Carnegie Mellon University

A Probabilistic Framework for Geometry and Motion Reconstruction Using Prior

BY

WENDE ZHANG

A Dissertation Submitted to the Graduate School in Partial Fulfillment of the Requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering

Committee members:
Prof. Tsuhan Chen (Advisor)
Prof. Vijayakumar Bhagavatula
Prof. Martial Hebert (RI, Carnegie Mellon)
Prof. José M. F. Moura
Dr. Mark J. Wolski (R&D, General Motors)

Pittsburgh, Pennsylvania
December, 2006

Acknowledgements

I would like to express my sincere gratitude to my advisor, Prof. Tsuhan Chen, for his guidance, encouragement, and support in my graduate study. His knowledge, kindness, patience, enthusiasm, and vision have brought out the best from me in my M.S. and Ph.D. study and provided me with lifetime benefits. He teaches me how to create valuable ideas, perform research, communicate efficiently, and work with team members. I feel very fortunately to join his research group and be one of his students at CMU.

I would also like to thank my other dissertation committee members: Prof. Vijayakumar Bhagavatula, Prof. Martial Hebert, Prof. José Moura, and Dr. Mark Wolski for their valuable comments and suggestions on my thesis, and also for their wonderful course instructions and enlightening discussions on projects during my entire graduate study. I would also like to thank Dr. Varsha Sedekar, my manager, and Dr. Mark Wolski, my supervisor at GM R&D, for their continuous encouragement and support of my GM internship and Ph. D. thesis work. I take this opportunity to thank many other faculty members at CMU: Richard Stern, Ozan Tonguz, Mike Reiter, Andrew Moore, and Tom Michael for their course instructions and academic guidance. I also gratefully thank Lynn Philibin, Elaine Lawrence, and Carol Patterson who made my life at CMU comfortable and enjoyable.

I also thank my wonderful AMP groupmates at CMU: Ta-Chien Lin, Deepak Turaga, Trista Chen, Howard Leung, Xiaoming Liu, Cha Zhang, Claire Fang, Edward Lin, Sam Chen, Jessie Hsu, Simon Lucey, Jack Yu, Kate Shim, Avinash Baliga, Michael Kaye, Todd Stephenson, David Liu, Akira Kubota, Carsten Schwicking, Ted Square, Amy Lu, Devi Parikh, Andrew Gallagher, Kshitiz Kumar, Nic Engel, Dhiraj Goel, Xinran Liang, Qi Wu, Brian Stancil, and Wei Yu for the enjoyable interactions and collaborations.

I thank many friends at CMU: Marios Savvides, Chunyan Xie, Derek Hoiem, Ke Yan, Xiang Li, Dapeng Wu, Jin Lu, Ying Sun, Haotian Zhang, Wen Wu, Krithika Venkataramani, Pablo Hennings-Yeomans, and Yi Zhang; and my colleagues at GM:

Jing Zhang, Yilu Zhang, Shuqing Zeng, Fan Bai, and John Dunne for their support of my research and sharing the happy time.

Finally, I express my sincere love and appreciation to my parents and my wife Lisa Yan, for their constant encouragement, support, and sacrifice. I dedicate this thesis to my parents and my wife.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

To record an exciting moment, we often capture scene activities from different viewpoints by moving the camera while capturing the video. During playback, it is desirable to have the ability to interactively control the timeline, e.g., for slow motion playback, and to control the viewpoint to view the activities.

In this thesis, we propose a probabilistic framework for reconstructing scene geometry and object motion utilizing prior knowledge of a class of scenes, for example, scenes captured by a camera mounted on a vehicle driving through city streets. In this framework, we assume the video camera is calibrated, i.e., the intrinsic and extrinsic parameters are known all the time. While we assume a single camera moving during capturing, the framework can be generalized to multiple stationary or moving cameras as well. Traditional approaches try to match the points, lines or patches in multiple images to reconstruct scene geometry and object motion. The proposed framework also takes advantage of each patch's appearance and location to infer its orientation and motion direction using prior based on statistical learning from training data. The prior hence enhances the performance of geometry and motion reconstruction. We show that prior-based 3D reconstruction outperforms traditional 3D reconstruction with synthetic data and real data for both stationary scenes and dynamic scenes, especially in the textureless areas for geometry estimation and faraway areas for motion estimation.

1. Introduction

To record an exciting moment, we often capture scene activities from different viewpoints by moving the camera while capturing the video. During playback, it is desirable to have the ability to interactively control the timeline, e.g., for slow motion playback, and to control the viewpoint to view the activities.

Scene reconstruction and rendering have been a popular research topic for decades. Given a set of images captured by one or more cameras, the goal of scene reconstruction and rendering is to reproduce a realistic image of the scene at an arbitrary time and viewpoint.

We categorize scenes by motion: stationary scenes and dynamic scenes. Stationary scenes contain no object motion, while dynamic scenes have at least one moving object.

1.1 Geometry Reconstruction/Rendering for Stationary Scenes

Image-based geometry reconstruction and rendering for stationary scenes have been intensively studied recently [1]-[6]. An early survey on various image-based geometry reconstruction and rendering techniques can be found in [7] and more recent ones are in [8][9]. A recent survey on various motion-parallax-based geometry estimation approaches to obtain a representation of geometry can be found in [10].

For a multi-camera system, in Lumigraph approach [11], a volumetric model of the captured scene was initialized by the octree construction algorithm [12] and refined by the visual hull algorithm [13]. Tomasi and Kanade [14] used an affine factorization

method to reconstruct geometry from multiple images. They assumed the orthographic projection. Pollefeys *et al.* [15] presented 3D reconstruction systems to automatically extract detailed 3D models from a sequence of images. They matched features and computed the relations between images. From this, both the structure of the stationary scene and the motion of the camera were reconstructed. Collins [16] proposed an efficient multi-image matching technique using plane-sweeping for geometry reconstruction. Recently, Akbarzadeh et al. [17] extended the plane-sweeping algorithm by sweeping planes in multiple directions for urban geometry reconstruction. Zitnick *et al.* [18] used the modified plane-sweeping algorithm to estimate the current scene's geometry with a smoothness constraint between patches and a spatial consistency constraint between images. Other methods for geometry estimation include voxel coloring [19] or stereo methods [20], etc.

1.2 Geometry and Motion Reconstruction/Rendering for Dynamic Scenes

There are mainly five approaches to reconstruct and rendering dynamic scenes: synchronized multi-view-based, model-based, scene-based or content-based, optical flow-based and time series analysis-based approaches.

Some previous work requires multiple synchronized cameras to reconstruct and render dynamic scenes. These algorithms, which were suitable for stationary scene reconstruction, could be applied to identify the current geometry, and render the scene based on the synchronized multiple cameras. Such work [6][11][17][18] used the stationary geometry estimation and view interpolation rendering without considering dynamic motion estimation or temporal consistency.

Reconstruction and rendering quality can be further improved if 3D motion is also estimated. To estimate scene geometry and object motion, some researchers assumed a model for a specific class of objects, e.g., the human body, and estimate its motion. Cheng and Moura [21] modeled the human body as an articulated object connected by joints and rigid parts, and modeled the human walking as a periodic motion. Gavrilla [22] introduced the marker-free motion capture algorithms that employed a prior model of human body. Carranza *et al.* [23] combined the marker-free motion estimation and the multi-view texture rendering to effectively synthesize a moving human at a novel viewpoint. Cheung *et al.* [24]-[26] first performed the 3D voxel-based Shape-From-Silhouette reconstruction to find the initial shape of the object. They further estimated the motion of the articulated objects over time to refine the reconstructed 3D shape. They finally rendered the tracked human motion using a view-dependent texture mapping algorithm with a human model consisting of articulated rigid body parts.

Given the segmented patches of objects and background in the scene, the scene-based video representation approach and the content-based video representation approach have been studied in [27][28] and [29] respectively. They modeled the moving foreground objects and the static background using a mosaic representation to estimate scene's geometry and motion.

Irani and Anandan [27][28] proposed to represent the scene using the scene-based mosaics with their corresponding depths. First, the static background was modeled by a single mosaic image without any moving objects. Then, the moving foreground objects were overlaid to the static background mosaic by indicating their 2D trajectories and appearances. The locations of the foreground objects were assumed to be at multiple

layers of frontal planes in 3D space. In some applications, both the foreground moving objects and the background may not be suitable for the frontal plane modeling.

Aguiar *et al.* [29] proposed to represent the scene using the content-based mosaics. If the objects were very far from the viewpoint in 3D space, both the static background and the moving foreground objects were modeled by the classical mosaic images with object shape and velocity in 2D image [30]-[33]. If the objects were close to the viewpoint, they recovered the depths of the objects in the scene due to the parallax [34][35]. They first segmented the images into patches by simply sliding a rectangular window across the images and detecting abrupt changes in the motion parameters [36]. Their proposed rank-1 weighted factorization [35] was then applied to the corresponding segmented patches to reconstruct a piecewise 3D polynomial surface and camera motion. The "sharper" feature points with the reliable 2D motion estimates were given more weights in the factorization process.

Vedula *et al.* [37][38][39] proposed an image-based spatial and temporal view interpolation algorithm for non-rigid dynamic objects using the estimated dense 3D scene flow and shape from multiple sparse distributed cameras. They developed an optical flow-based algorithm to recover the 3D scene flow and shape from multiple synchronized videos for high-quality rendering.

Some researchers used time series analysis, such as Kalman filter [40][41], the extended Kalman filter (EKF) [41], and particle filter [42] to estimate object motion and enforce the temporal consistency. Kalman filtering has been used to track feature points in video for scene reconstruction [43]. Larry *et al.* [43] proposed a Kalman filter-based algorithm for depth estimation based on the induced optical flow between adjacent

frames with a calibrated moving camera. EKF allows for more complex modeling than the Kalman filter to estimate the structure and motion of a rigid object, assuming smooth motion. Its use in vision has been refined, for example, by recursively recovering motion and geometry [44] and by assuming that all tracked points lie on a plane [45]. Particle filter provided an approximation to the tracking and were used by [46][47][48] in the template-based tracking, which adapted the number of particles used in the observation and velocity models. For scene modeling, the time-series-analysis-based algorithms typically used a static set of feature points/patches, which may not remain reliable as the scene evolves. [46] provided a framework for changing the size of the particle set over time; however, they did not make provisions for how to incorporate the particles/feature points which dynamically appeared. [49] incorporated the new feature points which best discriminated between the changing foreground object and background to improve the object tracking performance in the 2D image. [50][51][52] provided different mechanisms for generating and deleting feature points as they appeared or disappeared in the scene. However, all of them focused on tracking a sparse feature point set only of dominant features for scene modeling. [50] initialized the state of each new feature point typically using the state of its single nearest neighbor.

In contrast, for high-quality rendering, our previous work [53], as shown in Appendix A, not only tracked a dense 3D point set of both dominant and subtle features, but also initialized their underlying states (especially for subtle features) using the existing knowledge: the tracking results of their multiple neighboring states.

1.3 Prior-Based 3D Reconstruction

Most existing reconstruction approaches match points, lines or patches among multiple images for scene reconstruction. Considering that humans can easily understand the geometry structure of the scene from a single image based on prior knowledge, Hoiem *et al.* [54][55] proposed prior-based geometry estimation for outdoor stationary scenes using statistical learning. They could reconstruct a coarse 3D model from a single image by classifying each patch into ground, vertical or sky. Saxena *et al.* [56] applied supervised learning to predict the depth map of an outdoor scene from a single image. Their depth-map estimation model used a Markov Random Field that contained multi-scale local and global image features, and modeled both the depth at each individual point and the relation between depths at neighboring points.

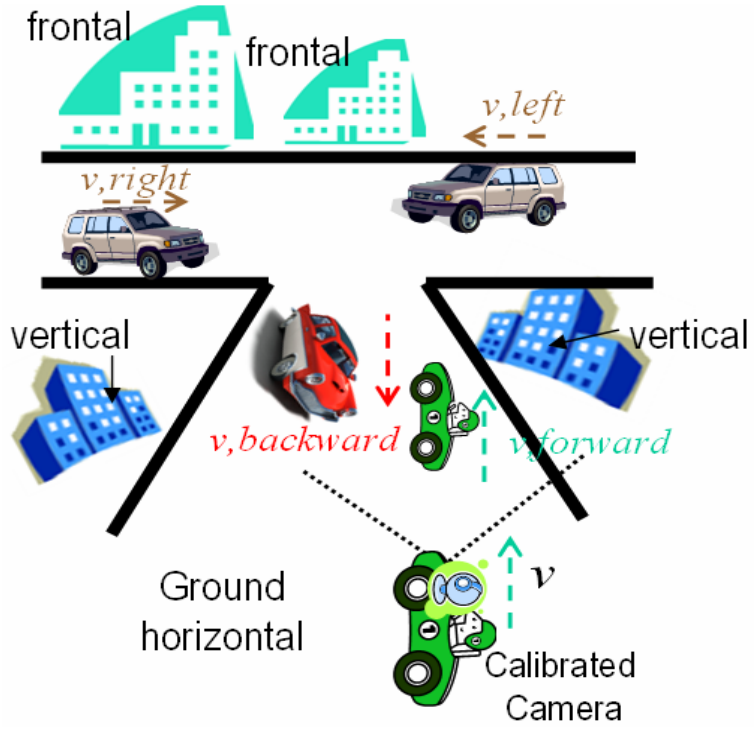


Figure 1. Illustration of scene capturing scenario

In this thesis, we reconstruct dynamic scenes from multiple images captured by a single calibrated camera mounted on a moving vehicle as shown in Figure 1. We assume

that the camera is calibrated based on the vehicle's GPS sensor, speed sensor, and gyro/yaw-rate sensor. We represent the scene by small patches with different orientations: horizontal (e.g., ground), vertical (e.g., building facets towards the street and parallel to the camera motion), and frontal (e.g., building facets towards the street and perpendicular to the camera motion); and objects (e.g., vehicles) move on the flat ground.

Figure 2. Sample images captured by a moving camera

We approximate the patch's motion directions to be either parallel or perpendicular to the camera motion (e.g., left, right, forward and backward) in a short time period. Since humans can also identify object's motion direction from a single image as shown in Figure 2, we take advantage of each image patch's appearance and location to infer its orientation and motion direction from prior information. Our prior-based geometry and motion reconstruction algorithm extends Hoiem's approach to reconstruct dense depth maps and motion maps from a moving calibrated camera as shown in Figure 3.

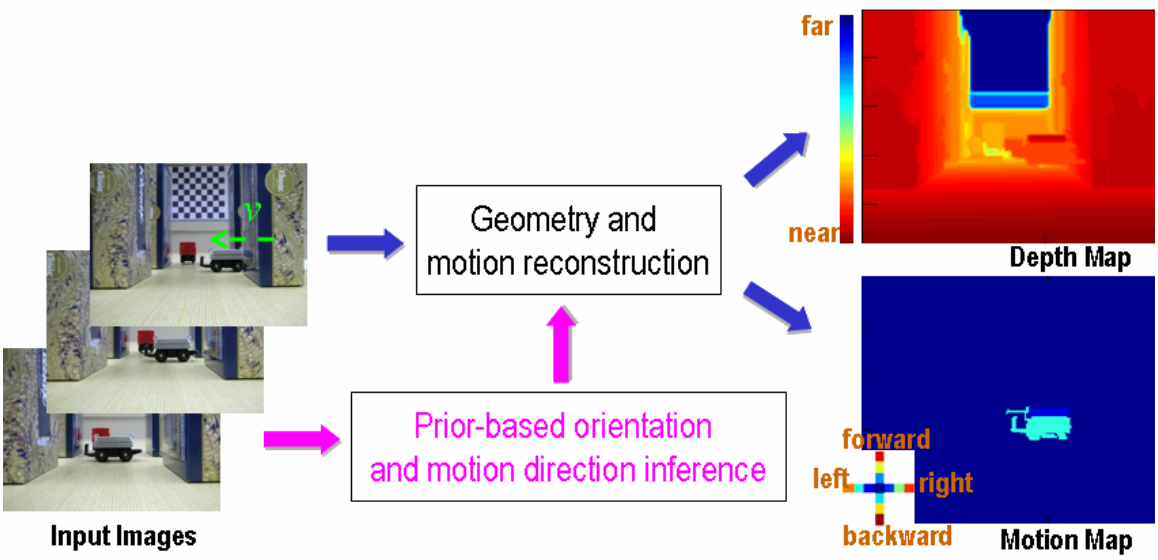


Figure 3. Prior-based geometry and motion reconstruction. The depth is represented by a color map. We assume that objects have no vertical motion. The motion vector (x, y) is quantized into fast, medium, slow or no motion in 4 directions, and represented by different colors for illustration.

1.4 Thesis Structure

The remaining of the thesis is organized as shown in Figure 4.

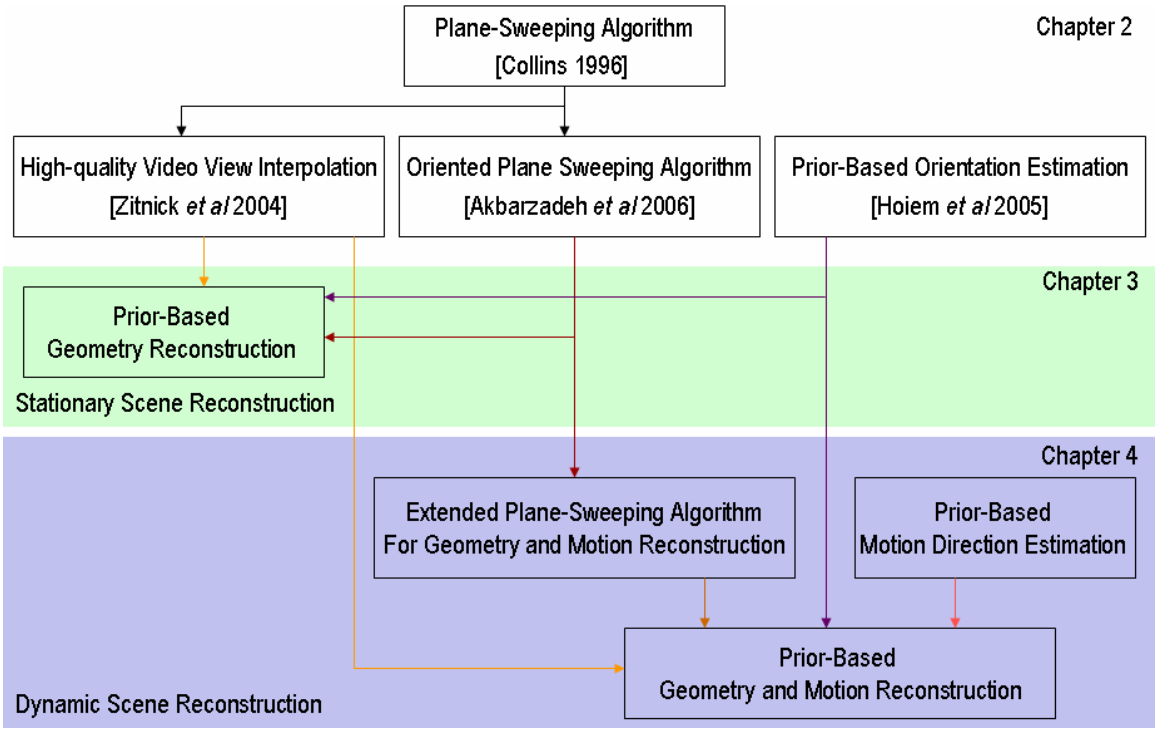


Figure 4. Algorithm outline

Chapter 2 introduces the previous work of the prior-based 3D reconstruction. We survey the previous work on 1) Collins's plane-sweeping technique for depth-map reconstruction, 2) Zitnick's modified plane-sweeping algorithm with the smoothness constraint between neighboring patches and the spatial consistency constraint between images, 3) Akbarzadeh's oriented plane-sweeping algorithm by sweeping planes in multiple directions, and 4) Hoiem's prior-based orientation estimation using statistical learning.

Chapter 3 presents our prior-based geometry reconstruction algorithm for stationary scene of combining the modified plane-sweeping algorithm, the oriented plane-sweeping algorithm, and the prior-based orientation estimation. We show an improvement of using the prior knowledge from the stationary scene reconstruction experiments.

Chapter 4 introduces our prior-based geometry and motion reconstruction algorithm for dynamic scene. We present our extended plane-sweeping algorithm by sweeping the plane in multiple directions and with different motions, and our prior-based motion direction estimation. The prior-based geometry and motion reconstruction algorithm combines the prior-based estimation of orientation and motion direction and the extended plane sweeping algorithm with a neighboring smoothness constraint and a spatial consistency constraint. We show the experimental results and compare the geometry and motion reconstruction quality of dynamic scenes between the algorithms with and without prior information. The prior-based method provides better reconstruction quality.

Finally, Chapter 5 concludes this thesis and points out the contributions. We discuss future work at the end.

2. Previous Work

The modern computer vision applications, such as view synthesis and image-based rendering, require the dense-depth-map estimates in all image regions, even those occluded or without texture. We first present the dense-depth-map estimation algorithms such as the Collins's plane-sweeping algorithm with its extensions, and then introduce Hoiem's prior-based orientation estimation for coarse 3D model reconstruction from a single image.

2.1 Plane-Sweeping Algorithm

Collins [16] proposed an efficient multi-image matching technique for depth map reconstruction based on the brightness constancy assumption as shown in Figure 5. He assumed that 2D projected points on the images corresponding to the same 3D point should have the same or similar color intensities. Therefore, if a point was at its correct depth plane, all the projected pixels should have similar values. To recover depth, a plane was swept through the space in steps along a pre-defined direction, usually orthogonal to the optical axis. At each step (depth), all images were projected onto plane's surface to test the color consistency. The estimated depth of each pixel in the current frame was the depth which provided the best color consistency score. In practice, we defined a support region for each pixel in the current frame to have a robust color consistency score. And the color consistency score at each depth could be simply defined as the sum of the absolute intensity differences between the image points within

the support region in the current frame and its corresponding points in all the other frames.

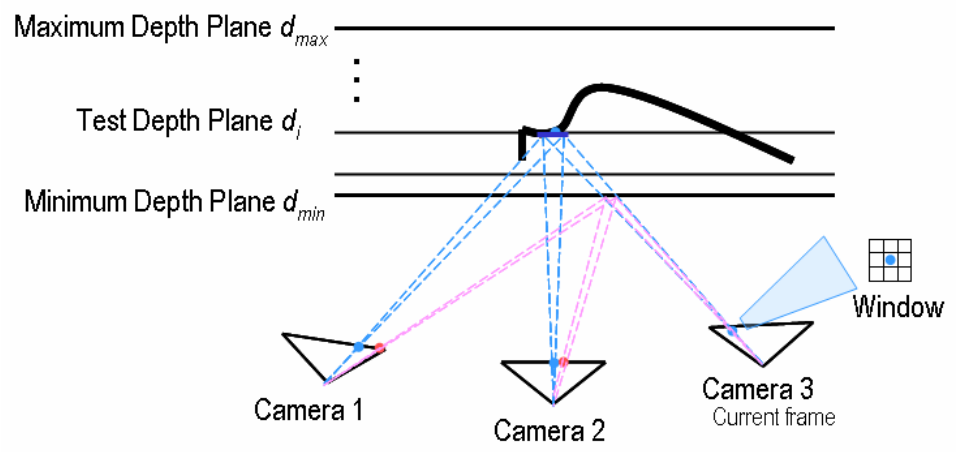


Figure 5. Plane-sweeping algorithm

2.2 Oriented Plane Sweeping Algorithm

Akbarzadeh *et al.* [17] extended the plane-sweeping algorithm for 3D urban environment reconstruction. Collins's plane-sweeping techniques swept frontal-parallel planes only. It did not account for perspective effect observed in the non-frontal surfaces, since the color consistency of each pixel was evaluated on a support region (patch) assumed to be frontal in space. They extended the algorithm by sweeping planes in multiple directions as shown in Figure 6, where the direction were aligned with the planar surfaces expected to be observed in the urban environment such as the ground and building facades. They assumed that the camera, mounted on the vehicle, moved parallel to the ground and to the building facades, and that the facades were vertical and met at right angles. They showed the better geometry reconstruction quality by sweeping planes in multiple directions.

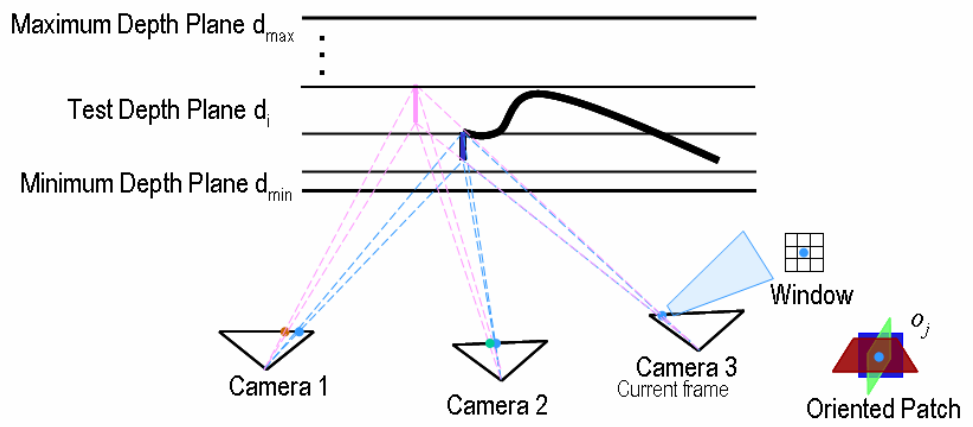


Figure 6. Oriented plane-sweeping algorithm

2.3 Modified Plane-Sweeping Algorithm with Smoothing

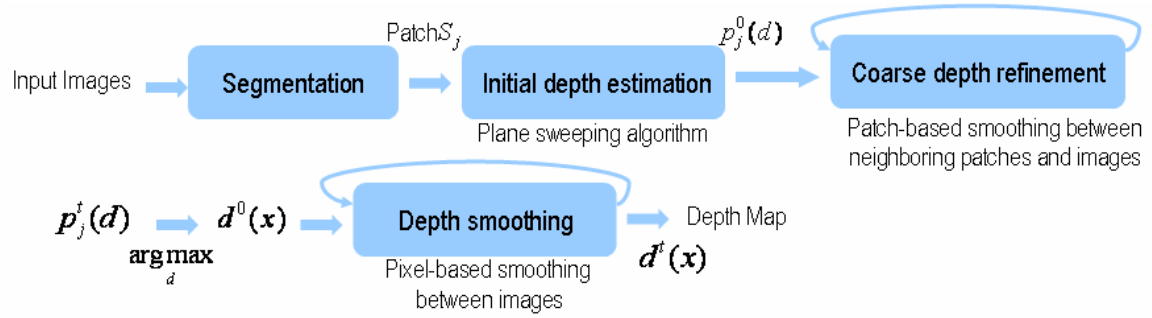


Figure 7. Depth estimation using the modified plane-sweeping algorithm

Zitnick *et al.* [18] extended the plane-sweeping algorithm with the smoothness constraint between neighboring patches and with the spatial consistency constraint between multiple images as shown in Figure 7. Their approach first segmented the input image into small patches S_j. They then calculated the color consistency of every patch in multiple images at the assumed depth d as the measure of the initial depth distribution $p_j^0(d)$ using the plane-sweeping algorithm. A coarse patch-based smoothing algorithm was applied to smooth the initial depth distribution $p_j^0(d)$ between the neighboring patches and between patch's corresponding regions at the other viewpoints iteratively.

The maximum likelihood estimates of patch's depth $\hat{d}$, based on the resulting $p_j^t(d)$, determined the initial depth map $d^0(x)$ at each pixel position x. The depth estimates were further refined iteratively per pixel among the neighboring images to have the final depth map $d^t(x)$.

2.4 Prior-Based Orientation Estimation

Hoiem *et al.* [54] proposed prior-based orientation estimation using statistical learning for outdoor scenes. They reconstructed a coarse, scaled 3D model from a single image by classifying each patch's orientation as ground, vertical or sky as shown in Figure 8.

The input image was first segmented into small patches. Color, texture and location features were then extracted from every patch. Based on their features, patches were grouped into multiple constellations that were likely to be in the same orientation category, and the likelihood of the patches in constellation having the same orientation was estimated using prior information. The generated multiple overlapping sets of possible constellations was used to help determine the final patch's orientation based on constellation's features (color, texture, location, shape and 3D geometry) using prior statistics. Each patch's orientation likelihood was determined by the constellation's orientation likelihoods and the likelihoods of the patches in the constellation having the same label.

They cut and folded the image, and reconstructed a 3D model of a scene directly based on patch's orientation estimated in a single image.

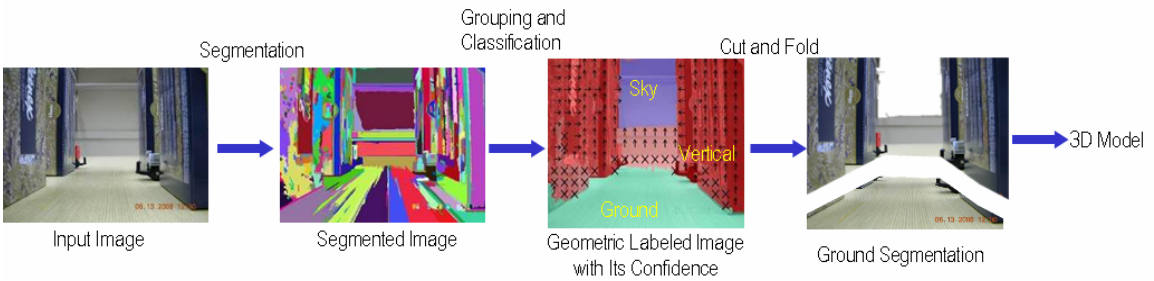


Figure 8. Prior-based orientation estimation for 3D modeling

3. Prior-Based Geometry Reconstruction of a Stationary Scene

In this chapter, we describe the prior-based geometry reconstruction of a stationary scene using a calibrated moving camera. We first provide an overview of the prior learning method and the prior-based geometry reconstruction method, then introduce each component in detail, and finally show the experimental results.

3.1 Overview

As shown in Figure 9, for prior learning, we first segment the training images into patches. We then train the orientation estimator based on the labeled patches.

For the prior-based geometry reconstruction, input images are first segmented into patches S_j. We then infer each patch's orientation distribution $P_j(o)$ using the orientation estimator. We calculate the color consistency of every patch among multiple images at the assumed depth d with a given orientation o to estimate the conditional probability $P_j(d \mid o)$. The initial likelihood of patch's geometry and motion $P_j^0(d,o)$ is approximated by the product of the prior probability $P_j(o)$ and the conditional probability $P_j(d \mid o)$. A coarse patch-based smoothing algorithm is then applied to refine the initial geometry likelihood $P_j^0(d,o)$ between its neighboring patches and between its corresponding regions at different viewpoints iteratively. The maximum likelihood estimates of patch's depth $\hat{d}$ and orientation $\hat{o}$, based on the resulting $P_j^t(d,o)$, determine the initial depth map $d^0(x)$ and the orientation map at each pixel position x.

The initial depth map $d^0(x)$ is further smoothed iteratively per pixel between images to create the refined depth map $d^t(x)$. Next we will explain each of these steps in more detail.

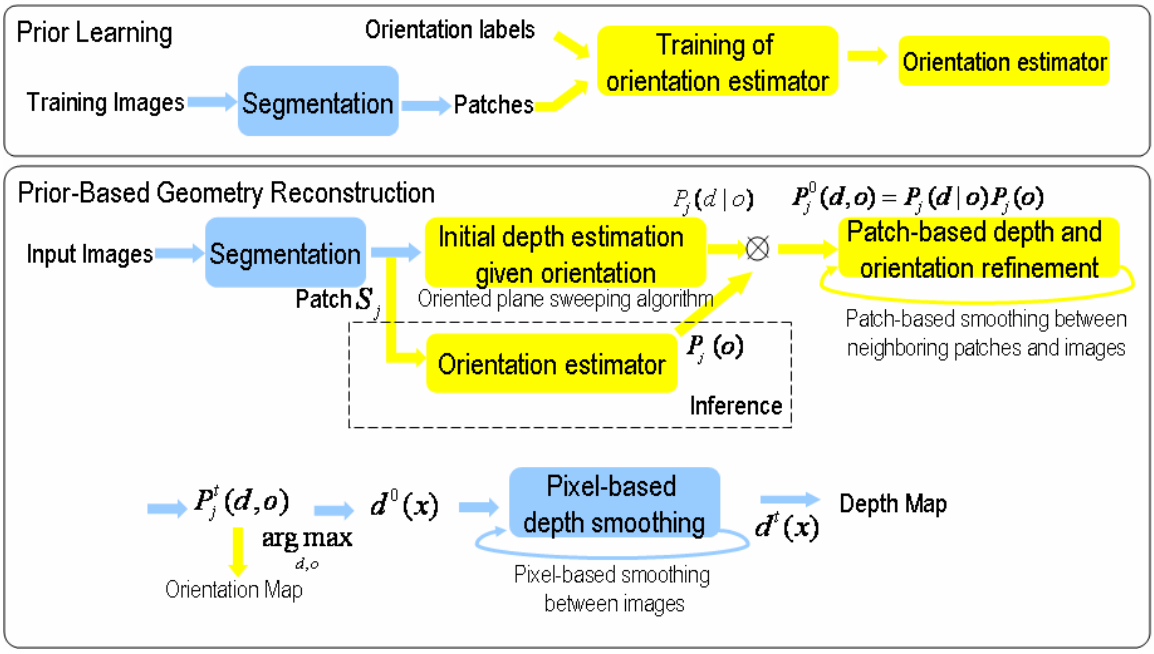


Figure 9. Prior learning and prior-based geometry reconstruction

3.2 Image Segmentation

Each image is represented by RGB pixels. The image pixels are grouped into small patches based on their intensities. The use of patches improves the computational efficiency of our algorithm to find the depth for each pixel in the image and allows complex statistics to be extracted for prior knowledge estimation. We use the efficient graph-based image segmentation technique proposed by Felzenszwalb and Huttenlocher [57].

3.3 Prior-based Orientation Estimation

Prior-based orientation estimation contains two stages: learning and inference. In the prior learning stage, we first extract the features from the training patches, and then train the orientation estimator using the patch features and the corresponding orientation labels provided.

Similar to human vision system, texture, color and location features are extracted from each patch as shown in Figure 10. The texture feature is the 15 mean values of the absolute responses of the Leung-Malik (LM) filter bank [59]. The color feature is the 6 mean values of RGB and HSV. And the location feature is the 2D mean location in the image coordinates. A detail discussion about the LM filter bank is attached in Appendix B.

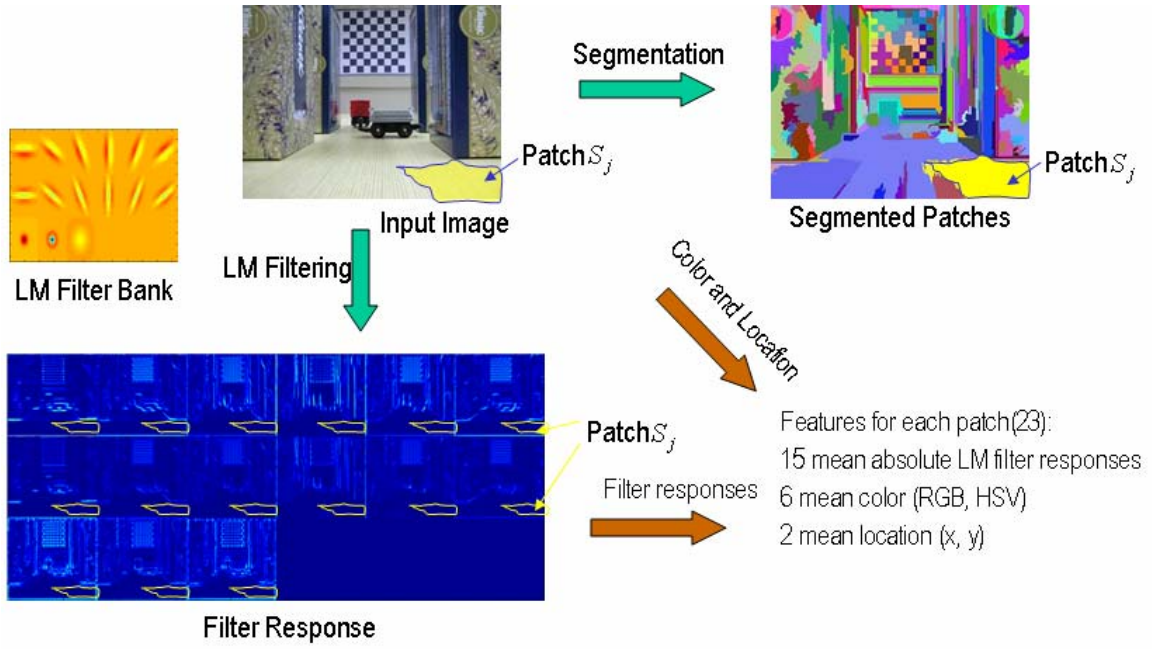


Figure 10. Feature extraction

For the training of the orientation estimator, we require patch's orientation label information (frontal, vertical, horizontal or sky) to be provided as shown in Figure 11.

Sky is treated as a special category, which is farthest-away frontal patch from the camera. We train the orientation estimator using Support Vector Machines (SVM) probability estimation [58] based on the labeled patch features. Compared to [54], we apply a weaker statistic learning approach only using patch's features without further grouping the patches.

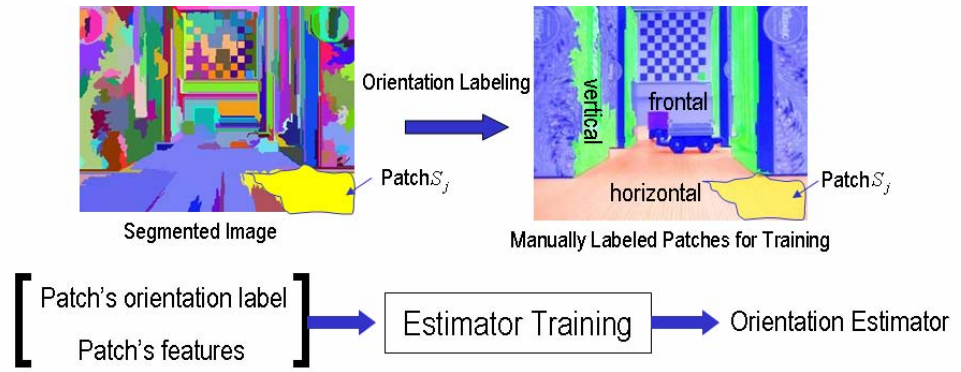


Figure 11. Training of the orientation estimator

In the inference stage, we calculate the prior distributions of patch's orientation. We first extract patch S_j's features, and then determine its orientation distribution $P_j(o)$ using the orientation estimator. The SVM-based estimator provides the probabilities of all possible orientations as shown in Figure 12.

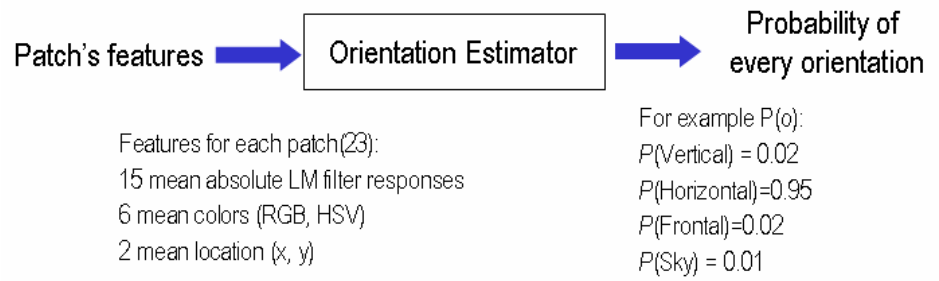


Figure 12. Patch's orientation distribution inference

3.4 Initial Geometry Estimation of Stationary Scene

The initial distribution of the patch's geometry $P_j^0(d,o)$ is evaluated by the product of the orientation probability $P_j(o)$ and the conditional probability $P_j(d \mid o)$ of the patch's depth *d* given the orientation *o*.

$$P_j^0(d,o) = P_j(o)P_j(d \mid o) \tag{1}$$

The conditional probability $P_j(d \mid o)$ is determined based on color consistency between images using the oriented plane-sweeping algorithm with the given orientation as discussed in Section 2.2. Patch S_j's depth with the given orientation is evaluated by its color consistency in multiple images at the current viewpoint (Camera 3) as illustrated Figure 6. (In our scenario, the camera would be moving towards the scene, which is different from the illustration.) We compare the RGB color difference between every pixel in patch S_j at the current viewpoint and its corresponding pixels at the other viewpoints *k*, $k = 1...N$, (in Camera 1 and Camera 2) to measure the color consistency $e_{diff}(S_j)$ using the following robust function with a parameter *th*:

$$e_{diff}(S_j) = \frac{1}{\text{num}_{S_j}} \sum_k \sum_{pixel \in S_j} \frac{\gamma_k^2}{\gamma_k^2 + th^2}, \tag{2}$$

where $\gamma_k = \left|r_{cur} - r_{cor,k}\right| + \left|g_{refcur} - g_{cor,k}\right| + \left|b_{cur} - b_{cor,k}\right|$ is the RGB color difference, and num_{S_j} is the number of the pixels in S_j.

The conditional probability $P_j(d \mid o)$ is determined by the color consistency measures $e_{diff}(S_j)$:

$$P_j(d \mid o) = \frac{g(d,o)}{\sum_{d'} g(d',o)}, \text{ where } \; g(d,o) = 1 - e_{diff}(S_j). \tag{3}$$

3.5 Patch-Based Smoothing for Stationary Scene

We refine patch's initial distribution $P_j^t(d,o)$ between its neighboring patches and between its corresponding regions at multiple viewpoints iteratively, which is similar to [18], with the extension of smoothing for additional orientation estimation. We enforce a smoothness constraint that the neighboring patches (S_j and s_l in Frame 1) with similar colors (blue) should have similar depths ($d \approx d_i$) and the same orientations (o = vertical) as shown in Figure 13. We also ensure scene's geometry consistency constraint between images. If we project a patch (S_j) with its ground-truth depth d and orientation o into a neighboring image, the projected region (in Frame 2) should have the similar depth and the same orientation if not occluded.

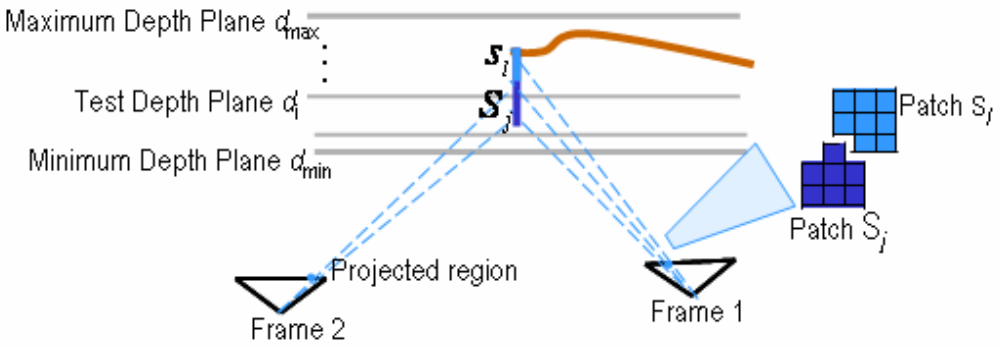


Figure 13. Patch-based smoothing for stationary scenes

The likelihood distribution of the patch's geometry $P_j^t(d,o)$ is updated iteratively with two constraints.

$$P_j^{t+1}(d,o) = \frac{n_j(d,o)\prod_{k\in N} c_{j,k}(d,o)}{\sum_{d',o'} n_j(d',o')\prod_{k\in N} c_{j,k}(d',o')} \tag{4}$$

where $n_j(d,o)$ enforces patch's smoothness constraint and $c_{j,k}(d,o)$ enforces patch's consistency constraint in each projected region at multiple viewpoints k, $k = 1\ldots, N$. The details are given below.

3.5.1 Patch-Based Smoothness Constraint for Stationary Scene

The geometry smoothness coefficient $n_j(d,o)$ enforces that the neighboring patches with similar colors should have similar depths and the same orientation. Let s_l denote one of patch S_j's neighboring patches as shown in Figure 13. $\hat{d}_l$ and $\hat{o}_l$ are the maximum likelihood estimates of its depth and orientation based on $P_l^t(d,o)$.

$$\left\langle \hat{d}_l \quad \hat{o}_l \right\rangle = \arg\max_{d,o} P_l^t(d,o) \tag{5}$$

We assume that if patches S_j and s_l have the same orientation, the depth d of patch S_j is modeled by a contaminated Gaussian distribution with the mean $\hat{d}_l$ and variance σ_l^2. We define $n_j(d,o)$ to be:

$$n_j(d,o) = \begin{cases} \prod_{s_l} N(d;\hat{d}_l,\sigma_l^2) + \varepsilon & o = \hat{o}_l \\ \varepsilon & o \neq \hat{o}_l \end{cases} \tag{6}$$

where $N(x;mean,\sigma^2)$ is the Gaussian distribution, and ε and c are small constants (e.g. 10^{-10}). We estimate the variance σ_l^2 using color similarity, neighboring measure and patch s_l's geometry maximum likelihood as defined below:

1. The color similarity of the patches $\Delta_{j,l}$, which measures the color difference between patches S_j and s_l.
2. The neighboring measure $b_{j,l}$, which is the percentage of the patch S_j's border between patches S_j and s_l.

3. The geometry maximum likelihood for patch s_l: $P_l(\hat{d}_l, \hat{o}_l)$, which represents the accuracy of the maximum likelihood estimates for patch s_l's geometry.

σ_l^2 is defined to be:

$$\sigma_l^2 = \frac{v}{P_l(\hat{d}_l, \hat{o}_l)^2 b_{j,l} N(\Delta_{j,l}; 0, \sigma_\Delta^2)} \tag{7}$$

where v and σ_Δ^2 are constants ($v = 8$ and $\sigma_\Delta^2 = 30$ in our experiment). Therefore, if patch S_j and its neighboring patch s_l have similar colors, and patch S_j's depth and orientation are consistent with its neighbor's depth and orientation maximum likelihood estimates, we expect $n_j(d, o)$ to be large.

3.5.2 Patch-Based Consistency Constraint for Stationary Scene

The spatial consistency coefficient $c_{j,k}(d, o)$ ensures that the patch S_j's depth and orientation estimates are consistent with the depth and orientation estimates at the viewpoint k. We compute $c_{j,k}(d, o)$ based on spatial consistency, visibility, and patch S_j's initial geometry likelihood:

1. Spatial consistency without occlusion. We first project patch S_j with the depth d and orientation o onto a neighboring image. We then calculate patch S_j's projecting distributions $b_{j,k}^t(d, o)$ based on the geometry distribution at the projected viewpoint k to estimate the spatial consistency without occlusion.

$$b_{j,k}^t(d, o) = \frac{1}{num_{S_j}} \sum_{x \in S_j} P_{r(k,x)}^t(d, o) \tag{8}$$

where $r(k,x)$ is the patch index at the viewpoint k, on which the corresponding pixel of the pixel position x on patch S_j is. And num_{S_j} is the number of the pixels on patch S_j. If the

projected region's depth and orientation maximum likelihood estimates are consistent with patch S_j's estimates, we expect $b^t_{j,k}(d,o)$ to be large when patch S_j is visible at the viewpoint k (Frame 2) as shown in Figure 14.

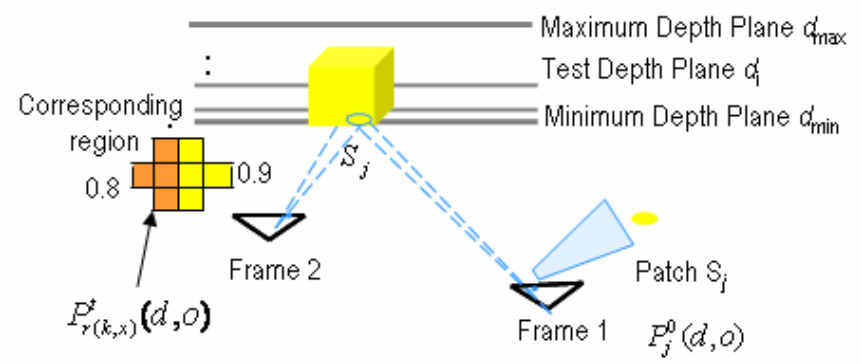


Figure 14. Yellow patch is visible in Frame 2 of a stationary scene.

2. Visibility. Due to the possible occlusions, a patch might not have the corresponding pixels at another viewpoint as shown in Figure 15. We estimate the overall visibility likelihood $v_{j,k}$, that the patch is visible, as follows:

$$v_{j,k} = \min\left(1.0, \sum_{d',o'} b^t_{j,k}(d',o')\right) \tag{9}$$

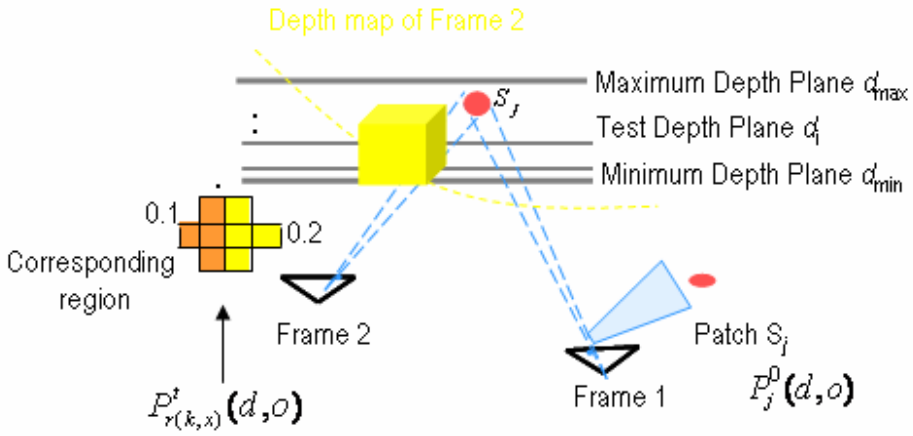


Figure 15. Red patch is occluded in Frame 2 of a stationary scene.

If the patch S_j is visible at the viewpoint k (Frame 2) as shown in Figure 14, we can find its corresponding region when we search the space of the depth d and orientation o. The ground-truth solution and its neighboring solutions offer large $b^t_{j,k}(d',o')$ values. If the patch S_j is occluded at the viewpoint k (Frame 2) as shown in Figure 15, we can not find its corresponding region when we search the space of the depth d and orientation o. No solution provides large $b^t_{j,k}(d',o')$ value. Therefore, we use $v_{j,k}$ as a robust and computational-efficient measure of patch's visibility.

We also estimate the specific visible likelihood $vc_{j,k}(d,o)$ that given the depth d and orientation o, patch S_j is visible at the viewpoint k.

$vc_{j,k}(d,o) = \frac{1}{num_{S_j}} \sum_{x \in S_j} P^t_{r(k,x)}(d,o) h(\hat{d}_{r(k,x)} - d)$, where $h(x)$ is the Heaviside step function and $\hat{d}_{r(k,x)}$ is the maximum likelihood depth estimate of patch $s_{r(k,x)}$.

This suggests that if patch S_j is visible at the viewpoint k, its estimated depth should not be under the surface of the estimated depth map at the viewpoint k as shown in Figure 15.

3. Initial geometry likelihood $P^0_j(d,o)$.

Now, we combine the visible and occluded cases. If the patch is visible, $c_{j,k}(d,o)$ is calculated from the visible consistency likelihood $b^t_{j,k}(d,o)P^0_j(d,o)$. Otherwise, its occluded consistency likelihood is $(1 - vc_{j,k}(d,o))P^0$, where the uniform prior $P^0 = \frac{1}{size(d)size(o)}$. $size(d)$ and $size(o)$ are the sizes of the depth and orientation hypothesis spaces. Therefore,

$$c_{j,k}(d,o) = v_{j,k} b^t_{j,k}(d,o) P^0_j(d,o) + (1 - v_{j,k})(1 - vc_{j,k}(d,o)) P^0 . \quad (10)$$

3.6 Pixel-Based Depth Smoothing for Stationary Scene

The maximum likelihood estimates of each patch's depth $\hat{d}$ and orientation $\hat{o}$ based on $P^t_j(d,o)$ determine the initial depth map $d^0(x)$ and the orientation map for each pixel x.

$$\left\langle \hat{d}_l \quad \hat{o}_l \right\rangle = \arg\max_{d,o} P^t_l(d,o) \quad (11)$$

We further refine the depth map $d^t(x)$ iteratively between images [18]. For each pixel x at the current viewpoint, we find its corresponding pixel y at the neighboring viewpoint k. If the corresponding pixel's depth $d^t_k(y)$ is similar to pixel x's depth $d^t(x)$, the pixel x's depth $d^{t+1}(x)$ is replaced by the average of $d^t(x)$ and $d^t_k(y)$. The iterative updating equation is

$$d^{t+1}(x) = \frac{1}{N} \sum_k \left(\delta_{d,k} \frac{d^t(x) + d^t_k(y)}{2} + (1 - \delta_{d,k}) d^t(x) \right) \quad (12)$$

where $\delta_{d,k} = | d^t(x) - d^t_k(y) | < \lambda_d$ is the indicator variable (0,1), testing input similarity with the threshold parameter λ_d, and N is the number of the neighboring images. After smoothing the depth maps across different viewpoints, we apply a 3x3 average filter to further smooth the results.

3.7 Experimental Results

We first showed the experimental results of the prior-based orientation estimation based on a single image.

We trained the SVM-based orientation estimator with 6670 labeled patches, extracted from 48 color images at 320x240 pixels. The images for training the orientation estimator were shown in Figure 16.

Figure 16. Images for training the orientation estimator

We inferred the orientation distribution of each image patch using the orientation estimator. In Figure 17, we showed the classification results of the orientation estimator on a sample image with the maximum likelihood estimates represented by the shaded colors: red (horizontal), green (vertical), and blue (frontal). It achieved the classification accuracy: 85% of the patches correctly labeled.

(a) Sample image (b) Classification results

Figure 17. Prior-based orientation estimation results

We also compared our simple orientation estimator with Hoiem's orientation estimator [54] using their online database [60]. We trained and tested our orientation estimator on their training data in Figure 18 and testing data in Figure 19, respectively. On a test set of 62 novel images, Hoiem reported that 87% of the pixels were correctly labeled into ground, vertical, or sky. We achieved the accuracy of 85% of the pixels correctly labeled, while our simple algorithm ran more than 3 times faster than Hoiem's algorithm as shown in Table 1.

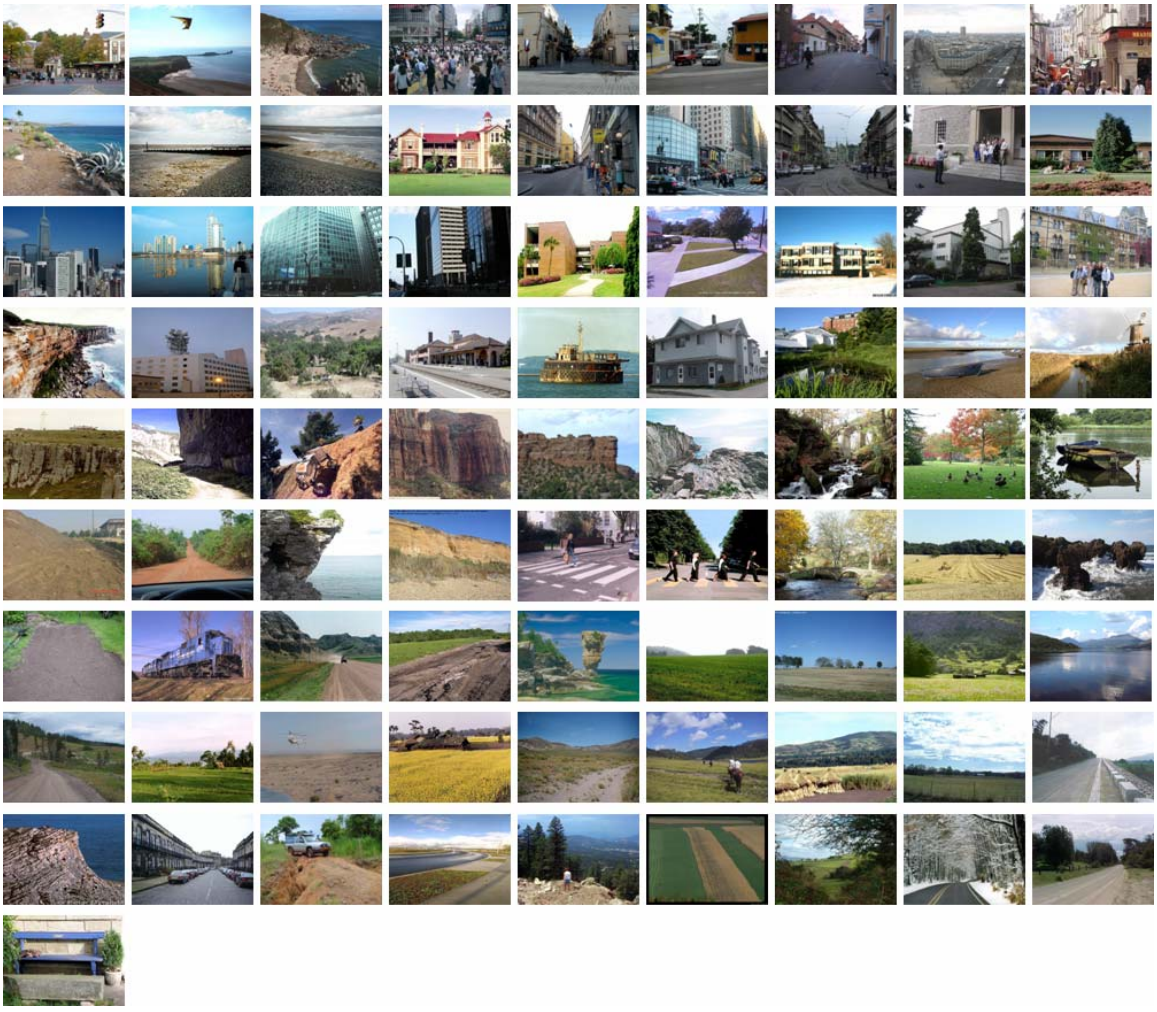


Figure 18. Images for training the orientation estimator in Hoiem's database

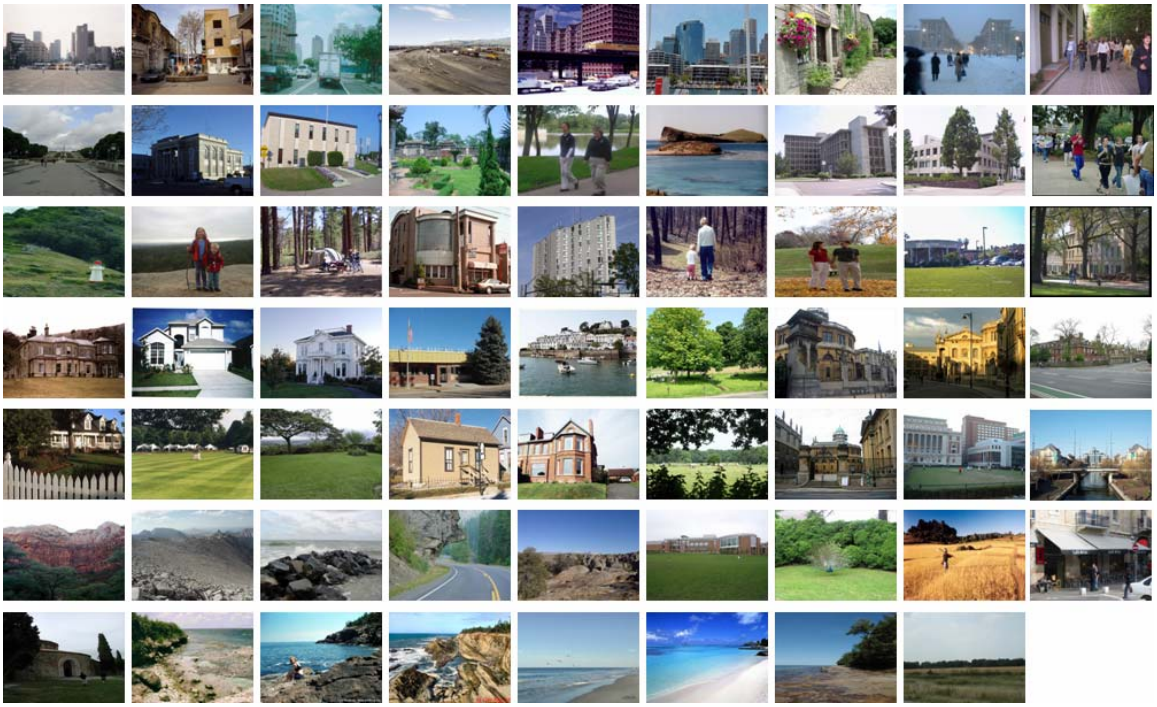


Figure 19. Images for testing the orientation estimator in Hoiem's database

Table 1. Performance comparison of orientation estimators

	Our algorithm	Hoiem's algorithm
Classification Accuracy	85%	87%
Time/Frame	1.8 sec	7.6 sec

Next, we showed the experimental results of the prior-based geometry reconstruction of stationary scenes.

We ran experiments on multiple synthetic images of a stationary street simulated by POV-ray [61]. As illustrated in Figure 20, six images were captured by a backward moving camera at 320x240 pixels with the known intrinsic and extrinsic camera calibration parameters as the input image sequence.

Figure 20. Input images of a stationary street

Each image was segmented into small patches, and patch's prior orientation probabilities were inferred based on patch's appearance and location. We applied the prior-based geometry reconstruction algorithm on these input images to reconstruct the depth map at each viewpoint as shown in Figure 21.

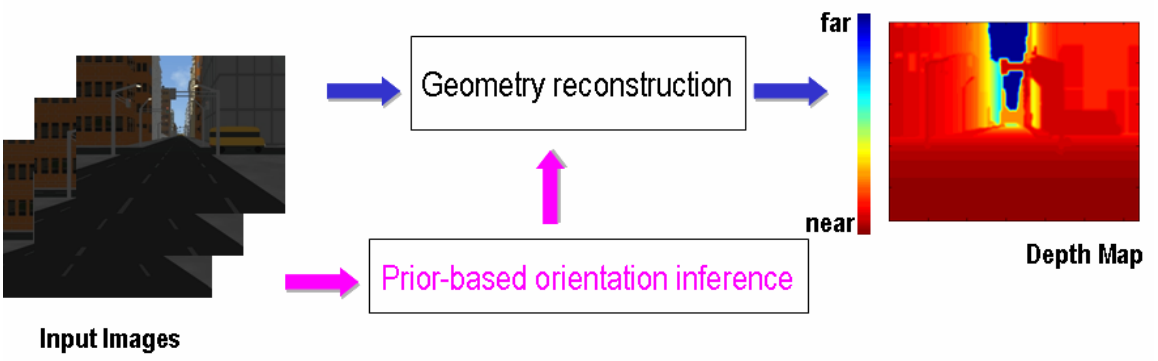


Figure 21. Prior-based geometry reconstruction. The depth is represented by a color map.

We compared the results of the proposed prior-based algorithm with the estimated prior distribution $P_j(o)$ and the results without using any prior, which were the oriented plane-sweeping algorithm [17] and our smoothed version of the oriented plane-sweeping algorithm with the neighboring patch smoothness constraint and the spatial consistency constraint for geometry reconstruction.

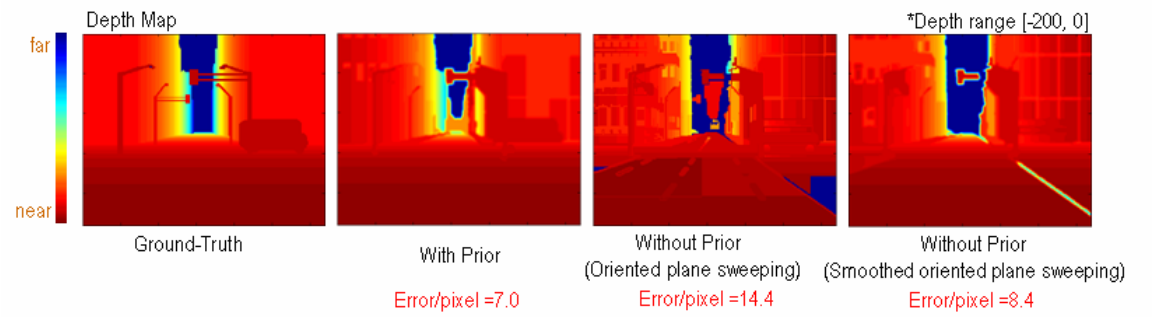


Figure 22. Depth map comparison of different algorithms for stationary scene reconstruction.

We compared the resulting depth maps at the first frame's viewpoint as shown in Figure 22. The dynamic range of depth map is assumed to be 200. Compared with the ground-truth depth map, the prior-based method provided the reconstructed depth map with 7/200=3.5% error per pixel on average, which is better than the traditional approaches. The oriented plane sweeping algorithm offered 14.4/200=7.2% reconstruction error per pixel. The smoothed oriented plane-sweeping algorithm with the smoothness constraint and the consistency constraint achieved better and smoother result (with 8.4/200=4.1% error per pixel) than the oriented plane-sweeping algorithm. The algorithms without prior knowledge had the difficulty in reconstructing the depth of school bus and ground areas as shown in Figure 22.

We also showed the experimental results in a real garden scene. A forward moving camera captured seven input images at 320x240 pixels as shown in Figure 23. We calibrated the camera's intrinsic parameters (camera's focal length and optical center) with checker board patterns offline and the extrinsic parameters (the translation vector and the rotation matrix) with markers on the ground using Zhang's method [62].

Figure 23. Input images of a stationary garden with a forward moving camera

We applied the prior-based geometry reconstruction algorithm on these input images to reconstruct the depth map and orientation map at each viewpoint. We compared the results of the proposed prior-based algorithm with the estimated prior distribution $P_j(o)$ and the results without using any prior, which were the oriented plane-sweeping algorithm [17] and our smoothed version of the oriented plane-sweeping algorithm for geometry reconstruction. The prior-based method provided better orientation estimation than the traditional approaches, especially in the textureless areas (ground and sky) in Figure 24. Although the smoothed oriented plane-sweeping algorithm had better and smoother results than the oriented plane-sweeping algorithm, it was still difficult to find the correct depth and orientation in the textureless areas without any prior knowledge. Therefore, the prior-based method had better estimated depth maps than the uniform prior approaches in Figure 25.

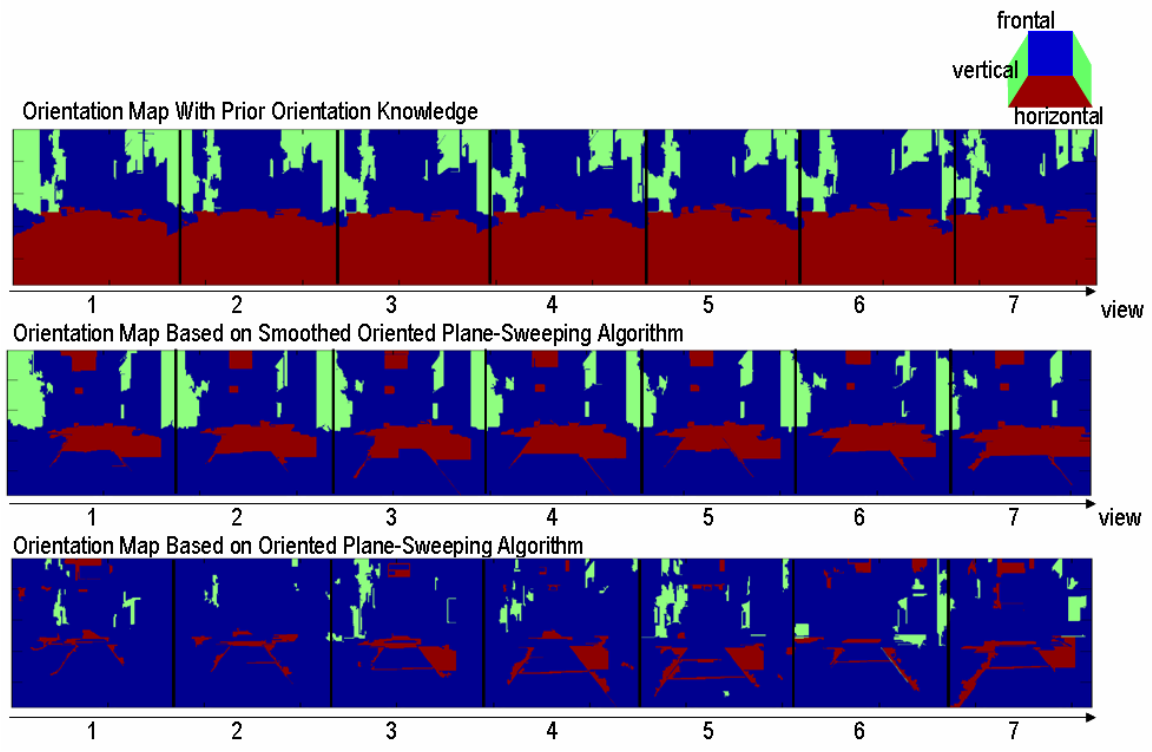


Figure 24. Orientation map comparison on a stationary garden scene

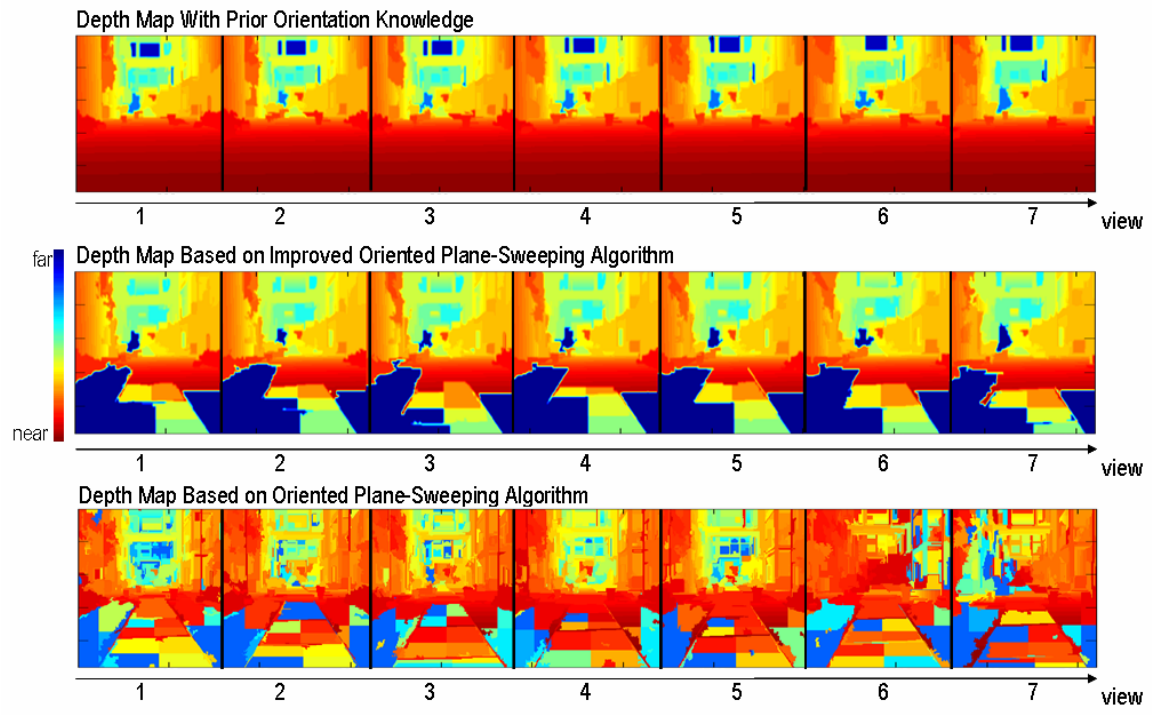


Figure 25. Depth map comparison on a stationary garden scene

4. Prior-Based Geometry and Motion Reconstruction of a Dynamic Scene

In this chapter, we describe the prior-based geometry and motion reconstruction of a dynamic scene using a calibrated moving camera. We first provide an overview of the prior learning method and the prior-based geometry and motion reconstruction of dynamic scenes, then introduce each component in detail, and finally show the experimental results with the comparison of the reconstruction quality between the algorithms with and without prior information.

4.1 Overview

As shown in Figure 26, for prior learning, we first segment the training images into patches. We then train the orientation estimator and the motion direction estimator using the labeled patches.

For the prior-based geometry and motion reconstruction, input images are first segmented into patches S_j. We then infer each patch's orientation distribution $P_j(o)$ and motion direction distribution $P_j(m_{dir})$ based on image patch's appearance and location using the orientation estimator and the motion direction estimator, respectively. The motion magnitude distribution $P_j(m_{mag})$ is heuristic and application-dependent. Meanwhile, we calculate the color consistency of every patch in multiple images at the assumed depth d with a given orientation o and motion vector $\mathbf{m}$ to estimate the

conditional probability $P_j(d \mid o, \mathbf{m})$, where $\mathbf{m} = [m_{dir}, m_{mag}]$. The initial likelihood of patch's geometry and motion $P_j^0(d, o, \mathbf{m})$ is approximated by the product of the probabilities $P_j(o)$, $P_j(m_{dir})$ and $P_j(m_{mag})$, and the conditional probability $P_j(d \mid o, \mathbf{m})$. A coarse patch-based smoothing algorithm is then applied to refine the initial geometry and motion likelihood $P_j^0(d, o, \mathbf{m})$ between its neighboring patches and between its corresponding regions at multiple times/viewpoints iteratively. The maximum likelihood estimates of patch's depth $\hat{d}$, orientation $\hat{o}$ and motion vector $\hat{\mathbf{m}}$, based on the resulting $P_j^t(d, o, \mathbf{m})$, determine the initial depth map $d^0(x)$, the orientation map, and the initial motion map $\mathbf{m}^0(x)$ at each pixel position x. The initial depth map $d^0(x)$ and the initial motion map $\mathbf{m}^0(x)$ are further smoothed iteratively per pixel between images to have the refined depth map $d^t(x)$ and the refined motion map $\mathbf{m}^t(x)$. Next we will explain each of these steps in more detail.

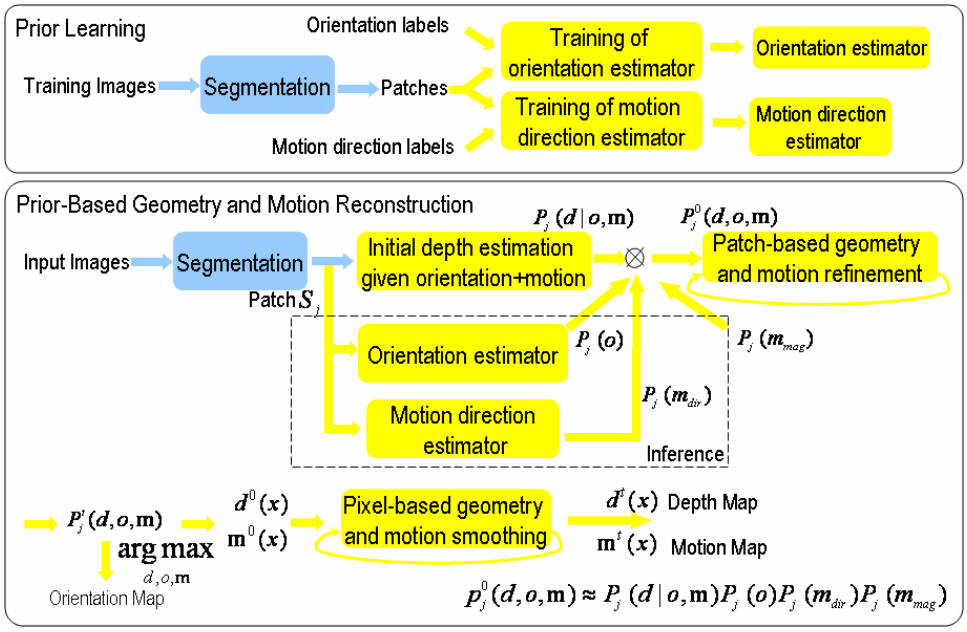


Figure 26. Prior learning and prior-based geometry and motion reconstruction

4.2 Prior Estimation

Prior estimation contains two stages: learning and inference. In the prior learning stage, we first extract the features from the segmented patches, and then train the estimators using the patch features and the label information provided.

For the training of the motion direction estimator as shown in Figure 27, we require patch's motion direction label information (stationary, forward, backward or left/right) to be provided. We train the motion direction estimator using the same approach to train the orientation estimator discussed in Section 3.3.

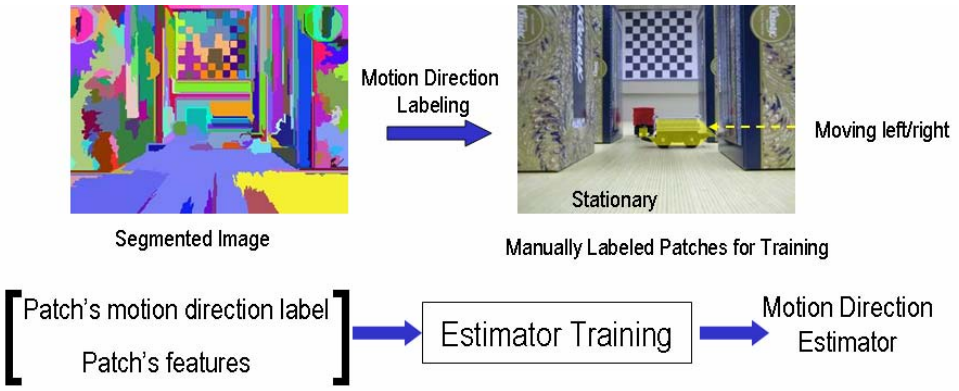


Figure 27. Training of the motion direction estimator

In the inference stage, we calculate the prior distributions of patch's orientation and motion direction. We first extract patch S_j's features, and then determine its orientation distribution $P_j(o)$ and motion direction distribution $P_j(m_{dir})$ using the orientation estimator and motion direction estimator, respectively. The SVM-based motion direction estimator also provides the probabilities of all possible labels as shown in Figure 28.

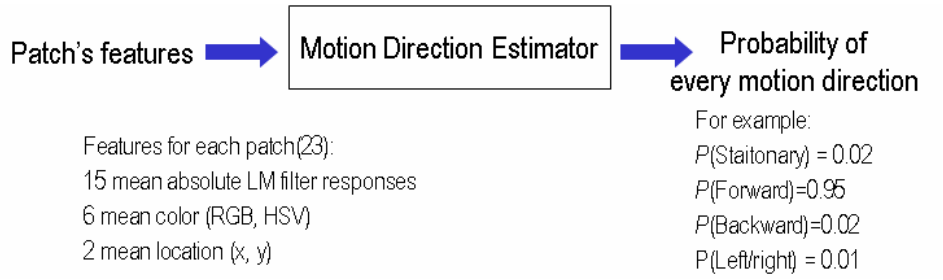


Figure 28. Patch's motion direction distribution inference

The prior distribution $P_j(m_{mag})$ is approximated by a heuristic distribution $P_j(m_{mag}) \propto \exp(-a \cdot m_{mag})$ with a pre-determined value *a* in our experiment.

4.3 Initial Motion and Geometry Estimation of Dynamic Scene

The initial distribution of patch geometry and motion $P_j^0(d, o, \mathbf{m})$ is evaluated by the product of the prior probabilities $P_j(o)$, $P_j(m_{dir})$, $P_j(m_{mag})$, and the conditional probability $P_j(d \mid o, \mathbf{m})$ of patch's depth *d* given the orientation *o* and motion **m,** where $\mathbf{m} = [m_{dir}, m_{mag}]$.

$$P_j^0(d, o, \mathbf{m}) = P_j(d \mid o, \mathbf{m}) P_j(o, \mathbf{m}), \tag{13}$$

where $P_j^0(o, \mathbf{m}) = P_j^0(o, m_{dir}, m_{mag}) \approx P_j(o) P_j(m_{dir}) P_j(m_{mag})$, since we assume that orientation *o*, motion direction m_{dir}, and motion magnitude m_{mag} are statistically independent of each other.

The conditional probability $P_j(d \mid o, \mathbf{m})$ is determined based on color consistency between images using our extended plane-sweeping algorithm with the given orientation and motion. We assume that scene patches follow a constant motion in a short time period. Patch S_j's depth with the given orientation and motion is evaluated by its color

consistency in multiple images at the current viewpoint (Camera 2) as illustrated in Figure 29. (In our scenario, the camera would be moving towards the scene, which is different from the illustration.) We compare the RGB color difference between every pixel in patch S_j at the current viewpoint and its corresponding pixels with motion shift at the other times/viewpoints k, $k = 1...N$, (in Camera 1 and Camera 3) to measure the color consistency $e_{diff}(S_j)$ defined in Section 3.4.

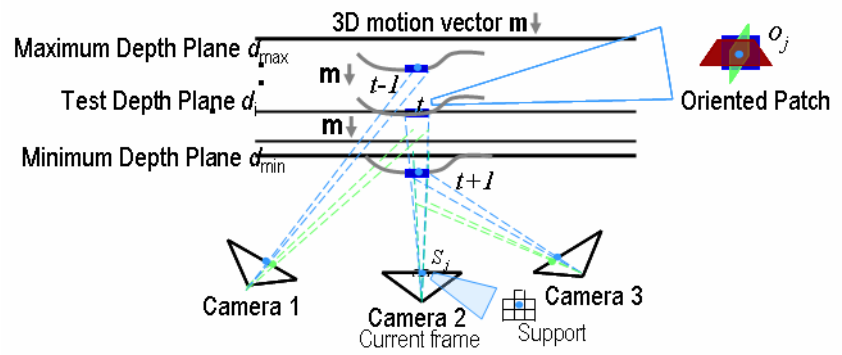


Figure 29.Extended plane-sweeping algorithm with orientation and motion hypotheses

The conditional likelihood $P_j(d \mid o, \mathbf{m})$ is determined by the color consistency measures $e_{diff}(S_j)$:

$$P_j(d \mid o, \mathbf{m}) = \frac{g(d, o, \mathbf{m})}{\sum_{d'} g(d', o, \mathbf{m})}, \text{ where } g(d, o, \mathbf{m}) = 1 - e_{diff}(S_j) \tag{14}$$

4.4 Patch-Based Smoothing for Dynamic Scene

We refine patch's initial geometry and motion distribution $P_j^0(d, o, \mathbf{m})$ between its neighboring patches and between its corresponding regions at multiple times/viewpoints iteratively, which is similar to [18], with the extension of smoothing for additional orientation and motion estimation. We enforce a smoothness constraint that the

neighboring patches (S_j and s_1 in Frame 1) with similar colors (blue) should have similar depths ($d \approx d_i$), orientation (o = vertical), and motion vectors (**m**) as shown in Figure 30. We also ensure scene's geometry and motion consistency constraint between images. We assume that scene patches follow a constant motion in a short time period. If we project a patch S_j with its ground-truth depth d, orientation o, and motion vector **m** into a neighboring image, the projected region (in Frame 2) should have the similar depth, orientation and motion if not occluded.

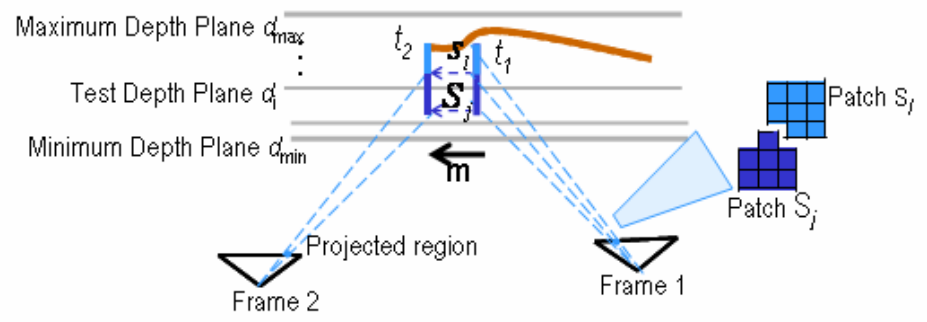


Figure 30. Patch-based smoothing for dynamic scenes

The likelihood distribution of patch's geometry and motion $P_j^t(d,o,\mathbf{m})$ is updated iteratively with three constraints.

$$P_j^{t+1}(d,o,\mathbf{m}) = \frac{n_j(d,o)n_j(\mathbf{m})\prod_{k\in N} c_{j,k}(d,o,\mathbf{m})}{\sum_{d',o',\mathbf{m}'} n_j(d',o')n_j(\mathbf{m}')\prod_{k\in N} c_{j,k}(d',o',\mathbf{m}')} \tag{15}$$

where $n_j(d,o)$ enforces patch's geometry smoothness constraint, and $n_j(\mathbf{m})$ enforces patch's motion smoothness constraint. $c_{j,k}(d,o,\mathbf{m})$ ensures patch's consistency constraint in each projected region at multiple times/viewpoints, $k = 1\ldots, N$. The details are given below.

4.4.1 Patch-Based Smoothness Constraint for Dynamic Scene

The smoothness constraints enforce that neighboring patches with similar colors should have similar depths, orientation and motion. We estimate $n_j(\mathbf{m})$ and $n_j(d,o)$ using a similar approach to estimate $n_j(d,o)$ introduced in Section 3.5.1. Let s_l denote one of patch S_j's neighboring patches in Figure 30. $\hat{d}_l$, $\hat{o}_l$ and $\hat{\mathbf{m}}_l$ are the maximum likelihood estimates of its depth, orientation, and motion based on $P_l'(d,o,\mathbf{m})$, respectively.

$$\left\langle \hat{d}_l \quad \hat{o}_l \quad \hat{\mathbf{m}}_l \right\rangle = \arg\max_{d,o,\mathbf{m}} P_l'(d,o,\mathbf{m}) \tag{16}$$

We define $n_j(d,o)$ the same approach as in Section 3.5.1, but with a new variance $\sigma_l'^2$ as follows:

$$n_j(d,o) = \begin{cases} \prod_{s_l} N(d;\hat{d}_l,\sigma_l'^2) + \varepsilon & o = \hat{o}_l \\ c + \varepsilon & o \neq \hat{o}_l \end{cases}, \tag{17}$$

where $\sigma_l'^2 = \dfrac{v}{P_l(\hat{d}_l,\hat{o}_l,\hat{\mathbf{m}}_l)^2 b_{j,l} N(\Delta_{j,l};0,\sigma_\Delta^2)}$, and $P_l(\hat{d}_l,\hat{o}_l,\hat{\mathbf{m}}_l)$ is the geometry and motion maximum likelihood for patch s_l.

The motion smoothness coefficient $n_j(\mathbf{m})$ enforces that the neighboring patches with similar colors should have similar motion. We assume that patch S_j's motion vector $\mathbf{m}$ is also modeled by a contaminated Gaussian distribution with mean $\hat{\mathbf{m}}_l$ and variance Σ_l. Therefore, we define $n_j(\mathbf{m})$ as follows:

$$n_j(\mathbf{m}) = \prod_{s_l} N(\mathbf{m};\hat{\mathbf{m}}_l,\Sigma_l) + \varepsilon, \text{ where } \Sigma_l = \sigma_l'^2 I_{2\times 2}. \tag{18}$$

If patch S_j and its neighboring patch s_l have similar colors, and patch S_j's motion is consistent with its neighbor's motion maximum likelihood estimate, we expect $n_j(\mathbf{m})$ to be large.

4.4.2 Patch-Based Consistency Constraint for Dynamic Scene

The spatial consistency coefficient $c_{j,k}(d,o,\mathbf{m})$ ensures that the patch S_j's depth, orientation and motion estimates are consistent with the depth, orientation and motion estimates at time/viewpoint k. We compute $c_{j,k}(d,o,\mathbf{m})$ based on spatial consistency, visibility, and patch S_j's initial geometry and motion likelihood, which contains the additional motion consideration compared to the ones discussed in 3.5.2:

1. Spatial consistency without occlusion. We first project patch S_j with the depth d, orientation o and motion vector $\mathbf{m}$ onto a neighboring image. We then calculate patch S_j's projecting distribution $b^t_{j,k}(d,o,\mathbf{m})$ based on the geometry and motion distribution at the projected time/viewpoint k to estimate the spatial consistency without occlusion.

$$b^t_{j,k}(d,o,\mathbf{m}) = \frac{1}{num_{S_j}} \sum_{x \in S_j} P^t_{r(k,x)}(d,o,\mathbf{m}) \tag{19}$$

where $r(k,x)$ is the patch index at the time/viewpoint k, on which the corresponding pixel of the pixel position x on patch S_j is. And num_{S_j} is the number of the pixels on patch S_j. If the projected region's depth, orientation and motion maximum likelihood estimates are consistent with patch S_j's estimates, we expect $b^t_{j,k}(d,o,\mathbf{m})$ to be large when patch S_j is visible at the time/viewpoint k (Frame 2) as shown in Figure 30.

2. Visibility. Due to the possible occlusions, a patch might not have the corresponding pixels at another time/viewpoint as shown in Figure 31. We estimate the overall visibility likelihood $v_{j,k}$, that the patch is visible, as follows:

$$v_{j,k} = \min\left(1.0, \sum_{d',o',\mathbf{m}'} b^t_{j,k}(d',o',\mathbf{m}')\right) \tag{20}$$

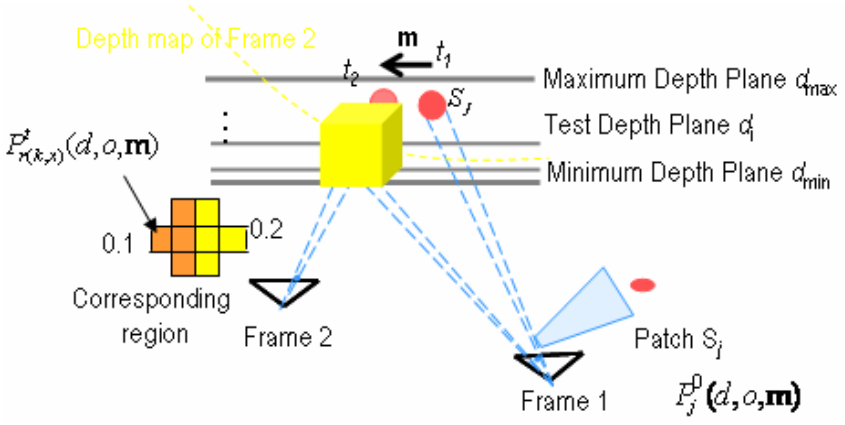


Figure 31. Red patch is occluded in Frame 2 of a dynamic scene.

If the patch S_j is visible at the time/viewpoint k (Frame 2), we can find its corresponding region when we search the space of the depth d, orientation o, and motion $\mathbf{m}$ as shown in Figure 30. The ground-truth solution and its neighboring solutions offer large $b^t_{j,k}(d',o',\mathbf{m}')$ values. If the patch S_j is occluded at the time/viewpoint k (Frame 2), we can not find its corresponding region when we search the space of depth d, orientation o, and motion $\mathbf{m}$ as shown in Figure 31. No solution provides large $b^t_{j,k}(d',o',\mathbf{m}')$ value. Therefore, we use $v_{j,k}$ as a robust and computational-efficient measure of patch's visibility.

We also estimate the specific visible likelihood $vc_{j,k}(d,o,\mathbf{m})$ that given the depth d, orientation o and motion $\mathbf{m}$, patch S_j is visible at the time/viewpoint k as follows:

$$vc_{j,k}(d,o,\mathbf{m}) = \frac{1}{num_{S_j}} \sum_{x \in S_j} P^t_{r(k,x)}(d,o,\mathbf{m}) h(\hat{d}_{r(k,x)} - d^+), \quad (21)$$

where $h(x)$ is the Heaviside step function. $\hat{d}_{r(k,x)}$ is the maximum likelihood depth estimate of patch $s_{r(k,x)}$, and d^+ is the depth of the patch S_j at the time/viewpoint k with motion shift.

This suggests that if patch S_j is visible at the time/viewpoint k, its estimated depth should not be under the surface of the depth map estimated at the time/viewpoint k as shown in Figure 31.

3. Initial patch S_j's geometry and motion likelihood $P^0_j(d,o,\mathbf{m})$. It is estimated in Section 4.3.

Now, we combine the visible and occluded cases. If the patch is visible, $c_{j,k}(d,o,\mathbf{m})$ is calculated from the visible consistency likelihood $b^t_{j,k}(d,o,\mathbf{m})P^0_j(d,o,\mathbf{m})$. Otherwise, its occluded consistency likelihood is $(1-vc_{j,k}(d,o,\mathbf{m}))P^0$, with the uniform prior $P^0 = \frac{1}{size(d)size(o)size(\mathbf{m})}$. $size(d)$, $size(o)$, and $size(\mathbf{m})$ are the sizes of depth, orientation and motion hypothesis spaces, respectively. Therefore,

$$c_{j,k}(d,o,\mathbf{m}) = v_{j,k} b^t_{j,k}(d,o,\mathbf{m}) P^0_j(d,o,\mathbf{m}) + (1-v_{j,k})(1-vc_{j,k}(d,o,\mathbf{m}))P^0. \quad (22)$$

4.5 Pixel-Based Geometry and Motion Smoothing for Dynamic Scene

The maximum likelihood estimates of each patch's depth $\hat{d}$, motion $\hat{\mathbf{m}}$ and orientation $\hat{o}$ based on $P_j^t(d,o,\mathbf{m})$ determine the initial depth map $d^0(x)$, the initial motion map $\mathbf{m}^0(x)$, and the orientation map for each pixel position *x*.

$$\left\langle \hat{d}_l \quad \hat{o}_l \quad \hat{\mathbf{m}}_l \right\rangle = \arg\max_{d,o,\mathbf{m}} P_l^t(d,o,\mathbf{m}) \tag{23}$$

We further refine the depth map $d^t(x)$ and motion map $\mathbf{m}^t(x)$ iteratively between images, which is an extension of [18] with motion smoothing. For each pixel *x* at the current time/viewpoint, we find its corresponding pixel *y* at the neighboring time/viewpoint *k* with the constant motion assumption. If the corresponding pixel's depth $d_k^t(y)$ is similar to pixel *x*'s depth $d^t(x)$, the pixel *x*'s depth $d^{t+1}(x)$ is replaced by the average of $d^t(x)$ and $d_k^t(y)$. If the corresponding pixel's motion $\mathbf{m}_k^t(y)$ is similar to pixel *x*'s motion $\mathbf{m}^t(x)$, the pixel *x*'s motion $\mathbf{m}^{t+1}(x)$ is replaced by the average of $\mathbf{m}^t(x)$ and $\mathbf{m}_k^t(y)$. The iterative updating equations are

$$d^{t+1}(x) = \frac{1}{N}\sum_k \left(\mu_k \frac{d^t(x) + d_k^t(y)}{2} + (1-\mu_k) d^t(x) \right) \tag{24}$$

$$\mathbf{m}^{t+1}(x) = \frac{1}{N}\sum_k \left(\mu_k \frac{\mathbf{m}^t(x) + \mathbf{m}_k^t(y)}{2} + (1-\mu_k) \mathbf{m}^t(x) \right) \tag{25}$$

where $\mu_k = (\delta_{d,k} \; and \; \delta_{d,k})$, and $\delta_{d,k} = | d^t(x) - d_k^t(y) | < \lambda_d$, $\delta_{\mathbf{m},k} = | \mathbf{m}^t(x) - \mathbf{m}_k^t(y) | < \lambda_{\mathbf{m}}$. δ is the indicator variable (0,1) testing input similarity with the threshold parameter λ, and *N* is the number of neighboring images. After smoothing the depth map across different times/viewpoints, we apply a 3x3 average filter to further smooth the results.

4.6 Experimental Results

We first showed the experimental results of the prior-based motion direction estimation based on a single image.

We trained the SVM-based motion direction estimator with 12249 labeled patches, extracted from 79 color images with the resolution of 320x240 pixel^2. The images for training the motion direction estimator were shown in Figure 32.

Figure 32. Images for training the motion direction estimator

We inferred the motion direction distribution of each image patch using the motion direction estimator. In Figure 33, we showed the classification results of the motion direction estimator on a sample image with the maximum likelihood estimates represented by the shaded colors: red (forward), green (backward), yellow (left/right) and

blue (stationary). It provided the classification accuracy: 88% of the patches correctly labeled.

(a) Sample image (b) Classification results

Figure 33. Prior-based motion direction estimation results

Next, we showed the experimental results of the prior-based geometry and motion reconstruction of dynamic scenes.

We ran experiments on multiple synthetic images of a dynamic street simulated by POV-ray [61]. As illustrated in Figure 34, six images were captured by a forward moving camera with the known intrinsic and extrinsic camera calibration parameters as the input image sequence. Only the yellow school bus in the scene was moving towards right.

Figure 34. Input images of a dynamic street

Each image was segmented into small patches, and patch's orientation and motion direction distributions were inferred based on patch's appearance and location. We applied the prior-based geometry and motion reconstruction algorithm on these input images to reconstruct depth map and motion map at each time instance as illustrated in Figure 3.

We compared the results of the proposed prior-based algorithm with the estimated prior distributions ($P_j(o)$ and $P_j(m_{dir})$) and the results of previous work without using any prior, which is effectively our algorithm with uniform prior distributions ($P_j(o) = c_o$ and $P_j(m_{dir}) = c_m$) for geometry and motion reconstruction.

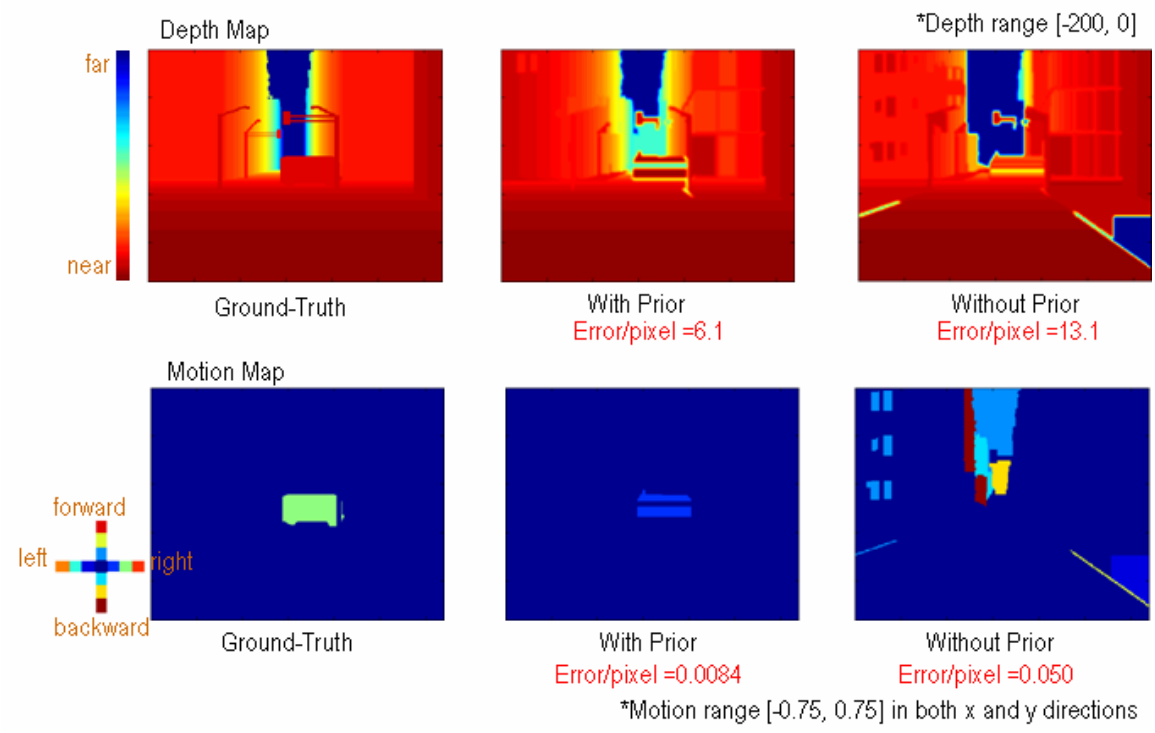


Figure 35. Depth map and motion map comparison of different algorithms for dynamic scene reconstruction.

We compared their resulting depth maps and motion maps at the first frame's viewpoint as shown in Figure 35. The dynamic range of depth map is assumed to be 200. The dynamic range of motion map is assumed to be 1.5. Compared with the ground-truth depth map, the prior-based method provided the reconstructed depth map with 6.1/200=3.1% error per pixel on average, while the approach without prior offered 13.1/200=6.6% reconstruction error per pixel. Compared with the ground-truth motion

map, the prior-based method provided the reconstructed motion map with 0.0084/1.5=0.56% error per pixel on average, while the approach without prior offered 0.050/1.5=3.3% reconstruction error per pixel. Therefore, the prior-based method provided better depth map and motion map than the traditional approach without prior knowledge, especially in the textureless (e.g. ground) areas.

We also showed the experimental results of the prior-based geometry and motion reconstruction of an indoor lab scene. A forward moving camera captured 7 input images as shown in Figure 36. Only the grey vehicle in the scene was moving towards left.

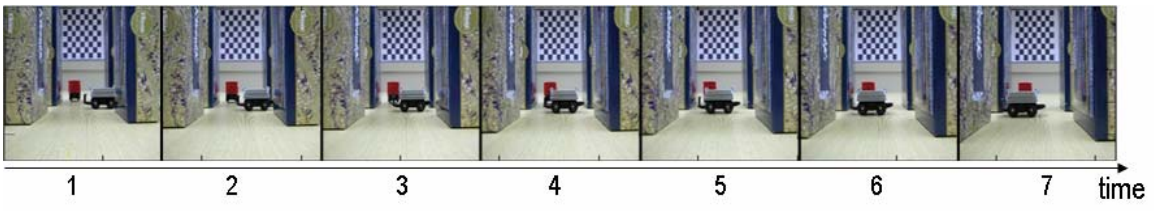


Figure 36. Input images of a dynamic lab scene

We calibrated the camera's intrinsic parameters (camera's focal length and optical center) with checker board patterns offline and the extrinsic parameters (the translation vector and the rotation matrix) with markers on the ground using Zhang's method [62]. We applied the prior-based geometry and motion reconstruction algorithm on these input images to reconstruct depth map, motion map and orientation map at each time instance.

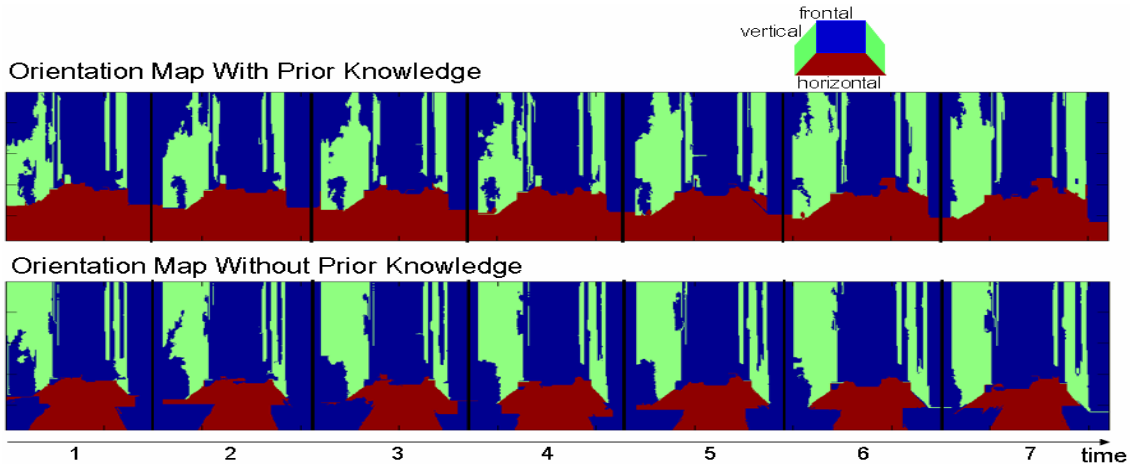


Figure 37. Orientation map comparison on a lab scene

We compared the results of the proposed prior-based algorithm with the estimated prior distributions ($P_j(o)$ and $P_j(m_{dir})$) and the results of previous work with uniform prior distributions ($P_j(o) = c_o$ and $P_j(m_{dir}) = c_m$) for geometry and motion reconstruction. The prior-based method provided better orientation maps than the traditional approach without prior knowledge, especially in the ground areas in Figure 37.

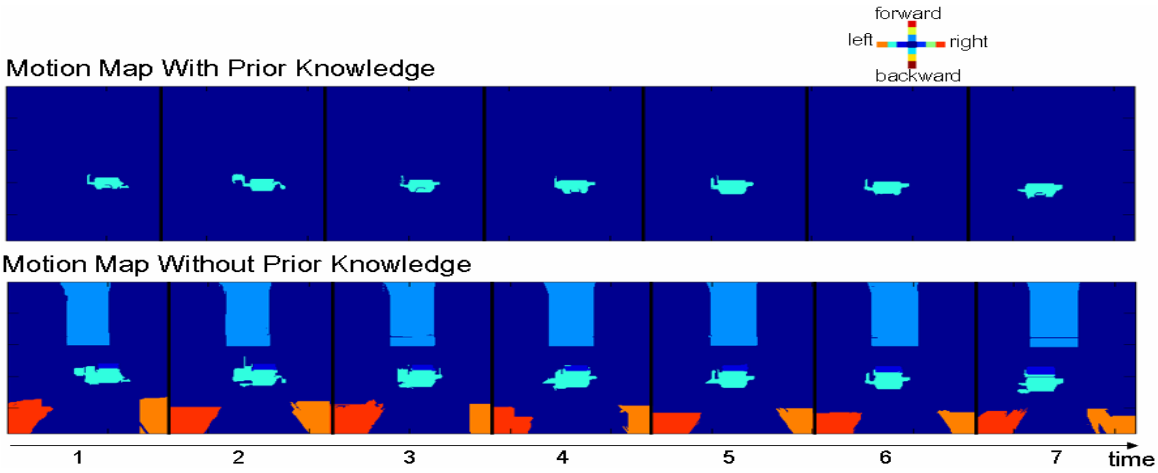


Figure 38. Motion map comparison on a lab scene

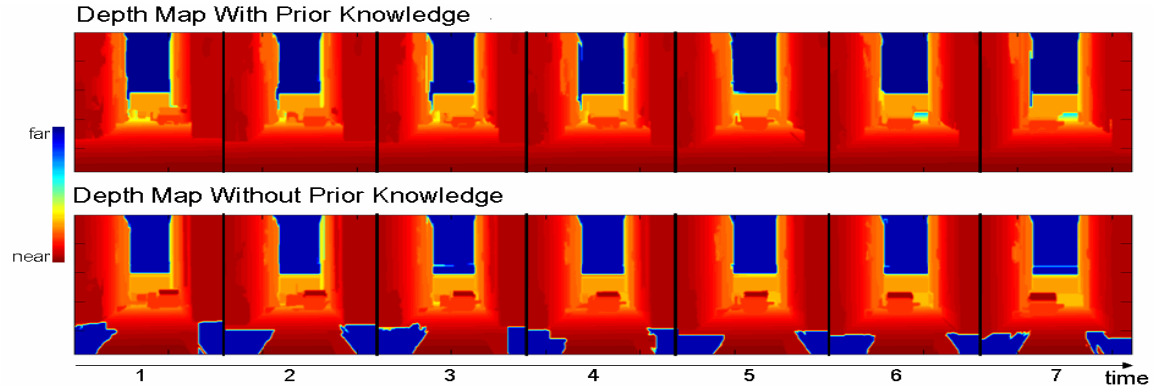


Figure 39. Depth map comparison on a lab scene

In Figure 38, we showed the comparison of resulting motion maps. The prior-based method also offered a better motion vector estimation, since it identified the grey vehicle's correct motion (median speed towards left). The method without prior knowledge provided wrong motion at the ground and faraway background areas, since the ground was textureless, and it was difficult to tell whether the faraway scene was stationary or moving slightly without any prior information. Therefore, the prior-based method had better reconstructed depth maps than the traditional approach in Figure 39.

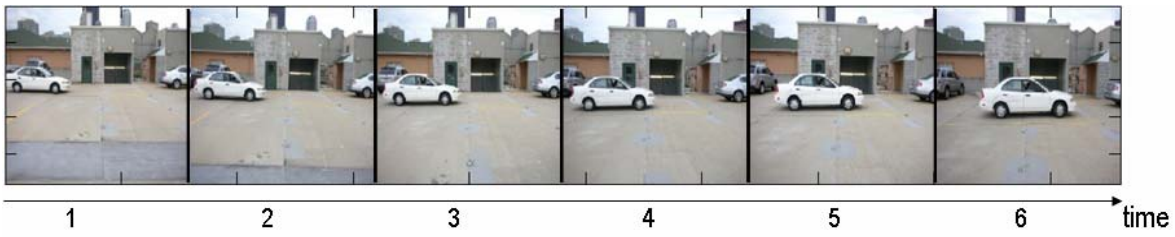


Figure 40. Input images of a parking lot

Finally, we showed experimental results of the prior-based geometry and motion reconstruction of a parking lot scene. A forward moving camera captured 6 images in an outdoor parking lot in Pittsburgh, PA with a white car as shown in Figure 40. Only the white car was moving towards right. We compared the results of our proposed prior-based algorithm with the estimated prior distributions ($P_j(o)$ and $P_j(m_{dir})$) and results

with uniform prior distributions ($P_j(o) = c_o$ and $P_j(m_{dir}) = c_m$) for geometry and motion reconstruction. The prior-based method provided better orientation maps than the traditional approach, especially in the ground (textureless) areas. However, it failed to identify all the vertical structures as shown in Figure 41.

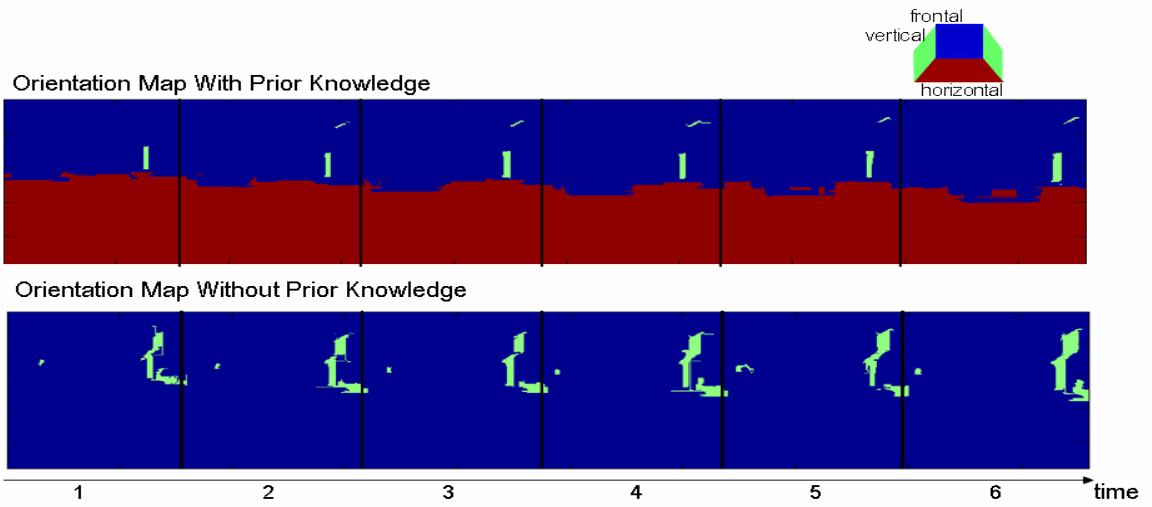


Figure 41. Orientation map comparison on a parking lot scene

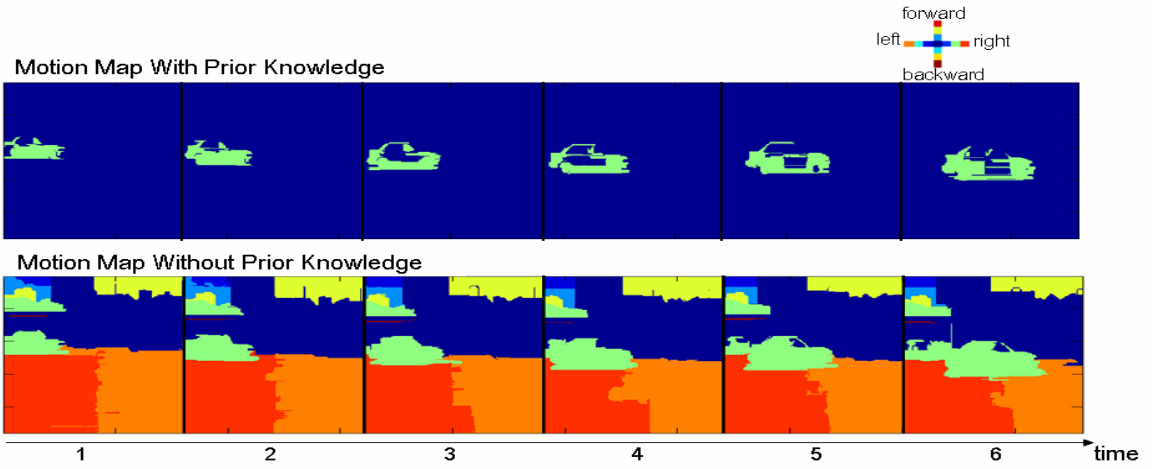


Figure 42. Motion map comparison on a parking lot scene

In Figure 42, we showed the comparison of resulting motion maps. The prior-based method also offered better motion maps, since it recovered the white car's correct motion (median speed towards right). The method without prior knowledge indicated wrong

motion at the ground and sky areas, since the ground was textureless, and it was difficult to tell whether the motion of the sky without any prior information. Therefore, the prior-based method had better estimated depth maps than the traditional approach in Figure 43.

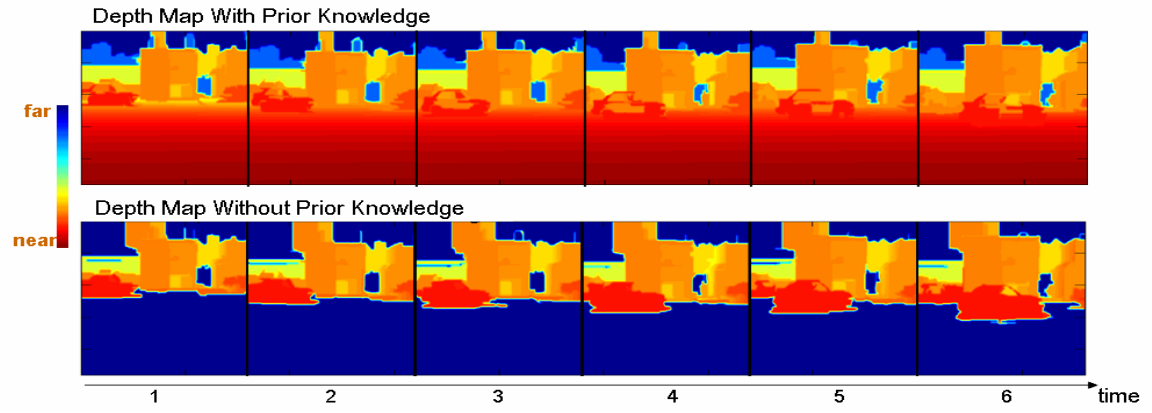


Figure 43. Depth map comparison on a parking lot scene

To have a better visualization of our results, we rendered scene at the time *1* at multiple viewpoints as shown in Figure 44.

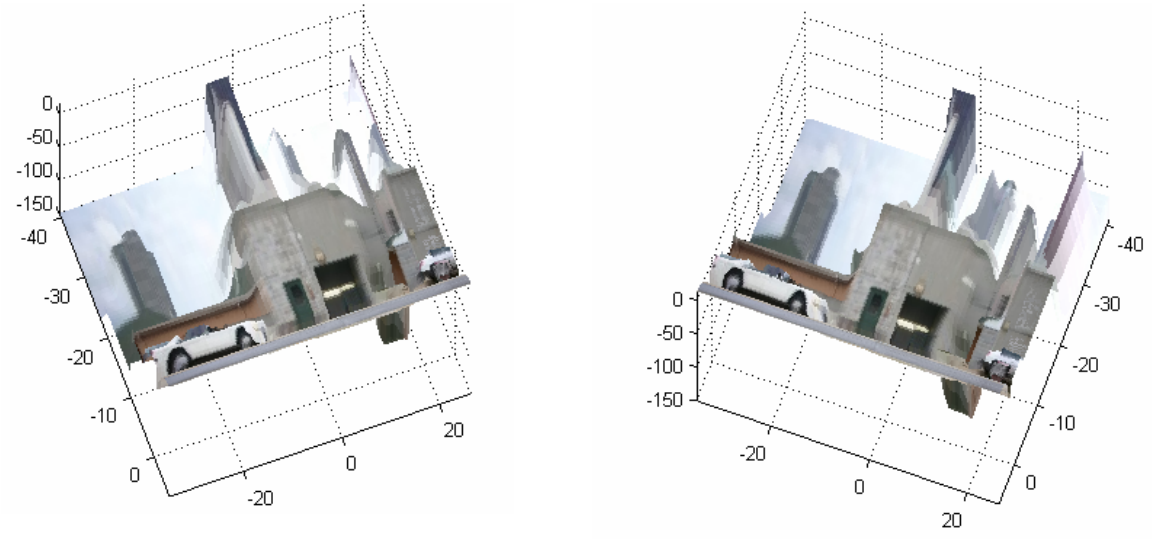


Figure 44. Rendering scene at multiple viewpoints

5. Conclusions and Future Work

In this thesis, we proposed a probabilistic framework for reconstructing scene geometry and object motion utilizing prior knowledge of a class of scenes, for example, scenes captured by a calibrated camera mounted on a vehicle driving through city streets. Traditional approaches try to match the points, lines or patches in multiple images to reconstruct scene geometry and object motion. Our framework also takes advantage of each image patch's appearance and location to infer its orientation and motion direction using statistical learning.

We proposed the prior-based geometry reconstruction method for stationary scene and the prior-based geometry and motion reconstruction method for dynamic scene. We showed that the prior-based reconstruction methods outperformed traditional reconstruction methods with synthetic data and real data for both stationary scenes as shown in Table 2 and dynamic scenes as shown in Table 3, especially in the textureless areas for geometry estimation and faraway areas for motion estimation.

In future, we will further improve the learning performance of prior knowledge by having more training samples and better learning algorithms. We will also include rotation into the motion modeling and estimation.

Table 2. The performance comparison of algorithms for stationary scene reconstruction

Synthetic Data	With Prior	Without Prior (Oriented plane sweeping)	Without Prior (Smoothed oriented plane sweeping)
Depth Map Reconstruction Error	3.5%	7.2%	4.1%

Table 3. The performance comparison of algorithms for dynamic scene reconstruction

Synthetic Data	With Prior	Without Prior
Depth Map Reconstruction Error	3.1%	6.6%
Motion Map Reconstruction Error	0.56%	3.3%

Appendix A. Video-Based Rendering Using Feature Point Evolution

A.1 Previous Work

In this section, we provide an overview of some previous work of the feature point evolution method. We first present the time-series analysis techniques: Kalman filter and particle filter, and then introduce the perspective camera model.

Kalman Filter

Kalman filtering [40] is proposed by R.E. Kalman describing a recursive solution to the linear filtering problem. A discrete-time process $\{\mathbf{x}_k\}$ is governed by a linear stochastic difference equation as the *state* equation, while the observation $\mathbf{y}_k$ is connected with the state $\mathbf{x}_k$ by a linear *measurement* equation.

State:
$$\mathbf{x}_{k+1} = F_k\mathbf{x}_k + \mathbf{u}_k + \mathbf{q}_k \tag{26}$$

Measurement:
$$\mathbf{y}_k = H_k\mathbf{x}_k + \mathbf{r}_k \tag{27}$$

where $\mathbf{q}_k \sim N(0,Q_k)$, and $\mathbf{r}_k \sim N(0,R_k)$. $\mathbf{u}_k$ is an offset term to the state $\mathbf{x}_k$.

The modeling errors $\mathbf{q}_k$ in the state process and $\mathbf{r}_k$ in the measurement process are assumed to be the Gaussian noises with zero mean. The initial state $\mathbf{x}_0$ also follows the Gaussian distribution with the mean $\bar{\mathbf{x}}_0$ and the variance P_0. $\{\mathbf{q}_k\}$, $\{\mathbf{r}_k\}$ and $\mathbf{x}_0$ are assumed to be statistically independent of each other.

With the initial condition, the estimation of $\mathbf{x}_k$ can be summarized by the following *prediction* and *correction* iteratively based on the observation $\mathbf{y}_k$. A detail discussion can be found in [40].

Initialization:

$$\mathbf{x}_0 \sim N(\bar{\mathbf{x}}_0, P_0), \tag{28}$$

Prediction:

$$\hat{\mathbf{x}}_{k+1}^- = F_k \hat{\mathbf{x}}_k + \mathbf{u}_k \tag{29}$$

$$P_{k+1}^- = F_k P_k F_k^T + Q_k \tag{30}$$

Correction:

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_{k+1}^- + K_{k+1}\mathbf{e}_{k+1}, \text{ where the estimation error } \mathbf{e}_{k+1} = (\mathbf{y}_{k+1} - H_{k+1}\hat{\mathbf{x}}_{k+1}^-). \tag{31}$$

$$K_{k+1} = P_{k+1}^- H_{k+1}^T (H_{k+1} P_{k+1}^- H_{k+1}^T + R_{k+1})^{-1} \tag{32}$$

$$P_{k+1} = (I - K_{k+1} H_{k+1}) P_{k+1}^- \tag{33}$$

where $\hat{\mathbf{x}}_{k+1}^-$ is the prediction of $\mathbf{x}_{k+1}$, given $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k$, and $\hat{\mathbf{x}}_{k+1}$ is the prediction of $\mathbf{x}_{k+1}$, given $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{k+1}$.

If the *measurement* equation (27) is non-linear, we can linearize it using the Taylor expansion to satisfy the linear measurement assumption of the Kalman filter as follows:

$$\mathbf{y}_k = f_k(\mathbf{x}_k) + \mathbf{r}_k = f_k(\hat{\mathbf{x}}_k^-) + H_k(\hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-) + \tilde{\mathbf{r}}_k \tag{34}$$

where H_k is the Jacobian matrix of the partial derivatives of the function f_k with respect to $\mathbf{x}_k$. The random variable $\tilde{\mathbf{r}}_k$ is the linearized measurement error, which is the sum of the measurement error $\mathbf{r}_k$ and the linearization error of f_k. Now we can apply the Kalman filtering approach to estimate $\mathbf{x}_k$, as long as $\tilde{\mathbf{r}}_k$ is assumed to have the Gaussian distribution with zero mean. The estimation process is similar to the estimation process of the Kalman filtering, but $\mathbf{e}_{k+1} = \mathbf{y}_{k+1} - f_{k+1}(\hat{\mathbf{x}}_{k+1}^-)$. Such a linearization approach is a special case of the extended Kalman filtering approach [41].

Particle Filter

If the modeling errors $\mathbf{q}_k$ in the state equation (35) and $\mathbf{r}_k$ in the measurement equation (36) are non-additive or non-Gaussian in the dynamic system, the sequential Monte Carlo approaches are proposed to analysis and make inference about the dynamic system.

$$\textit{State:} \qquad \mathbf{x}_{k+1} = g_k(\mathbf{x}_k, \mathbf{q}_k) \qquad \leftrightarrow \qquad p(\mathbf{x}_{k+1} \mid \mathbf{x}_k) \tag{35}$$

$$\textit{Measurement:} \qquad \mathbf{y}_k = f_k(\mathbf{x}_k, \mathbf{r}_k) \qquad \leftrightarrow \qquad p(\mathbf{y}_k \mid \mathbf{x}_k) \tag{36}$$

where $\mathbf{q}_k$ and $\mathbf{r}_k$ follow their distribution $p(\mathbf{q}_k)$ and $p(\mathbf{r}_k)$ respectively. $\{\mathbf{q}_k\}$, $\{\mathbf{r}_k\}$ and initial condition $\mathbf{x}_0$ (with its distribution $p(\mathbf{x}_0)$) are assumed to be statistically independent of each other.

The sequential importance sampling algorithm [64] is the basis for most sequential Monte Carlo filters. It is also known as particle filtering [65], condensation algorithm [66], and survival of the fittest [67]. The main idea of particle filtering is to represent the posterior distribution of the states $p(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k})$ by a set of random samples $\mathbf{x}_{0:k}^i$ with their corresponding weights w_k^i as follows.

$$p(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k}) \approx \sum_i w_k^i \delta(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^i) \tag{37}$$

where $\mathbf{x}_{0:k} = \{\mathbf{x}_j, j = 0,1,...,k\}$ is the set of all states, and $\mathbf{y}_{1:k} = \{\mathbf{y}_j, j = 1,2,...,k\}$ is the set of all observations up to time *k*. $\delta(\cdot)$ is the Dirac delta function. The sum of the weights is normalized to be 1. As the number of samples increases, the estimated posterior distribution represented by the samples will converge to its functional description. The posterior density distribution $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ can be approximated as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \approx \sum_i w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i), \tag{38}$$

$$\text{and the estimated } \hat{\mathbf{x}}_k = \sum_i w_k^i \mathbf{x}_k^i \approx E[\mathbf{x}_k] \tag{39}$$

If the samples $\mathbf{x}_{0:k}^i$ were drawn from an importance density $q(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k})$, the weights could be derived as follows:

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{y}_i \mid \mathbf{x}_k^i) p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i, \mathbf{y}_k)} \tag{40}$$

under the assumption that $q(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) = q(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{y}_k)$. The choice of importance density function $q(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{y}_k)$ is a critical design issue for successful particle filter. One possible choice of importance density function $q(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{y}_k)$ is the distribution $p(\mathbf{x}_k^i \mid \mathbf{x}_{k-1}^i)$. The equation (40) turns out to be $w_k^i \propto w_{k-1}^i p(\mathbf{y}_i \mid \mathbf{x}_k^i)$.

With its initial condition, the estimation of $\mathbf{x}_k$ can be summarized by the following *importance sampling* and *re-sampling* procedures iteratively based on the observation $\mathbf{y}_k$. A detail discussion can be found in [42].

Initialization:

For *i*=1,…,*N*, draw $\mathbf{x}_0^i$ from $p(\mathbf{x}_0)$, and set $w_0^i = 1/N$.

Importance sampling:

For *i*=1,…,*N*, draw $\mathbf{x}_k^i$ from $q(\mathbf{x}_k \mid \mathbf{x}_{k-1}^i, \mathbf{y}_k)$ as *prediction*, and assign the particle weight according to (40) as *correction*.

Normalize the weights so that $\sum_i w_k^i = 1$.

Re-sampling:

Generate a new set $\{\mathbf{x}_k^{i*}\}_{i=1}^N$ from an approximate discrete representation of $p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \approx \sum_i w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i)$ with the new weights $w_k^{i*} = 1/N$. The purpose of the re-sampling is to eliminate the particles with small weights and to concentrate on the particles with large weights.

Compared with the Kalman filtering, particle filtering has the advantage to deal with the dynamic system with the nonlinear or non-Gaussian noises in the modeling. Meanwhile, we have to estimate the noise distribution with high order statistics. Therefore, we might have less reliable estimates, since the estimated noise distribution is less reliable due to its complexity in the particle filter modeling.

Our proposed feature evolution process can be considered in high level as an extension to particle filtering for scene modeling as shown in Table 4. The main idea of the feature point evolution process is to represent the geometry and motion of the scene by a set of feature points with their corresponding positions and motions. While the feature points still follow the state equation (35) and measurement equation (36) in the dynamic system, we have *n* measurements $\mathbf{y}_{k,n}$ at each time instance.

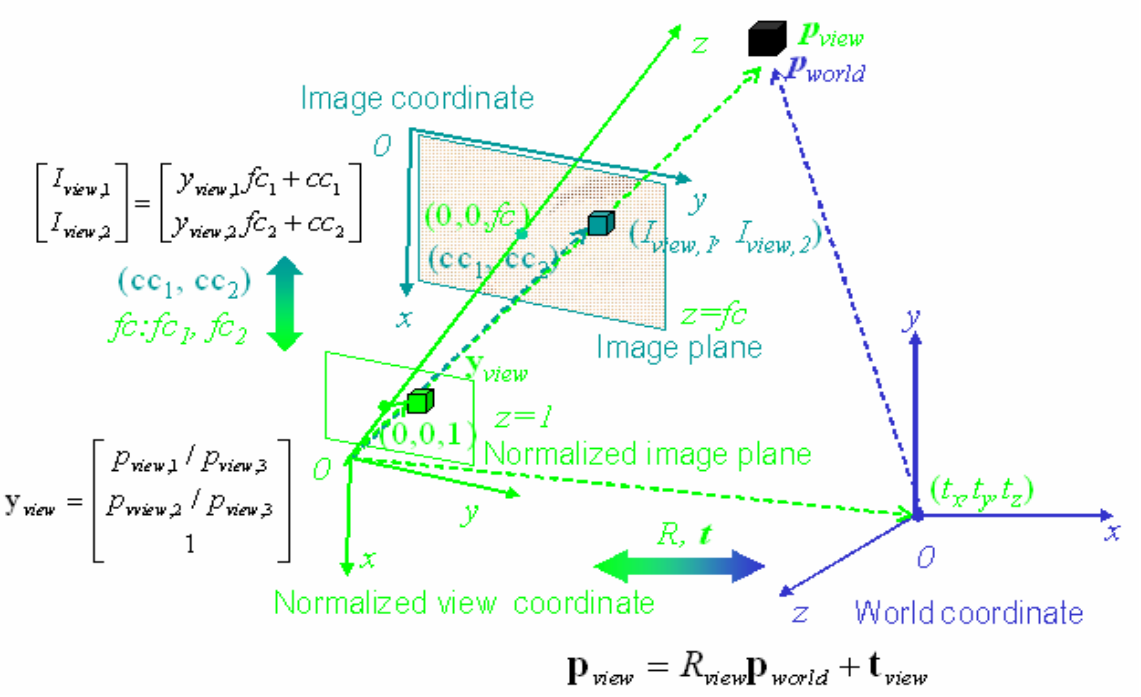
Table 4. Problem formulation comparison

	Particle filter	Feature point evolution
Representation	State's posterior pdf is described by a set of samples and their corresponding weights (scalar)	The scene geometry and motion is described by a set of feature points and their corresponding positions and motions (vector)
Number of measurements	1 at each time instance	*n* at each time instance
Correspondence	Given in measurement	Not given in measurement

Since the correspondence between $\mathbf{y}_{k,n}$ and $\mathbf{y}_{k-1,n}$ is not given, sometimes even does not exist, we first need to identify the reliable feature points, which have high probability to find the correct correspondence in the neighboring frames. We then identify the correspondence between the reliable feature points as the measurements $\mathbf{y}_{k,m}$ and $\mathbf{y}_{k-1,m}$ ($m < n$) based on image texture. In contrast to using the deterministic sample values in particle filter, we model the feature points probabilistically in the feature point evolution.

Given the correspondence between the reliable feature points, we model each feature point using the Kalman filtering or particle filtering based on the observation $\mathbf{y}_{k,m}$. The detail feature point evolution process will be discussed later.

Perspective Camera Model



$$\mathbf{p}_{view} = R_{view}\mathbf{p}_{world} + \mathbf{t}_{view}$$

Figure 45. Perspective camera model

To understand the perspective camera model [68], we project a point in 3D space to an image plane using the perspective camera model. As shown in Figure 45, there are three coordinates in the perspective camera model: the *world coordinate* (3D), the *normalized view coordinate* (3D), and the *image coordinate* (2D). The origin and the *z* axis of the normalized view coordinate are the camera position and the camera viewing direction respectively.

First, the coordinate transformation equation (41) converts a point's representation $\mathbf{p}_{world}$ in the world coordinate to its representation $\mathbf{p}_{view}$ in the normalized view coordinate with the rotation matrix R_{view} and the translation vector $\mathbf{t}_{view}$.

$$\mathbf{p}_{view} = R_{view}\mathbf{p}_{world} + \mathbf{t}_{view} \tag{41}$$

Then, we project the point $\mathbf{p}_{view}$ onto the normalized image plane at the point $\mathbf{y}_{view}$, which is the plane of $z = 1$ in the normalized view coordinate as shown in Figure 45.

$$\mathbf{y}_{view} = \begin{bmatrix} p_{view,1} / p_{view,3} \\ p_{view,2} / p_{view,3} \\ 1 \end{bmatrix}, \tag{42}$$

Finally, since the image plane is at $z=fc$ (camera focal length) in the normalized view coordinate, we calculate the projected point ($I_{view,1}$ $I_{view,2}$) on the image plane with the given camera center (cc_1, cc_2). In fact, we use the horizontal focal length fc_1 and the vertical focal length fc_2 in the real applications to offset the aspect ratio distortion as follows.

$$\begin{bmatrix} I_{view,1} \\ I_{view,2} \end{bmatrix} = \begin{bmatrix} y_{view,1} fc_1 + cc_1 \\ y_{view,2} fc_2 + cc_2 \end{bmatrix} \tag{43}$$

A.2 Modeling Evolution Scenes

In this section, we first introduce our novel tracking algorithm for the video-based rendering, and then present the reconstruction and rendering algorithm.

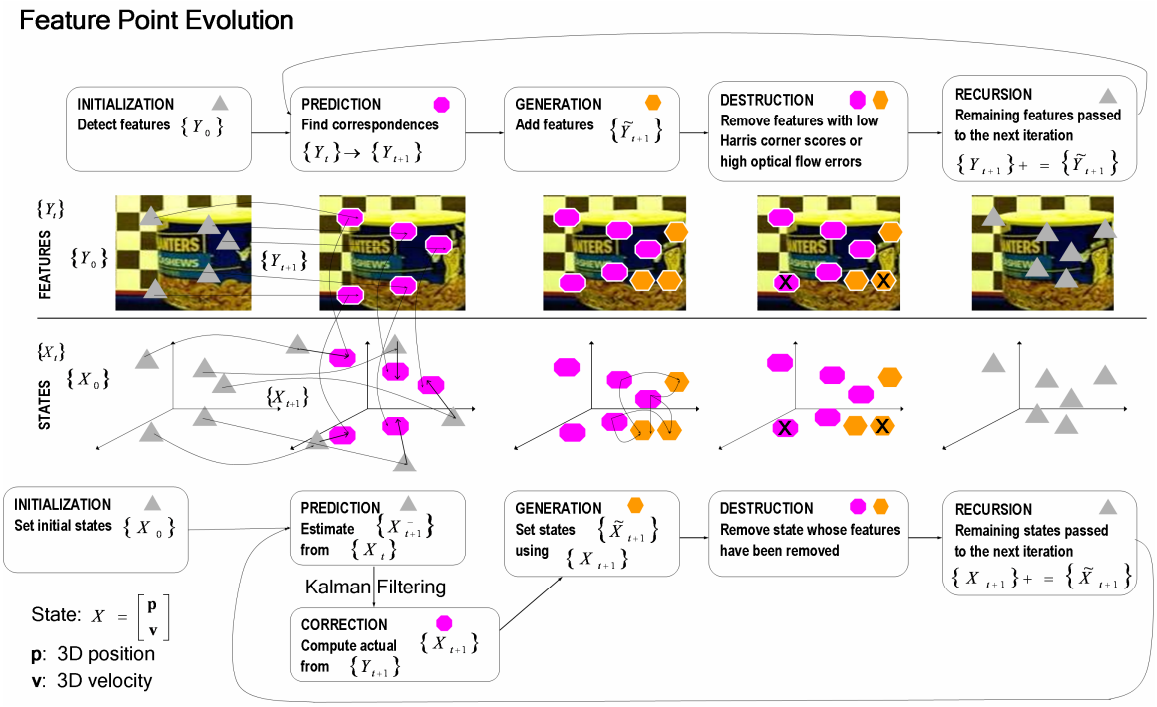


Figure 46. Evolution flow chart. The evolution of the features is modeled on top: images with sample feature points are marked. The respective evolution of the states is modeled on bottom: the estimated 3D positions and 3D motions of the sample feature points are plotted. INITIALIZATION, PREDICTION, and CORRECTION follow standard Kalman filtering while GENERATION and DESTRUCTION follow our proposed evolution framework.

Evolution, the core contribution of this work, is a dynamic feature point extractor embedded in standard time-series analysis (see Figure 46). As video progresses over time, certain tracked feature points (e.g., state X_t) will have noisy 2D image feature points Y_t that become difficult to track while, conversely, new feature points will appear that are robust, and easy-to-track. Hence, we only model each portion of the scene while it is easy to track. In addition to proposing which feature points (and associated states) to

model at each time frame, we also propose a novel "state passing" mechanism that initializes the states of the newly generated feature points in each frame. Evolution proceeds as follows, where Steps 1, 2, & 3 below correspond to the Extended Kalman filter (EKF)'s initialization, prediction, and correction and where Steps 4 & 5 below are the key contributions of this work:

We first introduce the EKF modeling used in evolution. Let $\{X_t\}$ be the set of the states, which are 3D positions $\mathbf{p}_t$ and 3D motions $\mathbf{v}_t$, and 2D image feature point set $\{Y_t\}$ be the observations. By assuming a constant velocity model for each feature point, we have

State equation: $$X_{t+1} = F_t X_t + Q_t, \quad Q_t \sim N(0, q_t) \tag{44}$$

Observation equation: $$Y_t = f_t(X_t) + R_t, \quad R_t \sim N(0, r_t) \tag{45}$$

where $X_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{v}_t \end{bmatrix}$, $F_t = \begin{bmatrix} I & I \\ 0 & I \end{bmatrix}$, and f_t projects the feature points' position $\mathbf{p}_t$ to the current image plane with known intrinsic and extrinsic camera calibration parameters. Q_t and R_t represent the Gaussian noises in the modeling with variance q_t and r_t, respectively. In the current implementation, the noise modeling q_t and r_t are time-independent.

1. Initialization (time $t = 0$ only)

- Find 2D feature point set $\{Y_0\}$ at time $t = 0$, using a Harris corner detector [69].
- For each $y_0 \in Y_0$, initialize its state $x_0 \in \{X_0\}$:

$$x_0 = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{v}_0 \end{bmatrix} \tag{46}$$

  where 3D position $\mathbf{p}_0$ is calculated by assuming a constant depth, and 3D motion $\mathbf{v}_0$ is set to zero.

2. Prediction

- For each $x_t \in X_t$, predict x_{t+1}^- using the EKF modeling above.
- For each $y_t \in Y_t$, find its corresponding position y_{t+1} in frame t+1 using the pyramid KLT tracker [70].

3. Correction

- For the predicted state $x_{t+1}^- \in X_{t+1}$, correct its value x_{t+1} using the EKF modeling.

4. Generation

- As in Step 1, find feature point set $\{\tilde{Y}_{t+1}\}$ using a Harris corner detector. These feature points are chosen independently of the predicted $\{Y_{t+1}\}$.
- For each of the new feature points $\tilde{y}_{t+1} \in \{\tilde{Y}_{t+1}\}$, find its corresponding position $\tilde{y}_t$ in frame t using (reverse) pyramid KLT tracker. Let $\{\tilde{X}_{t+1}\}$ be their (un-initialized) states.
- Initialize each new state $\tilde{x}_{t+1} \in \{\tilde{X}_{t+1}\}$. Let $\tilde{x}_{t+1}$ be initialized using weights $w_{x'_{t+1}}$ on the nearby existing states $X'_{t+1} = \{x_{t+1} \in X_{t+1} \,|\, \|y_{t+1} - \tilde{y}_{t+1}\| < Th_y\}$:

$$w_{x'_{t+1}} = \frac{\mathrm{Age}(x'_{t+1})}{\|y'_{t+1} - \tilde{y}_{t+1}\| \sum_{i=t+1-\mathrm{Age}(x'_{t+1})}^{t+1} |E_i(x'_i)|^2} \tag{47}$$

$$\tilde{x}_{t+1} = \frac{\sum_{x'_{t+1} \in X'_{t+1}} w_{x'_{t+1}} x'_{t+1} + \beta X_0}{\sum_{x'_{t+1} \in X'_{t+1}} w_{x'_{t+1}}} \tag{48}$$

where $\tilde{y}_t$, y_t, and y'_t are the observations of $\tilde{x}_t$, x_t, and x'_t, respectively; Th_y is the threshold for defining a new state's neighbors in the 2D image; β is the weight on the prior state X_0; $E_t(x'_t) = |y'_t - f_t(x'^-_t)|$ is the Kalman error of the state x'_t at time t, and $\mathrm{Age}(x'_{t+1})$ is the number of frames that state x'_{t+1} has been in existence.

5. Destruction

- Define $P_t(y'_t)$ as the square patch centered at pixel y'_t in time frame *t*. Determine the optical flow matching error for each existing $y_{t+1} \in \{Y_{t+1}\}$ and for each new feature point $\tilde{y}_{t+1} \in \{\tilde{Y}_{t+1}\}$, respectively:

$$E_{t+1}(y_t, y_{t+1}) = \|P_t(y_t) - P_{t+1}(y_{t+1})\| \tag{49}$$

$$E_{t+1}(\tilde{y}_{t+1}, \tilde{y}_t) = \|P_{t+1}(\tilde{y}_{t+1}) - P_t(\tilde{y}_t)\| \tag{50}$$

- Define $\mathrm{HC}_t(y_t)$ as the corner score returned by the Harris corner detector for feature point y_t at time frame *t*.
- Destroy any existing feature point y_{t+1} or new feature point $\tilde{y}_{t+1}$ that fails either of its respective tests:

$$y_{t+1} : E_{t+1}(y_t, y_{t+1}) < Th_E \tag{51}$$

$$\mathrm{HC}_{t+1}(y_{t+1}) > Th_{HC} \tag{52}$$

$$\tilde{y}_{t+1} : E_{t+1}(\tilde{y}_t, \tilde{y}_{t+1}) < Th_E \tag{53}$$

$$\mathrm{HC}_{t+1}(\tilde{y}_{t+1}) > Th_{HC} \tag{54}$$

  where Th_E is the threshold for the optical flow matching error and Th_{HC} is the threshold for the corner score.

- Let the sets of states and of feature points for the next iteration be:

$$\{X_{t+1}\} = \{X_{t+1}\} + \{\tilde{X}_{t+1}\} \tag{55}$$

$$\{Y_{t+1}\} = \{Y_{t+1}\} + \{\tilde{Y}_{t+1}\} \tag{56}$$

In summary, with evolution we detect additional new feature points in each frame. Instead of initializing the state of the newly generated feature point from scratch, we borrow information from its reliable neighbors. Only the feature points with good 2D correspondences (large corner scores and low matching errors) can be passed on to the

next iteration. Therefore, we allow those feature points with good 2D correspondence to continue in the future iterations where they would hopefully become more reliable and have a low Kalman error. We also let a feature point die if it does not have good 2D correspondence across neighboring frames since we cannot accurately reconstruct its 3D point position anyways.

A.3. Reconstruction and Rendering

In this section, we describe the dynamic-scene reconstruction and rendering based on the estimated feature point states in detail.

Once the tracking of the evolving points is complete, the underlying states can be used to construct depth maps. First, as we are tracking the feature points using EKF, we utilize the Kalman error to remove those feature points which were poorly tracked. The remaining, reliable states (3D positions and motions) are then used to build the time-dependent 3D mesh (depth map) for rendering at the desired time instance. Given the mesh and image textures, we are able to render the scenes at the desired time instance and viewpoint.

We first present the interpolation of the feature point's state information at a specific time instance. We then give a detail description about how to render the scene at the desired time instance and viewpoint based on the texture interpolation.

Feature Point's State Interpolation

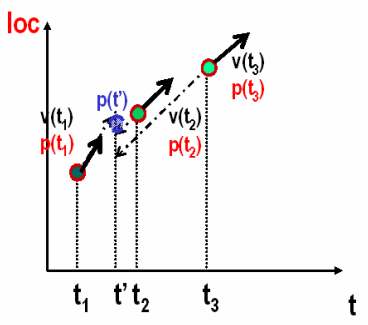


Figure 47. Position interpolation of the feature point at *t'* from its positions and motions at t_1, t_2, and t_3

Given one-to-one correspondence of a feature point at time t_i, where *i=1, 2, ..., N,* we determine its 3D position $\mathbf{p}_{w_t}(t')$ at the desired time *t'* from its available 3D positions $\mathbf{p}(t_i)$ and motions $\mathbf{v}(t_i)$ as shown in Figure 47.

The interpolated 3D position $\mathbf{p}_{w_t}(t')$ is simply the weighted average of the linear predicted positions based on its *N* available 3D positions and motions as follows.

$$\mathbf{p}_{w_t}(t') = \frac{\sum_i w_t(t_i)\left[\mathbf{p}(t_i) + \mathbf{v}(t_i)(t'-t_i)\right]}{\sum_i w_t(t_i)}, w_t(t_i) = \frac{1}{\left|t'-t_i\right|^2} \tag{57}$$

where the interpolation weight $w_t(t_i)$ depends on the time difference between *t'* and t_i.

Since the linear interpolation function $\mathbf{p}_{w_t}(t)$ is a weighted sum of the *N* linear predicted positions, we can prove that our proposed interpolation approach of the position $\mathbf{p}_{w_t}(t')$ and motion $\mathbf{v}(t')$ satisfies the following consistency conditions:

$$\mathbf{p}_{w_t}(t') = \mathbf{p}(t_i), \quad if \quad t' = t_i, \quad for\ i = 1, 2, ..., N \tag{58}$$

$$\mathbf{v}(t') = \mathbf{v}(t_i), \quad if \quad t' = t_i, \quad for\ i = 1, 2, ..., N \text{, where } \mathbf{v}(t') = \left.\frac{\partial \mathbf{p}_{w_t}(t)}{\partial t}\right|_{t=t'} \tag{59}$$

and for any two time t_i and t_j,

$$\lim_{t_i \to t_j} \left| \mathbf{p}_{w_t}(t_i) - \mathbf{p}_{w_t}(t_j) \right| = 0; \ \lim_{t_i \to t_j} \left| \mathbf{v}(t_i) - \mathbf{v}(t_j) \right| = 0; \quad (60)$$

where t_i and t_j can be any time.

Dynamic-Scene Rendering

We first determine the feature point locations and colors at the desired time instance based on the selected corresponding feature point set. We then build the texture mesh and render the scene.

1. Select feature points

To render the scene at time *t'* and viewpoint *view', w*e first identity the *N* neighboring images captured at time t_i and viewpoint $view_i$ as shown in Figure 48. The neighboring captured images are identified based on the distance measurement in the view and time domain with the tradeoff parameters a_t, $a_{location}$ and a_{angle} as follows.

$$\text{distance} = a_t diff(time) + a_{location} diff(camera\,location) + a_{angle} diff(camera\,angle) \quad (61)$$

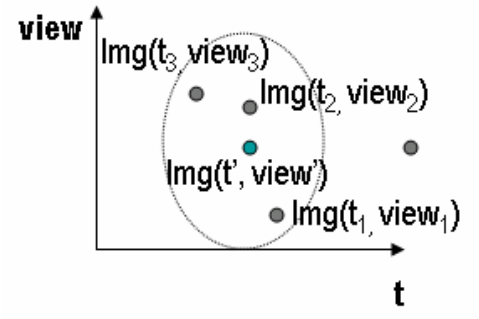


Figure 48. Determining the neighboring images for rendering

We then identify the feature point candidates on these selected images for rendering as shown in Figure 49. We use the feature points for rendering, if the number of their

corresponding feature points at different view points is more than M ($M \leq N$). To have one-to-one correspondence among the N neighboring captured images for interpolation later, we interpolate the states (3D position and motion) of those missing corresponding feature points from their existing corresponding feature points using the *feature point state interpolation* method.

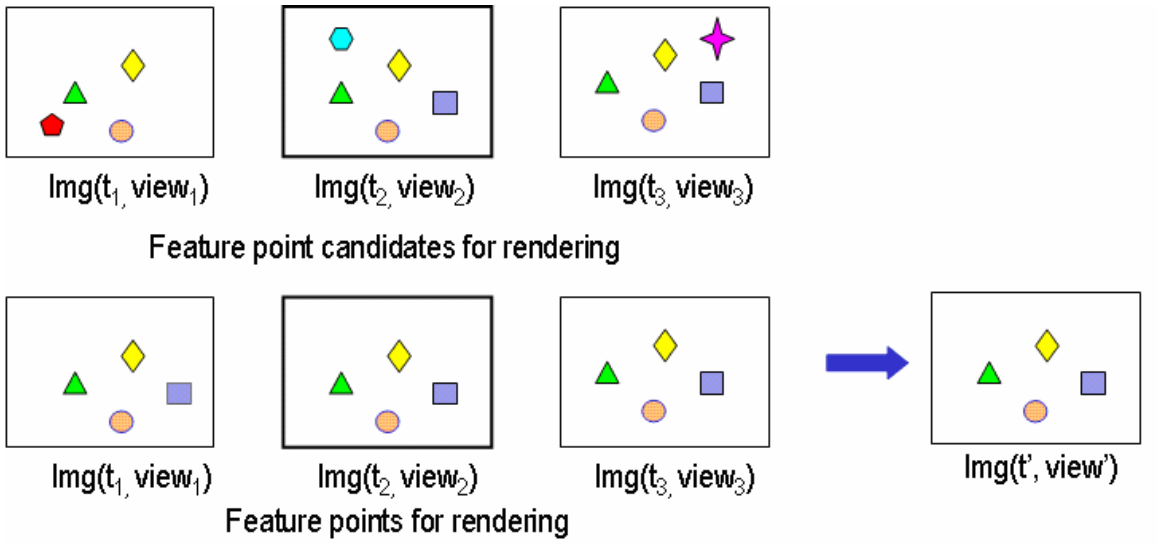


Figure 49. Feature point selection at *t'*

2. Determine feature point's location and color

After identifying the one-to-one correspondence of the feature points in the N neighboring images, we determine the state information of the feature points at time *t'* using the *feature point state interpolation* approach again.

Each feature point color at the time *t'* and the viewing direction θ' is interpolated from the corresponding feature point colors at time t_i based on the viewing direction θ_i as shown in Figure 50, where $i=1, 2, ..., N$. The proposed rendering method interpolates the color for each feature point at the time *t'* and viewing direction θ' as follows:

$$color(t',\theta') = \frac{\sum_i w(t_i,\theta_i)color(t_i,\theta_i)}{\sum_i w(t_i,\theta_i)} \tag{62}$$

The color interpolation weights $w(t_i,\theta_i)$ depend on the differences in viewing directions and in time as follows.

$$w(t_i,\theta_i) = w_t(t_i)w_{view}(\theta_i), w_t(t_i) = \frac{1}{|t'-t_i|^2}, w_{view}(\theta_i) = \frac{1}{\sin|\theta'-\theta_i|} \tag{63}$$

where $w_t(t_i)$ is the weight based on the time difference, and $w_{view}(\theta_i)$ is the weight based on the viewing direction difference. Therefore, the feature point's interpolated color at the time *t'* is similar to its neighbor's colors, if the feature point's viewing direction θ' and time *t'* is similar to the neighbor's viewing direction θ_i and time t_i.

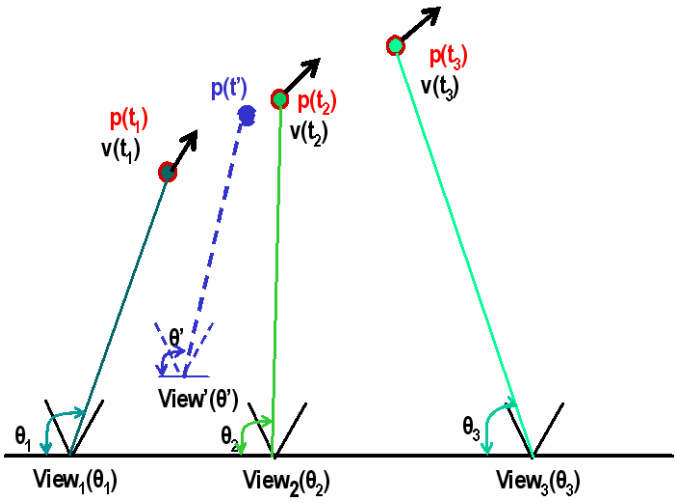


Figure 50. Interpolation for a feature point's color

3. Render the scene texture mesh based on the feature points

We project the feature points at the time *t'* to the image with the smallest viewing-direction-and-time distance (img(t_2 ,$view_2$) as shown in Figure 48). We use Delaunay triangulation [71] to connect them to generate the triangle mesh as shown in Figure 51.

We render the texture of each triangle by interpolating the texture of the corresponding triangles in the images captured at the neighboring viewpoints and times. The color interpolation weights for a point inside a triangle are linearly interpolated from the corresponding weights of the feature points at the vertices of the triangle based on distance.

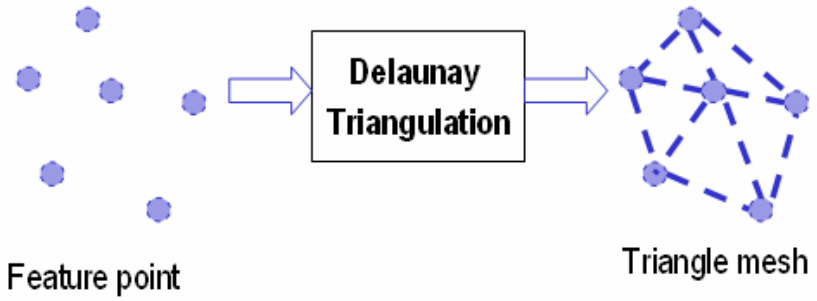


Figure 51. Delaunay triangulation

A.4. Experimental Results

We show the experimental results of our video-based rendering with a single moving camera.

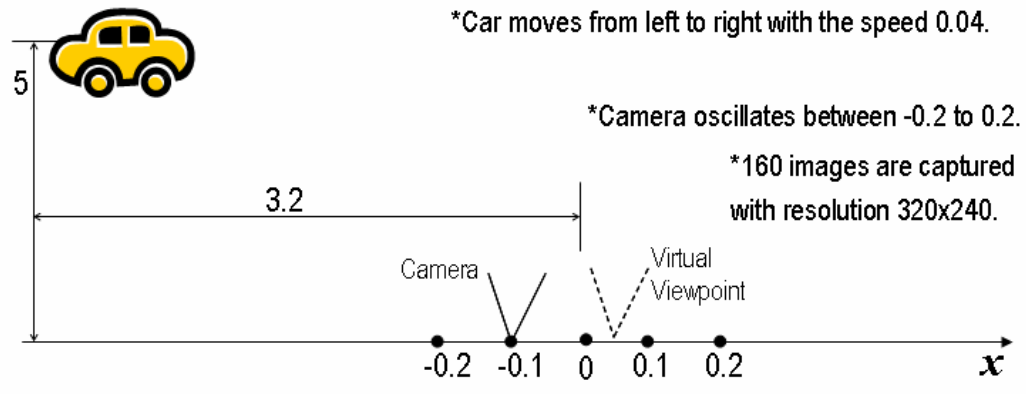


Figure 52. Experiment setup of a moving car

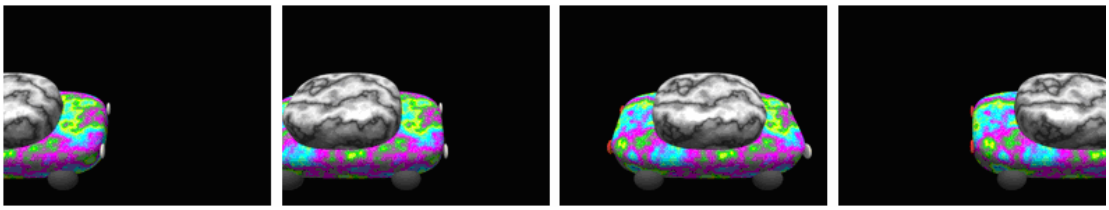
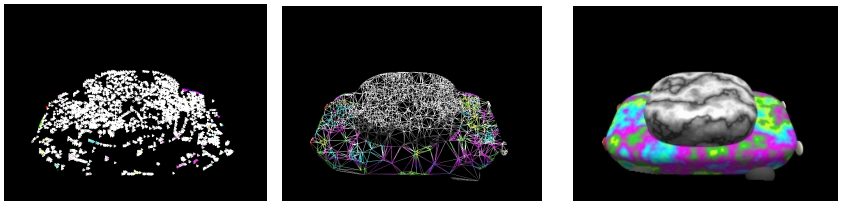
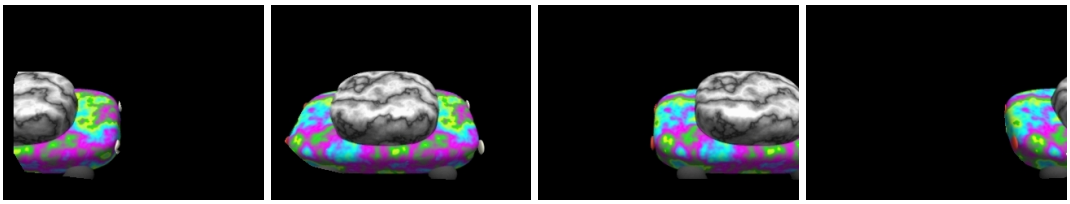


Figure 53. Sample input images of a moving car



(a) 3D points (b) 3D mesh (c) Rendered image

Figure 54. Reconstruction of a dynamic scene



(a) Frame 35 (b) Frame 55 (c) Frame 115 (d) Frame 155

Figure 55. Proposed video-based rendering at the virtual viewpoint

We first ran experiments on synthetic data. A moving toy car was captured by an oscillating camera simulated by POV-ray [61]. The car moved with the speed of 0.04 from left to right. The camera oscillated between -0.2 to 0.2, and was 5 away from the car center vertically. We rendered the scene activities at the virtual viewpoint $x = 0.05$ as shown in Figure 52. As illustrated in Figure 53, a total of 160 images were captured at a low resolution of 320x240 pixels with the known intrinsic and extrinsic camera calibration parameters as the input image sequence. As shown in Figure 54, we first reconstructed the feature points of the scene using our proposed feature point evolution algorithm. Based on the reconstructed feature point's positions and motions in space, we

then built the triangle mesh of the scene at the desired time. Finally, we rendered the scene at a novel virtual viewpoint (see Figure 52) using the time-dependent meshes and textures from the captured images. We showed that evolution had a good rendering quality in the experiment, for example, the car's geometry and motion were well rendered as shown in Figure 55. A video with both capturing, rendering, and the detail parameter settings can be downloaded at

ftp://amp.ece.cmu.edu/Outbox/ICIP2006/SynMoveCar.zip.

The rendering results in Figure 54 and in Figure 55 will be bad without the feature point evolution involved, since the object appearance relative to the camera changes dramatically between the first frame and the last frame as shown in Figure 53.

We also performed experiments on real data of a stationary scene. A peanut can with a checker board was captured by a single moving camera with pre-calibrated intrinsic camera parameters.

Figure 56. Sample input images of a peanut can

As illustrated in Figure 56, 35 images were captured at a high resolution of 2592x1944 pixels as the inputs. The extrinsic camera calibration parameters were derived from the checker board pattern in the image. Using 3D reconstruction results from stereo and from evolution, we rendered images at different viewpoints, as shown in Figure 57.

Figure 57. Rendering results of the real scene. Each row represents different viewpoint. The rendering quality in evolution is better compared to stereo, as illustrated by the ovals.

Therefore, the rendering in evolution is better compared to the stereo technique because of the reliable 3D reconstruction; notice the difference in the rendering of the peanut can lid.

A.5 Conclusions

We proposed a feature point evolution algorithm for dynamic scene reconstruction that exploits the characteristics of dynamic, evolving scenes. As the feature points initially extracted at the beginning of a video are only stable for a limited amount of time, we only track them while they are stable. Furthermore, as the video gets new perspectives on the scene, new, interesting feature points will appear and be tracked. As these new feature points may be in the vicinity of existing feature points, the new feature points' states can be estimated using the refined estimates of the existing feature points' states. The result is a model that is better constructed than using existing reconstruction techniques for rendering.

Appendix B. A Single Scale Leung-Malik (LM) Filter Bank

A single scale LM filter bank is a multi-orientation filter bank with 15 filters. It consists of first and second derivatives of Gaussians at 6 orientations with the base scale $\sigma = \sqrt{2}$ making a total of 12, 2 Laplacian of Gaussian (LOG) filters, and 1 Gaussian. The first and second derivative filters occur at the base scale with an elongation factor of 3 (i.e. $\sigma_x = \sigma$ and $\sigma_y = 3\sigma$). The Gaussians occur at the base scale, while the 2 LOG filters occur at σ and 3σ. The filter bank is illustrated in Figure 58.

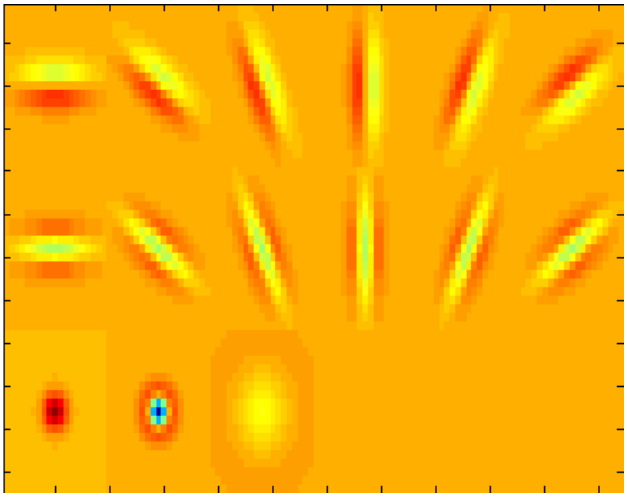


Figure 58. LM filter bank with a mix of edge, bar and spot filters at multiple orientations

BIBLIOGRAPHY

[1] S. Chen and L. Williams, "View Interpolation for Image Synthesis", *Computer Graphics (SIGGRAPH'93)*, pp. 279-288, Aug. 1993.

[2] S. M. Seitz and C.M. Dyer, "View Morphing", *Computer Graphics (SIGGRAPH'96)*, pp. 21-30, Aug. 1996.

[3] L. McMillan, "An Image-Based Approach to Three-Dimensional Computer Graphics", *Technical Report, UNC Computer Science* TR97-013, 1999.

[4] J. Shade, S. Gortler, L. W. He, and R. Szeliski, "Layered Depth Images", *Computer Graphics (SIGGRAPH'98)*, pp. 231-242, July 1998.

[5] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach", *Computer Graphics (SIGGRAPH'96)*, pp. 11-20, Aug. 1996.

[6] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured Lumigraph Rendering", *Computer Graphics (SIG-GRAPH'01)*, pp 425-432, Aug. 2001.

[7] S. B. Kang, "A Survey of Image-Based Rendering Techniques", *VideoMetrics, SPIE* Vol. 3641, 1999, pp. 2-16.

[8] H.Y. Shum, S.B. Kang, and S.C. Chan, "Survey of Image-Based Representations and Compression Techniques," *CirSysVideo(13)*, No. 11, pp. 1020-1037, November 2003.

[9] C. Zhang and T. Chen, “A Survey on Image-Based Rendering – Representation, Sampling and Compression”, *EURASIP Signal Processing: Image Communication*, Vol. 19, pp. 1-28, Jan. 2004.

[10] Y. Lu, J.Z. Zhang, Q.M.J. Wu and Z.N. Li, "A Survey of Motion-Parallax-Based 3-D Reconstruction Algorithms," *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34(4), pp 532-548, 2004.

[11] S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F. Cohen, “The Lumigraph”, *Computer Graphics (SIGGRAPH’96)*, August 1996, pp. 43-54.

[12] R. Szeliski, “Rapid Octree Construction from Image Sequences,” *CVGIP: Image Understanding 58,* 1, pp 23–32, July 1993.

[13] A. Laurentini, “The Visual Hull Concept for Silhouette Based Image Understanding”, *IEEE PAMI*, pp. 150-162, Vol. 16, No. 2, 1994.

[14] C. Tomasi and T. Kanade, “Shape and Motion from Image Streams Under Orthography: A factorization Approach”, *International Journal of Computer Vision*, 9(2):137-154, 1992.

[15] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, “Visual Modeling with a Hand-Held Camera,” *International Journal of Computer Vision* 59(3), 207-232, 2004.

[16] R. T. Collins, "A Space-Sweep Approach to True Multi-Image Matching," *IEEE Computer Vision and Pattern Recognition*, June 1996, pp. 358-363.

[17] A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nistér and M. Pollefeys, “Towards Urban 3D

Reconstruction From Video," *Invited paper, Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT 2006)*.

[18] L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-Quality Video View Interpolation Using a Layered Representation," *ACM Transaction on Graphics*, vol. 23, no. 3, pp. 598--606, Aug. 2004.

[19] S. M. Seitz and C. R. Dyer, "Photorealistic Scene Reconstruction by Voxel Coloring", *CVPR 1997*, pp. 1067-1073.

[20] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms", *International Journal of Computer Vision*, pp. 7-42, Vol. 47, No.1-3, April-June 2002.

[21] Jia-Ching Cheng and José M. F. Moura, "Automatic Recognition of Human Walking in Real Video Sequences," *Journal of VLSI Signal Processing*, vol. 20:(1/2), pp.107-120, October 1998.

[22] D. Gavrila, "The Visual Analysis of Human Movement: A Survey," *Computer Vision and Image Understanding*, Vol. 73, 1, January, 1999.

[23] J. Carranza, C. Theobalt, M. Magnor, and H. Seidel. "Free-Viewpoint Video of Human Actors," *Computer Graphics Annual Conference Series (SIGGRAPH'03)*, pp. 569-577, San Diego, CA, July 2003.

[24] German K.M. Cheung, S. Baker and T. Kanade, "Visual Hull Alignment and Refinement Across Time: A 3D Reconstruction Algorithm Combining Shape-From-Silhouette with Stereo," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2003 (CVPR'03)*, Vol. 2, pp. 375-382, 2003.

[25] German K.M. Cheung, S. Baker and T. Kanade, "Shape-From-Silhouette of Articulated Objects and Its Use for Human Body Kinematics Estimation and Motion Capture," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2003 (CVPR'03)*, Vol. 1, pp. 77-84, 2003.

[26] German K.M. Cheung, "Visual Hull Construction, Alignment and Refinement for Human Kinematic Modeling, Motion Tracking and Rendering," *Technical Report CMU-RI-TR-03-44*, PhD Thesis, Robotics Institute, Carnegie Mellon University, October 2003.

[27] M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu, "Mosaic Representations of Video Sequences and Their Applications," *Signal Processing: Image Communication*, Vol. 8, No. 4, May 1996.

[28] M. Irani and P. Anandan. "Video Indexing Based on Mosaic Representations", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 86(5):905-921, May 1998.

[29] P.M.Q. Aguiar, R. Jasinschi, J.M.F. Moura, and C. Pluempitiwiriyawej, "Content-Based Image Sequence Representation," ed. Todd Reed, in *Digital Image Sequence Processing: Compression and Analysis*, CRC Press Handbook, 2004.

[30] R.S. Jasinschi, J.M.F. Moura, J.C.Cheng, and A. Asif, "Video Compression Via Constructs," in *Proceedings of IEEE ICASSP*, Vol. 4, pp. 2165-2168, 1995.

[31] R.S. Jasinschi and J.M.F. Moura, "Content-Based Video Sequence Representation," in *Proceedings of IEEE ICIP*, pp. 229-232, 1995.

[32] R.S. Jasinschi and J.M.F. Moura, “Nonlinear Video Editing by Generative Video,” in *Proceedings of IEEE ICASSP*, 1996.

[33] R.S. Jasinschi and J.M.F. Moura, “Generative Video: Very Low Bit Rate Video Compression,” 1998.

[34] P.M.Q. Aguiar and J.M.F. Moura, "3D Modeling from 2D Video," *IEEE Transactions on Image Processing*, Volume: 10 (10), pp. 1541-1551, 2001.

[35] P.M.Q. Aguiar and J.M.F. Moura, “Rank 1 Weighted Factorization for 3D Structure Recovery: Algorithms and Performance Analysis,” I*EEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 25 (9), pp. 1134-1149, September 2003.

[36] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani, “Hierarchical Model-Based Motion Estimation,” In *Proc. ECCV*, pp. 237-252, 1992.

[37] S. Vedula, S. Baker, and T. Kanade, “Image-Based Spatio-Temporal Modeling and View Interpolation of Dynamic Events,” *ACM Transactions on Graphics*, Vol. 24, No. 2, April, 2005.

[38] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, “Three-Dimensional Scene Flow,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 3, March, 2005, pp. 475 - 480.

[39] S. Vedula, S. Baker, and T. Kanade, “Spatio-Temporal View Interpolation,” *Proceedings of the 13th ACM Eurographics Workshop on Rendering*, June, 2002.

[40] R.E. Kalman. “A New Approach to Linear Filtering and Prediction Problems,” *Transactions of the ASME: Series D, Journal of Basic Engineering*, 82:35.45, March 1960.

[41] H. Cox, "On the Estimation of State Variables and Parameters for Noisy Dynamic Systems," *IEEE Trans. on Automatic Control*, 9(1):5.12, 1964.

[42] A. Doucet, S. Godsill, and C. Andrieu, "On Sequential Monte Carlo sampling Methods for Bayesian Filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197-208, 2000.

[43] L. Matthies, R. Szeliski, and T. Kanade, "Kalman Filter-Based Algorithms for Estimating Depth from Image Sequences," *Tech. Report CMU-RI-TR-88-01*, Robotics Institute, Carnegie Mellon University, January, 1988.

[44] A. Azarbayejani, A.P. Pentland. "Recursive Estimation of Motion, Structure, and Focal Length," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 6, pp. 562-575, June 1995.

[45] J. Alon and S. Sclaroff, "Recursive Estimation of Motion and Planar Structure," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. II, pp 550-556, 2000.

[46] S. Zhou, R. Chellappa, and B. Moghaddam, "Visual Tracking and Recognition Using Appearance-Adaptive Models in Particle Filters," *IEEE Transactions on Image Processing (IP)*, Vol. 11, pp. 1434-1456, November 2004.

[47] S. Zhou, R. Chellappa and B. Moghaddam, "Appearance Tracking Using Adaptive Models in a Particle Filter," *Asian Conference on Computer Vision (ACCV)*, January 2004.

[48] S. Zhou, R. Chellappa and B. Moghaddam, "Adaptive Visual Tracking and Recognition Using Particle Filters," *IEEE International Conference on Multimedia & Expo (ICME)*, July 2003.

[49] R. Collins, Y. Liu, and M. Leordeanu, “On-Line Selection of Discriminative Tracking Features,” *IEEE Transaction on PAMI*, 27(10), pp 1631-1643, October 2005.

[50] M. Trajkovic and M. Hedley, “Robust Recursive Structure and Motion Recovery under Affine Projection,” *Proc. British Machine Vision Conference-97*, Essex, UK, September 1997.

[51] Y. S. Yao and R. Chellappa, “Tracking a Dynamic Set of Feature Points,” *IEEE Trans. Image Processing*, Vol. 4, No. 10, 1995.

[52] A. J. Davison and D. W. Murray, “Simultaneous Localization and Map-Building Using Active Vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):865—880, 2002.

[53] Wende Zhang and Tsuhan Chen, “Video-Based Rendering Using Feature Point Evolution,” *IEEE International Conference on Image Processing (ICIP)*, 2006.

[54] D. Hoiem, A.A. Efros, and M. Hebert, “Automatic Photo Pop-up,” *ACM SIGGRAPH*, August 2005.

[55] D. Hoiem, A.A. Efros, and M. Hebert, "Recovering Surface Layout from an Image," accepted by *International Journal of Computer Vision*.

[56] A. Saxena, S. H. Chung, A. Y. Ng. "Learning Depth from Single Monocular Images," In NIPS 18, 2005.

[57] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient Graph-Based Image Segmentation,” *International Journal of Computer Vision*, 59(2), 2004.

[58] T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability Estimates for Multi-Class Classification by Pairwise Coupling," *Journal of Machine Learning Research*, 5:975–1005, 2004.

[59] T. Leung and J. Malik, "Representing and Recognizing the Visual Appearance of Materials Using Three-Dimensional Textons," *International Journal of Computer Vision*, 43(1):29-44, June 2001.

[60] http://www.cs.cmu.edu/~dhoiem/projects/data.html

[61] Ray tracking software at http://www.povray.org/.

[62] Z. Zhang, "A Flexible New Technique for Camera Calibration," *Microsoft Technical Report MSR-TR-98-71*, Dec 1998.

[63] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. "Visual Modeling with a Hand-Held Camera," *International Journal of Computer Vision*, 59(3), 2004.

[64] A. Doucet, S. Godsill, and C. Andrieu, "On Sequential Monte Carlo Sampling Methods for Bayesian Filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197-208, 2000.

[65] A. Doucet, N. de Freitas, and N. Gordon, "Sequential Monte Carlo Methods in Practice," *New York: Springer-Verlag*, 2001.

[66] M. Isard and A. Blake, "Condensation -- Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29-1, pp. 5--28, 1998.

[67] K. Kanazawa, D. Koller, and S. Russell, "Stochastic Simulation Algorithms for Dynamic Probabilistic Networks," in *Proc. of 11th annual conference on uncertainty in AI*, pp. 346--351, 1995.

[68] J. Y. Aloimonos. "Perspective Approximations," *Image and Vision Computing*, 8(3), pp 177-192, August 1990.

[69] C. Harris, M. Stephens, "A Combined Corner and Edge Detector," In *Proc. of 4th Alvey Vision Conf.*, pp. 189-192, 1988.

[70] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework," *International Journal of Computer Vision*, Vol. 56, No. 3, pp. 221 – 255, March, 2004.

[71] A. Okabe, B. Boots, and K. Sugihara, "Spatial Tessellations: Concepts and Applications of Voronoi Diagrams," *New York, Wiley*, 1992.