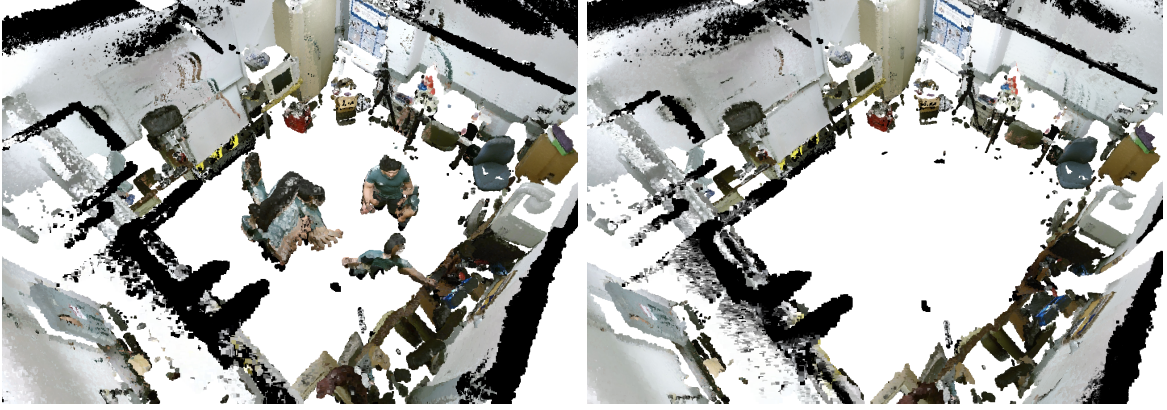
User selection In this approach we allow the user to select which objects are considered to be the foreground. To relax the amount of manual labor required, we use sKCF to track the user selected objects.

Building a Static Background Model

Once the foreground and background are separated, we feed the background only frames into the same RGB-D SLAM system used in Chapter 4 with loop closure detection [77, 76, 78]. The resulting model can, optionally, be further improved through pose graph refinement and sparse bundle adjustment [97]. To produce a mesh model we use the Poisson surface reconstruction algorithm [66] and perform a transfer of color attributes.

Superimposing Foreground Frames

Since there is a one-to-one correspondence for each foreground frame to a background frame, we can re-use the camera poses, $\mathcal{C}$, obtained from building a static background model $\mathcal{M}_{static}$. More specifically, we use $c_i \in \mathcal{C}$ to transform the foreground frames $f_{fg}^i \in \mathcal{F}_{input}$ into the same coordinate system as $\mathcal{M}_{static}$. Applying the camera intrinsics results in a properly superimposed 3D object that is consistent with $\mathcal{M}_{static}$. A triangular mesh of f_{fg}^i can easily be obtained by exploiting the grid structure of the image.



(a) Kinect SDK only

(b) Kinect SDK + sKCF

Figure 5.4: Static background models produced with background masks from the (a) Kinect SDK only (note the unsegmented foreground fused into the background) and (b) Kinect SDK + sKCF [145].

5.2.2 Dataset and Evaluation

To test, evaluate and validate our approach we introduce a dataset of 6 RGB-D image sequences along with a manually created binary mask for the foreground objects. The sequences were captured with two different sensors: a structured light and a Kinect V2. Note that our structured light sensor does not provide skeletal information.

Evaluation Metrics

Two main factors determine the quality of the final result: (1) the accuracy of the foreground background separator (FBSep) and (2) the static background reconstruction method. Since there already exists studies on RGB-D SLAM benchmarking and evaluation [33, 150], we focus our evaluation on our FBSep. More specifically, we examine the extracted foreground masks with respect to the manually labeled ground truth. Let GT_i and FG_i be the ground truth and extracted foreground masks at frame i , respectively. Listed below are our evaluation metrics and how they are measured per frame:

- **True Positive (TP):** $|GT_i \cap FG_i|/|GT_i|$
- **False Positive (FP):** $|(\overline{GT_i \cap FG_i}) \cap FG_i|/|FG_i|$
- **False Negative (FN):** $1 - TP$

Intuitively, TP is the percentage of ground truth foreground pixels extracted by the FBSep. The FP and FN are complementary in the sense that they both measure a more subtle visualization element of the final result. The FP measures the amount of background attached to the foreground frames during playback while the FN measures the amount of foreground which was fused into the static background model.

Evaluation Results

Using our proposed approach, we produced VR scenes for each of the sequences which were described earlier in this section. The results of applying our evaluation metrics is summarized in Table 5.1. Figs. 5.5 and 5.6 illustrate the TP and FP per frame (the FN is not shown as it is simply $1 - TP$) for each sequence in our dataset.

<i>Dataset name</i>	<i>FBSep Method</i>	<i>Avg. TP</i>	<i>Avg. FP</i>	<i>Avg. FN</i>	<i>Med. TP</i>	<i>Med. FP</i>	<i>Med. FN</i>
KinectTrans	Kinect Only	0.9547	0.2049	0.0453	0.9538	0.2134	0.0462
	Kinect + sKCF	0.9553	0.0215	0.0447	0.9552	0.0023	0.0448
KinectRot	Kinect Only	0.9352	0.0744	0.0648	0.9427	0.0207	0.0573
	Kinect + sKCF	0.9437	0.0060	0.0563	0.9444	0.0010	0.0556
KinectLoop	Kinect Only	0.7352	0.0070	0.2648	0.9640	0.0026	0.0360
	Kinect + sKCF	0.9683	0.0169	0.0317	0.9687	0.0001	0.0313
StructTrans	sKCF Only	0.9150	0.5123	0.0850	0.9138	0.5271	0.0862
StructPath01	sKCF Only	0.9037	0.5155	0.0963	0.9454	0.6179	0.0546
StructPath02	sKCF Only	0.9328	0.3962	0.0672	0.9345	0.4544	0.0655

Table 5.1: Evaluation results using the our metrics defined in Section 5.2.2 applied to two methods of foreground background separation (FBSep): (1) use only the Kinect SDK to track and segment dynamic objects and (2) our method where sKCF aids the Kinect SDK in tracking and object segmentation. The TP, FP and FN were computed per frame with the average and median values reported.

5.3 Discussion

From Table 5.1 we note that every image sequence from our dataset observed both a higher and lower median TP and FP (respectively) when using the sKCF tracker to aid the Kinect SDK. The most notable example is the KinectLoop sequence where the Kinect

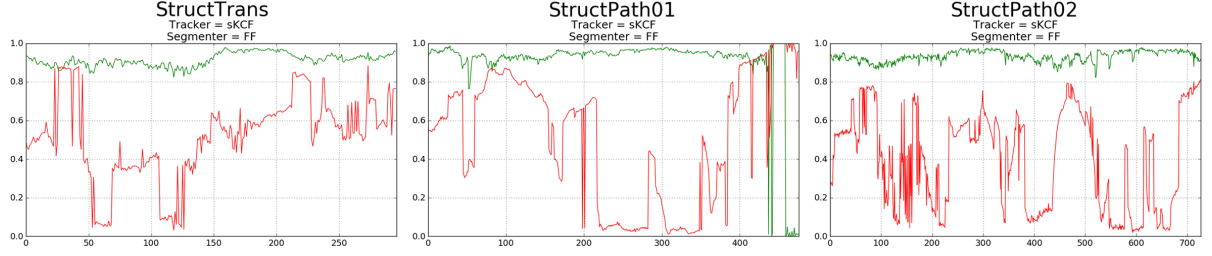


Figure 5.5: Per frame TP (green lines) and FP (red lines) results for the sequences captured using a structured light sensor. Vertical and horizontal axis denotes the TP or FP value and frame number, respectively. Columns correspond to a different image sequence from our dataset. Note, since the structured light sensor did not capture body poses, we manually initialized sKCF on the first frame.

SDK was unable to detect the targets skeleton for portions of the image sequence. These detection failures are visible as gaps in the bottom left graph from Fig. 5.6 and Fig. 5.4 shows the result of skeletal detection failures.

Our method achieved an average TP and FP of above 0.93 and below 0.03, respectively, for the image sequences captured using a Kinect V2. For the sequences captured with a structured light sensor we achieved an average TP and FP of above 0.90 and below 0.52, respectively. Manual inspection of the structured light sequences revealed that the high FP rate of the structured light sequences was mainly due two factors: (1) the target being at the limits of the sensor range thus resulting in a lower depth quality and (2) a number of static objects cluttering the scene which the classic flood fill algorithm included as part of the foreground. A second pass over the flood fill segmentation should be able to reduce the FP rate significantly and precisely how this can be accomplished is left as future work.

It should be noted that the method described in this chapter can achieve the first three properties of immersive VR scenes as described in Section 2.1 but not the fourth. In order to achieve the last property, we will need to fill in all the occlusions which can exist when using only a single sensor. This problem is known as *scene completion* and *shape completion*. Typically, for RGB-D data, scene completion refers to all objects within the FOV of a sensor and shape completion refers to specific objects. The topic of shape completion is thus the focus of the next chapter in this thesis.

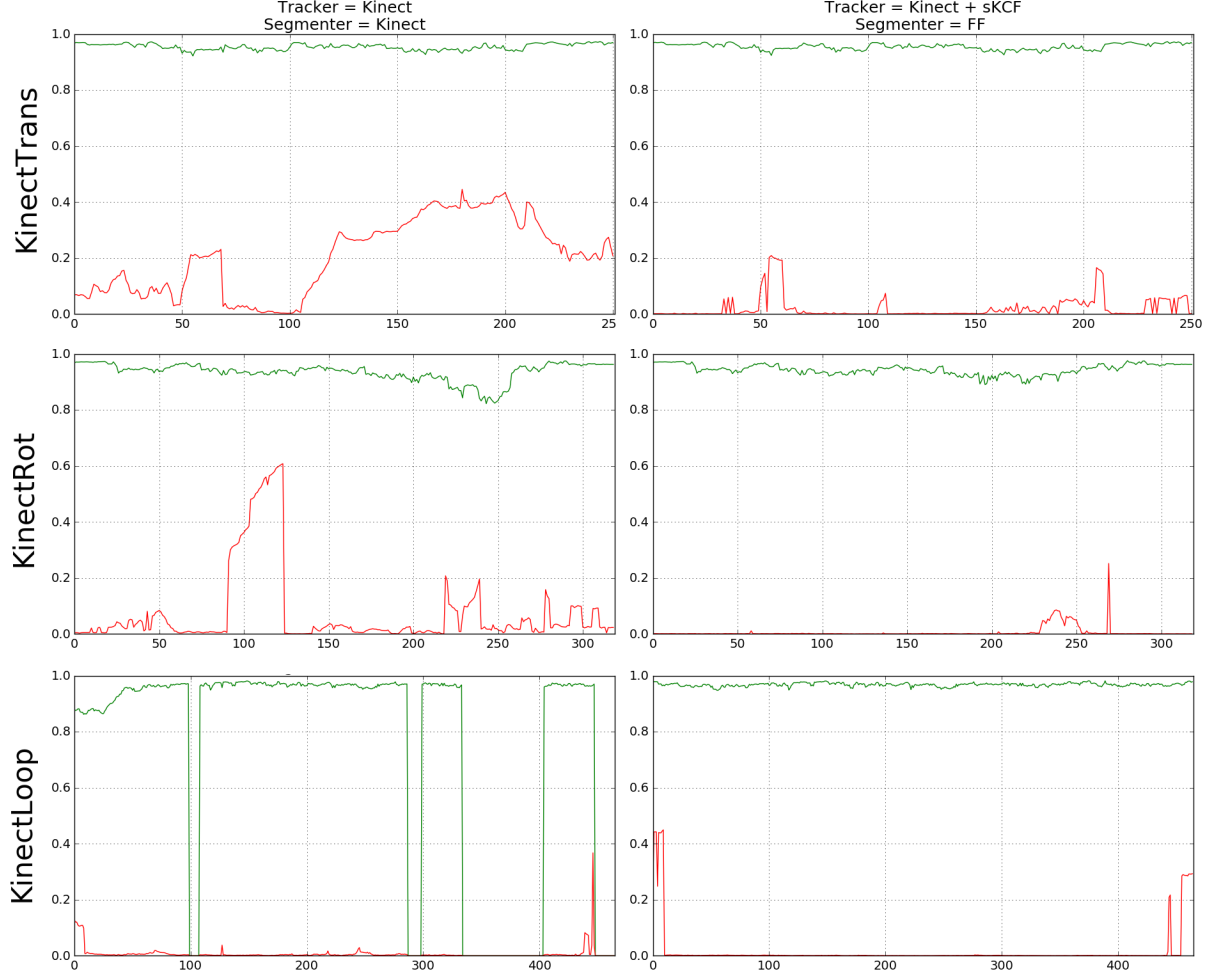


Figure 5.6: Per frame TP (green lines) and FP (red lines) results for the sequences captured using a Kinect V2. The left column contains results from using just the Kinect SDK to perform tracking and dynamic foreground object segmentation while the right column combines the sKCF tracker with the Kinect SDK to track and segment the dynamic foreground objects.

5.4 Summary

In this chapter we provided implementation details of our proposed system from Chapter 2 for producing VR scenes which include both a consistent background and dynamic objects using a single moving RGB-D sensor. Our implementation was tested over a proposed database of RGB-D image sequences and evaluation benchmarks specifically for the case of building VR scenes from single RGB-D sensors.

Chapter 6

Shape Completion of Single View Depth Maps

Advancements in sensor hardware has led to affordable commodity depth sensors [177, 130] facilitating the capture and modeling of large amounts of 3D data. By accurately tracking the sensor pose while maneuvering it around an object, multiple depth maps can be fused together and the 3D shape of the target object can be reconstructed [112]. In practice, due to time-space constraints, it may not be possible to maneuver the camera such that the entire surface of the object is captured. For example, a self-driving vehicle has both limited time and space to reconstruct the objects around it for obstacle avoidance systems. In relation to this thesis, 3D shape completion of objects from a single depth map would allow for the human actors from the dynamic VR scenes as produced in Chapter 5 to be as complete as possible. This 3D shape completeness is the fourth

property of immersive VR scenes which we identified in Chapter 2. Thus, completing the 3D shape of objects from a single depth map is of great interest.

The ability to complete and reconstruct the 3D geometry of an object from a single view depth map is particularly useful for VR/AR applications [136], robotics (i.e: grasping objects [160]), obstacle avoidance and content creation. However, reconstructing the 3D shape of objects from a single depth map is very challenging due to the inherent ill-posed nature of the problem - potentially infinite configurations and arrangement of shapes can produce the same depth map. Classical approaches for 3D reconstruction of partial structures include mesh hole filling [178] and Poisson surface reconstruction [67] but both of those approaches assume a strong global shape prior. That is, while the input is potentially sparse, it should not be missing entire portions of the object (i.e: a chair with missing legs).

Recent state-of-the-art approaches leverage the success of deep learning and learn 3D shape models directly from data [10, 136, 20, 151]. In these approaches a 3D skip-connected auto-encoder style network is typically used where the input and output is either a voxel occupancy grid or a signed distance field (SDF). These 3D networks essentially replace the 2D pixel array with its 3D analogue - a dense 3D voxel grid. Thus, computational and memory requirements for these networks scale *cubically*. As a result, these networks are typically trained with low resolutions (32^3 voxels) often leading to coarse outputs.

In this chapter we examine the single depth map to 3D shape completion problem from the perspective of just 2D convolutional neural networks (CNN). While it might seem counter intuitive to use a 2D CNN to predict 3D structures, other existing works have been successful in predicting depth maps from single color images using 2D CNNs [31, 81] to the level that it can be used in simultaneous localization and mapping (SLAM) applications [155]. Thus, 2D CNNs can still be expressive enough to infer 3D information with the benefit of having a lower computational and memory footprint. To accomplish our goal, we introduce the concept of an "additive depth map" for 3D shape completion and propose a 2D CNN to predict it.

The rest of this chapter is organized as follows. Section 6.1 provides the necessary background on CNNs - the basic component of our proposed 3D shape completion approach. Section 6.2 provides a survey of related works in 3D shape completion. Section 6.3 defines the additive depth map, how it can be used for shape completion and both it's advantages and disadvantages. Section 6.4 covers our CNN architecture that we will

use to predict the additive depth map. Section 6.5 goes over the datasets used to train our proposed network. Section 6.6 describes our experimental setup while Section 6.7 provides the results with synthetic and real data. We discuss the merits of our approach in Section 6.8 and finally Section 6.9 provides concluding remarks as well as potential future work.

6.1 Overview of Neural Networks

In this section we first provide a brief background on artificial neural networks in general (Section 6.1.1) before covering convolutional neural networks (Section 6.1.2). The basic building blocks for our 3D shape completion network (Section 6.4) are then described in Section 6.1.3.

6.1.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are constructed from a series of "layers" where each layer applies some computation to the input data before passing that data to the next layer. A layer consists of a collection of artificial neurons, which loosely model the neurons in a biological brain. Connections between layers thus realized through the connections of neurons from two different layers. These connections are weighted to represent the strength between two neurons. A strong connection implies that the input signal processed by the current neuron will be strengthen when it is passed to a connected neuron.

In mathematical terms, a layer is a function $\mathcal{F}$ which takes in an input x and passes it through one or more "neurons" and a non-linear activation function [19, 58] to produce some output y . A single neuron within a layer is typically represented as:

$$\mathcal{N}(x, w) = f\left(\sum_i^N w_i x_i + b\right) \quad (6.1)$$

where x , w , b are vectors of the inputs, weights and biases while f is the non-linear activation function. The power of an ANN does not come from a singular neuron, as it is limited to modeling linear functions, but from multiple neurons combined into a series of layers. More specifically, given a finite number of neurons arranged into multiple layers, with at least one layer which is "hidden", any continuous function (in $\mathbb{R}^n$) can be approximated when using a non-linear activation function. The non-linearity is what

allows for the linear combination of weights to be mapped to a range of values on a monotonic, real-valued function.

An ANN composed of n layers can then be represented as a composition of n different functions where a function $\mathcal{F}_n$ is the n^{th} layer. Thus an ANN is equal to the following:

$$\mathcal{F}(x) = \mathcal{F}_n(\mathcal{F}_{n-1}(\dots(\mathcal{F}_2(\mathcal{F}_1(x)))))) \quad (6.2)$$

The function $\mathcal{F}_1$ is then considered to be the input layer containing the data we want to process, while $\mathcal{F}_n$ is the output layer which contains the data representing the desired outputs. The functions $\mathcal{F}_2$ through $\mathcal{F}_{n-1}$ are then known as the hidden layers which can intuitively be thought of as the internal representations used to infer the final desired output. For example, if the input to an ANN is an image and the desired output is the detection of a face, the input layer would contain the input image and the output layer would contain data representing the region where there exists a face. The hidden layers could contain data representing locations of the eyes, mouth, nose and so on. In other words, the hidden layers would contain data of the *features* which help the ANN process the input image so that it is able to produce the desired output.

Deep ANNs refer to ANNs which have many hidden layers between the input layer and the final output layer. When an ANN is processing input in order to produce a prediction, it simply feeds the data into the input layer which is then processed by the subsequent hidden layers before arriving at the final output layer. The act of processing data from input layer to output layer is known as *forward propagation*. When the ANN is learning, *backwards propagation* is applied to update the weights of the connections so that they would more likely produce the desired output when fed a given input. More details on ANNs and precisely how their weights are updated can be found in the book by Bishop [6] and the book by Goodfellow et al. [49].

6.1.2 Convolutional Neural Networks

When ANNs are applied to images, a straight forward approach would consider the input image as a multi-dimensional array of numbers and each pixel would have a weight assigned to it. Thus given a RGB image of dimensions 32×32 , a single fully-connected neuron in the first hidden layer of an ANN would have $32 \times 32 \times 3 = 3072$ weights. As the image dimensions increase, the number of required weights would then scale quadratically. This dramatic increase in the total number of weights is further exasperated when processing 3D volumes as well as when we combine multiple neurons across a multitude

of layers. Thus, while this approach is perfectly valid, it can quickly lead to a massive number of weights which both increases the memory footprint of the ANN as well as potentially causing *overfitting*. Overfitting, in the context of machine learning, is a term which describes a model that has been too highly optimized to a particular dataset (usually the training data) and thus may fail to generalize. This means that a model which has overfit a particular dataset will perform very well for that dataset but perform poorly when presented additional data which the model has not processed before. Additionally, using fully connected neurons with images means that we do not take advantage of the local connectivity of images. That is, pixels which are closer together should have more influence than those which were further away.

The above limitations are what inspired the convolutional neural network (CNN) architecture which is now the back-bone of many solutions to problems in computer vision. The CNN belongs to the class of ANNs which are composed of "convolutional layers". Unlike standard layers from ANNs, convolutional layers arrange neurons in three dimensions: width, height and depth. Additionally, the output neurons are only connected to the input neurons within a local window (this is known as the *receptive field*) rather than the entire set of neurons from the input layer. These two properties allows for a reduction in the number of weights as well as producing features which directly exploit the inherent spatial structure of images. Producing an output data volume for the next layer is accomplished by sliding a *kernel*, a matrix of weights with the dimensions of the receptive field, over the input data volume. In convolutional layers, the weights being learned are contained within these kernels. Fig. 6.1 provides a visualization of the differences between fully connected and convolutional layers when applied to images. The online course from Stanford University: CS231n Convolutional Neural Networks for Visual Recognition is an excellent resource for additional information on the details of CNNs and ANNs in general [65].

6.1.3 Network Layers

Listed below are brief descriptions of the different layer types which are used in the rest of this chapter.

Fully Connected A standard ANN layer which connects all neurons from the input layer to the next layer.

Convolution A standard convolutional layer computes an output volume by sliding a

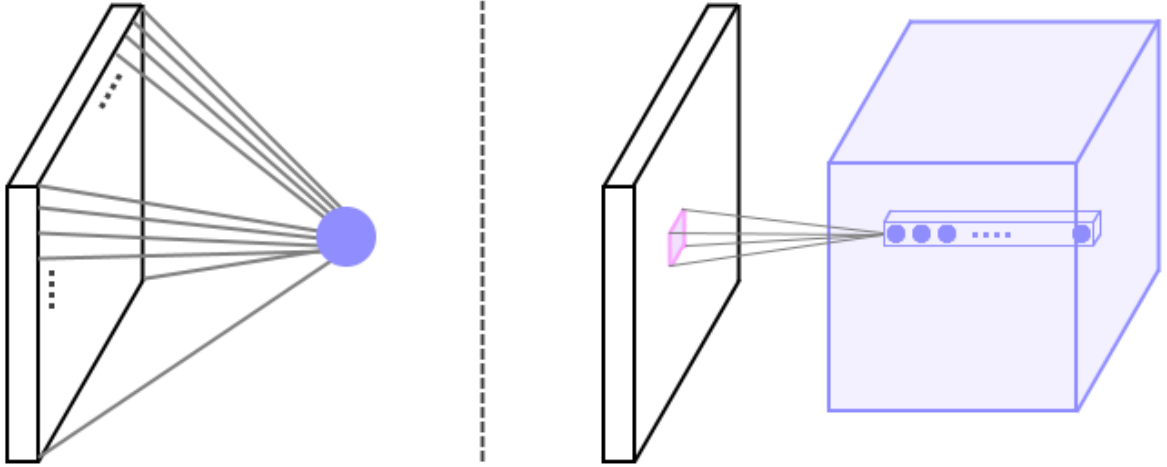


Figure 6.1: Illustration of the differences between a fully connected layer with one neuron (left) and a convolutional layer with an output volume (right) when applied to an image. The single neuron for a fully connected layer has weights for each pixel while a neuron from the output volume of the convolutional layer only has weights for a region (magenta) of the image. The fully connected neuron (left) must be replicated many times in order for a fully connected layer to produce the same output volume as the convolutional layer (right).

receptive field over the input volume. The exact computation performed is the dot product between the kernel weights and the numbers contained in the region covered by the receptive field.

Separable Convolutions There are two main types of separable convolutions: spatial and depthwise. Separable convolutions are named as such because they decompose the convolution kernel into pieces - that is they *separate* the convolution. Spatial separable convolutions deal with the spatial dimensions (width and height) of an image instead of the depth (i.e: number of channels). An example of a spatial separable convolution is the Sobel kernel used in the classic Sobel edge detector. Eq. 6.3 shows a decomposition of a Sobel kernel that computes the horizontal derivative of an image.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad (6.3)$$

The Sobel kernel is a special case where it can be cleanly decomposed into two

smaller kernels. The kernels which are learned by a CNN are more general and can contain numbers which make the decomposition as shown in Eq. 6.3 impossible. Depthwise separable convolutions were designed specifically to address this issue and allow for kernels to be decomposed in the general case. Depthwise separable convolutions consist of a *depthwise convolution*, a spatial convolution performed independently over each channel of the input data volume, and then followed by a *pointwise convolution*, which is a 1×1 convolution that projects the channels output by the depthwise convolution onto a new channel space. Fig. 6.2 provides an illustration and more details can be found in the paper which extended, and popularized, the depthwise separable convolution for CNNs [11].

ConvTranspose Intuitively this layer can be thought of as an up-sampling method but with learn-able parameters. Standard up-sampling in image processing involve a specific interpolation method which decides how exactly to fill the expanded regions created when we grow a given image along its dimensions. For example, nearest neighbor interpolation for images picks the value of the closest point to the location which we wish to interpolate and does not consider any of other neighboring values. The ConvTranspose layer performs a similar operation as a convolution but instead applies the transpose to the convolution matrix. Dumoulin and Visin [29] provides a detailed examination into the convolutional arithmetic involved with ConvTranspose operations.

Residual Blocks Introduced by He et al. [54], this technique allows for training of even deeper networks by joining the outputs of multiple layers with the inputs of prior layers. Experimental results [148, 53] have shown that simply adding more layers to obtain a deeper model results in accuracy saturating and higher training error. This phenomenon is known as the *degradation* problem and is not caused by overfitting. Intuitively, the degradation problem exists because it cannot be assumed that only the features learned at one layer are required for the next layer. It's possible that a layer needs the information from many other previous layers in order to properly learn a strong representation. The identity shortcut connection allows more detailed information to be considered rather than just the abstract representation from a single layer. Fig. 6.3 illustrates the residual block as described in the original paper by He et al. [54].

Rectified Linear Unit This is the most widely used activation function in the context

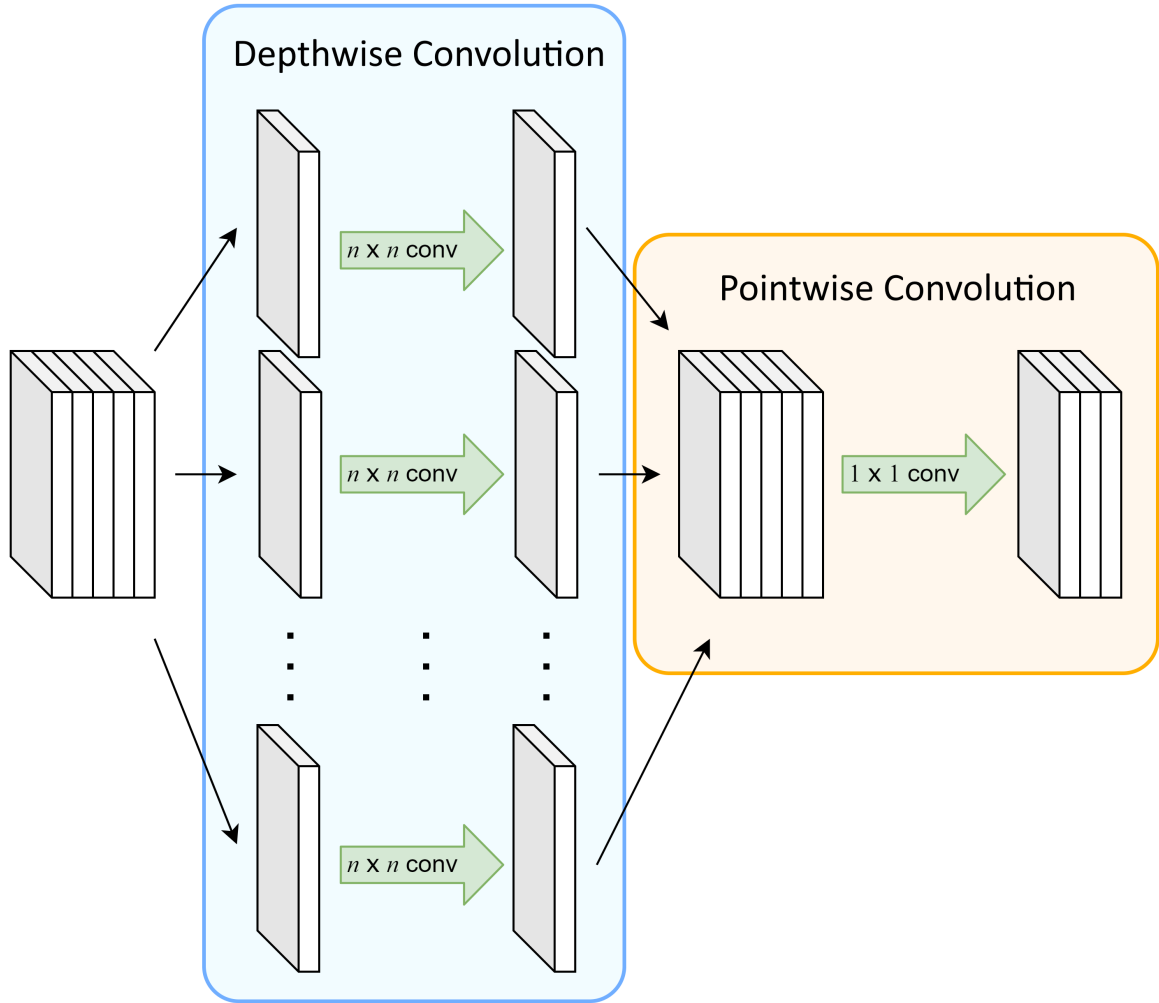


Figure 6.2: Diagram of the depthwise separable convolution. The blue region shows the $n \times n$ convolutions performed on each channel of the input data volume separately. The orange region shows the pointwise convolution which transforms the results of the depthwise convolutions into the desired output data volume.

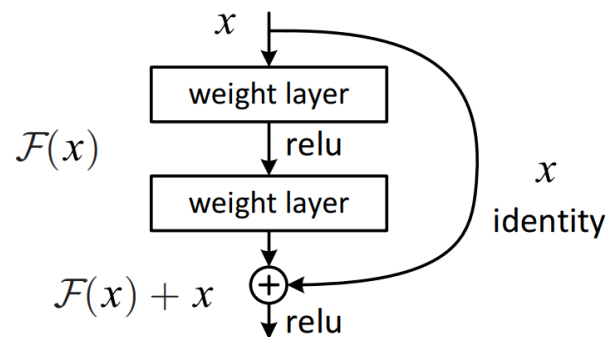


Figure 6.3: Diagram from He et al. [54] who originally introduced the residual block.

of deep ANNs [85]. The rectified linear unit (ReLU) is defined as follows:

$$f(x) = \max(0, x) \quad (6.4)$$

where x is the input to a neuron. It gained wide-spread use when it was found to enable training of deeper networks [46] when compared to other activation functions which were common at the time (circa 2011).

Max-Pooling This is a particular type of "pooling" layer which is used as a form of dimensionality reduction to help reduce the computational requirements of CNNs. In essence, pooling layers down-samples the input data volume based on a heuristic. This is accomplished by selecting a region of numbers from the input volume via a receptive field and then using the desired heuristic to pick a single number to be the representative of that region. As the name implies, Max-Pooling uses the $\max()$ function as the heuristic and thus takes the maximum value from the region covered by the receptive field.

Batch Normalization This technique was introduced to reduce the amount of *covariate shift* for the weights of hidden units within ANNs. Normally when we feed data to the input layer we *normalize* the numerical values to be in the range of 0 to 1 (for example). This is done because certain features of a dataset may have values which range over larger values than other features (i.e: price of a house vs number of bedrooms). As both features are important, we do not want one particular feature to dominate other features simply because its values have a larger numerical range. This same concept applies to the units of hidden layers as well. When data passes through several layers, certain connections (i.e: weights) may become much stronger than others. This "skews" the network for subsequent layers much like how a network would be skewed towards certain features if we did not normalize the input data. Batch normalization solves this problem by normalizing the weights before passing the results to the next layer in the network. This technique has shown to improve training time and performance which is why it is now widely adopted in many different approaches which involve deep ANNs. For more details see the original paper by Ioffe et al. [59].

Dropout This technique simply ignores a set of neurons during the network training phase. More precisely, each neuron (and its connections) of a particular layer which implements dropout has $1 - p$ probability of being "dropped" resulting in a reduced

network structure. These dropped neurons are then not considered during both the forward and backwards pass of training and thus their weights are not updated. The intuition behind dropout is to allow the network to learn more robust features rather than allowing the same neurons to always activate when given a particular input set of inputs. By learning more robust features which are not as dependent on specific neurons activating, dropout can regularize a network which helps with overfitting. For more details see the original paper by Srivastava et al. [147].

6.2 Related Works in 3D Shape Completion

Most approaches for 3D shape completion fall into two main categories: model matching and learning-based. In this section we first briefly review the model matching approaches before covering the more related learning-based methods. We then review approaches which aim at reducing the complexity and memory requirements of 3D CNNs.

Model matching. In model matching the problem is often framed as finding the closest match through a database of objects [168, 92, 126]. Wohlking et al. [168] rendered depth maps of potential candidate 3D models for matching against the input depth map. Li et al. [92] detect key points in order to register known models against similar objects encountered while scanning the environment with a depth sensor. Rock et al. [126] combines 3D model matching with mesh deformation in order to transfer surface symmetries and details to the matched model. A common weakness of these approaches is the assumption that shapes within the database are similar enough to be matched.

Learning-based. In learning based approaches, a CNN is trained from a dataset of models and a complete 3D shape is outputted from the partial input. Existing approaches have covered a number of input scenarios ranging from RGB images [153, 170] and 2D sketches [23, 100] to voxel grids [136, 160, 20, 151]. Some approaches adopt a two-stage reconstruction pipeline by fusing together network predictions in order to produce a mesh through other means while others train the CNN end-to-end and predict a voxel grid or signed distance field (SDF). Some approaches are fully supervised and others are generative. We briefly cover these different input scenarios before focusing on works most related to ours.

The network proposed in Tatarchenko et al. [153] takes as input a RGB image and a desired viewpoint to predict a RGB image and depth map of the object as viewed from the input viewpoint. Multiple views can then be predicted from the single RGB input

and the results fused together to produce a complete 3D model. Wu et al. [170]. take a generative approach instead and learn an encoding of input RGB images via a variational auto-encoder [74, 82] and then train a Generative Adversarial Network (GAN) to predict a voxel grid from the learned latent vectors.

Sketch based approaches [23, 100] take as input one or more 2D contour images and output a voxel grid. Lun et al. [100] concatenates the input contour images and feeds it into their network which outputs 12 depth and normal maps. The output depth and normal maps represent multiple views of the target object and are fused together to obtain a mesh model. Delanoy et al. [23] make use of two networks, one to predict the 3D shape from a single view and another which takes as input the single view prediction and updates an accumulated volume.

ShapeNets [171] represent early work in 3D shape completion of single view depth maps. Their model is generative and utilizes multiple iterations (50 in the original paper) of Gibbs sampling in order to get a completed shape with a low resolution of 32^3 voxels. Sharma et al. [136] take an unsupervised approach and uses a 3D de-noising auto-encoder and also completes 3D shapes at low resolution (32^3). In contrast, Varley et al. [160] uses an encoder to progressively shrink an input volume but does not perform progressive decoding like in an auto-encoder. Instead, they simply encode the input and follow up with two dense layers before a final layer of 64000 nodes which is then reshaped into a 40^3 occupancy grid. Dai et al. [20] combine the results of [171] with shape synthesis using a known database in order to transform a 32^3 voxel output into a 128^3 voxel grid. Stutz et al. [151] first train a de-noising variational auto-encoder [74] on a set of voxel grids before transferring the decoder portion to a new encoder-decoder model. This new model is then trained for the task of shape completion using an unsupervised maximum likelihood loss. In this way, the pre-trained decoder restricts the space of possible shapes while the loss function aligns the prediction shapes with the input. A common thread amongst most of these approaches is the low resolution (32^3) and the requirement of voxelizing the input depth maps.

Reducing memory requirements. Higher resolution 3D CNNs are difficult to train due to the cubic increase in computational and memory requirements. Prior works in 3D deep learning have tried to mitigate this problem by moving to alternative representations of 3D data such as point clouds [122, 123, 36], geometric images [142] and Octrees [154, 125].

Qi et al. [122] introduced PointNet, a point cloud based approach that allows training

CNNs on unordered set of points, and refined it in PointNet++ [123]. However their approach is for classification and segmentation tasks not 3D shape completion. Fan et al. [36] introduce a point cloud network which tackles this problem for RGB images. However, their output size is fairly low at 1024 points thus reconstructing a mesh model to include any preserved thin structures and details can be difficult.

Sinha et al. [142] convert 3D shapes into what they call a 2D "geometry image" through area preserving parameterization in the spherical domain. In essence, their approach "flattens" a 3D model into the plane much like UV mapping [40] in computer graphics. Then conventional 2D CNNs can be used to learn from these images. A weakness of this approach is that surface parameterization is only applicable to certain topologies.

Tatarchenko et al. [154] and Riegler et al. [125] both approach the problem of high resolution voxel grids via Octrees. In this way, they can exploit the sparsity of voxel grids to achieve faster training and lower memory requirements. The major dis-advantage of these approaches is the requirement to re-write standard convolutional layers to utilize the sparse Octree format. Additionally, existing data pipelines will need to adapt to the sparse Octree format as well which can be cumbersome.

6.3 Additive Depth Maps

Let d_f and d_b be the depth maps as shown in Fig. 6.4. The additive depth, d_{ad} , is then equal to $d_b - d_f$. Thus when given d_f , shape completion can be accomplished by predicting d_{ad} and reconstructing d_b via $d_f + d_{ad}$. The union of d_f and d_b is the completed shape which can be refined into a water tight mesh through classical approaches like the Poisson surface reconstruction [67].

When compared to voxel representations, the advantages of using the additive depth map for shape completion are as follows:

1. It is the minimum representation for 3D shape completion using structured data. Multi-view depth map fusion approaches [153, 100] require multiple depth maps and sometimes normal maps as well in order to perform reconstruction. Even sparse volumetric based approaches like Octrees [154, 125] still require more memory than a single depth map of equivalent side length. A 256^3 volume will require at least the same amount of memory to store a 256^2 depth map viewing that same volume. Only point cloud networks [36], which make use of unstructured 3D data, can

potentially be smaller at the cost of decreased detail and increased difficulty in meshing, especially thin structures.

2. Using images rather than volumes lowers computational and memory requirements for all aspects of the data pipeline from training to prediction and the data itself. This allows for higher resolutions to capture more details and potentially more complex shapes.
3. Finer details can be encoded in a depth map relative to the amount of memory needed for an equivalent number of voxels. For example, a depth map of dimensions 256^2 can potentially include thin structures at the resolution of a 256^3 volume. A voxel grid storing the equivalent amount of information as a 256^2 depth map can only have a dimension of approximately 40^3 .
4. Conventional 2D CNNs can be trained to predict depth maps and thus it is fairly simple to include additional data such as color images or semantic labels with little computational cost. For the equivalent 3D CNN analogue, it is not immediately clear how to effectively include color or semantic labels for the memory intensive task of 3D shape completion via volumes.

The main disadvantage of additive depth maps is the fact that even with a front and back depth map, the completed shape can still be missing some portions like internal structures. Thus, objects with lots of self-occlusion will be difficult to reconstruct. Additionally, reconstructing a water tight surface would require additional post processing similar to prior works [153, 100]. This disadvantage is a natural trade-off for using a data minimal 2D representation to model 3D data.

The reduction of 3D data into a 2D image for training 2D CNNs is conceptually similar to the approach by Sinha et al. [142] but differs in methodology and representation. A key difference between our approach and the one by Sinha et al. [142] is that we are not limited to parameterizable topologies. Additionally we only require one image, as opposed to three (one for each Cartesian dimension), to preserve 3D shapes. Reconstruction is also simpler for additive depth maps since only addition is required, while Sinha et al. [142] needs to undo the parameterization to recover the 3D information.

In terms of data representation, our additive depth map is most similar to the principle thickness images (PTI) used by Shu et al. [139] to perform 3D model classification and retrieval. The PTI is defined as a combination of three grayscale images where each pixel is a discrete value which captures the "thickness" of the voxelized input mesh as viewed

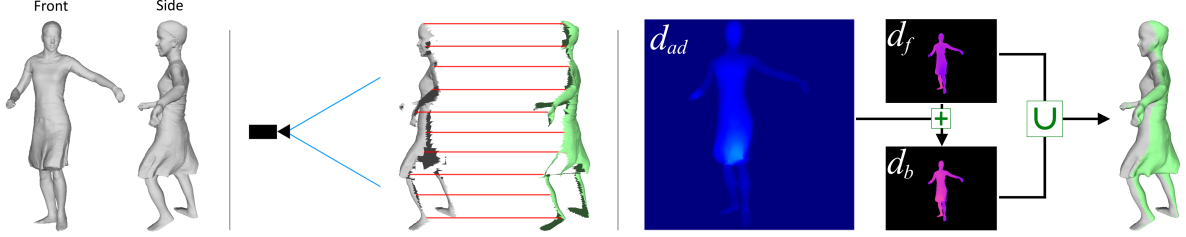


Figure 6.4: Illustration of the additive depth and its use in shape completion. On the left is the ground truth mesh viewed from the front and side. Next is a camera imaging the mesh to produce the training data. The additive depth, d_{ad} , is visualized as the (exaggerated) red lines from the camera facing mesh (in grey) to the back view of the mesh (in green). To the right is d_{ad} shown as an image as well as the depth maps d_f and d_b which correspond to the grey and green partial meshes, respectively. The union of d_f and d_b produces there mesh result shown in the far right. For all depth map images, brighter colors imply further distances.

from a particular viewpoint. In contrast, the pixel values of the additive depth map represent continuous distances between two points on the mesh surface. This difference is analogous to voxel occupancy grids and signed distance fields in the sense that both can represent volume but signed distance fields are better at representing surfaces of meshes. Thus, when it comes to the topic of shape completion, the additive depth map is a better representation since it (1) does not require voxelization of the input and (2) can be added directly to the input depth map without post-processing.

Our approach is also similar to the approaches by Lun et al. [100] and Tatarchenko et al. [153] in that we predict depth maps in order to perform shape completion via meshing. A key difference is that the additive depth map implicitly encodes the 3D shape and does not require multiple views for shape completion. This difference is both a strength and weakness since less resources are required but self-occlusions are more difficult to handle.

6.4 Network Architecture

Encoder-decoder models with skip connections have been successful in a variety of applications such as semantic segmentation [127, 95], object detection [5] and image restoration [103]. We take inspiration from these works and use an encoder-decoder model with max pooling layers to perform contraction and conv transpose [175] layers for expan-

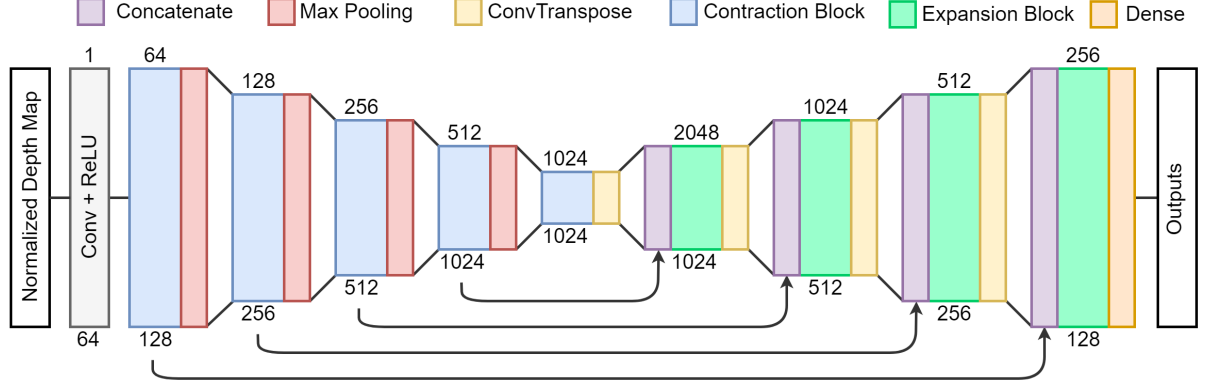


Figure 6.5: General structure of the network. The numbers at the top/bottom of each block indicates the number of incoming/outgoing feature maps.

sion. A stride length of 2×2 was used for both max pooling and conv transpose layers. Thus, during contraction the image dimensions halves while during expansion it doubles. We also incorporate residual blocks [54] in the encoder portion to further deepen our model without facing degradation or vanishing gradients. Fig. 6.5 provides a general visualization of the network.

The contraction blocks (blue) from Fig. 6.5 are composed of two pre-activation residual blocks as described in He et al. [55] where we use separable depthwise convolutions [11] with a kernel size of (3×3) as the weight layers. Additionally when the input number of features (d_{in}) is different from the output number of features (d_{out}), we apply a convolution with kernel size (1×1) to the input layer so that the shortcut will have the same dimensions as the output layer. The expansion blocks (green) first perform a standard 2D convolution on the incoming feature maps before applying a ReLU activation [108], batch normalization [59], dropout [147], separable depthwise convolution, another ReLU activation and finally a second dropout. Additionally, the conv transpose layers after each expansion block outputs half the number of input feature maps while doubling the image dimensions. See Fig. 6.6 for a visualization of the contraction and expansion blocks.

Finally, the dense portion (orange) from Fig. 6.5 varies depending on the outputs (see Fig. 6.6). In our experiments we examined the effects of multi-task learning the additive depth map d_{ad} with two additional outputs: the normal map of d_{ad} and the depth labels image of d_{ad} . For more details on these outputs see Section 6.5.

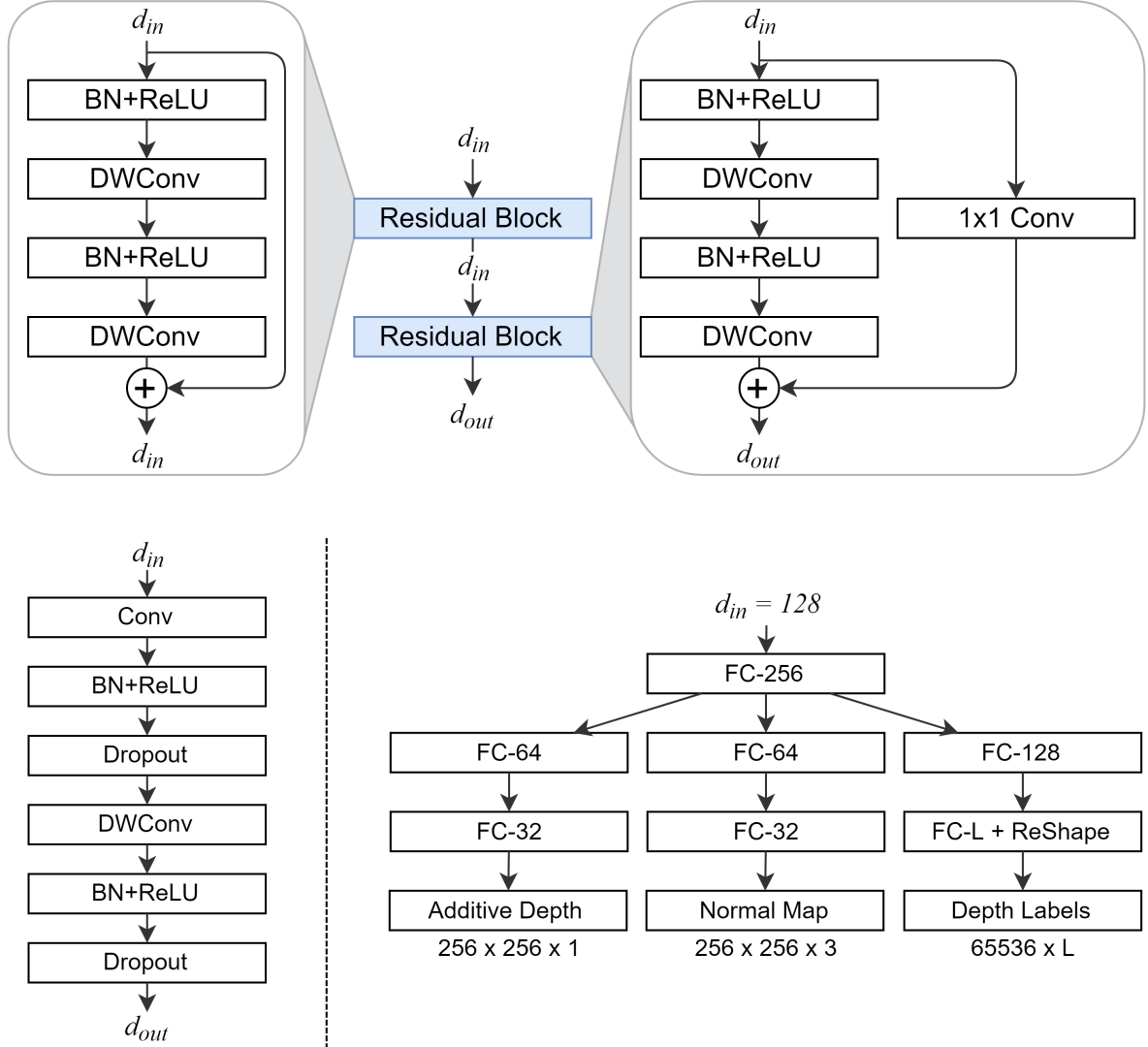


Figure 6.6: Detailed visualization of the contraction (top), expansion (bottom left) and dense (bottom right) blocks from Fig. 6.5. DWConv represents a separable depth-wise convolution, d_{in} is the number of input feature maps while d_{out} is the number of output feature maps. For the dense blocks, networks which do not have a particular output will have their dense branch removed. The variable L refers to the number of depth labels.

6.5 Data Generation

We utilize the SHREC’17 dataset [132] which is a subset of meshes derived from ShapeNet [10] and mesh models of clothed humans performing dynamic movement from Vlastic et al. [162]. A description of each dataset is provided below:

SHREC’17

This subset of ShapeNet [10] consists of 51162 3D mesh models grouped into 55 categories. Standard training data splits were also provided which we adopted in our experimental setup. The mesh models only contained geometric information and the model dimensions were normalized to a unit length (we consider this to be 1 meter) cube. Additionally, all the mesh models are consistently aligned - up right and front facing.

Clothed Human

This dataset consists of clothed male and female mesh model sequences produced through multi-view video recordings [162]. Not all of the sequences described in the original paper were publicly available - we used 8 sequences each of which had between 150 to 250 frames leading to a total of 1500 mesh models of clothed humans. From visual inspection, we found that 6 of the sequences feature two different male subject while the remaining sequences feature the same female subject.

Let d_f and d_b be the depth maps as shown in Fig. 6.4. For both datasets, we generate k depth map pairs (d_f, d_b) of dimensions $w \times h$ per mesh m by applying a random rotation to m and then positioning them three meters away, centered within the field of view of a virtual camera c at the origin. The depth map pair (d_f, d_b) is rendered by casting rays from c through m and re-projecting the set of ray-mesh intersection points, S , back onto the image plane of c . If we are rendering d_f we select $p_f = x | \min_{x \in S} D(x)$ for image plane re-projection and $p_b = x | \max_{x \in S} D(x)$ for rendering d_b . Here $D(x)$ is the Euclidean distance between a point x and c . Thus for each pixel location in d_f and d_b , the depth value is the distance to c via p_f and p_b , respectively. The additive depth, d_{ad} is then equal to $d_b - d_f$. Fig. 6.7 provides visualizations of the original mesh models and the associated depth maps generated for training.

For multi-task learning we experimented with normal maps and depth based labeled images. The normal maps were computed for the additive depth maps by exploiting the regular grid structure of images. By meshing this grid, a normal vector for each pixel

is found through the neighboring pixels. Depth based labeled images are essentially discretized versions of the additive depth maps. Let $L \geq 2$ be the number of labels, $\lambda > 0$ be a user-defined maximum additive depth value and d_{ad} be an additive depth map. The depth based labeled image of d_{ad} has L channels with each channel $ch(i)$ representing the set of pixels $p \in d_{ad}$ whose additive depth values are in the range $[(\lambda/L) \times (i - 1), (\lambda/L) \times i]$. For our experiments we set λ to be 2 meters since none of the mesh models occupied a volume larger than $2 \times 2 \times 2$ meters and we experimented with varying values of L . For more details see Section 6.6.

6.6 Experimental Setup

We train separate models for 6 of the SHREC’17 categories and also one model for the clothed human dataset. The categories from SHREC’17 were chosen for their shape features and a brief summary is provided in Table 6.1.

The input to the network is a 256×256 normalized depth map such that the minimum and maximum values have been re-mapped to $0 \rightarrow 1$. The output of interest is the additive depth map d_{ad} scaled to $0 \rightarrow 1$ using the λ value chosen in Section 6.5. We experiment with multi-task learning for the combinations of outputs as described in Section 6.5. Our networks were trained for 20 epoches with a batch size of 2 where one epoch processes every sample from the generated training data once. We optimize our network using ADAM with the default parameter values as suggested in the original paper [73].

For the SHREC’17 dataset we follow the data split as suggested by the original authors [132]. Since no training split was provided for the clothed human dataset, we use all 1500 mesh models to perform an 80/20 split via uniform random sampling. For both datasets, we then generate depth maps of the mesh models for the training and validation sets according to Section 6.5. We further augment the generated depth maps in real-time during training by (1) introducing small random noise to the input foreground depth values and (2) randomly zeroing portions of the input depth map. The purpose of these augmentations are to simulate sensor noise and error as well as encouraging the learning of more robust features. We set the probability to apply either augmentation to be 50%.

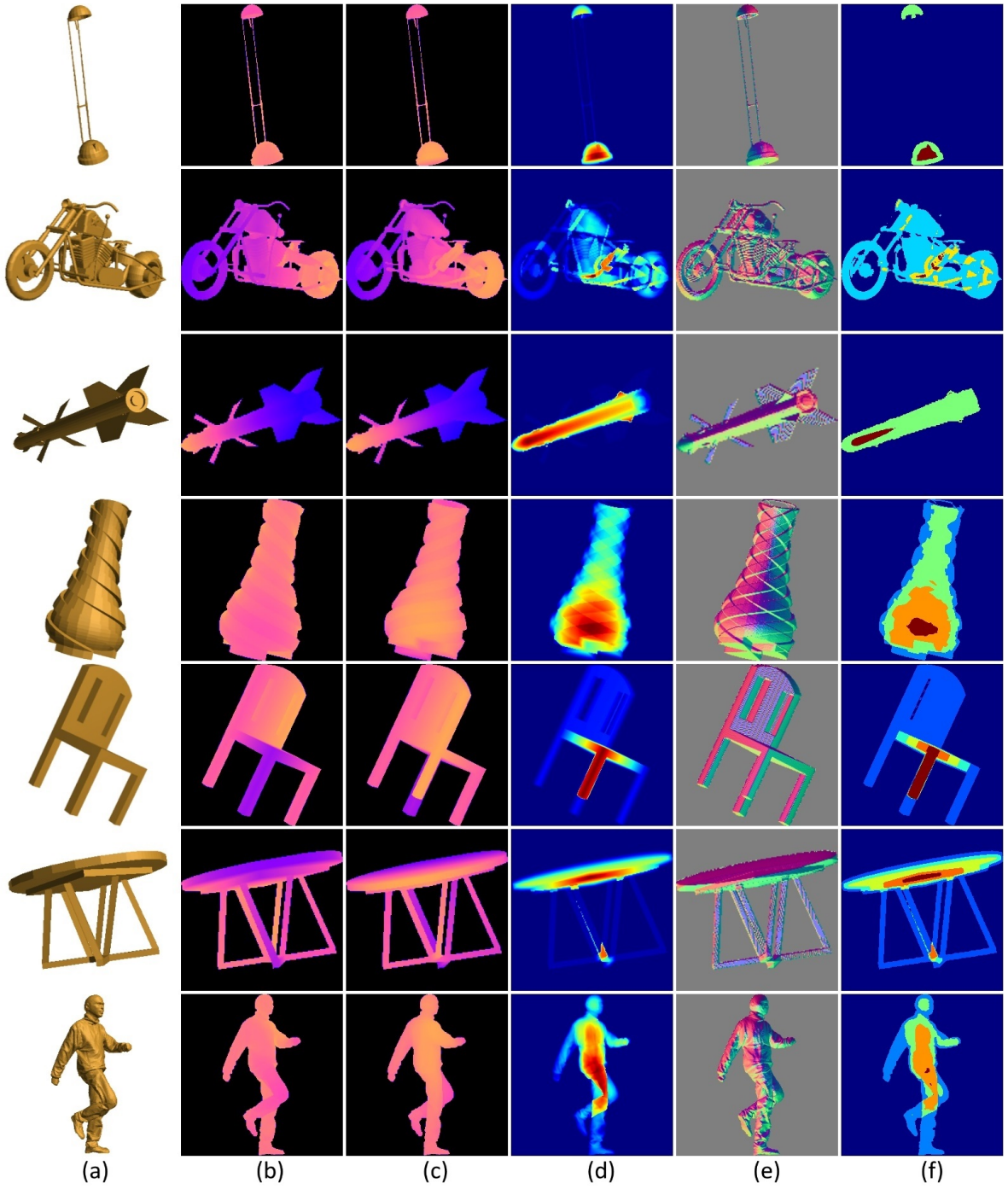


Figure 6.7: Examples of training data generated from mesh models. (a) Original mesh model, (b) front facing depth map d_f , (c) back facing depth map d_b , (d) the additive depth map $d_{ad} = d_b - d_f$. For both d_f and d_b , darker colors has lower depth value while for d_{ad} blue has lower depth value. (e) normal map of d_{ad} and (f) depth labels image of d_{ad} with $\lambda = 2.0$ and $L = 16$.

Categories	Shape features	# of training samples (train/val)
lamp	Consists of very thin structures (lamp poles and wires) which can be hard to preserve well volumetrically without a high enough resolution.	16200 / 2310
motorbike	The engine parts exhibit complex structures which will be difficult to reconstruct.	2350 / 340
rocket	Small thin structures exist as the fins on the rockets.	3540 / 90
vase	Broadly speaking, has simple structures on the surface with some vases being a smooth surface and others with sharp edges.	7600 / 630
chair	The legs can be completely occluded which makes reconstruction challenging.	23060 / 3305
table	Similar features to the chair.	58760 / 4190

Table 6.1: Categories of mesh models from SHREC’17 that we used in our experiments. Each of the mesh models were rotated randomly before a training sample was generated from one view.

6.7 Experimental Results

In this section we first describe our evaluation measures (Section 6.7.1) before examining the effects of multi-task learning (Section 6.7.2). Using the best performing model, we summarize our results for shape completion through depth map and volumetric evaluation measures in Sections 6.7.3 and 6.7.4, respectively. In Section 6.7.5 we demonstrate our approach on real world data without fine tuning. Lastly, Section 6.7.6 explores the effects of combining clothed body part semantics with our best performing model for 3D clothed human shape completion.

6.7.1 Evaluation Measures

As we are interested in using the predicted additive depth maps in order to reconstruct a depth map based on the input for 3D shape completion, we evaluate our approach using depth map error and accuracy measures typically used for single color image to depth map prediction tasks [81, 31, 91, 48]. These measures are defined as follows:

Mean relative error (rel):

$$\frac{1}{|T|} \sum_{d \in T} |\hat{d} - d|/d \quad (6.5)$$

Root mean squared error (rmse):

$$\sqrt{\frac{1}{|T|} \sum_{d \in T} \|\hat{d} - d\|^2} \quad (6.6)$$

Mean $\log_{10}$ error ($\log_{10}$):

$$\frac{1}{|T|} \sum_{d \in T} |\log_{10} \hat{d} - \log_{10} d| \quad (6.7)$$

Threshold ($\delta < thr$):

$$\% \text{ of } d_i \text{ s.t. } \max\left(\frac{\hat{d}_i}{d_i}, \frac{d_i}{\hat{d}_i}\right) = \delta < thr \quad (6.8)$$

where d is the ground truth depth, $\hat{d}$ is the estimated depth and T is the set of all pixels in the image.

We evaluate our approach using the above measures by reconstructing the back depth map and comparing it against the ground truth. More specifically, for an input depth map d_f and its associated ground truth d_b , we feed d_f into our network to predict $\overline{d_{ad}}$ which we add to d_f to obtain $\overline{d_b}$. The depth map error and accuracy measures are then applied to d_b and $\overline{d_b}$.

Since 3D shape completion using deep learning is often measured volumetrically, we also evaluate our approach using volumetric error and accuracy measures typically found in voxel based approaches for 3D shape completion [20, 151, 94]. These measures are defined as follows:

Intersection over union (IoU):

$$|v \cap \hat{v}|/|v \cup \hat{v}| \quad (6.9)$$

Hamming distance (Ham):

$$\text{hamming_distance}(v, \hat{v})/N^3 \quad (6.10)$$

Surface accuracy (Acc):

$$\frac{1}{|F(\hat{v})|} \sum_{x \in F(\hat{v})} D(x, v) \quad (6.11)$$

Surface completeness (Comp):

$$\frac{1}{|F(v)|} \sum_{x \in F(v)} D(x, \hat{v}) \quad (6.12)$$

where v is the ground truth voxel grid, $\hat{v}$ is the predicted voxel grid, N is the side length of the voxel grid, $F(g)$ is the set of filled cells in a voxel grid g and $D(x, g)$ is the minimum $\mathcal{L}_1$ distance from x to a filled cell in the voxel grid g .

6.7.2 Multi-Task Learning

We experimented and measured the performance of learning the following outputs combinations: (1) only the additive depth d_{ad} , (2) d_{ad} and d_n - the normal map of d_{ad} , (3) d_{ad} , d_n and d_L - the depth labels image and (4) d_{ad} and d_L only. We trained separate models for each of these four output combinations using the clothed human dataset with $L = 128$.

Since d_L is essentially a "semantic labels" image often seen in semantic segmentation tasks [95, 31], we also train two additional models exploring the effects of using binary cross entropy (BCE) and categorical cross entropy (CCE) as the loss functions for d_L . The reverse Huber [81] loss was applied for d_{ad} and d_n since it has achieved state-of-the-art performance for the task of single color image to depth map estimation.

To determine which combination performed best, we evaluated these 6 models using the depth map error and accuracy measures as described in Section 6.7.1. Jointly learning both d_{ad} and d_L performed best across all measures (Table 6.2). As the number of labels, L , for the depth labels image d_L is defined at the data generation stage (Section 6.5), we also perform experiments varying the value of L . We found a minor performance boost when increasing the value of L from 16 to 128 (Table 6.3).

6.7.3 Depth Map Evaluation

Since our depth maps are of single objects, pixels representing the background have depth values of zero. Thus, we exclude the background pixels when applying the depth map error and accuracy evaluation measures. Tables 6.2 to 6.4 summarize the results of depth map based evaluation. Figs. 6.8 and 6.9 provide visualizations of our reconstructions.

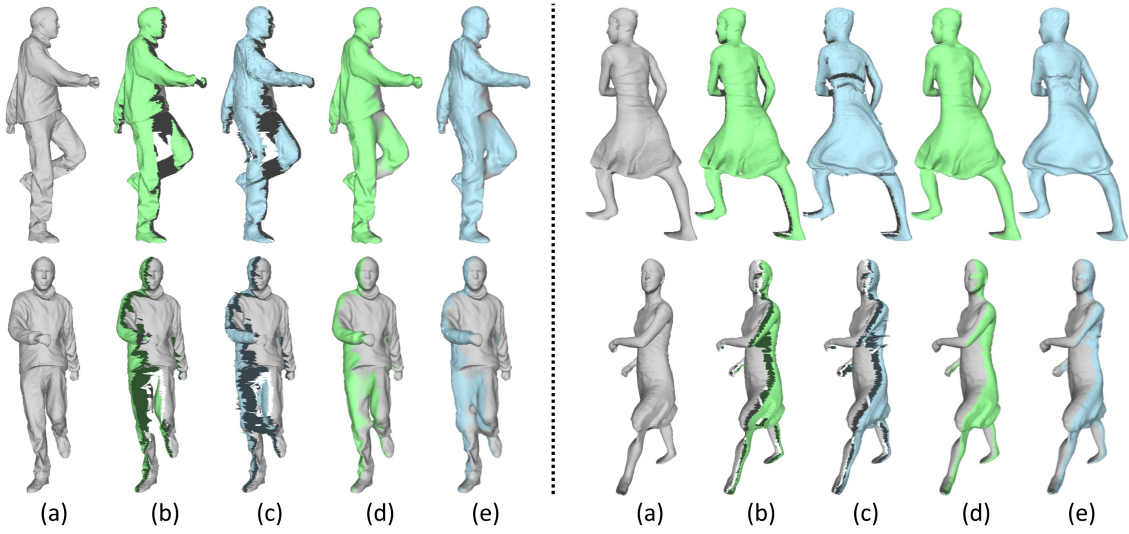


Figure 6.8: Example meshed outputs of a male (left) and female (right) from the clothed human dataset shown from two different views (top/bottom). The input is shown in grey, ground truth in green and the network prediction in blue. For both examples, from left to right, we have (a) the original mesh, (b) ground truth mesh from depth maps, (c) prediction mesh from depth maps, (d) applied Poisson surface reconstruction to results of (b) and (e) applied Poisson surface reconstruction to results of (c).

When evaluating depth maps predicted from single color images, most approaches report the threshold accuracy (Eq. 6.8) using the values $thr = 1.25$, 1.25^2 and 1.25^3 . We found that all of our networks could achieve the ideal accuracy value of one and thus we used stronger thresholds values ($thr = 1.01$, 1.02 , 1.03 , 1.05 and 1.10). This is likely because: (1) our targets are objects within the same class instead of complex indoor scenes and (2) the desired range of the additive depth maps was within 2 meters while other methods have depth maps with larger and more varied depth values.

Overall, the model trained on the clothed human dataset performed best with a threshold accuracy of 0.886 with $thr = 1.01$ and the lowest error across all measures. In contrast, the SHREC’17 models could only achieve a similar threshold accuracy with $thr = 1.05$ suggesting that the SHREC’17 dataset is more difficult. Intuitively this makes sense since non-human objects, such as chairs, have a wider range of possible shapes (e.g.: chairs do not always have four legs).

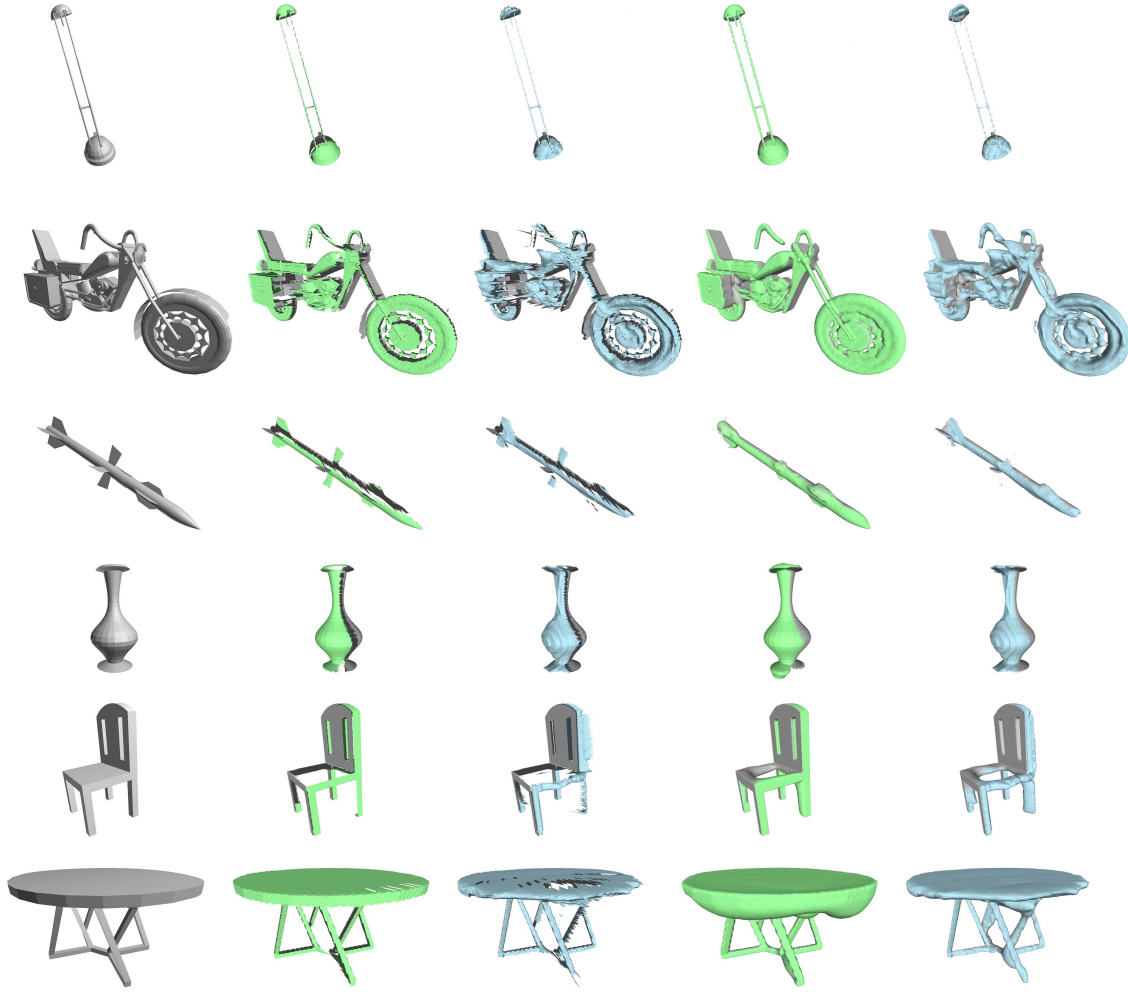


Figure 6.9: Example meshed outputs of lamp, motorbike, rocket, vase, chair and table categories from the SHREC'17 dataset. From left to right: original mesh model, ground truth, prediction, ground truth with Poisson surface reconstruction and prediction with Poisson surface reconstruction. As in Fig. 6.8, the input is shown in grey, ground truth in green and the network prediction in blue.

Outputs	d_L loss	Error (lower is better)			Accuracy (higher is better)				
		rel	rmse	log10	$\delta < 1.01$	$\delta < 1.02$	$\delta < 1.03$	$\delta < 1.05$	$\delta < 1.10$
d_{ad}	-	1.43×10^{-2}	6.45×10^{-2}	6.28×10^{-3}	0.522	0.751	0.881	0.962	0.994
$d_{ad} + d_n$	-	5.01×10^{-2}	1.64×10^{-1}	2.12×10^{-2}	0.023	0.065	0.129	0.470	0.990
$d_{ad} + d_n + d_L$	BCE	1.10×10^{-2}	4.97×10^{-2}	4.81×10^{-3}	0.607	0.885	0.945	0.978	0.996
$d_{ad} + d_n + d_L$	CCE	7.22×10^{-3}	3.85×10^{-2}	3.15×10^{-3}	0.812	0.944	0.970	0.986	0.996
$d_{ad} + d_L$	BCE	6.95×10^{-3}	3.80×10^{-2}	3.05×10^{-3}	0.817	0.925	0.961	0.986	0.997
$d_{ad} + d_L$	CCE	5.47×10^{-3}	3.22×10^{-2}	2.39×10^{-3}	0.886	0.961	0.977	0.989	0.997

Table 6.2: Depth map error and accuracy results of different multi-task learning output combinations and depth label loss functions for the clothed people dataset.

L	Error (lower is better)			Accuracy (higher is better)				
	rel	rmse	log10	$\delta < 1.01$	$\delta < 1.02$	$\delta < 1.03$	$\delta < 1.05$	$\delta < 1.10$
16	7.54×10^{-3}	3.88×10^{-2}	3.29×10^{-3}	0.799	0.932	0.964	0.984	0.996
64	6.83×10^{-3}	3.74×10^{-2}	2.98×10^{-3}	0.835	0.940	0.967	0.984	0.996
128	5.47×10^{-3}	3.22×10^{-2}	2.39×10^{-3}	0.886	0.961	0.977	0.989	0.997

Table 6.3: Effects of varying L , the number of depth labels, for the clothed human dataset.

Category	Error (lower is better)			Accuracy (higher is better)				
	rel	rmse	log10	$\delta < 1.01$	$\delta < 1.02$	$\delta < 1.03$	$\delta < 1.05$	$\delta < 1.10$
lamp	1.93×10^{-2}	5.67×10^{-2}	8.46×10^{-3}	0.514	0.714	0.805	0.896	0.972
motorbike	2.44×10^{-2}	6.86×10^{-2}	1.07×10^{-2}	0.411	0.592	0.709	0.850	0.963
rocket	1.24×10^{-2}	3.55×10^{-2}	5.43×10^{-3}	0.680	0.837	0.901	0.955	0.986
vase	2.51×10^{-2}	6.84×10^{-2}	1.09×10^{-2}	0.325	0.541	0.697	0.895	0.970
chair	3.54×10^{-2}	1.05×10^{-2}	1.57×10^{-2}	0.347	0.514	0.627	0.754	0.892
table	1.60×10^{-2}	5.90×10^{-2}	7.10×10^{-3}	0.652	0.819	0.880	0.930	0.971
Average	2.21×10^{-2}	6.55×10^{-2}	9.72×10^{-3}	0.488	0.669	0.770	0.880	0.959

Table 6.4: Depth map error and accuracy results for models trained on different categories from SHREC'17.

6.7.4 Volumetric Evaluation

Let d_f , d_b and $\overline{d_b}$ be the depth maps as defined in Section 6.7.1 and m_{ori} be the original rotated mesh model used to generate d_f and d_b . We first mesh the depth maps via the image grid structure and then voxelize the result. More specifically, for the depth map pairs (d_f, d_b) and $(d_f, \overline{d_b})$ we union each pair (see Fig. 6.4) into the meshes m_{gt} and m_{pred} , respectively. Furthermore, we apply the Poisson surface reconstruction [67] with

an Octree depth of 10 and the default parameters as suggested by the original paper to m_{gt} and m_{pred} to obtain the water tight versions PSR_{gt} and PSR_{pred} for more extensive comparisons.

In order to perform volumetric evaluation with m_{ori} we need a voxelization function $\mathcal{V}(m)$ which takes a mesh m and outputs a voxel occupancy grid g . We use two different voxelization functions: exact and the approach by Nooruddin and Turk [115]. The exact approach, which we denote as $\mathcal{V}_e(m)$, takes each triangle in $t \in m$ and considers voxels intersecting with t to be occupied. Thus $\mathcal{V}_e(m)$ only produces voxels at the surface of m and interior voxels are not present. The approach by Nooruddin and Turk [115], which we denote as $\mathcal{V}_f(m)$, uses parity counts and ray stabbing to obtain voxel grids of mesh models with the interior voxels filled in.

We apply $\mathcal{V}_f(m)$ when computing the Hamming distance since it requires the inputs to be of the same length. Thus, we cannot use just the outer voxels from $\mathcal{V}_e(m)$ which can vary in number from mesh to mesh. A similar argument applies to the IoU as the measure is only meaningful if the interior voxels are considered as well. The surface accuracy and completeness measures were designed to measure similarity between the surfaces of meshes [61] and thus when applied to voxel grids we only consider the exterior voxels from $\mathcal{V}_e(m)$.

We first determine the maximum potential of using the additive depth map for shape completion by comparing m_{ori} with m_{gt} and PSR_{gt} . We then compare m_{ori} with m_{pred} and PSR_{pred} to evaluate how well our method does in comparison to the ground truth. Tables 6.5 and 6.6 summarizes our results for volumetric evaluation on the SHREC’17 subset and the clothed human datasets.

Since our networks were trained on randomly rotated mesh models from SHREC’17, a subset of the full ShapeNet [10], a direct comparison to other approaches which make use of the consistently aligned ModelNet [171] or their own subset of ShapeNet does not make sense. However, since ModelNet is also a subset of Shape-Net, we can still make comparisons in order to gauge approximate relative performance. The goal is to determine how well our minimal 3D representation, the additive depth map, compares with other fully 3D CNN approaches. To that end we list results for the chair category from ShapeNet as reported in prior works. More specifically, we list the results as reported in Stutz et al. [151] for both Dai et al. [20] and Stutz et al. [151] as it is more complete and covers more measures than in their original paper. Although not directly related, we also list the results from Fan et al. [36] who predict voxel grids from single view color

images. Table 6.7 summarizes these comparisons. Figs. 6.10 and 6.11 provide a voxel visualization of our results.

Measure	Mesh	32^3	64^3	128^3	256^3
Ham ($\downarrow$)	m_{gt}	0.007	0.009	0.008	0.007
	m_{pred}	0.011	0.010	0.009	0.009
	PSR_{gt}	0.009	0.012	0.011	0.010
	PSR_{pred}	0.013	0.013	0.012	0.011
IoU ($\uparrow$)	m_{gt}	0.753	0.617	0.579	0.582
	m_{pred}	0.665	0.526	0.489	0.485
	PSR_{gt}	0.746	0.608	0.577	0.587
	PSR_{pred}	0.646	0.507	0.469	0.467
Acc ($\downarrow$)	m_{gt}	0.020	0.044	0.105	0.240
	m_{pred}	0.176	0.397	0.887	1.925
	PSR_{gt}	0.250	0.503	1.024	2.090
	PSR_{pred}	0.359	0.769	1.627	3.390
Comp ($\downarrow$)	m_{gt}	0.248	0.529	1.099	2.231
	m_{pred}	0.367	0.763	1.590	3.274
	PSR_{gt}	0.186	0.397	0.837	1.733
	PSR_{pred}	0.343	0.709	1.484	3.081

Table 6.5: Volumetric evaluation results for the SHREC’17 subset. The numbers represent the average of each category from Table 6.1. Per category results are listed in the Appendix.

6.7.5 Application to Real World Data

We demonstrate our approach for the task of clothed humans shape completion from a single depth map using real data. Five RGB-D videos of two different male subjects in regular clothing were captured from three calibrated sensors positioned around the target. A total of 5214 RGB-D frames per camera stream was recorded. Depth data from one of the cameras is then used as input to our network and a back depth map is produced for shape completion. Like in Fig. 6.4 we union the input and network prediction before performing Poisson surface reconstruction [67] to produce a water tight mesh.

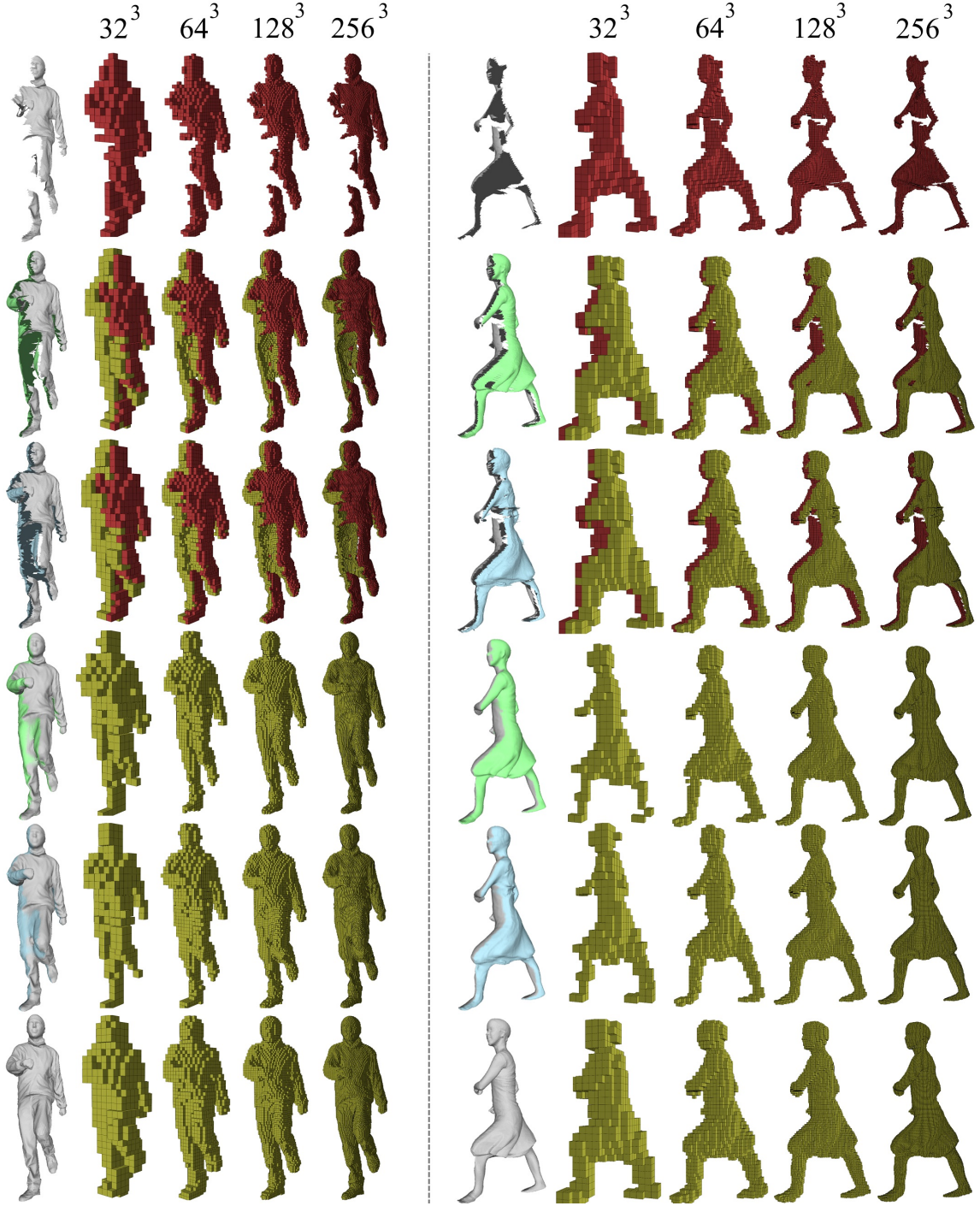


Figure 6.10: Voxelization results of a male (left) and female (right) from the clothed dataset. For both subjects, the input voxels are shown in red and the reconstruction in yellow. The top three rows visualize voxelizations of the input, ground truth and prediction. The bottom three rows visualize post processing with Poisson surface reconstruction prior to voxelization for the ground truth, prediction and the original mesh model.

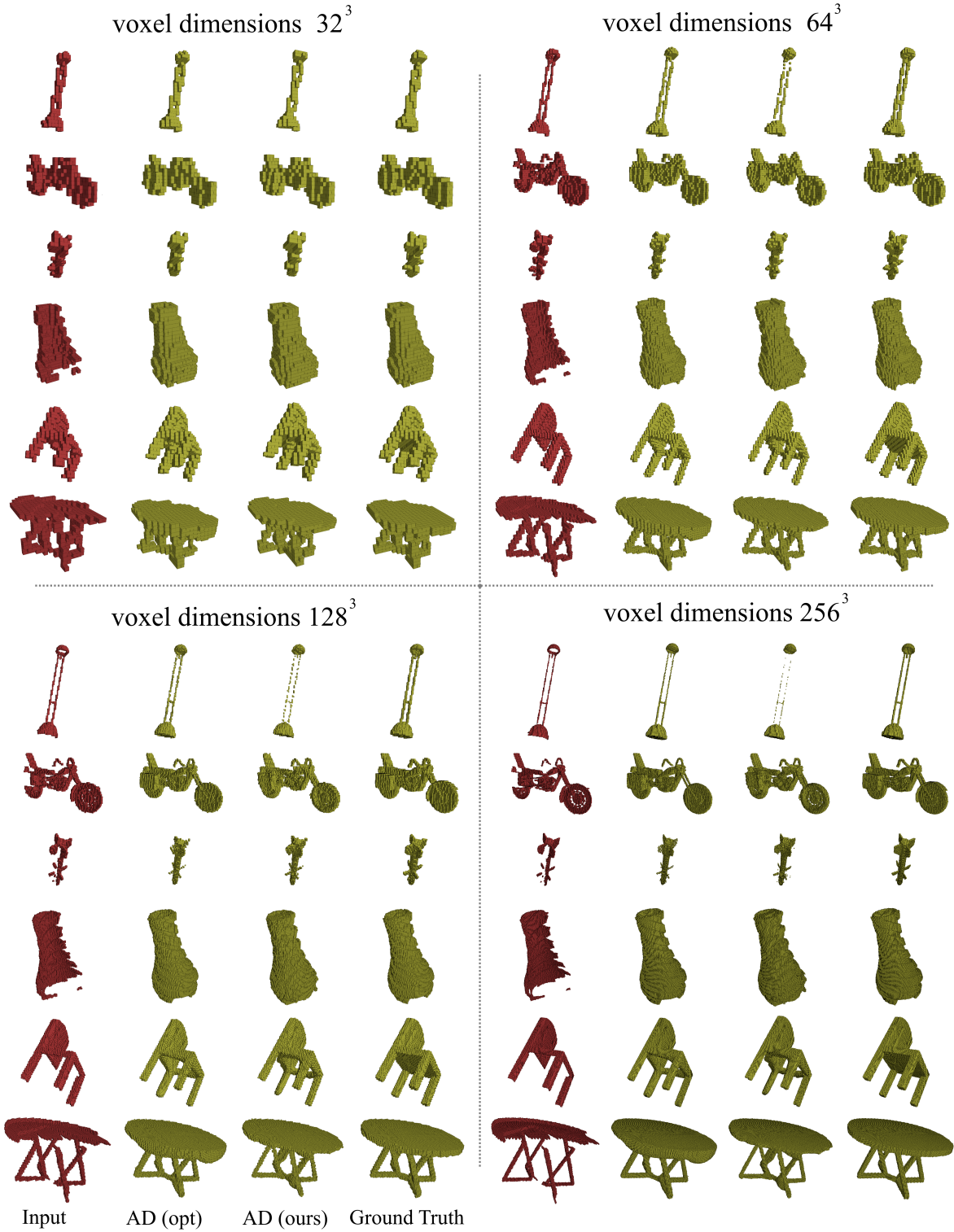


Figure 6.11: Voxelization of our shape completion results on six SHREC'17 categories. The inputs and outputs for each category are the same as those show in Fig. 6.9.

Measure	Mesh	32^3	64^3	128^3	256^3
Ham ($\downarrow$)	m_{gt}	0.003	0.002	0.002	0.002
	m_{pred}	0.004	0.003	0.002	0.002
	PSR_{gt}	0.003	0.001	0.001	0.001
	PSR_{pred}	0.003	0.002	0.002	0.002
IoU ($\uparrow$)	m_{gt}	0.772	0.841	0.868	0.867
	m_{pred}	0.750	0.810	0.828	0.827
	PSR_{gt}	0.797	0.884	0.919	0.921
	PSR_{pred}	0.769	0.839	0.861	0.861
Acc ($\downarrow$)	m_{gt}	0.021	0.047	0.102	0.213
	m_{pred}	0.060	0.127	0.283	0.640
	PSR_{gt}	0.035	0.071	0.152	0.328
	PSR_{pred}	0.065	0.145	0.331	0.748
Comp ($\downarrow$)	m_{gt}	0.083	0.188	0.411	0.869
	m_{pred}	0.111	0.240	0.530	1.161
	PSR_{gt}	0.042	0.086	0.181	0.380
	PSR_{pred}	0.090	0.184	0.399	0.874

Table 6.6: Volumetric results for the clothed human dataset.

To incorporate color, we trained an encoder-decoder style network on the color portion of the captured RGB-D videos. The inputs are then the color camera images, resized to 256×256 , and the target is the back color image obtained via re-projection of the other two cameras. Fig. 6.12 visualizes the structure of this network. For training, frames from all captured videos were placed into a 70/20/10 data split via uniform random sampling. The network was trained with ADAM [73] and the $\mathcal{L}_1$ loss for 100k iterations using a batch size of 20.

As the focus of this chapter is on 3D shape completion, we only use the color image predictions to demonstrate a proof of concept showing that color information is simple to incorporate into mesh models when working with additive depth maps. More detailed experiments exploring possibilities with combining color and the additive depth map into a single network is left as future work. Fig. 6.13 illustrates the results of combining color image predictions with the additive depth map before meshing.

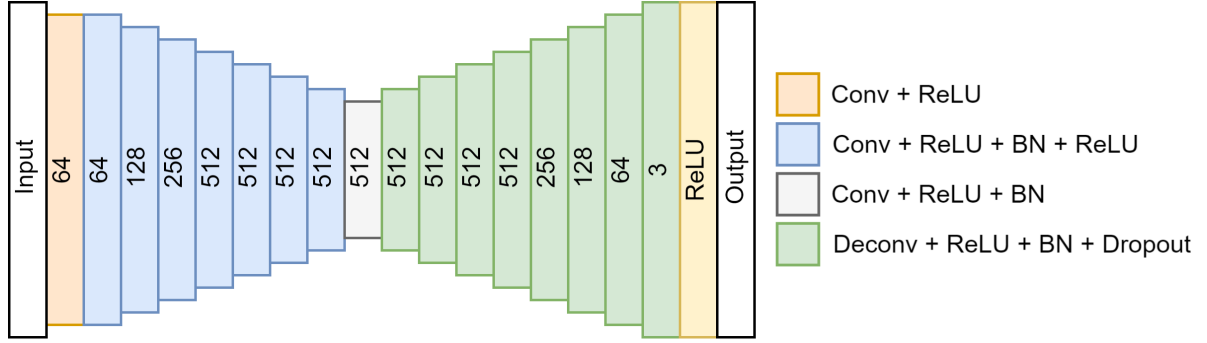


Figure 6.12: Encoder-decoder network for predicting the back color images. The numbers for each block represent the number of feature maps. Each convolution and deconvolution is applied with a kernel size (3×3). The orange, blue and green blocks also use a stride of (2×2) thus halving (orange, blue) and doubling (green) the image dimensions.

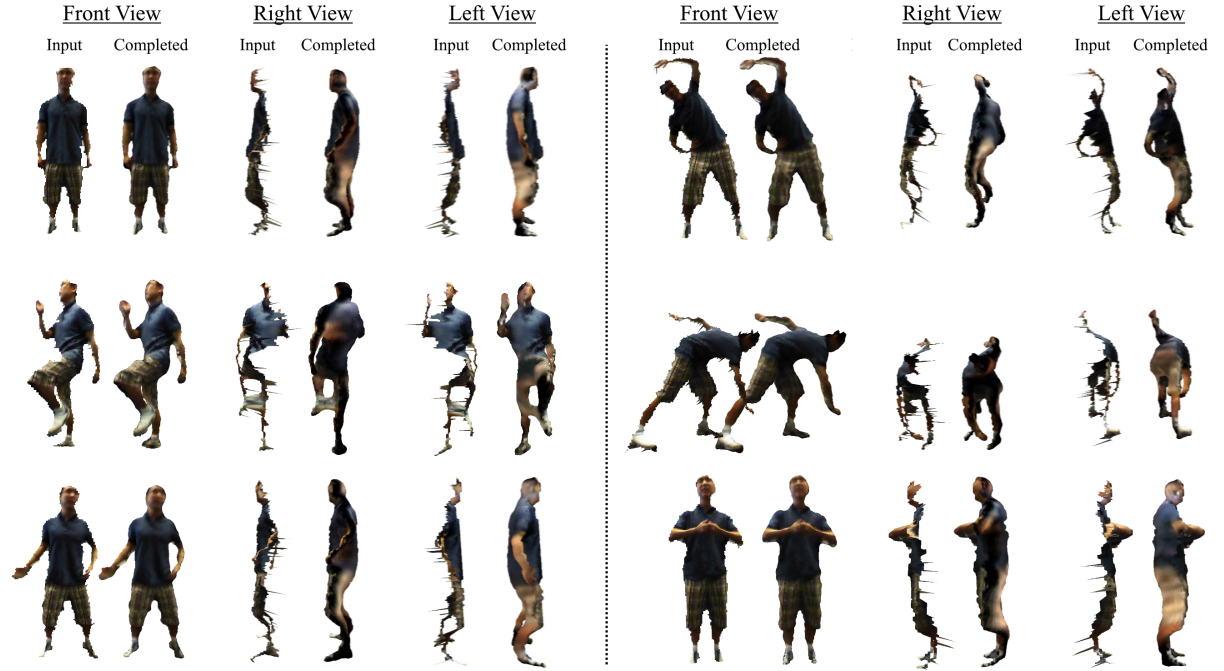


Figure 6.13: Mesh visualizations of completed human shapes from captured RGB-D video. Each row is a different frame from the video and each column presents an alternative view of the input and completed shape.

Measure	Method	32 ³	64 ³	128 ³	256 ³
Ham (↓)	AD (gt)	0.009	0.009	0.008	0.006
	AD (ours)	0.014	0.011	0.010	0.009
	Dai et al. [20]	0.019	0.016	-	-
	Stutz et al. [151]	0.033	0.026	-	-
IoU (↑)	AD (gt)	0.726	0.536	0.535	0.573
	AD (ours)	0.600	0.406	0.391	0.409
	Fan et al. [36]	0.544	-	-	-
	Choy et al. [12]	0.466	-	-	-
	Dai et al. [20]	0.610	0.548	-	-
	Stutz et al. [151]	0.414	0.333	-	-
Acc (↓)	AD (gt)	0.023	0.051	0.115	0.255
	AD (ours)	0.198	0.448	0.986	2.105
	Dai et al. [20]	0.663	0.470	-	-
	Stutz et al. [151]	1.088	0.893	-	-
Comp (↓)	AD (gt)	0.381	0.848	1.772	3.625
	AD (ours)	0.591	1.272	2.674	5.522
	Dai et al. [20]	0.671	0.530	-	-
	Stutz et al. [151]	0.785	0.852	-	-

Table 6.7: Comparison to state-of-the-art 3D shape completion on the chair category only. AD denotes our additive depth map approach while "gt" represents the optimal reconstruction using the ground truth additive depth map and "ours" is the network prediction. As many approaches only evaluated on IoU we only listed an approach if their original paper performed the evaluation. Bold numbers indicate the best performing network after excluding the results from AD (gt).

6.7.6 Learning with Semantic Labels

There exists a growing amount of research on human skeleton estimation and body part segmentation from depth maps [137, 172, 51] and more recently from just color images [105, 104]. Thus, we examined the effects of incorporating such data into our best performing clothed human model from Section 6.7.2. More specifically, we modified our network as described in Section 6.4 to accommodate an additional input image where

each pixel is a label of the background or a particular body part. Fig: 6.14 provides an illustration of the modified network.

In order to generate body part semantic images for training, we manually labeled the triangles of all 1500 mesh models from the clothed human dataset with the labels as shown in Fig. 6.15. As manually labeling a mesh model using traditional tools is a laborious task, we developed a virtual reality tool to quickly facilitate the labeling of triangles from mesh models through hand held motion controllers. Fig. 6.15 provides a visualization of the labeled meshes.

Note that the label boundaries on the mesh models were chosen to respect the clothing boundaries on the mesh. For example, cyan and red labels represent the arms of most subjects but one of them, as shown in Fig. 6.15, wears a thick shirt covering most of their arms. Thus, for those mesh models only the hands are colored cyan or red rather than the entire clothed arm. We choose this labeling for two reasons:

1. It will allow us to test the generality of our approach by having some data samples provide larger hints through showing more of the arms and some which only provide small hints with just showing the hands.
2. To introduce a new labeling of the clothed human dataset that respects the clothing outlines. To the best of our knowledge, such a labeling does not currently exist for the clothed human dataset.

We do not directly use the labels as shown in Fig. 6.15. Instead we combine the left and right arms into one label and the left and right legs into another. When generating the actual semantic images for training, we first render a "front" and "back" labels image before combining them as shown in Fig. 6.16. When two labels images are combined, it is possible for a pixel to have two different labels. These cases are resolved by assigning a "priority" value to each label. For our experiments we give the highest priority to the arms, followed by the legs, head and then the body. The intuition behind this ordering is to provide the network with a hint of where the occluded portions will be and the arms are the most common body part to be occluded by the body.

Using the semantically labeled training samples as shown in Fig. 6.16, we trained the modified network (Fig: 6.14) following the same training split and procedure as described in Section 6.6. Similar to our experiments with multi-task learning (Section 6.7.2), we evaluated the modified network using depth map error and accuracy measures as described in Section 6.7.1 and summarized the results in Table 6.8. Our experiments show

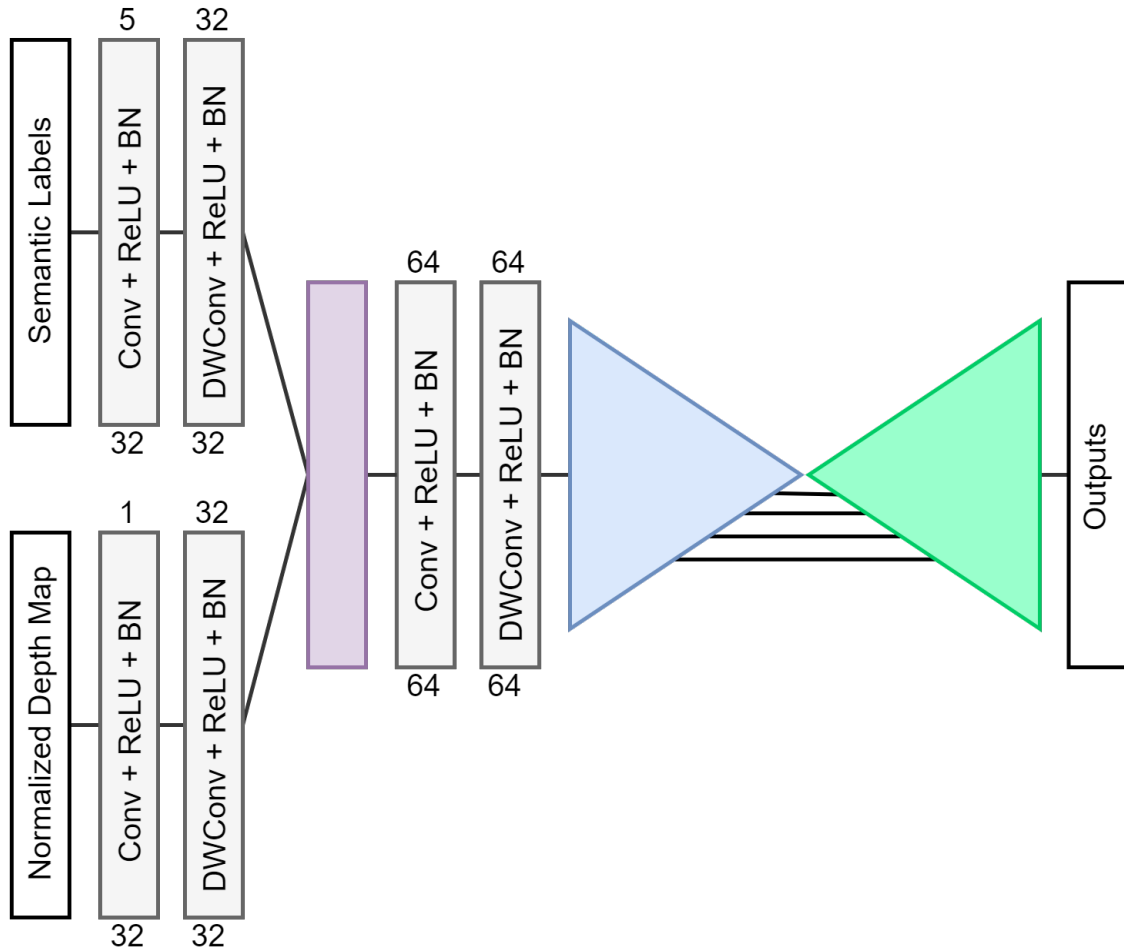


Figure 6.14: Modification of the network shown in Fig. 6.5 to allow for two inputs: the normalized depth map and the corresponding semantically labeled image. Like in Fig. 6.5, the numbers at the top/bottom of each block indicates the number of incoming/outgoing feature maps. The blue and green triangle shape connected by four black lines represents the contraction, expansion and dense portions of the original network.

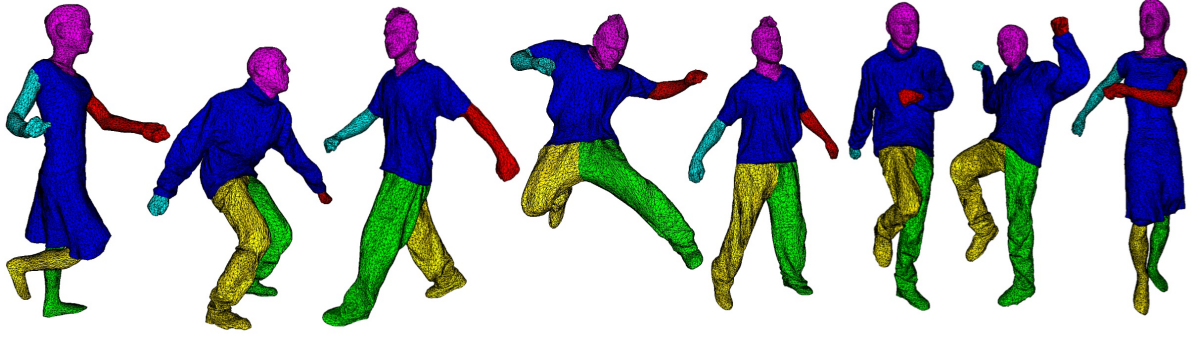


Figure 6.15: Examples of semantically labeled mesh data. The colors of each label represent the following body parts: clothed main body (blue), head (magenta), cyan (left arm), red (right arm), left leg (yellow) and right leg (green).

that including a semantically labeled image as input further improves performance across all evaluation measures. A visual comparison between the predictions and reconstructions from a model trained with and without semantic labels as input is shown in Fig. 6.17.

Network	Error (lower is better)			Accuracy (higher is better)				
	rel	rmse	log10	$\delta < 1.01$	$\delta < 1.02$	$\delta < 1.03$	$\delta < 1.05$	$\delta < 1.10$
N_d	5.47×10^{-3}	3.22×10^{-2}	2.39×10^{-3}	0.886	0.961	0.977	0.989	0.997
N_{d+s}	4.23×10^{-3}	2.49×10^{-2}	1.84×10^{-3}	0.923	0.974	0.986	0.994	0.999

Table 6.8: Comparison of depth map evaluation results between a network which only takes in a depth map as input (N_d) and one which takes in both the depth map and a semantics label image (N_{d+s}).

6.8 Discussion

From Figs. 6.8 to 6.11 we can observe some reconstruction characteristics of our approach. First, thin structures can be reconstructed because our network outputs depth maps of sufficient resolution. For example, our approach can reconstruct the thin supports for the lamp as well as the motorbike handle bars and spokes from Fig. 6.9. Additionally, the front-back nature of our reconstruction allows for stronger estimation of normal vectors directly from the depth map image structure. Stronger normals allows for easier post processing reconstructions. Second, smooth surfaces tend to induce a quantization effect on the reconstructed surface. An example of this is the vase from Fig. 6.9. This quantization effect is likely influenced by the depth labels image as it partitions

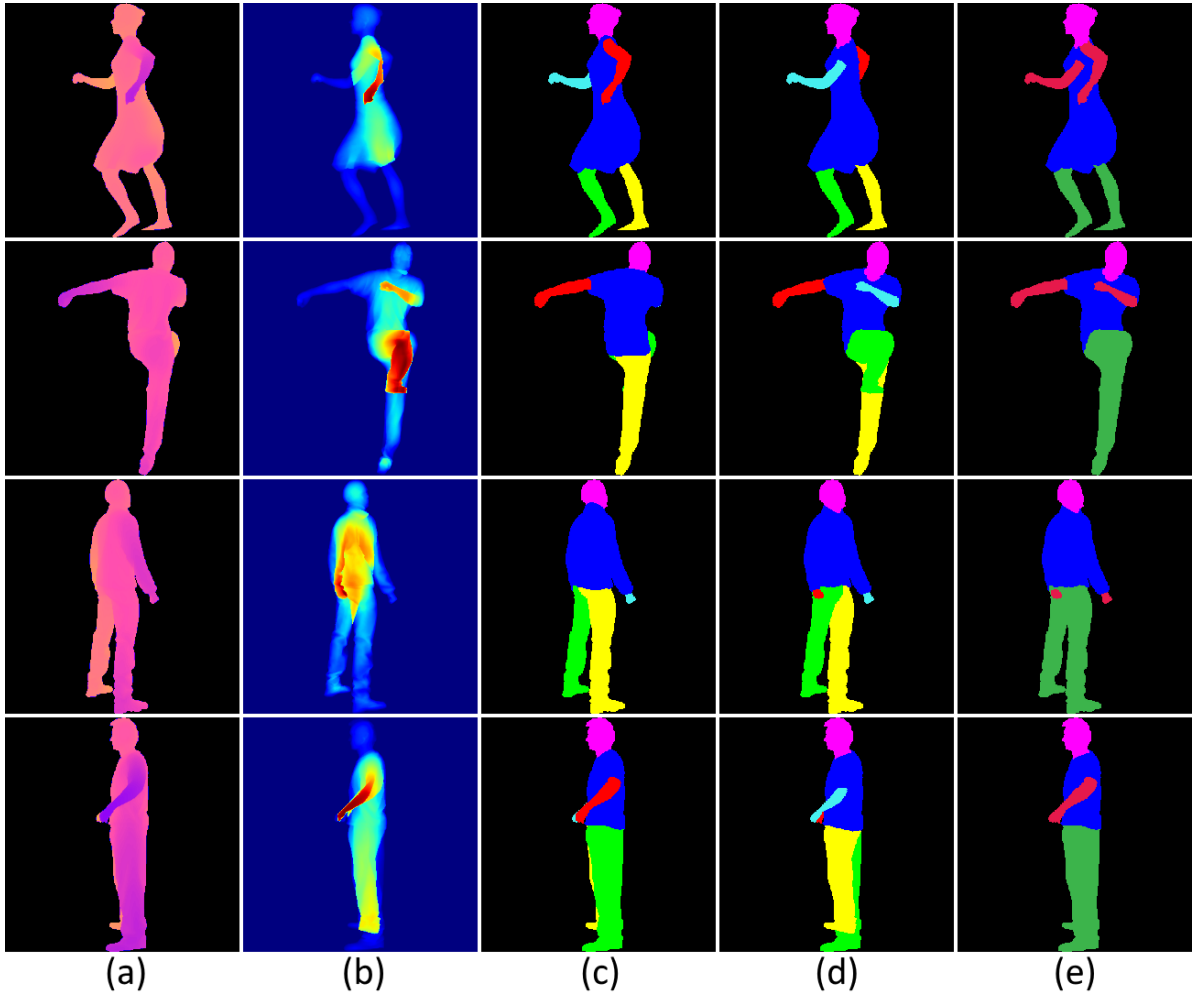


Figure 6.16: Examples of semantic images generated from the manually labeled clothed human mesh models. (a) Input depth map, (b) ground truth additive depth map, (c) labeling from mesh data of the front view, (d) labeling from mesh data of the back view, (e) combined labeling of (c) and (d) with left and right arms/legs merged into two distinct pixel groups. Note that we place the limb labels on top of the body labels as they tend to cross the body often.

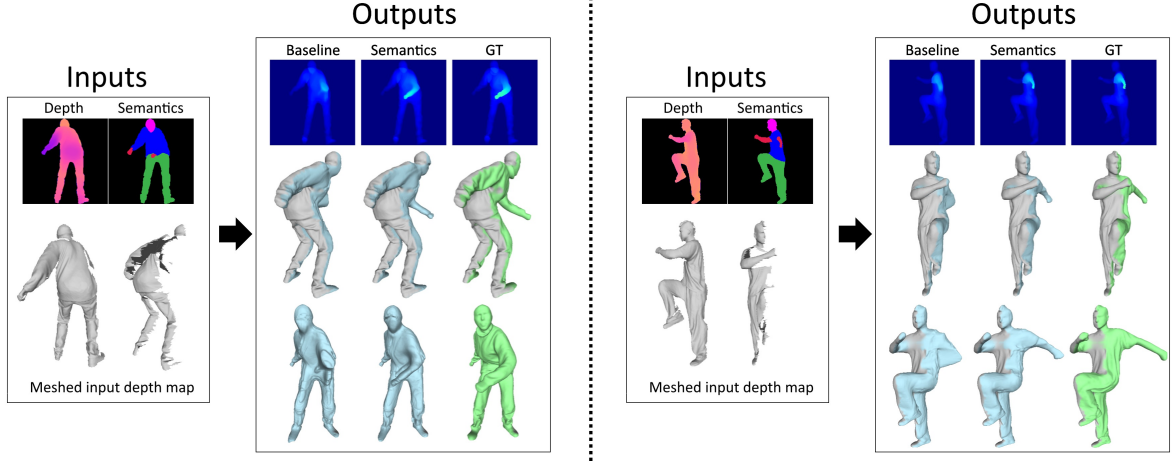


Figure 6.17: A visual comparison between the outputs and reconstructions from our best performing model (denoted as "Baseline") trained with just depth as the input and another model (denoted as "Semantics") trained with both depth and semantic labels as input. The input depth map from the two examples has one arm completely occluded which makes it difficult to perform reconstruction. Providing the network with a hint through the semantic labels allows it to reconstruct hidden body parts more accurately and completely.

the target additive depth map into L bins. Third, while our network can predict missing structures like a chair leg (Figs. 6.9 and 6.11), it can sometimes fail to fill in internal volumes such as the chair seat. This failure depends on the viewpoint as a front view with no observed depth values for the seat (i.e: the chair from Fig. 6.7) will cause the effects seen for chairs in Figs. 6.9 and 6.11. When depth values are observed for a portion of the table top, the results are much better and reconstruction of the flat table top is possible.

The third property from above helps explain the volumetric results seen in Table 6.7. In particular, our chair IoU for a voxel resolution of 32^3 is fairly competitive with other approaches but drops when moving to a higher resolutions of 64^3 . From there it stabilizes and does not change significantly when moving to even higher resolutions of 128^3 and 256^3 . This effect can be seen visually in Fig. 6.11 where moving to from 32^2 to 64^3 opens a hole where the seat should be. The remaining reconstruction does not change too much when moving to even higher resolutions and thus the IoU also remains stable.

Results for the clothed human dataset are much stronger when compared to the SHREC'17 dataset. From Table 6.6 we can see that the performance for both the Hamming distance and volumetric IoU increases as the resolution grows. This is because

a person is solid without internal structures and thus increasing the resolution also increases the number of internal voxels by a cubic amount which in turn increases the IoU. This result also shows that approximating the overall shape of humans using a front and back depth map is sufficient in most cases. A qualitative example is demonstrated in Section 6.7.5 where we apply our network to real life data without fine-tuning.

From Fig 6.17 we can visually see the effects of incorporating semantically labeled images into the network and training pipeline for the clothed human dataset. By including the semantic information about the body parts of a person, the network is more capable of reconstructing occluded limbs. In particular, Fig. 6.17 shows two examples in detail where one of the persons arms is completely occluded by their body. When the network is given just the input depth map, it can only make a guess as to where the occluded arm should be. This results in an "averaging" effect on the region of occlusion producing a sub-optimal reconstruction where the occluded arm is either a short stub or is partially blended with the main body. When the semantic labels are provided, the arms are better formed which leads to an overall improvement in reconstruction quality. Additionally, Fig. 6.17 also shows that our approach can reconstruct occluded limbs even when there is little semantic information available (i.e: just the hands and not the full arm). This result shows that the network has learned the relationship between the input semantic labels and the effects it should have on the output additive depth map.

Overall, our approach achieves competitive volumetric results when compared to the state-of-the-art in 3D shape completion at lower resolutions like 32^3 and 64^3 . This makes sense since the maximum amount of information contained within a 256×256 depth map is about the same as a 40^3 volume. Finally, it also interesting to note that the optimal performance of the additive depth map (i.e: using the ground truth back depth map for reconstruction) beats state-of-the-art approaches in most measures at voxel resolutions of 32^3 and 64^3 .

6.9 Summary

In this chapter, we presented the additive depth map as a minimal alternative representation of volume for the task of 3D shape completion from a single view depth map. We investigated the effects of multi-task training and found that jointly learning the additive depth map and a depth labels image produces the best results. Our experimental evaluation shows that the additive depth map is capable of state-of-the-art 3D shape

completion and we demonstrated an application to real world data without fine-tuning.

Since the additive depth map is an image, traditional 2D CNNs can be used and the result directly applied to the input depth map without first voxelizing the input. Thus, the completed shape is relative to the input data and requires no additional re-alignment. Another key advantage of using images is the simplicity in including color information in the completed 3D shapes. Our experiments with real world data demonstrated this advantage. In order to fuse color image data into completed voxel grids, it must first be meshed and then UV unwrapped.

We also showed that semantically labeled images are simple to include in the training pipeline by making a small modification to the network. Our experiments on the clothed human dataset showed that incorporating semantic information allows for stronger predictions - especially for heavily occluded inputs.

Our approach is not without limitations. Our experimental results show that objects with many or very large self-occlusions prove difficult to reconstruct completely with just a front and back depth map. This is an inherent flaw with using just one depth map for shape completion and it is the trade-off for using a minimal representation of volume.

In summary, the additive depth map is a data minimal representation for 3D shape completion which can recover 3D shape with fine details. Future work could be to include the color images as part of the input into a network for jointly predicting the 3D shape and color of the additive depth map. Another interesting direction could be to use the additive depth map as a way to fuse finer details to a lower resolution voxel grid.

Chapter 7

Conclusions

The work presented in this thesis describes a general framework, and examined the necessary components, for producing immersive virtual reality (VR) videos using a single RGB-D sensor. First, we identified four properties of immersive VR videos and then described the basic building blocks required in order to realize a framework which would satisfy those properties. Next, we explored approaches for solving the first building block, the separation of the foreground and background of video frames, in the context of video summarization. For the next building block we examined how static background models could be built in order to provide a viewer with a consistent and surrounding 3D background. An application for 3D change detection was also explored as that would provide a basis for how we might separate the foreground and background of RGB-D videos. Using the lessons learned from the applications of the first two building blocks, we then devised a system for producing VR videos which satisfied the first three properties of

immersive VR videos. To tackle the last property of immersive VR videos, we employed a deep learning based solution in order to complete the 3D shape of single view depth maps.

7.1 Summary

Our framework for immersive VR videos using single RGB-D sensors is composed of several building blocks, each of which solves a particular problem before feeding the output to the next building block. This framework addresses the limitations of current VR videos which restrict a viewer's range of motion. Additionally, current VR videos are also limited in terms of interactivity since they are simply video frames. A large part of immersion comes from the range of sensory modalities accommodated and thus the ideal approach would produce video frames which are full 3D objects. We designed our framework around the fact that multi-sensor solutions would most likely be priced out of range of average consumers as well as being too complex to setup in practice, especially for mobile environments. We then examined a method which could be used for each of the building blocks for this framework.

In **Chapter 1** we first provided an introduction to VR through the hardware which is now available at the consumer level. We then covered the properties and anatomy of VR head-mounted displays (HMD) which would serve as the medium from which the viewer perceives an immersive VR video. Next, a review of existing sensors, capture methods and their possible output formats with respect to how they might be used to produce VR videos is explored. We found that the existing approaches relied on panoramic or wide-angle video formats in order to surround a viewer with as much data as possible. The usage of standard video resulted in a limited range of motion for a viewer which in turn limits the amount of immersion they may experience. Using RGB-D video could produce videos where each frame was a full 3D object but there was still the question of how to handle the case of a single moving RGB-D sensor. Additionally once a viewer moved, occlusions would exist in both the dynamic foreground objects and the static background. These occlusions also limits the amount of immersion a viewer may experience. These limitations served as the basis for formulating the main research objectives for this thesis.

In **Chapter 2** we defined the technical components which would make a VR video immersive by adapting an existing framework which tackled how to describe the level of immersion when experiencing virtual environments. Since this framework was originally

designed for HMDs, we used it as a basis and derived our own definitions for the case of immersive VR video. Based on these definitions we then identified four properties which must hold in order for a VR video to be considered immersive. Failure to adhere to these properties would result in a reduced level of immersion. Then, a system for producing immersive VR videos from single RGB-D sensors which would satisfy the identified properties is outlined. Each of the main building blocks of this system was covered in general along with potential methods which could be used to realize each block.

In **Chapter 3** we first provided a brief literature review of existing background subtraction methods and then demonstrated an application of foreground and background separation for video summarization of surveillance cameras. The separation of the foreground from the background is one of the building blocks of the system described in Chapter 2. The video summarization application is one of the contributions of this thesis and the foreground/background separation approach used for video summarization was later applied in conjunction with another method from Chapter 4 to produce some initial results for Chapter 5. For our video summarization, we introduced a novel approach which aimed at retaining as much of the activity pixels (i.e: the pixels which would be important to a viewer) as possible in order to provide a coherent video summary. Additionally, we proposed new evaluation criterion to quantify the quality of a video summary in terms of the amount of activity retained in relation to summarized video length.

In **Chapter 4** we gave an overview of visual SLAM with a focus on methods which made use RGB-D sensors. The application of RGB-D SLAM was applied to reconstruct static 3D models which is one of the building blocks of the system described in Chapter 2. Next, an application of RGB-D SLAM involving the detection of changes from RGB-D videos captured from patrolling robots was outlined. These detected changes are then back-projected onto the input video so that an easy to understand video summary could be rendered. Displaying the detected changes as a video summary can be seen as superior to just viewing 3D data for three reasons: (1) the results can be viewed on nearly all devices from hand-held to desktops, (2) existing video compression algorithms can be applied to allow for easy remote viewing of the results and (3) the context of the detected changes can be understood through the sequence of video frames. The methods used in the change detection application were later combined with the approaches from Chapter 3 to produce some initial results for Chapter 5.

In **Chapter 5** we first briefly reviewed related works in VR scene creation that made use of the sensor capture and output formats as described in Chapter 1 before presenting an initial approach that made use of the concepts used in Chapters 3 and 4.

The review demonstrated that there existed approaches which could produce immersive VR videos which could satisfy the properties as outlined in Chapter 2 but required multiple sensors. While compelling, multi-view approaches utilized complex camera setups and thus are not practical in a mobile environment. In contrast, the single sensor approaches for VR scene reconstruction have either focused on building dynamic objects through precise tracking of the target while ignoring the background (a key property of immersive VR videos as defined in Chapter 2), or focused on producing static models without dynamic objects.

An initial approach was devised using the concepts from Chapter 3 and 4. The result were VR videos which technically addressed the first three properties of immersive VR videos from Chapter 2 but did not feel very dynamic due to the fact that the interesting foreground objects must enter and leave the field of view of a fixed camera pose. A freely moving camera would allow for more interesting and dynamic scenes as the actors may move around the environment rather than be restricted to a specific region. A contribution of this thesis, which addressed the case of a moving camera, is then presented which could produce VR scenes that satisfied the first three properties of immersive VR videos and allow for a moving camera. The approach relied on a combination of human pose recognition and visual object tracking to segment humans from the background of input RGB-D videos. The background only frames could then be fed into a RGB-D SLAM system to produce a static background and the dynamic objects could be superimposed back into the scene. We evaluated the approach based on the performance of the foreground/background separator as poor segmentation would result in the dynamic foreground objects being fused into the background which was undesirable for VR videos.

In **Chapter 6** we presented a method for performing 3D shape completion of single view depth maps in order to address the forth property of immersive VR videos from Chapter 2. As the approach utilized convolutional neural networks (CNNs), an overview of artificial neural networks in general was covered before focusing on CNNs. Next, a survey of related works on 3D shape completion was provided and we found that most existing works either used a known database in order to match and retrieve 3D models from the input partial 3D shapes, or they utilized 3D CNNs to learn the shape of 3D models in the form of voxel grids. A large drawback of utilizing 3D CNNs is

the cubic increase in memory requirements which limited the output models to coarse resolutions. A contribution of this thesis is then presented which addressed this limitation by examining 3D shape completion from the perspective 2D CNNs and depth maps. This contribution would complete the last building block of the system described in Chapter 2 and combined with the approach from Chapter 5 would produce immersive VR videos in which the dynamic foreground objects were as complete as possible.

In order to perform 3D shape completion using depth maps we introduced the concept of an *additive depth map*. The additive depth map represents the amount of depth needed to transform the input into the "back" depth map taken with a sensor exactly opposite of the input camera. A union of the input and back depth map is then the completed 3D shape. We generated a dataset of input depth maps, and the corresponding additive depth maps, by rendering the mesh models from SHREC'17 and clothed humans datasets. A residual encoder-decoder CNN was then devised to predict the additive depth map. A thorough evaluation was performed using depth map and volumetric error metrics and we found that our approach was comparable to the state-of-the-art 3D CNN approaches at low voxel resolutions (32^3 and 64^3) while also being capable of producing high resolution outputs which could capture finer details. We then demonstrated our approach by applying it to real world data captured with commodity depth sensors without fine-tuning the network. We further show that semantic information about the target objects, specifically humans, could be incorporated into the training pipeline to produce even stronger results. To incorporate semantic information, we produced a new dataset by labeling all 1500 of the clothed human mesh models so that semantically labeled images could be rendered.

7.2 Review of Research Questions

In Chapter 1 we presented five research questions in relation to this thesis. We now provide some partial answers based on the work presented in this thesis.

(1) What properties does an immersive VR video have and how can it be displayed?

In Chapter 2 we identified and provided a list of four properties which should hold for a VR video to be considered immersive. These properties were derived by following the spirit of an existing framework by Slater and Wilbur [144] which was originally designed

to describe immersive VR displays in terms of their technological components. Thus our identified properties described the technological components which should exist in immersive VR videos. These properties then served as a basis for the design of an overall system which displayed captured videos as a combination of static and dynamic fully 3D structures.

Our identified properties focused mainly on the visual components and did not expand on the other five senses. To be completely immersive, all five senses should be engaged. This is the main limitation of our current properties for immersive VR videos. While existing technological limitations do not allow for strong capture and reproduction of smell and taste, especially for consumer hardware, there does exist technology for allowing touch and hearing. In particular, current consumer VR HMDs are often also sold with motion controllers allowing users a degree of interaction with the 3D environments that they are viewing. Such interactions would need rules and properties defined for the case of immersive VR videos.

(2) How can we remove the background and capture all moving foreground objects from a single RGB-D video stream?

Two approaches for background removal and capturing moving foreground objects were investigated throughout this thesis. We began with the simplest case in Chapter 3 - the static camera and using only RGB video. When these approaches were applied to VR scene creation in Chapter 5, we found that the results lacked dynamism as viewers were forced into a frontal view only. This lead us to examine the case of a moving RGB-D sensor to capture a larger portion of the environment relative to the sensors field of view and the rest of Chapter 5 details this approach.

In both cases, the static RGB and the moving RGB-D camera, we have presented one potential approach and measured its performance against a manually labeled ground truth in terms of the portion of foreground object pixels that we have accurately captured. For the static RGB camera we used a combination of background subtraction, dense optical flow and pixel clustering to produce foreground object trajectories. A limitation of this approach is that we often also capture a large portion of the surrounding background pixels and can sometimes capture data which is not of interest (i.e: illumination changes). In the case of the moving RGB-D camera we relied on a human pose estimator and visual object tracking for foreground object tracking. A limitation of this approach is the fact that this method only works well with people. If we wanted to capture other

interesting objects, such as animals, we could only rely on the visual object tracker which requires some user assistance. Additionally, using flood fill on the depth map for object segmentation can sometimes include the background when the target is too close to it. The limitations for both approaches presented in this thesis could potentially be alleviated by augmenting them with a deep learning based approach for object detection and segmentation.

(3) How do we model and display the static portions of the input RGB-D video?

We can model the static portions of the input RGB-D video by first removing all of the dynamic elements and feeding the background only frames into existing RGB-D SLAM systems. The result is both a static 3D background and a set of camera trajectories. The background can be displayed as a point cloud if we just accumulate all of the projected RGB-D video frames or we can accumulate a truncated signed distance field (TSDF) and use marching cubes to extract a mesh from the volume. The former is ideal for resource limited systems while the latter can be used provided the system has enough memory to store the TSDF.

(4) How can we merge the results of (2) and (3) coherently into one dynamic scene?

Chapter 5 answers this question and describes a specific implementation. In summary, we simply re-use the camera poses which were computed during the static background building phase to transform the extracted foreground frames into the frame of reference of the static background model. This allows for both the dynamic foreground objects and the static background model to be displayed coherently together.

(5) Is it possible to "fill in" the occluded portions of the moving foreground objects?

Chapter 6 details a deep learning based approach for shape completion which allows one to fill in the occluded portions of objects within single view depth maps. The proposed method in this thesis assumes that the moving foreground objects have already been extracted and properly segmented - that is no background data is attached to the

foreground mask. As we explored an approach which is data minimal in order to use traditional 2D convolutional neural networks (CNNs) to train on higher resolution depth maps, objects which have large self occlusions proved difficult to completely reconstruct. A 3D CNN based approach could be used to perform shape completion at the cost of a lower resolution and coarser output.

7.3 Future Work

For future work, it would be interesting to apply the general framework proposed in this thesis to other sensors. For example, there exist deep learning based solutions for converting color images to depth maps. These approaches could replace the RGB-D sensors and thus allow for regular cameras to apply our framework. Additionally, producing depth maps from panoramic sensors, such as those used to produce 360° video could also be explored as they would be able to capture a larger range of the environment which in turn would allow for more complete VR videos. The RGB-D sensor used in this thesis has a limited FoV which means that a large amount of movement is required in order to capture enough of the background. Adapting the individual building blocks of the framework to operate on panoramic depth maps could be an interesting direction for future work as well.

Our implementation of the framework produces a static 3D background model and it also made use of human pose estimation but we have not explored direct manipulation of the 3D background model nor the estimated skeletons. Approaches for providing realistic or interesting feedback when the viewer interacts with the reconstructed humans or environment could be explored as well. This would allow for a more encompassing accommodation of the "touch" sensory modality which would further improve immersion. Furthermore, semantic segmentation of the input RGB-D video could extract objects, such as chairs, from the background model and create interactive elements. Precisely how this can be accomplished as well as how to establish the rules of interaction could be a potential direction for future work.

Finally, our framework has not accounted for lighting of the scene and objects. That is, the light is baked into the VR video when capturing the scene and dynamic objects. An in depth look at delighting and relighting the input RGB-D video could provide more realistic results as well as the possibility for custom lighting in the output VR video (ie: increase/decrease darkness, use different colored lights).

References

- [1] Henrik Andreasson, Martin Magnusson, and Achim Lilienthal. “Has something changed here? autonomous difference detection for security patrol robots”. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2007, pp. 3429–3435 (cit. on pp. 50, 51).
- [2] Henrik Andreasson, Rudolph Triebel, and Achim Lilienthal. “Vision-based interpolation of 3D laser scans”. In: *Proceedings of the 2006 IEEE International Conference on Autonomous Robots and Agents (ICARA 2006)*. Citeseer. 2006, pp. 455–460 (cit. on p. 50).
- [3] Nicolas Aspert, Diego Santa Cruz, and Touradj Ebrahimi. “MESH: measuring errors between surfaces using the Hausdorff distance.” In: *ICME (1)*. 2002, pp. 705–708 (cit. on pp. 53, 63).
- [4] Tim Bailey and Hugh Durrant-Whyte. “Simultaneous localization and mapping (SLAM): Part II”. In: *IEEE Robotics & Automation Magazine* 13.3 (2006), pp. 108–117 (cit. on p. 46).
- [5] Sean Bell et al. “Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2874–2883 (cit. on p. 81).
- [6] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006 (cit. on p. 71).
- [7] Randolph Blake and Hugh Wilson. “Binocular vision”. In: *Vision Research* 51.7 (2011). Vision Research 50th Anniversary Issue: Part 1, pp. 754 –770. ISSN: 0042-6989. DOI: <http://dx.doi.org/10.1016/j.visres.2010.10.009>. URL: <http://www.sciencedirect.com/science/article/pii/S0042698910005006> (cit. on p. 3).
- [8] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Institute, Amsterdam: Blender Foundation, 2018. URL: <http://www.blender.org> (cit. on p. 1).

- [9] Tom Botterill, Steven Mills, and Richard Green. “Bag-of-words-driven, single-camera simultaneous localization and mapping”. In: *Journal of Field Robotics* 28.2 (2011), pp. 204–226 (cit. on p. 22).
- [10] Angel X Chang et al. “Shapenet: An information-rich 3d model repository”. In: *arXiv preprint arXiv:1512.03012* (2015) (cit. on pp. 69, 84, 93).
- [11] François Chollet. “Xception: Deep Learning with Depthwise Separable Convolutions”. In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE. 2017, pp. 1800–1807 (cit. on pp. 74, 82).
- [12] Christopher B Choy et al. “3d-r2n2: A unified approach for single and multi-view 3d object reconstruction”. In: *European conference on computer vision*. Springer. 2016, pp. 628–644 (cit. on p. 99).
- [13] Alvaro Collet et al. “High-quality Streamable Free-viewpoint Video”. In: *ACM Trans. Graph.* 34.4 (July 2015), 69:1–69:13. ISSN: 0730-0301. DOI: [10.1145/2766945](https://doi.org/10.1145/2766945). URL: <http://doi.acm.org/10.1145/2766945> (cit. on p. 10).
- [14] Vincent Couture, Michael S Langer, and Sebastien Roy. “Panoramic stereo video textures”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1251–1258 (cit. on p. 9).
- [15] Marco Cristani et al. “Background subtraction for automated multisensor surveillance: a comprehensive review”. In: *EURASIP Journal on Advances in signal Processing* 2010 (2010), p. 43 (cit. on pp. 27, 28).
- [16] Mark Cummins and Paul Newman. “FAB-MAP: Appearance-based Place Recognition and Mapping Using a Learned Visual Vocabulary Model”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML’10. Haifa, Israel: Omnipress, 2010, pp. 3–10. ISBN: 978-1-60558-907-7. URL: <http://dl.acm.org/citation.cfm?id=3104322.3104324> (cit. on p. 48).
- [17] Mark Cummins and Paul Newman. “FAB-MAP: Probabilistic localization and mapping in the space of appearance”. In: *The International Journal of Robotics Research* 27.6 (2008), pp. 647–665 (cit. on p. 22).
- [18] Brian Curless and Marc Levoy. “A volumetric method for building complex models from range images”. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 303–312 (cit. on pp. xv, 47, 48).

- [19] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314 (cit. on p. 70).
- [20] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. “Shape Completion Using 3D-Encoder-Predictor CNNs and Shape Synthesis”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 6545–6554 (cit. on pp. 69, 77, 78, 88, 93, 99).
- [21] Dipankar Dasgupta. “A new gravitational clustering algorithm”. In: *Proceedings of the Third SIAM International Conference on Data Mining*. Vol. 112. SIAM. 2003, p. 83 (cit. on p. 31).
- [22] Marc Decombas, Frederic Dufaux, and B. Pesquet-Popescu. “Spatio-temporal grouping with constraint for seam carving in video summary application”. In: *Digital Signal Processing (DSP), 2013 18th International Conference on*. IEEE. 2013, pp. 1–8 (cit. on p. 30).
- [23] Johanna Delanoy et al. “What you sketch is what you get: 3D sketching using multi-view deep volumetric prediction”. In: *arXiv preprint arXiv:1707.08390* (2017) (cit. on pp. 77, 78).
- [24] Michael B. Dillencourt, Hanan Samet, and Markku Tamminen. “A general approach to connected-component labeling for arbitrary image representations”. In: *Journal of the ACM (JACM)* 39.2 (1992), pp. 253–280 (cit. on p. 5).
- [25] Michael Donoser and Dieter Schmalstieg. “Discriminative feature-to-point matching in image-based localization”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 516–523 (cit. on p. 22).
- [26] Mingsong Dou and Henry Fuchs. “Temporally enhanced 3D capture of room-sized dynamic scenes with commodity depth cameras”. In: *2014 IEEE Virtual Reality (VR)*. IEEE. 2014, pp. 39–44 (cit. on pp. 59, 63).
- [27] Mingsong Dou et al. “Exploring high-level plane primitives for indoor 3D reconstruction with a hand-held RGB-D camera”. In: *Asian Conference on Computer Vision*. Springer. 2012, pp. 94–108 (cit. on p. 59).
- [28] Mingsong Dou et al. “Fusion4D: Real-time Performance Capture of Challenging Scenes”. In: *ACM Trans. Graph.* 35.4 (July 2016), 114:1–114:13. ISSN: 0730-0301. DOI: [10.1145/2897824.2925969](https://doi.org/10.1145/2897824.2925969). URL: <http://doi.acm.org/10.1145/2897824.2925969> (cit. on p. 59).

- [29] Vincent Dumoulin and Francesco Visin. “A guide to convolution arithmetic for deep learning”. In: *arXiv preprint arXiv:1603.07285* (2016) (cit. on p. 74).
- [30] Hugh Durrant-Whyte and Tim Bailey. “Simultaneous localization and mapping: part I”. In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110 (cit. on p. 46).
- [31] David Eigen and Rob Fergus. “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2650–2658 (cit. on pp. 69, 88, 89).
- [32] Felix Endres et al. “3-D mapping with an RGB-D camera”. In: *IEEE Transactions on Robotics* 30.1 (2014), pp. 177–187 (cit. on pp. 47, 48).
- [33] Felix Endres et al. “An evaluation of the RGB-D SLAM system”. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 1691–1696 (cit. on p. 64).
- [34] Jakob Engel, Thomas Schöps, and Daniel Cremers. “LSD-SLAM: Large-scale direct monocular SLAM”. In: *Computer Vision–ECCV 2014*. Springer, 2014, pp. 834–849 (cit. on p. 20).
- [35] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: *The 2nd International Conference on Knowledge Discovery in Databases and Data Mining*. 1996, pp. 226–231 (cit. on p. 53).
- [36] Haoqiang Fan, Hao Su, and Leonidas Guibas. “A Point Set Generation Network for 3D Object Reconstruction from a Single Image”. In: July 2017, pp. 2463–2471. DOI: [10.1109/CVPR.2017.264](https://doi.org/10.1109/CVPR.2017.264) (cit. on pp. 78, 79, 93, 99).
- [37] Gunnar Farneback. “Two-frame motion estimation based on polynomial expansion”. In: *Image Analysis*. Springer, 2003, pp. 363–370 (cit. on p. 31).
- [38] Julien Favre et al. “Quaternion-based fusion of gyroscopes and accelerometers to improve 3D angle measurement”. In: *Electron. Lett* 42.11 (2006), pp. 612–614 (cit. on p. 5).
- [39] Ross Finman et al. “Toward lifelong object segmentation from change detection in dense RGB-D maps”. In: *Mobile Robots (ECMR), 2013 European Conference on*. IEEE. 2013, pp. 178–185 (cit. on pp. 50, 51).

- [40] Michael S Floater and Kai Hormann. “Surface parameterization: a tutorial and survey”. In: *Advances in multiresolution for geometric modelling*. Springer, 2005, pp. 157–186 (cit. on p. 79).
- [41] Jonathan Foote and Don Kimber. “Flycam: Practical panoramic video and automatic camera control”. In: *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*. Vol. 3. IEEE. 2000, pp. 1419–1422 (cit. on p. 8).
- [42] W. Förstner and B.P. Wrobel. *Photogrammetric Computer Vision: Geometry, Orientation and Reconstruction*. Geometry and Computing. Springer International Publishing, 2016. ISBN: 9783319115498. URL: <https://books.google.ca/books?id=odbJoQEACAAJ> (cit. on p. 20).
- [43] Nir Friedman and Stuart Russell. “Image segmentation in video sequences: A probabilistic approach”. In: *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 1997, pp. 175–181 (cit. on p. 28).
- [44] Vineet Gandhi, Jan Čech, and Radu Horaud. “High-resolution depth maps based on TOF-stereo fusion”. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 4742–4749 (cit. on p. 10).
- [45] Charles D Ghilani and Paul R Wolf. *Elementary surveying*. Pearson Higher Ed, 2014 (cit. on p. 8).
- [46] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 315–323 (cit. on p. 76).
- [47] Arren Glover et al. “OpenFABMAP: An open source toolbox for appearance-based loop closure detection”. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 4730–4735 (cit. on p. 48).
- [48] Clément Godard, Oisín Mac Aodha, and Gabriel J Brostow. “Unsupervised Monocular Depth Estimation with Left-Right Consistency”. In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE. 2017, pp. 6602–6611 (cit. on p. 88).
- [49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. 2016 (cit. on p. 71).

- [50] Steven J Gortler et al. “The lumigraph”. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 43–54 (cit. on p. 9).
- [51] Albert Haque et al. “Towards viewpoint invariant 3d human pose estimation”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 160–177 (cit. on p. 99).
- [52] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003 (cit. on pp. 7, 9).
- [53] Kaiming He and Jian Sun. “Convolutional neural networks at constrained time cost”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 5353–5360 (cit. on p. 74).
- [54] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on pp. 74, 75, 82).
- [55] Kaiming He et al. “Identity mappings in deep residual networks”. In: *European conference on computer vision*. Springer. 2016, pp. 630–645 (cit. on p. 82).
- [56] Paul S Heckbert. “A seed fill algorithm”. In: *Graphics gems*. Academic Press Professional, Inc. 1990, pp. 275–277 (cit. on p. 62).
- [57] Peter Henry et al. “RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments”. In: *The International Journal of Robotics Research* 31.5 (2012), pp. 647–663 (cit. on pp. 47, 48).
- [58] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366 (cit. on p. 70).
- [59] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *International Conference on Machine Learning*. 2015, pp. 448–456 (cit. on pp. 76, 82).
- [60] Itseez. *Open Source Computer Vision Library*. <https://github.com/itseez/opencv>. 2016 (cit. on p. 28).
- [61] Rasmus Jensen et al. “Large scale multi-view stereopsis evaluation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 406–413 (cit. on p. 93).

- [62] Pakorn KaewTraKulPong and Richard Bowden. “An improved adaptive background mixture model for real-time tracking with shadow detection”. In: *Video-based surveillance systems*. Springer, 2002, pp. 135–144 (cit. on p. 31).
- [63] O. Kahler et al. “Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device”. In: *IEEE Transactions on Visualization and Computer Graphics (Proceedings International Symposium on Mixed and Augmented Reality 2015)* 22.11 (2015) (cit. on p. 20).
- [64] Bernhard Kainz et al. “OmniKinect: real-time dense volumetric data acquisition and applications”. In: *Proceedings of the 18th ACM symposium on Virtual reality software and technology*. ACM. 2012, pp. 25–32 (cit. on p. 10).
- [65] Andrej Karpathy. *Stanford University CS231n Convolutional Neural Networks for Visual Recognition*. Stanford University, 2019. URL: <http://cs231n.github.io/> (cit. on p. 72).
- [66] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. “Poisson surface reconstruction”. In: *Proceedings of the fourth Eurographics symposium on Geometry processing*. Vol. 7. 2006 (cit. on pp. 48, 63).
- [67] Michael Kazhdan and Hugues Hoppe. “Screened poisson surface reconstruction”. In: *ACM Transactions on Graphics (ToG)* 32.3 (2013), p. 29 (cit. on pp. 69, 79, 92, 94).
- [68] Christian Keimel et al. “Video is a cube”. In: *Signal Processing Magazine, IEEE* 28.6 (2011), pp. 41–49 (cit. on pp. 29, 31).
- [69] Christian Kerl, Jurgen Sturm, and Daniel Cremers. “Dense visual SLAM for RGB-D cameras”. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 2100–2106 (cit. on p. 20).
- [70] Behrang Keshavarz et al. “Vection and visually induced motion sickness: how are they related?” In: *Frontiers in psychology* 6 (2015) (cit. on p. 3).
- [71] Kourosh Khoshelham and Sander Oude Elberink. “Accuracy and resolution of kinect depth data for indoor mapping applications”. In: *Sensors* 12.2 (2012), pp. 1437–1454 (cit. on p. 7).
- [72] Anthony Kim and MF Golnaraghi. “A quaternion-based orientation estimation algorithm using an inertial measurement unit”. In: *Position Location and Navigation Symposium, 2004. PLANS 2004*. IEEE. 2004, pp. 268–272 (cit. on p. 5).

- [73] Diederik P Kingma and Jimmy Lei Ba. “Adam: A method for stochastic optimization”. In: *Proceedings of the International Conference on Learning Representations*. 2015 (cit. on pp. 85, 97).
- [74] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *stat* 1050 (2014), p. 1 (cit. on p. 78).
- [75] Frank L Kooi and Alexander Toet. “Visual comfort of binocular and 3D displays”. In: *Displays* 25.2 (2004), pp. 99–108 (cit. on p. 4).
- [76] Mathieu Labbe and Francois Michaud. “Appearance-based loop closure detection for online large-scale and long-term operation”. In: *Robotics, IEEE Transactions on* 29.3 (2013), pp. 734–745 (cit. on pp. 53, 63).
- [77] Mathieu Labbé and François Michaud. “Memory management for real-time appearance-based loop closure detection”. In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE. 2011, pp. 1271–1276 (cit. on pp. 53, 63).
- [78] Mathieu Labbe and François Michaud. “Online global loop closure detection for large-scale multi-session graph-based slam”. In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE. 2014, pp. 2661–2666 (cit. on pp. 53, 63).
- [79] Po Kong Lai and Robert Laganière. “Creating Immersive Virtual Reality Scenes Using a Single RGB-D Camera”. In: *International Conference Image Analysis and Recognition*. Springer. 2017, pp. 221–230 (cit. on p. 60).
- [80] Po Kong Lai et al. “Video summarization of surveillance cameras”. In: *Advanced Video and Signal Based Surveillance (AVSS), 2016 13th IEEE International Conference on*. IEEE. 2016, pp. 286–294 (cit. on p. 29).
- [81] Iro Laina et al. “Deeper depth prediction with fully convolutional residual networks”. In: *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE. 2016, pp. 239–248 (cit. on pp. 69, 88, 89).
- [82] Anders Boesen Lindbo Larsen et al. “Autoencoding beyond pixels using a learned similarity metric”. In: *International Conference on Machine Learning*. 2016, pp. 1558–1566 (cit. on p. 78).

- [83] Steven M LaValle et al. “Head tracking for the Oculus Rift”. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 187–194 (cit. on pp. 4, 5).
- [84] Joseph J LaViola Jr. “A discussion of cybersickness in virtual environments”. In: *ACM SIGCHI Bulletin* 32.1 (2000), pp. 47–56 (cit. on p. 3).
- [85] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), p. 436 (cit. on p. 76).
- [86] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. “Epnnp: An accurate o (n) solution to the pnp problem”. In: *International journal of computer vision* 81.2 (2009), pp. 155–166 (cit. on p. 5).
- [87] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. “BRISK: Binary robust invariant scalable keypoints”. In: *2011 International conference on computer vision*. IEEE. 2011, pp. 2548–2555 (cit. on pp. xiii, 47).
- [88] Jesse Levinson et al. “Towards fully autonomous driving: Systems and algorithms”. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE. 2011, pp. 163–168 (cit. on p. 46).
- [89] Marc Levoy. “Light fields and computational imaging”. In: *Computer* 8 (2006), pp. 46–55 (cit. on p. 9).
- [90] Marc Levoy and Pat Hanrahan. “Light field rendering”. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 31–42 (cit. on p. 9).
- [91] Bo Li et al. “Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1119–1127 (cit. on p. 88).
- [92] Yangyan Li et al. “Database-assisted object retrieval for real-time 3d reconstruction”. In: *Computer Graphics Forum*. Vol. 34. 2. Wiley Online Library. 2015, pp. 435–446 (cit. on p. 77).
- [93] Zhuang Li, Prakash Ishwar, and Janusz Konrad. “Video condensation by ribbon carving”. In: *Image Processing, IEEE Transactions on* 18.11 (2009), pp. 2572–2583 (cit. on p. 30).

- [94] Yiyi Liao, Simon Donné, and Andreas Geiger. “Deep Marching Cubes: Learning Explicit Surface Representations”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2916–2925 (cit. on p. 88).
- [95] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440 (cit. on pp. 81, 89).
- [96] William E Lorensen and Harvey E Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *ACM siggraph computer graphics*. Vol. 21. 4. ACM. 1987, pp. 163–169 (cit. on pp. xiv, 48).
- [97] Manolis IA Lourakis and Antonis A Argyros. “SBA: A software package for generic sparse bundle adjustment”. In: *ACM Transactions on Mathematical Software (TOMS)* 36.1 (2009), p. 2 (cit. on pp. 22, 48, 63).
- [98] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110 (cit. on pp. xv, 47).
- [99] Feng Lu and Evangelos Milios. “Globally consistent range scan alignment for environment mapping”. In: *Autonomous robots* 4.4 (1997), pp. 333–349 (cit. on p. 46).
- [100] Zhaoliang Lun et al. “3D shape reconstruction from sketches via multi-view convolutional networks”. In: *3D Vision (3DV), 2017 International Conference on*. IEEE. 2017, pp. 67–77 (cit. on pp. 77–81).
- [101] Marcus Magnor, Prashant Ramanathan, and Bernd Girod. “Multi-view coding for image-based rendering using 3-D scene geometry”. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 13.11 (2003), pp. 1092–1106 (cit. on p. 9).
- [102] Martin Magnusson, Achim Lilienthal, and Tom Duckett. “Scan registration for autonomous mining vehicles using 3D-NDT”. In: *Journal of Field Robotics* 24.10 (2007), pp. 803–827 (cit. on p. 50).
- [103] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. “Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections”. In: *Advances in neural information processing systems*. 2016, pp. 2802–2810 (cit. on p. 81).

- [104] Dushyant Mehta et al. “Single-shot multi-person 3d pose estimation from monocular rgb”. In: *2018 International Conference on 3D Vision (3DV)*. IEEE. 2018, pp. 120–130 (cit. on p. 99).
- [105] Dushyant Mehta et al. “Vnect: Real-time 3d human pose estimation with a single rgb camera”. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), p. 44 (cit. on p. 99).
- [106] Facundo Mémoli and Guillermo Sapiro. “Comparing point clouds”. In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. ACM. 2004, pp. 32–40 (cit. on pp. 23, 24).
- [107] Sebastian Montabone and Alvaro Soto. “Human detection using a mobile platform and novel features derived from a visual saliency mechanism”. In: *Image and Vision Computing* 28.3 (2010), pp. 391–402 (cit. on p. 22).
- [108] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814 (cit. on p. 82).
- [109] Donald J Natale et al. “Using full motion 3D Flash LIDAR video for target detection, segmentation, and tracking”. In: *Image Analysis & Interpretation (SSIAI), 2010 IEEE Southwest Symposium on*. IEEE. 2010, pp. 21–24 (cit. on p. 8).
- [110] Ulrich Neumann, Thomas Pintaric, and Albert Rizzo. “Immersive panoramic video”. In: *Proceedings of the eighth ACM international conference on Multimedia*. ACM. 2000, pp. 493–494 (cit. on p. 8).
- [111] Richard A Newcombe, Dieter Fox, and Steven M Seitz. “Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 343–352 (cit. on pp. 11, 46, 59).
- [112] Richard A Newcombe et al. “KinectFusion: Real-time dense surface mapping and tracking”. In: *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE. 2011, pp. 127–136 (cit. on pp. 20, 48, 59, 68).
- [113] Ren Ng et al. “Light field photography with a hand-held plenoptic camera”. In: *Computer Science Technical Report CSTR* 2.11 (2005), pp. 1–11 (cit. on pp. 9, 10).

- [114] David Nister and Henrik Stewenius. “Scalable recognition with a vocabulary tree”. In: *Computer vision and pattern recognition, 2006 IEEE computer society conference on*. Vol. 2. Ieee. 2006, pp. 2161–2168 (cit. on p. 48).
- [115] Fakir S. Nooruddin and Greg Turk. “Simplification and repair of polygonal models using volumetric techniques”. In: *IEEE Transactions on Visualization and Computer Graphics* 9.2 (2003), pp. 191–205 (cit. on p. 93).
- [116] Andreas Nüchter et al. “6D SLAM—3D mapping outdoor environments”. In: *Journal of Field Robotics* 24.8-9 (2007), pp. 699–722 (cit. on p. 50).
- [117] Neeti A Ogale. “A survey of techniques for human detection from video”. In: () (cit. on p. 22).
- [118] Shmuel Peleg, Moshe Ben-Ezra, and Yael Pritch. “Omnistereo: Panoramic stereo imaging”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 23.3 (2001), pp. 279–290 (cit. on p. 9).
- [119] Massimo Piccardi. “Background subtraction techniques: a review”. In: *Systems, man and cybernetics, 2004 IEEE international conference on*. Vol. 4. IEEE. 2004, pp. 3099–3104 (cit. on p. 28).
- [120] Yael Pritch, Alex Rav-Acha, and Shmuel Peleg. “Nonchronological video synopsis and indexing”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30.11 (2008), pp. 1971–1984 (cit. on pp. 30, 31).
- [121] Yael Pritch et al. “Clustered synopsis of surveillance video”. In: *Advanced Video and Signal Based Surveillance, 2009. AVSS’09. Sixth IEEE International Conference on*. IEEE. 2009, pp. 195–200 (cit. on pp. 30, 31).
- [122] Charles R Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660 (cit. on p. 78).
- [123] Charles Ruizhongtai Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5099–5108 (cit. on pp. 78, 79).
- [124] Alex Rav-Acha, Yael Pritch, and Shmuel Peleg. “Making a long video short: Dynamic video synopsis”. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2006, pp. 435–441 (cit. on pp. 30, 31).

- [125] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. “OctNet: Learning Deep 3D Representations at High Resolutions”. In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE. 2017, pp. 6620–6629 (cit. on pp. 78, 79).
- [126] Jason Rock et al. “Completing 3d object shape from one depth image”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2484–2493 (cit. on p. 77).
- [127] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241 (cit. on p. 81).
- [128] Szymon Rusinkiewicz and Marc Levoy. “Efficient variants of the ICP algorithm”. In: *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. IEEE. 2001, pp. 145–152 (cit. on pp. 47, 48).
- [129] Hanan Samet and Markku Tamminen. “Efficient component labeling of images of arbitrary dimension represented by linear bintrees”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 10.4 (1988), pp. 579–586 (cit. on p. 5).
- [130] Hamed Sarbolandi, Damien Lefloch, and Andreas Kolb. “Kinect range sensing: Structured-light versus Time-of-Flight Kinect”. In: *Computer vision and image understanding* 139 (2015), pp. 1–20 (cit. on p. 68).
- [131] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. “Fast image-based localization using direct 2d-to-3d matching”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 667–674 (cit. on p. 22).
- [132] Manolis Savva et al. “Shrec’16 track large-scale 3d shape retrieval from shapenet core55”. In: *Proceedings of the eurographics workshop on 3D object retrieval*. 2016 (cit. on pp. 84, 85).
- [133] Daniel Scharstein and Richard Szeliski. “High-accuracy stereo depth maps using structured light”. In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2003, pp. I–195 (cit. on p. 7).

- [134] Steven M Seitz et al. “A comparison and evaluation of multi-view stereo reconstruction algorithms”. In: *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2006, pp. 519–528 (cit. on p. 10).
- [135] Jun’ichiro Seyama and Ruth S Nagayama. “The uncanny valley: Effect of realism on the impression of artificial human faces”. In: *Presence* 16.4 (2007), pp. 337–351 (cit. on p. 1).
- [136] Abhishek Sharma, Oliver Grau, and Mario Fritz. “Vconv-dae: Deep volumetric shape learning without object labels”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 236–250 (cit. on pp. 69, 77, 78).
- [137] Jamie Shotton et al. “Real-time human pose recognition in parts from single depth images”. In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 1297–1304 (cit. on p. 99).
- [138] Jamie Shotton et al. “Real-time human pose recognition in parts from single depth images”. In: *Communications of the ACM* 56.1 (2013), pp. 116–124 (cit. on pp. 60, 62).
- [139] Zhenyu Shu et al. “3D model classification via Principal Thickness Images”. In: *Computer-Aided Design* 78 (2016), pp. 199–208 (cit. on p. 80).
- [140] Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. *Image-based rendering*. Springer Science & Business Media, 2008 (cit. on p. 9).
- [141] Heung-Yeung Shum, Sing Bing Kang, and Shing-Chow Chan. “Survey of image-based representations and compression techniques”. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 13.11 (2003), pp. 1020–1037 (cit. on p. 9).
- [142] Ayan Sinha, Jing Bai, and Karthik Ramani. “Deep learning 3D shape surfaces using geometry images”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 223–240 (cit. on pp. 78–80).
- [143] J. Sivic and A. Zisserman. “Video Google: a text retrieval approach to object matching in videos”. In: *Proceedings Ninth IEEE International Conference on Computer Vision*. 2003, 1470–1477 vol.2. DOI: [10.1109/ICCV.2003.1238663](https://doi.org/10.1109/ICCV.2003.1238663) (cit. on p. 48).

- [144] Mel Slater and Sylvia Wilbur. “A framework for immersive virtual environments (FIVE): Speculations on the role of presence in virtual environments”. In: *Presence: Teleoperators & Virtual Environments* 6.6 (1997), pp. 603–616 (cit. on pp. 17, 111).
- [145] Andres Solis Montero, Jochen Lang, and Robert Laganieri. “Scalable Kernel Correlation Filter With Sparse Feature Integration”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2015, pp. 24–31 (cit. on pp. 60, 62, 64).
- [146] Luciano Spinello and Kai O Arras. “People detection in RGB-D data”. In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE. 2011, pp. 3838–3843 (cit. on p. 22).
- [147] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958 (cit. on pp. 77, 82).
- [148] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. “Highway networks”. In: *arXiv preprint arXiv:1505.00387* (2015) (cit. on p. 74).
- [149] Chris Stauffer and W Eric L Grimson. “Adaptive background mixture models for real-time tracking”. In: *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on*. Vol. 2. IEEE. 1999 (cit. on p. 28).
- [150] Jürgen Sturm et al. “A benchmark for the evaluation of RGB-D SLAM systems”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 573–580 (cit. on p. 64).
- [151] David Stutz and Andreas Geiger. “Learning 3D Shape Completion under Weak Supervision”. In: *arXiv preprint arXiv:1805.07290* (2018) (cit. on pp. 69, 77, 78, 88, 93, 99).
- [152] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010 (cit. on pp. 7, 9, 10, 12).
- [153] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. “Multi-view 3d models from single images with a convolutional network”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 322–337 (cit. on pp. 77, 79–81).

- [154] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. “Octree Generating Networks: Efficient Convolutional Architectures for High-Resolution 3D Outputs”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2088–2096 (cit. on pp. 78, 79).
- [155] Keisuke Tateno et al. “CNN-SLAM: Real-Time Dense Monocular SLAM With Learned Depth Prediction”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6243–6252 (cit. on p. 69).
- [156] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. 2005 (cit. on p. 46).
- [157] Beau Tippetts et al. “Review of stereo vision algorithms and their suitability for resource-limited systems”. In: *Journal of Real-Time Image Processing* 11.1 (2016), pp. 5–25 (cit. on pp. 7, 58).
- [158] Stavros Tzavidas and Aggelos K Katsaggelos. “A multicamera setup for generating stereo panoramic video”. In: *IEEE Transactions on multimedia* 7.5 (2005), pp. 880–890 (cit. on p. 58).
- [159] Arne Valberg. *Light vision color*. John Wiley & Sons, 2007 (cit. on p. 53).
- [160] Jacob Varley et al. “Shape completion enabled robotic grasping”. In: *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE. 2017, pp. 2442–2447 (cit. on pp. 69, 77, 78).
- [161] Roberto Vezzani and Rita Cucchiara. “Video surveillance online repository (visor): an integrated framework”. In: *Multimedia Tools and Applications* 50.2 (2010), pp. 359–380 (cit. on p. 36).
- [162] Daniel Vlasic et al. “Articulated mesh animation from multi-view silhouettes”. In: *ACM Transactions on Graphics (TOG)*. Vol. 27. 3. ACM. 2008, p. 97 (cit. on p. 84).
- [163] Hanzi Wang and David Suter. “A re-evaluation of mixture of Gaussian background modeling [video signal processing applications]”. In: *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP’05). IEEE International Conference on*. Vol. 2. IEEE. 2005, pp. ii–1017 (cit. on p. 28).
- [164] Wendell R Watkins, Jay B Jordan, and Mohan M Trivedi. “Novel applications of hyperstereo vision”. In: *10th Meeting on Optical Engineering in Israel*. International Society for Optics and Photonics. 1997, pp. 72–88 (cit. on p. 4).

- [165] T Whelan, M Kaess, and MF Fallon. “Kintinuous: Spatially Extended Kinect Fusion”. In: *Workshop on RGB-D: Advanced Reasoning with Depth Cameras, in conjunction with Robotics: Science and Systems*. 2012 (cit. on p. 50).
- [166] Thomas Whelan et al. “ElasticFusion: Dense SLAM without a pose graph”. In: *Robotics: Science and Systems* (cit. on pp. 46, 48).
- [167] Thomas Whelan et al. “Robust real-time visual odometry for dense RGB-D mapping”. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 5724–5731 (cit. on p. 50).
- [168] Walter Wohlkinger and Markus Vincze. “Shape-based depth image to 3d model matching and classification with inter-view similarity”. In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE. 2011, pp. 4865–4870 (cit. on p. 77).
- [169] Oliver J Woodman. “An introduction to inertial navigation”. In: (2007) (cit. on p. 5).
- [170] Jiajun Wu et al. “Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 82–90 (cit. on pp. 77, 78).
- [171] Zhirong Wu et al. “3d shapenets: A deep representation for volumetric shapes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920 (cit. on pp. 78, 93).
- [172] Lu Xia, Chia-Chih Chen, and Jake K Aggarwal. “View invariant human action recognition using histograms of 3d joints”. In: *Computer vision and pattern recognition workshops (CVPRW), 2012 IEEE computer society conference on*. IEEE. 2012, pp. 20–27 (cit. on p. 99).
- [173] Yasushi Yagi and Masahiko Yachida. “Real-time omnidirectional image sensors”. In: *International Journal of Computer Vision* 58.3 (2004), pp. 173–207 (cit. on p. 58).
- [174] Qingxiong Yang et al. “Fusion of active and passive sensors for fast 3d capture”. In: *Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on*. IEEE. 2010, pp. 69–74 (cit. on p. 59).

- [175] Matthew D Zeiler et al. “Deconvolutional networks”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 2528–2535 (cit. on p. 81).
- [176] Zhengyou Zhang. “Iterative point matching for registration of free-form curves and surfaces”. In: *International journal of computer vision* 13.2 (1994), pp. 119–152 (cit. on p. 47).
- [177] Zhengyou Zhang. “Microsoft kinect sensor and its effect”. In: *IEEE multimedia* 19.2 (2012), pp. 4–10 (cit. on pp. 58, 68).
- [178] Wei Zhao, Shuming Gao, and Hongwei Lin. “A robust hole-filling algorithm for triangular mesh”. In: *The Visual Computer* 23.12 (2007), pp. 987–997 (cit. on p. 69).
- [179] Jiejie Zhu et al. “Fusion of time-of-flight depth and stereo for high accuracy depth maps”. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8 (cit. on p. 10).
- [180] Zoran Zivkovic. “Improved adaptive Gaussian mixture model for background subtraction”. In: *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. Vol. 2. IEEE. 2004, pp. 28–31 (cit. on p. 28).