

Abstract of “Compression of 2-d and 3-d Dynamic scenes” by Octavian Biris, Ph.D., Brown University, May 2015.

This thesis is dedicated to exploring compression methods for large dynamic scenes. The first part presents a case study of compressing aerial video using background modeling and JPEG2000. The concepts used in the case study relate to state of the art compression frameworks and lay down the foundation for the methods used to compress 4-d scenes in the second part. The algebra of random variables and multiresolution analysis are used throughout this thesis as a means of compressing large dynamic data. The second part of thesis presents a novel method to estimate dense scene flow using volumetric and probabilistic 3-d models. The method first reconstructs 3-d models at each time step using images synchronously captured from multiple views. Then, the 3-d motion between two consecutive 3-d models is estimated using a formulation that is the analog of Horn and Schunck’s optical flow method. This particular choice of 3-d model representation allows estimating highly dense scene flow results, tracking of surfaces undergoing topological change and reliably recovering large motion displacements. The benefits of the method and the accuracy of 3-d flow results are demonstrated on recent multi-view datasets. The second goal of this work is to compress and reconstruct 3-d scenes at various time points using the estimated flow. A new method of scene warping is proposed that involves partitioning the optical flow field in regions of coherent motion which are subsequently parametrized by affine transformations. The compression objective of this work is achieved by the low storage requirements of the affine parameters that describe the optical flow field and the efficient reconstruction method through warping.

Compression of 2-d and 3-d Dynamic scenes

by

Octavian Biris

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Engineering at Brown University

Providence, Rhode Island

May 2015

© Copyright 2015 by Octavian Biris

This dissertation by Octavian Biris is accepted in its present form by
the Department of Engineering as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____
Joseph L. Mundy, Director

Recommended to the Graduate Council

Date _____
Gabriel Taubin, Reader

Date _____
Pedro Felzenszwalb, Reader

Approved by the Graduate Council

Date _____
Dean of the Graduate School

Acknowledgments

This dissertation was not possible without the contribution and support of many people to whom I am deeply thankful.

First of all I want to thank prof. Joseph L. Mundy for giving me the opportunity to work in his lab in 2009. He has guided and encouraged me towards every achievement I had earned during my career as a PhD candidate. The critical thinking, problem solving skills, presentation abilities and programming skills are only a few of the many assets prof. Mundy has imparted me during my studies. In addition, he was extremely positive and helped take the edge off in moments when obstacles seemed insurmountable to me. Overall, his influence and mentorship helped me become the person I had wished to be at the beginning of my post-graduate career. I would also like to thank my thesis committee members prof. Gabriel Taubin and prof. Pedro Felzenszwalb for the constructive suggestions that helped shape this dissertation. From my preliminary exam committee members I would like to thank prof. Thomas Banchoff for the invaluable perspective on mathematics he offered me during his differential geometry and combinatorial topology courses, as well as prof. Harvey F. Silverman whose lectures on Digital Signals Processing helped shape my career as an engineer. I would also like to thank my undergraduate mentor prof. Elaine Bearer for guiding me towards the field of image understanding and academic research and ultimately to my path towards graduate school.

I am grateful to my family, especially my mother Sofia Biris and sister Dr. Octavia Biris whose unwavering support propelled and guided me in the fields of physics and computer science starting from an early age. I would also like to thank my friends in Romania, Adrian Juverdeanu, Razvan Tuderoiu, Alin Ratiu, Paul Gherman, Dan Dragomir, Mirona Grecu, Mirona Urs, Bianca Varga and Roxana Lupei for making home seem as if I had never left, despite my sporadic visits during the last few years.

My undergraduate career at Brown University was made all the more pleasant by my gang, Dr. Teodor Mihai Moldovan, Adrian Vladu, Laurentiu Rodina, Tiberiu Florea, Dr. Iuliana Ene, Monica Mocanu, Alexandre de Chaumont Quitry, Max Harshorn Shimshak, Nick Imbriglia, Alexander Fye and Yagiz Ozyol.

My PhD career wouldn't have been as rewarding and enriching if it weren't for my fellow peers, Dr. Kumud Nepal, Dr. Cesare Ferri, Dr. Rotor Le, Onur Can Ulusel, Dr. Brandon Asher Mayer, Dr. Thomas Carle, Dimitra Papagiannopoulou, Eirini Kilikian, Michela Ronzani, Ynes Helou, Veronica Ciocanel. I would like to give special thanks Dr. Ali Osman Ulusoy whose help and insights were instrumental in my research and this dissertation. Also, I am indebted to Dr. Isabel Restrepo who has helped me tremendously with my research, offering me valuable suggestions and advice. Also, I am grateful towards my office buddy, gym buddy and good friend Marco Donato who was there for me during both the good and the more difficult times of grad school. Last but not least, I would like to thank my two roommates Dr. Ryan J. Cochran and Anil Usumezbas for making my stay at 111 Governor a pleasant one.

Finally, I would like to thank my ray of sunshine Dr. Anda Chirila who was by my side ever since I started my journey in 2004.

Contents

List of Tables	ix
List of Figures	x
1 Introduction	1
1.1 Prior Works	3
1.2 Thesis Contributions	8
1.3 Thesis Outline	9
2 JPEG2000 as a Model for Large Scale Encoding	11
2.1 The Discrete Wavelet Transform in JPEG2000	13
2.1.1 Wavelets and their Filters	13
2.1.2 Determining Daubechie Wavelet-Filter Coefficients	15
2.1.3 Polynomial Factorization of Wavelet-Filter Frequency Response	17
2.1.4 The 2-d Wavelet Transform in JPEG2000	18
2.2 Data organization and Coding in JPEG2000	19
2.2.1 JPEG2000 Bitplane Coding	20
2.2.2 Entropy Coding in JPEG2000	27
2.3 Conclusion	28
3 High Resolution Video Compression Using Background Modeling and JPEG2000	30
3.1 Surveillance Video Compression	32
3.1.1 Background Modeling	32
3.1.2 The JPEG2000 standard	35

3.1.3	Compression of Background Models using JPEG 2000	36
3.2	Experiments	41
3.2.1	Results	42
3.3	Conclusions	44
4	4-d Modeling	46
4.1	Mesh Based Representation and Compression	46
4.2	Point Based Representation and Compression	47
4.3	Probabilistic Volumetric Models	49
4.3.1	On-line PVM Inference	52
4.3.2	Occlusion Density and Visibility - How not to be Seen	59
4.3.3	GPU Implementation of the Non-Uniform Voxel Grid	64
4.4	Conclusions	66
5	Multiresolution 3-d Optical Flow	67
5.1	Introduction	67
5.2	Related Work in Scene Flow	70
5.3	Coarse-to-fine Scene Flow	72
5.3.1	Derivative Estimation on a Non-uniform Volumetric Grid	73
5.3.2	Scene Warping using Estimated Flow	73
5.4	Expected Flow in 2-d	75
5.5	Synthetic Data	77
5.6	Conclusions	79
6	Probabilistic Model Reconstruction via Estimated Motion	83
6.1	Optical Flow Field Decomposition into Affine Motion	85
6.2	Scene Warping Using Affine Motion	85
6.3	Scene Compression and Reconstruction Using Affine Motion	90
6.4	Results and Discussion	92
6.5	Conclusions	97

7 Conclusions and Future Work

100

- ★ Parts of this dissertation appeared in proceedings of conferences or in journals. In particular, Chapter 3 is an extended version of [13]

List of Tables

2.1	Zero Coding - LL and LH	23
2.2	Zero Coding - HL	23
2.3	Zero Coding - HH	23
2.4	Magnitude Refinement Coding Rules	23
2.5	Sign Coding Rules	24
2.6	The state machine table of the MQ coder [68]. The state index I is a function of the 19 different coding contexts <i>i.e.</i> $I(Cx)$. The table keeps track of the index of the next state based on whether an LPS or MPS was coded. For example if, an LPS is coded, the next state is $NLPS(I(Cx))$. Additionally, each state contains the current estimate of the LPS probability Q_e	29
3.1	Compression Ratios for two Video Sequences.	42
5.1	SSIM measure comparing renderings of the cube warped with the estimated flow and the synthetically rotated cube (target scene)	80
6.1	Average SSIM measure between warped scene renderings and original scene renderings across 100 frames. The resultant flow was used to warp the sence in the first column while the set of affine transforms was used in the second one	96
6.2	Compression ratio between the storage cost of the label field $\mathcal{L}$ and the entire flow field (left column) and scene parameters (right column)	97

List of Figures

1.1	Mesh Based Modeling in the HBO Series "Game of Thrones"	2
1.2	Inter-view prediction of the MVC extension of H.264 as described by Merkle and Smolic [92]	5
1.3	The pipeline proposed by Huang and Boyer [65]. Parameters of skeletal joint positions are estimated by fitting a reference human model (represented as a patched mesh and a skeleton) to reconstructed 3-d surfaces from imagery represented as a point cloud. The point cloud is then partitioned and compared against the estimated model at the previous frame, producing the current human model estimate. Image taken from Huang and Boyer [65].	7
1.4	Visual Hull as seen in [32]	7
2.1	JPEG2000 and JPEG compressed images at a compression rate of 1:80. While the artifacts on the JPEG-compressed image (right) are clearly noticeable, one needs to zoom in significantly to notice the ones resulting from the JPEG2000 encoded image (left)	12
2.2	(a): Haar wavelet and Scaling function; (b): Haar wavelet Lowpass and Highpass filter impulse responses $h_\phi[n]$ and $h_\psi[n]$, respectively	15
2.3	(a): The 5-3 Le Gall Scaling function ϕ and wavelet ψ used in JPEG2000 reversible compression; (b): The 5-3 Le Gall Low Pass and High Pass filter impulse responses $h_\phi[n]$ and $h_\psi[n]$, respectively. The number of filter taps in the two impulse responses earned its namesake.	18
2.4	JPEG2000 Data Organization and 2-d wavelet transform	21
2.5	Context Window (left) and Sign Context Window (right)	22

2.6	JPEG2000 Coding Pass Example. The second row represents an abstraction of the bitlists $\mathcal{P}^{k,s}$ corresponding to each coding pass. Note how the length $\mathcal{P}^{k,s}$ varies across bitplanes k and passes s . The bottom part shows a detailed view of the (Cx, D) pairs resulting from encoding the first bitplane	26
3.1	Background model maintaining the distribution of background as a mixture of Gaussians at each pixel	33
3.2	Convolution of two GMMs increases the number of components; The two lowest weighted components are merged until the desired number of components is reached	37
3.3	Probability map evaluation using LL5	39
3.4	(a): Percent of pixels requiring local refinement vs. bit rate. Each curve represents a different decision threshold T for local decompression. (b): ROC characteristic curves for various bitrates	41
3.5	Number of local refinements required with a decision threshold of 0.7 and a model compressed at a rate of (a) 0.05 bps and (b) 32 bps	42
3.6	(a) Mean image $\mathbf{M}_{k,50}$ (b) probability map evaluated with model encoded at 5 bps (c) Segmented Foreground (d) Reconstructed Frame $d = a + c$	43
3.7	Encoded foreground (a) lossy 0.01 bps (no mask), (b) lossy at 0.01 bps , (c) lossy at 0.05 bps and (d) lossless	44
4.1	Mesh based representation by Arcila <i>et al.</i> [8]. Here mesh connectivity is unable to deal with changing scene topology	47
4.2	Surfel Representation as presented by Tonnesen <i>et al.</i> [119]. The non-connectivity property of the point cloud allows the modeling of surface deformations such as a cloth tear.	48
4.3	The non-uniform voxel grid refines 3-d space corresponding to surfaces. Illustration taken from Crispell <i>et al.</i> [35]	50
4.4	4-d space representation using Time Trees as presented by Ulusoy <i>et al.</i> [123]	51
4.5	3-d space modeling using probabilistic volumetric models. Surface geometry is encoded using smaller volumetric elements (voxels) than the ones representing empty space. Scene structure is maintained through octrees. The surface and appearance parameters at each cell are learned from image intensity observations	52

4.6	Scene uncertainty and ambiguity as exemplified by Crispell <i>et al.</i> [35]. (a) Two cameras capture three distinct featureless surfaces. (b) The Visual Hull Reconstruction method. (c) The probabilistic model. Here, the true surface may lie anywhere within the shaded regions. (d) Reconstruction using a prior shape model, <i>i.e.</i> the dotted line.	56
4.7	The view dependent appearance model as explained by by Ulusoy <i>et al.</i> [123]. Note how the grayed directions do not contribute to the final distribution $p(I; V)$, as they face away from the camera ray V . Since V_1 faces V almost directly, it will have the highest contribution w_1 while canonical directions V_2 and V_3 will have some nonzero contributions w_2 and w_3 , respectively	58
4.8	Renderings of scene where appearance has been modeled using (a) Single-Gaussian, (b) Mixture of Gaussians,(c) the View-dependent Appearance model. Image taken from Ulusoy <i>et al.</i> [123]	59
4.9	Quantities α , ω and vis corresponding to a ray piercing a hollow, transparent cube. The voxels corresponding to the cube faces have surface probabilities that are less than 1. Note how $\alpha(s)$ is maximal at the ray-cube intersection points. The visibility drops as the ray pierces the cube faces. Evidently, the intersection point on the top face is more likely to be the ray occlusion point, as seen in the $\omega(s)$ plot	61
4.10	The partitioning of a ray into segments bounded by ray-voxel intersections, as illustrated by Crispell <i>et al.</i> [35]	63
4.11	Expected Image Rendering of the Barus & Holley building at Brown University . . .	64
4.12	Binary bit tree and associated binary word. Note how C's descendants are marked with an $\times$ and their corresponding bit is set to 0	65
4.13	Octree representation and associated data buffers	65
5.1	The estimated 3-d motion is visualized as 2-d line projections. The figure shows flow results from three consecutive frames of the "Boy playing ball" sequence. Note that the flow vectors correctly point in the direction of the ball's motion.	68
5.2	Estimated 3-d flow projected using the Expected-flow method onto novel view rendering of consecutive time instances of the "Adult Child Ball" sequence. Despite the ball's large motion (3-4 voxels at the coarsest level or 12 cm) the flow is recovered properly	70

5.3	Estimated 3-d flow projected onto Novel-view rendering of consecutive time instances of the “Dancer” sequence. The resultant flow in this sequence matches accurately the voxels in the next time instance. Full video Sequence available in the supplementary material	71
5.4	Estimated scene flow projected onto Novel-view renderings of consecutive time instances of the “Guard Punch Two” sequence. The flow is estimated robustly even in areas where surface geometry is poorly reconstructed due to lack of camera coverage	75
5.5	The expected flow is computed as the weighted sum of projected flow vector end-points. The weights represent the likelihood that a voxel is occluding the viewing ray	76
5.6	Adult Child Ball 2-d Flow compared methods: Proposed Expected Flow (a), Brox (b) [22], Farneback (c) [40], TV-L1 (d) [138]	77
5.7	Adult Child Ball 2-d Flow compared methods: Proposed Expected Flow (a), Brox (b) [22], Farneback (c) [40], TV-L1 (d) [138]	78
5.8	Dancer 2-d Flow compared methods: Proposed Expected Flow (a), Brox (b) [22], Farneback (c) [40], TV-L1 (d) [138]	79
5.9	Flow Angular error in radians (top) and End-point error in voxels (bottom) as a function of maximum displacement in voxels for the synthetic cube	80
5.10	Estimated flow for synthetic cube for various amounts of rotation	80
5.11	Synthetic Cube rotation and warping : Synthetically rotated (top), Warped with resultant flow (bottom)	81
5.12	Guard Punch Two 2-d Flow compared methods: Proposed Expected Flow (a), Brox (b) [22], Farneback (c) [40], TV-L1 (d) [138]	81
5.13	Boy Playing Ball 2-d Flow compared methods: Proposed Expected Flow (a), Brox (b) [22], Farneback (c) [40], TV-L1 (d) [138]	82
6.1	Regions of coherent motion and corresponding histogram of flow vector orientations for one instance of the "Dancer" sequence. The bins accumulate the magnitude of the optical flow vectors, <i>i.e.</i> $\mathbf{r} \cdot \mathbf{r}$	84
6.2	Four views of the Expected Flow (Top) and Affine Flow (Bottom) for one instance of the “Boy Playing Ball” sequence	86

6.3	Four views of the Expected Flow (Top) and Affine Flow (Bottom) for one instance of the “Dancer” sequence	87
6.4	Four views of the Estimated Flow (Top) and Affine Flow (Bottom) for one instance of the “Guard Punch Two” sequence	88
6.5	Two views of the Estimated Flow (Top) and Affine Flow (Bottom) for one instance of the “Adult Child Ball” sequence	89
6.6	Two views of the Estimated Flow (Top) and Affine Flow (Bottom) for one instance of the “Adult Child Ball” sequence	90
6.7	Renderings of original scene (Top) and its reconstruction using Affine Flow and the previous scene (Bottom) for one instance of the “Boy Playing Ball” sequence	91
6.8	Renderings of original scene (Top) and its reconstruction using Affine Flow and the previous scene (Bottom) for one instance of the “Dancer” sequence	92
6.9	Renderings of original scene (Top) and its reconstruction using Affine Flow and the previous scene (Bottom) for one instance of the “Guard Punch Two” sequence	93
6.10	Renderings of original scene (Top) and its reconstruction using Affine Flow and the previous scene (Bottom) for one instance of the “Adult Child Ball” sequence	94
6.11	Renderings of original scene (Top) and its reconstruction using Affine Flow and the previous scene (Bottom) for one instance of the “Adult Child Ball” sequence	95
6.12	Comparison with the results from [53] on the “Dancer” sequence. (A) Estimated flow for frame 11. Note how the dense optical flow is corresponding to the dancer’s the pirouette motion both on the arms and on the dress. (B) frame 11 and frame 12 are overlayed, underscoring the correct magnitude of the resultant flow. (C) Result obtained by [53] on the same frame. Although the vectors seem correct, the lack of density shows little to no flow on the dress.	96
6.13	Linearly fitted flow using Figure 6.5 (A_1) and original resultant flow (A_2); Rendering of warped scene using to affine flow (B_1) and estimated flow (B_2)	97
6.14	Top: Label field $\mathcal{L}$ containing the index of the most likely cluster at each voxel. Each color represents an integer index; Bottom: The posterior probability field corresponding to the most likely cluster, <i>i.e.</i> $\max_i \mathcal{P}$. Here, the $[0, 1]$ range has been mapped to the $R > G > B$ heatmap encoding	98

6.15 Segmentation into 5 clusters using position and estimated flow for one of the frames from each of the tested sequences. Note how voxels with similar motion belong to the same cluster	99
---	----

Chapter 1

Introduction

The field of computer vision is dedicated to explaining and characterizing the world through visual data. Up until recently, 2-d images were the de facto source on which inference about the world was made. However, with the advent of more powerful hardware, interests in computer vision have shifted to modeling and characterizing the 3-d world in its entirety. Since the mode of acquisition hasn't evolved much over time, 2-d images are still primarily the input for 3-d computer vision algorithms. As a result, the topics of 3-d reconstruction from images have experienced a surge of effort over the past decade. Moreover, older 2-d algorithms are being recast into 3-d in order to improve inference about the physical world by making use of the third dimension. Also, adding a temporal dimension to the 3-d world allows an even broader set of applications ranging from gaming, urban city monitoring [105][120][106], construction site monitoring [51][50], telepresence [52], 3-d-TV [89] [41] and freeviewpoint video [44] [113]. These applications do not have a standardized algorithm for modeling and reconstruction and, consequently, the quality of representation, ease of reconstruction and ease of world inference on the reconstructed scene vary significantly. However, the storage costs of an accurate depiction of the 3-d world are considerable, regardless of the reconstruction and modeling method used. Thus the objective of compressing a 4-d world representation follows immediately. Based on the modeling and reconstruction method used, redundancies in each dimension can be exploited differently. Consequently, compressing the 4-d world is intrinsic to its representation.

The most common methods of physical world modeling can be separated into two different categories, namely geometric based modeling and image based modeling. Geometric modeling in

3-d typically involves representing objects through meshes and more recently point clouds. Research in computer graphics, gaming and the movie industry adopted mesh-based geometric modeling as a means of representing objects in the 3-d world which in turn enabled hardware manufacturers to develop dedicated chips for mesh processing. However, in order to achieve photo-realism in a highly dynamic scene, depth information, surface geometry and camera geometry is needed a priori. Thus a high end production such as a movie requires user-driven interaction during the reconstruction process. Sec. 4.1 will elaborate more on meshes and their use in 4-d characterization. The first

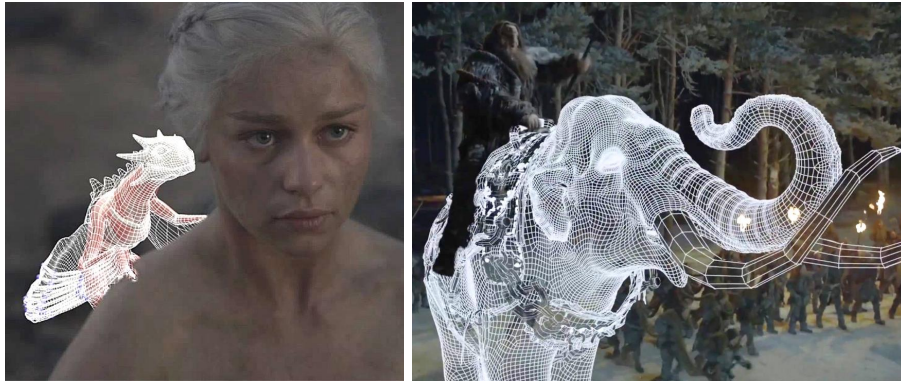


Figure 1.1: Mesh Based Modeling in the HBO Series "Game of Thrones"

existing image based modeling methods lie on the other extreme, avoiding the representation of 3-d geometry altogether. Instead, these methods provide only virtual views inferred from input imagery. The theoretical basis behind initial image-based modeling methods relies on the sampling of the plenoptic function [5]. This function parametrizes the intensity of light rays in 4-d space for every position, every direction, every wavelength at any given time. Andelson and Berger coined the term from the latin root plenus meaning complete or full, implying that the function provides a complete representation of visual fields. Examples include ray-space representations [44][45] and panoramic views which do not allow for completely arbitrary viewpoints [91] [110]. In addition, to be able to re-create a photo-realistic virtual 3-d scene, a dense sampling of views is required. A sparse distribution of cameras will significantly impact the scene reconstruction quality.

Some methods have the ability to represent pseudo 3-d surface geometry through depth-maps. Each pixel in the depth map or Z -map contains the Z -coordinate of its projection in 3-d, assuming the depth map frame is aligned with the 3-d $X - Y$ axis. These methods are referred to as 2.5-d as they do not explicitly represent surface geometry but rather give the impression of 3-d. The closest

image based-modeling methods to geometric modeling methods are hybrid methods that actually reconstruct surface geometry from imagery, including surface light fields [29] [136], view-dependent texture mapping methods [94] [60] and voxel models [88] [100] [35] [108]. For the purposes of this thesis the latter category of modeling is used as it assumes no prior information about scene geometry. A detailed treatment of the 4-d world representation used is given in Chapter 4.

The first part of this thesis will address large time-varying 2-d data compression in order to lay down the ground work for 4-d compression presented in the second part. The case study of compressing large 2-d dynamic scenes involves using the JPEG2000 standard in conjunction with background modeling. The JPEG2000 standard is used as a method for compression due to its multiresolution implementation and increased performance in low-bitrate encoding. The theoretical basis for this case study, including parameter estimation methods, multiresolution analysis and random variable algebra will be used in the compression method presented in the second part. More specifically, a motion based compression system based on the multiresolution 3-d Horn and Schunck optical flow over Probabilistic Volumetric Models is presented in the second part of the thesis. The thesis contributions are listed in detail in Sec. 1.2

1.1 Prior Works

As previously stated, in order to describe a 4-d compression system, the entire context of acquisition, reconstruction/representation, rendering, coding and transmission needs to be taken into account. Among the 4-d representations previously categorized, free-viewpoint video is of notable interest since it allows rendering from arbitrary viewpoints.

Compression techniques in the field of 4-d modeling via free-viewpoint video have been standardized for a particular set of representations, most of them falling into the class of image based modeling with no explicit geometry. The most widely used codec for 2-d video compression is based on the original MPEG standard [72], having many revisions with added functionality over the past two decades. The initial MPEG-4 standard has been augmented with a format for 4-d scenes known as MPEG-4 part 11 Binary Information for Scenes (BIFS) [73]. This format supports the integration of audio, video and 3-d graphics through a hierarchical scene graph. In particular, panoramic views are efficiently encoded through a patch decomposition system where only patches corresponding to the current view are transmitted to the receiver in real time [62]. Omnidirectional video, typically

captured by 11 cameras arranged on the faces of a dodecahedron is able to display a time varying 3-d world projected on a sphere. None of these representations, however, allowed for arbitrary adjustment of views, allowing only certain restricted camera positions.

Later, an extension of BIFS was released called AFX [74] supporting light-field mapping, depth-image based representation (DIBR), 3-d mesh object representation etc. Surface light-field mapping combines image based modeling methods based on ray-space representations with geometric modeling using meshes. Namely, a surface light field assigns a color to each ray originating from a surface, potentially the face of a polygonal mesh. Static objects can be rendered with photorealism using view-dependent texture mapping as seen in [94] [60]. However, generating the view-dependent textures automatically from a set of views requires reliable standardized tools. View acquisition also requires special equipment and preprocessing thus the 4-d representations in AFX are not supported. Instead the depth map 2.5-d representation is used to give the viewer the impression of 3-d depth. Consequently, each view comes with an associated depth map. For the case of stereo-vision where only two views are available, depth is estimated from camera geometry. Compression work has been done in encoding each view and the associated depth map [39] [125] [38][28]. Typically the H.264 codec [67] used in traditional video encoding has reported good results in depth map compression along with the associated view streams. However, the JPEG2000 standard was used to code key-frames in [90] while using depth-propagation [4] to recover depth for each view, reporting better results than just encoding each view and associated depth map separately using H.264. The MPEG committee later formed a group dedicated to researching the standardization of actual free-viewpoint video named 3DAV (3D Audio-Video) [112]. The group focused on modeling methods based on ray-spaces and light-fields. A light field is defined as radiance as a function of direction and position over regions of empty space [82]. There exist also works that perform compression of ray-spaces [84] [85], exploiting both temporal and inter-view redundancies. Note, however, that ray-spaces and light-fields again do not reconstruct geometry but rather provide virtual displays of 3-d space with arbitrary degrees of freedom. Matusik and Pfister describe a 3-d TV system from acquisition, transmission and rendering using light fields in [89]. However, the compression method used involves individual coding of view streams exploiting temporal redundancies using standard H.264 rather than exploiting multi-view dependencies at a given frame set. Although yielding higher compression, exploiting multi-view redundancy is not scalable w.r.t. to the number of views on a set-top-box processor designed for consumer use.

Another compression method in image based modeling is the Multi-View Coding (MVC) extension of the H.264/MPEG-4 AVC standard, first released in March 2009 [129]. Currently, the basic H.264 codec is the most widely used video compression system, reporting a compression rate from 50X to 150X depending on target quality and motion content [135]. H.264 relies on predicting frame segments called macro-blocks using motion vectors and a set of cached reference frames and neighboring frames. Specifically, I-frames or I-slices spatially predict macroblocks (typically 16×16 samples) from neighboring regions within a frame, thus coining the term intra-frame prediction. P-slices support intra-frame coding and inter-frame *predictive* coding, typically encoding one motion vector per macroblock. A reference index is used to select a cached reference frame from which the P-frame macroblock is reconstructed. B-frames support intra-frame coding, inter-frame coding, and additionally, inter-frame *bi-predictive* coding, maintaining two reference indexes and two motion vectors/ prediction signals. The predictive mode of B-frames is similar to that of P-frames except that the decoder now maintains two distinct reference frame lists instead of one. Thus B-frame reconstruction in predictive mode can use reference frames from either the first or the second image list based on a rate-distortion metric. The bi-predictive reconstruction uses a weighted sum of values derived in predictive mode from both reference image lists.

The MVC extension of H.264 allows for predicting views at a given time instance from different views corresponding to the same instance. This way, inter-view redundancy is exploited. A sketch of the MVC system is outlined in Fig. 1.2. Merkle and Smolic [92] demonstrated that the compression gain of the MVC method versus the simultaneous coding of each individual view stream can be as high as 50% with an average of 20%.

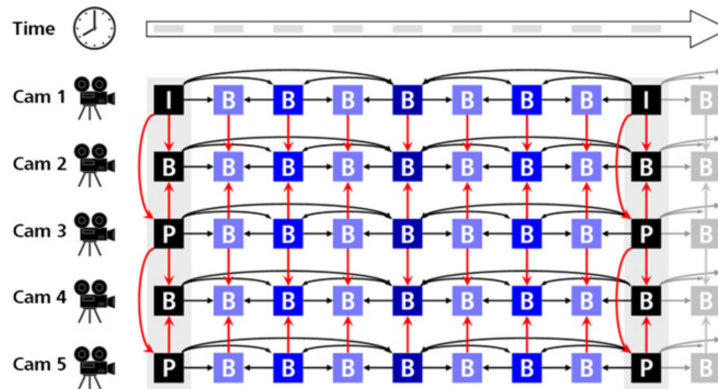


Figure 1.2: Inter-view prediction of the MVC extension of H.264 as described by Merkle and Smolic [92]

So far, image-based modeling methods that do not estimate 3-d scene geometry have been presented. The 3DAV experts group issued a “call for papers” requesting surface geometry reconstruction from imagery techniques. As AFX already supports meshes and more recently point clouds, geometric modeling is expected to be standardized in future releases of the MPEG-4 AFX amendment. However, representing surface geometry through meshes and point clouds comes with disadvantages, which will be outlined in 4. Regardless, reconstructing surfaces rather than virtual views enables a more accurate 4-d world representation, which in turn allows for easier physical world inference in all four dimensions, the main objective of the field of computer vision. Thus, the focus of this thesis will be steered towards 4-d modeling methods that recover surface geometry.

In the past, surface geometry has been typically modeled implicitly from imagery by fitting input images to a known geometric model, recovering motion and pose which yield an approximation to the actual 3-d geometry of the object [122][56] [20][83]. Over time there have appeared numerous works which parametrize complex models of objects typically found in scenes, such as the human body. For example Carranza and Theobalt [27] present a way to parametrize the human body using 16 segments corresponding to various body parts and 17 joints connecting each segment. Segments corresponding to limbs have four degrees of freedom consisting of local positions w.r.t. to the connecting joint and rotations w.r.t. to a local axis. The surface corresponding to each segment is represented by a closed triangular mesh that allows non-rigid deformations in order to accommodate various body types. Silhouettes extracted from input imagery are fitted to the parametrized human model using a pixel-wise energy function. Similar works achieved significant success in modeling and tracking human actors in 4-d, provided that the 3-d human body model representation is accurate [48] [37] [6] [57] [137] [65]. However, it is difficult to obtain or train models for generic scene objects, as well as keeping track of them across a multi-object 4-d scene. Consequently, the scope of this thesis is targeted towards generic scenes, where no geometric modeling is employed during reconstruction.

A popular scene geometry reconstruction technique that does not rely on models known a priori is the visual hull/space-carving method [75] [76]. The algorithm starts by backprojecting the voxels of a coarse bounding volume into each of the input views containing the silhouette of the object. If all backprojections of a particular voxel are encompassed by a silhouette-view, then that particular voxel is marked as being part of the object 3-d geometry. If a voxel’s backprojections are included only in some silhouettes, it will be split, typically into 8 octants as seen in [118]. The process is re-iterated until either a maximum octtree depth is reached or all voxels are marked. Improvements on the visual

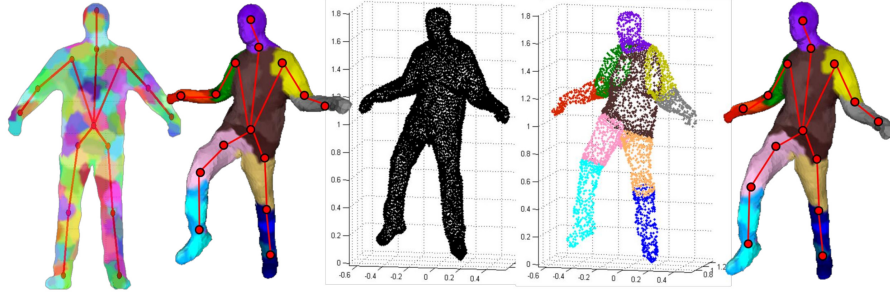


Figure 1.3: The pipeline proposed by Huang and Boyer [65]. Parameters of skeletal joint positions are estimated by fitting a reference human model (represented as a patched mesh and a skeleton) to reconstructed 3-d surfaces from imagery represented as a point cloud. The point cloud is then partitioned and compared against the estimated model at the previous frame, producing the current human model estimate. Image taken from Huang and Boyer [65].

hull method have been demonstrated in [115] [98] [114] [32] where photo-consistency measures across views are used to improve the voxel carving process of the bounding volume. However, this form of reconstruction is somewhat error-prone, especially in the presence of 3-d surface concavities. Bhotika *et al.* [12] gave a stochastic treatment of the visual hull method, representing space occupancy as a distribution rather than a binary in order to deal with scene uncertainty and ambiguity. This treatment is closely related to the theoretical framework used in this thesis as a 4-d representation which will be elaborated in chapter 4.3.

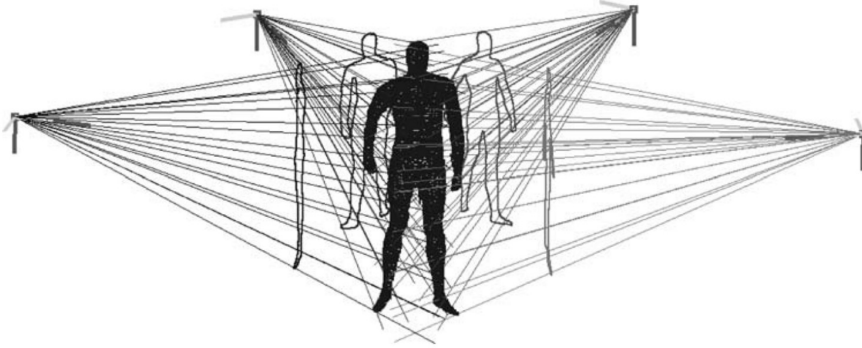


Figure 1.4: Visual Hull as seen in [32]

Thus this thesis will describe a motion-based compression system for a generic scene with no a priori information about the content and minimal modeling assumptions. Camera geometry is assumed to be known and input imagery is synchronized. The 4-d models used are based on Crispel's representation in 3-d [35] and their advantages are discussed in Chapter 4.3.

1.2 Thesis Contributions

The chief contributions of this thesis can be summarized as follows:

- An extension of the JPEG2000 standard that supports the encoding of probabilistic models. To do so, the algebra of random variables is explored, specifically for Gaussian mixture models (GMMs). The sum of two GMMs is derived in closed form. The algebra of random variables is used during the computation of the Discrete Wavelet Transform of Random Variables (DWTRV).
- An algorithm used to reconstruct large aerial scenes based on background modeling and the JPEG2000 encoding of background models. The likelihood of a pixel being background is measured against the compressed background model. If this likelihood falls beneath a confidence threshold, the actual distribution from the full resolution model is retrieved using the local-decompression functionality of JPEG2000. The compression gain for 2-d scenes with sparse motion, a characteristic of large aerial scenes, is as high as 100X.
- An multiresolution 3-d optical flow algorithm based on Horn and Schunck for probabilistic volumetric models (PVMs). Typical scene flow algorithms rely on image correspondences or depth maps to compute scene flow. The embedded multiresolution nature of PVMs and their dense sampling of 3-d space allow for an efficient implementation on a Graphics Processing Unit (GPU). GPUs have become widely popular during the past decade due to their immense parallel computing power and relatively small costs. The output flow field is one of the densest to the best of the author's knowledge.
- A segmentation scheme based on a velocity and position feature vector that decomposes the scene into regions of coherent motion. The regions are approximated by affine transformations allowing for a novel warping technique that is equivalent to finding the inverse of the flow field as a set of affine transform inverses.
- A 4-d compression and reconstruction scheme that uses the optical flow field in its affine parameter representation to predict new time frames. The compression gain per frame can be as high as 120X.

1.3 Thesis Outline

The outline of this thesis can be broken down in two main parts: Chapter 2 and 3 are dedicated to exploring the JPEG2000 standard and 2-d aerial video compression using background modeling, respectively while, Chapter 4, 5 and 6 explores the motion based compression of 4-d models.

Taking a closer look at Chapter 2, the details of wavelet-based multiresolution analysis are first explored in order to better understand the principles of JPEG2000 compression. Next, the embedded-block-coding technique is presented, casting light on JPEG2000's stream and layer organization of the compressed data. In addition, context-adaptive sample coding is described. The JPEG2000 arithmetic coder will be presented last, giving an insight into how binary entropy coding is performed. It's worth noting that these principles, are re-used in other modern compression systems such as H.264, Google's WebP and VP8 image and video compression standards. Chapter 3 reviews a compression system tailored to aerial video based on background modeling and JPEG2000. By modeling a background distribution for every pixel, only moving objects need to be stored for subsequent frames. JPEG2000 allows the caching of a very coarse resolution of the background model as a result of the wavelet transform. In order to compress an array of random variables, the algebra of random variables that are distributed according to a Gaussian mixture model is explored. This technique will also be used in Chapter 5. When decoding a new frame, every pixel is compared against the coarse background distributions. Under some conditions, it is necessary to refine the coarse distributions through the inverse wavelet transform and local decompression of patches, features which make JPEG2000 highly useful in this application.

Chapter 4 first explores the advantage of probabilistic volumetric models versus mesh-based and point-based representations. Next, a theoretical treatment of learning, rendering and implementing the PVM is given, as initially demonstrated by Crispell *et al.* [35]. Chapter 5 presents a novel method of estimating Optical-Flow on probabilistic volumetric models. The classical Horn-and-Schunck optical flow is recast into a 3-d multiresolution framework. The warping step specific to all multiresolution optical flow approaches is based on the algebra of random variables presented in Chapter 3. Rendering the flow in a probabilistic context is demonstrated and compared to the output of 2-d image flow. Chapter 6 presents a method of compressing the flow field by reparametrizing it as a set of affine transformations. A reconstruction technique is given based on warping a scene using the set of affine transformations and the algebra of random variables. The

set of affine transformations is computed using a scene-segmentation process based on a position-velocity feature and a Gaussian mixture model. Using this reconstruction method, the scene can be compressed based on transmitting the tree structure and the re-parametrized flow field. Finally Chapter 7 presents conclusions and future work.

Chapter 2

JPEG2000 as a Model for Large Scale Encoding

Large dynamic scenes such as the ones produced by Wide Area Video Collectors [79] are the most suitable for compression using background modeling. A background model typically estimates the distribution of background at each pixel in a video. In the context of aerial video collection, the background model is stored in the limited on-board memory of the collector which precludes the need of recording every frame. However, the task of encoding a large data structure such as a background model requires a feature rich standard with excellent compression performance. JPEG2000 has many attractive features applicable to large scale images and video including but not limited to bit-error detection and correction, local decompression, Region of Interest (ROI) coding, progressive transmission of data, etc. However, one of the most attractive feature of this standard is the superior quality of encoded images at very low bit rates w.r.t. to other standards. Since lossy compression of data in 2-d and 3-d is one of the objectives of this thesis, JPEG2000 becomes an attractive choice for a background based compression system. This chapter will present the details of the JPEG2000 standard underscoring the theoretical aspects which will be used during the next chapters.

JPEG2000 is developed by the Joint Photographic Experts Group (JPEG) [31] and was designed to supersede the original, highly popular JPEG standard [66]. JPEG2000 comes with many improvements over the original JPEG standard. As mentioned previously, JPEG2000 has the ability to encode images at bitrates in the order of 0.25 bits per sample (bps) without significant loss in



Figure 2.1: JPEG2000 and JPEG compressed images at a compression rate of 1:80. While the artifacts on the JPEG-compressed image (right) are clearly noticeable, one needs to zoom in significantly to notice the ones resulting from the JPEG2000 encoded image (left)

quality. Fig. 2.1 presents an image compressed by a factor of 1:80 with the JPEG and JPEG2000 standard, respectively. While the artifacts from JPEG compression are prominent, the JPEG2000 compressed image shows little loss in quality. The next section describes how the JPEG2000 internal engine avoids the blocking effect present in the JPEG-compressed images at low bitrates. In addition to superior quality vs. compression results, JPEG2000 adds functionality to how an image is decompressed, its paradigm being “compress once, decompress in many ways”. One of these improvements is JPEG2000’s ability to progressively transmit data starting from a very coarse resolution up to the finest available. This transmission mode is very useful in the context of large aerial data, especially in applications where not all high frequency information is needed immediately. An additional feature that JPEG2000 has over its predecessor is random access in the compressed code stream which allows local decompression of spatial regions. Decompressing a rectangular patch from the compressed code stream is desirable when dealing with large images and/or low bandwidth communication channels. These features were implemented by redesigning the internal structure and reformulating the compression engine rather than building upon or improving the original JPEG standard. The following section will present the structure and the underlying mechanisms of the JPEG 2000 standard which will underscore its high applicability to the compression of large data.

2.1 The Discrete Wavelet Transform in JPEG2000

Unlike JPEG's discrete cosine transform (DCT), the core of the JPEG2000 uses the discrete wavelet transform (DWT). The output of the DWT is a series of high-pass and low-pass coefficients obtained after convolving a signal with a high-pass and low-pass filter, respectively. The motivation behind using the wavelet transform in compression systems is that the magnitude of high-pass coefficients is small enough to be encoded with a small number of bits. By recursively applying the DWT R times to the high dynamic range low-pass coefficients, their number will be halved each time, effectively decomposing the original 1-d signal into R sets of high-pass coefficients and a single set of "DC" coefficients. The following section will give an overview of wavelets, their properties in a multiscale context, as well as methods of generating a wavelet basis including the Cohen-Daubechie-Feauveau (CDF) wavelet used in JPEG2000 encoding.

2.1.1 Wavelets and their Filters

Given a family of orthonormal functions defined in Eq. 2.2, the continuous wavelet transform of a signal $x(t)$ using the wavelet basis is defined as

$$W_{\psi}f(2^{-j}, 2^{-j}k) = \frac{1}{\sqrt{2^{-j}}} \int_{-\infty}^{\infty} \psi\left(\frac{2^{-j}x - k}{2^{-j}}\right) \quad (2.1)$$

$$\psi = \{\psi_{jk}|j, k \in \mathbb{Z}, \psi_{jk}(x) = 2^{j/2}\psi(2^jx - k)\} \quad (2.2)$$

The family of functions ψ_{jk} is generated via translating and dilating an original function ψ , also called the "mother" wavelet. In general, the wavelet transform $W_{\psi}f(a, b)$ is a function of the scaling and translation parameters a, b which define how the child wavelets are generated from the mother wavelet. The example above presents the most commonly used wavelet generation, the binary dilation $a = 2^{-j}$ and the binary position $b = 2^{-j}k$.

The study of Multiresolution Analysis of signals [7] demonstrates that the Discrete wavelet transform decomposes a signal $x[n]$ into low-pass and high-pass coefficients by projecting it onto the basis $\{\phi_{jk}; \psi_{jk}\}$ defined as:

$$\phi_{jk}(x) = 2^{j/2}\phi(2^jx - k) \quad (2.3)$$

$$\psi_{jk}(x) = 2^{j/2}\psi(2^jx - k) \quad (2.4)$$

The coefficients $W_\phi[j_0, k]$ and $W_\psi[j, k]$ are expressed as discrete scalar products below:

$$W_\phi[j_0, k] = \frac{1}{\sqrt{M}} \sum_n x[n] \phi_{j_0, k}[n] \quad (2.5)$$

$$W_\psi[j, k] = \frac{1}{\sqrt{M}} \sum_n x[n] \psi_{j, k}[n], j \geq j_0 \quad (2.6)$$

Using these coefficients the signal $x[n]$ can be approximated as:

$$x[n] = \frac{1}{\sqrt{M}} \sum_n W_\phi[j_0, k] \phi_{j_0, k}[n] + \frac{1}{\sqrt{M}} \sum_{j=j_0}^{\infty} \sum_n W_\psi[j, k] \psi_{j, k}[n] \quad (2.7)$$

Defining $V_j = \text{Span}\{\phi_{jk}\}$ and $W_j = \text{Span}\{\psi_{jk}\}$, a requirement of multiresolution function spaces is $V_j \oplus W_j = V_{j+1}$. This means that spaces spanned by lower order scaling functions are included in the spaces spanned by higher order scaling functions. It follows immediately, that one can generically express ϕ and ψ as a linear combination of recursive lower order scaling functions, as such:

$$\phi(x) = \sum_{k=0}^{N-1} h_\phi[k] \sqrt{2} \phi(2x - k) \quad (2.8)$$

$$\psi(x) = \sum_{k=0}^{M-1} h_\psi[k] \sqrt{2} \phi(2x - k) \quad (2.9)$$

Typically, Eq. (2.8) is referred to as refinement, multiresolution or dilation equation. Note that the Eq. (2.9) is a linear combination of recursive scaling functions ϕ , being consistent with the requirement that $V_j \oplus W_j = V_{j+1}$. For example, the multiresolution requirements for the Haar wavelet shown in Fig. 2.2, are defined as :

$$\phi_{0,0}(x) = \phi(x) = \frac{1}{\sqrt{2}} \phi_{1,0}(x) + \frac{1}{\sqrt{2}} \phi_{1,1}(x) \quad (2.10)$$

$$\psi(x) = -\frac{1}{\sqrt{2}} \phi_{1,0}(x) + \frac{1}{\sqrt{2}} \phi_{1,1}(x) \quad (2.11)$$

In order to find the filter impulse responses $h_\phi[n]$ and $h_\psi[n]$ that fit the above requirements, the sum in Eq. (2.8) is expanded for $M = N = 2$ obtaining:

$$\underbrace{\phi(x)}_{\phi_{0,0}(x)} = h_\phi[0] \underbrace{\sqrt{2}\phi(2x)}_{\phi_{1,0}(x)} + h_\phi[1] \underbrace{\sqrt{2}\phi(2x-1)}_{\phi_{1,1}(x)} \quad (2.12)$$

The above equation can be written in terms of the basis functions ϕ_{jk} defined in (2.3). Consequently, by looking closely at the Haar system requirements in Eq. (2.10), it follows that the coefficients of the filter that gives the Haar scaling function are $h_\phi[n] = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$.

Similarly, by expanding Eq. (2.9), and taking into account the Haar requirements from (2.10) it follows that $h_\psi[n] = \left[-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$. Thus the task of representing and generating a wavelet ψ involves finding its scaling function ϕ and the coefficients h_ϕ, h_ψ .

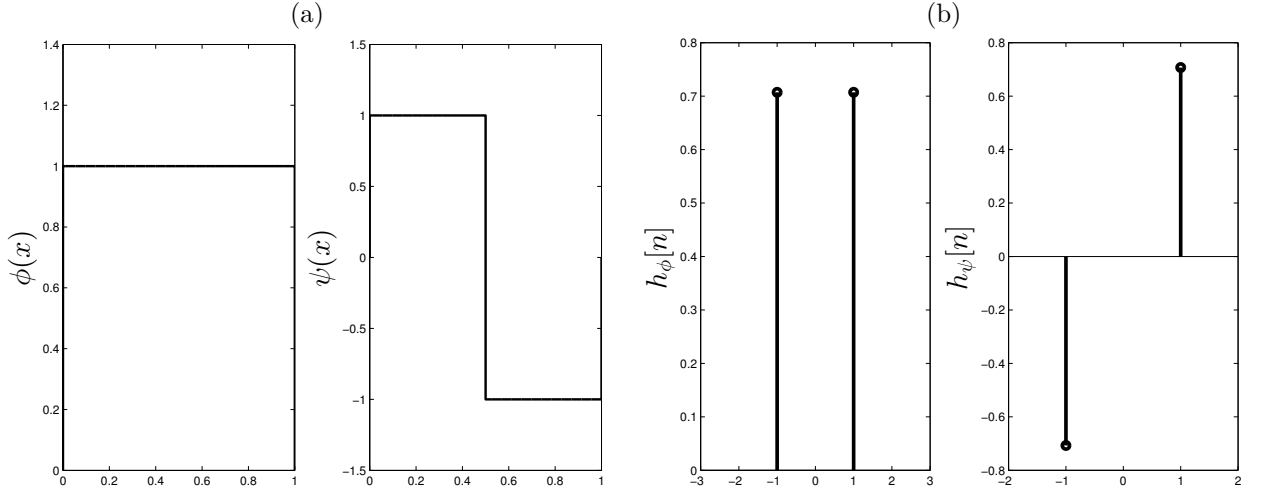


Figure 2.2: (a): Haar wavelet and Scaling function; (b): Haar wavelet Lowpass and Highpass filter impulse responses $h_\phi[n]$ and $h_\psi[n]$, respectively

2.1.2 Determining Daubechie Wavelet-Filter Coefficients

By substituting $x = x/2$ and taking the Fourier transform, Eq. (2.8) and Eq. (2.9) become:

$$\frac{1}{\sqrt{2}}\Phi(2\omega) = H_\omega\Phi(\omega) \quad \frac{1}{\sqrt{2}}\Psi(x) = H_\psi\Phi(\omega) \quad (2.13)$$

By applying the recursive relationship, the Fourier transform of the scaling function is expressed as:

$$\Phi(\omega) = \prod_{p=1}^{\infty} \frac{H_\phi(\omega)}{\sqrt{2}} \Phi(0) \quad (2.14)$$

Thus, the scaling function or father wavelet $\phi(x)$ can be determined, if the frequency domain filter $H_\phi(\omega)$ is known. The mother wavelet ψ is determined immediately as a result of Eq. (2.13). Finding the final solution now involves solving a filter design problem. Daubechie [30] *et al.*,

specified requirements for the set of high-pass and low-pass filters H_ψ and H_ϕ respectively and their implications. For a detailed overview please refer to page 260-280 of [87]. A summary of the filter design specifications is presented below, resulting from bi-orthogonality and linear phase:

$$|H_\phi(\omega)|^2 + |H_\phi(\omega + \pi)|^2 = 2 \quad |H_\psi(\omega)|^2 + |H_\psi(\omega + \pi)|^2 = 2 \quad (2.15)$$

$$H_\psi(\omega)H_\phi^*(\omega) + H_\psi(\omega + \pi)H_\phi^*(\omega + \pi) = 0 \quad (2.16)$$

A trivial result of the above constraints is found below, relating the dual filters,

$$H_\psi(\omega) = -e^{-j\omega}H^*(\omega + \pi) \quad \xleftrightarrow{\mathcal{F}} \quad h_\psi[n] = (-1)^n h_\phi[1 - n] \quad (2.17)$$

Next the first p vanishing moments of a wavelet are introduced as

$$\int_{-\infty}^{\infty} x^k \psi(x) dx = 0, \text{ for } 0 \leq k < p \quad (2.18)$$

This integral diverges for integers $k > p$. It is proved that both $\Psi(\omega)$ and $H_\psi(\omega)$ along with their derivatives have p zeros at $\omega = \pi$. For a detailed proof, please consult Theorem 7.4 of [87]. The latter implies that the Fourier representation of $H_\phi(\omega)$ contains the term $(1 + e^{-j\omega})^p$ in its factorization. Hence,

$$H_\phi(\omega) = \sqrt{2} \left(\frac{1 + e^{-j\omega}}{2} \right)^p R(\omega) \quad (2.19)$$

Taking the magnitude of Eq. (2.19) the following is obtained:

$$\begin{aligned} |H_\phi(\omega)|^2 &= H_\phi(\omega)H_\phi^*(\omega) \\ &= 2 \left(\frac{(1 + e^{-j\omega})(1 + e^{j\omega})}{4} \right)^p |R(\omega)|^2 \\ &= 2 \cos^{2p} \left(\frac{\omega}{2} \right) P \left(\sin^2 \left(\frac{\omega}{2} \right) \right) \end{aligned} \quad (2.20)$$

In the above equation, knowing that $|R(\omega)|^2$ is real and even it can be expressed as a polynomial in $\cos(\omega)$, or equivalently, using the half angle formula, $\sin^2(\omega/2)$. By defining $y \triangleq \sin^2(\omega/2)$ and using the results in Eq. (2.20) the magnitude constraint in Eq. (2.15) becomes:

$$(1 - y)^p P(y) + y^p P(1 - y) = 1 \quad (2.21)$$

2.1.3 Polynomial Factorization of Wavelet-Filter Frequency Response

The Daubechie wavelets use the minimum degree polynomial and Bezout's [58] theorem to determine the form of P .

$$P(y) = \sum_{n=0}^{p-1} \binom{p+n-1}{n} y^n \quad (2.22)$$

The term $|R(\omega)|^2$ can be expressed in terms of $P(\omega)$ as such

$$P(\sin^2(\omega/2)) = P\left(\frac{2 - e^{j\omega} - e^{-j\omega}}{4}\right) = R(\omega)R^*(\omega) \quad (2.23)$$

Depending on how $R(\omega)$ is factorized, different wavelet generating polynomials are obtained. For example, by decomposing R as a product over its roots Eq. (2.23) becomes:

$$P\left(\underbrace{\frac{2 - e^{j\omega} - e^{-j\omega}}{4}}_y\right) = r_0^2 \prod_{k=0}^m (1 - r_k e^{-j\omega}) (1 - r_k e^{j\omega}) \quad (2.24)$$

Equating the left hand side of the above equation to 0, the set of roots $\{r_i, \frac{1}{r_i} \mid 0 \leq i < m\}$ is determined. Daubechie demonstrates that the degree of the polynomial is $m = p - 1$ and that the minimum size filter h has a tap size of $2p$. [30]. This provides a straightforward way of generating the frequency response of $H_\phi(\omega)$ and subsequently $H_\psi(\omega)$ for any p using equation (2.19). Take for example $p = 2$, which yields $P(y) = 1 + 2y$ after using equation (2.22). After substituting $y = (2 - z - z^{-1})$ and solving $P(z) = 0$ the roots are found to be $2 \pm \sqrt{3}$. Thus the closed form of P can be determined using Eq. (2.24) which factorizes over its roots. The wavelets used in JPEG2000 however, factorize as $P = QQ'$. In the case of the previous example for $p=2$, the 5-3 CDF wavelet or Le Gall wavelet is obtained by setting $Q = (1 + 2y)$ and $Q' = 1$. This wavelet is used for lossless image encoding in JPEG2000. The wavelet ψ , scaling function ϕ and the corresponding filter impulse responses are plotted in Fig. 2.3

Once the filter $H_\phi(\omega)$ is known, Eq. (2.14) provides a theoretical way of finding the scaling function $\phi(x)$. In practice, however, the wavelet coefficients of a *discrete* signal, defined in Eq. (2.5), (2.6) are determined using the Fast Wavelet Transform algorithm [86]. The implementation relies on a recursive relationship between discrete wavelet coefficients at different scales j ,

$$W_\phi[j, k] = h_\phi[-2k] \oplus W_\phi[j + 1, 2k] \quad W_\psi[j, k] = h_\psi[-2k] \oplus W_\psi[j + 1, 2k] \quad (2.25)$$

For a detailed proof please consult [86]. By considering the original signal as a J -th order wavelet approximation, *i.e.* $x[n] = W_\phi[J, n]$, the implementation recursively applies the filters $h_\phi[n], h_\psi[n]$ starting at $x[n]$ and downsamples the outputs by a factor of two. JPEG2000 computes the wavelet coefficients using the lifting scheme which is generally faster than traditional convolution [36]. A brief overview of lifting will be given in the following section. Thus the scaling function ϕ is never explicitly computed for the purposes of the DWT. However, it is essential to understand how ϕ and ψ relate to the wavelet-impulse responses h_ϕ and h_ψ and their construction.

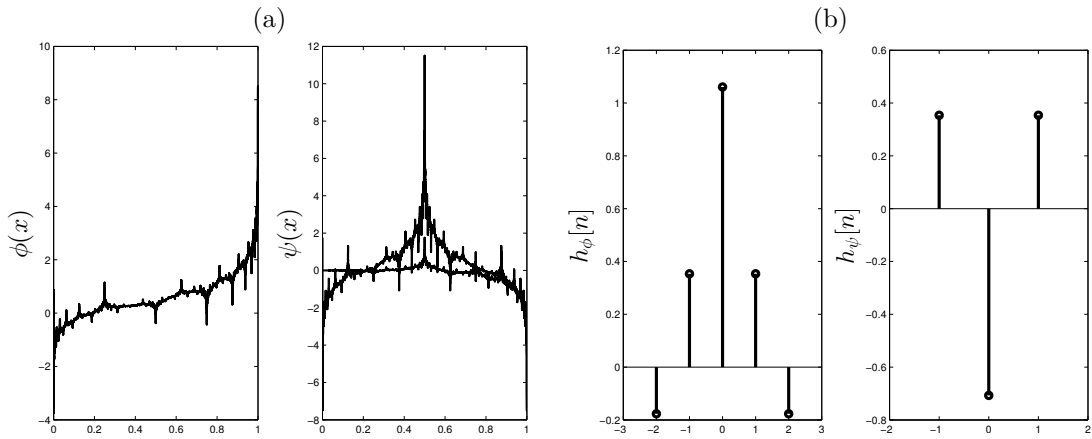


Figure 2.3: (a): The 5-3 Le Gall Scaling function ϕ and wavelet ψ used in JPEG2000 reversible compression; (b): The 5-3 Le Gall Low Pass and High Pass filter impulse responses $h_\phi[n]$ and $h_\psi[n]$, respectively. The number of filter taps in the two impulse responses earned its namesake.

2.1.4 The 2-d Wavelet Transform in JPEG2000

The previous section demonstrated how the wavelet transform decomposes a signal into a series of high-pass and low-pass coefficients. Moreover, the scaling function and the wavelet used as a basis for the DWT need not be computed explicitly. Instead a filter bank h_ϕ, h_ψ consisting of a low-pass filter and a high-pass filter is sufficient. The same principles can be extended in 2-d, however, instead of obtaining two subbands after convolving once with each of the filters, four subbands are obtained. The output subbands at a current resolution is typically labeled as LL, LH, HL and HH. The labeling results from the order in which the filters are applied starting with the rows, *i.e.* subband HL is obtained by first applying the the high-pass filter h_ψ on the rows, followed by the low-pass filter h_ϕ on the columns of the previous output. The transform is applied recursively on the LL band, J times, thus the only LL subband explicitly represented is the LL^0 subband. The latter represents a

coarser representation of the original image, its dimensions reduced by a factor of 2^J . It is obvious now that different resolution levels of the original signal can be obtained by applying the inverse transform starting from the LL^0 subband, *i.e.* to obtain resolution level k , $k < J$ one must apply the inverse wavelet transform iDWT k times to obtain LL^k . The multi-resolution decomposition of the image also avoids the prominent blocking artifacts present in low-bitrate encoding of JPEG images as a result of the tiling of the original image into 8×8 blocks. Instead of traditional convolution, the DWT and iDWT are computed using the lifting scheme [36]. This method computes wavelet coefficients in-place, reducing memory requirements, while exposing a high-degree of parallelism as seen in [77]. During the first stage of lifting, a 1-d signal assumed to be a J^{th} order approximation is split into even and odd coefficients $S^J[n] = X[2n]$ and $D^J[n] = X[2n + 1]$. The second stage evaluates the next level coefficients using the following equations:

$$D^{J-1}[n] = D^J[n] + a(S^J[n] + S^J[n + 1]) \quad (2.26)$$

$$S^{J-1}[n] = S^J[n] + b(D^J[n - 1] + D^J[n]) \quad (2.27)$$

For the Le Gall (5,3) Wavelet $a = -0.5$ and $b = 0.25$. The 2-d lifting simply applies the same equations to the rows and then to the columns of the previous output.

2.2 Data organization and Coding in JPEG2000

In addition to the multiresolution representation of the image, JPEG2000 encodes and orders the wavelet coefficients in a way that enables the features briefly described at the beginning of this chapter. For example, JPEG2000 introduces the concept of layers, allowing the decoding of wavelet coefficients up to a certain numeric precision. However, the specified precision is not global, but varies per sample such that a global convex function of quality distortion versus code length is optimized. To implement such a feature along with the others that were previously mentioned, it is necessary to present the organization of the code stream in JPEG2000.

Before taking the DWT of the image, a multicolor transform is applied such that color components are in the YC_bC_r color space. This transform results in components being minimally correlated and resulting in a higher coding efficiency since components are minimally correlated [121]. After the DWT has been applied for a specified number of levels, the wavelet coefficients are interleaved as

a result of the lifting implementation [36]. The interleaving of samples allows for all subband coefficients corresponding to a certain spatial region to be stored in adjacent memory locations for every resolution. These regions are called *Precincts* and allow for local decompression of data. One can imagine that local decompression from a file stream would be cumbersome if the wavelet coefficients were organized in memory according to the pyramidal representation in Fig 2.4. Each *Precinct* is divided into rectangular regions named *Code Blocks* typically having a nominal width and height of 64 or 128 samples, except at *Precinct* edges. A quantization step is applied to the samples contained within each code-block. For more detail on the quantization and code block specifications in JPEG2000 please consult [121].

Each codeblock is independently processed by the core coding engine of JPEG2000, the Embedded Block Coder with Optimal Truncation (EBCOT). EBCOT is a two-tiered engine, Tier 1 Coding is responsible Bitplane Coding and Entropy Encoding of each bit while Tier 2 handles rate distortion optimization and code stream organization and formation. A diagram of the JPEG2000 data hierarchy and coding flow is presented in Fig. 2.4. An overview of the Tier 1 coding mechanisms that enable most of the features of JPEG2000 is given in the next section. For the purposes of this thesis, the details of Tier 2 are omitted here, since they mostly handle the code-stream organization through tag-trees, packets and various error-correction markers. The interested reader can consult chapter 13 of Taubman’s book [121] for more details on Tier 2 coding.

2.2.1 JPEG2000 Bitplane Coding

The Tier 1 coding stage of EBCOT processes each code block independently, first starting with the task of bit-plane coding. Instead of coding each codeblock sample directly, the bit-plane coder processes the code-block bitplane by bitplane starting from the most significant bit (MSB). Bitplane K of a codeblock is defined as the 2-d array containing the K^{th} bit of each sample in that code block, starting from the MSB. Each bitplane is traversed in raster order, one stripe at a time. The stripe is defined as a column of 4 bits in a given bitplane. A detailed view of bit-plane coding partitioning is presented in Fig. 2.6 The complexity of the bit plane coder is augmented by the introduction of three distinct coding passes. The next section will discuss the coding passes used during JPEG2000 bit plane coding (BPC) and their role in compression. However, before expounding on the concept of passes, the term “coding” in JPEG2000 is first explained.

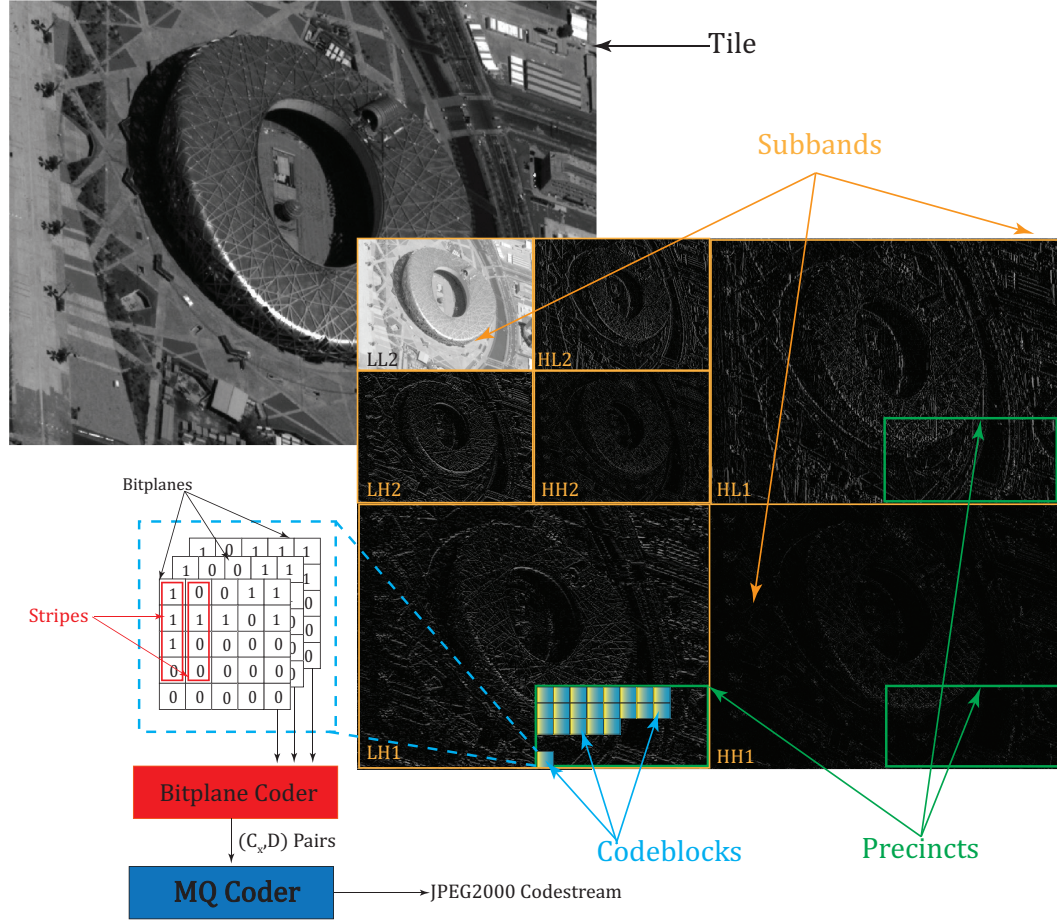


Figure 2.4: JPEG2000 Data Organization and 2-d wavelet transform

JPEG2000 Coding Primitives

JPEG2000 employs four different coding operations or primitives termed “Zero Coding” (ZC), ”Sign Coding”,(SC) “Run Length Coding” (RLC)and “Magnitude Refinement Coding“(MRC). Each of these operations gathers certain state information from the neighborhood of the coded sample. This information is called the *context* and is defined over a 3×3 window centered around that sample. Based on a sample’s context, the coding operation produces an output and potentially changes state information. The specific state information that the bit plane coder of JPEG2000 keeps track of is stored in three different 2-d arrays labeled σ, σ' and η , each having the same dimension as the current code block. In addition to the state arrays, a sign array χ and the original sample array v are maintained. The sign array indicates whether a sample is negative, *i.e.* $\chi_{i,j} = 1$ if $v_{i,j} < 0$. The

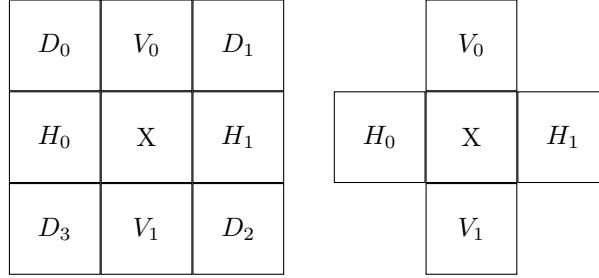


Figure 2.5: Context Window (left) and Sign Context Window (right)

state arrays have the following properties:

- The array σ is called the significance array. $\sigma_{i,j} = 1$ if the first nonzero bit of sample $v_{i,j}$ has been coded. Otherwise, $\sigma_{i,j} = 0$. Samples that have $\sigma = 1$ are considered “significant” samples. The values in σ are conserved after each bitplane encoding up to and including the last.
- The array σ' represents the refinement state array. $\sigma'_{i,j} = 1$ if any of the bits of sample $v_{i,j}$ were coded using the Magnitude Refinement Coding (MRC) operation. Similar to the σ state array, σ' maintains its values until all bit-planes in the current code block are encoded.
- The array η represents the zero state array. $\eta_{i,j} = 1$ if a Zero Coding (ZC) operation has been applied to sample $v_{i,j}$. Unlike the previous two state arrays, η is reset after each bitplane encoding.

Each coding operation analyzes the context of a sample via the 3×3 context window and produces a tuple containing that sample’s context Cx and a decision bit D . The context is an integer in the range $[0, 18]$ and is a function of the σ state bits contained in the 3×3 context window shown in Fig. 2.5. Below are illustrated the specifics of each coding operation. As previously mentioned, these operations are performed on bit samples in a given bit-plane.

- *Zero Coding (ZC)*. This operation determines the context of a sample according to the rules in Tables 2.1, 2.2, 2.3. Each table has different context formation rules based on the subband that sample lies in. For example, the second row of table 2.1 indicates that if a sample has more than one significant vertical neighbors, exactly one significant horizontal neighbor, its context is 7. The X’s in the tables stand for “don’t care(s)”. The decision value generated by

LL and LH Subbands			Context
$\sum_n H_n$	$\sum_n V_n$	$\sum_n D_n$	Cx
2	x	x	8
1	≥ 1	x	7
1	0	≥ 1	6
1	0	0	5
0	2	x	4
0	1	x	3
0	1	≥ 2	2
0	0	1	1
0	0	0	0

Table 2.1: Zero Coding - LL and LH

HL Subbands			Context
$\sum_n H_n$	$\sum_n V_n$	$\sum_n D_n$	Cx
x	2	x	8
≥ 1	1	x	7
0	1	≥ 1	6
0	1	0	5
2	0	x	4
1	0	x	3
0	1	≥ 2	2
0	0	1	1
0	0	0	0

Table 2.2: Zero Coding - HL

the ZC operation is equal to the sample's bit in the current bit-plane. The ZC operation will mark an insignificant sample as significant *i.e.* set $\sigma_{i,j} = 1$ if the sample's first non-zero bit is encountered. Thus the ZC operation may change the significance state information.

- *Run Length Coding*(RLC) This operation codes four consecutive samples in a column, *i.e.* an entire stripe if none of them become significant in the current pass. Run mode assumes that the run length is four samples. However, if a sample becomes significant the run is interrupted and the length of the run is also coded. Samples coded using RLC are assumed to contribute little to image quality thus they are essentially packed into a single (Cx, D) pair.

HH Subbands		Context
$\sum_n (H_n + V_n)$	$\sum D_n$	Cx
x	≥ 3	8
≥ 1	2	7
0	2	6
≥ 2	1	5
1	1	4
0	1	3
≥ 2	1	2
0	1	1
0	0	0

Table 2.3: Zero Coding - HH

$\sigma'_{i,j}$	$\sum_n (H_n + D_n)$	Cx
1	x	16
0	≥ 1	15
0	0	14

Table 2.4: Magnitude Refinement Coding Rules

- *Magnitude Refinement Coding* (MRC). Unlike the previous operation, the state array used by this operation is σ' instead of σ . More specifically, the context is generated based on how many neighbors in the context window have had a Magnitude Refinement Operation performed. The rules are presented in Table 2.4. For example, if the current sample had experienced an MRC

operation during a previous bitplane encoding, the generated context for that sample is 16. Similarly, if the sample didn't experience the MRC operation yet, but has at least one diagonal or horizontal neighbor that did, its context is 15. During the description of each coding-pass it will be revealed that the MRC operation is only performed on samples that are already significant, *i.e.* the first non-zero bit of that sample has been coded in a previous bitplane. Hence the name magnitude *refinement*. The decision bit accompanying the generated context is again equal to the sample's bit at the current bit-plane.

- *Sign Coding* (SC). Sign coding uses the horizontal and vertical neighbors presented in the Sign Coding Window in Fig. 2.5. First, the following quantities are defined,

$$H_{sc} = \min(1, \max(-1, \sigma_{i,j-1} \times (1 - 2\chi_{i,j-1}) + \sigma_{i,j+1} \times (1 - 2\chi_{i,j+1}))) \quad (2.28)$$

$$V_{sc} = \min(1, \max(-1, \sigma_{i-1,j} \times (1 - 2\chi_{i-1,j}) + \sigma_{i+1,j} \times (1 - 2\chi_{i+1,j}))) \quad (2.29)$$

After their evaluation, Table 2.5 is used to determine the context generated by the SC operation. The decision value is equal to

H_{sc}	V_{sc}	$\overline{\chi_{ij}}$	Cx
1	1	1	13
1	0	0	12
1	-1	0	11
0	1	0	10
0	0	0	9
0	-1	1	10
-1	1	1	11
-1	0	1	12
-1	-1	1	13

Table 2.5: Sign Coding Rules

Bitplane coding Passes

The JPEG2000 coder traverses each bitplane three times until all the bits in a given bitplane are coded. It is important to note that each bit will be coded once only during one of the three passes. This scheme is based on the fractional bitplane coding scheme proposed by David Taubman in [121]. It can be intuitively thought of partitioning a given bitplane into three disjoint bit lists. Each list has an associated score equal to the amount of image quality lost when the bits it contains are

truncated from the corresponding samples. Thus the fractional bit-plane coding scheme gives the user more flexibility in lossy coding. Instead of truncating *all* samples at a given precision K , the fractional bit-plane passes or bit-lists allow for only *some* samples to have their K^{th} bit truncated. Since the bit-lists or passes have an associated distortion score, the truncation decision at a sample is given by the pass affiliation of its K^{th} bit. Consequently, the concept of truncation in JPEG2000 is recast from truncating the codestream at a precision K to truncating after pass p of bitplane K . The coding passes will lay the foundation to JPEG2000 quality layers. To better illustrate how the passes can essentially partition the bitplane into bit lists ranked according to their contribution to image quality, an overview is given below:

- **Significance Propagation Pass (SPP).** During the first pass of bitplane processing, bits will be coded if they are insignificant *i.e.* $\sigma_{i,j} = 0$ but have at least one significant neighbor in their context window. This condition was motivated by the fact that insignificant bits with significant neighbors are likely to become significant during the coding of the current bitplane. It has been demonstrated that the bits coded during this pass are likely to yield the largest distortion reduction relative to code length [95]. The coding operations that are applied to bit samples in this pass are ZC, and SC. As mentioned during the description of ZC, if a sample's first non-zero bit is encountered, it will become significant. In addition, any sample coded in this pass will have its corresponding zero state array updated, *i.e.* $\eta_{i,j} = 1$. The SC primitive is invoked as soon as a sample becomes significant. One can realize that, according to these rules, this pass is likely to chain-toggle the significance of samples with similar magnitudes when traversing the bitplane in raster order.
- **Magnitude Refinement Pass (MRP).** The second pass handles bits that have already become significant in a *previous* bitplane. The only coding operation used is MRC. This should be evident, as SC is applied only when a sample becomes significant.
- **Cleanup Pass (CUP).** As the name suggests, this pass codes all the bits that were not coded in the previous two passes, *i.e.* bits that have $\eta_{i,j} = 0$ and $\sigma_{i,j} = 0$. Note that, the zero-state array η indicates whether a bit had experienced a zero coding operation during the SPP, as a consequence of being a neighbor of a significant sample. As a result, it should be obvious that these bits correspond to insignificant samples. The ZC, RLC and SC primitives are used in this pass. Note that it is possible for a sample to become significant during this pass if its first

non-zero bit is encountered. Evidently, the bits coded during the cleanup pass will contribute less to image quality, thus their distortion-reduction score relative to length is low.

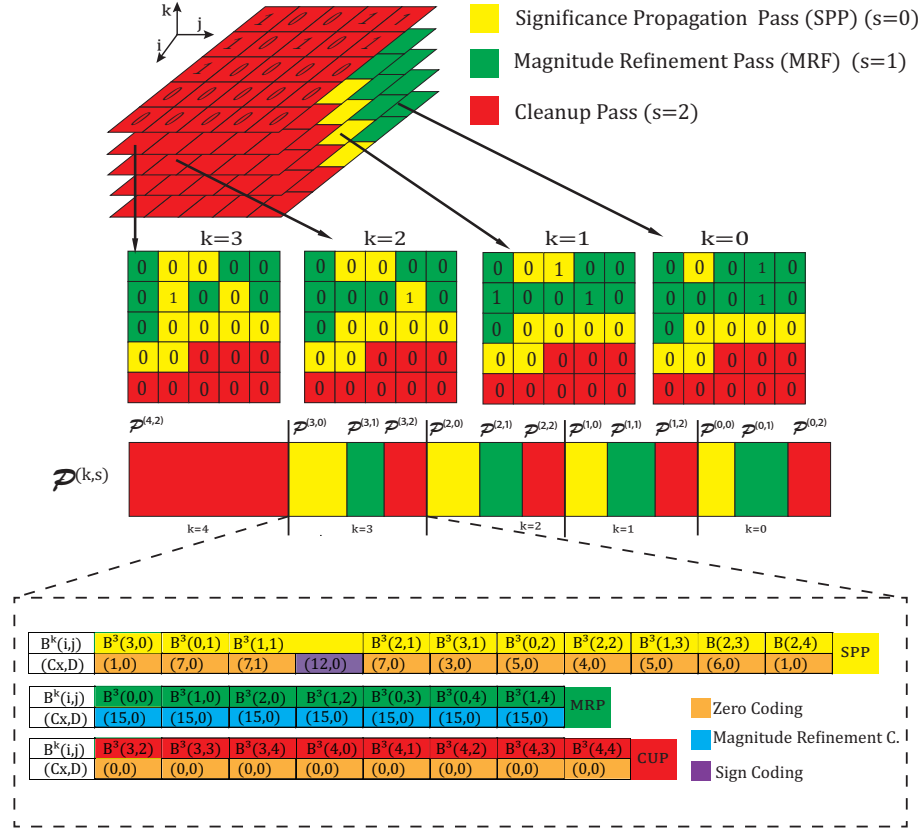


Figure 2.6: JPEG2000 Coding Pass Example. The second row represents an abstraction of the bit-lists $\mathcal{P}^{k,s}$ corresponding to each coding pass. Note how the length $\mathcal{P}^{k,s}$ varies across bitplanes k and passes s . The bottom part shows a detailed view of the (Cx, D) pairs resulting from encoding the first bitplane

From Coding Passes to Quality Layers

It is worth mentioning that the bits in the first bitplane will all be coded during the CUP. This is because no sample has a significant neighbor yet (σ is initialized with 0s), thus the rules for the SPP are not met. The same goes for the MRP, which requires a sample to become significant during a *previous* bitplane coding. Thus, for codeblock samples having a maximum of B bits of precision, there are $3B - 2$ coding passes, as the first two passes are completely omitted during the first bitplane coding. Thus on a Distortion vs Length plot, there will be a total of $3B - 2$ discrete points. By contrast, without the coding passes, the same plot will only have B points, evenly spaced along the

Length axis. When attempting to optimize the Distortion vs Length plot, it is desirable to have a denser sampling of points, which is exactly what the coding passes provide. For a more detailed treatment of Rate-Distortion Optimization in JPEG2000 please consult section 8.3.5 of [121]. The output of the optimization will yield a truncation point L from one of the $3B - 2$ total coding passes. Thus JPEG2000 uses the coding passes as *quality layers* and will optimize to find the layer that is closest to the target bit-rate.

An added benefit of coding passes other than layer formation and added truncation flexibility is increased entropy coding performance which will be discussed in the next section. The example in Fig. 2.6 shows the quality layers/coding passes resulting from the fractional bitplane coding technique applied on a 5×5 abstract codeblock.

2.2.2 Entropy Coding in JPEG2000

Sec. 2.2.1 described how each coding operation produces a (Cx, D) tuple containing the context of the current bit sample and the a decision value. These values are sent to the entropy coder to produce a compressed representation of the codestream.

Entropy coding in JPEG2000 is based on legacy arithmetic coding and is performed by the MQ coder [18]. This binary coder creates a sequence of nested intervals based on a sample's probability. The coder keeps track of two symbols, the More Probable Symbol (MPS) and the Less Probable Symbol (LPS). As decision bits are fed to the MQ coder, they are compared against the value of the current MPS, which typically switches from 0 to 1 many times during encoding. The current estimate of the probability of encountering a LPS is labeled Q_e . The interval A that the MQ coder starts off with is in the range $[0, 1)$ and is parametrized by the base of the interval b_i and its length l_i . The rules of subdivision are given below:

$$\text{if } v_i = MPS \begin{cases} l_i = A \times (1 - Q_e) \\ b_i = b_i + A \times Q_e \end{cases} \quad \text{if } v_i = LPS \begin{cases} l_i = A \times Q_e \\ b_i = b_{i-1} \end{cases} \quad (2.30)$$

Thus encountering an MPS frequently will cause the interval to shrink less, thus requiring less bits to represent. Encountering an LPS would reduce A by a factor of $Q_e < 0.5$ every time, requiring more precision bits to represent. The MQ coder was designed to avoid the multiplication operation.

By keeping the range of A around 1, the following approximation is used:

$$\text{if } A \approx 1 \begin{cases} A \times (1 - Q_e) \approx A - Q_e \text{ interval corresponding to MPS} \\ A \times Q_e \approx Q_e \text{ interval corresponding to LPS} \end{cases} \quad (2.31)$$

The probability estimation for Q_e is hard-coded using a lookup table built from a Markov model proposed by Langdon *et al.* [18] consisting of a Finite State Machine with 46 states, presented in Table 2.6. The Contexts from the (Cx, D) tuple govern the rules of state transition, as well as the current MPS. Thus the MQ coder decides whether to encode a sample decision D by checking whether $D = MPS(Cx)$ and changes the next state according to the current context Cx . Thus the MQ coder is a context adaptive encoder. Each state contains the current estimate of Q_e and a flag indicating whether the current MPS switches with the current LPS. In addition to the state dependent role reversal, The MPS and LPS exchange roles when the interval adjustment causes the segment corresponding to the MPS to be smaller than the one corresponding to the LPS, *i.e.* $A - Q_e < Q_e$. A renormalization of A occurs in this case and the roles of MPS and LPS are again switched.

Since there are a total of 19 different contexts, specific only to certain coding operations and passes, the probability estimates for Q_e are designed in a way to maximize coding efficiency.

2.3 Conclusion

This chapter demonstrated the internals of JPEG2000 and their relevance to the encoding of large data files. The discrete wavelet transform stage was described in detail as it relates to the multi-resolution analysis which stands at the basis of coarse-to-fine optical flow. The next chapter will extend the highly useful JPEG2000 standard to compressing a background models. These models are used in the segmentation and compression of large 2-d scenes.

I	Qe Value			NMPS	NLPS	SWITCH
	(hexa)	(binary)	(decimal)			
0	0x5601	0101 0110 0000 0001	0.503 937	1	1	1
1	0x3401	0011 0100 0000 0001	0.304 715	2	6	0
2	0x1801	0001 1000 0000 0001	0.140 650	3	9	0
3	0x0AC1	0000 1010 1100 0001	0.063 012	4	12	0
4	0x0521	0000 0101 0010 0001	0.030 053	5	29	0
5	0x0221	0000 0010 0010 0001	0.012 474	38	33	0
6	0x5601	0101 0110 0000 0001	0.503 937	7	6	1
7	0x5401	0101 0100 0000 0001	0.492 218	8	14	0
8	0x4801	0100 1000 0000 0001	0.421 904	9	14	0
9	0x3801	0011 1000 0000 0001	0.328 153	10	14	0
10	0x3001	0011 0000 0000 0001	0.281 277	11	17	0
11	0x2401	0010 0100 0000 0001	0.210 964	12	18	0
12	0x1C01	0001 1100 0000 0001	0.164 088	13	20	0
13	0x1601	0001 0110 0000 0001	0.128 931	29	21	0
14	0x5601	0101 0110 0000 0001	0.503 937	15	14	1
15	0x5401	0101 0100 0000 0001	0.492 218	16	14	0
16	0x5101	0101 0001 0000 0001	0.474 640	17	15	0
17	0x4801	0100 1000 0000 0001	0.421 904	18	16	0
18	0x3801	0011 1000 0000 0001	0.328 153	19	17	0
19	0x3401	0011 0100 0000 0001	0.304 715	20	18	0
20	0x3001	0011 0000 0000 0001	0.281 277	21	19	0
21	0x2801	0010 1000 0000 0001	0.234 401	22	19	0
22	0x2401	0010 0100 0000 0001	0.210 964	23	20	0
23	0x2201	0010 0010 0000 0001	0.199 245	24	21	0
24	0x1C01	0001 1100 0000 0001	0.164 088	25	22	0
25	0x1801	0001 1000 0000 0001	0.140 650	26	23	0
26	0x1601	0001 0110 0000 0001	0.128 931	27	24	0
27	0x1401	0001 0100 0000 0001	0.117 212	28	25	0
28	0x1201	0001 0010 0000 0001	0.105 493	29	26	0
29	0x1101	0001 0001 0000 0001	0.099 634	30	27	0
30	0x0AC1	0000 1010 1100 0001	0.063 012	31	28	0
31	0x09C1	0000 1001 1100 0001	0.057 153	32	29	0
32	0x08A1	0000 1000 1010 0001	0.050 561	33	30	0
33	0x0521	0000 0101 0010 0001	0.030 053	34	31	0
34	0x0441	0000 0100 0100 0001	0.024 926	35	32	0
35	0x02A1	0000 0010 1010 0001	0.015 404	36	33	0
36	0x0221	0000 0010 0010 0001	0.012 474	37	34	0
37	0x0141	0000 0001 0100 0001	0.007 347	38	35	0
38	0x0111	0000 0001 0001 0001	0.006 249	39	36	0
39	0x0085	0000 0000 1000 0101	0.003 044	40	37	0
40	0x0049	0000 0000 0100 1001	0.001 671	41	38	0
b41	0x0025	0000 0000 0010 0101	0.000 847	42	39	0
42	0x0015	0000 0000 0001 0101	0.000 481	43	40	0
43	0x0009	0000 0000 0000 1001	0.000 206	44	41	0
44	0x0005	0000 0000 0000 0101	0.000 114	45	42	0
45	0x0001	0000 0000 0000 0001	0.000 023	45	43	0
46	0x5601	0101 0110 0000 0001	0.503 937	46	46	0

Table 2.6: The state machine table of the MQ coder [68]. The state index I is a function of the 19 different coding contexts *i.e.* $I(Cx)$. The table keeps track of the index of the next state based on whether an LPS or MPS was coded. For example if, an LPS is coded, the next state is $NLPS(I(Cx))$. Additionally, each state contains the current estimate of the LPS probability Q_e .

Chapter 3

High Resolution Video

Compression Using Background

Modeling and JPEG2000

Before proceeding to motion based compression in 4-d, the 2-d compression of Aerial Video is explored as a case study in this chapter, offering insights into the design of a compression system in higher dimensions. Recent development of wide area aerial video collectors (WAVC) that acquire 1.5 Gpixel images at ten frames per second [79] imposes novel challenges for compression and transmission of the video data. Acquisition and manipulation of wide area aerial surveillance video is a challenging task due to limited on-board storage and bandwidth available for transferring video to the ground. A collection mission of two hours produces 350 TeraBytes of data and a bandwidth of 50 Giga Bytes/sec to record a three-channel video at 10 frames per second. These high bandwidth processing and storage requirements warrant the need for an efficient compression scheme.

The current approach to managing WAVC data is to encode the video with JPEG2000 on a frame-by-frame basis using multiple Analog Devices ADV212 chips, operating on sections of the video frame in parallel. However, with lossless compression this approach results in only a 3:1 compression ratio and cannot achieve the required frame rate. Applying higher compression ratios is not feasible since the loss of fidelity for small moving objects significantly reduces the performance of automated

algorithms, such as video processing.

The overall objective of this chapter is to describe an approach to the compression of high resolution surveillance video using a background model that tolerates frequent variations in intensity and also apparent intensity change due to frame mis-registration. Since total pixel area of moving objects in a scene is relatively small, an approach based on selectively encoding moving objects in each frame and only transmitting a full frame occasionally is likely to produce a high compression factor. The success of this strategy depends on the ability to accurately detect foreground. It is proposed to use a background model based on a mixture of Gaussians (GMM), where the model is compressed using JPEG2000. This approach leads to an efficient foreground detection algorithm and a model that is relatively inexpensive to compute and store.

Alternative strategies such as Motion JPEG and MPEG-4 Part 10/AVC (also known as H264) [67] video compression standards are not practical in this application. Both methods require the memory storage of past frames, especially in the case of H-264 which uses up to 16 bi-predictive frames in motion estimation as well as multiple motion vectors for each block which point to different reference frames. These reference frames would have to be stored in high-speed memory, which is very limited and largely occupied with the formation of video frames, e.g. Bayer color restoration.

Several implementations of video compression based on background-foreground segmentation exist [9] [107] but none suggest a practical solution for the case of ultra-high resolution aerial video. Moreover, pixel-based background models which are less computationally demanding than block-based models require very large memory. For example, the robust pixel-based background modeling scheme proposed by C. Stauffer and W. Grimson [116] uses a mixture of weighted normal distributions at each pixel. Consequently, for a 3-channel video a model with three mixture components at every pixel requires 21 floating point numbers per pixel, or a storage of over 130 GBytes per frame. W.R. Schwartz and H. Pedrini [107], extend the motion estimation approach of Babu on foreground objects by projecting intra-frame blocks on an eigenspace computed using PCA over a set of consecutive frames, thus exploiting the spatial redundancy of adjacent blocks. The cost of estimating the PCA basis as well as the requirement of observing foreground-free frames during the estimation process renders this approach unsuitable.

3.1 Surveillance Video Compression

In the approach to be described, foreground pixels are detected using a Gaussian mixture model (GMM), which provides rapid adaptation to changing imaging conditions as well as a probabilistic framework. Since a GMM is stored at each pixel, the storage requirement would be prohibitive without some strategy for model compression. In the following, a technique for significant model data reduction without loss in detection accuracy is described. The description starts with a review of the GMM background model.

3.1.1 Background Modeling

The extensive literature on background modeling methods can be assigned to two major categories. The first one exploits temporal redundancy between frames by applying a statistical model on each pixel. Model parameters are estimated either on-line recursively or off-line using maximum likelihood. Although the normal distribution seems sound and inexpensive at first, it cannot cope with wide variations of intensity values such as reflective surfaces, leaf motion, weather conditions or outdoor illumination changes. A natural improvement is to use a mixture of weighted normal distributions (GMMs), a widely used appearance model for background and foreground modeling. However, the amount of storage required to maintain a GMM at each pixel is impractically large for the WAVC application. In order for the GMM representation to be effective, the storage requirement must be reduced by at least an order of magnitude. This section presents an innovative approach to the compression of such models in order to detect moving objects in very large video frames. Before presenting the new compression method, a survey of the GMM background modeling approach is provided as background. Without compression, such models would require an impractically large amount of storage.

Friedman and Russell successfully implemented a GMM background model over a traffic video sequence, each parameter being estimated using the general Expectation-Maximization algorithm [43]. However, the most popular pixel-based modeling scheme is that implemented by Stauffer and Grimson [116], which uses a fast on-line K-means approximation of the mixture parameters. Several variations of this method were developed improving parameter convergence rate and overall robustness [78][141].

The second category of background models analyzes features from neighboring blocks thus exploiting

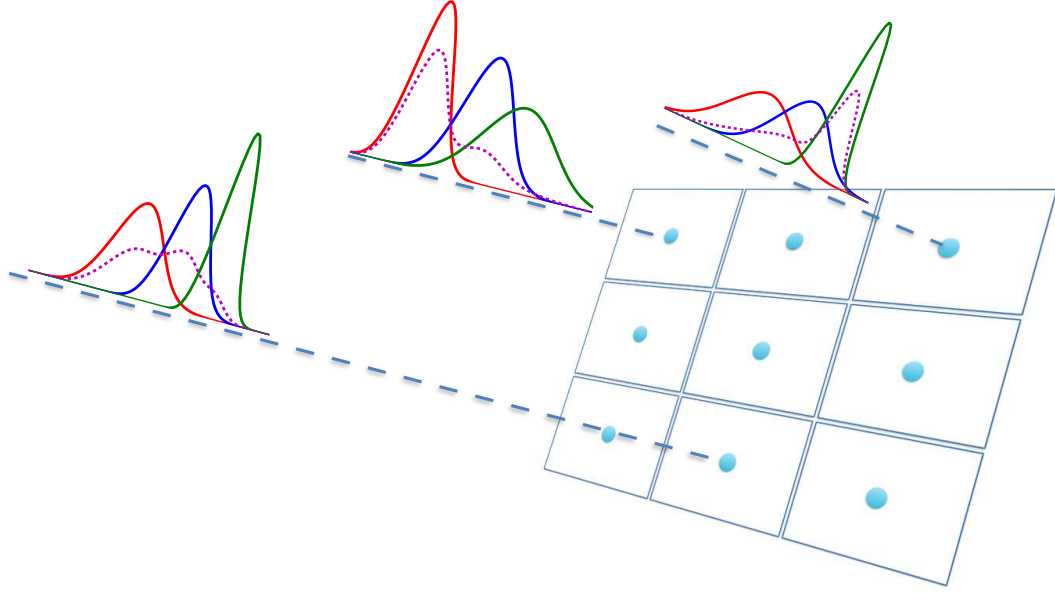


Figure 3.1: Background model maintaining the distribution of background as a mixture of Gaussians at each pixel

spatial redundancy within frames. Although Heikkilä, and Pietikäinen [59] implemented an operator that successfully depicts background statistics through a binary pattern, the relatively high computational cost prevent its use in this application. W.R. Schwartz and H. Pedrini [107], propose a method in which intra-frame blocks are projected on an eigenspace computed using PCA over a set of consecutive frames, thus exploiting the spatial redundancy of adjacent blocks. The cost of estimating the PCA basis as well as the requirement of observing foreground-free frames during the estimation process renders this approach unsuitable. The same reason makes other block-based methods that capture histogram, edge, intensity [69][70] and other feature informations unsuitable for high resolution surveillance video.

In the proposed approach, the background model is based on a fast-converging extension of the Stauffer and Grimson approximation presented by Dar-Shyang Lee [78] to model background. The extension of Lee is explained by starting with a summary of the basic Stauffer and Grimson algorithm. The value of each pixel is described by a mixture of normal distributions, as seen in Fig.

3.1. Thus, the probability of observing a particular color tuple $\mathbf{X}$ at time t is given by

$$\mathbf{Pr}(\mathbf{X}_t) = \sum_{i=0}^{K-1} \omega_{i,t} \cdot \mathcal{N}(\mathbf{X}_t, \boldsymbol{\mu}_{i,t}, \Sigma_{i,t}) \quad (3.1)$$

K is the number of distributions in the mixture (typically 3 to 5) and $\omega_{i,t}$ represents the weight of distribution i at time t . Each distribution in the mixture (also referred to as mixture component) is normal with *Pdf*:

$$\mathcal{N}(\mathbf{X}_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (\mathbf{X}_t - \mu)^T \Sigma^{-1} (\mathbf{X}_t - \mu) \right) \quad (3.2)$$

The proposed method checks to see if a new incoming pixel color tuple $\mathbf{X}_{t+1}$ is within a factor f (typically 2.5) standard deviations from a normal distribution in the mixture. If no match is found the least weighted component is discarded in favor of a new one with mean $\mathbf{X}_{t+1}$ and a high variance. The weights change according to:

$$\omega_{i,t+1} = (1 - \alpha) \omega_{i,t} + \alpha \cdot M_{i,t} \quad (3.3)$$

The value of $M_{i,t}$ is 1 for the distribution with the closest match (if more than one distribution matches, the one with the highest match ratio (*i.e.* $\omega_{i,t}/|\Sigma_i|$) is chosen and 0 for the rest of the distributions. The learning rate α represents how fast should the new weight change when a match is found. Each component i in the mixture will be updated as follows:

$$\boldsymbol{\mu}_{t+1,i} = (1 - \rho_{t,i}) \boldsymbol{\mu}_{t,i} + \rho_{t,i} \mathbf{X}_t \quad (3.4)$$

$$\Sigma_{t+1} = (1 - \rho_{t,i}) \Sigma_{t,i} + \rho_{t,i} (\mathbf{X}_t - \boldsymbol{\mu}_t)^T (\mathbf{X}_t - \boldsymbol{\mu}_t) \quad (3.5)$$

Essentially, ρ is the probability of observing the tuple X_t given the mixture component i scaled by the learning rate.

$$\rho_{i,t} = \alpha \mathbf{Pr}(\mathbf{X}_t | i, \theta_{i,t}) = \alpha \mathcal{N}(\mathbf{X}_t, \boldsymbol{\mu}_{i,t}, \Sigma_{i,t}) \quad (3.6)$$

The parameter α causes many inaccuracies in various applications since a small value leads to slow convergence and a large value will make the model sensitive to rapid intensity variations. This problem is addressed by Lee's implementation in which each mixture component i has its own

adaptive learning rate which is a function of a global parameter α and a match count $c_{i,t}$ (*i.e.* the number of times component i was a match up until the current time t). Let $q_{i,t}$ be 1 if component i is the closest match at time t and 0 otherwise. The weight is updated as follows:

$$\omega_{i,t+1} = (1 - \alpha)\omega_{i,t} + \alpha q_{i,t} \quad (3.7)$$

The key difference from the Stauffer and Grimson algorithm is the following update equation,

$$\rho_{i,t} = q_{i,t} \alpha \left(\frac{1 - \alpha}{c_{i,t}} + \alpha \right) \quad (3.8)$$

Since each component maintains a history of observations, the convergence rate of the true background distribution can be achieved much faster while maintaining robustness in the early stages of learning. The background model for video frames of dimension $w \times h$ at time t can be regarded as an image of random variables

$$\Theta = \left\{ \mathbf{X}_t^{ij} \mid i < w, j < h, \mathbf{X}_t^{i,j} \sim \mathcal{M}(\omega_t^r, \boldsymbol{\mu}_t^r, \Sigma_t^r) \right\} \quad (3.9)$$

The sample space for each pixel, X_t^{ij} is the set of all possible color tuples (e.g. all 8-bit RGB value combinations) and the probability function is the mixture of normal distributions $\mathcal{M}(\omega_t^r, \boldsymbol{\mu}_t^r, \Sigma_t^r)$. Storing Θ losslessly requires a large memory space is not a practical solution. A highly compressed representation of Θ will make implementations tractable but with the risk of inaccurate classification of foreground objects. As will be seen, JPEG2000 provides an effective compression scheme, since regions that are detected to contain foreground based on a highly compressed model can be refined locally without decompressing the entire model, and thus obtain the accuracy of the original background model.

3.1.2 The JPEG2000 standard

As seen in Sec. 2.1 JPEG2000 applies a transform (DWT) to the image and then truncates the bit resolution of the wavelet coefficients. The coefficients are then encoded using image neighborhood context analysis followed by entropy coding. In the case of large single frames, JPEG2000 has better compression quality, compared to other coding schemes such as JPEG or H264. The standard also supports the concept of levels, where quality can be flexibly balanced with compression ratio.

Additionally the hierarchical nature of the DWT intrinsically provides an image pyramid, which is useful for visualizing large images.

A discrete wavelet transform (DWT) decomposes a signal into low and high frequency coefficients. A single level of the 2-d transform divides the image in four high and low frequency subbands along each direction (e.g. the HL subband emphasizes the high frequencies in the horizontal direction and low frequencies in the vertical direction). The subband that contains low frequencies in both horizontal and vertical directions (LL) represents a low-pass filtered and downsampled representation of the original image. A recursive application of the transform on the LL band yields a pyramid with multiple levels of decomposition of the original image. The subband size in each level is one fourth the size of corresponding one from the level before.

The effective tiered decomposition of the original image in JPEG2000 permits its decompression at various intermediate resolutions before reaching the original image resolution. Once the wavelet domain is computed via the lifting scheme with the Daubechies 9/7 or 5/3 wavelet filters, the coefficients are quantized and entropy coded. To further achieve scalability, the reader is reminded that JPEG2000 introduces the concept of coding passes when sending wavelet coefficients' bits to the entropy encoder. Instead of using a raster-scan order to code the n^{th} bit of each sample, the coding passes prioritize the coding of bits that will reduce distortion the most from the overall image. In the case of lossy encoding, instead of truncating the same number of bits for every sample in a region, JPEG2000 truncates a certain number of coding passes, effectively performing a "selective" bit truncation per sample. Furthermore, JPEG2000 has a highly hierarchic partitioning policy which permits random access and decoding of spatial regions in the codestream.

A detailed treatment of the codestream formation process in JPEG2000 is given in Sec. 2.2.1 of Chapter 2.

3.1.3 Compression of Background Models using JPEG 2000

In order to compress the background model, which is an array of GMM distributions, it is necessary to derive the associated GMM distribution for the DWT coefficients at each subband at each level of the wavelet decomposition. Since the wavelet transform involves basic arithmetic operations such as addition and scalar multiplication, the required transform of the GMM will be evaluated according to the presented novel technique based on the algebra of random variables.

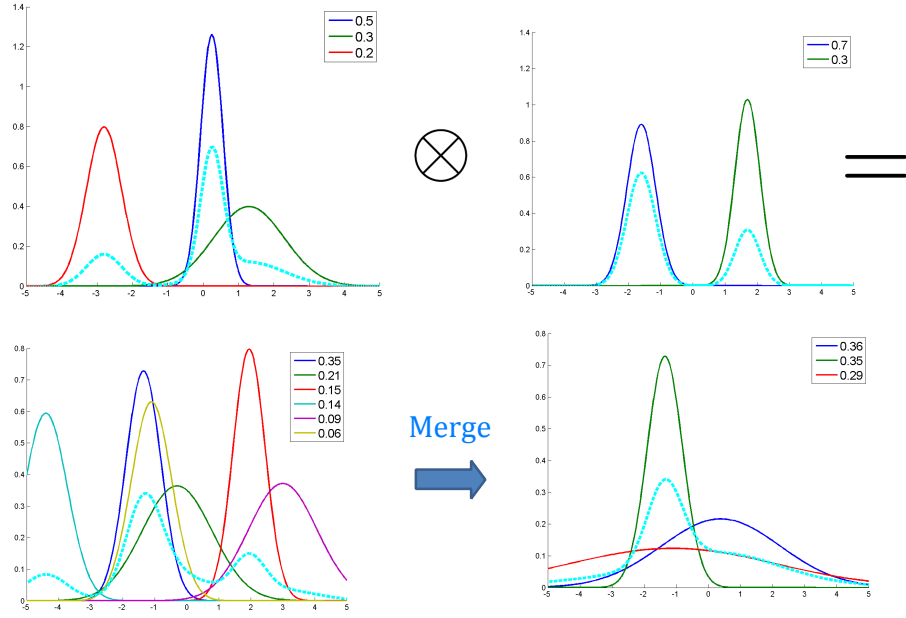


Figure 3.2: Convolution of two GMMs increases the number of components; The two lowest weighted components are merged until the desired number of components is reached

Algebra of Random Variables

To obtain the distribution of the sum of two independent random variables knowing each of their distribution, one must convolve one pdf with the other. Mathematically,

$$P_{X+Y}(z) = P_X(x) \otimes P_Y(y) \quad (3.10)$$

The operator $\otimes$ stands for convolution. Similarly, one can determine the distribution of an invertible function g of a random variable as such [132]:

$$P_{g(X)}(y) = P_X(g^{-1}(y)) \cdot \frac{dg^{-1}(y)}{dy} \quad (3.11)$$

for our purposes let g be a linear function of the form $Y = g(X) = s \cdot X$. Thus (3.11) becomes

$$P_Y = \frac{1}{s} P_{Y/s} \quad (3.12)$$

Extending these to normally distributed random variables we have for the sum operator [134]:

$$\mathcal{N}(X, \mu_X, \Sigma_X) \otimes \mathcal{N}(Y, \mu_Y, \Sigma_Y) = \quad (3.13)$$

$$= \mathcal{N}(X + Y, \mu_X + \mu_Y, \Sigma_X + \Sigma_Y) \quad (3.14)$$

Similarly for scaling:

$$\frac{1}{s} \mathcal{N}\left(\frac{Y}{s}, \mu, \Sigma\right) = \mathcal{N}(Y, s \cdot \mu, s^2 \cdot \Sigma) \quad (3.15)$$

The order of summation and integration can be transposed thus obtaining,

$$\mathcal{M}(\theta^r) \otimes \mathcal{M}(\theta^q) = \int_z \sum_{i=0}^m \omega_i^r P_{X_i}(z) \sum_{j=0}^n \omega_j^q P_{Y_j}(x - z) dz \quad (3.16)$$

$$= \sum_{j=0}^n \sum_{i=0}^m \omega_i^r \omega_j^q \int_z P_{X_i}(z) P_{Y_j}(x - z) dz \quad (3.17)$$

$$= \sum_{j=0}^n \sum_{i=0}^m \omega_i^r \omega_j^q \mathcal{N}(X_i, \theta_i^r) \otimes \mathcal{N}(X_j, \theta_j^q) \quad (3.18)$$

Note that the convolution of two mixtures of size m and n generally yields $m \cdot n$ modes. The scalar multiplication of mixtures simply scales each mode thus:

$$s \cdot \mathcal{M}(\omega^r, \boldsymbol{\mu}^r, \Sigma^r) = \mathcal{M}(\omega^r, s\boldsymbol{\mu}^r, s^2\Sigma^r) \quad (3.19)$$

The wavelet transform of Random Variables

Based on these operations over random variables, the distribution for each wavelet coefficient can be obtained. One issue is that convolution of distributions produces a number of components equal to the product of the number of components in each distribution. It is necessary to prune back the extra components. One approach is to delete the lowest weight components and then renormalize the weights of the remaining components. Alternatively Z. Zhang *et al.* [140] propose an elegant way of merging modes in a mixture. Essentially, two mixture modes with weights ω_i and ω_j will yield a new mode with weight $\omega_k = \omega_i + \omega_j$ after the merger. The underlying property of the newly obtained mode is:

$$\omega_k \mathbf{Pr}(\mathbf{X}|k) = \omega_i \mathbf{Pr}(\mathbf{X}|i) + \omega_j \mathbf{Pr}(\mathbf{X}|j) \quad (3.20)$$

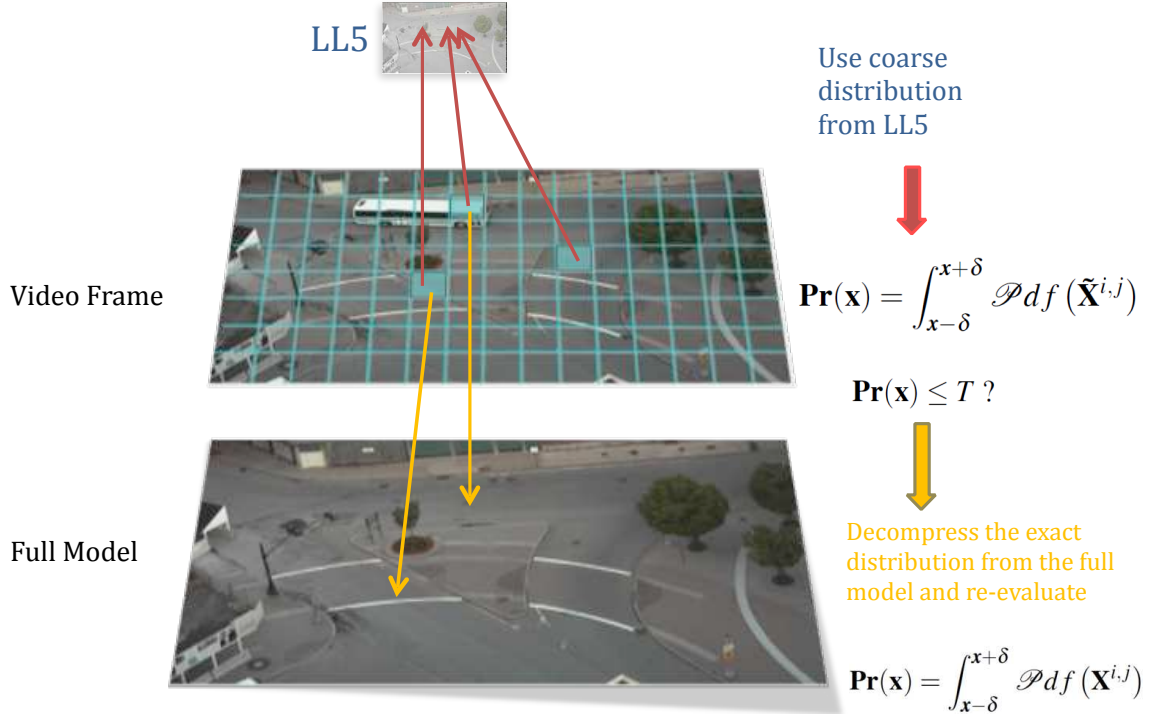


Figure 3.3: Probability map evaluation using LL5

Taking the expectation operator on each side will give out the mean $\boldsymbol{\mu}_k$ of the new distribution. The covariance is similarly obtained by solving $\Sigma_k = \mathbb{E}[\mathbf{X}\mathbf{X}^T|k] - \boldsymbol{\mu}_k\boldsymbol{\mu}_k^T$. Finally we end up with the following merger relationships:

$$\omega_k \boldsymbol{\mu}_k = \omega_i \boldsymbol{\mu}_i + \omega_j \boldsymbol{\mu}_j \quad (3.21)$$

$$\omega_k (\Sigma_k + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^T) = \omega_i (\Sigma_i + \boldsymbol{\mu}_i \boldsymbol{\mu}_i^T) + \omega_j (\Sigma_j + \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T) \quad (3.22)$$

After each addition operation, the extra modes are merged until the desired mixture size obtained, e.g. three or five components, as seen in Fig 3.2. Using the lifting scheme [36] the approximate distribution of wavelet coefficients is obtained. For each frame $\mathbf{F}$ in the video, the probability map $\mathbb{P}$ is obtained by evaluating at every pixel (i, j) $\Pr(\mathbf{X}_t^{i,j} = F^{i,j})$. Thresholding and binarizing $\mathbb{P}$, a mask is obtained to select the foreground pixels which will be encoded using standard JPEG2000. According to A. Perera *et al.* [99] H.264 is reputed to have better performance in encoding foreground blocks. However, as mentioned earlier, its memory cost preclude its application in wide area aerial video collection.

It is desirable to work with a JPEG2000 compressed representation of Θ when obtaining $\mathbb{P}$. A sound

implementation is to store in high speed memory the lowest resolution LL band (typically LL5) and use its random variables to evaluate the probability map. Let the lowest LL band in the wavelet transform decomposition of Θ be Θ_{LL} . Because the size of Θ_{LL} is $2^5 = 32$ times smaller than Θ in each dimension, each distribution in Θ_{LL} will be used to measure the probability of the pixels in a 32x32 patch in the video frame $\mathbf{F}$. A less accurate probability map will result than the one obtained using the full model Θ . However, by taking advantage of the scalability features of JPEG 2000, the accuracy of this probability map can be increased according to the method described below. Low probability pixels are assumed to be due to the result of actual foreground or possibly due to the inaccuracy of the distribution in Θ_{LL} . Distributions from Θ_{LL} are refined by local decompression from the codestream in order to distinguish true foreground from model inaccuracy. Pixels that are found to have low probability in a frame will have their corresponding distribution from Θ determined via local JPEG2000 decompression. The probability for those pixels is then re-evaluated with the decompressed distributions which are close to the distributions of the in original model Θ , as shown in Figure 3.3. The model will not be exactly recovered due to the fact that JPEG2000 irreversible compression is employed on Θ in order to obtain higher compression gains.

It is safe to assume that foreground pixels exist in coherent regions. Therefore it is efficient that a pixel needing local decompression causes the neighboring distributions to also be decompressed due to the pyramid structure of the DWT. Thus, the overhead involved with performing the inverse DWT and bitplane de-coding is minimized.

Algorithm 1: Frame Encoding using JPEG2000 compressed Background Model

Input : Compressed background model $\tilde{\Theta}$, coarsest resolution n , input frame $\mathbf{F}$, refinement decision probability threshold τ_p
Output: Compressed Foreground $\tilde{\mathbb{F}}$

```

1  $\Theta_{LL} = \text{J2KGetSubband}(\tilde{\Theta}, 'LL', n)$ 
2 foreach  $\mathbf{i} \in \mathbf{F}$  do
3    $\mathbb{P}(\mathbf{i}) = \text{Pr}(\mathbf{F}(\mathbf{i}), \Theta_{LL}(\mathbf{i}))$ 
4   if  $\mathbb{P}(\mathbf{i}) < \tau_p$  then
5      $\mathbb{P}(\mathbf{i}) = \text{Pr}(\mathbf{F}(\mathbf{i}), \text{J2KDecompress}(\tilde{\Theta}, \mathbf{i}))$ 
6   end
7 end
8  $\tilde{\mathbb{F}} = \text{ApplyMask}(\mathbf{F}, \text{Threshold}(\mathbb{P}, 0.9))$ 
9 return  $\text{J2KCompressImage}(\tilde{\mathbb{F}})$ 
```

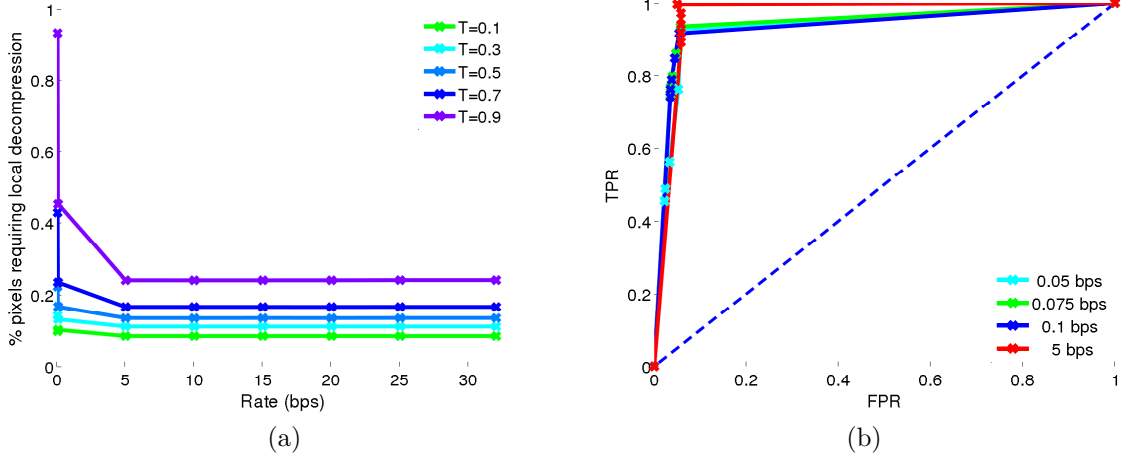


Figure 3.4: (a): Percent of pixels requiring local refinement vs. bit rate. Each curve represents a different decision threshold T for local decomposition. (b): ROC characteristic curves for various bitrates

3.2 Experiments

In the first experiments, data that has been obtained from a high-definition video camera is used to evaluate the proposed scheme. In a final experiment, the compression performance is evaluated on ARGUS wide-area aerial video data taken from one of the focal planes. [121] As mentioned above, if the probability of a certain pixel measured with Θ_{LL} falls below a certain value, the pixel's corresponding distribution from the compressed Θ is extracted from the codestream. Several experiments have been run with different decision thresholds, namely $\{0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99\}$. Background models were encoded at various bitrates also starting at 0.05 and ending at 32 bps (bits per sample). A 1280 x 720 background model having a maximum of three components per mixture and each component having an independent covariance matrix takes up 5.5 KB of storage when JPEG2000 compressed at 0.05 bps. A higher rate like 32 bps will increase the storage cost per frame to 3184 KB. On the other hand, higher bitrate models require a smaller number of local decompressions when evaluating foreground probability. It can be noted that even the higher rate produces a model that is approximately 100 times smaller than the original GMM and comparable in storage to a single uncompressed color video frame. The JPEG 2000 library used was D Taubman's "Kakadu" 2.2 library [121]

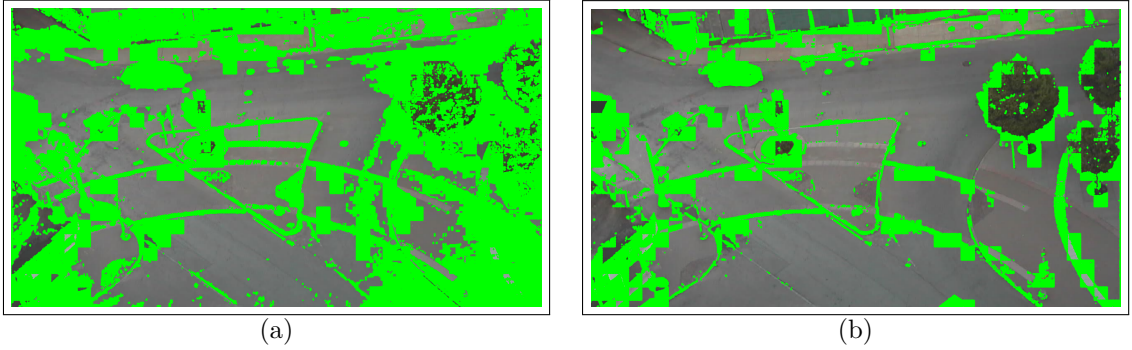


Figure 3.5: Number of local refinements required with a decision threshold of 0.7 and a model compressed at a rate of (a) 0.05 bps and (b) 32 bps

Figure 3.5 shows the pixels which require local decompression when evaluating the probability map on one of the frames with two differently encoded background models. Figure 3.4 (a) shows that the number of lookups drops dramatically as bitrate increases from a fractional value to an integer one. Moreover, the receiver operator characteristic (ROC) curves in figure 3.4 (b) depict that the True Positive Rate (TPR) vs. False Positive Rate (FPR) pairs approach the top left corner rapidly as a function of the bitrate of the model used to measure foreground probability. From both figures, it is clear that models encoded at bitrates ranging from 5bps and above exhibit very similar characteristics both in the ability to correctly identify background and the in number of local decompressions required during probability evaluation.

3.2.1 Results

Video id	Model Θ	Video - Lossless	Video - Lossy @ 0.05 bps
Still 720p Camera	96	4	31
ARGUS City Scene	96	16	87

Table 3.1: Compression Ratios for two Video Sequences.

After each probability map is evaluated with the method described above, a binary mask is derived from it via probability thresholding and is applied on the corresponding frame. The resulting foreground objects are encoded using JPEG2000. Once every 50 frames the mean image $\mathbf{M}_{t,k}$ of the

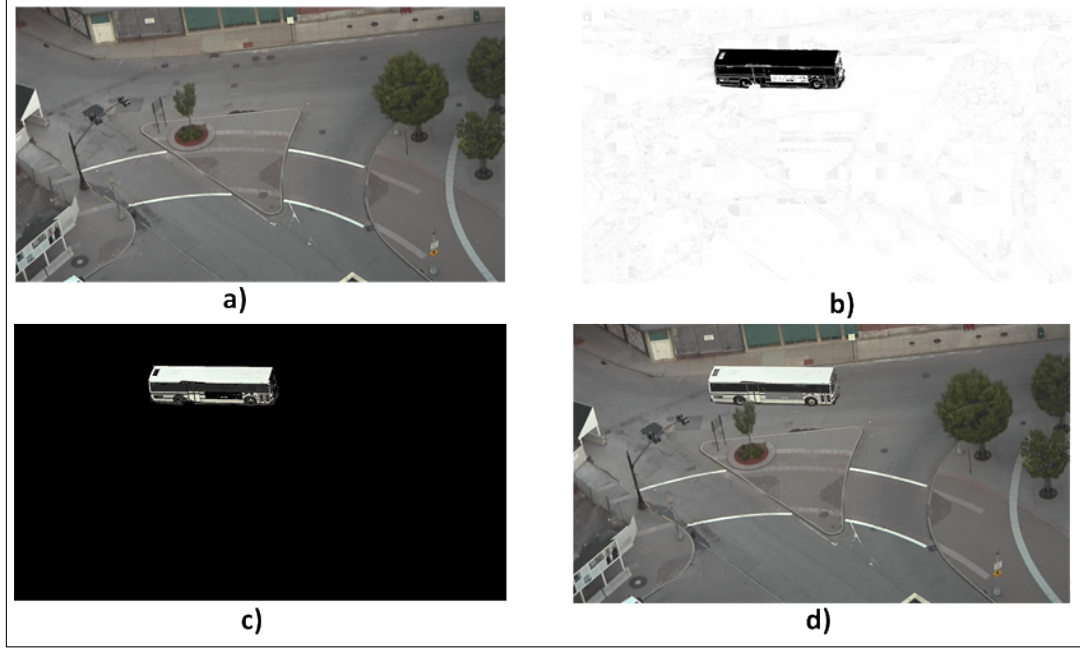


Figure 3.6: (a) Mean image $\mathbf{M}_{k,50}$ (b) probability map evaluated with model encoded at 5 bps (c) Segmented Foreground (d) Reconstructed Frame $d = a + c$

highest weighted component of the background model is encoded, where

$$\mathbf{M}_{t,k} = \{\boldsymbol{\mu}_{tk}^{ij} | i < w, j < h, k = \arg \max_r (\omega_t^r), \mathbf{X}_t^{i,j} \sim \mathcal{M}(\omega_t^r, \boldsymbol{\mu}_t^r, \Sigma_t^r)\} \quad (3.23)$$

Figure 3.6 shows a video frame and associated probability map, foreground set and its reconstruction post compression. A 600 frame 720p video, having each foreground object losslessly compressed according to the described method, will reduce its overall storage reduced by a factor of 4. Each foreground frame has an average of 0.02 bps. The lossy encoding of foreground objects is possible, at the expense of reconstruction artifacts. These are due to the fact that JPEG2000 smooths with each DWT level abrupt transitions from RGB values at foreground edges to the 0-value background label. This behavior has been reported by Perera *et al.* [99]. One solution is to losslessly encode a binary mask corresponding to the foreground and apply it on the decoded foreground to eliminate the smoothing artifacts. The results of this masking technique are shown in Figure 3.7. A second aerial video sequence, acquired from one of the ARGUS focal planes, has frame size 2740x2029 and the pixel resolution of moving objects is 25 times lower than for the stationary camera.

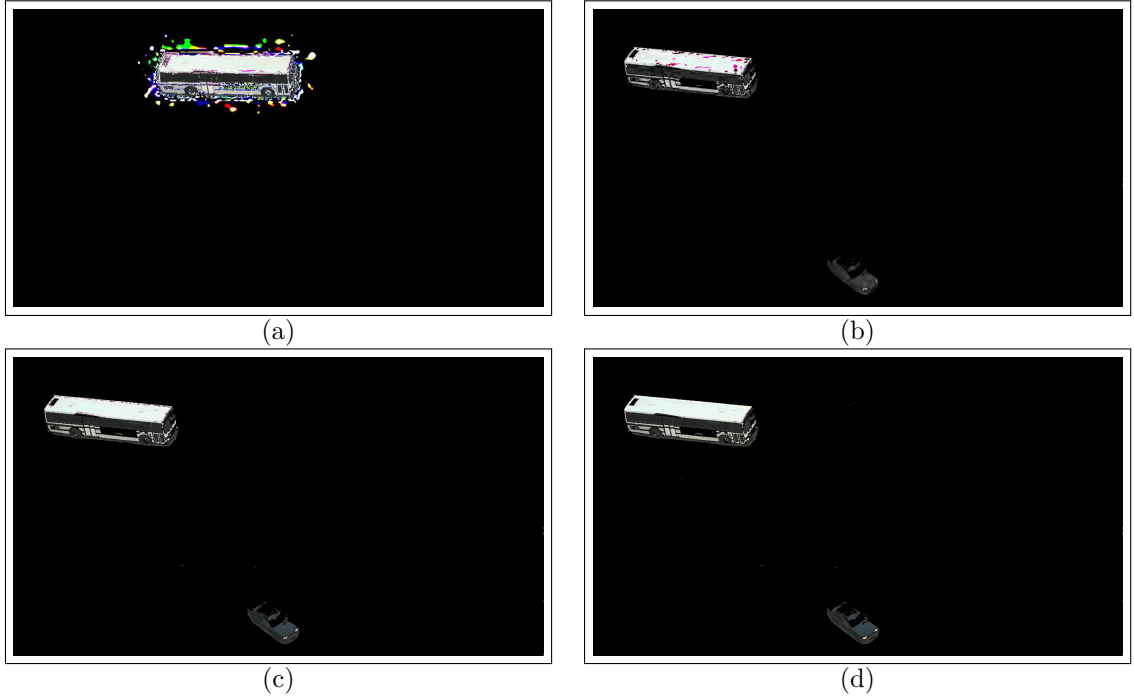


Figure 3.7: Encoded foreground (a) lossy 0.01 bps (no mask), (b) lossy at 0.01 bps , (c) lossy at 0.05 bps and (d) lossless

As a consequence, a high compression ratio is achieved since the relative area of moving objects is much smaller. Scaling the results to the full 1.5 GByte ARGUS sequence, the encoding of moving objects requires only 90 MBytes. The results for video and background model compression are summarized in Table 1. In each case, moving objects are encoded with no compression. It should be noted that in the ARGUS sequence additional bits were spent on encoding pixel intensities near discontinuities (edges) that are labeled as foreground due to frame misalignment. However, misalignment still accounts for a small fraction of the pixels. In addition, the representation in Section 4.3 will provide the 3-d, more general solution to representing the appearances of objects, not only limited to background. Representing aerial scenes in the 3^{rd} dimension will obviate the need for co-registering frames with homographies.

3.3 Conclusions

It has been demonstrated that efficient foreground detection and frame encoding can be achieved by exploiting the intrinsic mechanisms of the JPEG2000 coding scheme. By encoding the probability

distributions it is possible to reduce the storage cost of GMM per pixel to the same order as a single video frame. The resulting accuracy in foreground detection, even for video that is registered to a single ground plane, enables a significant advance in compression ratio without sacrificing the quality needed for computer vision algorithms such as tracking.

Future work will focus on a GPU implementation of the proposed algorithm. Pixel-wise and frame-wise parallelism is inherent will be exploited in the implementation. Another consideration is to develop algorithms for the lossy encoding of foreground objects to further improve the compression ratio. As noted by Perera *et al.* [99], such compression is not a trivial task since JPEG2000 smooths with each DWT level abrupt transitions from RGB values at foreground edges to the 0-value background label. Moreover, such encoding will inevitably require closer integration with the computer vision algorithms, such as encoding only the information that is actually used in tracking.

Chapter 4

4-d Modeling

As seen in Sec. 1.1 the best compression factors in 2-d video compression are obtained when temporal redundancy is exploited efficiently through detected foreground motion. Therefore, this thesis will focus on an approach to 4-d compression where 3-d object dynamics are directly encoded. Time varying object surface position is estimated using 3-d optical flow vectors. As a result, the chosen representation of 4-d space will be intrinsically linked to the accuracy of the estimated flow and ultimately to compression gain and compressed scene quality. This chapter is first dedicated to describing properties of commonly used methods of modeling 4-d space and their suitability for the flow estimation process and compression.

Due to the high accuracy and robustness in dealing with scene uncertainty and ambiguity, the probabilistic volumetric model is an attractive method of representing 4-d space which will enable accurate motion estimation essential in a compression scheme. Consequently, the second part of this chapter defines in detail the underpinnings of probabilistic volumetric models, the 4-d representation used in this thesis.

4.1 Mesh Based Representation and Compression

Meshes are one of the most popular methods of representing 3-d space and time varying 3-d space due to their early adoption in the field of computer graphics and theoretical formulation based on combinatorial topology. Their early widespread popularity spurred the development of dedicated GPU chips for mesh rendering and APIs specifically designed for mesh processing. A polygon mesh



Figure 4.1: Mesh based representation by Arcila *et al.* [8]. Here mesh connectivity is unable to deal with changing scene topology

is a set of vertexes, edges and faces that collectively represent the 3-d object. Faces are typically triangles, convex polygons or quadrilaterals. Mesh based reconstruction generally involves fitting the elements of the mesh to a known 3-d shape. The method in [131] involves deforming then refining an initial mesh according to an objective function that maximizes a photo-consistency metric. Despite successes in both indoor and outdoor data, these approaches require a priori topological information of the initial mesh. Moreover, in 4-d mesh models, changing scene topology that's not consistent with the initial mesh topology will be modeled incorrectly. A scene consisting of a bouncing ball between two actors will be difficult to reconstruct using meshes if no information is known about the topology of the ball and the arms of the two actors, as is seen in Fig. 4.1 There are works that have been successful in detecting and adjusting scene topology accordingly for simple objects and small deformations [101, 139], but not for scenes with complex objects with rapid topological changes. Methods that employ a global topology that is consistent with all temporal instances have proven reasonably successful in indoor studios where a set of mesh refinement non-linear registration and topological change detection tools were used [16, 80]. However, long outdoor scenes where a global topology cannot be easily provided remain challenging. For these reasons, estimating the motion of a dynamic scene with varying topology will prove to be a difficult task if meshes are the representation of choice.

4.2 Point Based Representation and Compression

Point clouds and time-varying point clouds are characterized by a set of data points in a given coordinate system. Under this representation, the data at each coordinate typically contains texture, normal information, probability information etc. Since point connectivity is not explicit, point-clouds are more flexible than meshes when dealing with topologically varying shapes. Among the attributes

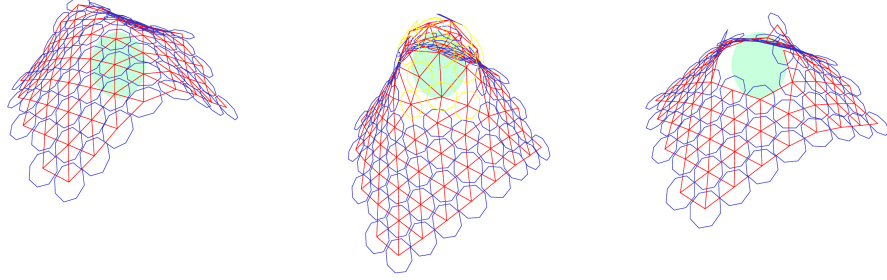


Figure 4.2: Surfel Representation as presented by Tonnesen *et al.* [119]. The non-connectivity property of the point cloud allows the modeling of surface deformations such as a cloth tear.

listed above, point clouds can be augmented with a surface representation. For example, the work of Szeliski and Tonnesen [119] uses small surface elements (surfels) at each coordinate allowing the modeling complex shape deformations. In the context of 3-d reconstruction works like [26] estimate attributes of surfels such as curvature, motion and position and use them to reconstruct small 3-d volumes without a priori information. Patch segmentation and tracking were also utilized [102] with reasonable success.

While these methods can be used to model arbitrarily changing shapes [26, 55, 71], point clouds obtained from multi-view camera setups are typically sparse. The output of 3-d reconstruction methods, including the state of the art patch-based multi-view stereo (PMVS) proposed by Furukawa *et al.* [47] typically exhibits surface irregularities, errors and holes especially in the presence of calibration errors, featureless surfaces and scene ambiguity in general. Additionally, since neighborhood information isn't explicit, many processing operations (*e.g.* filtering, interpolation) require added complexity or additional data structures. For this reason, an implementation of an optical flow algorithm performed directly on 3-d point-cloud will prove cumbersome. Also, due to the sparse nature of point-clouds and generally poor quality of the resulting 3-d model, the accuracy of the optical flow estimation is guaranteed to suffer. Recent work by Park *et al.* [71] show robust tracking of thousands of points can be achieved using 480 cameras. However, this thesis focuses on more traditional and practical motion capture studios containing at most 16 cameras while producing highly dense motion estimates as shown in Ch. 5

4.3 Probabilistic Volumetric Models

As noted in the previous sections, it is desirable to represent dynamic scenes over a continuous domain. A dense sampling of space will enable a more accurate description of the motion contained in a scene which in turn will allow for better compression and reconstruction. Voxel based representations partition 3-d space into cubic primitives with a desired size which can record attributes such as color or occupancy. The voxel grid thus enables a continuous domain over which surfaces can be represented. Unlike meshes and point-clouds, the voxel grid is defined everywhere within the scene domain. Thus, modeling topological changes of scene geometry over time is unnecessary since neighborhood relations are maintained unambiguously by the grid. This continuous representation allows for dense flow estimation, which will be key in scene reconstruction and ultimately compression. Another advantage of the voxel formulation is the ease of retrieving neighborhoods around a certain location, unlike point-based methods where interpolation or additional data structures such as K-d-trees are required.

It is desirable to combine the continuous formulation of PVMs with probabilistic inference at every voxel in order to model scene uncertainty and ambiguity. There are many probabilistic methods used in conjunction with the volumetric representation [12] [21] [17] [100], many of which estimate the likelihood of a cell being occupied by a surface element via Bayesian inference. The PVM framework was first introduced by Pollard and Mundy [100]. In their representation, the scene volume is decomposed into a uniform grid of voxels, each voxel $\mathbf{X}$ containing a *surface probability*, *i.e.* the probability that the voxel is occupied by a surface element, that is denoted by $P(\mathbf{X} \in S)$. Each voxel also contains a *surface appearance distribution*. The term appearance refers to the observed intensity or color of a surface element arising from its reflectance and incoming scene illumination. Although this distribution is only meaningful for voxels containing surfaces, it is defined everywhere since each voxel has some non-zero probability of containing a surface. Pollard and Mundy utilized a mixture of Gaussians (MoG) distribution to model Lambertian surface appearance. The multi-modal nature of the MoG makes it robust to camera calibration errors, occlusions and transient objects. More specifically, the appearance distribution $P(I_R | V_{R(X)} = X)$ represents the likelihood that voxel $V_{R(X)}$ along ray R was responsible for the observed intensity I_R . To reconstruct the PVM from images (with known camera pose and calibration), they proposed an algorithm that *updates* the surface probability and appearance distributions in the volume using a single image.

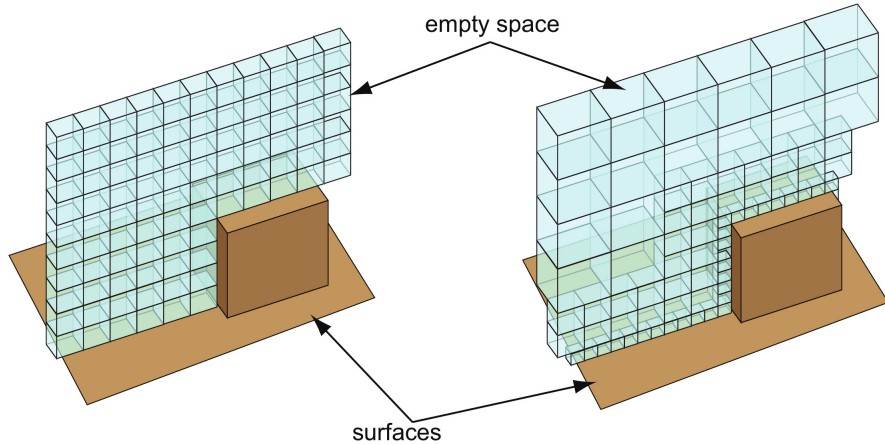


Figure 4.3: The non-uniform voxel grid refines 3-d space corresponding to surfaces. Illustration taken from Crispell *et al.* [35]

This online algorithm follows from the Bayesian update equations and jointly updates the geometry and appearance in the model using pixel observations from the image. Section 4.3.1 will be dedicated to describing in detail the underpinnings of the learning process of PVMs.

The PVM formulation for uniform grids was later adopted to non-uniform grids, and in particular, the octree by Crispell *et al.* [35]. This octree extension led to higher quality 3-d models while requiring less storage than the uniform grid implementation. In this work, the reparametrization of surface probability per voxel as a probability density along a ray allowed for a non-uniform subdivision of the voxel grid. The details along with the advantages of this implementation will be presented in Sec. 4.3.2. The octree implementation was further refined using data structures that are amenable to GP-GPU processing by Miller *et al.* [93]. Miller *et al.* also exploited parallel execution in the GPU to gains on the order of 600x in speed for scene reconstruction and rendering. Their implementation allows reconstruction of large-scale high resolution 3-d models particularly using aerial photo collections [25, 93, 104]. Section 4.3.3 will offer details on how the surface probability reparametrization allows for non-uniform grids which enable a significant reduction in storage.

Despite success in reconstructing complex urban scenery that includes featureless surfaces, moving objects, occlusions and specularities, volumetric models have not gained popularity due to their prohibitive storage cost.

A seminal work in 4-d PVM compression is proposed by Ulusoy *et al.* [123]. The 4-d world is represented by a time-space-partitioning (TSP) tree which was first introduced by Shen *et al.* [109]. This data structure stores binary time trees at the leaves of the spatial octree. Specifically, a location

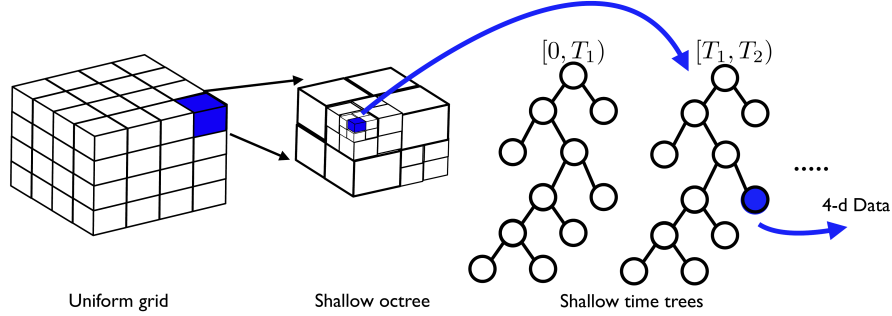


Figure 4.4: 4-d space representation using Time Trees as presented by Ulusoy *et al.* [123]

in space corresponding to an octree cell will contain a history of all the values of the parameters at that location up to the current time. Thus the binary-tree corresponding to an octree cell will adaptively subdivide to accommodate new time instances at that location. The gain in compression comes from the fact that the binary time tree at each octree cell is only subdivided when the surface and appearance parameters at that location change significantly over time. Thus, static regions will contain root-level binary time trees while regions that change over time due to motion will cause the binary time tree to subdivide. The appearance and surface information of the new time instance is then stored at the newly created leaf. Motion is detected in this framework via the Kullback-Leibler(KL) divergence. $D_{KL}(p||q)$ is a measure of how well q can approximate p and is defined in Eq. (4.1) [33]. According to this metric, the algorithm decides whether the current surface and appearance parameters are sufficient to approximate the corresponding parameters at the next time instance. If the distance exceeds a threshold, the time-tree is subdivided to ingest the new set of parameters. Specifically, Ulusoy *et al.* define an internal appearance distribution threshold τ_I and surface probability distribution threshold τ_S . The subdivision in time will occur if the inequality in Eq. (4.2) is satisfied.

$$D_{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} \quad (4.1)$$

$$D_{KL}(\mathbf{Pr}(X^t \in S) || \mathbf{Pr}(X^{t+1}) \in S) > \tau_S \vee D_{KL}(p(I; X^t) || p(I; X^{t+1})) > \tau_I \quad (4.2)$$

The defined thresholds can be varied in order to achieve desired compression ratios and corresponding scene qualities. However, because the time-tree depth is limited to 5 in order to allow efficient access on the GPU fast memory, the 4-d model compression ratio can be at most 32. For this reason, a system that actually estimates motion can achieve a higher degree of compression by exploiting temporal redundancies more efficiently. The next section will present the atomic elements

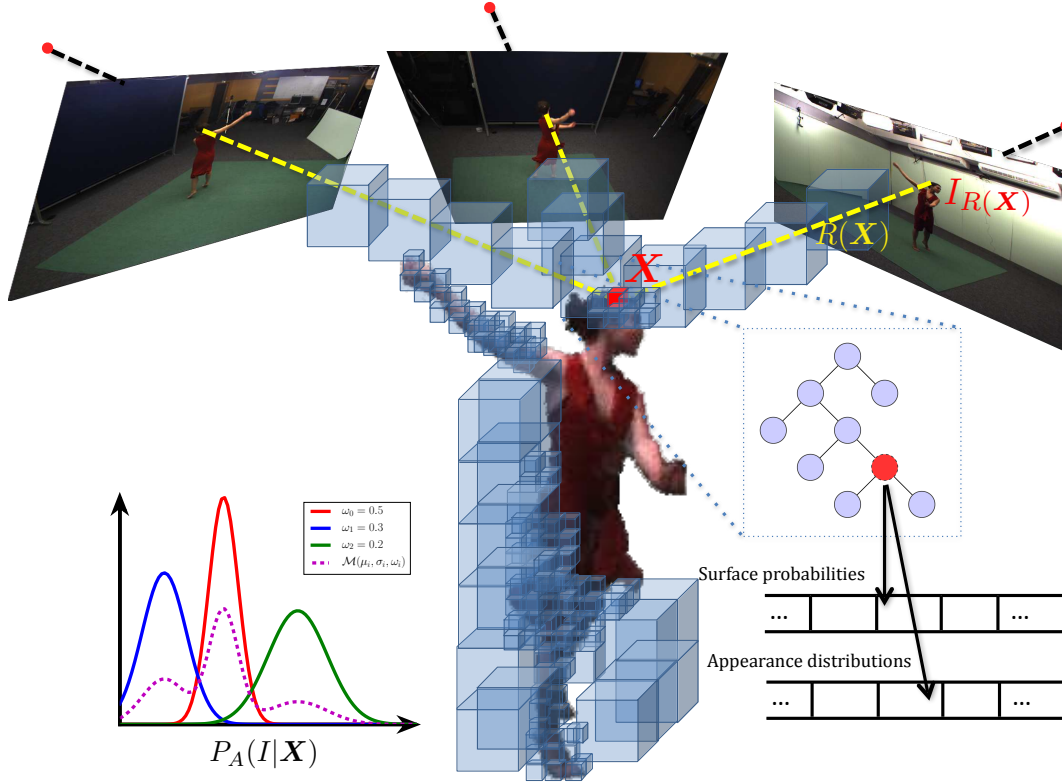


Figure 4.5: 3-d space modeling using probabilistic volumetric models. Surface geometry is encoded using smaller volumetric elements (voxels) than the ones representing empty space. Scene structure is maintained through octrees. The surface and appearance parameters at each cell are learned from image intensity observations

that constitute the PVM, namely the appearance and surface distributions, along with the process of learning their parameters. The theory behind these concepts will lay the foundation for the next chapters where 3-d optical flow estimation and compression using optical flow is presented.

4.3.1 On-line PVM Inference

The on-line learning scheme proposed by Pollard and Mundy is used to estimate the surface appearance and occupancy probability in each voxel [100]. The appearance is modeled as a mixture of Gaussians (MoG) where parameters are trained jointly from surface information using an on-line algorithm similar to the one proposed by Stauffer and Grimson [116]. Every voxel in the grid contains an appearance distribution and a non-zero probability of being a surface. Pollard and

Mundy's assumption is that only one voxel along the camera ray $R_{\mathbf{X}}$ is responsible for the observed image intensity or color $I_{R(\mathbf{X})}$. The random variable $V_{R(\mathbf{X})}$ models this responsibility for each voxel along the ray $R_{\mathbf{X}}$ originating from the camera center to $\mathbf{X}$. As mentioned in Sec. 4.3, the likelihood of observing $I_{R(\mathbf{X})}$ knowing voxel $\mathbf{X}$ was responsible is modeled by the distribution $\mathbf{Pr}(I_{R(\mathbf{X})}|V_{R(\mathbf{X})} = \mathbf{X})$. For a ray $R_{\mathbf{X}}$ intersecting voxel $\mathbf{X}$ and intensity observation $I_{R(\mathbf{X})}$ from the input image $t + 1$, the posterior surface probability $\mathbf{Pr}^{t+1}(\mathbf{X} \in S|I_{R(\mathbf{X})})$ is updated according to an online Bayesian formulation for every voxel $\mathbf{X}$ along the ray.

$$\mathbf{Pr}^{t+1}(\mathbf{X} \in S|I_{R(\mathbf{X})}) = \mathbf{Pr}^t(\mathbf{X} \in S) \frac{\mathbf{Pr}^t(I_{R(\mathbf{X})}|\mathbf{X})}{\mathbf{Pr}^t(I_{R(\mathbf{X})})} \quad (4.3)$$

Looking closely at the terms in Eq. (4.3), one can identify that the prior $\mathbf{Pr}^n(\mathbf{X} \in S)$ at iteration n is scaled by the ratio between the probability of observing intensity $I_{R(\mathbf{X})}$ given that $\mathbf{X}$ is a surface voxel, *i.e.* $\mathbf{Pr}^n(I_{R(\mathbf{X})}|\mathbf{X} \in S)$ and the probability of observing $I_{R(\mathbf{X})}$ "in general". Unfortunately, the voxel that produced the intensity is not known a priori, thus the likelihood in the numerator is expanded as a sum over all the voxels along the ray according to Eq. (4.4).

$$\mathbf{Pr}^n(I_{R(\mathbf{X})}|\mathbf{X} \in S) = \sum_{\mathbf{X}' \in R(\mathbf{X})} \mathbf{Pr}^n(I_{R(\mathbf{X})}|V_{R(\mathbf{X})} = \mathbf{X}', \mathbf{X} \in S) \mathbf{Pr}^n(V_{R(\mathbf{X})} = \mathbf{X}'|\mathbf{X} \in S) \quad (4.4)$$

Conditioning on $V_{R(\mathbf{X})} = \mathbf{X}'$ in the first term of the sum implies that $\mathbf{X}'$ produced the observed intensity. Consequently, the fact that $\mathbf{X} \in S$ is no longer important. Thus the first term becomes $\mathbf{Pr}^n(I_{R(\mathbf{X})}|V_{R(\mathbf{X})} = \mathbf{X}')$, which is the appearance model at voxel $\mathbf{X}'$. The next step of the process involves modeling the distribution of the r.v. $V_{R(\mathbf{X})}$. Pollard and Mundy assume that a voxel $\mathbf{X}'$ will produce the observed intensity along a ray if it is visible and is a surface voxel. Thus it follows that,

$$\mathbf{Pr}^n(V_{R(\mathbf{X})} = \mathbf{X}') = \mathbf{Pr}^n(\mathbf{X}' \text{ is visible}) \mathbf{Pr}^n(\mathbf{X}' \in S) \quad (4.5)$$

Consequently, the probability of a voxel $\mathbf{X}'$ being visible is expressed as a product over all the voxels along the ray that are before $\mathbf{X}'$

$$\mathbf{Pr}^n(\mathbf{X}' \text{ is visible}) = \prod_{\mathbf{X}'' < \mathbf{X}'} (1 - \mathbf{Pr}^n(\mathbf{X}'' \in S)) \triangleq \text{vis}(\mathbf{X}') \quad (4.6)$$

Thus expanding both the numerator and the denominator in Eq. (4.3) as a sum over all voxels along the ray the update ratio becomes,

$$\frac{\sum_{\mathbf{X}' \in R(\mathbf{X})} \mathbf{Pr}^n(I|V = \mathbf{X}') \mathbf{Pr}(V = \mathbf{X}'|\mathbf{X} \in S)}{\sum_{\mathbf{X}' \in R(\mathbf{X})} \mathbf{Pr}^n(I|V = \mathbf{X}') \mathbf{Pr}(V = \mathbf{X}')} \quad (4.7)$$

Since the second term in the sum in the numerator conditions on $\mathbf{X} \in S$, one can set $\mathbf{Pr}(\mathbf{X} \in S) = 1$, for all the voxels $\mathbf{X}' < \mathbf{X}$. A further implication on this conditioning is that all the voxels $\mathbf{X}' > \mathbf{X}$ cannot be responsible for the observed intensity, thus $\mathbf{Pr}(V = \mathbf{X}'|\mathbf{X} \in S) = 0, \forall \mathbf{X}' > \mathbf{X}$. Thus the numerator can be expanded as follows:

$$\sum_{\mathbf{X}' \in R(\mathbf{X})} \mathbf{Pr}^n(I|V = \mathbf{X}') \mathbf{Pr}(V = \mathbf{X}'|\mathbf{X} \in S) = \mathbf{Pr}^n(I|V = \mathbf{X}) \underbrace{\mathbf{Pr}(V = \mathbf{X}|\mathbf{X} \in S)}_{1 \cdot \text{vis}(\mathbf{X})} + \quad (4.8)$$

$$+ \sum_{\mathbf{X}' < \mathbf{X}} \mathbf{Pr}^n(I|V = \mathbf{X}') \mathbf{Pr}(V = \mathbf{X}') + \sum_{\mathbf{X}' > \mathbf{X}} \mathbf{Pr}^n(I|V = \mathbf{X}') \underbrace{\mathbf{Pr}(V = \mathbf{X}'|\mathbf{X} \in S)}_0 = \quad (4.9)$$

$$= \sum_{\mathbf{X}' < \mathbf{X}} \mathbf{Pr}^n(I|V = \mathbf{X}') \mathbf{Pr}(V = \mathbf{X}') + \mathbf{Pr}^n(I|V = \mathbf{X}) \text{vis}(\mathbf{X}) \quad (4.10)$$

Similarly, consider the sum of the voxels after $\mathbf{X}$ in the sum in the denominator of Eq. (4.7) which can be expanded as follows:

$$\sum_{\mathbf{X}' > \mathbf{X}} \mathbf{Pr}^n(I|V = \mathbf{X}') \mathbf{Pr}(V = \mathbf{X}') = \quad (4.11)$$

$$\sum_{\mathbf{X}' > \mathbf{X}} \mathbf{Pr}^n(I|V = \mathbf{X}') \mathbf{Pr}^n(\mathbf{X}' \in S) \prod_{\mathbf{X}'' < \mathbf{X}'} \mathbf{Pr}^n(\mathbf{X}'' \notin S) = \quad (4.12)$$

$$\mathbf{Pr}^n(\mathbf{X} \notin S) \sum_{\mathbf{X}' > \mathbf{X}} \mathbf{Pr}^n(I|V = \mathbf{X}') \mathbf{Pr}^n(\mathbf{X}' \in S) \underbrace{\prod_{\mathbf{X}_p < \mathbf{X}} \mathbf{Pr}^n(\mathbf{X}_p \notin S)}_{\text{vis}(\mathbf{X})} \prod_{\mathbf{X} < \mathbf{X}'' < \mathbf{X}'} \mathbf{Pr}^n(\mathbf{X}'' \notin S) = \quad (4.13)$$

$$\mathbf{Pr}^n(\mathbf{X} \notin S) \text{vis}(\mathbf{X}) \sum_{\mathbf{X}' > \mathbf{X}} \mathbf{Pr}^n(I|V = \mathbf{X}') \mathbf{Pr}^n(\mathbf{X}' \in S) \prod_{\mathbf{X} < \mathbf{X}'' < \mathbf{X}'} \mathbf{Pr}^n(\mathbf{X}'' \notin S) \quad (4.14)$$

Here the probability of not being a surface is evidently $\mathbf{Pr}^n(\mathbf{X} \notin S) = 1 - \mathbf{Pr}^n(\mathbf{X} \in S)$. In order to analyze the properties of Eq. (4.7) in detail, it is useful to express the observation likelihoods as a sum over voxels that are encountered before $\mathbf{X}$ *i.e.* $\mathbf{X}' < \mathbf{X}$ and after $\mathbf{X}$, *i.e.* $\mathbf{X}' > \mathbf{X}$, respectively.

Namely,

$$\text{pre}_{\mathbf{X}} = \sum_{\mathbf{X}' < \mathbf{X}} \mathbf{Pr}^n(I|V = \mathbf{X}') \mathbf{Pr}^n(\mathbf{X}' \in S) \prod_{\mathbf{X}'' < \mathbf{X}'} (1 - \mathbf{Pr}^n(\mathbf{X}'' \in S)) \quad (4.15)$$

$$\text{post}_{\mathbf{X}} = \sum_{\mathbf{X}' > \mathbf{X}} \mathbf{Pr}^n(I|V = \mathbf{X}') \mathbf{Pr}^n(\mathbf{X}' \in S) \prod_{\mathbf{X} < \mathbf{X}'' < \mathbf{X}'} (1 - \mathbf{Pr}^n(\mathbf{X}'' \in S)) \quad (4.16)$$

Combining Eq. (4.8), (4.11) and using the quantities defined in Eq. (4.15) and (4.16) into Eq. (4.7) the multiplier of $\mathbf{Pr}(\mathbf{X} \in S)$ becomes:

$$\frac{\text{pre}_{\mathbf{X}} + \text{vis}(\mathbf{X}) \mathbf{Pr}^n(I|V = \mathbf{X})}{\text{pre}_{\mathbf{X}} + \text{vis}(\mathbf{X}) (\mathbf{Pr}^n(\mathbf{X} \in S) \mathbf{Pr}^n(I|V = \mathbf{X}) + (1 - \mathbf{Pr}^n(\mathbf{X} \in S)) \text{post}_{\mathbf{X}})} \quad (4.17)$$

Convergence The surface probability of a voxel is increased if it provides a better estimate of the observed intensity than other voxels on the ray. Namely, the surface evidence of voxel $\mathbf{X}$ increases if $\mathbf{Pr}^n(I|V = \mathbf{X}) > \text{post}_{\mathbf{X}}$. The algorithm is said to converge when

$$\lim_{n \rightarrow \infty} \mathbf{Pr}^n(\mathbf{X} \in S) = 1 \leftrightarrow \lim_{n \rightarrow \infty} \mathbf{Pr}^n(\mathbf{X} \notin S) = 0 \forall \mathbf{X} \quad (4.18)$$

In addition, each appearance distribution converges if it contains only one mode with minimally allowed nonzero variance. The latter implies that a surface voxel projects the same intensity in all of the input images on convergence. Given enough viewpoints and iterations, a true surface voxel's evidence will increase with each iteration since the quantity $\text{post}_{\mathbf{X}}$ becomes lower, which causes the ratio multiplier to be greater than 1. More specifically, if the appearance distribution $\mathbf{Pr}^n(I|V = \mathbf{X})$ explains a new observation better than the voxels after $\mathbf{X}$ its surface evidence will increase. Conversely, as surface voxels' evidence converges to certainty, empty voxels will have increasingly lower evidence since the denominator in the update ratio increases with larger $\text{post}_{\mathbf{X}}$ relative to $\mathbf{Pr}^n(I|V = \mathbf{X})$. The resulting empty voxels will have a uniform weight and high variance distribution as a result of non-consistent observations from multiple view-points/iterations. In reality, however, total convergence is never achieved. Featureless surfaces (lack of texture) will enable a multitude of solutions that explain well all of the observations. In addition, some surfaces are non-Lambertian which will cause fluctuations in intensity or color depending on the view point. These issues are referred to as scene ambiguity and scene uncertainty [12] and although the probabilistic approach handles them well to a certain degree, 3-d reconstruction is still an ill-posed problem. As seen in Fig. 4.6, there is an infinite number of reconstruction solutions for the featureless colored segments. The

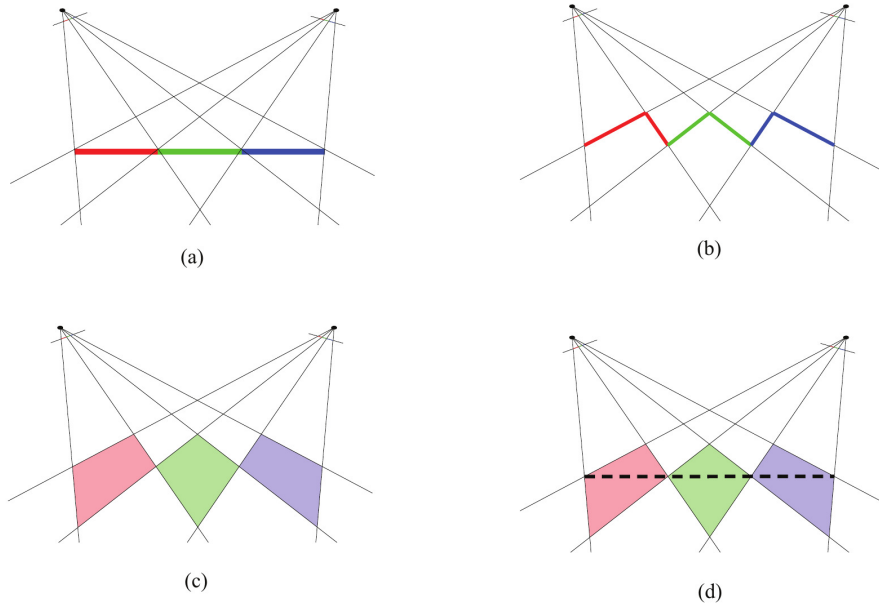


Figure 4.6: Scene uncertainty and ambiguity as exemplified by Crispell *et al.* [35]. (a) Two cameras capture three distinct featureless surfaces. (b) The Visual Hull Reconstruction method. (c) The probabilistic model. Here, the true surface may lie anywhere within the shaded regions. (d) Reconstruction using a prior shape model, *i.e.* the dotted line.

following section will focus on how appearance can be better modeled when there is a dependence on viewpoint.

Appearance Modeling As mentioned in Sec. 4.3.1, the initial implementation of Pollard and Mundy uses mixture of gaussians (MoG) as a model for surface appearance, typically containing 3 components as in Eq. (4.19). This multi-modal distribution accounts for appearance variations due to occlusions, illumination conditions, transient objects and calibration errors. Its parameters are learned using a slightly modified version of the on-line update scheme used by Stauffer and Grimson [116]. Modes are ranked the same way as presented in Sec. 3.1.1 using the value of ω_i/σ_i , thus preferring higher weight and lower variances. If a new observation I is within 2.5 standard deviations from the highest ranked component, its parameters will be updated with I . Since a voxel's probability of being a surface is not always 0, the appearance of all the voxels along the ray need to be updated with the new observation I , despite that only one voxel is assumed to have produced the observation. Consequently, a weighting scheme is introduced where voxels' appearance

parameters are weighted by their likelihood of being responsible for the observation, *i.e.* $\mathbf{Pr}(V = \mathbf{X})$.

$$p(I = I_{R(\mathbf{X})} | V_{R(\mathbf{X})} = \mathbf{X}) = \sum_{i=1}^3 \frac{\omega_i}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(I_{R(\mathbf{X})} - \mu_i)^2}{2\sigma_i^2}\right) \quad \sum_{i=1}^3 \omega_i = 1 \quad (4.19)$$

As previously stated, surface evidence increases with how well the appearance distribution can explain a new observation. However, if a surface has non-Lambertian reflectance the assumption of a constant observed intensity for a surface element is violated. Specifically, specular surfaces will be modeled by a high variance, uniform weight MoG since observations will be inconsistent across views. This causes surface evidence of specular regions to decrease, similar to those of an empty voxel. Ulusoy *et al.* [123] propose an appearance modeling scheme that deals with specular surfaces efficiently. A specular variation could potentially be modeled by increasing the number of MoG components, however, this would increase the storage cost of appearance parameters for each voxel. Ulusoy *et al.* suggest that non-Lambertian surfaces' appearance is highly correlated with the viewpoint, thus can be better modeled by a *view-dependent* distribution. Thus, for a given observation I from viewpoint $\mathcal{V}$ only the distribution corresponding to that viewpoint will be updated with said observation *i.e.* $p_{\mathcal{V}}(I)$. However, storing a distribution for each possible viewpoint is highly impractical, as the parameter space scales with the number of available viewpoints. Moreover, training such distributions would require a significant number of observations per viewpoint in order to reach convergence. To address this problem, Ulusoy *et al.* introduce the concept of canonical viewpoints, a set of n vectors $\{\mathcal{V}_i\}^n$ known a priori which are distributed to roughly cover the same area as that of the camera views. Thus for a an arbitrary direction $\mathcal{V}$, the resulting appearance will be a weighted sum of appearances that are roughly in the same direction. A natural measure of proximity to the current view is the negative inner product $-\mathcal{V}_i \cdot \mathcal{V}$. This metric will attenuate the contribution of canonical directions that don't face the current view. Thus, the view-dependent appearance model is defined as,

$$p(I, \mathcal{V}) = \frac{1}{\sum \omega_i} \sum_{i=1}^n \omega_i p_{\mathcal{V}_i}(I) \quad (4.20)$$

$$\omega_i = \begin{cases} -\mathcal{V}_i \cdot \mathcal{V} & \text{if } \mathcal{V}_i \cdot \mathcal{V} < 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.21)$$

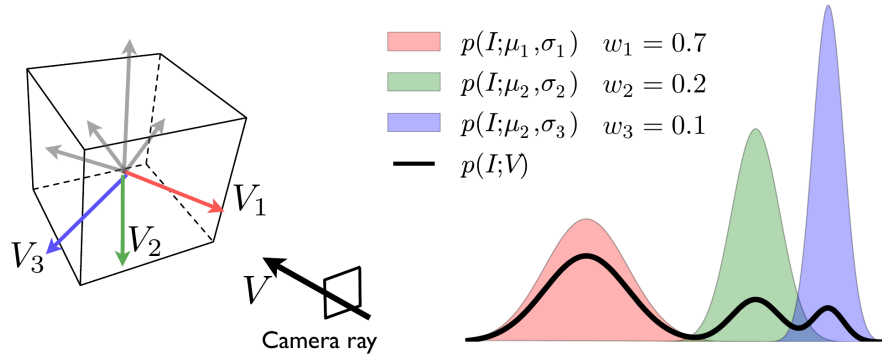


Figure 4.7: The view dependent appearance model as explained by by Ulusoy *et al.* [123]. Note how the grayed directions do not contribute to the final distribution $p(I; V)$, as they face away from the camera ray V . Since V_1 faces V almost directly, it will have the highest contribution w_1 while canonical directions V_2 and V_3 will have some nonzero contributions w_2 and w_3 , respectively

Pollard and Mundy demonstrated the robustness of the MoG when modeling intensity variation and dealing with occlusion across multiple views. However, such distribution for each canonical viewpoint will not be practical due to storage requirements and parameter estimation feasibility. More specifically, for a set of 8 canonical viewpoints modeled using a 3-component MoG per viewpoint the parameter space per voxel will be 88. The cost of storing such an appearance model for a 30M cell scene is prohibitive even for state of the art GPU hardware. Additionally, the number of viewpoints required to estimate the parameters of each view-dependent distribution has to be significantly larger as viewpoints with 0 weight do not contribute to a distribution's update as seen in Eq. (4.21). Consequently, Ulusoy *et al.* use the single Gaussian per canonical viewpoint thus reducing the parameter space by a factor of roughly 4. This enables amenable storage costs and feasible parameter estimation. Although, not as robust as the MoG when dealing with occlusions, the single Gaussian update can be improved by re-initializing the parameters during the initial iterations. The re-initialization will erase the parameters which might have been updated with observations from occluding surfaces. Ulusoy *et al.* tested the view-dependent appearance in time-varying 3-d reconstructions using multiple synchronized videos captured in an indoor studio setup. In particular, they use sequences from the INRIA 4-d repository that provide videos from 16 cameras distributed around a subject [2]. The proposed view-dependent distribution of surface appearance by Ulusoy *et al.* was shown to capture the view-dependent variations in surface appearance and enables reconstruction of reflective non-Lambertian surfaces as seen in 4.8. This is a major improvement over the MoG PVM appearance model and many other multi-view reconstruction algorithms which

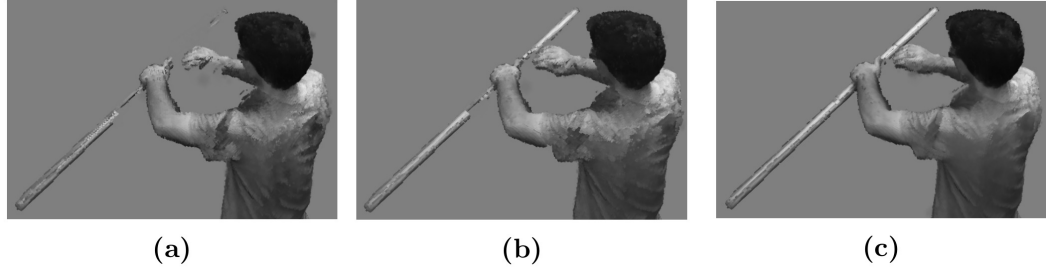


Figure 4.8: Renderings of scene where appearance has been modeled using (a) Single-Gaussian, (b) Mixture of Gaussians, (c) the View-dependent Appearance model. Image taken from Ulusoy *et al.* [123]

rely on the Lambertian surface reflectance assumption and therefore have difficulty reconstructing such reflective surfaces.

4.3.2 Occlusion Density and Visibility - How not to be Seen

As mentioned in Sec. 4.3, the implementation of the PVM on a non-uniform grid was introduced by Crispell [35]. In order to be able to describe the process of Optical Flow estimation on the non-uniform grid, the re-parametrization of surface voxels using occlusion density must be first presented. The initial step of this re-parametrization involves revisiting the concept of visibility. As seen in Eq. (4.6), each voxel intersected by the ray contributes its full probability amount to the product $\prod_i (1 - \Pr(\mathbf{X} \in S))$. Crispell proposed an alternative where the probability of each voxel $\Pr(\mathbf{X} \in S)$ is related to the length of the ray-voxel intersection. This way, voxels that contain larger segments of the intersecting ray will have a higher surface probability contribution, as seen in Fig. 4.10. In order to find the relationship between visibility and the length of ray-voxel intersections, Crispell first defines the probability of a point q being visible at distance s along a ray as $\Pr(\mathcal{V})$ or, as defined in the previous section $\text{vis}(s)$. Additionally the probability of the occluding point is introduced as $\Pr(\mathcal{Q}_s^l)$, *i.e.* point s is visible but point $s + l$ is occluded along the ray. The two concepts can be related as,

$$\Pr(\mathcal{Q}_s^l) = \Pr(\overline{\mathcal{V}_{s+l}} | \mathcal{V}_s) = 1 - \Pr(\mathcal{V}_{s+l} | \mathcal{V}_s) \quad (4.22)$$

$$\Pr(\mathcal{Q}_s^l) = 1 - \frac{\overbrace{\Pr(\mathcal{V}_s | \mathcal{V}_{s+l})}^1 \Pr(\mathcal{V}_{s+l})}{\Pr(\mathcal{V}_s)} = \frac{\text{vis}(s) - \text{vis}(s+l)}{\text{vis}(s)} \quad (4.23)$$

Taking the limit $\lim_{l \rightarrow 0} \mathbf{Pr}(\mathcal{Q}_s^l)$ and dividing by ds , the probability per unit length, *i.e.* the occlusion density, takes the following differential form,

$$\mathbf{Pr}(\mathcal{Q}_s^{ds})/ds = -\frac{d \text{vis}(s)}{\text{vis}(s)} = -\frac{\text{vis}'(s)}{\text{vis}(s)} \triangleq \alpha(s) \quad (4.24)$$

After integrating over the length of segment l , the probability that a point is visible over the interval $[0, l]$ is expressed as:

$$\text{vis}(l) = \exp \left(\int_0^l \alpha(s) ds \right) \quad (4.25)$$

Now that visibility is expressed as a function of intersection length and the density per unit length $\alpha(s)$, Crispell continues by expressing occlusion likelihood, $\mathbf{Pr}(\mathcal{Q}_s^l)$ using the same quantities. Specifically,

$$\mathbf{Pr}(\mathcal{Q}_s^l) = 1 - \exp \left(\int_s^{s+l} \alpha(s') ds' \right) \quad (4.26)$$

An important observation that Crispell makes is that occlusion probability, a continuous quantity, can be related to voxel surface probability by abandoning the ray geometry and using a canonical length as the domain of integration. Assuming the occlusion density is constant inside a voxel and using the voxel side-length $\hat{l}$ as the canonical length, a voxel's occluding likelihood or surface probability can be expressed as:

$$\mathbf{Pr}(\mathcal{Q}_s^l) = 1 - \exp(-\hat{l}\alpha) \quad (4.27)$$

Finally, using the re-parametrization of surface probability as occlusion density, the update formula from Eq. (4.7) can be expressed as:

$$\mathbf{Pr}(\alpha|I) = \mathbf{Pr}(\alpha_i) \frac{\text{vis}_i \mathbf{Pr}(I|s_i \leq V < s_{i+1}) \sum_{j=0}^{i-1} (1 - \exp(-\alpha_j l_j)) \text{vis}_j \mathbf{Pr}(I|s_j \leq V < s_{j+1})}{\sum_{j=0}^{\infty} (1 - \exp(-\alpha_j l_j)) \text{vis}_j \mathbf{Pr}(I|s_j \leq V < s_{j+1})} \quad (4.28)$$

Please note that the appearance model is now conditioned over the voxel that contains the ray intersection segment $[s_j, s_{j+1})$ as $\mathbf{Pr}(I|s_j \leq V < s_{j+1})$. Also, the visibility is computed using the discrete form of Eq. (4.25), namely

$$\text{vis}_i = \exp \left(- \sum_{j=0}^{i-1} \alpha_j l_j \right) \quad (4.29)$$

The motivation behind expressing the update and rendering equations as a function of partitioned

ray segments is tied to the adaptive refinement of the voxel grid. The scene surface parameters are a function of ray length intersections *i.e.* $\alpha(s)$ rather than being constant over a fixed-size voxel location, *i.e.* $\Pr(\mathbf{X} \in S)$ as in the case of the first implementation. Assuming that the occlusion density is constant within a voxel, splitting a voxel in half won't change the occlusion probability of the resulting two voxels. Thus a non-uniform sampling of 3-d space is enabled by this reparametrization. It should be noted that this continuous representation is essentially ray dependent and so model properties at a given voxel are dependent on the image viewing directions that provided update information to the model.

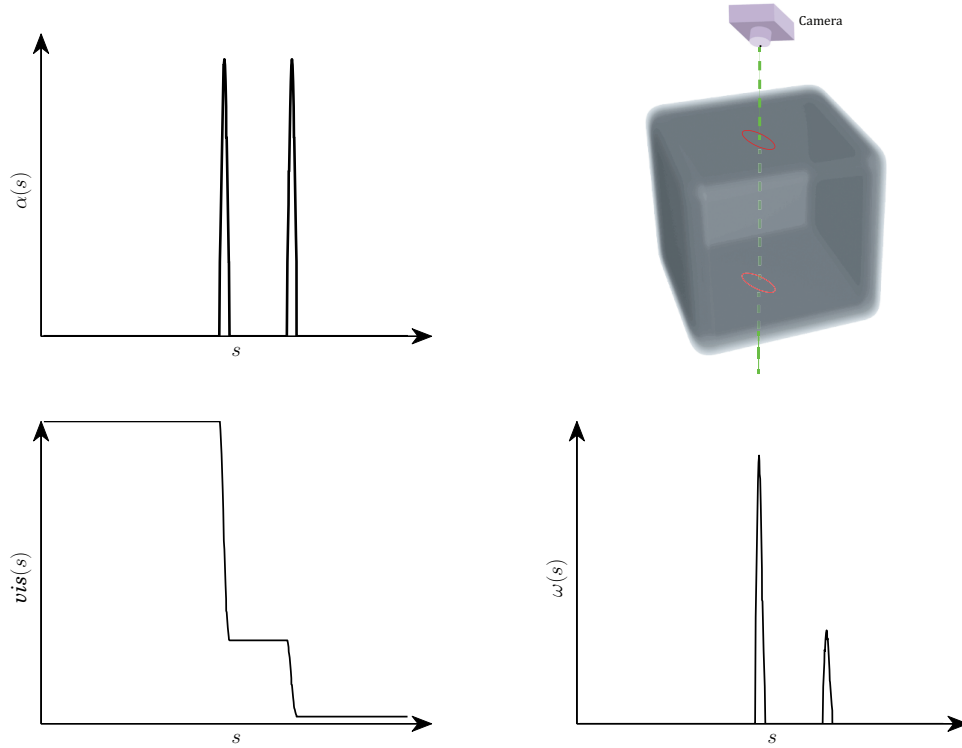


Figure 4.9: Quantities α , ω and vis corresponding to a ray piercing a hollow, transparent cube. The voxels corresponding to the cube faces have surface probabilities that are less than 1. Note how $\alpha(s)$ is maximal at the ray-cube intersection points. The visibility drops as the ray pierces the cube faces. Evidently, the intersection point on the top face is more likely to be the ray occlusion point, as seen in the $\omega(s)$ plot

The Expected Image A desirable property of the PVM is to be able to compute the expected intensity from a given ray R . This will enable the user to render the model from various perspectives, other than the ones used to train the model. In addition, the expected image can be used as a measure

of model quality which will be later presented during the evaluation of compression and optical flow. Before formally defining the expected image, the complement of visibility along a ray is introduced as $\Omega(s) = 1 - \text{vis}(s)$. This quantity represents the likelihood that the ray will become occluded at distance s from the camera center. Crispell defines this quantity as a probability per unit length, $\omega(s)$

$$\begin{aligned}\omega(s) &= \frac{d}{ds}\Omega(s) = \frac{d}{ds}(1 - \text{vis}(s)) \\ \omega(s) &= -\text{vis}'(s) = \alpha(s) \text{vis}(s)\end{aligned}\tag{4.30}$$

Note that the last line of the above equation was derived using the differential form of visibility defined in Eq. (4.24). It follows from the definition of ω in Eq. (4.30) that the point of occlusion along the camera ray needs to be both visible to the camera *and* have high surface likelihood. This traces back to the discrete random variable $V_{R(\mathbf{X})} = \mathbf{X}$ defined in Eq. (4.5), which represents the likelihood that voxel $\mathbf{X}$ was responsible for a given observation along a ray. Pollard and Mundy's assumption was that the responsible voxel had to be both visible and have high surface evidence which is equivalent to $\Omega(s)$. In order to better grasp the meaning of the quantities $\text{vis}(s)$, $\omega(s)$ and $\alpha(s)$ please refer to Fig. 4.9. It is also necessary to define the ancillary quantity $\text{vis}(\infty)$ which represents the likelihood that a ray passes through space unoccluded and is defined as:

$$\text{vis}(\infty) = 1 - \int_0^\infty \omega(s)ds\tag{4.31}$$

Previously, appearance distributions were presented in the context of model learning and updating and were evaluated at the observed intensity I via the $\mathbf{Pr}(\cdot)$ operator. By contrast, the expected image formulation will use the probability distribution function (PDF) of the appearance model. Specifically, $p_A(I|R, s)$ represents the appearance PDF of the voxel at distance s along camera ray R . Additionally, the corresponding appearance model of a ray R that passes unoccluded through space is defined as $p(I|R, \infty)$ and is typically modeled in practice by a uniform distribution over gray values. Finally, the intensity distribution of an image point parametrized by camera ray R is defined as,

$$\mathcal{P}(I|R) = \int_0^\infty \omega(s)p_A(I|R, s)ds + \text{vis}(\infty)p_A(I|R, \infty)\tag{4.32}$$

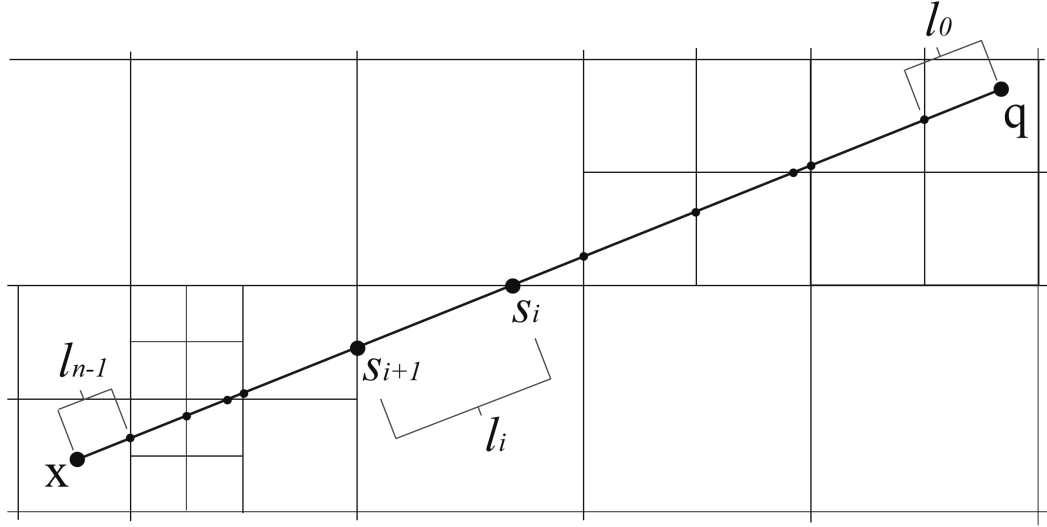


Figure 4.10: The partitioning of a ray into segments bounded by ray-voxel intersections, as illustrated by Crispell *et al.* [35]

The expected value of this distribution will yield a pixel value in the range $[I_0, I_1]$ for image-ray R ,

$$\mathbb{E}[\mathcal{P}(I|R)] = \frac{1}{I_1 - I_0} \int_{I_0}^{I_1} \omega(s) \int_0^\infty p_A(I|R, s) ds di + \text{vis}(\infty) p_A(I|R, \infty) di \quad (4.33)$$

$$= \int_0^\infty \omega(s) \mathbb{E}[p_A(I|R, s)] ds + \text{vis}(\infty) \mathbb{E}[p_A(I|R, \infty)] \quad (4.34)$$

In practice, however, the expected intensity is computed much easier through the discrete partitioning of the ray R into ray-voxel intersection segments as seen in figure 4.10.

When integrating the second relationship in Eq. (4.30) and assuming $\omega(s)$ is constant over the segment $[s_i, s_{i+1})$ the probability of occluding the ray is:

$$\Omega_i = \int_{s_i}^{s_{i+1}} \omega(s) ds = - \int_{s_i}^{s_{i+1}} \text{vis}'(s) ds = \text{vis}(s_i) - \text{vis}(s_{i+1}) \quad (4.35)$$

Thus the discrete form of Ω_i is the negative difference of two consecutive visibility probabilities. Using the discrete Ω_i 's at each voxel as weights, the discrete form of Eq. (4.33) is computed as the weighted sum of the expected values of each appearance distribution $p_A(I|R, s_i, s_{i+1})$ along the partitioned ray R ,

$$\mathbb{E}[I] = \text{vis}_\infty \mathbb{E}[p_A(I|R, \infty)] + \sum_{i=0}^{S+1} \Omega_i \mathbb{E}[p_A(I|R, s_i, s_{i+1})] \quad (4.36)$$



Figure 4.11: Expected Image Rendering of the Barus & Holley building at Brown University

Henceforth, the term ‘rendering’ in the context of PVMs will refer to the process of computing the expected image. A typical expected image is presented in Fig. 4.11. Expected images or *scene-renderings* will be used throughout this thesis to measure reconstruction quality when warping a scene instance into the next using the estimated flow.

4.3.3 GPU Implementation of the Non-Uniform Voxel Grid

Having presented the theoretical considerations of PVMs, it is necessary to give a brief overview of the GPU implementation. The spatial structure of the PVM is tailored for fast memory operations on the GPU allowing efficient neighborhood access. More specifically, A PVM’s voxel grid is implemented as a 3-d array of complete octrees, *i.e.* each octree node has either 8 children or none. This property allows the octree to be easily encoded as a binary word, each bit indicating whether a specific node has descendants. The k^{th} bit of the encoded word corresponds to the k^{th} node encountered during a breadth first search (BFS) traversal of the octree. For example, a complete binary bit-tree with 4 levels and its associated binary word is presented in Fig. 4.12. The bit under each node indicates whether that node has children. The placeholder “ $\times$ ” indicates nodes that do not exist in the tree. In this case, a 0 bit is placed in their corresponding location in the word. It is clear from this example that the bit corresponding to the first child of node i is $n \cdot i + 1$, where $n = 8$ for the case of the octree and $n = 2$ for the binary tree. Similarly, the parent of a node i is located at $\lfloor \frac{i-1}{n} \rfloor$. These formulas provide trivial access to neighboring nodes, precluding the need to store additional child or parent pointers. Moreover, the binary words can be stored in the fast, but limited GPU local memory. In order to take advantage of fast memory access via local registers, the tree-depth is limited to 4 which in turn limits the tree binary word length to less than 16 bytes. In order to access

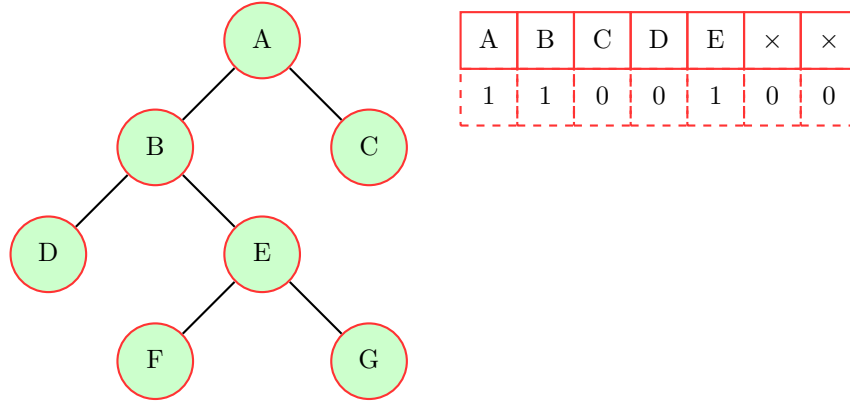


Figure 4.12: Binary bit tree and associated binary word. Note how C's descendants are marked with an $\times$ and their corresponding bit is set to 0

the appearance and surface parameters of a given node a lookup table is used to map bit indexes to pointers into the slower GPU global memory where these quantities are stored. A representation of this octree structure is shown in Fig. 4.13. A consequence of representing occlusion probability

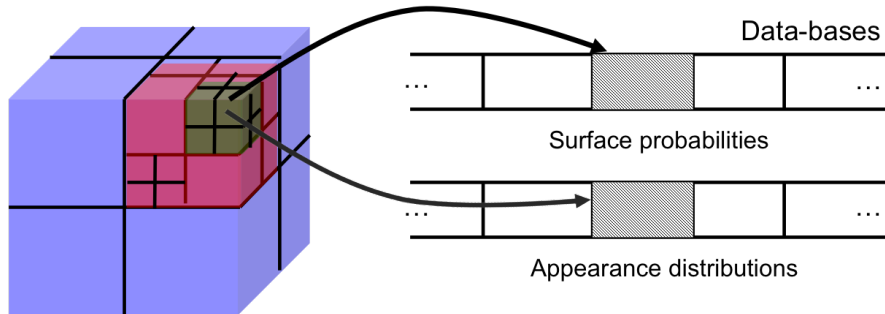


Figure 4.13: Octree representation and associated data buffers

as a density per unit length is the ability to refine only the cells of surface voxels . The scene is initialized as a uniform grid of root-only octrees. As soon as surface evidence for a given cell exceeds a threshold, that cell is split into its 8 children, each one inheriting their parent's occlusion density while their appearance information is reset. The refinement check occurs after each input image update. Miller *et al.* [93] provides an efficient scheme to refine octrees in parallel by keeping track of pointers into an array of pre-allocated size. This algorithm achieves speedup of 600X over the C++ implementation, parallellizing the ray image space for each input image, *i.e.* a GPU thread handled the update/expected value of all the voxels along a given ray . Thus 480p expected images are rendered

at 30 fps. It should be evident now that parametrizing voxels using $\alpha(s)$ conserves probability across 3-d space after the refinement operation. By integrating a child's inherited occlusion density over its side-length which is half the length of its parent, each child receives the same probability contribution. The encoding of occlusion density per voxel as well as the resulting refinement process will be essential during the optical flow computation which will be presented in the next chapter.

4.4 Conclusions

This chapter summarized various ways of representing 4-d space along with their feasibility in estimating motion for the purpose of compression. Due to its accurate representation of 4-d space and robustness in the presence of scene uncertainty and ambiguity the probabilistic volumetric model is chosen as the framework for this thesis. The underpinnings of training the parameters of the PVM using Pollard and Mundy's initial method were first presented. Next, the improvements brought by Ulusoy *et al.* and Crispell *et al.* in appearance modeling and surface likelihood reparametrization were discussed, respectively. Additionally, a method of rendering the PVM from arbitrary viewpoints is presented which is based on computing the weighted sum of expected appearances along a ray. This method will be used as a means of evaluating the quality of scene reconstruction using optical flow warping. A similar quantity, the expected flow, will be presented in the following chapters which will be used to render 3-d probabilistic flow onto the expected image. This theoretical foundation of PVMs will provide a complete picture to the context of optical flow estimation and compression in 4-d space.

Chapter 5

Multiresolution 3-d Optical Flow

5.1 Introduction

Progress in the accuracy and speed of 3-d reconstruction algorithms has motivated research in the construction and analysis of dynamic 3-d scenes [24, 71, 97, 123, 126, 127]. A key attribute of dynamic scenes is the instantaneous 3-d motion of surfaces in the scene. This motion field establishes correspondences between surfaces at two consecutive time steps and it is commonly referred to as *scene flow* [128].

This chapter presents a new method to estimate *dense* scene flow from calibrated multi-view image sequences. The proposed method reconstructs 3-d models independently for each time step using images synchronously captured from multiple viewpoints. Then, the method computes the 3-d flow between two consecutive 3-d models. The first contribution of the chapter is to formulate the scene flow problem using a volumetric and probabilistic representation of 3-d surface geometry and appearance. A typical result of the proposed algorithm is shown in Fig. 5.1. The 3-d models are encoded in a multi-resolution grid that stores occupancy probability in each voxel, as well as a surface appearance distribution. Here, surface appearance refers to the combination of surface reflectance and illumination giving rise to the pixel intensity or color seen in images of the surface. The second contribution of this work is presenting an improved method for scene reconstruction and its applications in compression. A desirable property of the optical flow field is a compact representation from which the inverse quantity can be derived. By modeling position and flow vectors with a Gaussian mixture model (GMM), a scene can be decomposed into regions of similar

motion. The flow field within each partition is assumed to be locally affine and consequently fitted to a 12-parameter transform. Using the sets of affine transformations, a receiver can then reconstruct scene instances after the first one, obviating the need to send the full model at every time step.

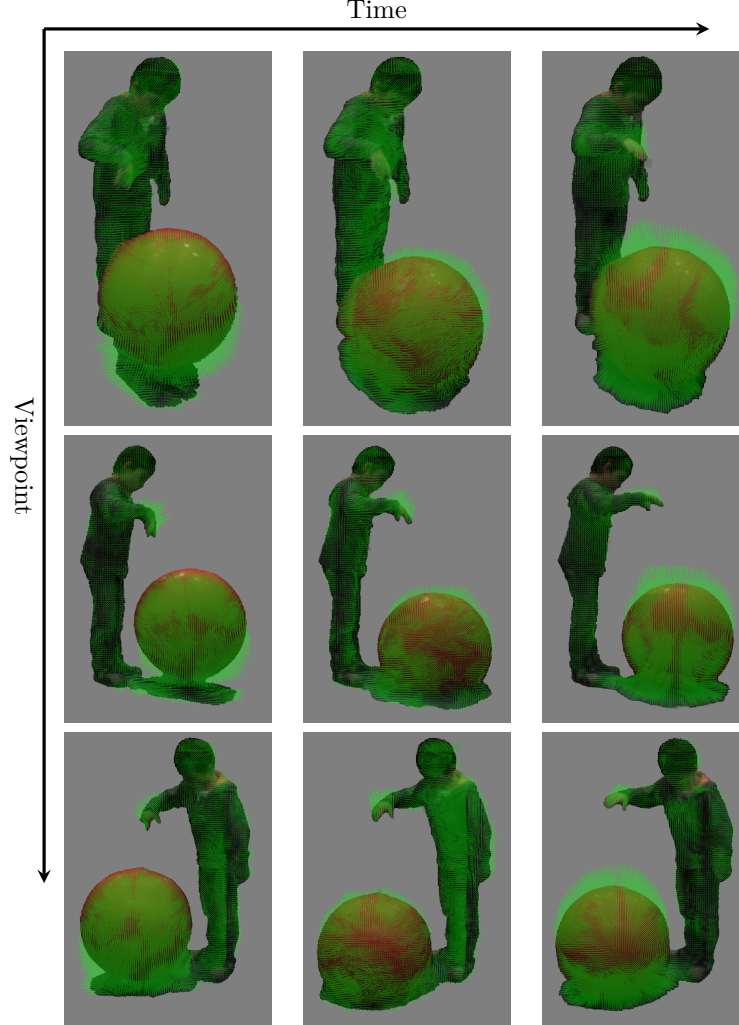


Figure 5.1: The estimated 3-d motion is visualized as 2-d line projections. The figure shows flow results from three consecutive frames of the “Boy playing ball” sequence. Note that the flow vectors correctly point in the direction of the ball’s motion.

The choice of 3-d model representation yields several advantages in estimating scene flow. First of all, the representation encodes surface occupancy and appearance information in a dense manner, *i.e.* estimates are available everywhere in the volume. This representation yields an analog of the image domain in 3-d. An image encodes the perspective projection of surface appearance in a 2-d regular pixel grid. This representation encodes probabilistic estimates of surface locations and their

appearances in a 3-d voxel grid. This 3-d volumetric image representation enables the computation of scene flow by classical optical flow methods. In this paper, the Horn and Schunck formulation for optical flow [63] is adapted to compute dense scene flow in such 3-d models. It is emphasized that the adaptation is not straightforward, given the probabilistic 3-d appearance and surface representation of the PVM.

A second motivation is the recent progress in multi-view 3-d reconstruction algorithms that estimate such representations from input images. In particular, Ulusoy *et al.* demonstrate that high quality 3-d models which achieve free-viewpoint video realism can be obtained for the datasets used in this chapter [123]. The results presented here utilize 3-d models obtained with their method. The probabilistic treatment of this 3-d reconstruction method not only helps achieve better quality 3-d models as demonstrated earlier [25, 104] but also allows retaining the uncertainties of reconstruction from images, which is utilized by the subsequent scene flow algorithm.

The 3-d representation is stored in an octree, where fine resolution voxels are allocated only near surface locations. This adaptive decomposition of space is essential to manage computational requirements and in particular, processing in the GPU [35, 93, 123]. This embedded hierarchical representation naturally provides a structure, similar to an image pyramid, to compute scene flow using a coarse-to-fine scheme. As known from research in optical flow, such hierarchical computation is essential to recover large motion displacements [117]. Results indicate the motion of fast moving objects, such as the hands of the dancer in Figure 5.3 can be captured using this scheme.

Finally, this volumetric representation allows tracking surfaces undergoing changes in surface topology. In contrast, many related methods utilize a mesh representation [6, 23, 37, 46, 48, 142]. As seen in Sec. 4.1 The mesh representation complicates recovering motions that violate the topology of the initial mesh and therefore such methods cannot deal with changes in topology. This advantage of volumetric representations has also been a motivating factor for level-set methods [34, 96].

This chapter presents a new formulation of dense scene flow using probabilistic and volumetric 3-d models. To the best of the authors' knowledge, the results demonstrate the densest flow fields achieved in current datasets. Unfortunately, there exist no ground truth dataset to measure the accuracy of the flow results. Instead, this work presents novel-view rendering experiments where the scene flow that is computed between two models t and $t + 1$ is used to warp the model at t and then compare the result with the $t + 1$ model. High quality rendering results obtained in these experiments are indicative of the accuracy of the scene flow estimates. This method of validation

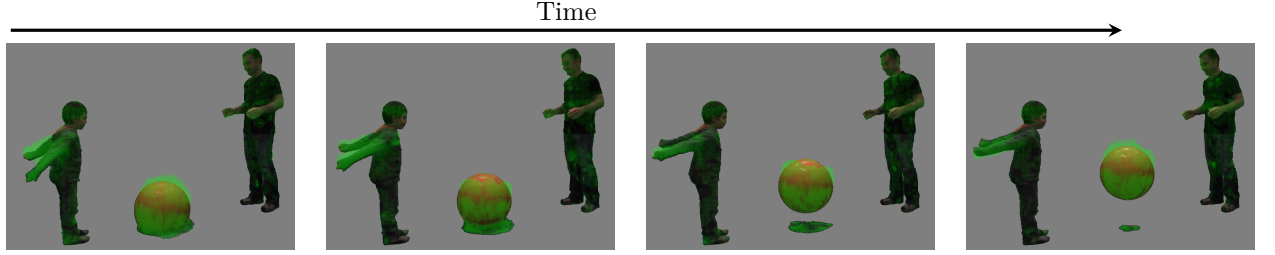


Figure 5.2: Estimated 3-d flow projected using the Expected-flow method onto novel view rendering of consecutive time instances of the “Adult Child Ball” sequence. Despite the ball’s large motion (3-4 voxels at the coarsest level or 12 cm) the flow is recovered properly

fits perfectly with the goals of scene reconstruction and compression of this thesis. More specifically, it is desirable to measure the quality of scenes reconstructed using the re-parametrized flow and the original flow itself. The affine re-parametrization process is a novel method that significantly reduces flow field storage requirements and enables a more efficient scene warping technique.

5.2 Related Work in Scene Flow

Scene flow is a highly studied problem in computer vision. This section reviews some of the most related works for the purposes of this chapter.

In recent years, there has been renewed interest in scene flow algorithms, particularly using stereo [42, 111, 124, 130] or depth sensor input [54, 61, 64, 81, 103]. Since the input data is from one or two views, these methods work with a view-centric 2.5-d surface representation, *i.e.* a depth map. This thesis considers input from more than two cameras distributed roughly around the subject, which allows producing much more complete 3-d models. This kind of data is typically obtained in indoor studio settings and the datasets used in this paper are freely available at the INRIA 4-d repository [2]. This type of input data allows building dense 3-d models and recover dense scene flow.

Most methods that consider similar input data work with a mesh-based surface representation and track the vertices of a base mesh obtained either through multi-view stereo in the first frame [24, 46] or a detailed laser scan of the subject prior to recording [6, 37, 48]. The vertices are typically tracked using image pixel intensities/colors, silhouette inputs while penalizing implausible deformations. As mentioned earlier, the mesh representation complicates tracking motions that change the mesh topology. Current research show robust topological change detection requires highly clean and

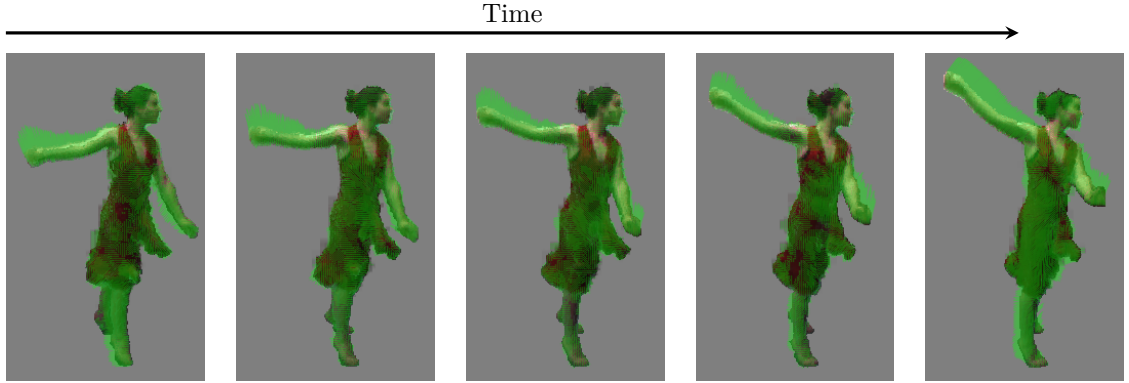


Figure 5.3: Estimated 3-d flow projected onto Novel-view rendering of consecutive time instances of the “Dancer” sequence. The resultant flow in this sequence matches accurately the voxels in the next time instance. Full video Sequence available in the supplementary material

manifold meshes [15]. It remains to be seen whether these methods can be applied to uncertain and ambiguous 3-d data obtained from sparse camera setups considered in this paper.

Point clouds provide an alternative representation [26, 55, 71]. While they can be used to model arbitrarily changing shapes, point clouds obtained from multi-view camera setups are typically sparse. Recent work by Park *et al.* show robust tracking of thousands of points can be achieved using 480 cameras. By contrast, the work reported here focuses more on traditional setups containing at most 16 cameras while producing highly dense motion estimates as shown in Figures 5.1 and 5.3.

Volumetric models have also been proposed for 3-d motion estimation. Basha *et al.* utilize a volumetric image similarity cost function to estimate scene flow [11]. Their method doesn’t explicitly compute surface occupancy and therefore cannot deal with occlusions. Guan *et al.* use silhouette inputs and solve for a 3-d model and flow vectors jointly [53]. The method presented here would potentially benefit from such a joint approach too. Both works [11, 53] utilize an MRF-based deformable registration algorithm by Glocker *et al.* [49] which produces a discrete set of flow vectors. In contrast, the flow formulation in this thesis is continuous.

The rest of the chapter is organized as follows. Section 4.3 contains background information on the 3-d models and associated reconstruction algorithms. The scene flow formulation is presented in Section 5.3. Section 5.4 demonstrates a method of flow rendering in a probabilistic context while synthetic results for optical flow estimation are presented in Section 5.5. The chapter is concluded with a summary in Section 5.6.

5.3 Coarse-to-fine Scene Flow

This section describes the 3-d multiscale adaptation of Horn and Schunck [63] to the probabilistic volume model representation. The continuous formulation of Horn and Schunck lies at the theoretical basis of most modern dense flow algorithms making it an attractive choice for an initial implementation on the PVM. Improvements coming from feature matching and penalty functions involve significant overhead in a GPU application with potentially only minor gains for the purposes of this paper. According to [10], the normalized interpolation error of Horn and Schunck evaluated on realistic, high frame rate (HFR) dynamic scenes is on par with state of the art methods. Compared to the other error metrics in [10], normalized interpolation error is the most relevant to the objectives of reconstruction and compression of this paper. The best method within this evaluation subset has an improvement in error of about 9% on average and 3% for 3 out of the 4 HFR datasets, which doesn't warrant the added complexity.

Since the voxel-world model stores data at each node in the tree, a multi-scale approach to Horn and Schunck follows immediately. The data term is similar to the one in the classical Horn and Schunck formulation, however, image intensity is replaced by the expected appearance measure [104] which is the expected value of the appearance model I at a particular voxel $\mathbf{X}$ multiplied by its surface probability $P(\mathbf{X} \in S)$. Assuming collected imagery is photoconsistent, the 3-d model appearance and probability distributions are invariant to motion as in the optical flow brightness constancy assumption. Moreover, the expected value operator takes into account evidence from all the images at a particular location, providing added robustness in the presence of image-based uncertainty. Hence the expected appearance is the appropriate substitute for image intensity in the data term of the functional.

$$\mathcal{E}(\mathbf{X}) \triangleq \mathbb{E}[I(\mathbf{X})] P(\mathbf{X} \in S) \quad (5.1)$$

Thus the Horn and Schunck energy functional becomes in 3-d:

$$E = \iiint_V (\mathcal{E}(\mathbf{x} + \mathbf{r}, t + 1) - \mathcal{E}(\mathbf{x}, t))^2 + \alpha^2 (||\nabla u||^2 + ||\nabla v||^2 + ||\nabla w||^2)^2 d\mathbf{v} \quad (5.2)$$

For convenience, denote the derivatives of the warped scene as such:

$$\mathcal{E}_x \triangleq \frac{\partial \mathcal{E}(\mathbf{x} + \mathbf{r})}{\partial x} ; \mathcal{E}_y \triangleq \frac{\partial \mathcal{E}(\mathbf{x} + \mathbf{r})}{\partial y} ; \mathcal{E}_z \triangleq \frac{\partial \mathcal{E}(\mathbf{x} + \mathbf{r})}{\partial z} \quad (5.3)$$

Minimizing the functional in (5.2) via Euler-Lagrange, the update equations for the incremental flow take a similar form in 3-d as the ones in 2-d presented in [63]:

$$\begin{aligned} u^{(k)} &= \bar{u}^{(k-1)} - \mathcal{E}_x \frac{\mathcal{E}_x \bar{u}^{(k-1)} + \mathcal{E}_y \bar{v}^{(k-1)} + \mathcal{E}_z \bar{w}^{(k-1)}}{\alpha^2 + \mathcal{E}_x^2 + \mathcal{E}_y^2 + \mathcal{E}_z^2} \\ v^{(k)} &= \bar{v}^{(k-1)} - \mathcal{E}_y \frac{\mathcal{E}_x \bar{u}^{(k-1)} + \mathcal{E}_y \bar{v}^{(k-1)} + \mathcal{E}_z \bar{w}^{(k-1)}}{\alpha^2 + \mathcal{E}_x^2 + \mathcal{E}_y^2 + \mathcal{E}_z^2} \\ w^{(k)} &= \bar{w}^{(k-1)} - \mathcal{E}_z \frac{\mathcal{E}_x \bar{u}^{(k-1)} + \mathcal{E}_y \bar{v}^{(k-1)} + \mathcal{E}_z \bar{w}^{(k-1)}}{\alpha^2 + \mathcal{E}_x^2 + \mathcal{E}_y^2 + \mathcal{E}_z^2} \end{aligned}$$

This iterative scheme is applied several times per resolution level. After each pass and resolution, the computed increment $(u^{(k)}, v^{(k)}, w^{(k)})$ is added to a quantity representing the overall displacement which is then used to warp the scene.

5.3.1 Derivative Estimation on a Non-uniform Volumetric Grid

The expected surface gradient $\nabla \mathcal{E}$ is computed with a bank of three isotropic 3x3x3 kernels representing the first derivative w.r.t. to each canonical direction of a Gaussian with mean 0 and $\sigma = 1$. The sizes are expressed in voxels at the coarsest level. When dealing with variable-sized grid filtering, the occlusion density parametrization [35], helps estimate surface probability contributions from larger cells that are not refined up to the current resolution level. The density is simply integrated over the length of the voxel at the current resolution. The Laplacian of the incremental flow field derived from the Euler-Lagrange minimization is approximated at every location with the difference between the neighborhood average and the value at the location, i.e. for the x component of the flow the Laplacian is $\nabla^2 u = \bar{u} - u$.

5.3.2 Scene Warping using Estimated Flow

The unknown flow solved for at resolution level ℓ of the 3-d model at a particular voxel $\mathbf{x}$ is $\mathbf{v}^{(\ell)}(\mathbf{x}) = (u, v, w)$. For the purposes of compression and tracking, a direct mapping from t to $t + 1$ at every voxel is essential, thus the *resultant* flow is defined as :

$$\mathbf{r}^\ell(\mathbf{x}) = \mathbf{r}^{\ell-1}(\mathbf{x}) + \mathbf{v}^\ell(\mathbf{x} + \mathbf{r}^{\ell-1}) \quad (5.4)$$

This process produces a direct mapping between scene instances since it's equivalent to tracking

all voxel positions throughout the iterative warping process.

The incremental flow obtained at a particular resolution $\ell - 1$ is resampled to the next resolution level and the resultant flow $\mathbf{r}^\ell$ is evaluated according to (5.4). At each level ℓ the original scene is warped according to $\mathbf{r}^\ell$, establishing the initial conditions for the optical flow calculation at level ℓ .

The warping is performed on the actual model thus the problem of resampling random variables has to be addressed. This problem has been treated in detail in Sec. 3.1.3 treated this problem in detail thus the random variables modeling a voxel's appearance can be resampled accordingly. Since surface probabilities are parametrized by α they are not modeled by a random variable, but are rather individual samples. For a view-dependent distribution located at voxel $\mathbf{X} = (x, y, z)$, parametrized by the set of means and variances μ_i , σ_i corresponding to each of the C canonical direction define:

$$P(x, y, z) \triangleq \{P(I|\mu_i, \sigma_i)\}_{i=0}^{C-1} \quad (5.5)$$

Note that the scalar multiplication or r.v's $(\star)$ defined in Eq (3.15) applies to each canonical Normal distribution of the view-dependent appearance model as such

$$s \star P(I|\mu_i, \sigma_i) = P(I|s\mu_i, s^2\sigma_i) \quad 0 \leq i < C \quad (5.6)$$

Recall that the sum of two distributions is equal to their convolutions $(\otimes)$ in Eq. (3.10). Thus the process of trilinear interpolation of appearance distributions is performed as such:

$$\begin{array}{c} w_x = \frac{x - x_0}{x_1 - x_0} \quad w_y = \frac{y - y_0}{y_1 - y_0} \quad w_z = \frac{z - z_0}{z_1 - z_0} \\ \hline P_{X_{00}} = (1 - w_x) \star P(x_0, y_0, z_0) \otimes w_x \star P(x_1, y_0, z_0) \\ P_{X_{10}} = (1 - w_x) \star P(x_0, y_1, z_0) \otimes w_x \star P(x_1, y_1, z_0) \\ P_{X_{01}} = (1 - w_x) \star P(x_0, y_0, z_1) \otimes w_x \star P(x_1, y_0, z_1) \\ P_{X_{11}} = (1 - w_x) \star P(x_0, y_1, z_1) \otimes w_x \star P(x_1, y_1, z_1) \\ \hline P_{Y_{00}} = (1 - w_y) \star P_{X_{00}} \otimes w_y \star P_{Y_{10}} \\ P_{Y_{01}} = (1 - w_y) \star P_{X_{01}} \otimes w_y \star P_{Y_{11}} \\ \hline P(x, y, z) = (1 - w_z) \star P_{Y_{00}} \otimes w_z \star P_{Y_{01}} \end{array} \quad (5.7)$$

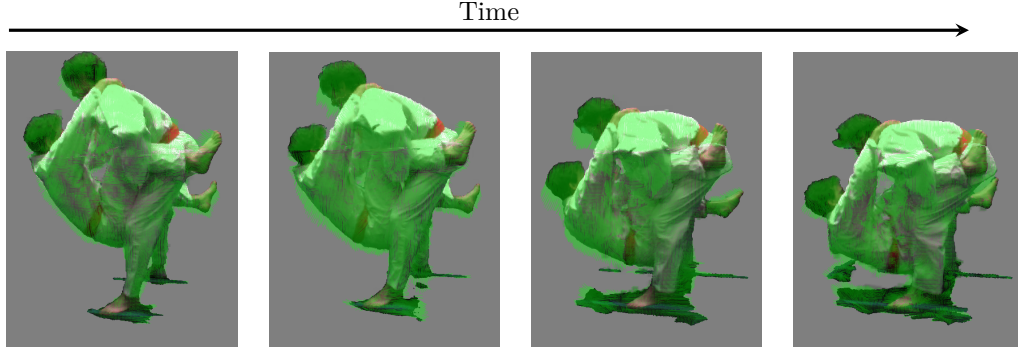


Figure 5.4: Estimated scene flow projected onto Novel-view renderings of consecutive time instances of the “Guard Punch Two” sequence. The flow is estimated robustly even in areas where surface geometry is poorly reconstructed due to lack of camera coverage

The final optical-flow algorithm for the PVM can be summarized as follows: The motion tolerance τ_m and the maximum iterations MAX_ITER are user defined. The maximum levels $L = 4$ of the octree are set to 4.

Algorithm 2: 3-d Scene Flow on PVMs

Input : Scene $\mathcal{S}_t, \mathcal{S}_{t+1}, \tau_m$ where $0 \leq \ell < L$
Output: Flow field $\mathbf{r} = \{(u_p, v_p, w_p) \mid p \in \mathcal{S}_t\}$

```

1 for  $\ell \leftarrow 0$  to  $L$  do
2    $i \leftarrow 0$ 
3   while ( $d\mathbf{r} < \tau_m \vee i < MAX\_ITER$ ) do
4      $\nabla \mathcal{S}_{t+1}^\ell = [\mathcal{E}_x, \mathcal{E}_y, \mathcal{E}_z, \mathcal{E}_t]^\ell = \text{SceneGradient}(\mathcal{S}_{t+1}^\ell)$ 
5      $d\mathbf{r} = \text{HornAndSchuckIteration}(\nabla \mathcal{S}_{t+1}^\ell)$ 
6      $\mathbf{r} \leftarrow \mathbf{r} + d\mathbf{r}$ 
7      $\mathcal{S}_{t+1}^\ell \leftarrow \text{Warp}(\mathcal{S}_{t+1}^\ell, \mathbf{r})$ 
8   end
9 end
10 return  $\mathbf{r}$ 

```

5.4 Expected Flow in 2-d

Once the algorithm produced the resulting flow field $\mathbf{r}$, it can be projected onto a 2-d image for visualization. Each 3-d point along a camera ray R will project into the same 2-d point $\mathbf{i}$ in the image view with corresponding camera $\mathcal{K}$. Thus the origin of all 3-d flow vectors along ray R will be at $\mathbf{i}$. However, the direction/ end point will be most likely different for each 3-d point along R . The question is which particular flow vector end-point should be projected onto the image from all

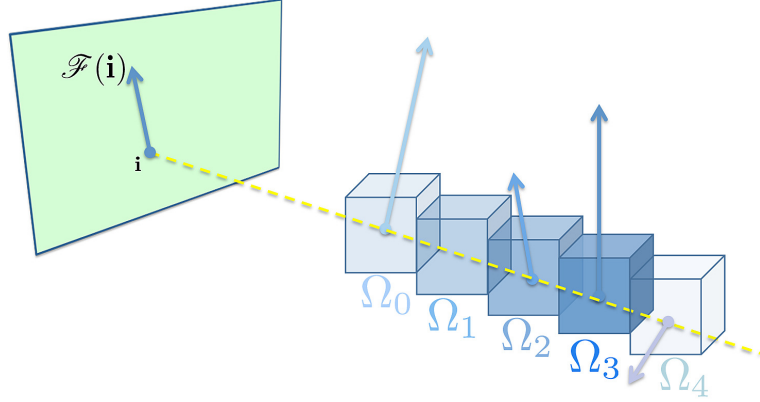


Figure 5.5: The expected flow is computed as the weighted sum of projected flow vector end-points. The weights represent the likelihood that a voxel is occluding the viewing ray

the possible flow vectors on the ray. Since the PVM maintains a surface likelihood, it makes sense to consider the point along the ray that is occluding the rest. Recall that the quantity $\Omega(s)$ presented in Sec. 4.3.2 modeled the probability that the point at distance s along the viewing ray is the occluding point. It follows therefore to use $\omega(s)$, *i.e.* the occluding density per unit length as the weight of each flow value end-point along the viewing ray, as seen in Fig 5.5. Thus the expected flow end-point at the local 2-d image view coordinate $\mathbf{i}$, is computed as

$$\mathcal{F}(\mathbf{i}) = \int_0^\infty \omega(s) \mathcal{K} \cdot (\mathbf{p}(s) + \mathbf{r}(\mathbf{p}(s))) ds \quad (5.8)$$

Note that the scene coordinates of 3-d point $\mathbf{p}$ located at distance s along the ray is $\mathbf{p}(s)$. Thus when the ray is partitioned into voxel-ray intersections, the discrete form of (5.8) is:

$$\mathcal{F}(\mathbf{i}) = \sum_{i=0}^{S+1} \Omega_i \mathcal{K} \cdot (\mathbf{X}(s_i, s_{i+1}) + \mathbf{r}(\mathbf{X}(s_i, s_{i+1}))) \quad (5.9)$$

It should be evident that this is similar to equation (4.36), where the 3-d scene coordinates of the voxel containing the ray segment $[s_i, s_{i+1})$ are defined as $\mathbf{X}(s_i, s_{i+1})$.

In order to offer a qualitative comparison, a number of expected flow images are compared with 2-d flow evaluations using state of the art methods. Specifically, the latest OpenCV 3.0 implementation [19] of the Brox Flow [22], Farneback Flow [40] and TV-L1 Flow [138] was applied to scene renderings. Results are presented in Figures 5.6, 5.7, 5.8, 5.12, 5.13. It is clear that in most cases, the expected flow performs better. The method proposed by Brox *et al.* recovers flow accurately,

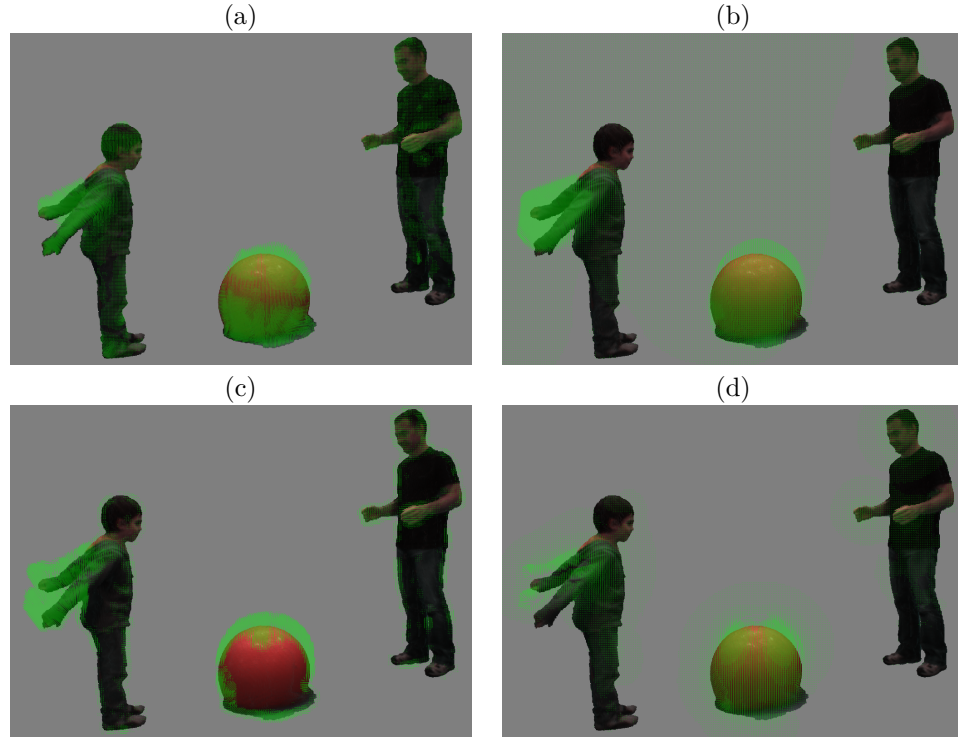


Figure 5.6: Adult Child Ball 2-d Flow compared methods: Proposed Expected Flow (a), Brox (b) [22], Farneback (c) [40], TV-L1 (d) [138]

especially for the “Boy Playing Ball Sequence”. However, their method is highly expensive in terms of computation, even in 2-d which precludes the feasibility of a 3-d GPU implementation tailored to the PVM for the time being.

5.5 Synthetic Data

In order to obtain direct 3-d validation of the flow result, a synthetic data experiment that consists of a rotating cube, is also created. The accuracy of the resulting flow field is measured for a range of rotation angles. A cube twenty voxels wide at the coarsest level was rotated by amounts ranging from 5 degrees to 45 degrees and the flow was computed for each rotation. In Figure 5.11, the synthetically rotated scene along with the original scene warped with the resultant flow are shown. For higher rotation amounts, it is clear from the produced artifacts that the scene displacement cannot be recovered accurately by the algorithm. However, for small rotation angles, the rendering of the warped scene matches the one of the target scene. The angular error between the estimated

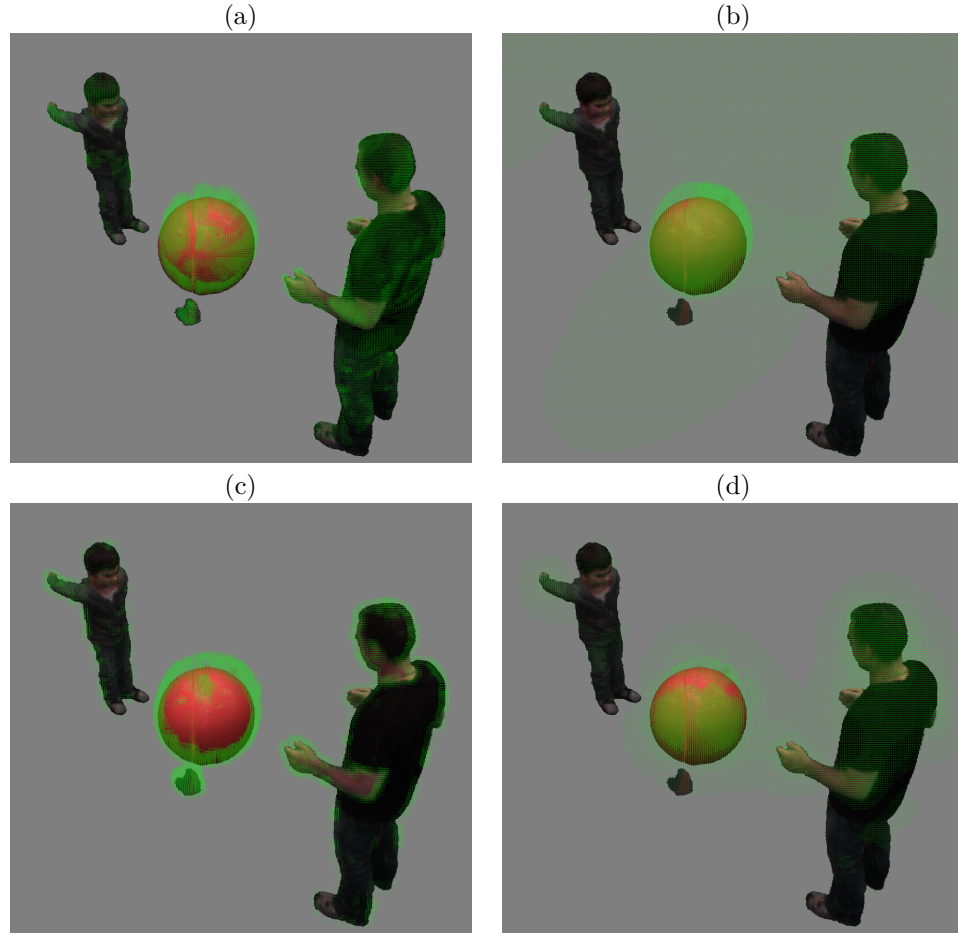


Figure 5.7: Adult Child Ball 2-d Flow compared methods: Proposed Expected Flow (a), Brox (b) [22], Farneback (c) [40], TV-L1 (d) [138]

flow field and the field corresponding to small rotations exceeded no more than 10%. The flow vectors for each rotation are displayed in Figure 5.10. The flow error is quantified again using the SSIM measure as above and the results are presented in Table 5.1.

As an additional quantitative evaluation, the synthetic cube was translated by various amounts ranging from 2 to 8 voxels along each of the 3 canonical axes. Typically, error in estimated optical flow is measured as the angle between the estimated flow and the true flow, or, as the magnitude of the difference between the estimated flow and the true flow. These quantities are defined as angular error and endpoint error, respectively and are plotted in Fig. 5.9 against the maximum displacement of the synthetic cube.

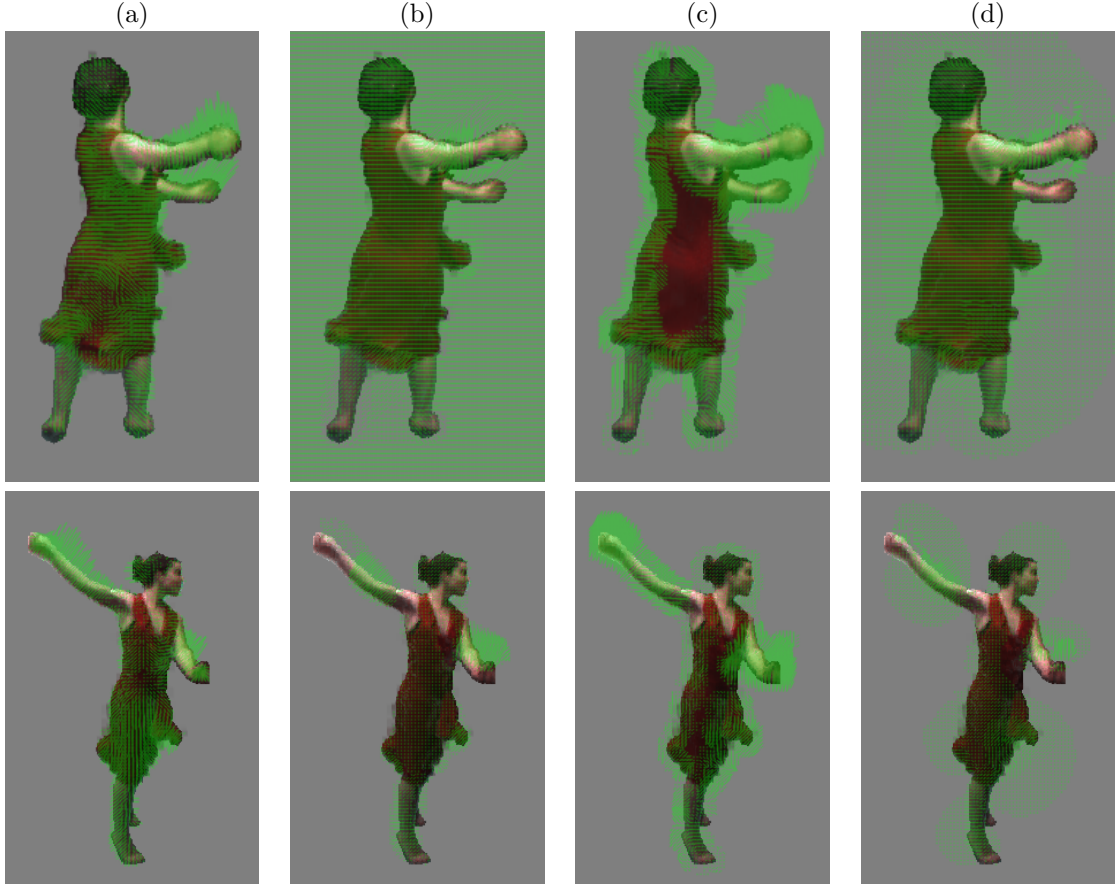


Figure 5.8: Dancer 2-d Flow compared methods: Proposed Expected Flow (a), Brox (b) [22], Farneback (c) [40], TV-L1 (d) [138]

5.6 Conclusions

This chapter presented a novel method to compute scene flow from synchronized multi-view image sequences. The first important contribution of the method is to utilize probabilistic and volumetric (dense) 3-d models for scene flow estimation. Formulating scene flow in this context offers a number of advantages over previous works. Most importantly, the presented flow field results are the densest to the best of the author’s knowledge. Moreover, the method can track the motion of surfaces undergoing topological change, which is a highly difficult problem for methods that use mesh representations. While the datasets used in the experiments do not provide scene flow ground truth, the results presented qualitatively show the high accuracy flow obtained by the method. In addition, qualitative comparisons of projected expected flow and state of the art methods were presented, indicating the accuracy of the proposed algorithm.

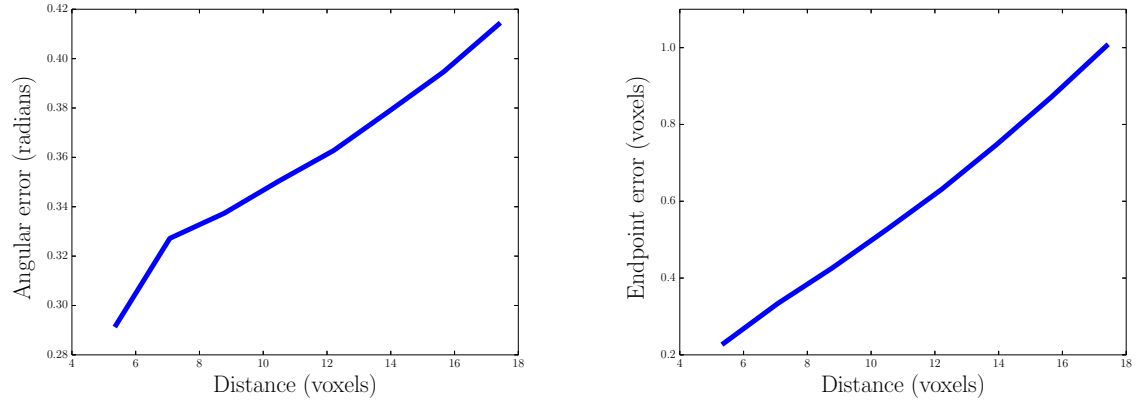


Figure 5.9: Flow Angular error in radians (top) and End-point error in voxels (bottom) as a function of maximum displacement in voxels for the synthetic cube

Rotation amount	5°	10°	30°	45°
Score	0.973	0.964	0.902	0.873

Table 5.1: SSIM measure comparing renderings of the cube warped with the estimated flow and the synthetically rotated cube (target scene)

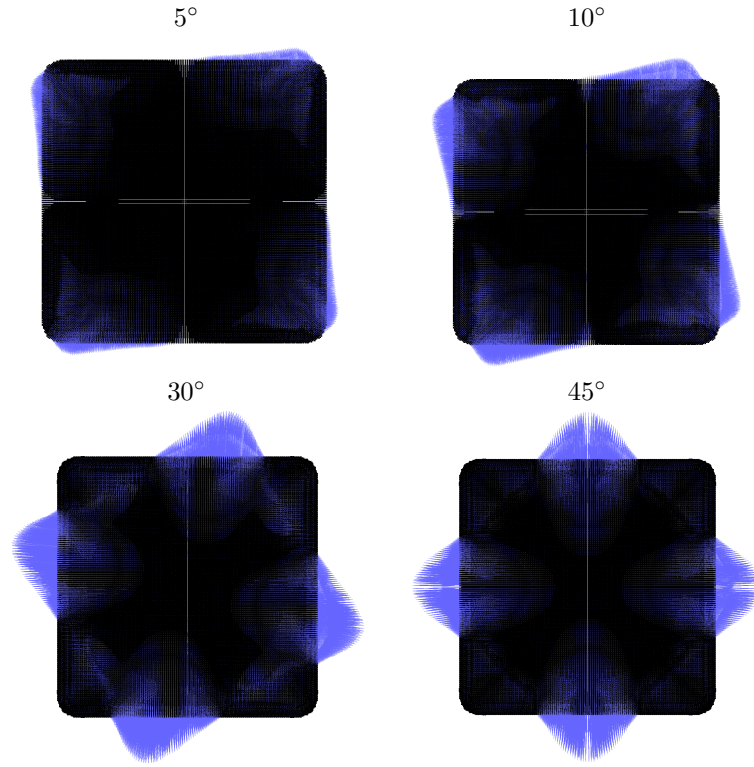


Figure 5.10: Estimated flow for synthetic cube for various amounts of rotation

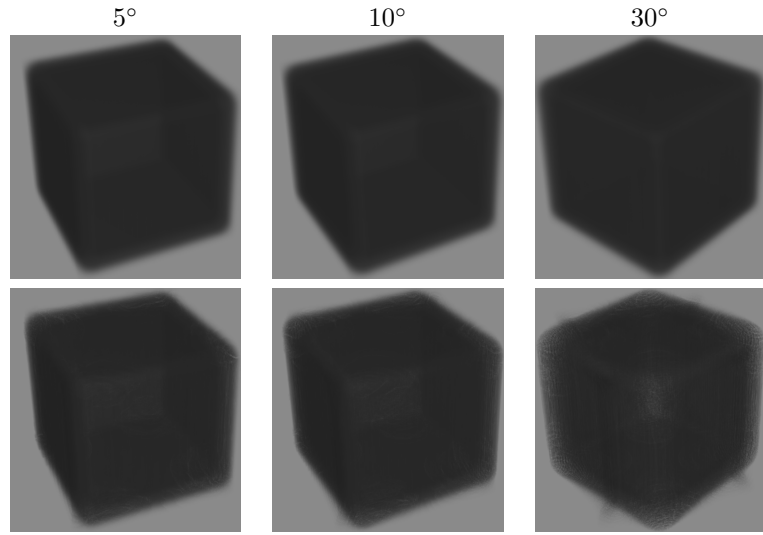


Figure 5.11: Synthetic Cube rotation and warping : Synthetically rotated (top), Warped with resultant flow (bottom)

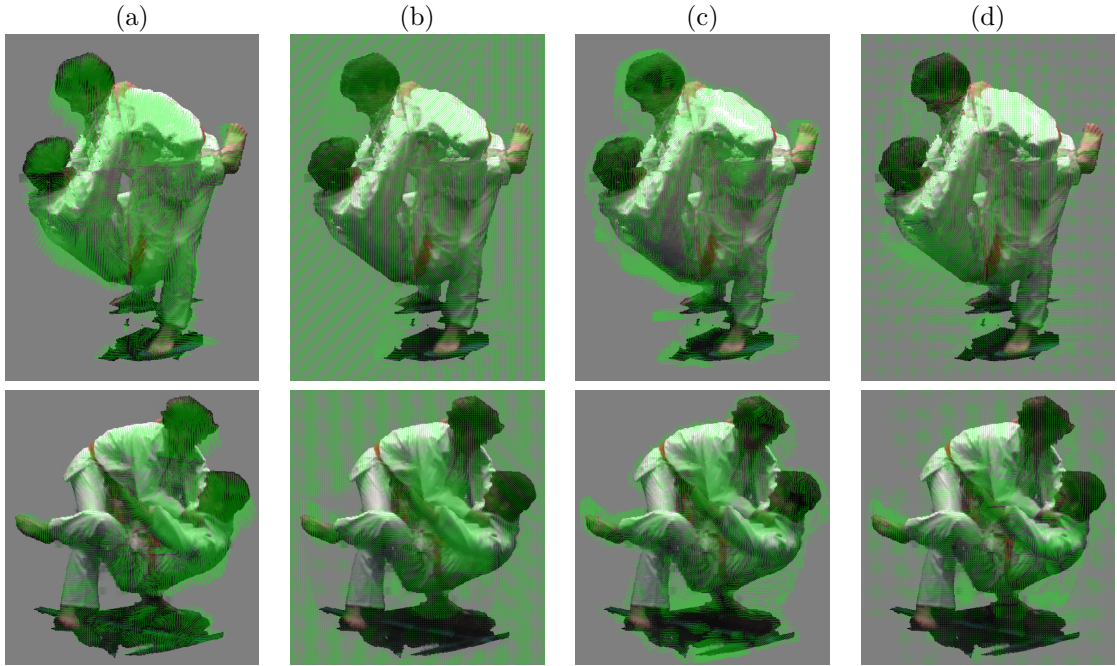


Figure 5.12: Guard Punch Two 2-d Flow compared methods: Proposed Expected Flow (a), Brox (b) [22], Farneback (c) [40], TV-L1 (d) [138]

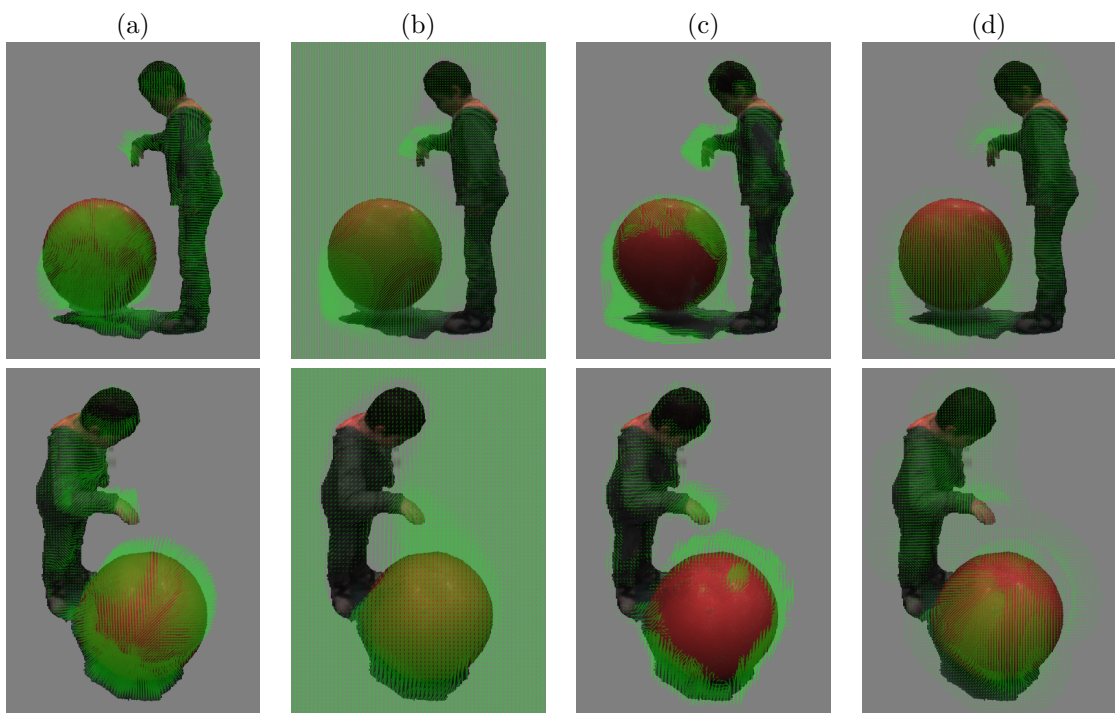


Figure 5.13: Boy Playing Ball 2-d Flow compared methods: Proposed Expected Flow (a), Brox (b) [22], Farneback (c) [40], TV-L1 (d) [138]

Chapter 6

Probabilistic Model

Reconstruction via Estimated Motion

Motion provides an effective way of partitioning the scene into regions of similar spatio-temporal characteristics. To demonstrate this benefit, an Expectation-Maximization (EM) algorithm is proposed to cluster spatially close voxels with similar flow vectors. In particular, the algorithm estimates the parameters of a K -component Gaussian mixture model (GMM) over the feature $\mathbf{f} = (x, y, z, u, v, w)$ [14]. The cluster means, covariances and weights $\boldsymbol{\theta} = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i^2, \omega_i\}$ are estimated using the standard EM equations 6.1 and 6.2 iteratively until convergence on the all N voxels in the scene.

$$\omega_i = \frac{1}{N} \sum_{k=1}^N \Pr(\mathbf{f}_k|i, \boldsymbol{\theta}) \quad \boldsymbol{\mu}_i = \frac{\sum_{k=1}^N \Pr(\mathbf{f}_k|i, \boldsymbol{\theta}) \mathbf{f}_k}{\sum_{k=1}^N \Pr(\mathbf{f}_k|i, \boldsymbol{\theta})} \quad (6.1)$$

$$\boldsymbol{\Sigma}_i = \frac{\sum_{k=1}^N \Pr(\mathbf{f}_k|i, \boldsymbol{\theta}) \mathbf{f}_k \mathbf{f}_k^t}{\sum_{k=1}^N \Pr(\mathbf{f}_k|i, \boldsymbol{\theta})} - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^t \quad (6.2)$$

The initial cluster centers (position and associated flow value) are picked at random from the scene, with the constraint that they are at least d units apart, where d is a user initialization constant. With the use of atomic arithmetic operations, which mitigate parallel accesses to the same GPU memory location, parameter convergence was achieved in under one minute for all scenes used in

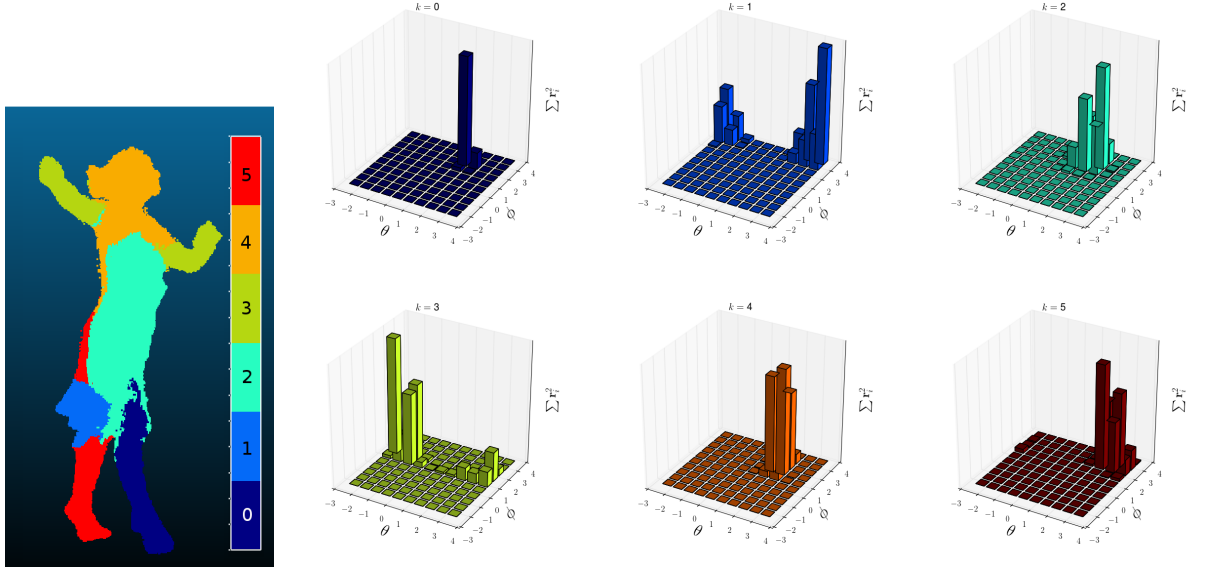


Figure 6.1: Regions of coherent motion and corresponding histogram of flow vector orientations for one instance of the "Dancer" sequence. The bins accumulate the magnitude of the optical flow vectors, *i.e.* $\mathbf{r} \cdot \mathbf{r}$

this paper. The cluster assignment $\mathcal{L}$ for each point p in scene $\mathcal{S}$ characterized by feature vector $\mathbf{f}_v$ is determined by maximizing the cluster posterior probabilities according to Eq. (6.3).

$$\mathcal{L}(p) = \arg \max_i \Pr(\mathbf{f}_v(p)) = \arg \max_i \frac{w_i \Pr(\mathbf{f}_v | i, \theta)}{\sum_{k=1}^K \Pr(\mathbf{f}_v | k, \theta)} \quad \forall p \in \mathcal{S} \quad (6.3)$$

Similarly, the posterior probability field at each point $p \in \mathcal{S}$ is formally defined as:

$$\mathcal{P} = \{\Pr(\mathbf{f}_v(p)) \mid \forall p \in \mathcal{S}\} \quad (6.4)$$

A key goal of this partitioning scheme is to produce a compact representation of the flow field. Since PVMs are expensive in terms of storage, the need for compressing scene data is warranted. Given flow field segmentation, the flow vectors in each cluster can be represented using a low-dimensional motion model, *e.g.* rigid body or affine motion. The parameters for these motion models can be used to encode the flow field instead of storing the flow value per voxel. Provided that the flow field can be accurately represented with significantly fewer clusters than the number of voxels, the cost of storing the motion parameters per cluster is much lower compared to the cost of storing the individual flow value per voxel.

6.1 Optical Flow Field Decomposition into Affine Motion

A reasonable assumption in realistic scenes is that instantaneous motion is locally linear. Thus it follows immediately that the optical flow contained within the motion partitions can be characterized by affine transformations. Let $\mathbf{T}_k$ be the 12-parameter affine transformation that maps points $\mathbf{p}$ in cluster k to their displaced positions $\mathbf{p} + \mathbf{r}$ by the flow field. The set of parameters $T_k^0, T_k^1, \dots, T_k^{11}$ can be found by solving the matrix equation in (6.5) for the n cells found in cluster k . For convenience the notation $\mathbf{p}_i = [x_i, y_i, z_i, 1]$ and $\mathbf{r}_i = [u, v, w, 0]^t$ is used for points and flow vectors, respectively. The null 4-vector is defined as $\mathbf{0} = [0, 0, 0, 0]$. A solution is easily obtained by applying SVD on the $3n \times 12$ sparse matrix.

$$\begin{bmatrix} \mathbf{p}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{p}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{p}_0 \\ \mathbf{p}_1 & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{p}_{n-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{p}_{n-1} \end{bmatrix} \begin{bmatrix} T^0 \\ T^1 \\ \vdots \\ T^{11} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 + \mathbf{r}_0 \\ \mathbf{p}_1 + \mathbf{r}_1 \\ \vdots \\ \mathbf{p}_{n-1} + \mathbf{r}_{n-1} \end{bmatrix} \quad (6.5)$$

The affine motion fitting process is given by the following algorithm:

Algorithm 3: Optical Flow Field approximation using Affine Motion

Input : Scene $\mathcal{S}$, flow field $\mathbf{r} = (u, v, w)$, feature vector $\mathbf{f}_v$
Output: Affine transformation set $\{T_i\}_{i=0}^{K-1}$
1 $\mathcal{P} = \text{PerformSceneEM}(\mathcal{S} | \mathbf{f}_v = (x, y, z, u, v, w))$ //compute posterior field
2 $\mathcal{L}(p) = \arg \max_i \mathcal{P}(p)$, $0 \leq i < K$, $\forall p \in \mathcal{S}$ //compute label field as the index of the most likely cluster for the feature at p
3 **for** $i \leftarrow 0$, **to** $K - 1$ **do**
4 | $T_i \leftarrow \text{SolveAffineFitMatrix}(\mathcal{S}, \mathcal{L}^{-1}(i), \mathbf{r})$ // fit the flow of 3-d points with label i
5 **end**
6 **return** $\{T_i\}_{i=0}^{K-1}$

6.2 Scene Warping Using Affine Motion

A key problem of warping scene $\mathcal{S}_A$ into scene $\mathcal{S}_B$ using the estimated flow is the fact the flow field $\mathbf{r}$ may not be “onto”. That is, $\exists$ a point $p' \in \mathcal{S}_B$ such that $\forall p \in \mathcal{S}_A$, $p + \mathbf{r}(p) \neq p'$. One example is the

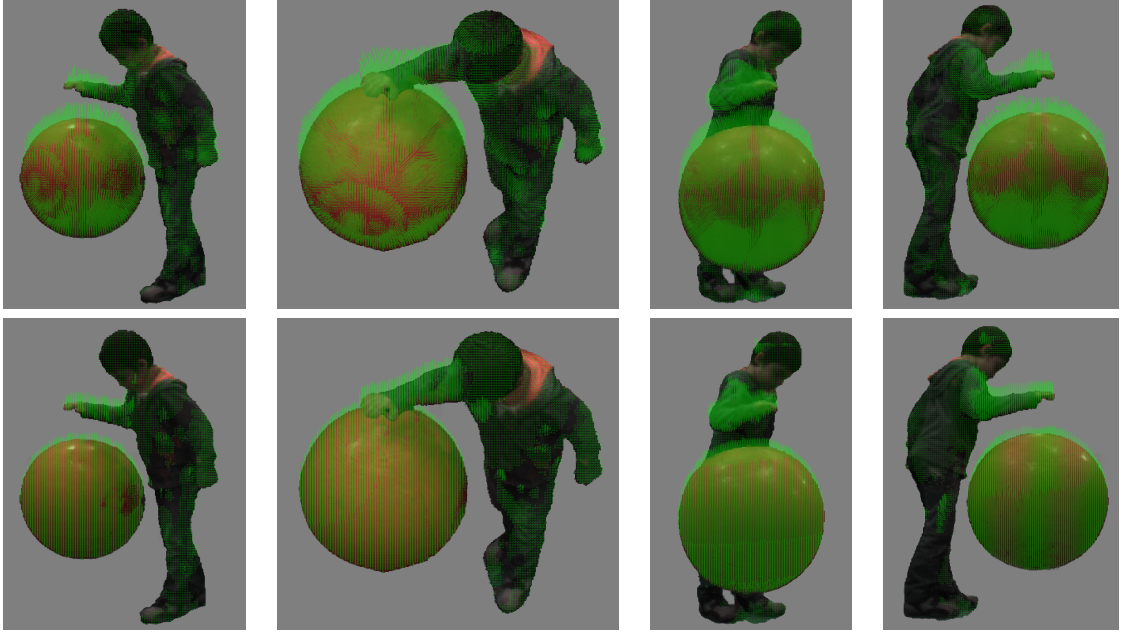


Figure 6.2: Four views of the Expected Flow (Top) and Affine Flow (Bottom) for one instance of the “Boy Playing Ball” sequence

flow field corresponding to a scaling deformation. In addition, error in optical flow estimation can yield flow vectors pointing to the same coordinate which will result in points in scene $\mathcal{S}_B$ not having correspondences in $\mathcal{S}_A$. However, if the flow is represented by an invertible transformation then the warping can be carried out in the inverse direction ensuring that all target points are accounted for. Interpolation in the source 3-d model ensures accurate parameter estimates.

This section proposes a robust method of scene reconstruction through warping by using the label field $\mathcal{L}$ and the set of K affine transformations $\mathbf{T}$, determined according to the method in Sec. 6.1.

To estimate the parameters in the target scene, a decision needs to be made on which matrix T_i should be used to return to the original scene. To be able to carry out that decision, points in the target scene, *i.e.* scene $\mathcal{S}_B$ need to have a cluster assignment. Let the cluster posterior probability field of scene $\mathcal{S}_A$ be $\mathcal{P}_A$.

Note that for each voxel, the posterior probability is a K -vector, where K represents the number of clusters. Using the optical flow field $\mathbf{r}$, the probability field $\mathcal{P}_A$ can be warped into scene $\mathcal{S}_B$, obtaining $\mathcal{P}_B$.

$$\mathcal{P}_B = \mathbf{r} \circ \mathcal{P}_A \quad (6.6)$$



Figure 6.3: Four views of the Expected Flow (Top) and Affine Flow (Bottom) for one instance of the “Dancer” sequence

Some points in $\mathcal{S}_B$ might not have correspondences in $\mathcal{S}_A$ as a result of $\mathbf{r}$ not being “onto”. However, assuming the field $\mathcal{P}_B$ is continuous, the posterior of points $p_B \in \mathcal{S}_B$ without correspondences is determined by sinc-interpolating $\mathcal{P}_B$ over a neighborhood around those points. This neighborhood is assumed to have at least one assigned posterior as a result of the warping operation $\circ$. Consequently, now that all points in $\mathcal{S}_B$ have an assigned posterior, a weighted sum of the cluster inverse matrices is used to map points $p_A \in \mathcal{S}_A$ to $p_B \in \mathcal{S}_B$ according to:

$$p_A = \sum_{i=0}^{K-1} \mathcal{P}_B(p_B|i) (\mathbf{T}_i^{-1} \cdot p_B) \quad (6.7)$$

From continuous posteriors to discrete labels The costs of storing and warping $\mathcal{P}_A$ are very high, taking up K floats per voxel. Consequently, a more efficient method in terms of storage is to warp the label field $\mathcal{L}_A$ corresponding to $\mathcal{S}_A$. The storage cost of $\mathcal{L}_A$ is one int per voxel with values in the range $[0, K)$. Thus one can obtain the label $\mathcal{S}_B$ ’s label field through warping, *i.e.* $\mathcal{L}_B = \mathbf{r} \circ \mathcal{L}_A$. This time, points in $\mathcal{S}_B$ without correspondences in $\mathcal{S}_A$ need to be assigned a cluster



Figure 6.4: Four views of the Estimated Flow (Top) and Affine Flow (Bottom) for one instance of the “Guard Punch Two” sequence

index instead of a cluster posterior. To do so, perform a local neighborhood search around those points and assign the label of their closest neighbor. This serves as a nearest-neighbor interpolation on the flow-warped label field $\mathcal{L}_B$. Finally, now that $\mathcal{S}_B$ ’s label field $\mathcal{L}_B$ has been augmented, every point $p_B \in \mathcal{S}_B$ will have a correspondence in $\mathcal{S}_A$ through Eq. (6.8). Thus to reconstruct $\mathcal{S}_B$, map back every point p_B into p_A using the corresponding inverse transformation T_i^{-1} , where $i = \mathcal{L}_B(p_B)$, and fill its surface and appearance parameters by resampling around p_A , using the trilinear interpolation of random variables defined in (5.7).

$$p_A = \mathbf{T}_{\mathcal{L}_B(p_B)}^{-1} \cdot p_B \quad (6.8)$$

The algorithm for warping $\mathcal{S}_A$ to $\mathcal{S}_B$ is described in Algorithm 4.

In order to find missing cluster posteriors/indexes, it might seem that sinc-interpolating the continuous probability field $\mathcal{P}_B$ is better than interpolating the discrete field $\mathcal{L}_B$ using nearest-neighbor. However, the results turn out to be similar. Reconstruction results show only a slight improvement in the SSIM score when using (6.7) instead of Eq. (6.8). Note that these equations are equivalent if the posterior of the most likely cluster is 1, *i.e.* $\mathbf{Pr}(\mathbf{f}_v|\hat{i}, \boldsymbol{\theta}) = 1$, where the most likely cluster $\hat{i} = \mathcal{L}_A(v) = \arg \max_i \mathcal{P}_A$. The latter implies that the GMM model converged to a single

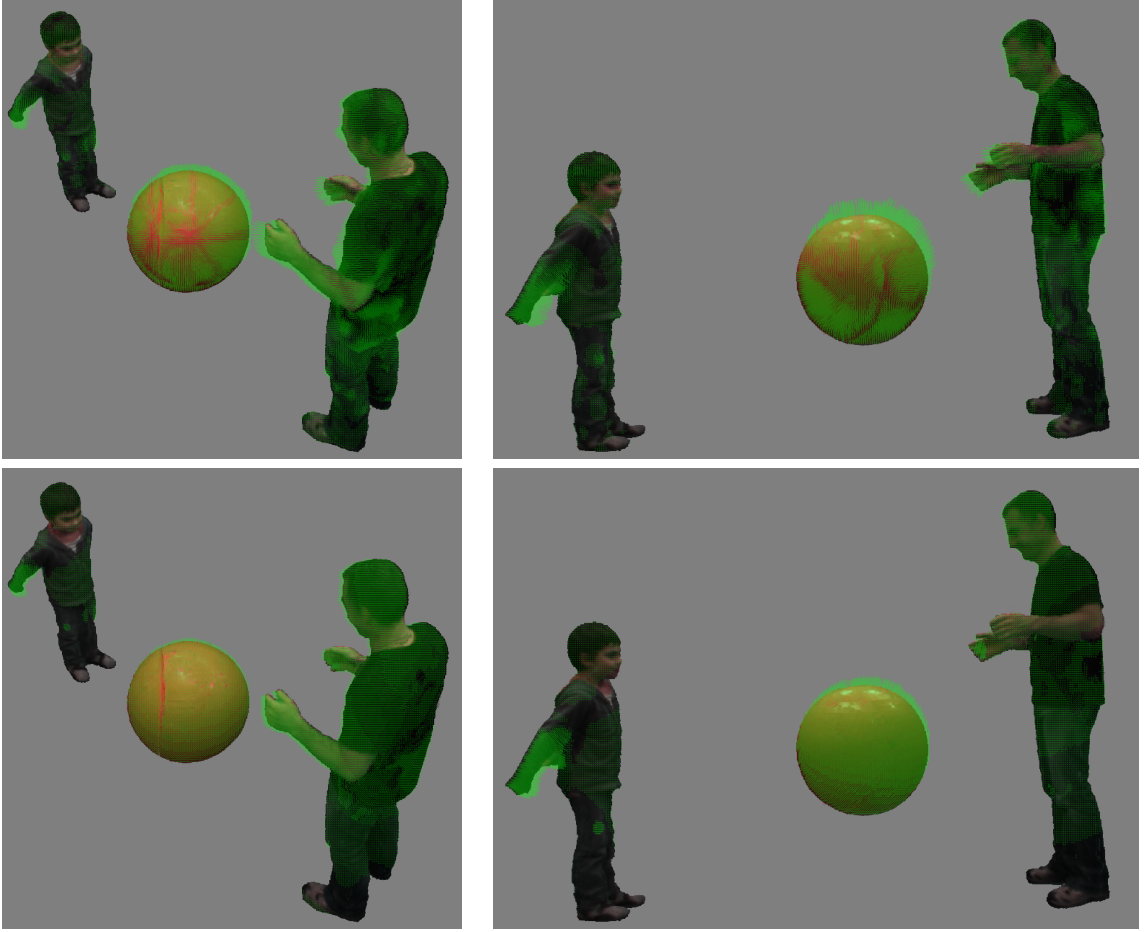


Figure 6.5: Two views of the Estimated Flow (Top) and Affine Flow (Bottom) for one instance of the “Adult Child Ball” sequence

mode. In practice, most voxels in $\mathcal{S}_A$ have the posterior $\Pr(\mathbf{f}_v|\hat{i}, \boldsymbol{\theta}) = 1$. As seen in Fig. 6.14, only the voxels corresponding to cluster boundaries have the posterior of the most likely cluster smaller than 1. These boundary voxels account for a very small fraction of the total surface voxels, thus discretizing the probability field $\mathcal{P}_A$ into $\mathcal{L}_A$ through the arg max operation won’t impact the inverse mapping significantly.

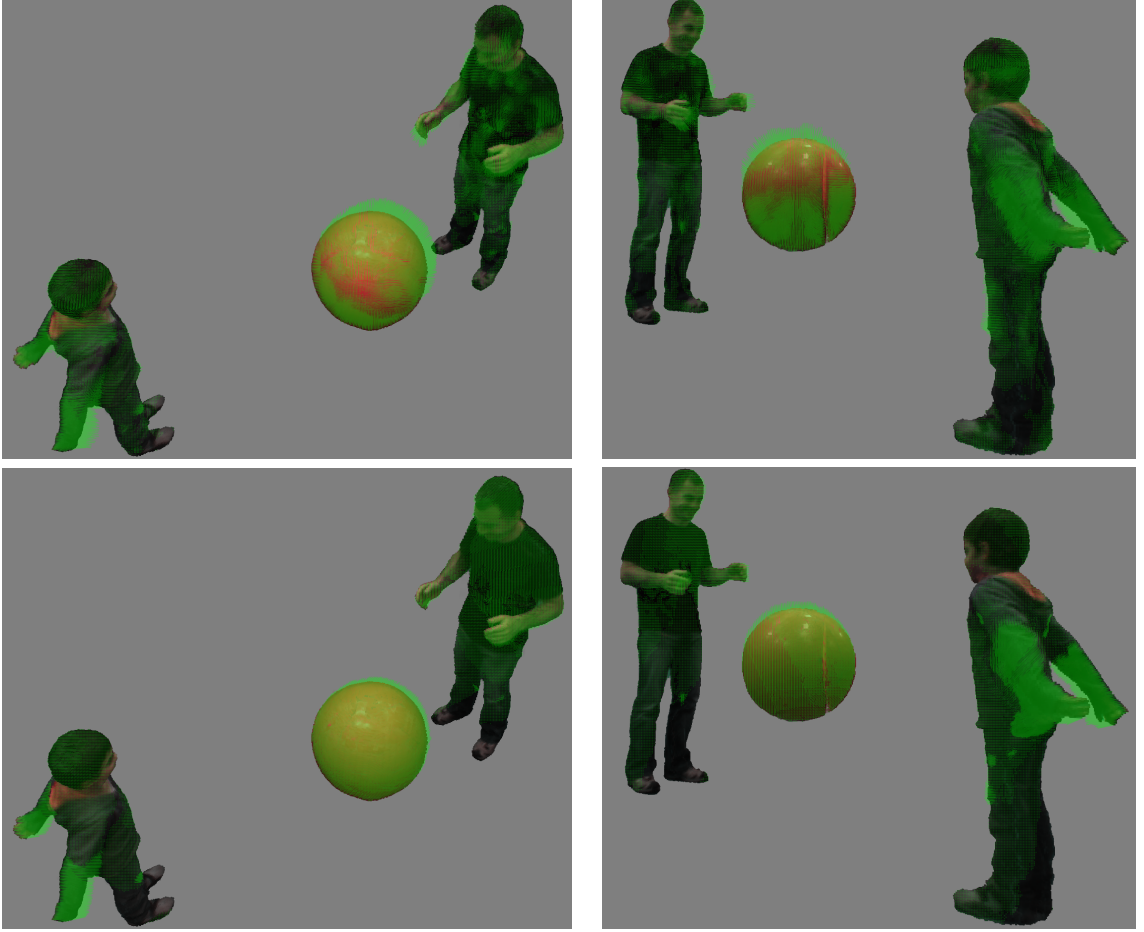


Figure 6.6: Two views of the Estimated Flow (Top) and Affine Flow (Bottom) for one instance of the “Adult Child Ball” sequence

6.3 Scene Compression and Reconstruction Using Affine Motion

Having characterized the motion in a scene through K affine transformations and an integer label field $\mathcal{L}$, a method for compressing scene data follows immediately. The receiver will reconstruct frames after the first one by using just the set of affine matrices $\mathbf{T}$ and the label field $\mathcal{L}$ as described by 6.2. Since the cluster indexes are integers in the $[0, K)$ range, they can be compressed with high efficiency using standard entropy coding techniques. The full scene model is sent entirely to the receiver only when the error due to warping exceeds a threshold. Note that this compression scheme exploits the temporal redundancy in the scene similar to MPEG-4 video encoding. Qualitative scene

Algorithm 4: Scene Warping using Motion Partitions and their Affine Transforms

Input : Scene $\mathcal{S}_A$ with surface and appearance parameters Θ_A , flow field $\mathbf{r} = (u, v, w)$,
Affine transformation set $\{T_i\}_{i=0}^{K-1}$

Output: Resampled Scene $\mathcal{S}_B$ with surface and appearance parameters Θ_B

```

1  $\mathcal{L}_B(p_B) = \text{Warp}(\mathcal{L}(p_A), \mathbf{r})$ 
2 foreach  $p_B \in \mathcal{S}_B$  do
3   if  $\mathcal{L}_B(p_B) = -1$  then
4      $\mathcal{L}_B(p_B) \leftarrow \text{NearestNeighborSearch}(p_B)$ 
5   end
6    $i \leftarrow \mathcal{L}_B(p_B)$ 
7    $p_A \leftarrow T_i^{-1} \cdot p_B$ 
8    $\Theta_B(p_B) \leftarrow \text{ResampleRandVarTrilinear}(p_A, \Theta_A)$ 
9 end
10 return  $\mathcal{S}_B, \Theta_B$ 

```

reconstruction results are presented in Figures 6.7 ,6.8 ,6.9 ,6.10 and 6.11, where reconstructed scene instances are compared to the original scenes. The flow corresponding to the displacement produced by the affine transformations $\mathbf{T}$ is coined as “affine flow”. The original estimated flow and the affine flow is shown in Figures 6.2,6.3,6.4,6.5 and 6.6

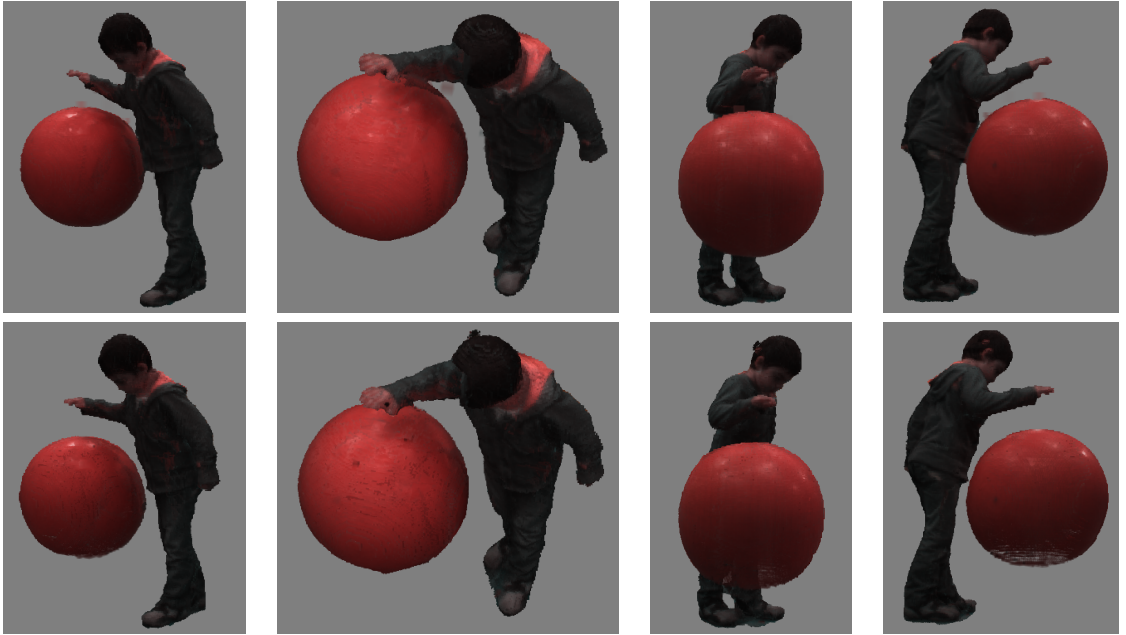


Figure 6.7: Renderings of original scene (Top) and its reconstruction using Affine Flow and the previous scene (Bottom) for one instance of the “Boy Playing Ball” sequence



Figure 6.8: Renderings of original scene (Top) and its reconstruction using Affine Flow and the previous scene (Bottom) for one instance of the “Dancer” sequence

6.4 Results and Discussion

The implementation is based on the VXL project [3]. Experiments are presented on sequences obtained from the public INRIA 4-d repository [2] as well as a synthetic experiment to study the flow results more carefully. The repository provides synchronized videos captured from multiple viewpoints, as well as foreground masks for each frame and camera parameters. The 3-d models are estimated from sixteen images with resolution 1624×1224 . Then the proposed coarse-to-fine scene flow formulation is used to compute the 3-d flow in each voxel. However, before the flow algorithm is applied on the INRIA data, the interior voxels within each scene surface are removed to avoid corruption of the flow field. The content of interior voxels is arbitrary since they are isolated from image information after a few iterations of the online update algorithm as surface voxels attain a high occlusion density value. Using the visibility measure defined in [35] voxels falling below a visibility threshold are set to a state representing empty space.



Figure 6.9: Renderings of original scene (Top) and its reconstruction using Affine Flow and the previous scene (Bottom) for one instance of the “Guard Punch Two” sequence

The scene flow algorithm is iteratively executed at each resolution, until the computed incremental flow falls below a threshold. After each pass of the algorithm, the scene is warped with the flow estimate from the previous iteration. Starting from the coarsest level, this process is repeated for all resolutions. For all experiments in this thesis, the octree depth is limited to four levels for efficient processing in the GPU [93]. Flow results for scenes “Dancer”, “Boy Playing Ball”, “Guard Punch Two” and “Adult Child Ball” (names taken from the dataset) can be seen in Figures 5.3, 5.1, 5.4 and 5.2 respectively. For scenes with better camera coverage such as “Adult Child Ball” and “Boy playing ball” the flow was matched better than “Guard punch two” where the geometry was modeled incorrectly in some locations due to reduced visibility from multiple viewpoints. Nevertheless, the flow is estimated correctly for visible surfaces and robustly in the presence of change in surface topology.

Unfortunately the INRIA dataset does not provide ground truth 3-d flow which could be used to evaluate the 3-d accuracy of the flow field. In fact, to the best of our knowledge, there exist no multi-view datasets with a wide coverage of views that provides ground truth 3-d motion. While the flow renderings display correct direction and magnitude, it is not possible to validate them quantitatively. Nonetheless, the following experiment is proposed to quantify error. The first scene is warped according to the flow estimate and then compared to the second scene. If the flow estimate

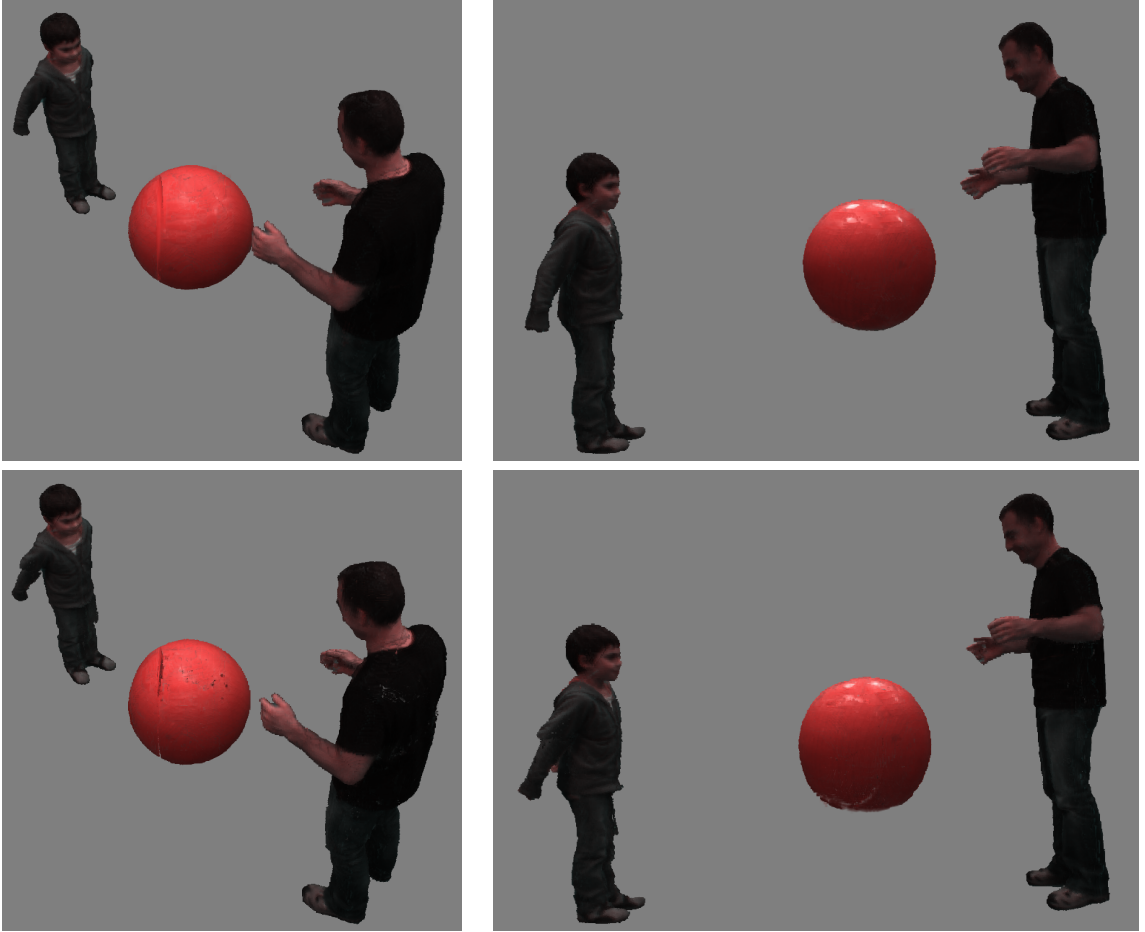


Figure 6.10: Renderings of original scene (Top) and its reconstruction using Affine Flow and the previous scene (Bottom) for one instance of the “Adult Child Ball” sequence

is correct, then the warped scene and the second scene would match. This similarity between 3-d models is quantified using a rendering experiment. The warped scene is rendered from multiple-viewpoints and these renderings are compared to the renderings produced from the original second scene. The images are compared using the Structural Similarity Index Measure (SSIM) [133]. The scores are in the $[0,1]$ range, 1 being the maximum value attainable when the source image matches the target image. The results are averaged for all viewpoints over 100 frames and are tabulated in Table 6.1. Scenes with fast motion such as “guard punch two” had a lower score due to inaccurate flow estimation on fast moving surfaces. However, the scores for the rest of the sequences were generally high. Experiments similar to the ones in [10] were also performed in which intermediate frames are predicted using optical flow and interpolation error is evaluated. However, the results

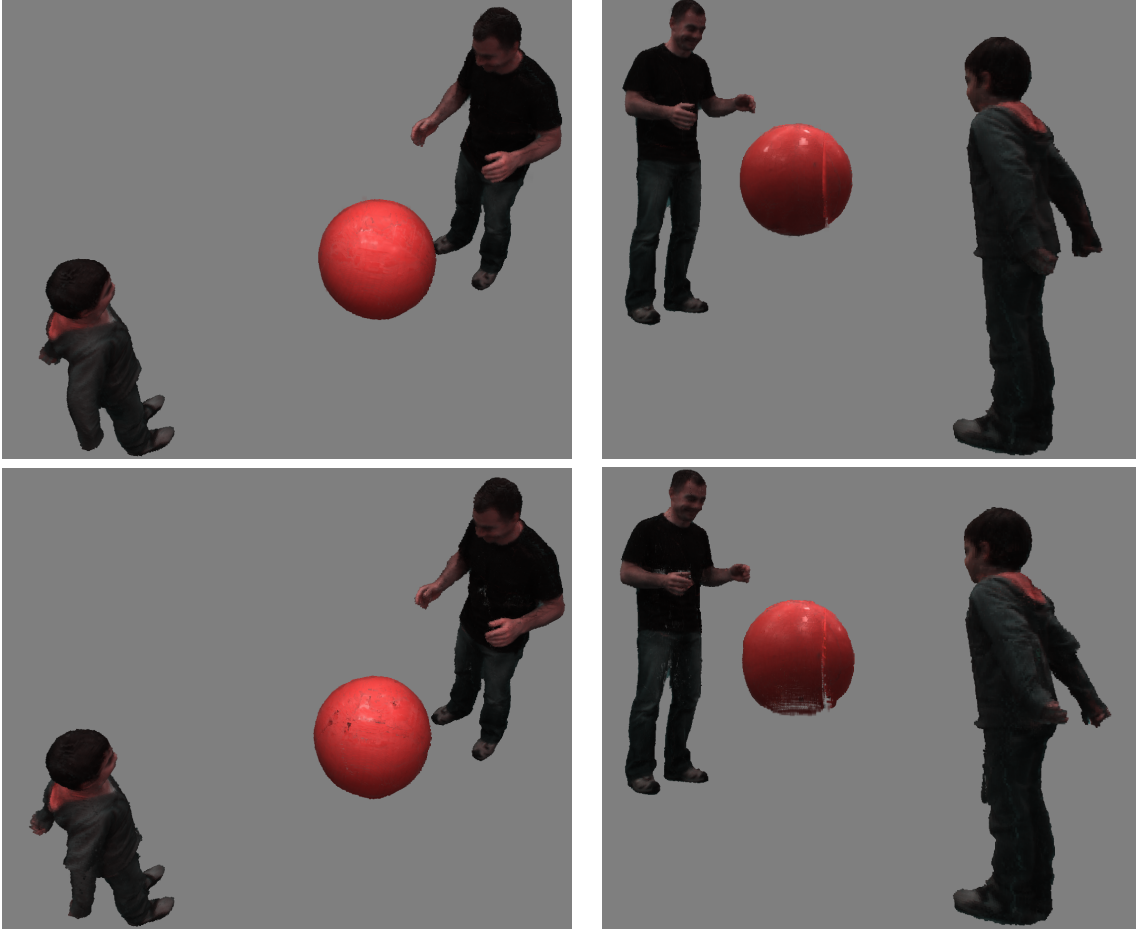


Figure 6.11: Renderings of original scene (Top) and its reconstruction using Affine Flow and the previous scene (Bottom) for one instance of the “Adult Child Ball” sequence

were unsuccessful since INRIA scene motion between non-consecutive frames is too large. A motion capture studio with high speed cameras such as the ones used in [10] would provide non-consecutive time instances with spatio-temporal continuity.

Segmentation results are presented in Figure 6.15 for each of the sequences. No more than 8 clusters were needed to partition each scene in clusters of coherent motion. The clusters were used to find a the set of affine parameters in Section 6.1. An example comparison of the original flow field and the flow resulting from the local affine transformations on one of the frames of the ‘Dancer’ sequence is presented in Figure 6.13. The SSIM is used once more to assess the quality of scenes warped with the set of affine transformations. Results presented in Table 6.1 show that the warping method proposed in Sec. 6.2 achieves better SSIM scores. This increase in accuracy can

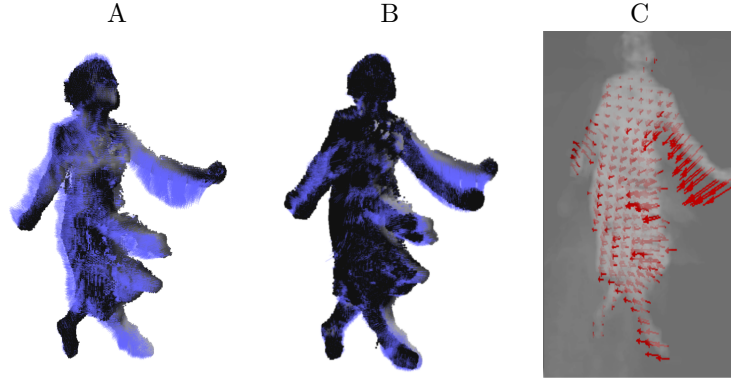


Figure 6.12: Comparison with the results from [53] on the “Dancer” sequence. (A) Estimated flow for frame 11. Note how the dense optical flow is corresponding to the dancer’s the pirouette motion both on the arms and on the dress. (B) frame 11 and frame 12 are overlaid, underscoring the correct magnitude of the resultant flow. (C) Result obtained by [53] on the same frame. Although the vectors seem correct, the lack of density shows little to no flow on the dress.

Sequence	Resultant Warp	Affine Warp
Dancer	0.71	0.74
Boy playing ball	0.78	0.82
Adult child ball	0.81	0.85
Guard punch two	0.68	0.69

Table 6.1: Average SSIM measure between warped scene renderings and original scene renderings across 100 frames. The resultant flow was used to warp the sence in the first column while the set of affine transforms was used in the second one

be appreciated on the rendering at the bottom of Figure 6.13. Since all points in the target scene have a correspondences in the source scene through the inverse transforms, a higher quality warp is achieved. In addition, the linear parametrization of local flow attenuates small errors in estimation.

On average, the compression factor of storing the entire flow field versus the field of cluster labels per scene is roughly 176X. Results averaged over 100 frames for each sequence are tabulated in Table 6.2. Because the number of clusters experimented with vary from 5 to 12, the buffer that stores the label field $\mathcal{L}$ is highly redundant from a compression point of view. Thus the general purpose compression library *zlib* [1] was able to compress the label buffer much more efficiently that the estimated flow buffer. The compression gain of storing $\mathcal{L}$ instead of the surface and appearance parameters is also reported. Increasing the number of clusters K would yield a poorer estimation of affine parameters as the regions of coherent motion would become too small to correctly describe

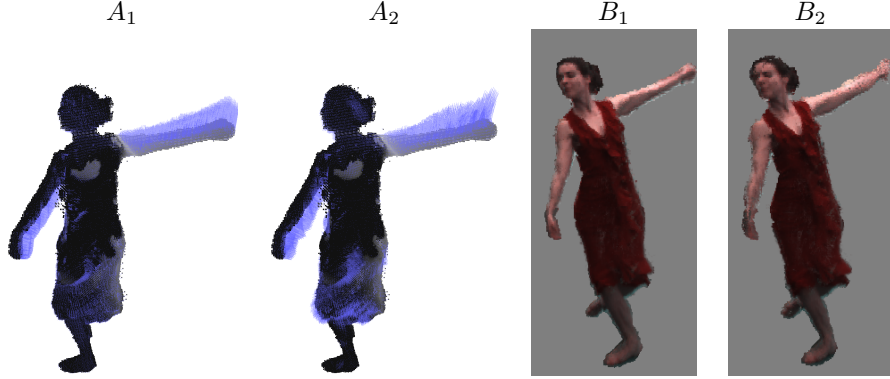


Figure 6.13: Linearly fitted flow using Figure 6.5 (A_1) and original resultant flow (A_2); Rendering of warped scene using to affine flow (B_1) and estimated flow (B_2)

Sequence	Flow Field (X)	Surface and Appearance Parameters (X)
Dancer	237	251
Boy playing ball	159	163
Adult child ball	155	180
Guard punch two	156	168

Table 6.2: Compression ratio between the storage cost of the label field $\mathcal{L}$ and the entire flow field (left column) and scene parameters (right column)

the actual motion. In addition, the compression gain of using *zlib* on the label field was similar despite varying K . Having many surfaces in motion, the “dancer” sequence naturally achieved a higher compression ratio than sequences such as “Adult Child Ball” where only some surfaces were in motion (i.e. the ball).

6.5 Conclusions

This chapter demonstrated a reconstruction method based on scene warping with a reparametrized optical flow field. Decomposing the field into regions of coherent motion allows the fitting of the flow vectors to an affine parameter space for each cluster. Reconstruction results prove successful for most frames, as indicated by the SSIM quality measure on scene renderings from multiple viewpoints as well as the qualitative renderings of reconstructed scenes. However, in locations where the optical flow was incorrectly estimated, the accuracy of the segmentation and resulting affine fit dropped. Although the algorithm is implemented in a coarse-to-fine manner, the motion of some very fast moving objects, such as the left hand of the falling karate fighter is problematic as seen in the

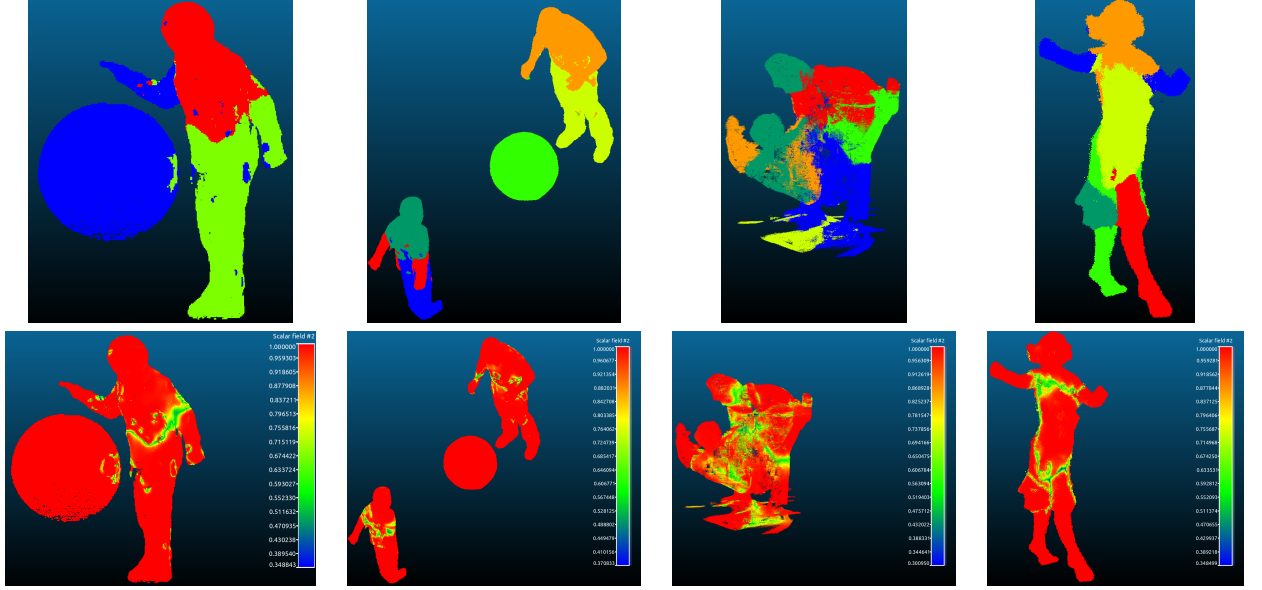


Figure 6.14: Top: Label field $\mathcal{L}$ containing the index of the most likely cluster at each voxel. Each color represents an integer index; Bottom: The posterior probability field corresponding to the most likely cluster, *i.e.* $\max_i \mathcal{P}$. Here, the $[0, 1]$ range has been mapped to the $R > G > B$ heatmap encoding

renderings in Fig. 6.9. The flow estimates in these locations were not accurate. Increasing the number of octree resolution levels as well as enclosing the data and smoothness term in the energy functional with robust penalizing functions would potentially overcome this limitation. Since the overall performance of this system depends on optical flow estimation quality, there are multiple avenues for future work in pursuing better methods in flow estimation for PVMs.

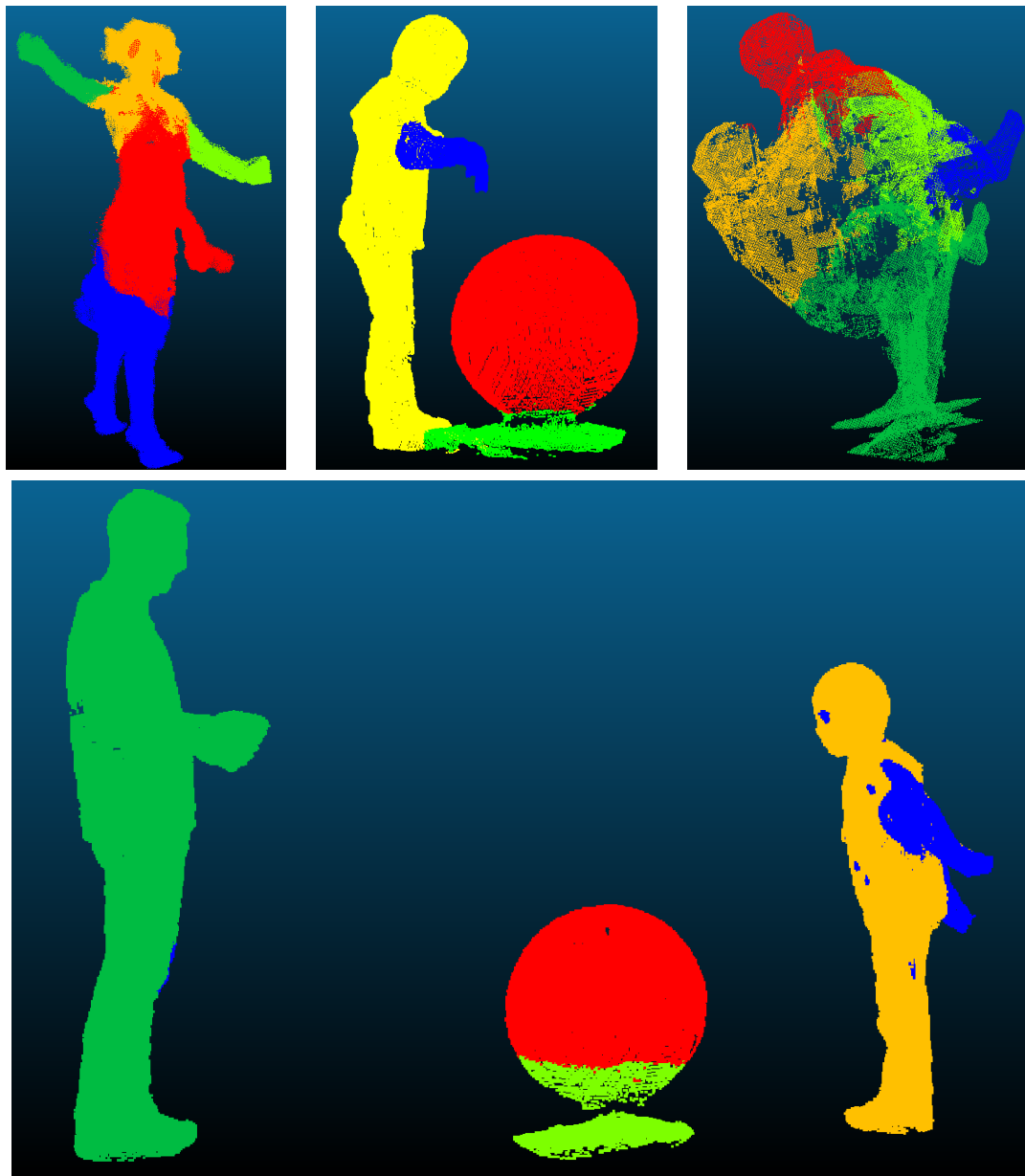


Figure 6.15: Segmentation into 5 clusters using position and estimated flow for one of the frames from each of the tested sequences. Note how voxels with similar motion belong to the same cluster

Chapter 7

Conclusions and Future Work

This thesis gave a self-contained treatment of compression techniques designed for dynamic scenes in 2-d and 3-d. The first tier of this thesis explored the case study of compressing videos acquired by Large Aerial Video collectors. The methods used in this case study established the fundamental concepts employed in compressing 4-d scenes. The principles of wavelets, embedded block coding, context adaptive coding and multiplier-free arithmetic coding are widely used in most performance compression systems today which motivates the use of JPEG2000 as a compression engine. The algebra of random variables enables a sound theoretical treatment of manipulating random variable spaces which are different from the image domain. This treatment enabled the formulation of the wavelet transform of pixel-wise background models in 2-d and the resampling of probabilistic volumetric models in 3-d. Results were presented in 2-d, demonstrating that large video can be encoded with just a coarse resolution of the background model and only occasionally resorting to the decompression of full resolution patches.

The second tier of this thesis described a method of estimating optical flow in 3-d on PVMs based on Horn and Schunck's method. Adapting this method to the PVM was not a trivial task, as the domain of appearance random variables and surface probabilities as well as the non-uniform voxel grid differs from the image domain and the isotropic image grid. Nonetheless, accurate flow estimation results were demonstrated for multiple scenes. Validation was evaluated using synthetic generated data and quality of rendering using the SSIM measure. The accuracy and usefulness of flow results were also demonstrated by segmentation and compression experiments. The second contribution of this tier consists in significantly reducing the storage requirements of time-varying

PVMs by as high as 230X, through optical flow field re-parametrization and warping.

There exist many directions of future work for the framework. First of all, the formulation in this thesis is derived from the classical Horn and Schunck optical flow method, which was proposed in 1981. The Horn and Schunck optical flow algorithm when properly implemented in a multi-resolution framework performs comparably to other methods, but evaluating other state of the art methods [117] is an important next step. Another interesting future direction is to jointly reconstruct 3-d models and estimate scene flow as proposed by Guan *et al.* [53] but with the dense flow enabled by the PVM [10].

An additional improvement on the 4-d compression framework is by adjusting the EM initialization such that the final cluster solution is not a local minimum of the likelihood function. Also, increasing the feature size to include expected appearance or surface probability will likely improve motion partitions. Imposing connectivity between clusters after EM convergence will also improve the fitting process of flow vectors. This in turn will allow for more accurate reconstruction using the fitted flow.

A final step in connecting the first tier of this thesis with the second one is to formulate the 3-d Wavelet Transform on the non-uniform voxel grid. Unlike the 2-d wavelet transform, the 3-d wavelet transform will produce eight subbands instead of four namely, LLH , LHL , LHH , LLL , HLH , HHL , HHH , HLL . On a uniform grid the 3-d lifting scheme can be used to obtain the interleaved wavelet coefficients of a given scene. These coefficients can be stored at the leafs of each octree. Thus instead of storing the spatial data, the octree leafs can now be used to store the eight wavelet coefficients of a single level of wavelet decomposition. As before, the transform can be recursively applied on the collection of leafs representing the LLL subband. The resulting 2^{nd} level coefficients can be now stored at the next level in the octree. Having presented the algebra of random variables for view-dependent appearance models, the 3-d Wavelet Transform for probabilistic volumetric follows immediately. This in turn will open new avenues for JPEG2000 coding in 3-d. The bitplane coding passes and contexts were designed for four 2-d subbands only, thus the problem of adding additional contexts is raised. Evidently, more contexts will most likely change the state transition system in the MQ coder in JPEG 2000 presented in Table 2.6. Also, it is not clear whether traversal of 3-d Codeblocks slice by slice and applying traditional 2-d bitplane coding for each slice will have the same performance in 3-d.

However, a 3-d JPEG2000 compressed probabilistic volumetric model similar to the compressed

2-d background model proposed in Sec. 3.1.1 will be highly useful in the presented motion-based compression system in Sec. 6.3. When the full model is needed by the receiver as a result of accumulated reconstruction error, it can be sent as a 3-d JPEG2000 compressed bitstream. Additionally, scene reconstruction and flow estimation with limited memory can also be addressed by using a compressed representation of the PVM. For example, the initial optical flow iterations can be done over the compressed representation of the scene *i.e.* the LLL^k subband. Since JPEG2000 allows for progressively transmitting resolution levels, the optical flow estimation process can be scaled to fit the memory requirements.

While this sketch enables a straightforward way of evaluating the 3-d wavelet transform and its subsequent use as a basis for 3-d JPEG-2000 compression, it remains to be seen how to account for unrefined voxels during the filtering process, reformulate embedded block coding and context adaptive coding as well as entropy coding. Regardless, these avenues are equally promising as they are unexplored.

Bibliography

- [1] Rfc 1950 - zlib compressed data format specification version 3.3, 1996. <http://www.faqs.org/rfcs/rfc1950.html>.
- [2] Inria 4d repository, 2012. URL <http://4drepository.inrialpes.fr/public/datasets>.
- [3] The vxl project, 2012. URL <http://http://sourceforge.net/projects/vxl/>.
- [4] Yuv soft technologies - depth propagation, 2015. URL <http://www.yuvsoft.com/stereo-3d-technologies/depth-propagation/>.
- [5] E. Adelson and J. Bergen. The plenoptic function and the elements of early vision. *Computational models of visual ...*, pages 3–20, 1991. ISSN 03741338.
- [6] E. D. Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. *ACM Transactions on Graphics (TOG)*, 2008. URL <http://dl.acm.org/citation.cfm?id=1360697>.
- [7] A. N. Akansu and R. A. Haddad. *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets*. Academic Press, Inc., Orlando, FL, USA, 1992. ISBN 012047140X.
- [8] R. Arcila, C. Cagniart, F. Hétroy, and E. Boyer. Temporally coherent mesh sequence segmentations. (January), 2012.
- [9] R. V. Babu and A. Makur. Object-based Surveillance Video Compression using Foreground Motion Compensation. *2006 9th International Conference on Control, Automation, Robotics and Vision*, pages 1–6, 2006. doi: 10.1109/ICARCV.2006.345186. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4150096>.

- [10] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. ISSN 0920-5691. doi: 10.1007/s11263-010-0390-2. URL <http://dx.doi.org/10.1007/s11263-010-0390-2>.
- [11] T. Basha, S. Avidan, a. Hornung, and W. Matusik. Structure and motion from scene registration. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1426–1433, June 2012. doi: 10.1109/CVPR.2012.6247830. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6247830>.
- [12] R. Bhotika, D. J. Fleet, and K. N. Kutulakos. A Probabilistic Theory of Occupancy and Emptiness. pages 112–132, May 2002. URL <http://dl.acm.org/citation.cfm?id=645317.649335>.
- [13] O. Biris and J. Mundy. *High Resolution Surveillance Video Compression Using JPEG2000 Compression of Random Variables*, volume 359 of *Communications in Computer and Information Science*. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-38240-6. doi: 10.1007/978-3-642-38241-3_20. URL http://dx.doi.org/10.1007/978-3-642-38241-3_20.
- [14] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.
- [15] M. Bojsen-Hansen, H. Li, and C. Wojtan. Tracking surfaces with evolving topology. *ACM Transactions on Graphics*, 31(4):1–10, July 2012. ISSN 07300301. doi: 10.1145/2185520.2335404. URL <http://dl.acm.org/citation.cfm?doid=2185520.2335404>.
- [16] M. Bojsen-Hansen, H. Li, and C. Wojtan. Tracking surfaces with evolving topology, 2012. ISSN 07300301.
- [17] J. D. Bonet and P. Viola. Poxels: Probabilistic voxelized volume reconstruction. *Proceedings of International Conference on ...*, pages 1–8, 1999.
- [18] W. B. Pennebaker, J. J. L. Mitchell, G. G. Langdon, and R. B. Arps. An overview of the basic principles of the Q-coder adaptive binary arithmetic coder. *Image Processing, IEEE Transactions on*, 32(6):717, Nov. 1988.
- [19] G. Bradski. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000.

- [20] a. J. Bray. Tracking Objects Using Image Disparities. *Proceedings of the Alvey Vision Conference 1989*, pages 14.1–14.6, 1989. doi: 10.5244/C.3.14. URL <http://www.bmva.org/bmvc/1989/avc-89-014.html>.
- [21] A. Broadhurst, T. Drummond, and R. Cipolla. A probabilistic framework for space carving. *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, 1, 2001. doi: 10.1109/ICCV.2001.937544.
- [22] T. Brox, N. Papenberg, and J. Weickert. High Accuracy Optical Flow Estimation Based on a Theory for Warping. 4(May):25–36, 2004.
- [23] C. Cagniart, E. Boyer, and S. Ilic. Probabilistic Deformable Surface Tracking. 6314:326–339, 2010.
- [24] C. Cagniart, E. Boyer, and S. Ilic. Probabilistic Deformable Surface Tracking. *ECCV*, pages 326–339, 2010.
- [25] F. Calakli, A. O. Ulusoy, M. I. Restrepo, G. Taubin, and J. L. Mundy. High Resolution Surface Reconstruction from Multi-view Aerial Imagery. In *3D Imaging Modeling Processing Visualization Transmission (3DIMPVT)*, IEEE. IEEE, 2012. URL <http://mesh.brown.edu/gssd/pdf/Calakli-3dimpvt2012.pdf>.
- [26] R. Carceroni and K. Kutalakos. Multi-view scene capture by surfel sampling: from video streams to non-rigid 3D motion, shape and reflectance. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 60–67. IEEE Comput. Soc, 2001. ISBN 0-7695-1143-0. doi: 10.1109/ICCV.2001.937603. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=937603>.
- [27] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Transactions on Graphics*, 22(3):569, July 2003. ISSN 07300301. doi: 10.1145/882262.882309. URL <http://dl.acm.org/citation.cfm?id=1201775.882309>.
- [28] G. Cernigliaro, M. Naccari, F. Jaureguizar, J. Cabrera, and N. Garcia. Depth perceptual video coding for free viewpoint video based on H.264/AVC. *2012 Picture Coding Symposium, PCS 2012, Proceedings*, pages 153–156, 2012. doi: 10.1109/PCS.2012.6213309.

- [29] W.-C. Chen, J.-Y. Bouguet, M. H. Chu, and R. Grzeszczuk. Light field mapping: efficient representation and hardware rendering of surface light fields, 2002. ISSN 07300301.
- [30] A. Cohen, I. Daubechies, and J.-C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 45(5):485–560, 1992. ISSN 1097-0312. doi: 10.1002/cpa.3160450502. URL <http://dx.doi.org/10.1002/cpa.3160450502>.
- [31] J. Committee. Joint photographic experts group (JPEG). <http://www.jpeg.org/committee.html>.
- [32] S. Corazza, L. Mündermann, a. M. Chaudhari, T. Demattio, C. Cobelli, and T. P. Andriacchi. A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach. *Annals of Biomedical Engineering*, 34(6):1019–1029, 2006. ISSN 00906964. doi: 10.1007/s10439-006-9122-8.
- [33] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, New York, NY, USA, 1991. ISBN 0-471-06259-6.
- [34] D. Cremers and K. Kolev. Multiview stereo and silhouette consistency via convex functionals over convex domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(6):1161–1174, 2011. ISSN 01628828. doi: 10.1109/TPAMI.2010.174. URL <http://www.ncbi.nlm.nih.gov/pubmed/20820076>.
- [35] D. Crispell, J. Mundy, and G. Taubin. A Variable-Resolution Probabilistic Three-Dimensional Model for Change Detection. *IEEE Transactions on Geoscience and Remote Sensing*, 50(2):489, 2012.
- [36] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl*, 4:247–269, 1998.
- [37] E. De Aguiar, C. Theobalt, C. Stoll, and H. P. Seidel. Marker-less deformable mesh tracking for human shape and motion capture. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007. ISSN 10636919. doi: 10.1109/CVPR.2007.383296.
- [38] J. Duan and J. Li. Compression of the layered depth image. *IEEE Transactions on Image Processing*, 12(3):365–372, 2003. ISSN 10577149. doi: 10.1109/TIP.2003.809010.

- [39] E. Ekmekcioglu, S. T. Worrall, and A. M. Kondo. A temporal subsampling approach for multiview depth map compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(8):1209–1213, 2009. ISSN 10518215. doi: 10.1109/TCSVT.2009.2020336.
- [40] G. Farneb. Two-Frame Motion Estimation Based on. *Lecture Notes in Computer Science*, 2749(1):363–370, 2003. doi: 10.1007/3-540-45103-X_50.
- [41] C. Fehn, P. Kauff, M. O. D. Beeck, F. Ernst, W. IJsselsteijn, M. Pollefeys, L. V. Gool, E. Ofek, and I. Sexton. An Evolutionary and Optimised Approach on 3D-TV. *In Proceedings of International Broadcast Conference*, (March 2002):357–365, 2002. URL http://tests-zingarelli.googlecode.com/svn-history/r210/trunk/2-Artigos-Projeto/Usados-projeto-FAPESP/Fehn_etall-An-evolutionary-and-optimized-approach-on-3D-TV-2002.PDF.
- [42] F.Huguet, D. P. Team, and I. R.-a. Montbonnot. A Variational Method for Scene Flow Estimation from Stereo Sequences. 2007.
- [43] N. Friedman and S. Russell. Image segmentation in video sequences : A probabilistic approach 1 Introduction. *UAI*, pages 175–181, 1997.
- [44] T. Fujii and M. Tanimoto. Free-viewpoint TV system based on ray-space representation. *SPIE ITCOM*, 4864:175–189, 2002. ISSN 0277786X. doi: 10.1117/12.454905. URL http://spie.org/x648.html?product_id=454905.
- [45] T. Fujii, T. Kimoto, and M. Tanimoto. A new flexible acquisition system of ray-space data for arbitrary objects. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(2):218–224, 2000. ISSN 10518215. doi: 10.1109/76.825721.
- [46] Y. Furukawa and J. Ponce. Dense 3d motion capture from synchronized video streams. *CVPR*, 2008. URL <http://www.springerlink.com/index/VN133336Q3K6W775.pdf>.
- [47] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards Internet-scale Multi-view Stereo. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.
- [48] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel. Motion capture using joint skeleton tracking and surface estimation. *CVPR*, June 2009. doi: 10.

- 1109/CVPR.2009.5206755. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5206755>.
- [49] B. Glocker, A. Sotiras, N. Komodakis, and N. Paragios. Deformable medical image registration: setting the state of the art with discrete methods. *Annual review of biomedical engineering*, 13:219–44, Aug. 2011. ISSN 1545-4274. doi: 10.1146/annurev-bioeng-071910-124649. URL <http://www.ncbi.nlm.nih.gov/pubmed/21568711>.
- [50] M. Golparvar-Fard, F. Peña Mora, and S. Savarese. Integrated Sequential As-Built and As-Planned Representation with Tools in Support of Decision-Making Tasks in the AEC/FM Industry, 2011. ISSN 0733-9364.
- [51] M. Golparvar-Fard, F. Pena-Mora, and S. Savarese. Monitoring changes of 3D building elements from unordered photo collections. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 249–256, 2011. doi: 10.1109/ICCVW.2011.6130250.
- [52] M. Gross, S. Würmlin, and M. Naef. blue-c: a spatially immersive display and 3D video portal for telepresence. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2003*, 22(3):819–827, 2003. ISSN 07300301. doi: 10.1145/882262.882350. URL <http://dl.acm.org/citation.cfm?id=882350>.
- [53] L. Guan and J. Franco. Probabilistic 3D occupancy flow with latent silhouette cues. *Computer Vision and ...*, 2010, 2010. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5539807.
- [54] S. Hadfield and R. Bowden. Kinecting the dots: Particle based scene flow from depth sensors. *2011 International Conference on Computer Vision*, pages 2290–2295, Nov. 2011. doi: 10.1109/ICCV.2011.6126509. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126509>.
- [55] S. Hadfield and R. Bowden. Scene particles: unregularized particle-based scene flow estimation. *IEEE transactions on pattern analysis and machine intelligence*, 36(3):564–76, Mar. 2014. ISSN 1939-3539. doi: 10.1109/TPAMI.2013.162. URL <http://www.ncbi.nlm.nih.gov/pubmed/24457511>.

- [56] C. G. Harris. Tracking with Rigid Models. In *Active Vision*, pages 263–284. 1992. ISBN 0262023512. URL <http://portal.acm.org/citation.cfm?id=163217&dl=GUIDE>.
- [57] N. Hasler, B. Rosenhahn, T. Thormählen, M. Wand, J. Gall, and H. P. Seidel. Markerless motion capture with unsynchronized moving cameras. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, pages 224–231, 2009. ISSN 1063-6919. doi: 10.1109/CVPRW.2009.5206859.
- [58] B. Hassett. Introduction to Algebraic Geometry. *October*, 58(1):1–7, 2007. ISSN 16828658. doi: 10.2307/2307774. URL <http://ebooks.cambridge.org/ref/id/CB09780511755224>.
- [59] M. Heikkilä and M. Pietikäinen. A texture-based method for modeling the background and detecting moving objects. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):657–662, Apr. 2006. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.68. URL <http://www.ncbi.nlm.nih.gov/pubmed/16566514>.
- [60] L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. Real-time photo-realistic 3D mapping for micro aerial vehicles. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4012–4019, 2011. ISBN 9781612844541. doi: 10.1109/IROS.2011.6048818.
- [61] E. Herbst and D. Fox. RGB-D Flow : Dense 3-D Motion Estimation Using Color and Depth. *ICRA*, 2014.
- [62] S. Heymann, a. Smolic, K. Mueller, Y. Guo, J. Rurainsky, P. Eisert, and T. Wiegand. Representation, coding and interactive rendering of high-resolution panoramic images and video using MPEG-4. *Proc. Panoramic Photogrammetry Workshop (PPW)*, 2005. URL <http://www.ki.informatik.hu-berlin.de/viscom/papers/ppw05b.pdf>.
- [63] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3): 185–203, Aug. 1981. ISSN 00043702. doi: 10.1016/0004-3702(81)90024-2. URL <http://www.sciencedirect.com/science/article/pii/0004370281900242>.
- [64] M. Hornacek, A. Fitzgibbon, and C. Rother. SphereFlow: 6 DoF Scene Flow from RGB-D Pairs. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3526–3533, June 2014. doi: 10.1109/CVPR.2014.451. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6909846>.

- [65] C.-H. Huang, E. Boyer, and S. Ilic. Robust Human Body Shape and Pose Tracking. *2013 International Conference on 3D Vision*, pages 287–294, 2013. doi: 10.1109/3DV.2013.45. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6599088>.
- [66] ITU-T. Digital compression and coding of continuous-tone still images. Recommendation ITU-T T.81 | International Standard ISO/IEC 10918-1, Sep 1992. URL <http://handle.itu.int/11.1002/1000/2633>.
- [67] ITU-T and ISO/IEC. Advanced video coding for generic audiovisual services. Recommendation ITU-T H.264 | International Standard ISO/IEC 14496-10, Feb 2014. URL <http://handle.itu.int/11.1002/1000/12063>.
- [68] T. S. S. (ITU-T). T.800 : Information technology JPEG 2000 image coding system: Core coding system, 2001.
- [69] S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld. Detection and location of people in video images using adaptive fusion of color and edge information. In *ICPR'00*, pages 4627–4631, 2000.
- [70] O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *Motion and Video Computing, 2002. Proceedings. Workshop on*, pages 22 – 27, Dec. 2002. doi: 10.1109/MOTION.2002.1182209.
- [71] H. Joo, H. Park, and Y. Sheikh. MAP Visibility Estimation for Large-Scale Dynamic 3D Reconstruction. *CVPR*, 2014. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6909543.
- [72] I. JTC1/SC29/WG11. Information technology – coding of audio-visual objects – part 1: Systems. Ballot No. ISO/IEC 14496-1, Oct 1999.
- [73] I. JTC1/SC29/WG11. Information technology – coding of audio-visual objects – part 11: Scene description and application engine. Ballot No. ISO/IEC 14496-11, Nov 2011.
- [74] I. JTC1/SC29/WG11. Information technology – coding of audio-visual objects – part 16: Animation framework extension (afx), amendment 4: Pattern-based 3d mesh coding. Ballot No. ISO/IEC 14496-16, Dec 2014.

- [75] K. N. Kutulakos, K. N. Kutulakos, S. M. Seitz, and S. M. Seitz. A theory of shape by space-carving. *Int. J. Comput. Vision*, vol(3):38no3pp197–216, 2000.
- [76] A. Laurentini. Visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994. ISSN 01628828. doi: 10.1109/34.273735.
- [77] R. Le, J. Mundy, and I. Bahar. High performance parallel JPEG2000 streaming decoder using gpgpu-cpu heterogeneous system. In *Application-specific Systems, Architectures and Processors, 2012 IEEE 23rd International Conference on*, july 2012.
- [78] D.-S. Lee. Effective gaussian mixture learning for video background subtraction. *IEEE transactions on pattern analysis and machine intelligence*, 27(5):827–32, May 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.102. URL <http://www.ncbi.nlm.nih.gov/pubmed/15875805>.
- [79] J. Leininger B., Edwards. Autonomous real-time ground ubiquitous surveillance-imaging system (argus-is). In *Defense Transformation and Net-Centric Systems 2008*, volume 6981, 2008.
- [80] A. Letouzey and E. Boyer. Progressive shape models. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:190–197, 2012. ISSN 1063-6919. doi: <http://doi.ieeecomputersociety.org/10.1109/CVPR.2012.6247675>.
- [81] A. Letouzey, B. Petit, and E. Boyer. Scene Flow from Depth and Color Images. *Proceedings of the British Machine Vision Conference 2011*, pages 46.1–46.11, 2011. doi: 10.5244/C.25.46. URL <http://www.bmva.org/bmvc/2011/proceedings/paper46/index.html>.
- [82] M. Levoy and P. Hanrahan. Light field rendering. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, pages 31–42, 1996. ISSN 00978930. doi: 10.1145/237170.237199. URL <http://portal.acm.org/citation.cfm?doid=237170.237199>.
- [83] D. G. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, 1992. ISSN 09205691. doi: 10.1007/BF00127170.

- [84] M. Magnor and B. Girod. Data compression for light-field rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):338–343, 2000. ISSN 10518215. doi: 10.1109/76.836278.
- [85] M. Magnor, P. Ramanathan, and B. Girod. Multi-view coding for image-based rendering using 3-D scene geometry. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(11), 2003. ISSN 1051-8215. doi: 10.1109/TCSVT.2003.817630.
- [86] S. Mallat. A theory for multiresolution signal decomposition: the waveletrepresentation. *Pami*, 11(7):674–693, 1989. ISSN 0162-8828. doi: 10.1109/34.192463.
- [87] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, Boston, third edition edition, 2009. ISBN 978-0-12-374370-1.
- [88] T. Matsuyama and T. Takai. Generation, visualization, and editing of 3D video. *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*, 2002. doi: 10.1109/TDPVT.2002.1024068.
- [89] W. Matusik and H. Pfister. 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. *International Conference on Computer Graphics and Interactive Techniques*, 23(3):814, 2004. ISSN 0730-0301. doi: 10.1145/1186562.1015805. URL <http://portal.acm.org/citation.cfm?id=1015805>.
- [90] S. Matyunin and D. Vatolin. 3d video compression using depth map propagation. 368: 153–166, 2013. doi: 10.1007/978-3-642-38559-9_14. URL http://dx.doi.org/10.1007/978-3-642-38559-9_14.
- [91] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 95: 39–46, 1995. ISSN 00978930. doi: 10.1145/218380.218398. URL <http://portal.acm.org/citation.cfm?id=218398>.
- [92] P. Merkle, K. Müller, A. Smolic, and T. Wiegand. Efficient compression of multi-view video exploiting inter-view dependencies based on H.264/MPEG4-AVC. In *2006 IEEE International Conference on Multimedia and Expo, ICME 2006 - Proceedings*, volume 2006, pages 1717–1720, 2006. ISBN 1424403677. doi: 10.1109/ICME.2006.262881.

- [93] A. Miller, V. Jain, and J. L. Mundy. Real-time rendering and dynamic updating of 3-d volumetric data. *Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units - GPGPU-4*, page 1, 2011. doi: 10.1145/1964179.1964190. URL <http://portal.acm.org/citation.cfm?doid=1964179.1964190>.
- [94] K. Mueller, A. Smolic, M. Droese, P. Voigt, and T. Wienand. Multi-texture modeling of 3D traffic scenes. *2003 International Conference on Multimedia and Expo. ICME '03. Proceedings (Cat. No.03TH8698)*, 1, 2003. doi: 10.1109/ICME.2003.1221003.
- [95] E. Ordentlich, M. Weinberger, and G. Seroussi. A low-complexity modeling approach for embedded coding of wavelet coefficients. In *Data Compression Conference, 1998. DCC '98. Proceedings*, pages 408–417, Mar 1998. doi: 10.1109/DCC.1998.672180.
- [96] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*, volume 57. Springer Verlag, 2002. ISBN 0387954821. doi: 10.1115/1.1760520. URL <http://link.aip.org/link/AMREAD/v57/i3/pB15/s1&Agg=doi>.
- [97] M. R. Oswald and D. Cremers. A Convex Relaxation Approach to Space Time Multi-view 3D Reconstruction. In *ICCV Workshop on Dynamic Shape Capture and Analysis (4DMOD)*, 2013.
- [98] M. R. Oswald and D. Cremers. A convex relaxation approach to space time multi-view 3D reconstruction. *Proceedings of the IEEE International Conference on Computer Vision*, pages 291–298, 2013. doi: 10.1109/ICCVW.2013.46.
- [99] A. Perera, R. Collins, and A. Hoogs. Evaluation of compression schemes for wide area video. In *Applied Imagery Pattern Recognition Workshop, 2008. AIPR '08. 37th IEEE*, pages 1–6, Oct. 2008. doi: 10.1109/AIPR.2008.4906459.
- [100] T. Pollard and J. L. Mundy. Change Detection in a 3-d World. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6. IEEE, June 2007. ISBN 1-4244-1179-3. doi: 10.1109/CVPR.2007.383073. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4270098>.
- [101] J. P. Pons and J. D. Boissonnat. Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay triangulation. *Proceedings of the IEEE Computer Society*

- Conference on Computer Vision and Pattern Recognition*, 2007. ISSN 10636919. doi: 10.1109/CVPR.2007.383019.
- [102] T. J. Popham and A. Bhalerao. Multi-frame scene-flow estimation using a patch model and smooth motion prior. pages 1–11, 2010. URL <http://www.bmva.org/bmvc/2010/workshop/paper2/index.html>.
- [103] J. Quiroga, T. Brox, and J. Crowley. Dense Semi-Rigid Scene Flow Estimation from RGBD Images. *ECCV*, 1:1–16, 2014.
- [104] M. I. Restrepo, B. A. Mayer, A. O. Ulusoy, J. L. Mundy, S. Member, and G. S. Member. Characterization of 3-D Volumetric Probabilistic Scenes for Object Recognition. *IEEE Journal of Selected Topics in Signal Processing*, 6(5):522–537, Sept. 2012. ISSN 1932-4553. doi: 10.1109/JSTSP.2012.2201693. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6208809>.
- [105] K. Sakurada, T. Okatani, and K. Deguchi. Detecting changes in 3D structure of a scene from multi-view images captured by a vehicle-mounted camera. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 137–144, 2013. ISBN 978-0-7695-4989-7. doi: 10.1109/CVPR.2013.25.
- [106] G. Schindler and F. Dellaert. Probabilistic temporal inference on reconstructed 3D scenes. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1410–1417, 2010. ISSN 10636919. doi: 10.1109/CVPR.2010.5539803.
- [107] W. R. Schwartz, H. Pedrini, and L. S. Davis. Video Compression and Retrieval of Moving Object Location Applied to Surveillance. In *Proceedings of the 6th International Conference on Image Analysis (ICIAR)*, pages 906–916, 2009.
- [108] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999. ISSN 09205691. doi: 10.1023/A:1008176507526.
- [109] H.-W. Shen, L.-J. Chiang, and K.-L. Ma. A fast volume rendering algorithm for time-varying fields using a time-space partitioning (TSP) tree. *Proceedings Visualization '99 (Cat. No.99CB37067)*, 1999. ISSN 1. doi: 10.1109/VISUAL.1999.809910.

- [110] H. Y. Shum, K. T. Ng, and S. C. Chan. A virtual reality system using the concentric mosaic: Construction, rendering, and data compression. *IEEE Transactions on Multimedia*, 7(1):85–95, 2005. ISSN 15209210. doi: 10.1109/TMM.2004.840591.
- [111] M. Sizintsev and R. P. Wildes. Spacetime Stereo and 3D Flow via Binocular Spatiotemporal Orientation Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2241–2254, Nov. 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2014.2321373. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6809175>.
- [112] A. Smolić and D. McCutchen. 3DAV Exploration of Video-Based Rendering Technology in MPEG. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(3):348–356, 2004. ISSN 10518215. doi: 10.1109/TCSVT.2004.823395.
- [113] a. Smolic, K. Mueller, P. Merkle, T. Rein, M. Kautzner, P. Eisert, and T. Wiegand. Free viewpoint video extraction, representation, coding, and rendering. *2004 International Conference on Image Processing, 2004. ICIP '04.*, 5, 2004. ISSN 1522-4880. doi: 10.1109/ICIP.2004.1421816.
- [114] J. Starck and A. Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007. ISSN 02721716. doi: 10.1109/MCG.2007.68.
- [115] J. Starck, G. Miller, and a. Hilton. Volumetric stereo with silhouette and feature constraints. pages 1–10, 2006. doi: 10.5244/C.20.121. URL <http://epubs.surrey.ac.uk/111068/>.
- [116] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, pages 246–252. doi: 10.1109/CVPR.1999.784637. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=784637>.
- [117] D. Sun, S. Roth, and M. Black. Secrets of optical flow estimation and their principles. *Computer Vision and Pattern ...*, pages 2432–2439, June 2010. doi: 10.1109/CVPR.2010.5539939. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5539939>http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5539939.

- [118] R. Szeliski. Rapid Octree Construction from Image Sequences. *CVGIP: Image Understanding*, 58(1):23–32, 1993. ISSN 1049-9660. doi: 10.1006/ciun.1993.1029. URL <http://www.sciencedirect.com/science/article/pii/S1049966083710296>.
- [119] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *ACM SIGGRAPH Computer Graphics*, 26(2):185–194, 1992. ISSN 00978930. doi: 10.1145/142920.134037.
- [120] A. Taneja, L. Ballan, and M. Pollefeys. City-Scale Change Detection in Cadastral 3D Models Using Images. *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 113–120, 2013. doi: 10.1109/CVPR.2013.22.
- [121] D. Taubman and M. Marcellin. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, Third Printing 2004 ISBN: 9780792375197, 2004.
- [122] E. W. Thompson and J. L. Mundy. Model-based motion analysis—motion from motion. In *Proceedings of the 4th International Symposium on Robotics Research*, pages 299–309, Cambridge, MA, USA, 1988. MIT Press. ISBN 0-262-02272-9. URL <http://dl.acm.org/citation.cfm?id=57425.57457>.
- [123] A. O. Ulusoy, O. Biris, and J. L. Mundy. Dynamic Probabilistic Volumetric Models. In *ICCV*, 2013.
- [124] L. Valgaerts, A. Bruhn, H. Zimmer, J. Weickert, C. Stoll, and C. Theobalt. Joint estimation of motion, structure and geometry from stereo sequences. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6314 LNCS(PART 4):568–581, 2010. ISSN 03029743. doi: 10.1007/978-3-642-15561-1_41.
- [125] G. Van Wallendael, S. Van Leuven, J. De Cock, F. Bruls, and R. Van De Walle. Multiview and depth map compression based on HEVC. In *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, pages 168–169, 2012. ISBN 9781457702310. doi: 10.1109/ICCE.2012.6161792.
- [126] K. Varanasi, A. Zaharescu, E. Boyer, and R. Horaud. Temporal Surface Tracking using Mesh Evolution. 5303:30–43, 2008.

- [127] S. Vedula. *Image Based Spatio-Temporal Modeling and View Interpolation of Dynamic Events*. PhD thesis, Carnegie Mellon University, 2001.
- [128] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-Dimensional Scene Flow. *ICCV*, 1999.
- [129] B. A. Vetro, F. Ieee, T. Wiegand, and G. J. Sullivan. Overview of the Stereo and Multiview Video Coding Extensions of the H . 264 / MPEG-4 AVC Standard. *Proceedings of the IEEE*, 99(4):626–642, 2011. ISSN 00189219. doi: 10.1109/JPROC.2010.2098830.
- [130] C. Vogel and S. Roth. Piecewise Rigid Scene Flow. *ICCV*, 2013.
- [131] H. H. Vu, P. Labatut, J. P. Pons, and R. Keriven. High accuracy and visibility-consistent dense multiview stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):889–901, 2012. ISSN 01628828. doi: 10.1109/TPAMI.2011.172.
- [132] R. Wackerly D., Scheaffer. *Mathematical statistics with applications*. Duxbury -Thomson Learning, ISBN: 0534377416 9780534377410, 2002.
- [133] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, April 2004. ISSN 1057-7149. doi: 10.1109/TIP.2003.819861.
- [134] E. W. Weisstein. "normal sum distribution. *MathWorld—A Wolfram Web Resource*. <http://mathworld.wolfram.com/NormalSumDistribution.html>.
- [135] T. Wiegand. Overview of the H. 264/AVC video coding standard. *...and Systems for Video ...*, 13(7):560–576, 2003. ISSN 1051-8215. doi: 10.1109/TCSVT.2003.815165.
- [136] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle. Surface light fields for 3D photography. *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 287–296, 2000. doi: 10.1145/344779.344925. URL <http://doi.acm.org/10.1145/344779.344925>.
- [137] M. Ye and R. Yang. Real-time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera. *Multi-Camera ...*, (January), 2014. doi: 10.1109/CVPR.2014.301.

- [138] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L-1 optical flow. *Pattern Recognition, Proceedings*, 4713(1):214–223, 2007. doi: 10.1007/978-3-540-74936-3_22. URL <GotoISI>://WOS:000249745000022.
- [139] A. Zaharescu, E. Boyer, and R. Horaud. Topology-adaptive Mesh deformation for surface evolution, morphing, and multiview reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):823–837, 2011. ISSN 01628828. doi: 10.1109/TPAMI.2010.116.
- [140] Z. Zhang. EM algorithms for Gaussian mixtures with split-and-merge operation. *Pattern Recognition*, 36(9):1973–1983, Sept. 2003. ISSN 00313203. doi: 10.1016/S0031-3203(03)00059-1. URL <http://linkinghub.elsevier.com/retrieve/pii/S0031320303000591>.
- [141] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, pages 28–31 Vol.2, 2004. doi: 10.1109/ICPR.2004.1333992. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1333992>.
- [142] M. Zollhöfer and M. Nieß ner. Real-time non-rigid reconstruction using an rgb-d camera. *TOG*, 2014. URL <http://graphics.stanford.edu/~niessner/papers/2014/5deformables/zollhoefer2014deformable.pdf>.