Image Based View Synthesis

Jiangjian Xiao
University of Central Florida

Image Based View Synthesis

by

Jiangjian Xiao
B.E. Beijing University of Aeronautics and Astronautics
M.E. Beijing University of Aeronautics and Astronautics
M.S. University of Central Florida

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Fall Term
2004

Major Professor:
Mubarak Shah

Abstract

This dissertation deals with the image-based approach to synthesize a virtual scene using sparse images or a video sequence without the use of 3D models. In our scenario, a real dynamic or static scene is captured by a set of un-calibrated images from different viewpoints. After automatically recovering the geometric transformations between these images, a series of photo-realistic virtual views can be rendered and a virtual environment covered by these several static cameras can be synthesized. This image-based approach has applications in object recognition, object transfer, video synthesis and video compression. In this dissertation, I have contributed to several sub-problems related to image based view synthesis.

Before image-based view synthesis can be performed, images need to be segmented into individual objects. Assuming that a scene can approximately be described by multiple planar regions, I have developed a robust and novel approach to automatically extract a set of affine or projective transformations induced by these regions, correctly detect the occlusion pixels over multiple consecutive frames, and accurately segment the scene into several motion layers. First, a number of seed regions using correspondences in two frames are determined, and the seed regions are expanded and outliers are rejected employing the graph cuts method integrated with level set representation. Next, these initial regions are merged into several initial layers according to the motion similarity. Third, the occlusion order constraints on multiple frames are explored, which guarantee that the occlusion area increases with the temporal order in a short period and effectively maintains segmentation consistency over multiple consecutive frames. Then the correct layer segmentation is obtained by using a graph cuts algorithm, and the occlusions between the overlapping layers are explicitly determined. Several experimental results are demonstrated to show that our approach is effective and robust.

Recovering the geometrical transformations among images of a scene is a prerequisite step for image-based view synthesis. I have developed a wide baseline matching algorithm to identify the correspondences between two un-calibrated images, and to further determine the geometric relationship between images, such as epipolar geometry or projective transformation. In our approach, a set of salient features, edge-corners, are detected to provide robust and consistent matching primitives. Then, based on the Singular Value Decomposition (SVD) of an affine matrix, we effectively quantize the search space into two independent subspaces for rotation angle and scaling factor, and then we use a two-stage affine matching algorithm to obtain robust matches between these two frames. The experimental results on a number of wide baseline images strongly demonstrate that our matching method outperforms the state-of-art algorithms even under the significant camera motion, illumination variation, occlusion, and self-similarity.

Given the wide baseline matches among images I have developed a novel method for Dynamic view morphing. Dynamic view morphing deals with the scenes containing moving objects in presence of camera motion. The objects can be rigid or non-rigid, each of them can move in any orientation or direction. The proposed method can generate a series of continuous and physically accurate intermediate views from only two reference images without any knowledge about 3D. The procedure consists of three steps: segmentation, morphing and post-warping. Given a boundary connection constraint, the source and target scenes are segmented into several layers for morphing. Based on the decomposition of affine transformation between corresponding points, we uniquely determine a physically correct path for post-warping by the least distortion method. I have successfully generalized the dynamic scene synthesis problem from the simple scene with only rotation to the dynamic scene containing non-rigid objects. My method can handle dynamic rigid or non-rigid objects, including complicated objects such as humans.

Finally, I have also developed a novel algorithm for tri-view morphing. This is an efficient image-based method to navigate a scene based on only three wide-baseline un-calibrated images without the explicit use of a 3D model. After automatically recovering corresponding points between each pair of images using our wide baseline matching method, an accurate trifocal plane is extracted from the trifocal tensor implied in these three images. Next, employing a trinocular-stereo algorithm and barycentric blending technique, we generate an arbitrary novel view to navigate the scene in a 2D space. Furthermore, after self-calibration of the cameras, a 3D model can also be correctly augmented into this virtual environment synthesized by the tri-view morphing algorithm. We have applied our view morphing framework to several interesting applications: 4D video synthesis, automatic target recognition, multi-view morphing.

Dedicated to my parents.

Acknowledgments

I would like to thank my advisor Dr. Murbarak Shah for his guidance, encouragement and support, particularly when I changed my speciality to computer vision. I deeply feel that I am very fortunate to have an opportunity to work with him during my Ph.D time. I also thank my committee members, Dr. Lobo, Dr. Pattanaik, Dr. Rolland, who gave me lots of valuable comments during my study.

During my research, I have benefited from interactions with many people. I would like to thank Rao, Lisa, Alper, Yunjun, Xiaochun, Yaser, and Khurram for many illuminating discussions. Special thanks to Lisa and Saad for proof reading my papers.

Most of all, I am deeply grateful to my parents for supporting my study for up to twenty years. And also, I would like to thank my wife Mian, who gave me her support, encouragement, and love.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A camera can capture pictures of a real scene using a perspective projection, which is similar to human eyes. In real world, human eyes can extract 3D information from the pictures projected on the retina to perceive and understand the 3D world. In computer vision, the task to recover 3D information of real scenes from 2D images is still an open problem for researchers. Inversely, computer graphics begins with geometric models, textures and illumination to generate a visual representation for the 3D world. Currently, there is a lot of interest to combine computer graphics and computer vision, and use them in synergy to solve a variety of problems between the two areas[55], such as:

- Dynamic image-based rendering or view interpolation.
- Virtual reality (VR) using image based rendering.
- Reconstruction of highly realistic 3D static models from calibrated or non-calibrated image sequences.
- Building dynamic deformation models from real video sequence.

- Model based tracking, registration, motion estimation.

In this thesis, we attempt to solve some of challenging problems at the confluence of computer vision and computer graphics. Particularly, given a set of sparse uncalibrated images, we want to generate a photo-realistic virtual scene, where the user can immerse and navigate it. We consider that such scene can contain several rigid or non-rigid (articulated) moving objects, where each of them can rotate around any axis or translate in any direction. Employing computer vision techniques, the correspondences among multiple images are estimated and a novel view is further generated by using image based rendering technique. As a result, a continuous and seamless transition over the original views can be obtained by 2D warping of shapes and blending the colors without using 3D models.

1.1 MOTIVATION AND GOAL

Using geometry to reconstruct the 3D model from the images has been studied for a long time in computer vision and computer graphics. Such approaches usually need a large number of views and tedious manual work. Another kind of approach is based on image based rendering (IBR), which does not need a precise 3D model and avoids the tedious modeling procedure. Using image warping and interpolation, we can also recover the transformation and illumination variations of the scene from the images. IBR is broadly used to generate virtual views to visualize and navigate within the synthesized environment. Instead of using geometrical primitives (such as a polygon mesh) as in traditional graphics, these approaches discover the geometrical relationships between multiple images based on perspective principles. Finally, ray-correspondences or point-correspondences are recovered to render a new virtual view by a back-tracing projection or forward warping.

Most IBR methods are based on a plenoptic function, which represents ray properties of a scene. Therefore, in order to record each ray from the scene, these approaches, such as light fields [54], lumigraphs [31], panorama mosaics [79], and visual hulls [59], require a large number of pictures for a static scene. Another limitation of the existing approaches is that they require special markers in images for camera calibration [117, 59, 6, 31, 16, 54]. Some of them even need an explicit 3D model or geometric proxy to trace ray correspondences [16, 6, 31], and others require an assumption about depth [79].

Another class of approaches for interpolating two reference images is called image morphing and view morphing [102]. These approaches can transform a source view into a target view, which can be categorized into three groups: shape blending [1, 90], feature-based morphing [56, 11], and view morphing [23, 73, 105]. Compared with the ray tracing methods, these approaches use relatively few images. However, since shape blending and feature-based morphing usually cannot retain strict geometrical properties of the scene, they are mainly used for images of *different* scenes (e.g. morphing of an elephant into a deer). View morphing and view interpolation are based on perspective geometry principles, which can provide more realistic results when blending the images of the *same* scene. Another drawback is that all of these approaches can only be used to fill the gap between two views (i.e. navigation is limited to a straight line between two original views), therefore cannot be used to navigate in a 2D virtual environment.

However, the ultimate goal in this thesis is to relax the assumptions made by the previous IBR methods and get even better results with less input data and interactions. In particular, we are interested to see:

- Can we determine corresponding feature points and the epipolar geometry over a pair of images with different appearances?
- Can we perform accurate segmentation for a video sequence to extract 2.5D information?
- Can we create a virtual environment by using only three uncalibrated wide-baseline images?
- Can we generate a virtual moving video camera using the videos acquired by a few fixed video cameras?
- Can we fill in the gaps in a dynamic scene based on only sparse still images?
- Can we morph the deformed objects in a scene only using two images? and what kind of deformation we can handle?

1.2 OUR APPROACH AND CONTRIBUTIONS

In this dissertation, we propose a novel technique to synthesize a series of virtual views for a complicated 3D scene, where the objects may move or deform. Our input data are a set of sparse images of the scene, which may be taken at the different locations, orientations, and times. After acquiring the sparse images of the scene, the epipolar geometry among the original cameras can be automatically determined by recovering corresponding feature points between each image pair. Based on the initial projective relationship of the cameras, the disparity or depth can be recovered using stereo-shading method. As a result, we can generate an arbitrary novel view by interpolating the original images, and easily navigate through the scene.

Our approach consists of the following key steps as the follows:

- Wide baseline matching:

  In our input data set, the reference images are wide baseline images under significant camera motions with illumination changes, occlusions, and self-similarities. We use affine invariant features, edge-corners, as robust and consistent matching primitives. Then, based on Singular Value Decomposition (SVD) of an affine matrix, a two-stage affine matching algorithm is designed to obtain robust matches over two frames.

- Segmentation and compositing:

For dynamic view synthesis, the moving objects have to be extracted from the reference image into different layers. Each layer is assumed to have a different epipolar geometry in all reference images.

- Morphing and interpolation:

  After recovering epipolar geometry amongst the reference views, the original images can be rectified by using projective geometrical transformation. Then, employing linear interpolation of the rectified views, a novel view can be efficiently blended.

- Postwarping:

  In order to recover a reasonable view position and orientation, we re-project the morphing image to correct the warping path. Using the least distortion method, we can uniquely determine an optimized postwarping path with the least shape distortion.

The main contributions of this work are:

- We have successfully solved the problem of how to obtain reliable corresponding points over two wide baseline frames with fairly different appearances.
- We design an effective algorithm to obtain accurate layer segmentation from a video sequence.
- We have successfully solved the dynamic view interpolation problem by combining the least distortion method and boundary connection constraint.

- We have developed a system to synthesize a virtual environment for navigation using only sparse images.
- We present a novel approach to implement 4D video based on sparse static video cameras.

Our approach can be also used in several other applications, such as filling a gap in a movie, creating virtual views from static images, movie-making, compressing movie sequences for fast transmission, and switching camera views from one to another.

1.3 DISSERTATION OVERVIEW

The dissertation is organized as follows. Chapter 2 discusses the related work. Chapter 3 presents a novel approach to recover robust corresponding points over two wide baseline images. Chapter 4 provides another powerful tool to extract accurate layers from video sequences. Chapter 5 starts by analyzing simple dynamic view morphing for a rigid object with translation and rotation respectively, and then explains how to combine the least distortion method and the boundary connection constraint to obtain a reasonable postwarping. Chapter 6 proposes a novel framework for tri-view morphing, 4D video synthesis, and multiple view morphing. Chapter 7 summarizes this dissertation.

CHAPTER 2

RELATED WORK

In this chapter, we will review the related approaches for matching, shape blending, view interpolation, and object segmentation.

2.1 TRACKING AND MATCHING

A popular technique for small baseline motion tracking is to minimize the sum of squared differences (SSD) of image intensities over two consecutive frames. However, SSD-based frame-to-frame trackers can deal with translation motion only. Therefore, these trackers may accumulate errors over time due to other geometric deformations, like rotation, scaling, and shearing. Shi and Tomasi demonstrated that the use of an affine model can effectively compensate the errors in video frames [85]. At the same time, they proposed a monitor to evaluate the quality of the correspondences based on the image residuals. Following this direction, Fusiello et al. presented a statistical approach to monitor and reject the outliers [29], and Jin et al. explicitly used illumination information to improve the monitor [42].

Shi and Tomasi [85] claimed that the affine model works very well when the image warping includes scaling, shearing and small rotation, since all of them can be approximated by using the linear expansion of brightness constancy equation. However, the affine model fails to correctly track if a large rotation or a significant scaling between the two frames is introduced.

Pritchett and Zisserman estimated reliable point correspondences based on local planar homographies [87, 68]. These homographies were determined by parallelogram structures or using motion pyramids. Based on these homographies, the matches can be increased by cross-correlation at the sub-pixel level. However, the parallelograms may not be present in certain images, or similar parallelograms may be repeated due to self-similar structure. Hence, it is challenging to find the exactly corresponding structures between wide baseline frames. Moreover, their method only recovers the homographies related to the parallelograms, and misses some other important homographies implied in the frames.

Baumberg detected Harris features using the second moment matrix and scale space [5]. After obtaining affine texture invariant corners, a Mahalanobis distance metric was used to measure similarity between the two feature vectors. Similarly, Tuytelaars and Van Gool extracted affine invariant regions based on a combination of corners and edges, then matched these features using color moments [91]. Brown and Lowe proposed to use groups of interest points to form geometrically invariant descriptors of image regions [9]. Mikolajczyk and Schmid presented recent work on an affine invariant interest point detector [61, 60]. Since they used multi-scale space to determine feature points, their method worked very well for the significant scaling case (homography case). However, their corner descriptors employing high order derivatives cannot fully recover non-linear rotation transformations implied in the affine matrix. For instance, they failed to get correct matches for the “House” sequence frames when the view angle changed significantly (3D case). Ferrari et al. [28] used multiple color images as input to extract the pairwise correspondences. In their method, they also noticed that an affine decomposition is helpful to determine the correspondences over wide baseline images as in our work [107].

2.2 SHAPE BLENDING AND FEATURE BASED MORPHING

Shape blending is an important topic in computer graphics. Given source and target shapes referred to as key shapes, shape blending algorithms can determine a smooth transformation between the key shapes, and generate a series of in-between shape sequence.

Sederberg and Greenwood [77] used a physical mode to establish correspondences between two polygonal shapes. Their method constructed the contour of an object by connected wires. Using the relationship among the polygon angles, they presented an algorithm to search the global minimum work solution by means of a directed graph. The drawback of their method is that it is difficult to specify the values of the parameters of work function in order to achieve the desired result. Based on this method, Sederberg et al. [78] proposed an intrinsic solution to 2-D shape blending by interpolating shape with boundary vertices. Their approach can avoid the shrinkage that normally occurs when a rigid object is rotating. Shapira and Rappoport [82] extended this concept by introducing the internal shape represented by star-skeleton. They modeled the interior of the shape using a star-skeleton scheme. Using this method, the shape and area of the in-between polygons can be controlled better than previous approaches.

Cohen et al. [17] used piecewise curve to fit the boundary with the least square, and then used dynamic programming to optimize the curve matching in the parameter space. Their method established the correspondences by maximizing the sum of the inner products between tangent vectors at the points on the source and target shapes. However, if the orientations of the shapes are different, the method cannot give a correct solution. Tal and Elber [90] extended this approach and used Bspline curves matching to determine the boundary correspondences, and then established compatible triangulation for two shapes. Based on triangulation, they applied texture mapping on each shape. Since the compatible triangulation heavily relied on the boundary points, the internal shape may have large distortion during transformation, even though they tried to use interior hole to improve it.

Recently, Alexa et al. [1] proposed another approach to construct isomorphic triangular meshes using boundary vertex correspondences, and then to interpolate the two images by both boundary and interior vertices. Their approach obtained as rigid as possible shape interpolation by the least distortion method. However, the interior vertices are generated by a compatible triangulation, and they did not guarantee that the corresponding interior points come from the same source point. Therefore, the distortion in the internal area is still difficult to avoid.

All of the above methods are 2D based image morphing methods. They attempt to prevent shape bleeding and make the in-between shapes retain their original appearance as much as possible. These methods are useful and powerful when the two objects are different and the goal is to obtain a smooth transformation between the key frames. However, if the two reference frames come from the same source, these approaches cannot preserve 3D shape information implied in the images.

In order to obtain high quality image sequence blended from reference images, the interior features (landmarks) of the objects are as important as boundary information. If the corresponding features can be extracted correctly from the images, the in-between image sequence can be generated realistically.

In [102], the meshing warping was introduced. This approach is simple and intuitive, where the objects are covered by a nonuniform mesh to specify the coordinates of control points and landmarks. Both meshes have the same topology, the intermediate mesh is linearly interpolated from the two meshes. Beier and Neely [11] introduced a field morphing technique to improve 2D mesh warping. In their method, a uniform 2D mesh is mapped on the objects in the reference images respectively. Rather than requiring all the corresponding features, only line pairs and key feature points were manually marked on each mesh. They utilized the continuity of mesh to control image warping based on fields of influence. The drawback of their approach is that an unexpected distortion or ghost may be produced.

Lee et al. [52] introduced snakes to determine the landmarks semi-automatically. Using snakes, the salient edges can be refined, and the features can be located more precisely than previous methods. Lee applied multilevel free-form deformation (MFFD) across a hierarchy of control lattices to generate one-to-one and C^2-continuous warp function.

Johan et al. [43] extended the work of Sederbeg [77] and Cohen [17] to feature based morphing. Their method established a cost function based on the angle and parameter costs. By minimizing the cost function, an optimized curve matching is obtained. In their method, a dependency graph was constructed for representing the relationship among the feature curves. Based on the directed dependency graph, the intermediate feature curves were generated. In the morphing step, they used field morphing technique to implement the image warping function.

For morphing two different objects, feature based morphing can carry more information including inside and boundary landmarks, and shape blending only use the boundary points. If the objects are from the same source, the corresponding features may implicitly contain some 3D information, even though this may not satisfy epipolar constraints, such as fundamental matrix. Therefore, this group of methods usually can obtain better appearance and effect than the previous ones.

2.3 VIEW MORPHING

View morphing is a different approach based on the principles of projective geometry, which can explicitly preserve 3D information by employing the fundamental matrix.

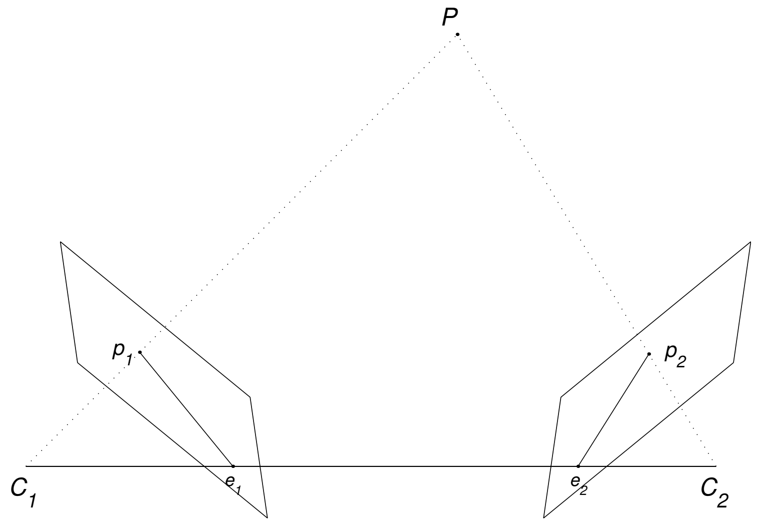


Figure 2.1: The real world point P is projected respectively at p_1,p_2 on projection plane of camera C_1 and C_2, where e_1 and e_2 are the epipoles of the cameras.

Considering two images of an object taken from different view points, the corresponding points in the two images come from the same 3D point in the real world as shown in Figure 2.1. The relationship between two images can be represented by Equation 2.1.

$$p_1^T F p_2 = 0, \tag{2.1}$$

where p_1 and p_2 are corresponding points in reference images, F is the fundamental matrix, which can be computed by linear or nonlinear algorithms [41, 95, 34].

Based on the fundamental matrix, Seitz and Dyer introduced the approach to rectify the reference images to parallel views, and then linearly blended the corresponding points on each scanline [71, 73, 72]. This method can effectively synthesize a physically correct view from two static views. However, they did not consider the partial occlusion case.

Manning and Dyer extended this approach to dynamic view morphing [57]. In their dynamic scenario, the objects were restricted to be rigid and could only have translate motion. By separating the objects into different layers, they computed the fundamental matrix for each layer and applied the static view morphing algorithm to blend the objects.

Welxer and Shashua presented another approach to use more than two images to interpolate a new view. They assumed that a rigid object can move along a straight line path [103]. In their method, the dense correspondence was assumed to be known using cross correlation. Based on this assumption, they not only interpolated the original views, but also extrapolated them to get new views for the extra time steps by dual Htensor, which connects together three views of a coplanar configuration of static and moving points.

For view morphing, we do not need to recover internal and external parameters of the camera from images. Images are directly aligned based on the fundamental matrix. The most important advantage of view morphing is that it can reflect the physical change of the scene accurately without 3D knowledge of the scene.

How to compute the postwarping path is still an ill-posed problem. In Seitzs' method, the path is approximated by the linear interpolation of four control points [75], which is correct only for the translation case with a dominated plane. If the object has a large rotational motion, the shape cannot be maintained and the distortion will be intolerable. The reason is that the depth can be recovered implicitly by interpolating in the translation case, but cannot be recovered in the rotation case.

2.4 MULTI-VIEW SYNTHESIS

Avidan and Shashua proposed their work on tri-view synthesis by using trifocal tensor [83]. They generated an arbitrary novel view for any 3D view position based on three *small* baseline images, where the disparity can easily be determined by the Lucas-Kanade optical-flow method. It will be almost impossible for the Lucas-Kanade method to work for the *wide* baseline images. Since they did not describe the warping function among these three images, they could only demonstrate the novel view synthesis between two images, where their third image was specified as the same as the second one. Maurizio Pilu et. al. determined edge correspondences and used interpolation to generate a new view over trinocular images [67]. However, since they cannot guarantee that the edge correspondences are correct, their disparity map was computed using the conventional edge-scanline algorithm, which is not clean. Therefore, their results encountered a lot of artifacts due to some incorrect correspondences.

Pollefeys and Van Gool combined 3D reconstruction and IBR to render a new view using a sequence of images [64, 65]. They first determined the relative motion between consecutive images, and then recovered the structure of the scene. Next, employing unstructured light field rendering, they generated a virtual view by using view-dependent texture. With the help of the image sequences (small baseline), they estimated dense surface of the scene more accurately, which can efficiently improve the visual effect of their results. However, they could only reproduce one visualization trajectory, which was followed by the operator during the image capturing.

Recently, Zhang et al. proposed to use feature-based morphing with light fields to obtain realistic 3D morphing [117]. In their approach, a large number of images (hundreds of pictures) were taken for each object using an array of calibrated cameras. Then, several feature polygons were manually determined employing a user interface. Using the corresponding feature polygons, they generated a 4D lighting field and grouped the corresponding ray bundles for reference images. Finally, a novel view was synthesized using blending and warping functions on reference images.

2.5 SEGMENTATION

Before performing view interpolation, we first need to segment the image into several layers, such that each layer only includes one rigid object, even though there may be apparent deformation at the joints between the parts.

Segmentation is a fundamental topic in computer vision. There are an abundance of segmentation methods, which can be classified as follows:

2.5.1 SINGLE IMAGE SEGMENTATION

Given a single image, segmentation is to group similar pixels and form a set of uniform, piecewise regions. A simple approach is to use thresholds and histograms to detect the histogram peaks. A good peak can be considered as a candidate of region, which can be segmented by threshold at the valleys between peaks. This is followed by a connected component algorithm to determine the different regions. The approach can be refined by using information of the region's shape and semantics.

Another kind of approach is to perform segmentation based on shape, where the image is broken into disjoint regions based on contour with or without color information [58, 12]. In these methods, a good edge detection or contour analysis should be provided as preliminary information. Medioni introduced a method using tensor voting to detect the saliency of the shape [58]. In their method, sparse tokens are encoded into tensor tokens, and a saliency tensor field is obtained after tensor votings. The saliency tensor field can provide the shape information, which is extracted by using feature extraction process. Based on the contour information, the shape can be divided into several pieces by edges and corners based on the semantics of the region [114].

Recently, two filter-based approach are proposed: one is the bilateral filter [93], the other is the mean-shift filter [18, 20, 21, 22]. In these approaches, a non-linear filter is designed to make a large influence on the pixel when its neighboring pixels are similar to it. Then a group method can be used to partition this smoothed image.

In computer vision area, the most popular methods for image segmentation and restoration are level set [76, 63, 89] and graph cut methods [7, 14, 81]. In these methods, after given the layer number for the image, an energy function is constructed and minimized to assign the pixels into a proper layer label. In the two-label case, these methods can achieved a global solution. For the multi-label case, an approximate global solution can be obtained.

2.5.2 MOTION SEGMENTATION

Automatic extraction of layers from a video sequence has broad applications, such as video compression and coding, recognition, synthesis, registration [112], and completion [119, 118]. In an earlier work, Wang and Adelson proposed the use of optical flow to estimate the motion layers, where each layer corresponds to a smooth motion field [99]. Ayer and Sawhney combined Minimum Description Length (MDL) and Maximum-Likelihood Estimation (MLE) in Expectation-Maximization (EM) framework to estimate the number of layers and the motion model parameters for each layer [4]. Several other approaches used Maximum A-Posteriori (MAP) or MLE for estimation of model parameters assuming different constraints and motion models [66, 94, 48, 116, 101, 74]. Khan and Shah [48] employed MAP framework combining with multiple cues, like spatial location, color, and motion, for segmentation. Smith et. al. [74] integrated the edge information in an EM algorithm to segment the video into regions along the edges. Gaucher and Medioni [33] presented using dense tensor field to detect the discontinuities of motion.

Another class of motion segmentation approaches is to group the pixels in a region by using linear subspace constraints. In [47, 46, 45], Ke and Kanade first expanded the seed regions into the initial layers by using k-connected components. After enforcing a low dimensional linear affine subspace constraint on multiple frames, they clustered these initial layers into several groups, and then assigned the image pixels to these layers. Zelnik-Manor and Irani also used the homography subspace for planar scenes to extract a specific layer and to register the images based on this layer [115].

In motion segmentation area, only few researchers have tried to formulate the occlusion problem between overlapping layers. Giaccone and Jones proposed to use four-label system, "background, uncovered, covered and foreground", to label image pixels [32]. The uncovered and covered pixels correspond to occluded or reappeared respectively between two frames. However, they only restricted their framework to two-layer (foreground and background) sequences, and the segmentation results were not piecewisely smooth. Based on the observation that the segmentation boundary is usually not accurate due to the occlusion, Bergen and Meyer proposed to use motion estimation error to estimate the depth order and refine the segmentation boundary, where no occlusion pixel was identified [10]. Compared with the previous work, our approach clearly formulates the occlusion problem into graph cut framework, and also explicitly identifies the occluded pixels for the scene which contains multiple layers.

Beyond the 2D motion segmentation, multi-body or 3D motion segmentation is another interesting topic in computer vision [27, 19]. In this area, the layer clustering is based on 3D geometry information, such as fundamental matrix [92, 98] and trifocal tensors [97, 40]. Currently, the 3D segmentation is mainly focused on sparse point segmentation and clustering, the dense segmentation of 3D scene is an active research area.

CHAPTER 3

WIDE BASELINE MATCHING

Tracking feature points is a basic computer vision problem. Currently, the tracking techniques over *small* baseline frames, such as a video sequence, are fully developed [25, 85]. In these methods, some matches are automatically detected over an image sequence, and then the outliers are discarded by a corner monitor or a similar rejection criteria. The epipolar geometry between two successive frames are then robustly determined. Once epipolar geometry is available, stereo matching technique can recover disparity, motion, or determine other structure feature correspondences such as lines or curves.

However, a number of interesting applications need to obtain reliable corresponding features over a *wide* baseline, such as view morphing [73] and reconstruction from multiple images [84, 41, 26]. In these applications, the external and internal parameters between any two views are significantly different, and illumination may also be significantly different. Therefore, the corresponding features in two wide baseline frames cannot be determined effectively using the traditional matching or tracking methods due to the large geometric transformation and illumination changes. Figure 3.1 shows two sequences of wide baseline frames at different viewing angles. Even though point correspondences between consecutive frames (e.g. frame 0 and 1) may be easily determined, it is a challenging problem to fully and automatically recover correspondences between non-consecutive frames (e.g. frame 0 and 6). This is the focus of this chapter.

In this chapter, we present a new approach to obtain a number of reliable corresponding points. First, an edge-corner detector is used to determine affine invariant edge-corners. These corners are located at the intersection of two or more edges. Second, after decomposing the affine matrix, we found that the affine matching space between two corners can be approximately divided into two independent spaces by rotation angle and scaling factor. We use a two-stage affine matching algorithm to obtain robust matches over these two frames to recover the geometric transformation and illumination changes between corresponding points. In the first stage, we explicitly search these two spaces to obtain a good initial estimate using the gradient descent minimization. In the second stage, the residue between two corners can be minimized by using Newton-Raphson iteration from this initial estimate. Third, we use the robust fundamental matrix estimation [113, 26, 41] to eliminate the outliers and estimate the initial epipolar geometry. Fourth, using the initial epipolar geometry constraint and the common motion constraint, we increase and refine the matches in the images, which can efficiently eliminate false matches due to the self-similarities. Finally, we reestimate the fundamental matrix and get more robust corresponding points, and generate a series of virtual views to synthesize a 3D virtual scene by employing image based rendering techniques.

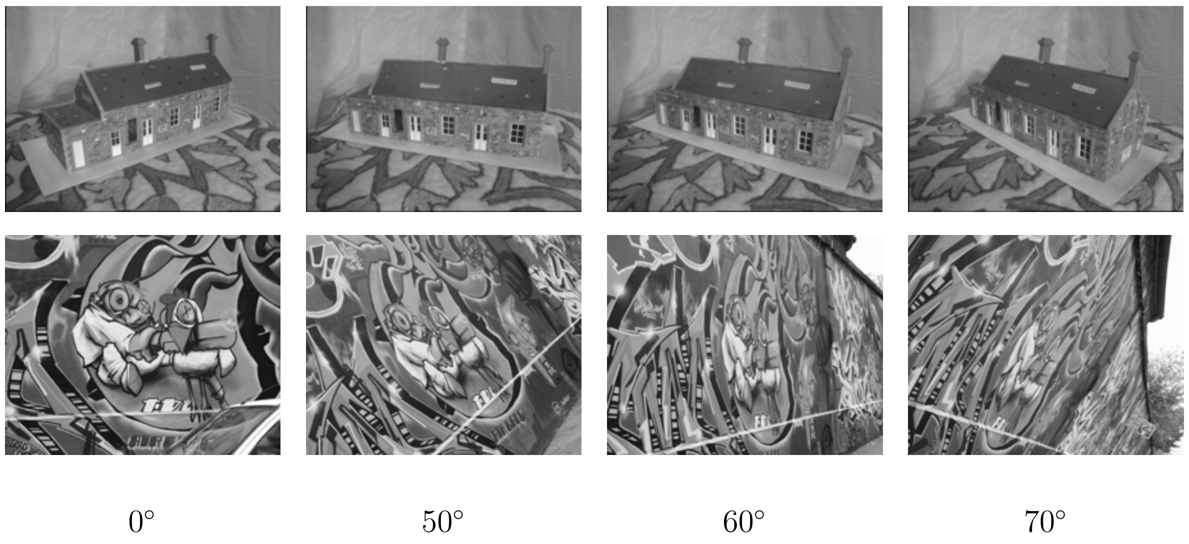
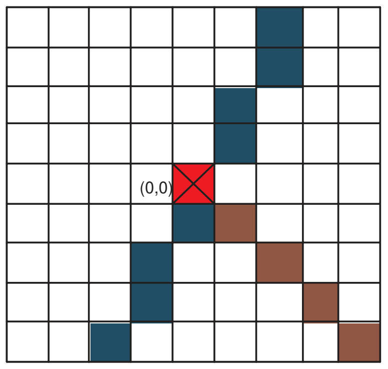
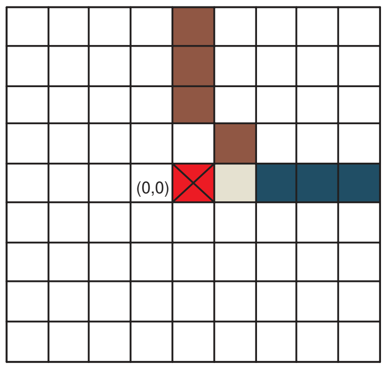


Figure 3.1: Wide baseline frames. Top: the “House” sequence from Oxford University are 3D views. Bottom: the “Graffiti-6” sequence from INRIA are planar views (Homography). The corresponding viewing angle is indicated below each image.

Section 3.1 introduces edge-corners and describes how to determine edge-corners using a local Hough transform. Section 3.2 gives a complete mathematical analysis for affine invariant corner matching, then shows how to get the reliable matches using our two-stage matching method based on SVD of the affine matrix. Section 3.3 illustrates how to use the initial epipolar geometry and common motion constraint to increase and refine the correspondences. Finally, in Section 3.4, we demonstrate several results obtained by our approach, and show that using our corresponding points and the estimated epipolar geometry, we can efficiently synthesize a 3D virtual scene from two wide baseline images.





(a) (b)

Figure 3.2: The red pixel '$\times$' is an edge pixel (a) or a pixel neighboring edge pixels (b). It is located at the origin of a $n \times n$ window (n is chosen from $20 \sim 30$). The edge passing through the origin can be quickly detected.

3.1 EDGE-CORNERS

A corner is a useful feature for motion correspondence and stereo matching. The common Harris corner detector uses a corner operator,

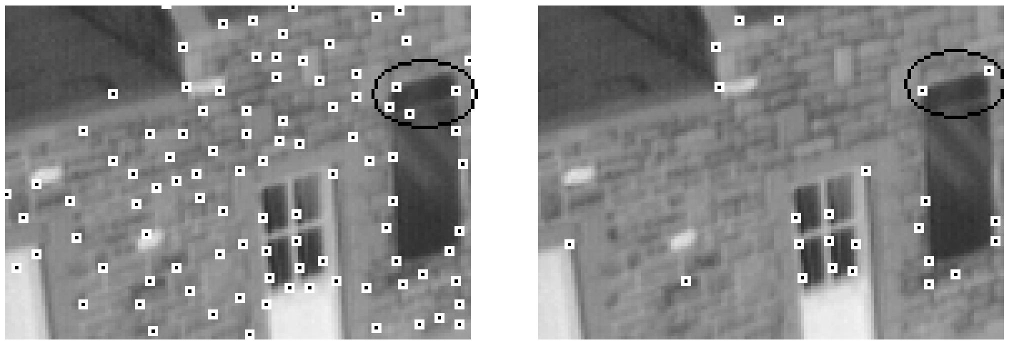
$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}, \tag{3.1}$$

over a window around a pixel, where C is the second moment matrix. This operator detects points where the image gradients, I_x and I_y, change significantly along the horizontal and vertical directions.

Rangarajan, et. al. defined a corner as the junction of two or more straight line edges, and determined an optimal function to detect the corner [69]. Smith and Brady used non-linear filtering to obtain a fast, noise resistant corner over an image [70]. Recently, Shen and Wang used a modified Hough transform to dynamically detect edges and corners [86]. They used a new dynamic edge detector to simplify line detection and avoid rounding problems in a local window. However, the edge detector is not robust when it is applied to real images. In our experiments, we found that the existing corner detectors usually do not provide consistent and reliable results when the motion gap between two frames is large.

In order to obtain a consistent and affine invariant corner over a wide baseline gap, we combine the Canny edge detector, the local Hough transform, and the Harris corner operator together to design a new matching primitive, the edge-corner. One important property of an edge-corner is that the corner is guaranteed to be located at the intersection of multiple edges. Since the intersection of edges is projected as an edge intersection under perspective projection (except in cases when the viewpoint is located on the plane containing these edges), the edge-corner is a consistent, affine invariant primitive for accurate matching over wide baseline frames. Our edge-corner detector is implemented using the following steps.

First, we use the Canny edge detector to get the edges. Then, we select each edge pixel and its neighboring pixels as corner candidates. The neighboring pixels are selected to avoid the corners missed by the Canny edge detector (or due to rounding problems). Figure 3.2.b shows that even though the pixel marked with '$\times$' is not lying on the edge, it still has a chance to become an edge corner.



(*a*) (*b*)

Figure 3.3: Corners in house frame 0. (*a*) corners detected by Harris corner detector. (*b*) edge-corners detected by our method.

Next, for each candidate, we consider an $n \times n$ window around this pixel, and set the origin at the window's center to build a local coordinate system (Figure 3.2). Every straight line across the origin has a fixed angle, which can be denoted as

$$\theta = \tan^{-1}(y/x).$$

The space of parameter θ can be quantized from $[-\pi/2,\pi/2]$ by $2(n-1)$ (half of the number of boundary points). For each interval θ_i, if the number of edge points on the line through the origin at angle θ_i is greater than a threshold, there is an edge with angle θ_i passing through the corner candidate. If the number of edges passing through this origin is more than one, this point becomes a potential corner.

Third, we select a threshold λ and use the classic corner operator (Eq.3.1) to evaluate the goodness of this corner. A non-maxima suppression method is applied in a small window area to find the best corner candidates. Figure 3.3 compares Harris corners with edge-corners for “House” frames. Our edge-corners are more precisely located on the junction of edges than Harris corners, and some poor Harris corners are eliminated, which can speed up the matching process in the next step.

3.2 CORNER MATCHING

In this section, we first derive the affine matching equation. Next, we show that our matching scheme with the affine matching equation can obtain stable matches over wide baseline images, which do not contain significant inplane rotation or scaling. Furthermore, we show that using affine matrix decomposition, the affine matching space between two corners can be approximately divided into two independent spaces by the rotation angle and the scaling factor. Then, a two-stage affine matching algorithm is used to overcome the scaling and inplane rotation limitations and determine reliable matches for general wide baseline images.

3.2.1 AFFINE MATCHING

Since our edge-corners are affine invariant, the matching between corresponding points can be represented as:

$$I_2(A\mathrm{x} + d) = I_1(\mathrm{x}), \tag{3.2}$$

where I_1 and I_2 are two original images, A is a 2D matrix and d is the translation vector. Eq.3.2 uses affine motion to model the geometric transformation between two image patches (or corners). However, in wide baseline images, the illumination may also change significantly. In order to eliminate the illumination effect, we modify Eq.3.2 and obtain

$$\mu I_2(A\mathrm{x} + d) + \delta = I_1(\mathrm{x}), \tag{3.3}$$

where μ depends on the reflection angle of the light source, and δ depends on the camera gain. We can compute the best match by minimizing the residual

$$\epsilon = \sum_{W} [(\mu I_2(A\mathrm{x} + d) + \delta) - I_1(\mathrm{x})]^2, \tag{3.4}$$

where W is the image patch. This function can be minimized by using Newton-Raphson iteration starting from $A = \mathrm{I}$ (identity matrix), $d = 0$, $\mu = 1$, and $\delta = 0$.

In wide baseline matching, the search process is more difficult than the small baseline frames. Figure 3.4.a shows a traditional searching scheme, which starts from position of p_{1i} following the initial gradient descent direction. After encountering a local minimum or a corner (a corner may be a kind of local minimum), the algorithm will stop and usually can not jump out of the trap by using Newton-Raphson iteration. Since the displacement between two corresponding points in the wide baseline frames can be up to $100 \sim 400$ pixels, there may be many local minima located in this large search area. As a result, the traditional searching scheme usually cannot obtain the correct solution even for a simple translation or looming case as shown in Figure 3.5.a and Figure 3.6.a (we used the KLT code of [85] with a large search range. The code is available online).

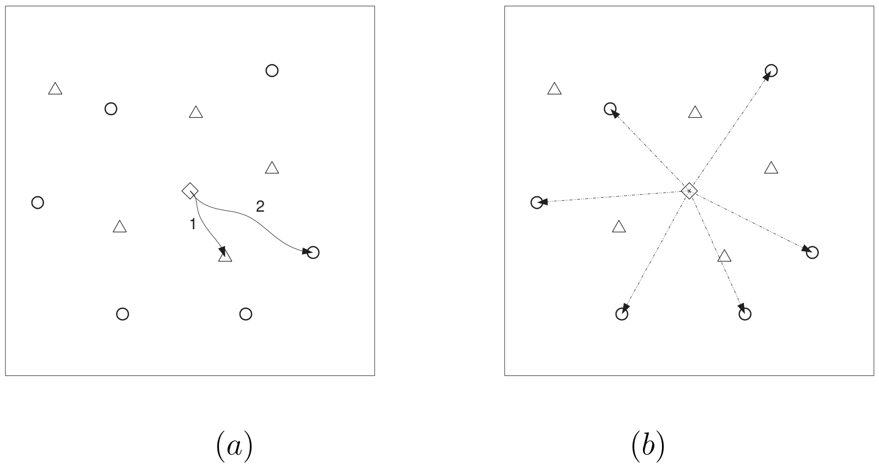
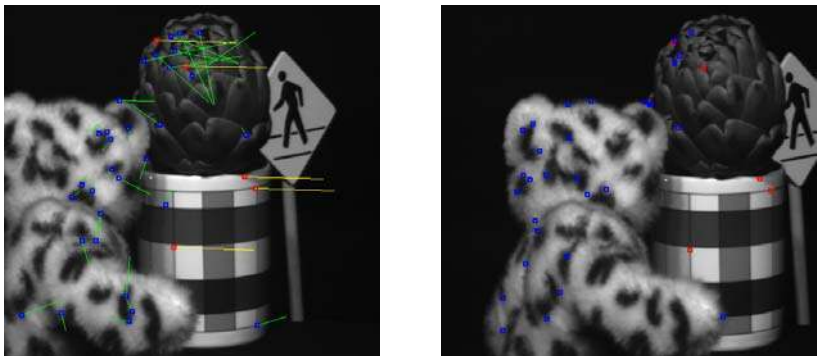


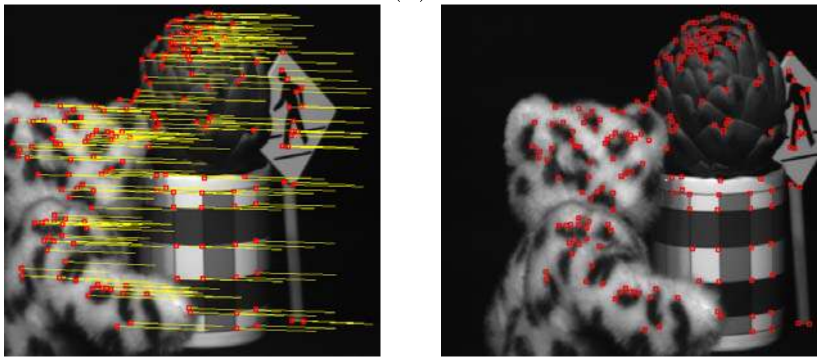
Figure 3.4: Two corner matching schemes. Point '$\diamond$' is the starting position of p_{1i}, which is located at the center of a search area (big square). Points '$\triangle$' are the local minima. Points '$\circ$' are the corner points p_{2j}. (a) Search by traditional gradient descent scheme. The search may be stalled at some local minimum such as 1 and 2. (b) corner matching scheme, which directly compares the corners p_{2j} with p_{1i} in the search area.

We use our corner matching scheme (Figure 3.4.b) to overcome this problem by employing the corner matching property, which guarantees that the match only happens between the corners. In this scheme, we only compare the matches between edge-corners, which can efficiently reduce the search space and avoid the trapping problem due to the local minima.

In our approach, we initialize the translation $d = 0$ in Eq.3.3 and set the origin of the corner's window at p_{1i} in I_1 and p_{2j} in I_2 respectively. The new equation is $\mathrm{x}_2 = A\mathrm{x}_1 + d$, where x_1 and x_2 are the local coordinates of the corners. Next, we apply this scheme using Eq.3.3. Since our edge-corner is consistent in most cases, we add a constraint $d \leq 2$ pixels

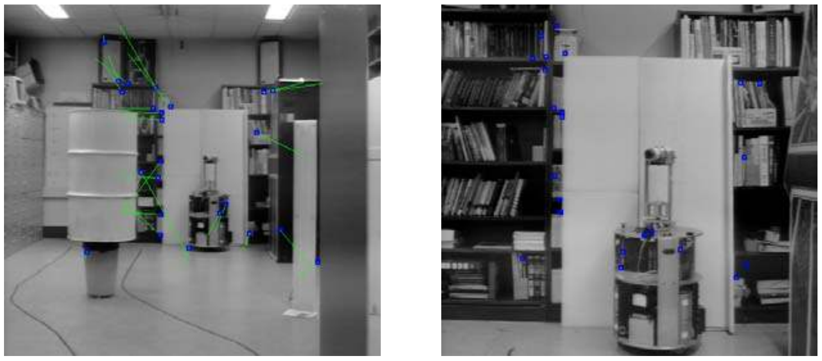


(*a*)

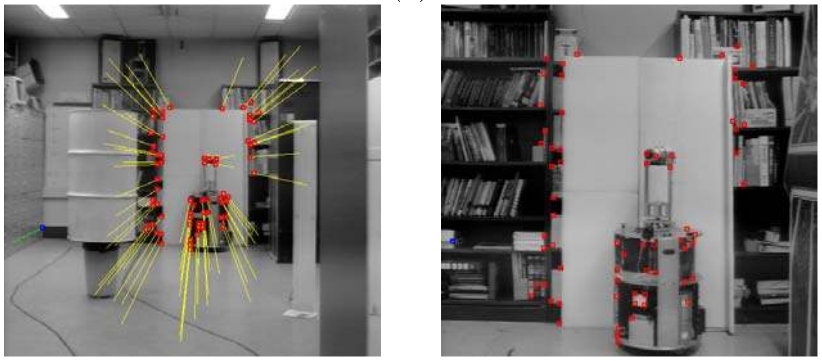


(*b*)

Figure 3.5: Comparison of KLT algorithm and the proposed corner matching scheme for translation case. (*a*) shows the results obtained by using KLT code. There are 5 correct matches in (*a*)(frame 0 and 100 of "Artichoke" from CMU). (*b*) shows the results by using the proposed scheme. (*b*) shows 184 correct inliers for this translation case (*a*), the RMS error is 0.277. Blue points are incorrect inliers, but are consistent with the estimated fundamental matrix. In our results, the outliers have been eliminated by the robust fundamental matrix estimation.



(a)



(b)

Figure 3.6: Comparison of KLT algorithm and the corner matching scheme for looming case. (a) shows the results obtained by using KLT code. There are no correct matches in (a) (frame 0 and 40 of "Lab" from CMU). (b) shows the results by using the proposed scheme. (b) shows 67 correct inliers for the looming case (a), the RMS error is 0.435. Red points are correct inliers. Blue points are incorrect inliers, but are consistent with the estimated fundamental matrix.

and only allow a small translation of p_{2j}, which can effectively avoid the serious divergence problem in the Newton-Raphson iteration. After the iteration, the corner p_{2j} may move to a

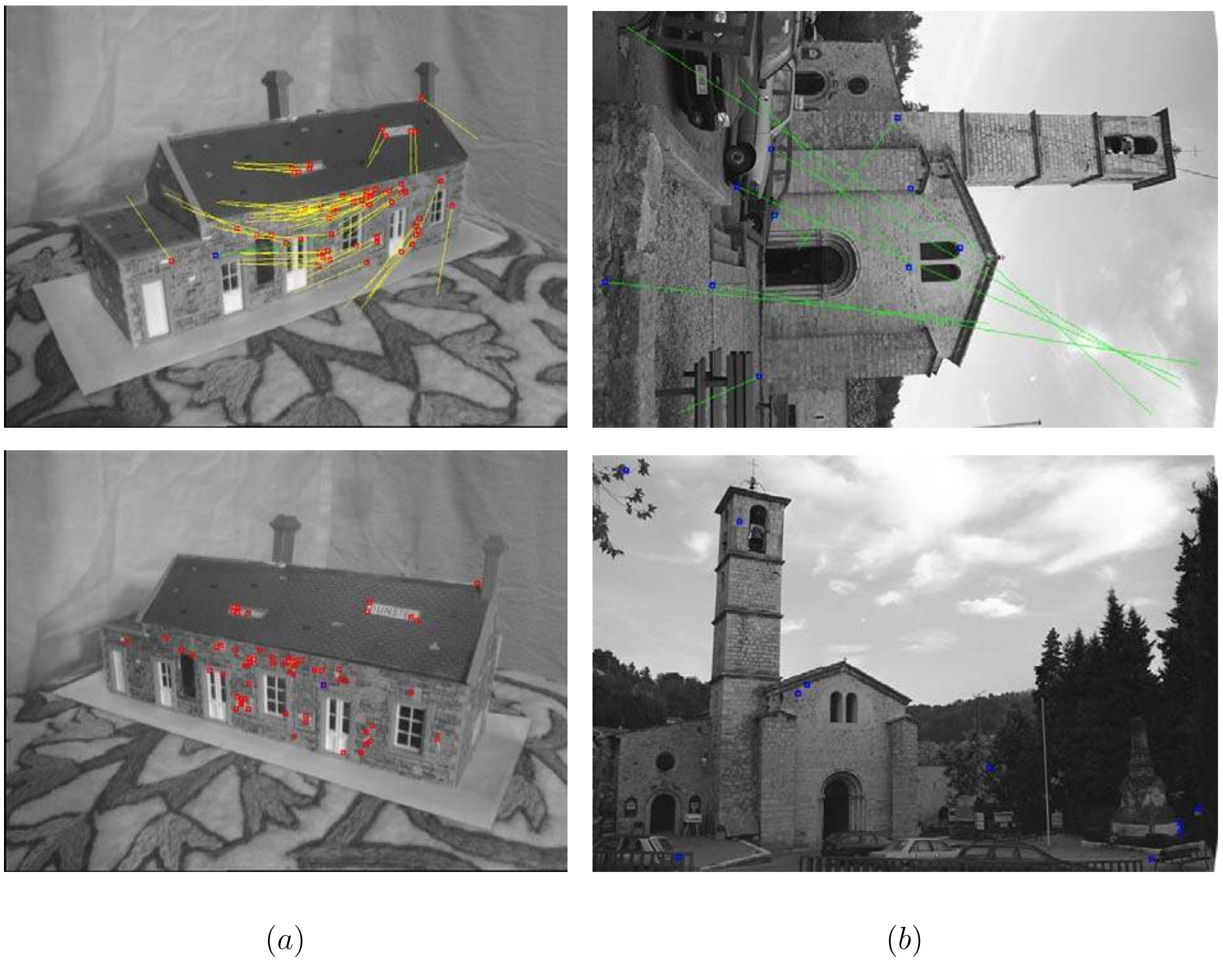


Figure 3.7: The corner matching scheme for two large rotation cases. (a) (frame 0 and 6 of “House”) shows 59 correct inliers located on the house. No correct correspondences are recovered on the ground due to the in-plane rotation. (b) (image 3 and 8 of “Valbonne” from Oxford Univ.) shows that no correct inliers are obtained. Red points are correct inliers. Blue points are incorrect inliers, but are consistent with the estimated fundamental matrix. In our results, the outliers have been eliminated by the robust fundamental matrix estimation.

new position p'_{2j}, where $d = |p'_{2j} - p_{2j}| \leq 2$. The residue ϵ_{1i2j}(Eq.3.4) is used to evaluate the goodness of the match. After searching the whole area, we consider the corner p'_{2j} with the smallest ϵ_{1i2j} as the best match for corner p_{1i}. Third, we use the robust fundamental matrix estimation to eliminate the outliers in these matches.

In this chapter, we use RMS (Root-Mean-Squared) distance error (measured in pixels) to evaluate our results [41].

$$\epsilon_{RMS} = \sqrt{\left(\frac{1}{2n}\sum_{i=1}^{n}\frac{d(m_{2i}, Fm_{1i})^2 + d(m_{1i}, F^T m_{2i})^2}{2}\right)}, \tag{3.5}$$

where m_{1i} and m_{2i} are corresponding points, F is the estimated fundamental matrix.

Figure 3.5 and Figure 3.6 show that our method works very well if the major motion of camera is translation (Figure 3.5.b) or looming ($\leq 2\times$ scaling) (Figure 3.6.b).

However, in general, the camera motion between wide baseline frames is not restricted to translation and looming, but may be combined with some significant rotation. Part of the 3D rotation of the camera (R_x or R_y) may be converted to stretch-shearing or translation after projection, and the rotation component around the Z axis, R_z, remains as the in-plane rotation. In these complicated cases, this scheme cannot accurately recover point correspondences. Figure 3.7.a and b show that the correct matches cannot be obtained if there are significant in-plane rotations.

3.2.2 AFFINE MATRIX DECOMPOSITION

The reason for the above problems is that we minimize the error ϵ (Eq.3.4) using only the first order Taylor expansion and employing the Newton-Raphson iteration method. This method is sensitive to the initial state, which is usually set as $A = \mathrm{I}$, $d = 0$, $\mu = 1$, and $\delta = 0$. This initial state is only correct for a small in-plane rotation and scaling. Therefore, if the initial state is not correct, the search will follow the wrong initial gradient descent direction and stop at some local minimum, or will diverge.

In order to find a reasonable initial state for the Newton-Raphson iteration, we decompose the affine matrix A into three components, rotation matrix R, scaling matrix S, and stretch-shearing matrix E the Singular Value Decomposition (SVD).

$$\begin{aligned}
A &= UDV' = U(V'V)DV = (UV')(VDV') \\
&= \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} v_1 & v_h \\ v_h & v_2 \end{bmatrix} \\
&= \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \kappa & 0 \\ 0 & \kappa \end{bmatrix} \begin{bmatrix} \frac{1}{\kappa} & 0 \\ 0 & \frac{1}{\kappa} \end{bmatrix} \begin{bmatrix} v_1 & v_h \\ v_h & v_2 \end{bmatrix} \\
&= \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \kappa & 0 \\ 0 & \kappa \end{bmatrix} \begin{bmatrix} v_1/\kappa & v_h \\ v_h & v_2/\kappa \end{bmatrix} \\
&= R(\alpha)S(\kappa)E, \qquad (3.6)
\end{aligned}$$

where U and V are the orthogonal matrices, D is a diagonal matrix, and E is a positive symmetric matrix. In 2D space, U, V and $R(\alpha)$ are represented as rotation matrices, $S(\kappa)$ is a scaling matrix, E is represented as a stretch-shearing matrix.

After decomposing the affine matrix, it is clear that this affine matrix implies a non-linear rotation component, $R(\alpha)$, which depends on, R_z, one component of the 3D camera rotation. The scaling matrix $S(\kappa)$ depends on the focal length (zooming) and 3D translation component T_z (looming) of the camera. Compared with $R(\alpha)$ and $S(\kappa)$, the matrix E is complicated, which may be due to R_x, and R_y. Therefore, if the viewing angle is large or if there is some significant scaling between two images, it is difficult to obtain the correct convergence by using the linear Taylor expansion with the Newton-Raphson iteration from the initial state, $A = \mathrm{I}$, $d = 0$, $\mu = 1$, and $\delta = 0$. Figure 3.8.c shows the image patches with a significant rotation or scaling that do not converge to the correct solution using our preliminary corner matching scheme.

In order to obtain a reasonable initial state, we simplify the affine matching matrix

$$A \approx R(\alpha) S(\kappa).$$

As a result, this affine matching space can be divided into two independent spaces, rotation and scaling. The rotation angle α and scaling factor κ can easily be quantized into different ranges.

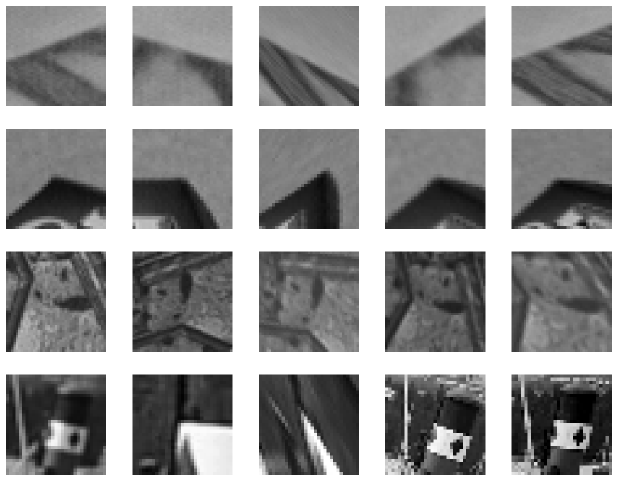


Figure 3.8: Matching procedures for two corresponding corners (30×30). The first row: rotation case, the corners from "House" (Figure 3.7.a). The second row: rotation case, the corners from "Car" (Figure 3.14.b). The third row: rotation case, the corners from "Valbonne" (Figure 3.7.b). The last row: scaling case, the corners from 'Boat" (Figure 3.12.a). (a) and (b) are the windows around the original corners in two frames. (c) is the warped version of (b) using the corner matching scheme. (d) is the warped version of (b) after finding the best rotation and scaling in the first matching stage. (e) is the final result after using our two-stage matching algorithm. It is clear that our method maintains the appearance (compare (a) and (e)) of the patch over two wide baseline frames, and provides an accurate matching for the corresponding edge-corners.

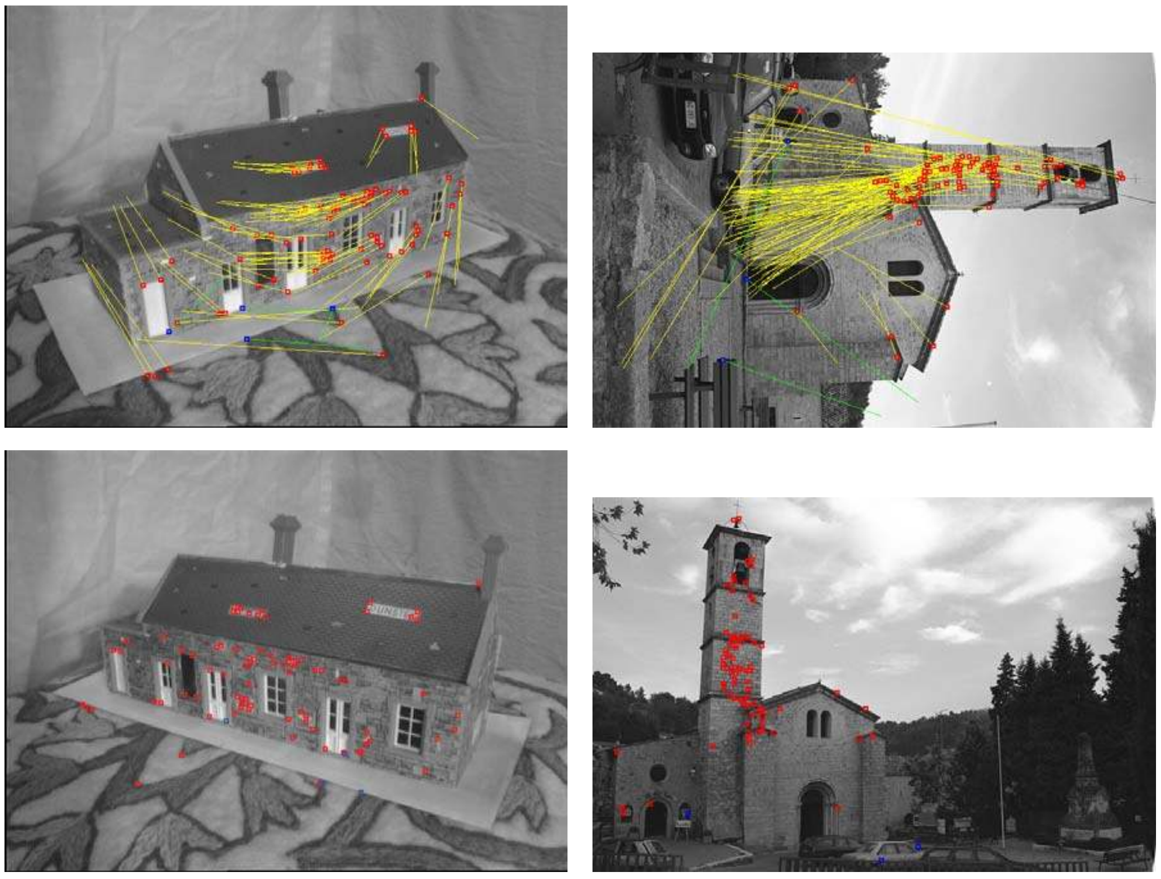
3.2.3 TWO-STAGE MATCHING

Using this quantization of the affine matrix, we design a two-stage algorithm for corner matching on a general wide baseline image pair.

The first stage is to find a reasonable initial state for the Newton-Raphson iteration. For any corner p_{1i} in I_1, we select p_{2j} from I_2 in a large search area. Then, we apply a rotation-scaling warping $R(\alpha)S(\kappa)$ to a window around p_{2j} and compare this with a window around p_{1i} in I_1 by using SSD. We search for a minimal image residual by varying α from α_l to α_h. At each rotation angle, we use five different ranges of κ for the scaling matrix S, which are 4, 2, 1, 0.5 and 0.25. In our implementation, we use an N-split tree (large step of 20° and small step of 4°) to speed up the search between α_l and α_h.

After computing the best estimates of matrices $R(\alpha)$ and $S(\kappa)$, we initialize the affine matrix $A = R(\alpha)S(\kappa)$ in Eq.3.3, which provides a reasonable gradient descent direction for the Newton-Raphson iteration. Then, following this gradient descent direction, the correct affine matching matrix A between two corners is determined and the residual is minimized.

There are three advantages of our approach. First, we only compare corners with corners instead of searching every pixel in a large area, that may have several local minima. Second, the first step of our matching method quantizes affine space by using the rotation angle and scaling factor. It avoids the computation of non-linear components $R(\alpha)$ and $S(\kappa)$ in terms of gradients I_x and I_y, but recovers an approximate state of the geometric deformation due to the large rotation and scaling. Third, based on the reasonable initial value of $A = R(\alpha)S(\kappa)$,



(a) (b)

Figure 3.9: Results obtained by using our two-stage matching scheme. (a) shows 83 correct inliers, several of them are on the ground. (b) shows 78 correct inliers.

the second step follows the correct initial gradient descent direction and quickly finds an optimized solution of A between p_{1i} and p_{2j}, which efficiently avoids the local minima.

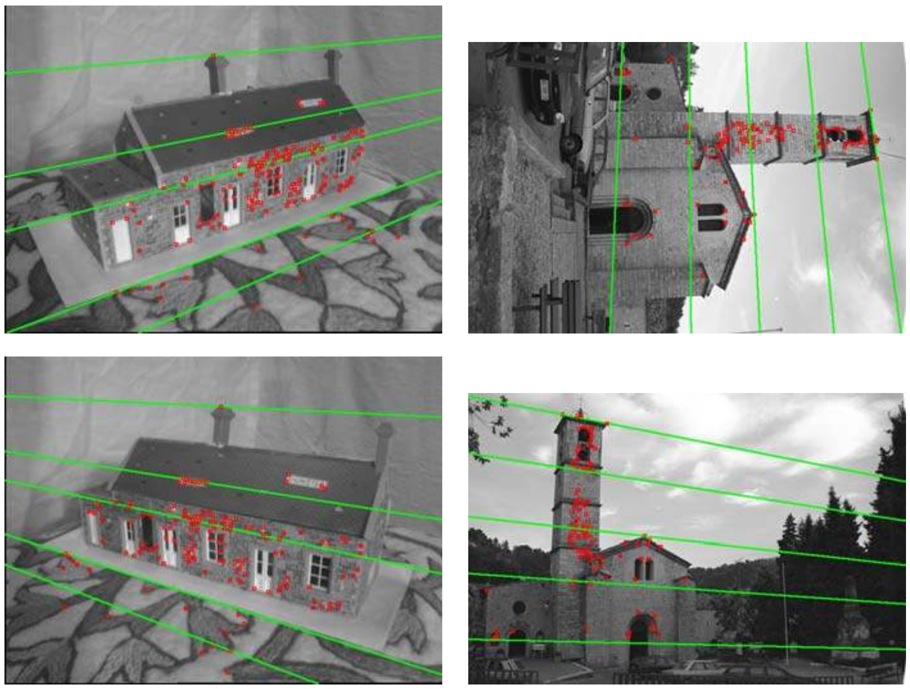
Compared to conventional scale-space matching [5, 61], which only changes the scaling factor, our method explicitly changes two independent factors, α and κ, which can effectively approximate and quantize the affine matching space, and provide a reasonable initial state for the Newton-Raphson iteration. Figure 3.8 compares the results by using our two matching schemes, and illustrates the detailed steps of our matching procedure. It is clear that a correct initial state is required to obtain a reliable optimized solution for correspondences matching. Figure 3.9 shows that using the two-stage matching algorithm, we can overcome the in-plane rotation problem and recover more reliable correspondences.

3.3 CORRESPONDENCES REFINEMENT

Even though we obtain many reliable correspondences over two images by using our two-stage matching algorithm, some correspondences cannot be determined due to the self-similar structures and some false matches. These false matches are consistent with the estimated fundamental matrix and cannot be eliminated by using the robust fundamental matrix algorithm (blue points in Figure 3.9). We propose a new approach to increase the correspondences and eliminate these false matches by the estimated epipolar geometry and local motion vector.

After robust fundamental matrix estimation, the epipolar geometry can be obtained. Let F denote the fundamental matrix, the epipolar line of p_{1i} on the second image is $l_{1i} = Fp_{1i}$. The point corresponding to p_{1i} should be located around this epipolar line on I_2. Therefore, for each point p_{2j}, we check the distance d_{1i2j}, such that

$$d_{1i2j} = \sqrt{(p_{2j}^T F p_{1i})^2 \left(\frac{1}{(Fp_{1i})_1^2 + (Fp_{1i})_2^2} \right)}. \tag{3.7}$$



(a) (b)

Figure 3.10: Refined results obtained by our approach. (a) shows 150 correct inliers, and the RMS error is 0.418. (b) shows 102 correct inliers, the RMS error is 0.632. Several green corresponding epipolar lines are drawn on the images.

If the distance is less than a threshold ξ ($\xi \in (2 \sim 4$ pixels$)$), we use the two-stage matching algorithm to reestimate the residue ϵ_{1i2j} between these two points p_{1i} and p_{2j} in the two images. Since the probability of self-similar corners is dramatically reduced within a small band around the epipolar line, this method can efficiently increase the matches, which may be missed due to self-similarity.

However, it cannot eliminate the false matches that are consistent with the fundamental matrix F. In order to eliminate these false matches, we assume that local motion should be consistent. Hence, the motion between a pair of correct correspondences should be similar to the neighboring corresponding pairs; this is called *the common motion constraint*. For each pair of correspondences m_{1i} and m_{2i}, the motion vector $\zeta_i = m_{1i} - m_{2i}$. We select the neighboring corresponding pairs close to m_{1i} in an adaptive window N to compute the average local motion $\overline{\zeta}$. After checking the direction and magnitude differences between ζ_i with $\overline{\zeta}$, we eliminate the matches, which are not consistent with the local motion. Figure 3.10 shows the final matching results obtained by our approach, where more corresponding points are determined.

3.4 EXPERIMENTAL RESULTS

In this section, we demonstrate our results on wide baseline matching. All of our experiments used gray scale images to compute correspondences, and most of the images are from public domain resources (CMU, Oxford, OSU, INRIA, etc). Then, we illustrate several view morphing examples, where the epipolar geometries are estimated by the proposed method, to show that our matching algorithm is effective and stable.

Figure 3.11 shows the results for two sequences of real scenes (shown in Figure 3.1), where the viewing angle ranges from $0°$ to $70°$. The RMS distance error increases as the

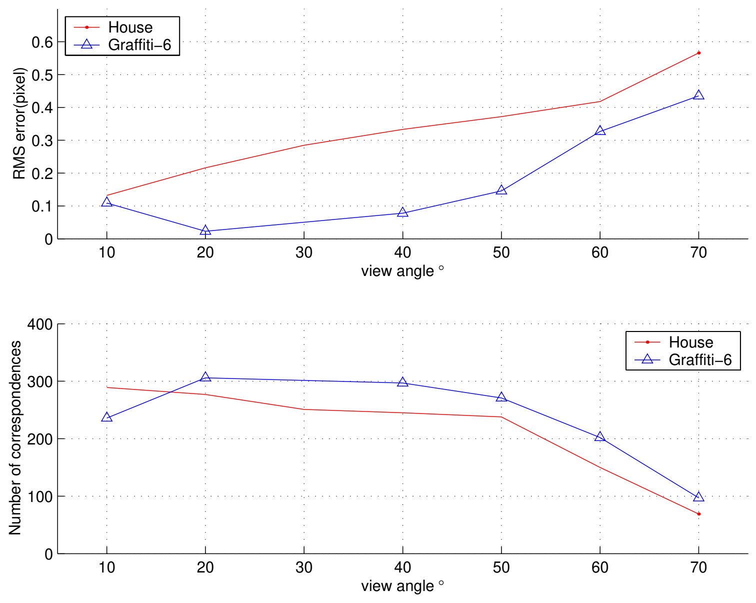


Figure 3.11: RMS error and number of corresponding points of "House" and "Graffiti-6" sequences under different viewing angles.

angle increases, but is less than 0.6 pixel. After the viewing angle reaches 60°, the number of correct corresponding points decreases due to the severe occlusion in the "House" sequence and shrinking in the "Grafitti-6" sequence.

Figure 3.12 and Figure 3.13 show the matching results for the different scenes, which include several kinds of significant camera motions (rotation, translation, or scaling) with some severe occlusion, illumination changes, and the presence of self-similar structures. Figure 3.12.*a* and *b* show significant scaling (about 4×) between the two images. Figure 3.12.*c*, Figure 3.13.*a* and *b* show non-inplane rotation angles over 50°. Figure 3.13.*c* shows severe occlusions in the scene.

Table 3.1: The comparison of Affine-Harris method with our approach.

Name (Frame No)	**A-H**	**T-S (RMS)**	**RF (RMS)**
Graffiti-5 (4,8)	33	276 (0.110)	299 (0.127)
Graffiti-6 (1,5)	27	135 (0.341)	202 (0.327)
Boat (0,5)	22	163 (0.301)	175 (0.560)
Valbonne (5,14)	14	57 (0.665)	185 (0.658)
Valbonne (9,13)	22	155 (0.486)	194 (0.661)
UBC (8,10)	34	184 (0.266)	227 (0.242)

We also tested a number of image pairs from INRIA and compared our results with the Affine-Harris method of [61] in Table 3.1, which compares the correspondence number and RMS obtained by using different methods. A-H are the results using the Affine-Harris method. T-S are the results using two-stage matching without refinement. RF are the results after increasing correspondences using epipolar geometry. In all of the test cases, our method was able to determine more corresponding points than the Affine-Harris method even without the refinement step.

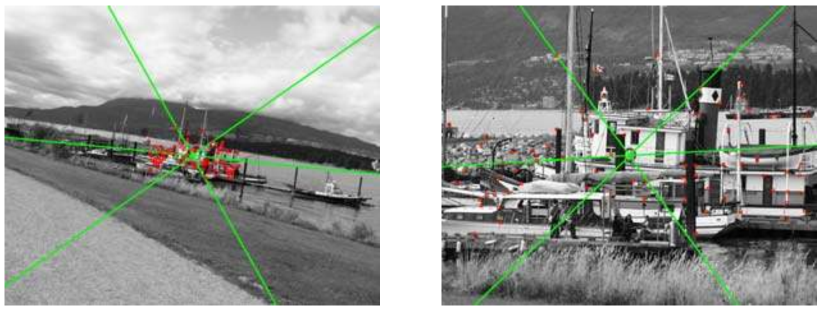
Figure 3.14 shows three examples of view synthesis results using our estimated matches as well as view morphing techniques. After automatically determining corresponding points between two original wide baseline images, we rectified the original images to parallel views and morphed them using a linear ratio. Then, using a 5-point postwarping algorithm [110], we postwarped the blended images into the final view position. As a result, a series of virtual views were rendered following the baseline between two original cameras, providing a realistic 3D visual effect from a static scene.

Note: All of these results are available on our web site [39].

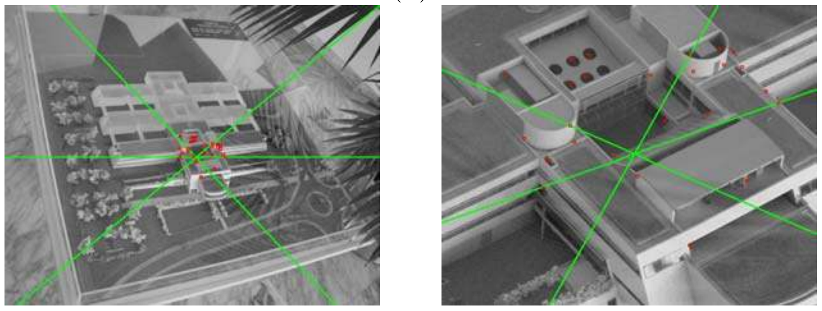
3.5 SUMMARY

In this chapter, we successfully solved the problem of how to obtain reliable corresponding points over two wide baseline frames. First, a number of affine-invariant edge-corners are detected in both images. Then, based on singular value decomposition of the affine matrix, we found that the affine matching space between two corners can be approximately divided into two independent spaces by rotation angle and scaling factor. Using this property of the affine matrix, we designed a novel two-stage matching algorithm to determine the robust matches between edge-corners, which effectively overcome the significant affine transformation in the wide baseline images. Moreover, employing the estimated epipolar geometry and common motion constraint, we efficiently refined and increased the matches. Finally, we demonstrated that a series of virtual views can be correctly synthesized using our correspondences and estimated epipolar geometries.

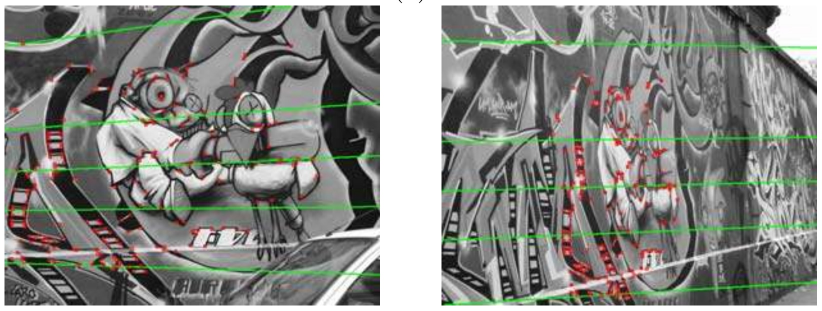
We have tested a number of wide baseline image pairs under different severe camera motions with illumination changes, occlusions, and self-similarities, and have obtained excellent results for all of these cases.



(*a*)

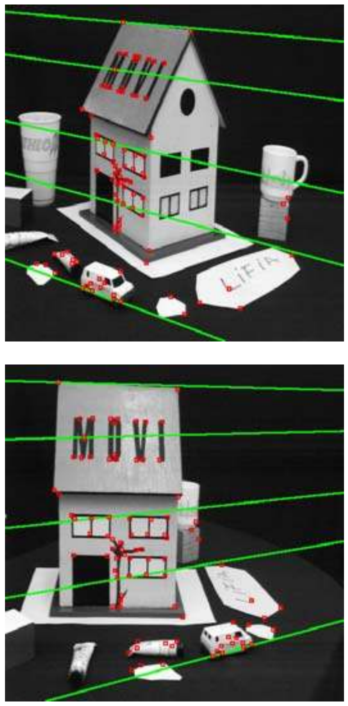


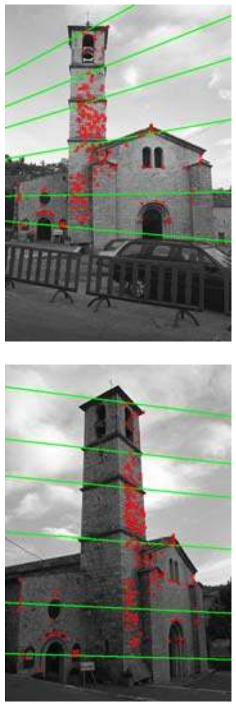
(*b*)



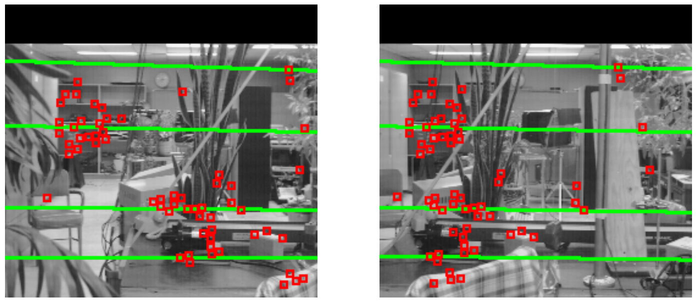
(*c*)

Figure 3.12: Final matching results by our algorithm. (*a*) (frame 8 and 0 of "Boat" from INRIA.) shows 75 inliers, the RMS is 0.686. (*b*) (frame 0 and 10 of "INRIA-Model" from INRIA.) shows 25 inliers, the RMS is 0.583. (*c*) (frame 1 and 5 of "Graffiti-6" from INRIA) shows 202 inliers, the RMS is 0.327. Several green corresponding epipolar lines are drawn in each pair of images.



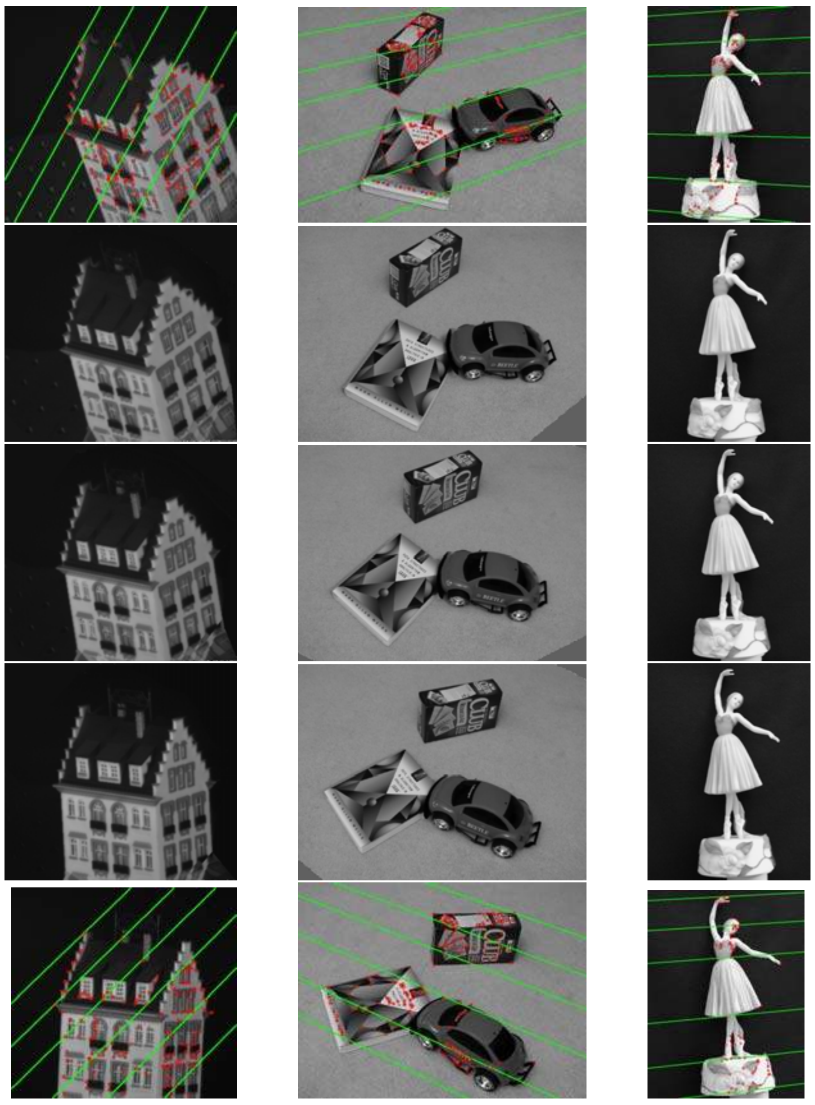


(a) (b)



(c)

Figure 3.13: Final matching results by our algorithm. (a) (frame 5 and 9 of “Movi” from OSU) shows 89 inliers, the RMS is 0.471. (b) (frame 5 and 14 of “Valbonne” from Oxford Univ.) shows 185 inliers, the RMS is 0.658. (c) (frame 1 and 90 of “SRI” from CMU.) shows 61 inliers, the RMS is 0.268. Several green corresponding epipolar lines are drawn in each pair of images.



(a) (b) (c)

Figure 3.14: View synthesis results after matching. The first and the last images are the original frames. (a) "Hotel"(frame 0 and 100 of "Hotel" from CMU), there are 131 inliers, the RMS is 0.664. (b) "Car", there are 176 inliers, the RMS is 0.519. (c) "Dancing", there are 71 inliers, the RMS is 0.752. The rest of images are virtual views synthesized using the reference images.

CHAPTER 4

MOTION SEGMENTATION

Layer based motion segmentation has been investigated by computer vision researchers for a long time [109, 46, 66, 101, 4, 99, 2, 2]. Once motion segmentation is achieved, a video sequence can be efficiently represented by different layers. The major steps of motion segmentation consist of: (1) determining the layer descriptions, which include the layer number and the motion parameters for each layer; (2) assigning each pixel in the image sequence to the corresponding layer and identifying the occluded pixels. In general, this is a hard problem since it may not be possible to segment a scene just based on 2D parametric motion as shown in Figure 4.1.c. Hence, this kind of scene cannot be simply represented by planar layers and it may be necessary to compute the full disparity map using a stereo algorithm. However, in the motion segmentation area, we assume that the pixels in a given video can be partitioned into several layers, and each layer can share a single 2D motion, such as affine or projective transformation, as shown in Figure 4.1.b.

Using this assumption, several different approaches have been proposed to solve this problem. The typical motion segmentation methods employ: optical flow with K-mean

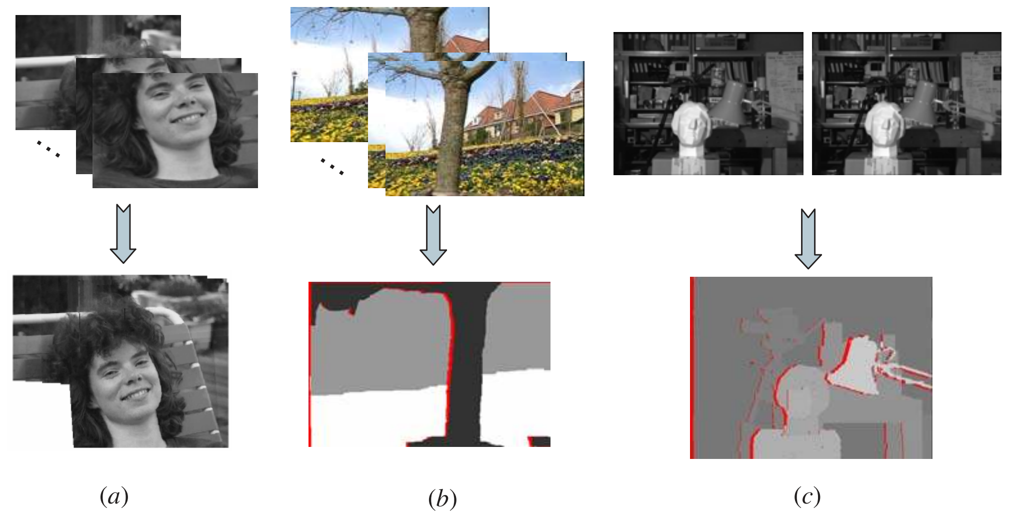


Figure 4.1: Depending on the complexity of a scene, it can be represented by one layer (*a*. image registration or mosaic), multiple layers (*b*. motion segmentation), or disparities (*c*. stereo).

clustering [99], Expectation-Maximization (EM) framework [4], normalized graph cut [80], or linear subspace [46]. While most of the existing approaches have only focused on the layer description and representation, comparatively little work has been done to handle the occlusion problem between the overlapping layers. In contrast, the occlusion problem has been widely studied in the context of stereo algorithms [49, 51, 50]. Accurate detection of the occluded areas is important to improve the dense disparity map and the quality of 3D reconstruction. Similarly, in the motion segmentation area, this occlusion problem is essential to detect the discontinuities between the overlapping layers and improve the quality of the layer boundaries.

Another important problem in motion segmentation is how to determine layer description or layer clustering. Merging the similar motion of a set of small patches is a popular approach used in different methods such as K-mean clustering, or linear subspace. However, the motion parameters computed from a small patch usually are not reliable due to the over-fitting problem. Therefore, for layer clustering, it is essential to expand the region and detect outliers.

In this chapter, we propose a novel approach to extract accurate layer representations from a video sequence and explicitly determine occlusions between the overlapping layers. Our algorithm is implemented in two stages as shown in Figure 4.2. In the first stage, we determine seed correspondences over a short video clip (3-5 frames). Then, we gradually expand each seed region from an initial rectangular patch of fixed dimensions into an enlarged support region of an arbitrary shape to eliminate the over-fitting problem and detect the outliers. This is achieved using a graph cut approach integrated with the level set representation. After that, we employ a two-step merging process to obtain a layer description of the video clip. In the second stage, we introduce the occlusion order constraint over a multiple frame segmentation, which guarantees that the occlusion areas increase with the temporal order and effectively maintains segmentation consistency in consecutive frames. After applying this constraint on the graph cut framework, we obtain an accurate and stable video segmentation in terms of layers and their 2D motion parameters. At the same time, the occluded pixels between overlapping layers are correctly identified, which greatly improves the quality of the layer boundaries.

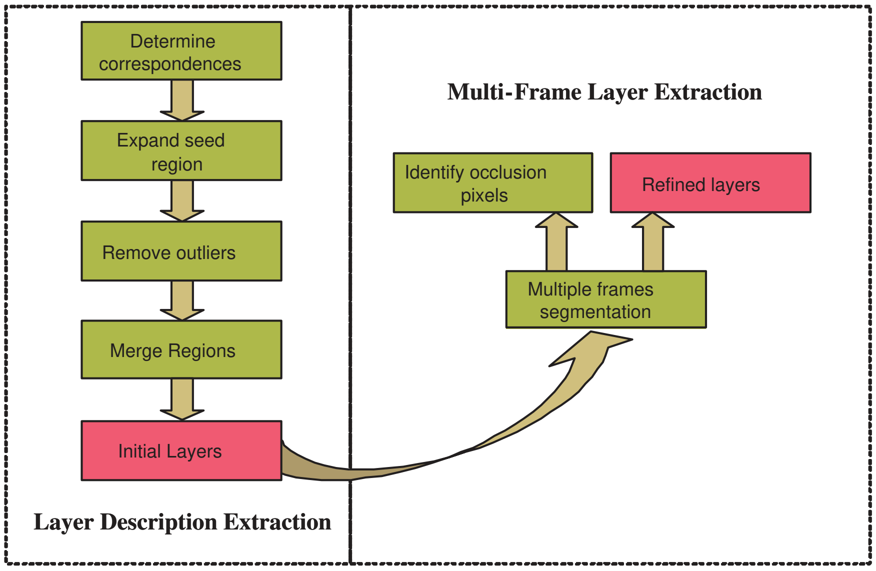

Determine correspondences
Expand seed region
Remove outliers
Merge Regions
Initial Layers
Layer Description Extraction
Multi-Frame Layer Extraction
Identify occlusion pixels
Refined layers
Multiple frames segmentation

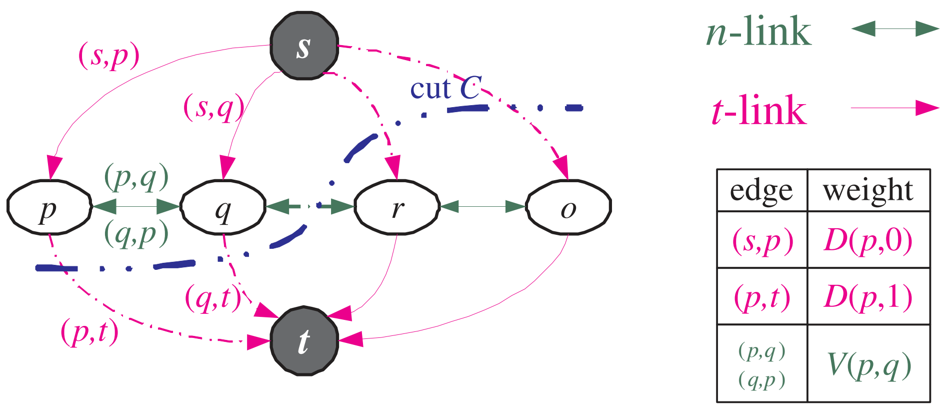
Figure 4.2: The flow chart of our algorithm.

The chapter is organized as follows. Section 4.1 reviews the previous work related to layer extraction. Section 4.2 addresses how to extract layer descriptions from short video clips. Section 4.3 deals with the use of the occlusion order constraint, three-state pixel graph, and a multi-frame graph cut algorithm for obtaining precise layer segmentation in the presence of occlusion. Finally, in Section 4.4, we demonstrate several results obtained by our method on different applications.

4.1 GRAPH CUT AND NOTATION

Recently, graph cut approaches [14, 7, 15, 51, 50] were proposed to successfully minimize energy functions for various computer vision problems, such as stereo, image segmentation, image restoration, and texture synthesis [44]. After formulating these different energy minimization problems into a graph cut framework, an approximate optimal solution can be obtained in a polynomial time. As an extension of the smoothness of the geodesic active contour approach, Boykov and Kolmogorov modified the n-neighbor system of the graph cut to simulate Riemannian metrics and reduced metrication error on 2D and 3D image restoration [8]. Xu et al. also presented an approach to refine an active contour by iteratively using the graph cut method [104]. Birchfield and C. Tomasi proposed an approach to use a graph cut framework to combine layer segmentation and stereo for the scene with slant surfaces, where the stereo disparities and discontinuities are improved by the explicit layer segmentation [13].

In the motion segmentation area, Shi and Malik first used the normalized graph cut to extract layers from a video sequence [80]. However, since they grouped pixels based on the affinity of motion profile, a local measurement, their method ignored the global constraints and appeared unstable for noisy image sequences. Wills et al. proposed the use of graph cut to extract layers between two wide baseline images [100]. After employing the RANSAC technique, they clustered the correspondences into several initial layers, then performed the dense pixel assignment via graph cut.



edge	weight
(s,p)	$D(p,0)$
(p,t)	$D(p,1)$
(p,q) (q,p)	$V(p,q)$

Figure 4.3: An example of a graph $\mathcal{G}$ for a 1D image. Nodes p, q, r, and o correspond to the pixels in the image. After computing minimum cut $\mathcal{C}$, the nodes are partitioned into supporting pixels p, q (source) and unsupporting pixels r, o (sink). The weights of the links are listed in the table on the right.

4.1.1 GRAPH CUT NOTATION

In this chapter, we use the terminology and notations similar to [15]. For example, Figure 4.3 shows a typical weighted graph with four nodes. This graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is defined by a set of nodes $\mathcal{V}$ (image pixels) and a set of directed edges $\mathcal{E}$ which connect these nodes as shown in Figure 4.3. In this graph, there are two distinct nodes s and t, called the source and sink respectively. The edges connected to the source or sink are called t-links, such as (s,p) and (p,t). The edges connected to two neighboring pixel nodes are called n-links, which are bi-directional, such as (p,q) and (q,p). The weights of these n-links in both directions may not be equal. A cut $\mathcal{C}$ is a subset of edges which separates the nodes into two parts; one part belongs to the source and the other belongs to the sink. The cost of a cut is the summation of the weights of its edges. Using a standard max-flow algorithm [3], the cost of the cut is globally minimized, and such a cut is called a minimum cut. Given a labeling system f, each pixel in an image will be assigned one label. In this chapter, we use f_p to represent the label of pixel p. $D(p, f_p)$ is the data penalty of pixel p when it is assigned a label f_p. $V(p, q)$ is a smoothness penalty function between two neighboring pixels p and q.

4.2 LAYER DESCRIPTION EXTRACTION

In our approach, the first stage is to extract the layer descriptions from the video sequence, which includes the number of layers and the motion parameters for each layer. In this stage, we first detect the robust seed correspondences over a short video clip. Then, using the shape prior of the previous seed region, the region's front is gradually propagated along the normal direction using a bi-partitioning graph-cut algorithm integrated with the level set representation. Third, we design a two-step merging process to merge the seed regions into several groups, such that each group belongs to a single motion field.

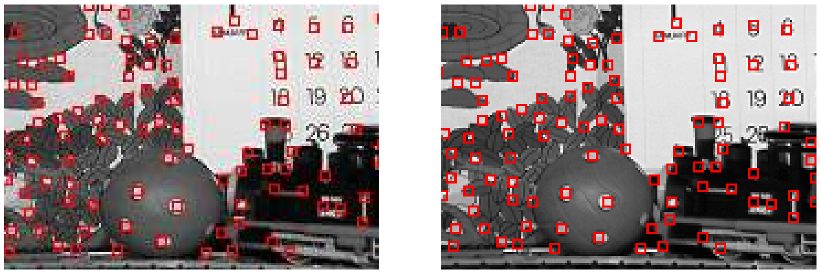


Figure 4.4: The corner tracking results for frames 1 and 5 of the mobile-calendar sequence.

4.2.1 DETERMINING SEED CORRESPONDENCES

In order to correctly extract the layer descriptions, we consider a short video clip $\mathcal{K}$ instead of only two consecutive frames. The reason is that if the motion between two consecutive frames is too small, the motion parameters between different layers are not distinct. Therefore, we use an average pixel flow $\bar{\nu}$ of the seed correspondences as a measurement to decide the number of frames in the video clip $\mathcal{K}$.

$$\bar{\nu} = \frac{1}{N} \sum_{i=0}^{N} |\nu_i|,$$

where ν_i is the pixel flow from the current frame I_n to the first frame I_1 for correspondence i, and N is the total number of the seed correspondences. If $\bar{\nu}$ between I_n and I_1 is greater than some threshold (i.e 3 pixels), the number of frames $\mathcal{K}$ is set to n.

In our approach, we detect the Harris corners in the first frame, then we use the KLT tracking algorithm [85] or our matching algorithm [107] to track the corners over this short period using a 17×17 pixel window. Compared with the KLT algorithm, our wide baseline matching approach can efficiently compensate for the rotation component between two corners by breaking the affine transformation into different parts. Figure 4.4 shows the tracking results for frames 1 to 5 of the mobile-calendar sequence using [107], where the corresponding corners with a large rotation are more accurate.

Since the Harris corners are located in the textured areas, we can obtain reliable affine transformations for the seed regions, and skip the non-textured areas, where the motion parameter estimation is unreliable.

4.2.2 EXPANDING SEED REGIONS

Once the seed correspondences are determined between frames I_1 and I_n[1], we consider a patch (17×17 window) around each seed corner as an initial layer, which corresponds to a planar patch in the scene. This way, we get a number of initial layers, and each layer is supported by a small patch with a corresponding affine transformation. Nevertheless, the affine motion parameters estimated using the small patches may over-fit the pixels inside the region, and may not correctly represent the global motion of a larger region. Particularly, when the corner is located at the boundary of two true layers, the over-fitting may introduce a serious distortion on this patch after applying the affine transformation.

One straightforward solution is to simply extend the region by including neighboring pixels which are consistent with the affine transformation. Such pixels can be determined

[1]In the rest part of this section, we refer I_1 as the first frame and I_n as the second frame for the convenience of description since our layer clustering algorithm will only use these two frames.

by applying a threshold to the SSD (Sum of Squared Differences) computed between the original and warped windows. However, this scheme has two problems: First, the resulting expanded region may not be compact and smooth. Second, the new patch may include pixels from multiple layers, and may not be consistent with a single planar patch in the scene. Figure 4.5.*b* shows one sample result obtained by using this simple scheme. The seed region is originated from the seed on the rotating ball (Figure 4.5.*a*). After expanding the boundary and bi-partitioning by applying a simple threshold, the region is not smooth, and it also includes pixels from other layers.

In order to deal with these problems, we propose a novel approach to gradually expand the seed region by identifying the correct supporting pixels by using the bi-partitioning graph cut method and employing the level set representation. We introduce a smoothness energy term, which can maintain the partitions piecewisely smooth and naturally solve the first problem. Then, using a level set representation, the contour of the seed region is gradually evolved by propagating the region's front along its normal direction, which effectively eliminates the second problem.

The process of expanding the seed region can be easily formulated into the graph cut framework [15] as a bi-partitioning problem of a node set. In this framework, we seek the labeling function f by minimizing the energy

$$E = E_{smooth}(f) + E_{data}(f) = \sum_{(p,q)\in\mathcal{N}} V(p,q) + \sum_{p\in\mathcal{P}} D(p, f_p), \tag{4.1}$$

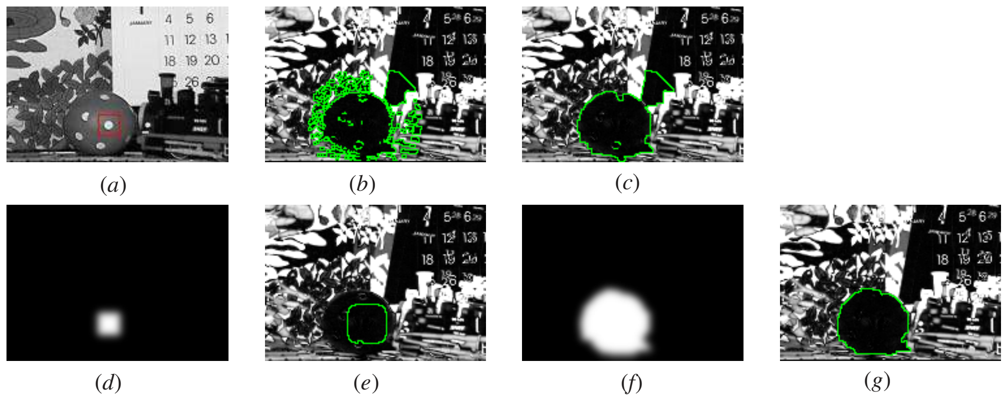


Figure 4.5: A procedure for expansion of an initial 17×17 seed region (a) to a large support region. (b) The result after simple expansion and partitioning. (c) The result after bi-partitioning without the level set representation. $(d) - (g)$ are the intermediate steps of bi-partitioning with the level set representation. (d) and (f) respectively are the expansions of the seed region during the first and fourth iterations using the level set representation. (e) and (g) are the results obtained after the graph cut partitioning, where the new region can have an arbitrary compact contour. *Note:* The red box is the initial seed region. The green contours are obtained after using bi-partitioning algorithm.

where E_{smooth} is a piecewise smoothness term, E_{data} is a data error term, $\mathcal{P}$ is the set of pixels in the image, $V(p,q)$ is a smoothness penalty function (Eq. 4.2), $D(p, f_p)$ is a data penalty function (Eq. 4.3), $\mathcal{N}$ is a 4-neighbor system, and f_p is the label of a pixel p. In this bi-partitioning problem, the label f_p of the pixel p is assigned either 0 or 1. If $f_p = 1$, the

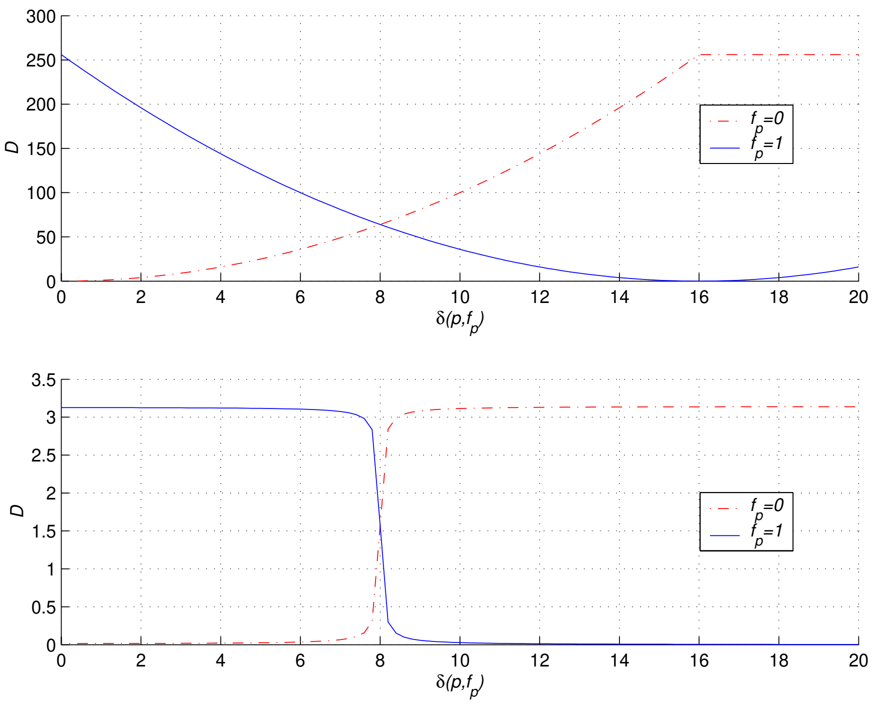


Figure 4.6: Comparison between the approximate Heaviside function and the truncated quadratic function. Top: The truncated quadratic function. Bottom: Heaviside function.

pixel p is supporting this seed region, otherwise this pixel is not supporting the region.

$$V(p,q) = \begin{cases} 3\lambda & \text{if } \max(|I_1(p) - I_1(q)|, \quad |I_n(p) - I_n(q)|) < 8, \\ \lambda & \text{otherwise,} \end{cases} \tag{4.2}$$

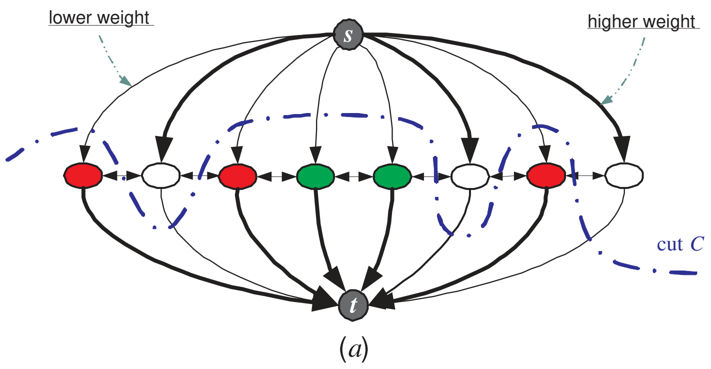
Instead of using the truncated quadratic function $D(p, f_p) = \min(|\delta(p) - c(f_p)|^2, const)$, we use an approximate Heaviside function to compute the data penalty as follows:

$$D(p, f_p) = \begin{cases} \tan^{-1}(\delta(p)^2 - \tau) + \pi/2 & \text{if } f_p = 0, \\ \pi/2 - \tan^{-1}(\delta(p)^2 - \tau) & \text{if } f_p = 1, \end{cases} \tag{4.3}$$

where $\delta(p)$ is the absolute intensity difference of pixel p between the first frame and the warped version of the second frame, $c(f_p)$ is the ideal value for layer f_p, and τ is an empirical threshold, which is set to 64 for all our experiments. If there is more noise on the images, we can slightly increase τ to compensate the noise affect. When $\delta(p) = \tau$, $D(p, 0) = D(p, 1)$ and the corresponding value $D = \pi/2$, which is called a critical value $\hat{D}$. The value of λ in Eq.4.2 is related to the critical value $\hat{D}$. Here we usually set $\lambda = \hat{D}$. Since λ is used to control smoothness penalty, a large λ will cause more piecewisely smooth segmentation.

Figure 4.6 compares the Heaviside function with the truncated quadratic function. Since the Heaviside function provide a sharp change around $\sqrt{\tau}$, it effectively distinguish the value in an ambiguity region around $\sqrt{\tau}$. Our experiments also show that using the Heaviside function the partition is more prone to obtain good discontinuities. In Figure 4.3, we show the detailed graph for this bi-partition problem. After assigning weights $D(p, 0)$ to the source side t-links, $D(p, 1)$ to the sink side t-links, and $V(p, q)$ to n-links in graph $\mathcal{G}$, we can compute the minimum cut $\mathcal{C}$ using the standard graph cut algorithm and obtain a piecewise smooth partition of the supporting region.

However, the partitioning using graph cut cannot guarantee the gradual expansion or shrinking of a region along the normal direction as shown in Figure 4.5.c, where some pixels not belonging to this region are also included. As a result, the computed transformation may not be represented as the real layer. Since the contour information of the initial seed region is not integrated in the function given in equation 4.1, the graph cut algorithm cannot correctly evolve the region contour along the normal direction (Figure 4.7.a). In order to solve this problem, we use the contour of the seed region as a prior to compute the level set, v, of this region. Then, we apply v on the t-links at the sink side and adjust the weights



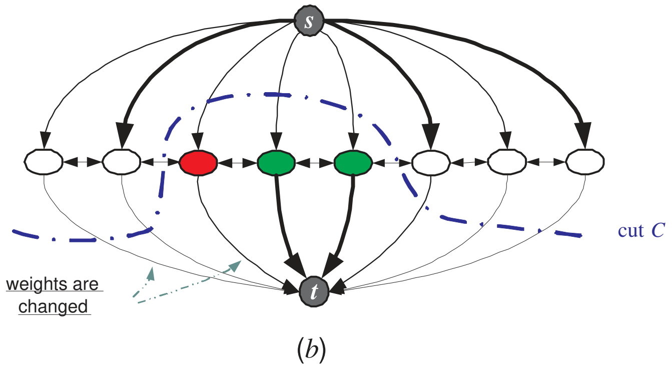


Figure 4.7: Comparison between the graph cut method without the level set representation (a) and that with the level set representation (b). The green nodes corresponds to the center of the initial seed region, which belong to the sink. After bi-partitioning in (a), the nodes far away from the center may be assigned the sink label, which cannot guarantee the gradual expansion or shrinking of a region along the normal direction. After integrating the level set representation, the weights of the links on the sink side are changed. Therefore, the cut is always made at the adjacent area around the original seed region (b). *Note:* We use the thickness to represent the weights of the links.

of the t-links for pixels outside of the region in graph $\mathcal{G}$. Therefore, we effectively restrict the graph cut algorithm to gradually expand the seed region (Figure 4.7.b). The detailed process is described as follows:

- *Step 1*: Construct a mask β of the original seed region, which has a value in [0, 1], where the inside pixels of the region are marked by 1 and the others are marked by 0. Then, compute a level set v (Figure 4.5.d) simply by convolving the region mask, β, with a Gaussian kernel such as: $v = G * \beta$, where G is the Gaussian kernel.

  Note: For a pixel i inside of the seed region, v_i, has a high constant value, and the v_i outside of the region falls down along the normal direction of the contour until $v_i = 0$ (Figure 4.5.d). Therefore, we obtain an implicit surface for this contour evolution, which can be represented by level set [76, 63]. Here we propose a novel approach to evolve the region contour by integrating the level set representation into graph cut method as the next two steps.

- *Step 2*: Warp the second image using the corresponding affine transformation, and compute SSD between the warped image and the first frame. Construct a graph $\mathcal{G}$ for the pixel with $v_i > 0$. Compute data penalty D according to the computed SSD and smoothness penalty $V(p, q)$ for each link in this graph.

- *Step 3*: Convolute v on the t-links at the sink side and change the sink weights for these pixels, then compute the minimum cut $\mathcal{C}$.

Note: The weights of the pixels inside the region are almost not changed, while the weight (p, t) will decrease when the pixel p is further from the boundary as shown in Figure 4.7.b. As a result, the minimum cut $\mathcal{C}$ is most likely to cut the outside pixels, and label them as the unsupporting pixels for this region. This way, the seed region will gradually propagate from the center outward (Figure 4.5.e).

- *Step 4*: Use the new computed region as the seed region to compute a new affine transformation by minimizing the image residue inside the region, then goto *Step 1* to do the next iteration. If the new region shrinks to a fraction of the original seed region area (e.g. 75% coverage threshold), it is discarded as a poor initial layer.

After a few iterations of the above steps, the front of the seed region will either expand or shrink along the normal direction of the contour.

Figure 4.5 and Figure 4.8.$a-b$ show the detailed process for seed region expansion started from different seeds. Figure 4.5.d shows the level set representation obtained from the initial seed region (Figure 4.5.a). Figure 4.5.e and 4.5.g are the partitioning results after the first and fourth iterations. In Figure 4.8.c, we show some good results for seed region expansion of the mobile-calendar and flower-garden sequences. Figure 4.8.d shows that we can identify the poor seed regions using the coverage threshold. Most of these poor seed regions are located at the boundary of multiple layers.

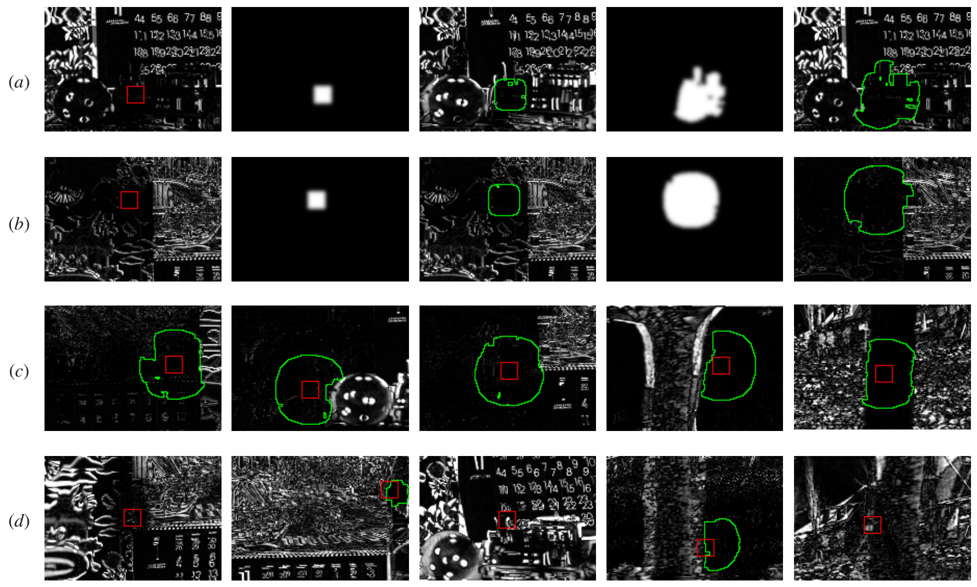


Figure 4.8: Region expansion process. (*a*) Seed region expansion started from a seed on the train in the mobile-calendar sequence, which is similar to the process in Figure 4.5.*d*-*g*. (*b*) Seed region expansion started from a seed on the background in the mobile-calendar sequence. (*c*) Some results of the good regions (inliers) after expansion. (*d*) Some results of the poor regions (outliers) after expansion, where the new region cannot cover the most of the area of the original seed. *Note:* The red box is the initial seed region. The green contours are obtained after using the bi-partitioning algorithm.

4.2.3 REGION MERGING

After expanding the regions, each good seed region becomes an initial layer. Most of these layers may share the same affine transformation. Therefore, we use a two-step merging algorithm to merge these layers to obtain the layer descriptions.

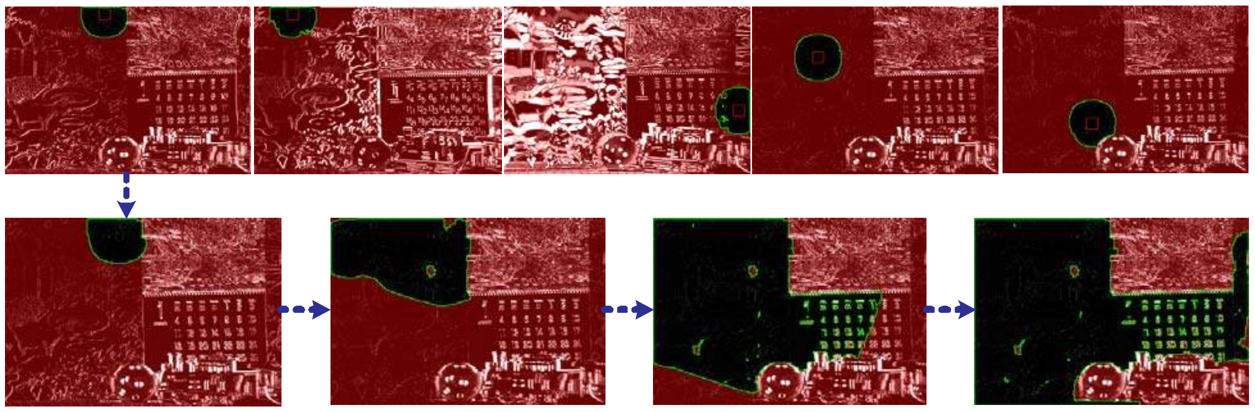


Figure 4.9: Merging process. Top: Several initial seed regions sharing the same affine transformation. Bottom: The intermediate steps of the merging process when it starts from the top left seed region.

In the first step, we only merge the layers that overlap with each other. Given two regions R_1 and R_2, we test whether the number of overlapping pixels is more than half of the pixels in the smaller region. If this is true, we compute the SSD by warping the first region, R_1, using the transformation H_2 of the second region R_2. Using this SSD as the measure and employing the graph cut algorithm, we can detect how many pixels support H_2. If the majority (say 80%) of pixels of R_1 support R_2, we merge these two regions and recompute the motion parameters using the merged pixels. After that, we use the bi-partitioning graph cut algorithm again to prune the unsupporting pixels from the new region.

If only a few pixels of R_1 support H_2, we repeat the process by warping R_2 using the transformation H_1 of R_1. In order to achieve large merged regions, we iterate the whole process a few times (typically 3 to 4) to make sure the merging process converges.

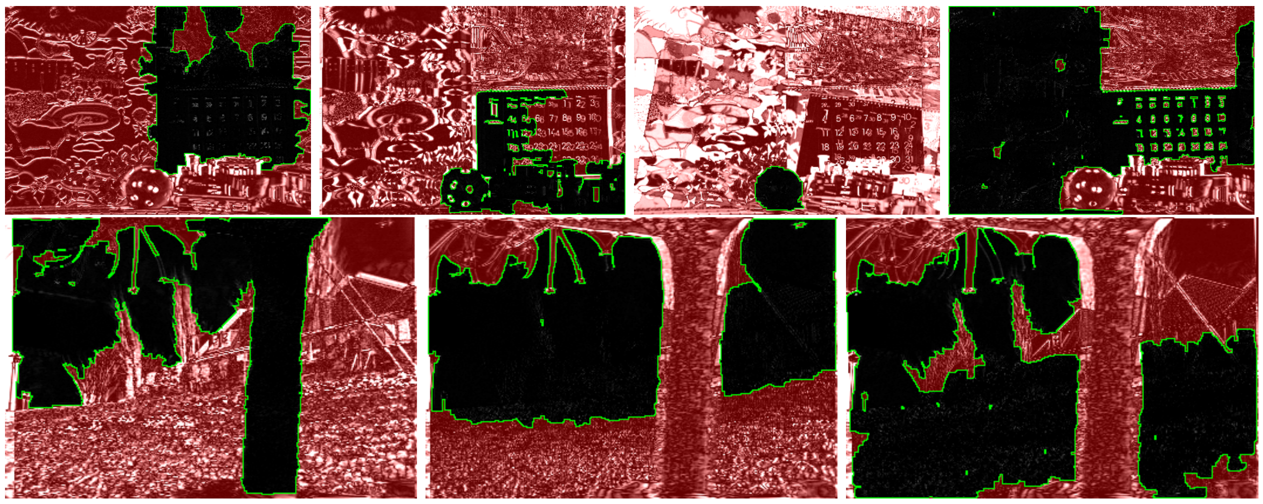


Figure 4.10: Extracting layer descriptions. Top: Four layers of the mobile-calendar sequence, which correspond to the calendar, train, ball, and wall respectively. Bottom: Three layers of the flower-garden sequence, which correspond to the tree, house, and flower-garden respectively. The green contour is the region boundary, and non-supporting pixels are marked by red. *Note:* The non-textured areas may belong to several layers due to their ambiguities, such as the white paper at the lower part of the calendar in the mobile-calendar sequence, and the blue sky in the flower-garden sequence.

After the first step of merging, only a few large regions may survive. Some of the non-overlapping regions may still share a single motion transformation. During the second step, we also merge these non-overlapping regions. Figure 4.10 shows the results for the mobile-calendar and flower-garden sequences.

4.3 MULTI-FRAME LAYER SEGMENTATION

After extracting layer descriptions in a short video clip using our proposed method in the previous section, the number of layers in the scene and the corresponding motion transformation of each layer are known. However, these layer descriptions can only provide rough layer representations and layer boundaries are incorrect. Moreover, some non-textured areas may have multiple labels due to their ambiguities as shown in Figure 4.10. In this section, we will solve this problem: Given the extracted layer descriptions, compute an accurate layer segmentation in presence of occlusion using *multiple* frames from the video sequence.

In this process, we explicitly identify the occluded pixels which will be assigned a new occlusion label, ζ. Then using the occlusion order constraint over multiple frames, the consistency of the layer segmentation between frame pair $(1, 2)$ and frame pair $(1, j)$ $(j > 2)$ is maintained, and the quality of segmentation boundary can be visibly improved. First, we will state the occlusion order constraint. Next, we introduce a three-state pixel graph to handle occlusion problem between two frames in motion segmentation. Third, based on this pixel graph, we integrate this occlusion order constraint in a novel multi-frame graph model which can be minimized using the graph cut algorithm.

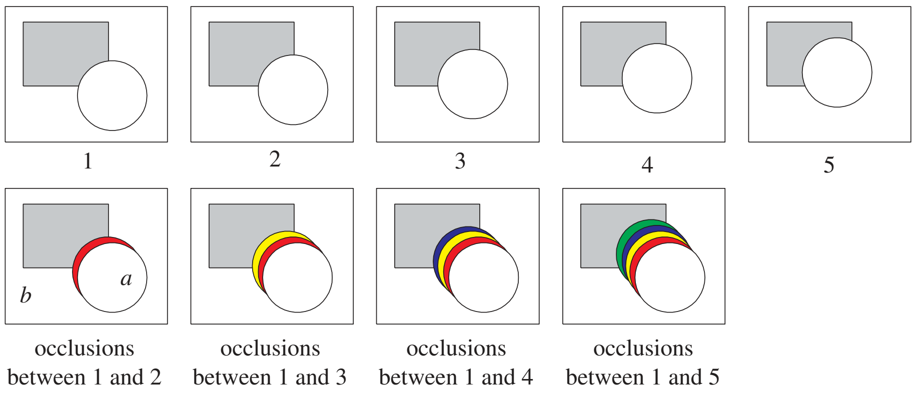

1
2
3
4
5
b
a
occlusions between 1 and 2
occlusions between 1 and 3
occlusions between 1 and 4
occlusions between 1 and 5

Figure 4.11: The occlusion order in a short video clip containing five consecutive frames (the first image is the reference image). The top row shows the five-frame sequence, where a solid circle is moving along the left-top direction. The bottom images show the occlusions (color areas) between the first frame and other frames. It is clear that the occlusion area is increasing with time.

4.3.1 OCCLUSION ORDER CONSTRAINT

With the intention of computing an accurate motion layer segmentation of a video clip, let's first take a look of the occlusion process over a temporal domain. Figure 4.11 shows the occlusion has a temporal order for a linearly moving object. It is obvious that occlusion area is increasing with the temporal order. During a short period (3-5 frames), this observation is not violated if the object is not thin and not moving fast. Therefore, based on this assumption, we state the occlusion order constraint as follows:

- *Rule 1*: During a short period, if a pixel is occluded between frames 1 and j, this pixel will also be occluded between frames 1 and $(j+1)$.

- *Rule 2*: If a pixel, p_{j-1}, is assigned a label $f_{p_1} \neq \zeta$ between frames 1 and j, then pixel p_{k-1} should be assigned either f_{p_1} or ζ between frames 1 and k, where $k > j$; and pixel p_{k-1} should be assigned f_{p_1} between frames 1 and k, where $k < j$. [2]

According to this occlusion order constraint, only the pixels at the same image coordinates in two consecutive frame pairs can influence each other, such as p_1 and p_2 in frame pair $(1,2)$ and $(1,3)$, which can effectively maintain the segmentation consistency between the consecutive frame pair.

Now, the multi-frame motion segmentation problem can be formulated as an energy minimization problem of the following energy function:

$$E = \sum_{j=1}^{n-1}(E_{smooth_j}(f) + E_{data_j}(f) + E_{oc_j}(f)) + \sum_{j=1}^{n-2} E_{order_j}(f), \tag{4.4}$$

where j is the frame number, and n is the total number of frames. Compared to Eq. 4.1, there are two additional terms in this equation. The first one is $E_{oc}(f)$, which is used to impose the occlusion penalties for the occluded pixels between frames 1 and $(j+1)$. The second one is $E_{order}(f)$, which is used to impose occlusion order penalties for maintaining the occlusion order constraint on each consecutive image pairs, such as frame pair $(1,2)$ and

[2]If we have only one image pair, we use p to refer the pixel in reference image for this image pair. If we have multiple image pairs, we use p_1 to refer the pixel in the reference image for image pair $(1,2)$, use pixel p_2 to refer the pixel in the reference image for image pair $(1,3)$, and pixel p_{k-1} to refer the pixel in the reference image for image pair $(1,k)$, and so on. *Note*: p_1, p_2, $\cdots$, p_{k-1} have the same location p in the reference image for every frame pair. The reference image is always frame 1.

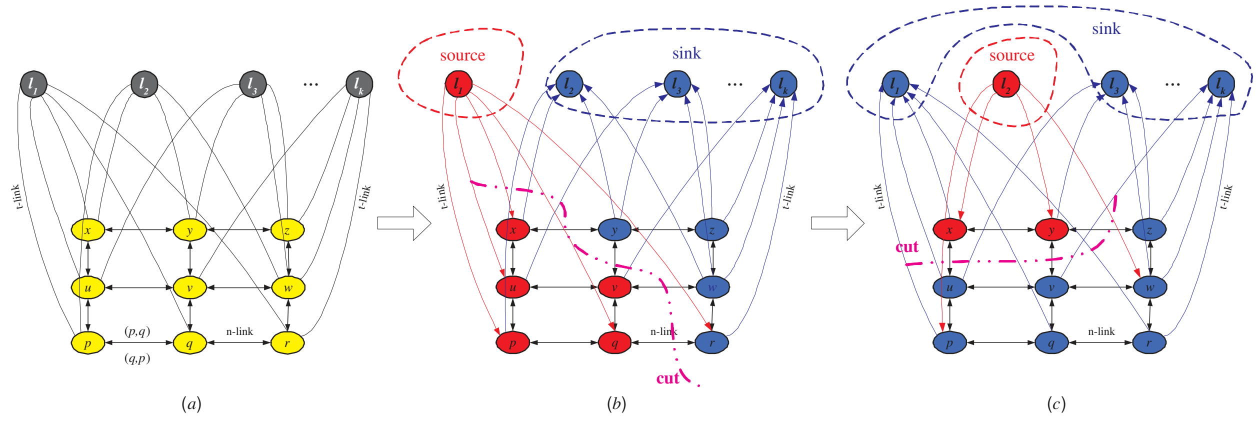


Figure 4.12: A typical multiway cut problem. (a) $\{l_1, l_2, \cdots, l_k\}$ is the label set. $p, q, r, \cdots, x, y, z$ are the nodes associated with the pixels in an image. (b) After selecting l_1 as the α label (or source) and the other label terminals are grouped as one sink, a bi-partition can be achieved, where red nodes are assigned label α and the blue nodes will keep original label f_p. (c) An example for another step of α-expansion. After these two steps ((b) and (c)), p, q, u, and v will be assigned by label l_1, and x and y will be assigned by label l_2.

$(1, 3)$. In this multiple labeling system, given a pixel p_j, the label f_{p_j} of pixel p_j is assigned one label from a label set $\mathcal{L} = \{l_1, l_2, \cdots, l_k\} \cup \{\zeta\}$, where k is the real labeling number extracted from the layer descriptions obtained in the previous section.

4.3.2 THREE-STATE PIXEL GRAPH

In order to minimize the multiple label energy function, we have to use α-expansion or $\alpha - \beta$ swapping techniques [50, 15] to solve the multiway cut problem [13]. In a multiway cut problem, there is a label set $\mathcal{L} = \{l_1, l_2, \cdots, l_k\}$ as shown in Figure 4.12.a, and each node in the graph will finally be assigned one of these labels according to its data energy (t-links) and smoothness energy (n-links). However, Dahlhaus et al. have already shown that to find a minimum cost multiway cut is NP-complete [24]. One feasible solution for this problem is to use multiple two-terminal subgraphs to achieve an approximate result as shown in Figure 4.12.b and c. In each step, we randomly or sequentially pickup a label as the source terminal which is named as α (such as l_1 in Figure 4.12.b), and merge the other labels as one sink terminal. Then, the maximum flow algorithm will give an optimal solution for this bi-partition problem at a linear computational time, and the total energy of this graph will be reduced. In each step, the α-expansion of f allows any set of pixels to change their original labels to α in one step as shown in Figure 4.12.b and c. Finally, a global approximation of the multiway cut problem can be obtained until the energy is not reduced for each label in $\mathcal{L}$.

Traditionally, each node in graph is only associated with one pixel such as Figure 4.3 and Figure 4.12. Thus, each pixel has one individual two-state pixel graph as shown in Figure 4.13.a. In this pixel graph, the states of each pixel will be assigned by new label α

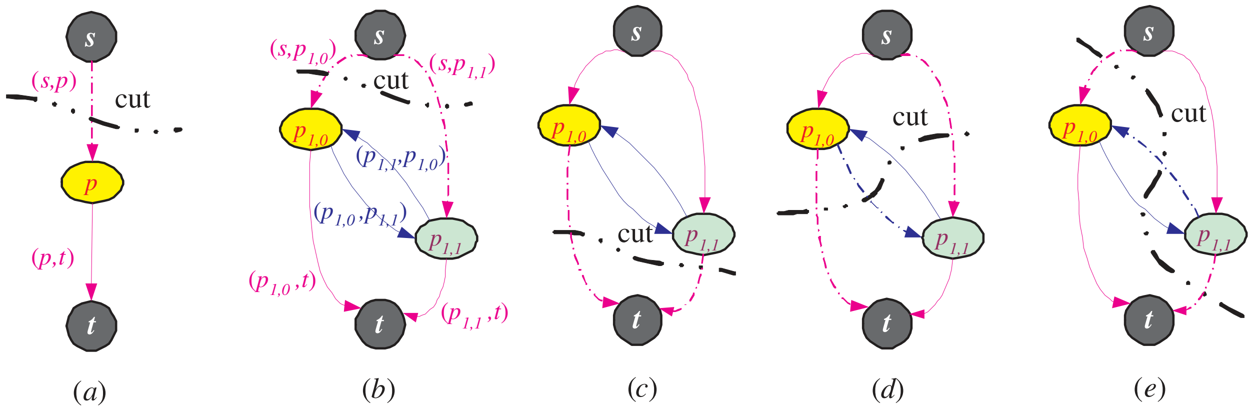


Figure 4.13: Pixel graphs. (a) A two-state pixel graph has two states which are corresponding to the sink or source respectively. $(b-e)$ Three-state pixel graphs which can handle the occlusion label. After one α-expansion of an independent pixel graph p, there three possible cuts. (b) The two nodes belong to the sink, and p will be assigned the new label α. (c) The two nodes belong to the source, p will keep the original label f_p. (d) One node belongs to source and the other belongs to the sink, therefore p is occluded and assigned the label ζ. (e) Impossible case due to the link $p_{1,1}, p_{1,0} = \infty$.

or keep original label f_p, which can be naturally represented by the two states, [0] or [1], of each node[3].

However, in motion segmentation application, one pixel can be assigned by three labels α, f_p, or ζ at one step α-expansion. If using one node to represent one pixel, this pixel will not have three states. In order to provide three states for a single pixel, we use two nodes

[3]After the graph cut partition, each node will be assigned either source [0] or sink [1]. There are two possible states for each node.

to construct a pixel graph, and four possible combination of these two nodes are available for this pixel, such as [0,0], [0,1], [1,0], and [1,1].

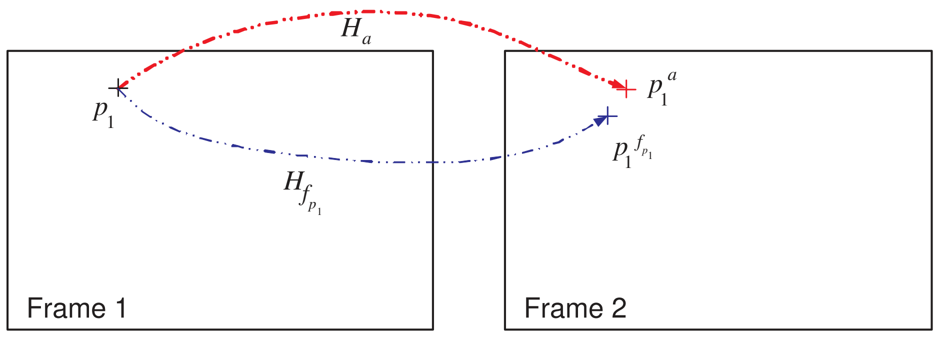
Given a pixel p_1 in image pair (1,2), the corresponding pixel graph is constructed in Figure 4.13.$b-e$. There are two nodes, $p_{1,0}$ and $p_{1,1}$, and one pair of *occlusion n-links*, $(p_{1,0}, p_{1,1})$ and $(p_{1,1}, p_{1,0})$, associated with this pixel. If the minimum cut, $\mathcal{C}$, cuts the link $(p_{1,0}, p_{1,1})$, the pixel p_1 is occluded. Using the link weights given in Table 4.1, Figure 4.13 shows three cases for $\mathcal{C}$ after bi-partition on this pixel graph. Let f_{p_1} be the original label of pixel p_1 in the reference image. The pixel will be assigned a new label $f_{p_1}^{\mathcal{C}}$ as follows:

$$f_{p_1}^{\mathcal{C}} = \begin{cases} \alpha & \text{if } (s, p_{1,0}) \in \mathcal{C},\ (s, p_{1,1}) \in \mathcal{C} \text{ (Figure4.13.}b\text{)}, \\ f_{p_1} & \text{if } (p_{1,0}, t) \in \mathcal{C},\ (p_{1,1}, t) \in \mathcal{C} \text{ (Figure4.13.}c\text{)}, \\ \zeta & \text{if } (p_{1,0}, t) \in \mathcal{C},\ (s, p_{1,1}) \in \mathcal{C}, (p_{1,0}, p_{1,1}) \in \mathcal{C} \text{ (Figure4.13.}c\text{)}, \end{cases} \tag{4.5}$$

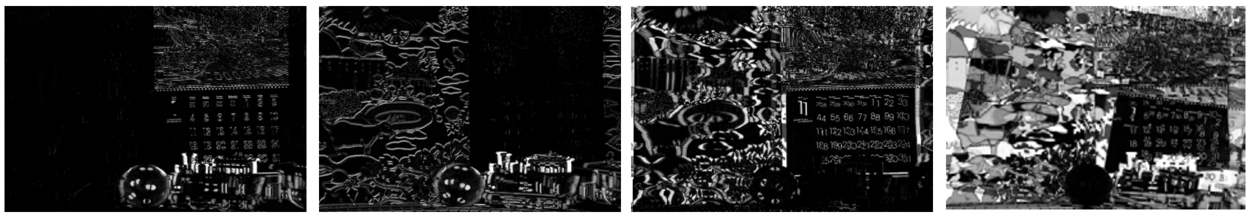
where ζ is the occlusion label. In the first case, pixel p_1 is assigned a new label α, where the node state is [0,0]. In the second case, pixel p_1 will keep its original label, where the node state is [1,1]. In the occlusion case, both data penalties $D(p_1, \alpha)$ and $D(p_1, f_{p_1})$ of pixel p_1 are greater than the occlusion penalty $D(p_1, \zeta)$, which is a fixed empirical value[4]. This means that it is not suitable to assign either the original label f_{p_1} or the new label α to this pixel. Hence, this pixel is an occluded pixel and is assigned ζ (Figure 4.13.c), where the node state is [0,1]. Due to infinite weight of $(p_{1,1}, p_{1,0})$, the fourth node state [1,0] is disabled.

To compute the data penalties $D(p_1, f_{p_1})$ and $D(p_1, \alpha)$, we first need to determine the image difference $\delta(p_1)$ related to each label for pixel p_1. As shown in Figure 4.14.a, pixel p_1 in

[4]We usually set the value of $D(p_1, \zeta)$ is slightly larger than the critical value $\hat{D}$ (see Eq.4.3). If $D(p_1, \zeta)$ is increasing, the number of the occlusion pixels will be reduced.



(*a*)



(*b*) (*c*) (*d*) (*e*)

Figure 4.14: Determine data penalty for pixel p_1. (a) Pixel p_1 in the first frame may be projected on different locations, $p_1^{f_{p_1}}$ and p_1^{α}, in the second frame by transformations $H_{f_{p_1}}$ and H_α respectively. $(b-e)$ are difference maps corresponding to background, calendar, mobile, and ball respectively, where the image intensities are corresponding to the differences.

frame 1 may be projected on different locations, $p_1^{f_{p_1}}$ and p_1^{α}, in frame 2 by transformations $H_{f_{p_1}}$ and H_α respectively. Therefore, $\delta(p_1)$ can be obtained as follows:

$$\delta(p_1, f_{p_1}) = |I_1(p_1) - I_2(p_1^{f_{p_1}})| \quad (4.6)$$

$$\delta(p_1, \alpha) = |I_1(p_1) - I_2(p_1^{\alpha})| \quad (4.7)$$

where $I_1(p_1)$ is the intensity value of pixel p_1 in the first frame, $I_2(p^{f_{p_1}})$ is the intensity value of pixel $p_1^{f_{p_1}}$ in the second frame. p_1 and $p_1^{f_{p_1}}$ are correspondences by transformation

$H_{f_{p_1}}$. The notations related to α have the similar meanings. Then, the data penalties are computed as follows:

$$D(p_1, f_{p_1}) = \tan^{-1}(\delta(p_1, f_{p_1})^2 - \tau) + \pi/2 \quad (4.8)$$

$$D(p_1, \alpha) = \tan^{-1}(\delta(p_1, \alpha)^2 - \tau) + \pi/2 \quad (4.9)$$

where τ is set as same as Eq.4.3. To improve computational performance, we precompute the difference map $\delta(p_j)$ for each possible label as shown in Figure 4.14.$b - e$, where four difference maps are computed by using frame 1 and 5 from mobile-calendar sequence.

4.3.3 MULTI-FRAME MOTION SEGMENTATION VIA GRAPH CUT

Based on basic graph element, pixel graph, the multi-frame motion segmentation graph model can be constructed as Figure 4.15. To illustrate occlusion order constraint in multi-frame segmentation, we stack four pairs of image nodes together in this graph. Note that each image pair involves the first frame (the reference frame) and one of the other frames, which is consistent with Figure 4.11.

In Figure 4.15, each image pair is separated by the red dotted lines. In each image pair $(1, j+1)$, $j > 1$, only the pixel in the reference image (frame 1) will be assigned one pixel graph. For each pixel p_j[5], a pixel graph is created with two nodes $p_{j,0}$ and $p_{j,1}$. In each image

[5] p_j has the same location p in frame 1 for every frame pair $(1, j+1)$.

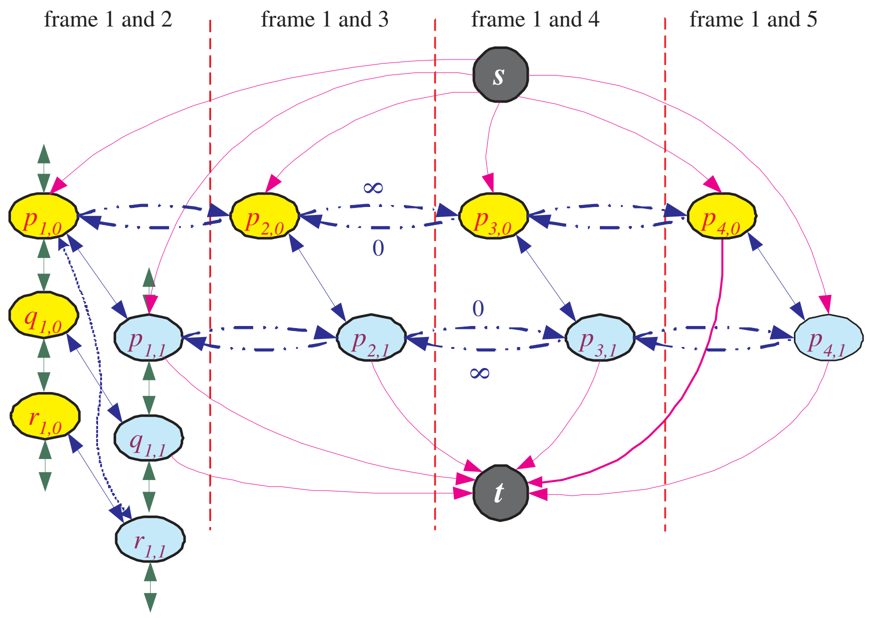


Figure 4.15: This graph is constructed using five consecutive frames, which have four image pairs related to the reference image. The red lines separate each pair of images into one block. The blue n-links are introduced to maintain the occlusion order constraint. *Note: Only some of the nodes and links are shown here.*

pair $(1, j+1)$, p_j belongs to a pixel set, $\mathcal{P}_j$, where $\mathcal{P}_j$ is the set of pixels in the reference image for image pair $(1, j+1)$. In Figure 4.15, there are four pixels subset $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, and $\mathcal{P}_4$ corresponding to each image pair.

According to the occlusion order constraint, a set of *order n-links* (blue edges), such as $(p_{3,0}, p_{2,0})$ and $(p_{2,0}, p_{3,0})$, are added in the graph $\mathcal{G}$ to interact with the pixel graph at the same image coordinates. To simplify graph $\mathcal{G}$, we only show two nodes from one particular

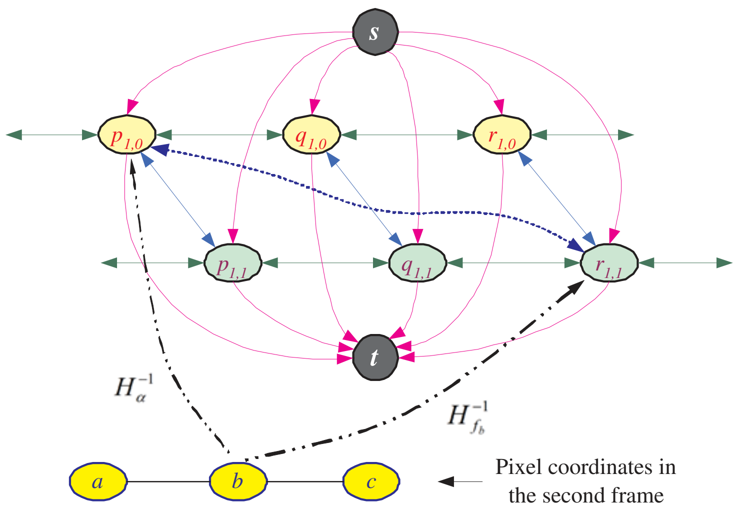


Figure 4.16: A graph $\mathcal{G}_{1,2}$, where three basic pixel graphs are shown corresponding to pixels p, q, and r respectively. The n-links between neighboring pixels are to enforce the smoothness penalties, such as $(p_{1,1}, q_{1,1})$ and $(q_{1,1}, p_{1,1})$. After applying the inverse transformations H_α^{-1} and $H_{f_b}^{-1}$ to pixel b in the second frame such that $b^\alpha = p_{1,0}$ and $b^{f_b} = r_{1,1}$, a pair of n-links is introduced to enforce the symmetric property of the occlusion, such as $(p_{1,0}, r_{1,1})$ and $(r_{1,1}, p_{1,0})$.

pixel p_j for each image pair to illustrate these *order n-links*. The detailed sub-graph $\mathcal{G}_{1,2}$ for the first image pair is redrawn in Figure 4.16.

Before we describe how to minimize the energy E for the whole graph $\mathcal{G}$, we first discuss the interaction of the nodes in sub-graph $\mathcal{G}_{1,2}$, and then discuss how to assign the weights to these links. To reduce the complexity, we show only three pixels p_1, q_1, and r_1 of frame 1 in graph $\mathcal{G}_{1,2}$, where each pixel has one pixel graph.

Table 4.1: Weights of the links. Occlusion penalty $D(p, \zeta)$ is a empirical constant. *Note:* $b^{\alpha}_{j,1}$ and $b^{f_b}_{j,0}$ are the symmetric node pair to enforce the symmetric occlusion property, such as the nodes $p_{1,0}$ and $r_{1,1}$ in Figure 4.16.

Edge	**Weight**	**for**
$(s, p_{j,1}), (p_{j,0}, t)$	0	$p_j \in \mathcal{P}_j$
$(p_{j,1}, t)$	$D(p_j, f_{p_j})$	$p_j \in \mathcal{P}_j, f_{p_j} \neq \zeta$
$(p_{j,1}, t)$	∞	$p_j \in \mathcal{P}_j, f_{p_j} = \zeta$
$(s, p_{j,0})$	$D(p_j, \alpha)$	$p_j \in \mathcal{P}_j$
$(p_{j,0}, p_{j,1})$	$D(p_j, \zeta)$	$p_j \in \mathcal{P}_j$
$(p_{j,1}, p_{j,0})$	∞	$p_j \in \mathcal{P}_j$
$(p_{j,i}, q_{j,i})$ $(q_{j,i}, p_{j,i})$	$V(p_{j,i}, q_{j,i})$	$\{p_j, q_j\} \in \mathcal{N}$,$\{p_j, q_j\} \in \mathcal{P}_j$
$(b^{\alpha}_{j,0}, b^{f_b}_{j,1})$	$D(p, \zeta)$	$b_j \in \mathcal{P}_j, b^{\alpha}_j \in \mathcal{P}_j, b^{f_b}_j \in \mathcal{P}_j$
$(b^{\alpha}_{j,1}, b^{f_b}_{j,0})$	∞	$b_j \in \mathcal{P}_j, b^{\alpha}_j \in \mathcal{P}_j, b^{f_b}_j \in \mathcal{P}_j$
$(p_{(j+1),0}, p_{j,0})$	0	$p_j \in \mathcal{P}_j, p_{(j+1)} \in \mathcal{P}_{(j+1)}$
$(p_{j,0}, p_{(j+1),0})$	∞	$p_j \in \mathcal{P}_j, p_{(j+1)} \in \mathcal{P}_{(j+1)}$
$(p_{(j+1),1}, p_{j,1})$	∞	$p_j \in \mathcal{P}_j, p_{(j+1)} \in \mathcal{P}_{(j+1)}$
$(p_{j,1}, p_{(j+1),1})$	0	$p_j \in \mathcal{P}_j, p_{(j+1)} \in \mathcal{P}_{(j+1)}$

In graph $\mathcal{G}_{1,2}$ (Figure 4.16), the smoothness energy function, $E_{smooth}(f)$, is implemented by the *smoothness n-links*, which connect each pair of neighboring pixel graphs such as $(q_{1,1}, p_{1,1})$ and $(p_{1,1}, q_{1,1})$. In order to compute the smoothness penalty term $V(p_{1,i}, q_{1,i})$ of a link $(p_{1,i}, q_{1,i})$, we warp the second image I_2 to obtain the warped image $I_2^{H_{f_i}^{-1}}$ by applying the inverse motion transformation $H_{f_i}^{-1}$, corresponding to label f_i, for each label in the layer descriptions. Here

$$f_i = \begin{cases} \alpha & \text{if } i = 0 \\ f_{p_j} & \text{if } i = 1 \end{cases} . \tag{4.10}$$

Therefore, the smoothness penalty term $V(p_{j,i}, q_{j,i})$ can be computed as

$$V(p_{j,i}, q_{j,i}) = \begin{cases} 4\lambda & \text{if } \max(|I_1(p) - I_1(q)|, \quad |I_{(j+1)}^{H_{f_i}^{-1}}(p) - I_{(j+1)}^{H_{f_i}^{-1}}(q)|) < 4, \\ 2\lambda & \text{if } 4 \leq \max(|I_1(p) - I_1(q)|, \quad |I_{(j+1)}^{H_{f_i}^{-1}}(p) - I_{(j+1)}^{H_{f_i}^{-1}}(q)|) < 8, \\ \lambda & \text{otherwise,} \end{cases} \tag{4.11}$$

where I_1 is the first frame, $I_{(j+1)}^{H_{f_i}^{-1}}$ is the warped version of $I_{(j+1)}$ obtained by applying inverse transformation $H_{f_i}^{-1}$, and λ is an empirical constant as Eq.4.2.

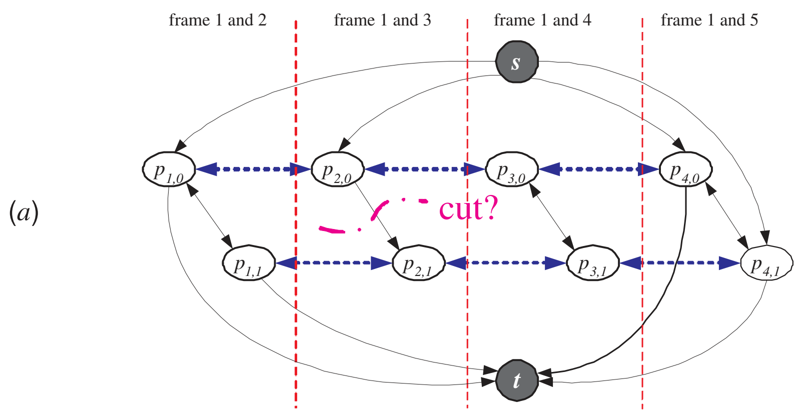
To deal with the symmetric properties of the occlusion, a set of new symmetric occlusion n-links are added to connect the related nodes. Given a pixel b in the second frame, two pixels b^{α} and b^{f_b} in the first frame can be mapped to the same pixel b by transformations H_{α} and H_{f_b}, where f_b is the current label of b in the second frame. After applying the inverse transformations H_{α}^{-1} and $H_{f_b}^{-1}$ to b, we can determine both b^{α} and b^{f_b}. Then, a pair of n-links are added to connect these two nodes, such as the blue dotted links $(r_{1,1}, p_{1,0})$ and $(p_{1,0}, r_{1,1})$ shown in Figure 4.16. With the help of these symmetric occlusion n-links, the occlusion penalties from frame 2 to frame 1 are also specified.

4.3.4 ENERGY MINIMIZATION

After assigning weights 0, ∞, ∞, and 0 to *n-links* $(p_{(i+1),0}, p_{i,0})$, $(p_{i,0}, p_{(i+1),0})$, $(p_{(i+1),1}, p_{i,1})$, and $(p_{i,1}, p_{(i+1),1})$ respectively, the occlusion order constraint is fully satisfied. Therefore, the energy function in Eq.4.4 can be rewritten as

$$\begin{aligned}
E &= \sum_{j=1}^{n-1}(E_{smooth_j}(f) + E_{data_j}(f) + E_{oc_j}(f)) + \sum_{j=1}^{n-2} E_{order_j}(f) \\
&= \sum_{j=1}^{n-1}(\sum_{i=0}^{1}(\sum_{(p_j,q_j)\in\mathcal{N}} V(p_{j,i}, q_{j,i})) + \sum_{p_j\in\mathcal{P}_j}(D(p_j, f_{p_j})\cdot T(f_{p_j} \neq \zeta)) \\
&+ \sum_{p_j\in\mathcal{P}_j}(D(p_j,\zeta)\cdot T(f_{p_j} = \zeta))) + \sum_{j=1}^{n-2}(\sum_{p_j\in\mathcal{P}_j}(\infty\cdot T(f_{p_{j+1}} \neq \zeta \wedge f_{p_j} \neq f_{p_{j+1}})) \\
&= \sum_{j=1}^{n-1}(\sum_{i=0}^{1}(\sum_{(p_j,q_j)\in\mathcal{N}} V(p_{j,i}, q_{j,i})) + \sum_{p_j\in\mathcal{P}_j} D(p_j, f_{p_j})) \\
&+ \sum_{j=1}^{n-2}(\sum_{p_j\in\mathcal{P}_j}(\infty\cdot T(f_{p_{j+1}} \neq \zeta \wedge f_{p_j} \neq f_{p_{j+1}})), \qquad (4.12)
\end{aligned}$$

where $V(p_{j,i}, q_{j,i})$ is smoothness penalty term, $D(p_j, f_{p_j})\cdot T(f_{p_j} \neq \zeta)$ is data penalty term, $D(p_j,\zeta)\cdot T(f_{p_j} = \zeta)$ is occlusion penalty term, and $\infty\cdot T(f_{p_{j+1}} \neq \zeta \wedge f_{p_j} \neq f_{p_{j+1}})$ is occlusion order penalty term. Using three state pixel graph system, the data penalty term and occlusion penalty term can be merged together to obtain a new data and occlusion penalty term $D(p_j, f_{p_j})$. $T(\cdot)$ is 1 if its argument is true and 0 otherwise. From the occlusion order penalty term, we can see that if the label of pixel p_{j+1} is not ζ, pixels p_{j+1} and p_j should have the same label, otherwise an infinity penalty will be imposed on the corresponding occlusion order n-links. This is consistent with our occlusion order constraint listed in Section 4.3.1.



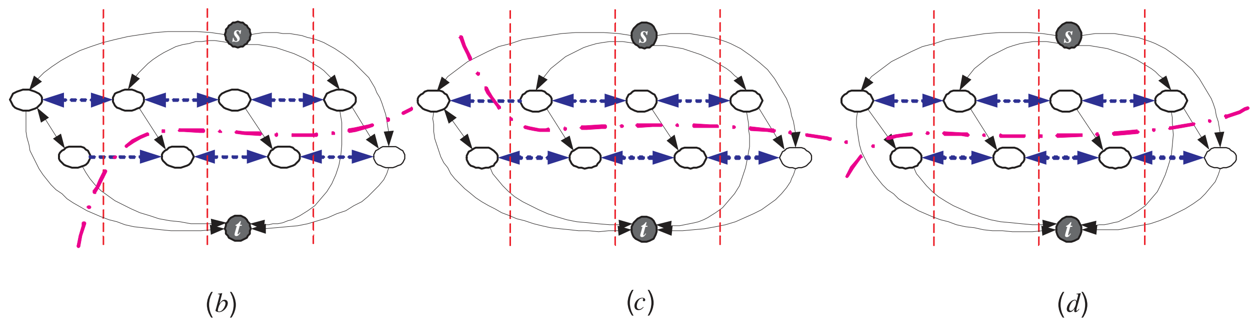


Figure 4.17: Verification for *Rule 1.* Three possible cases $(b-d)$ after assuming a cut across $(p_{2,0}, p_{2,1})$ in (a). All of the occlusion links after $(p_{2,0}, p_{2,1})$ are definitely crossed by the minimal cut $\mathcal{C}$, while the occlusion links before $(p_{2,0}, p_{2,1})$ may or may not be cut.

We can easily verify the occlusion constraint by assuming the minimum cut position. For example, to verify *Rule 1*, we assume that a minimum cut $\mathcal{C}$ cuts the *occlusion n-link* $(p_{2,0}, p_{2,1})$ in the second block as shown in Figure 4.17.a, and therefore pixel p_2 is occluded between frame pair $(1, 3)$. According to the weights of these links, $\mathcal{C}$ should not cut the *order n-links* $(p_{2,0}, p_{3,0})$ and $(p_{3,1}, p_{2,1})$ since their weights are ∞. Therefore, this minimal cut $\mathcal{C}$ will definitely cut the *occlusion n-links* $(p_{3,0}, p_{3,1})$. Similarly, $(p_{4,0}, p_{4,1})$ will also be cut. As a result, pixel p_3 will be occluded between frame pair $(1, 4)$, and pixel p_4 will be occluded

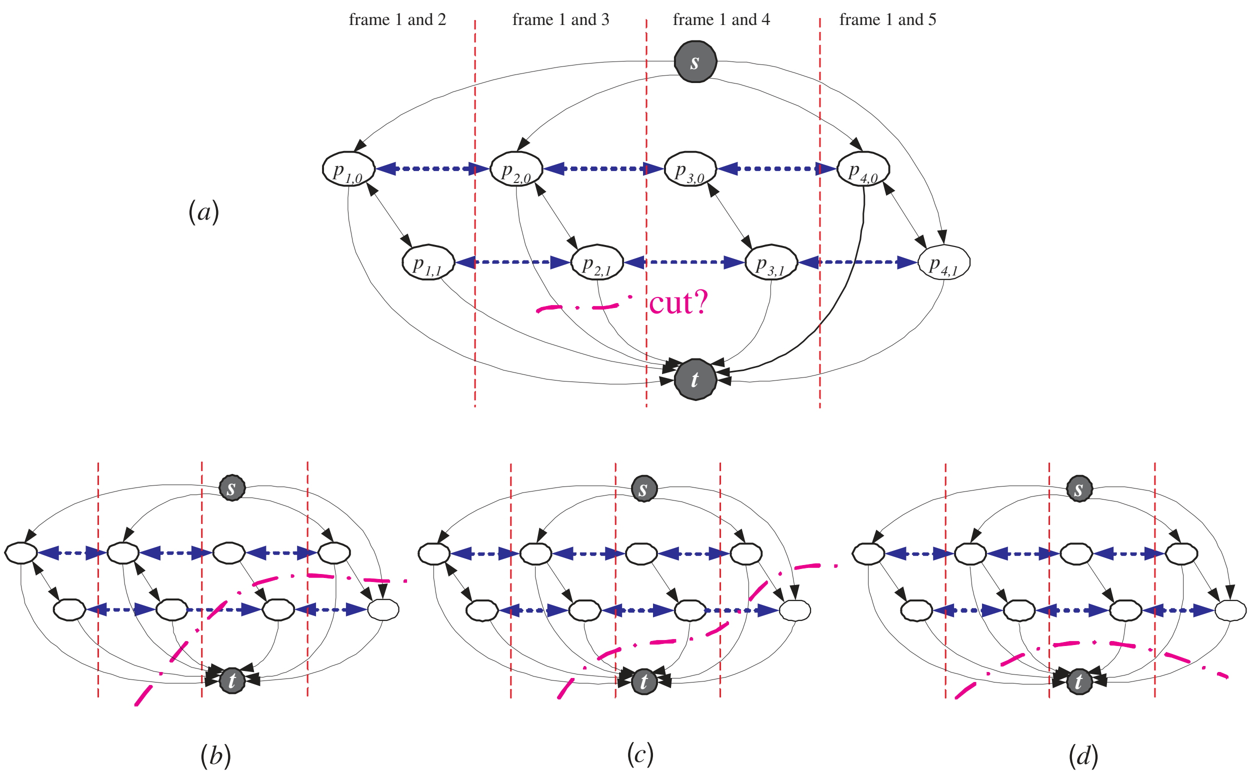


Figure 4.18: Verification for *Rule 2*. Three possible cases $(b-d)$ after assuming a cut across $(p_{2,0},t)$ and $(p_{2,1},t)$ in (a). All of the sink side t-links before the $(p_{2,0},t)$ and $(p_{2,1},t)$ are definitely crossed by the minimal cut $\mathcal{C}$, while the sink side t-links after the $(p_{2,0},t)$ and $(p_{2,1},t)$ may or may not be cut. *Note*: in this case none of the source side t-links can be crossed.

between frame pair $(1,5)$. On the other hand, the label of pixel p_1 in the first block $(1,2)$ is not affected by the second block $(1,3)$ since the weights of *occlusion n-links* $(p_{2,0},p_{1,0})$ and $(p_{1,1},p_{2,1})$ are 0. Therefore, there are three possible cases after cutting the link, $(p_{2,0},p_{2,1})$, as shown in Figure 4.17.*b*-*d*.

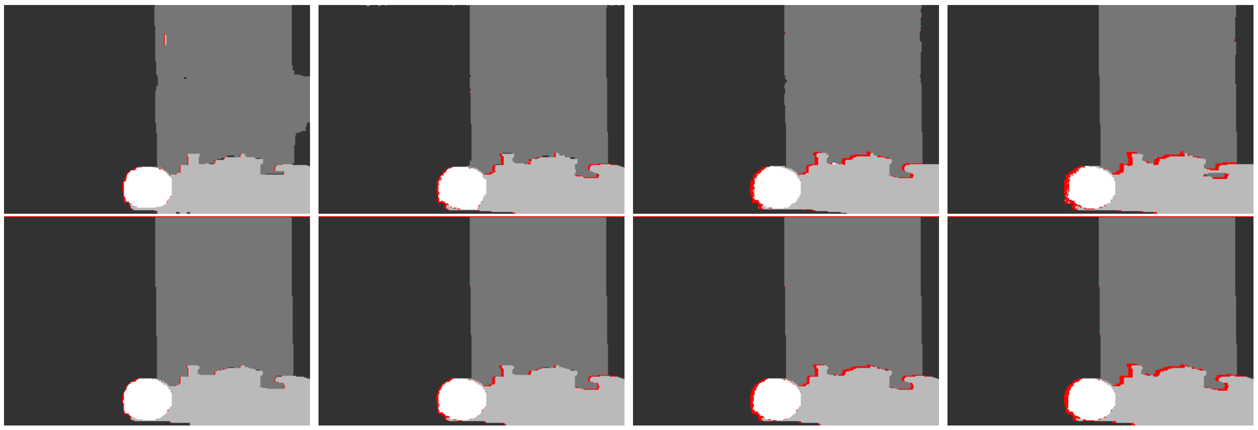


Figure 4.19: Segmentation results for frame 1 of the mobile-calendar sequence. Top: The segmentation results obtained by using only two frames. Bottom: The segmentation results obtained by using five frames (1-5) with the occlusion order constraint. (a) Segmentation results between frames 1 and 2. (b) Segmentation results between frames 1 and 3. (c) Segmentation results between frames 1 and 4. (d) Segmentation results between frames 1 and 5. The red pixels in the segmented images are the occluded pixels, which are consistently increasing with time in the bottom row. After using the occlusion order constraint on these frames, the segmentation results on the reference image (frame 1) are much more consistent than those without the occlusion order constraint (top).

In order to verify *Rule 2*, we need to assume that some pixel has a label $f_{p_j} \neq \zeta$. For example, after assuming that the pixel p_2 keeps the original label f_{p_2} between frames 1 and 3 during α-expansion (Figure 4.18.a), the nodes $p_{2,0}$ and $p_{2,1}$ belong to the source, and the cut will cross the t-links $(p_{2,0}, t)$ and $(p_{2,1}, t)$. Since the weight of n-link $(p_{2,1}, p_{1,1})$ is ∞, node

$p_{1,1}$ cannot belong to the sink. Thus, $p_{1,0}$ and $p_{1,1}$ both belong to the source, and the pixel p_1 between frame pair $(1, 2)$ will also keep the original label f_{p_1}. Similarly, node $p_{3,0}$ will belong to the source due to $(p_{2,0}, p_{3,0}) = \infty$. As a result, there are also three possible cases as shown in Figure 4.18.$b - d$.

Figure 4.19 compares the segmentation results obtained using five frames with those obtained using only two frames. Due to the use of multiple frames with the occlusion order constraint, the artifacts are removed and the segmentation results are more consistent as shown in Figure 4.19.$b - d$. Moreover, it is obvious that the occluded areas between the overlapping layers increase over time.

In order to obtain segmentation results for each pair of neighboring fames in a sequence, we always use 3-5 consecutive frames to preform segmentation since our assumption is more valid and feasible in a short period (we will discuss in the next section). For example, after obtaining initial layer descriptions between frames 1 and 5 in Section 4.2, we can easily estimate the motion parameters between frame pairs $(1, 2)$, $(1, 3)$, and $(1, 4)$ for each real layer label. Based on these motion parameters, five consecutive frames 1,2,3,4,5 are used in multi-frame graph cut algorithm to compute the segmentation, where we can simultaneously achieve the segmentation for each frame pair $(1, j), 2 \leq j \leq 5$. In order to perform segmentation for frame 2 and 3, we first estimate the initial layer descriptions (support regions and corresponding motion parameters of each layer) from the previous segmentation results between $(1, 2)$. Then, based on the initial layer descriptions, another five consecutive frames 2,3,4,5,6 will be used to refine the segmentation between $(2, 3)$ by employing the multi-frame graph cut algorithm. Finally, the segmentation of the whole sequence can be achieved by repeating the process.

4.4 EXPERIMENTS

In this section, we demonstrate our results on two standard motion sequences, mobile-calendar (Figure 4.20) and flower-garden (Figure 4.21), and other sequences.

Figure 4.20 and 4.21 show the segmentation results for the mobile-calendar and flower-garden sequences. We used five frames to extract the layers for the mobile-calendar sequence, and used three frames to extract the layers for the flower-garden sequence. We also compared our results with other methods [4, 99, 46, 45] for these two standard sequences. Since the ground truth for these sequences is not available, we have to limit our analysis to qualitative comparisons. Figure 4.22 shows the comparison on these two standard motion sequences. Compared with the previous approaches, our method not only explicitly determines the occluded pixels, but also provides more precise and finer boundaries between overlapping layers than the previous methods. Figure 4.23 shows the energy reducing for these two standard sequences. After several iterations (usually around real layer number), The energy is fast converged to an approximately optimized solution.

We also applied our method to our own sequence with a large occlusion, car-map (Figure 4.24), where the car is moving behind the map and the scale of the car is apparently

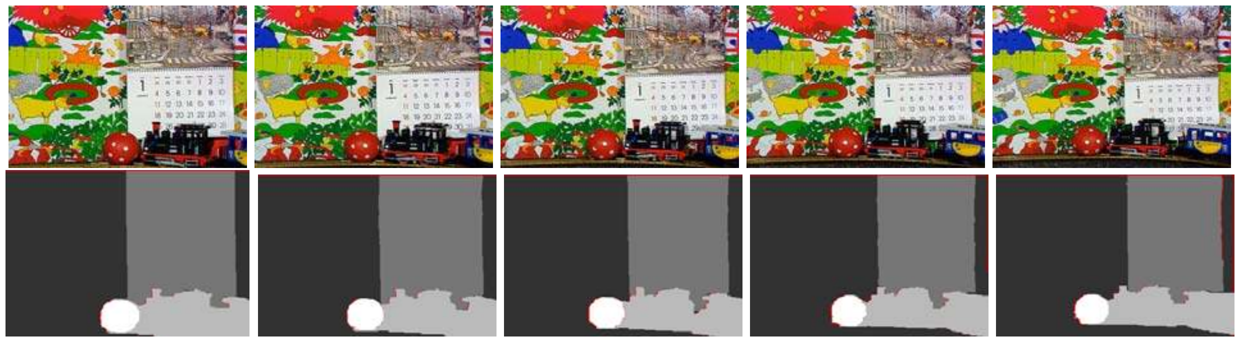


Figure 4.20: The segmentation results for the mobile-calendar sequence. The red pixels are occluded pixels.

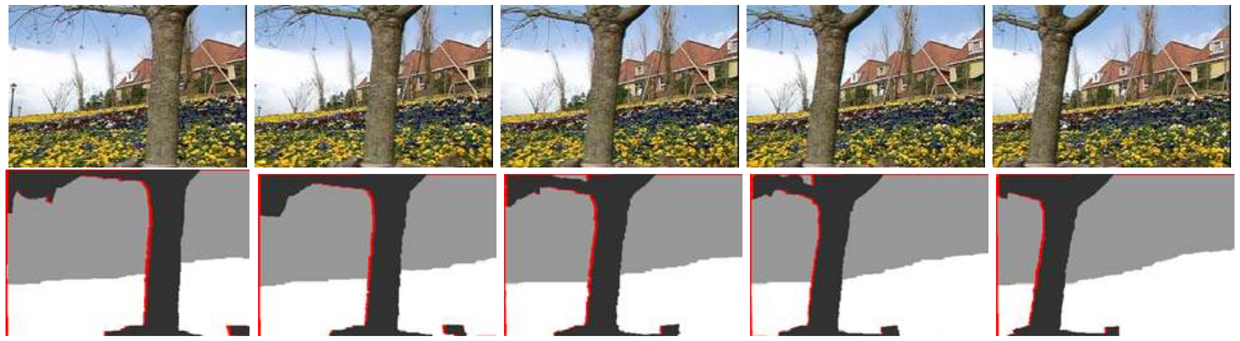


Figure 4.21: The segmentation results for the flower-garden sequence. The red pixels are occluded pixels.

changed. The sequence is taken by a hand-held moving video camera. During some frames, most parts of the car are occluded by the map. Once the car moves behind the map, it is difficult to compute the correct motion parameters for the car layer based on a small region of the car due to the over-fitting problem. Therefore, we use a common tracking technique to predict the motion parameters based on the previous frames. If the region shrinks by some amount (say 20%) and the predicted motion parameters are much different than the

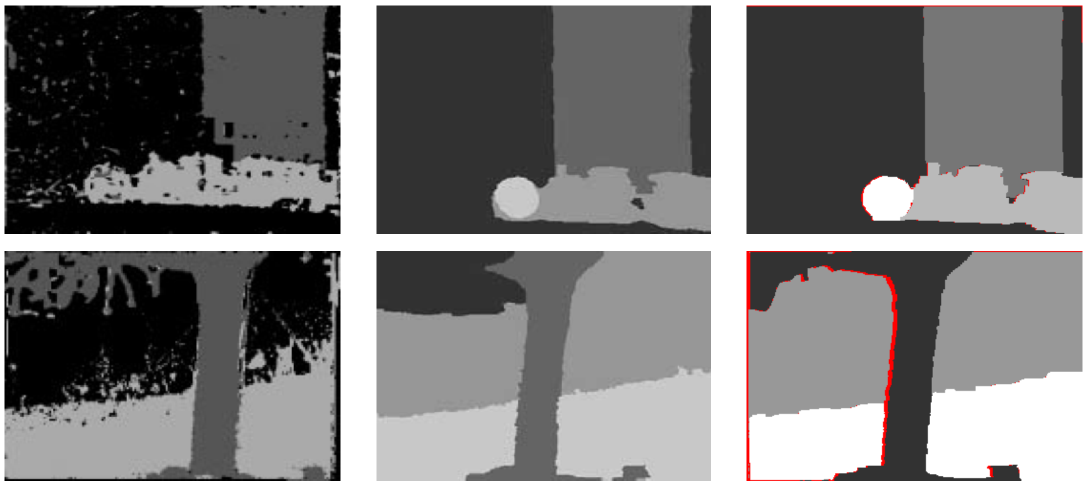


Figure 4.22: Comparison on mobile-calendar and flower-garden sequences with two previous methods. (a) Results of Ayer and Sawhney [4]. (b) Results of Ke and Kanade [46]. (c) Our results. In our results, we can not only obtain the accurate layer segmentation, but also explicitly detect the occluded pixels (red pixels). *Note*: (a) and (b) are reproduced from papers [4, 46] respectively.

new estimated parameters, we keep the predicted parameters to perform the segmentation. The results are shown in Figure 4.24.

Figure 4.25 shows a scene, box-card, with multiple layers, which was also taken by a hand-held moving video camera. In order to align the imagery, we have to segment this video clip into four planar layers such that each layer shares one projective transformation. Due to the apparently projective transformation and large ego-rotation of the camera, the affine transformation cannot fully capture the transitions between the consecutive frames. Instead

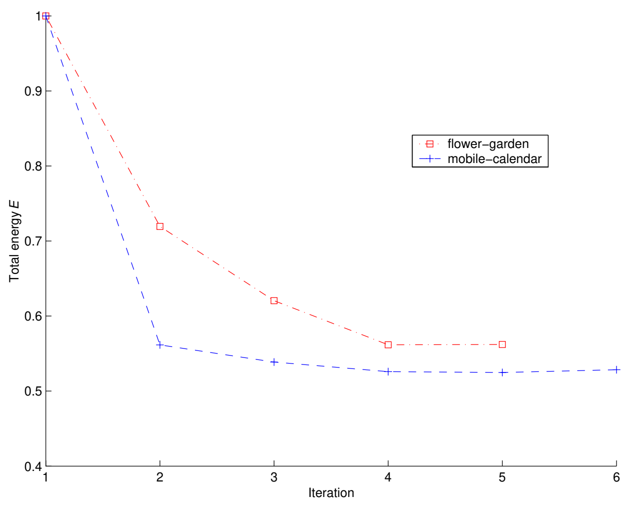


Figure 4.23: Energy reducing for two standard sequences. Initially, we assign the occlusion label to all the pixels in our graph model. In each iteration, one real label is selected as α label. After several iterations, an approximately global solution is found in both cases.

of the affine transformation used before, we used the Levenberg-Marquardt method [88] to compute the homography transformation for each layer to compensate for the projective deformation. It is clear that we have obtained good results for this sequence as shown in Figure 4.25.

Since our occlusion order constraint is based on the assumption that the moving object is not thin and not moving fast. If the object is thin or moving back and forth randomly, the occlusion may not have a temporal order. In this case, the segmentation around this object may not be accurate as the tree branch in flower-garden sequence(Figure 4.21). Nevertheless, in a real video sequence, it is rare to see an object moving randomly. In a short period (3-5 frames), a linear approximation of the object movement is a good choice for most of vision

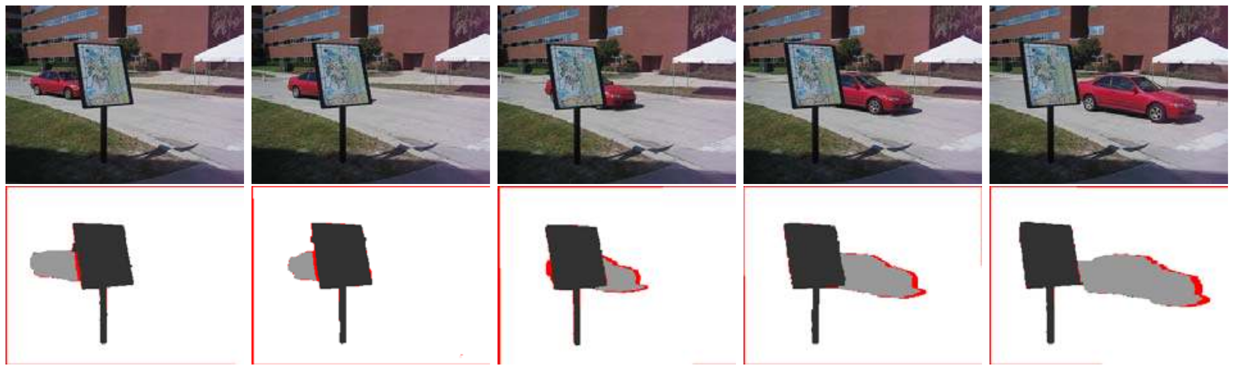


Figure 4.24: Segmentation results for the car-map sequence. Top: Several frames from the sequence. Bottom: The segmentation results, where the layers are accurately extracted even though the most parts of the moving car are occluded in some frames. There are three layers corresponding to map, car, and background building respectively.

applications, such as object tracking. Consequently, as we expected, the experiment results with this constraint are apparently improved when compared with those without using this constraint (Figure 4.19).

In all of our experiments, once the layer descriptions are extracted, the average computational time for one frame segmentation is less than 30 seconds on Pentium IV 2.0G. *Note: All of our results are also available on our web site [36].*

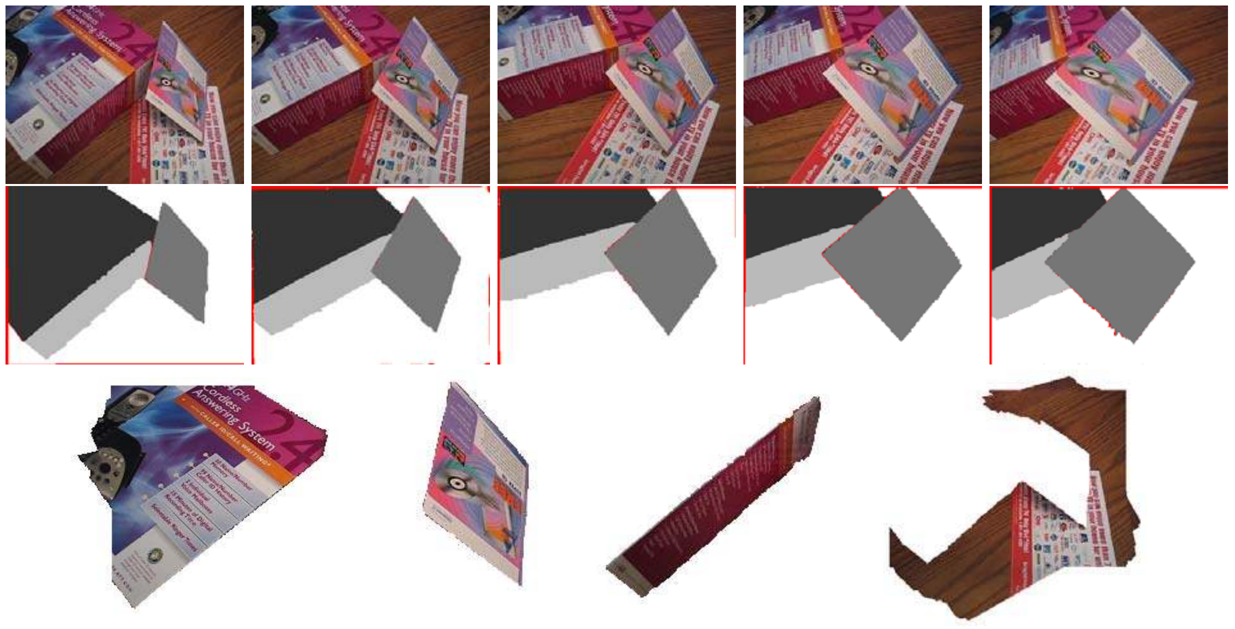


Figure 4.25: Segmentation results for the box-card sequence using projective transformation. Top: Several frames from the sequence. Middle: The segmentation results using homography transformation. There are four layers corresponding to two sides of the box, card, and desk respectively. Bottom: The mosaics of four layers after registering all frames on the first frame.

4.5 SUMMARY

In this chapter, we presented an effective method to extract robust layer descriptions and to perform an accurate layer segmentation for image sequences containing 2-D motion (affine or homography). Our contributions consist of: (1) Initial layer descriptions by integrating the level set representation into the graph cut method to obtain gradually expanding seed regions. (2) Using the occlusion order constraints, we successfully combine multiple frames to compute accurate and consistent layer segmentation and explicitly detect the occluded pixels, which has not been done before.

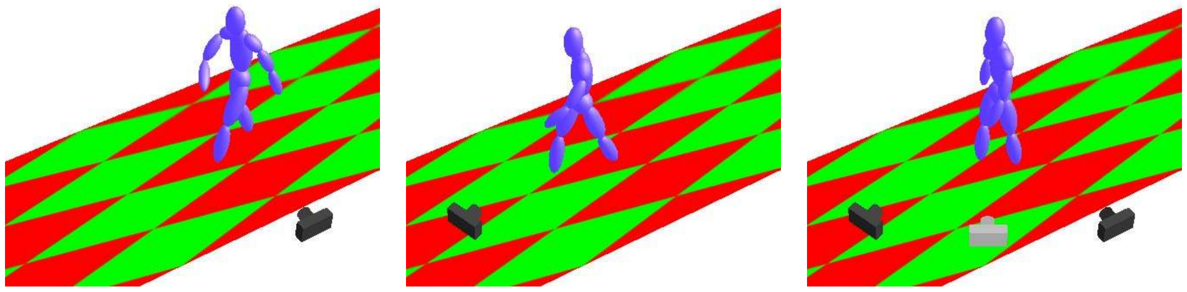
Furthermore, we have also successfully applied our segmentation framework on two related applications: layer based video registration [111, 112] and video completion [119].

CHAPTER 5

TWO-FRAME DYNAMIC VIEW SYNTHESIS

Dynamic view morphing deals with the scenes containing moving objects in presence of camera motion; and the moving object can be rigid or non-rigid object as shown in Figure 5.1. In our scenario, only two reference views are taken at different times from different viewpoints. During this period, the objects move, deform, or interact with the environment. Our aim is to synthesize the intermediate views for the gap between two original views. In this chapter, we only consider dynamic view with rigid objects, and prove the existence of the in-between views based on geometric properties between the 2D images and the 3D real world.

In section 5.1, we introduce the static view morphing as preliminary. Next, we analyze the relationship between moving and static cameras in section 5.2. After that, we show how to extend view morphing algorithm into dynamic view with rigid and non-rigid moving objects.



(a) (b) (c)

Figure 5.1: A typical dynamic scenario. In this scene, a person is walking along a straight line. (a) and (b) are two reference views taken at different times from different viewpoints. (c) is the in-between view synthesized from the reference views. The camera shown in gray is the virtual camera, which is on the line connecting the two original cameras.

5.1 BACKGROUND: STATIC VIEW MORPHING

In this section, we introduce the simple case of view interpolation–static view morphing [73]. In this case, two pictures are taken of a static scene from different viewing direction. Since the scene is static, any pair of corresponding points on two images are from the same 3D points in real world as shown in Figure 5.2.

Here $P = [X \ \ Y \ \ Z \ \ 1]^T$ is the real world point, $p = [x \ \ y \ \ 1]^T$ is the image point, and the fundamental matrix between these two images can be represented as $p_1^T F p_2 = 0$.

In this chapter, we use a 3×4 matrix to represent perspective cameral model as Equation 5.1.

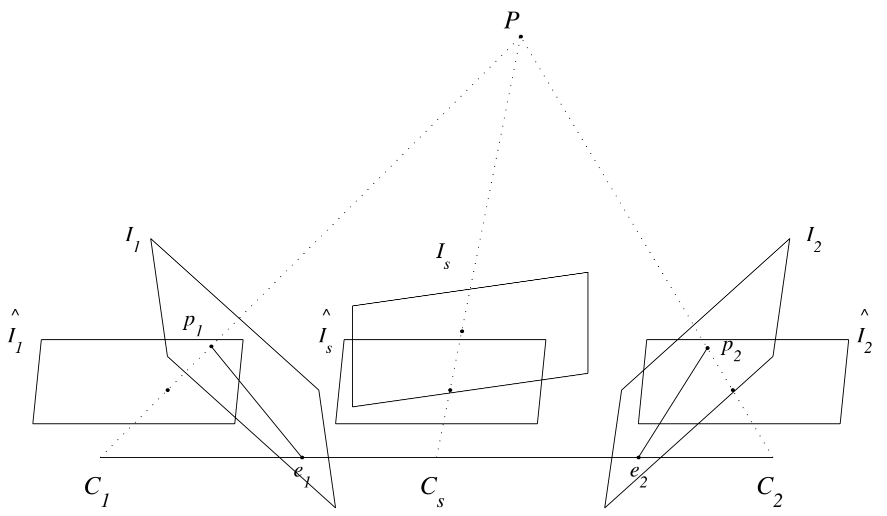


Figure 5.2: C_1 and C_2 are the camera centers of images I_1 and I_2. p_1 and p_2 are the projection of 3D point P on I_1 and I_2. $\hat{I_1}$ and $\hat{I_2}$ are the rectified images of I_1 and I_2 after prewarping.

$$\begin{aligned} M &= \begin{bmatrix} -fr_{11} & -fr_{12} & -fr_{13} & fR_1^TC \\ -fr_{21} & -fr_{22} & -fr_{23} & fR_2^TC \\ r_{31} & r_{32} & r_{33} & -R_3^TC \end{bmatrix} \\ &= [H|HC], \end{aligned} \tag{5.1}$$

where $H = \begin{bmatrix} -fr_{11} & -fr_{12} & -fr_{13} \\ -fr_{21} & -fr_{22} & -fr_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$, C is the camera center.

The images I_1, I_2 are the perspective views projected by matrices $M_1 = [H_1| - H_1C_1]$ and $M_2 = [H_2| - H_2C_2]$. The epipoles e_1 and e_2 are the projection of C_2 and C_1 on image I_1 and I_2.

$$e_1 = M_1 \begin{bmatrix} C_2 \\ 1 \end{bmatrix} = H_1C_2 - H_1C_1 \tag{5.2}$$

$$e_2 = M_2 \begin{bmatrix} C_1 \\ 1 \end{bmatrix} = H_2C_1 - H_2C_2 \tag{5.3}$$

And the fundamental matrix $F = [e_2]_\times H_2$ [53].

The original images should be reprojected to new projection plane to construct parallel view $\hat{I_1}$ and $\hat{I_2}$. The projection matrices of $\hat{I_1}$ and $\hat{I_2}$ can be given by $\hat{M_1} = [I|C_1]$ and $\hat{M_2} = [I|C_2]$.

For any pair of parallel views, the corresponding points $\hat{p_1}$ and $\hat{p_2}$ are in the same scanline and have the same y value ($y_{\hat{p_1}} = y_{\hat{p_2}}$). One of the fundamental matrix $\hat{F}$, satisfying $\hat{p_1}\hat{F}\hat{p_2} = 0$, can be expressed as

$$\hat{F} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{5.4}$$

Image I projected by matrix $M = [H| - HC]$ and the rectified image $\hat{I}$ projected by matrix $\hat{M} = [\hat{H}| - \hat{H}C]$. Since $p = MP$ and $\hat{p} = \hat{M}P$, we can get

$$\begin{aligned}\hat{p} &= [\hat{H}| - \hat{H}C]P \\ &= \hat{H}H^{-1}[H| - HC]P \\ &= \hat{H}H^{-1}p \end{aligned} \tag{5.5}$$

When $\hat{H}_1 = I$ and $\hat{H}_2 = I$, $\hat{p_1} = H_1^{-1}p_1$ and $\hat{p_2} = H_2^{-1}p_2$. The fundamental matrix $\hat{F}$ can be rewritten as

$$\hat{F} = H_1^T F H_2. \tag{5.6}$$

In Seitz's paper [75], H_1 and H_2 can be computed by applying 3D rotation and 2D affine transformation to prewarp the images into parallel views.

After prewarping, the rectified images can be used for linear interpolation of corresponding points p_1 and p_2 by Equation 5.7.

$$p_s = (1 - s)p_1 + sp_2 \tag{5.7}$$

The last step of view morphing is to find the postwarping path to warp the new image to correct viewing space. In Seitz's method, they selected four control points on reference images, and used linear interpolation to generate the final position for control points. By these control points, the perspective postwarping transformation can be computed for warping.

If the camera motion is translation or has a small rotation, this method can work very well. In Seitz's paper [73], they illustrated an example depicting two reference views where the

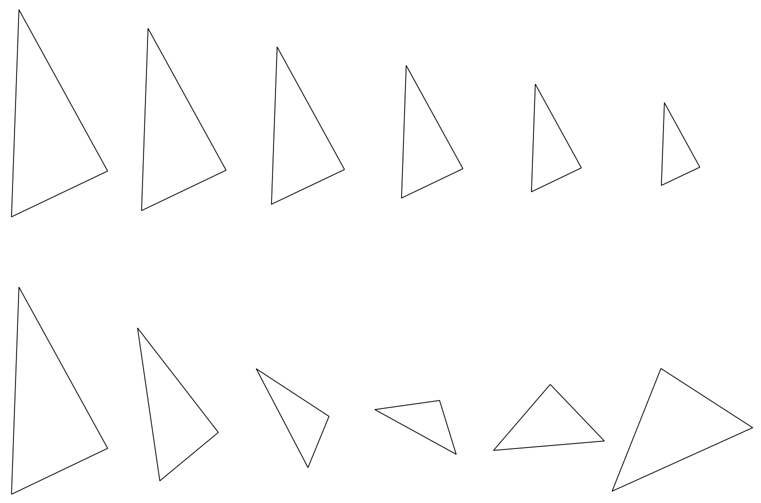


Figure 5.3: Interpolation comparison. Top: linear interpolation of the leftmost triangle into the rightmost triangle when the motion only including translation and scaling, where the shape of the object can be maintained. Bottom: linear interpolation when the motion including rotation, translation and scaling, where the shape of the object can be deformed and unpreserved.

posture of a face is changed. In general, small changes (including translation or rotation)of the surface features are reasonably captured due to the morphing process.

However, if the scenes contain large amounts of movement, static view morphing will generate unsatisfactory results. Figure 5.3 shows that the distortion will be out of control and unacceptable when the rotation is big. In this chapter, we use the least distortion method to improve the postwarping path in the following sections.

5.2 FIXED CAMERA V.S. MOVING CAMERA

In our scenario, two pictures are captured by uncalibrated cameras at different viewpoints and time. During this period, the intrinsic and extrinsic parameters of camera are changed with time.

In perspective camera model, the extrinsic parameters are comprised of rotation matrix R and translation vector T. The relation between the coordinates of a point P in the world and camera frame, P_w and P_c respectively, is

$$\begin{aligned} P_c &= R(P_w - T) \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} (P_w - T). \end{aligned} \tag{5.8}$$

The matrix for extrinsic parameter can be written as

$$M_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & -R_1^T T \\ r_{21} & r_{22} & r_{23} & -R_2^T T \\ r_{31} & r_{32} & r_{33} & -R_3^T T \end{bmatrix}. \tag{5.9}$$

For a fixed camera, the extrinsic parameters are fixed, but the intrinsic parameters may still change. Instead of using moving camera model, we reformulate the scenario so that the camera centers are at the same fixed location and orientation.

For the translation case, the camera translation u can be compensated by adding $-u$ to the camera and every object in the scene. For the rotation case, the rotation θ also

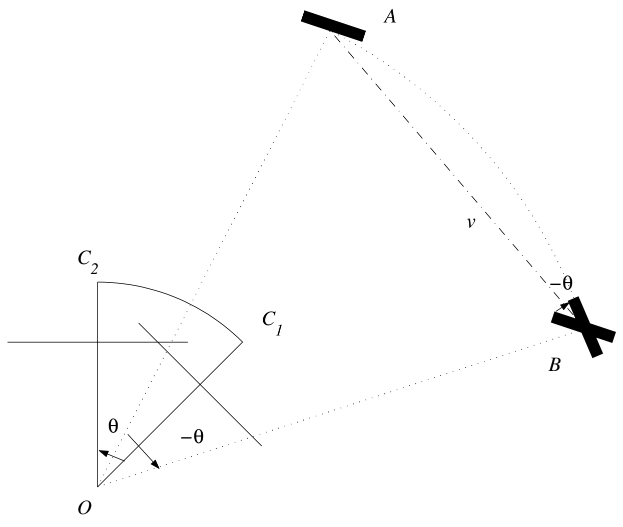


Figure 5.4: The orientation of the camera rotates around the optical center from OC_1 to OC_2, the rotation angle is θ. This rotation can be compensated by adding rotation $-\theta$ to the object in the scene with a translation v.

can be compensated by adding $-\theta$ to the camera, and every object rotates $-\theta$ around its own axes with some amount of translation v (as Figure 5.4), even though for each object the translation may be different. As we know, every camera motion can be decomposed into translation and rotation components. Therefore, for every element of the scene, the moving camera model can be converted into the fixed camera model by adding rotation and translation to it.

This reformulation not only converts moving camera model to the fixed camera model, but also guarantees that the virtual trajectory of each object is linear in translation and rotation angle. This is useful to simplify the analysis of the dynamic view interpolation.

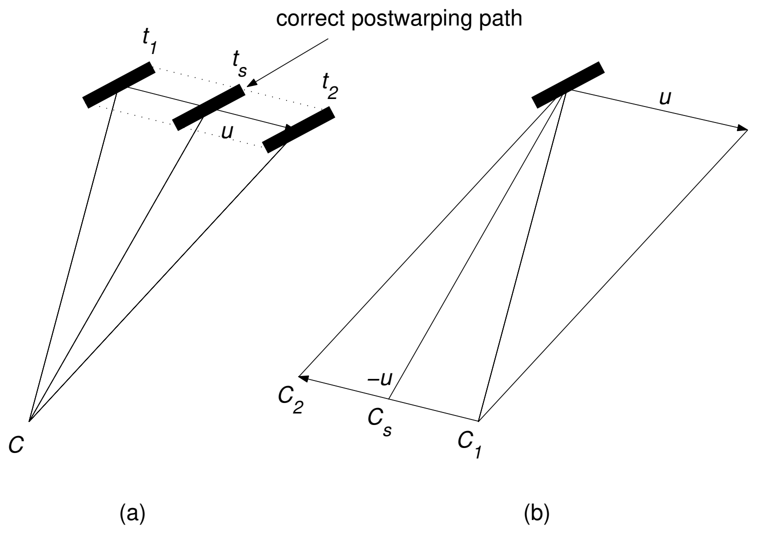


Figure 5.5: Translation case. (a) Original scenario, the object translates along the direction u and camera C is fixed. (b) Equivalent scenario, the camera translates along the inverse direction $-u$ and the object is fixed. The camera centers C_1 at t_1, C_s at t_s, C_2 at t_2.

5.3 DYNAMIC VIEW INTERPOLATION WITH RIGID OBJECTS

5.3.1 DYNAMIC VIEW INTERPOLATION WITH TRANSLATION

In dynamic scenario, beside the camera motion, the object also can translate or rotate during the period of capturing two pictures. In this section, we discuss the dynamic view interpolation with translating object.

First, we consider a simple case named fixed camera model in Figure 5.5.a: the camera C is fixed, and two pictures I_1 and I_2 are taken at time t_1 and t_2, and an object moves along the straight line u with constant speed v.

If we don't consider the background, Figure 5.5.b gives the equivalent case for this scene. The object is fixed, and the camera moves along the inverse direction $-u$ with a constant speed v, and two pictures I'_1 and I'_2 are taken at time t_1 and t_2. After comparing these two images, we will find that the objects have the same location and shape in the images I_i and I'_i.

Next, we extend the scenario to a general case, where the cameras are not at the same position. The translation of camera can be denoted as q. If we add $-q$ to the translation of each element in the scene, the original scenario can be converted to the fixed camera model. As a result of the conversion, the object's translation is changed from u to $u - q$ in fixed camera model.

In general, if the camera has only translation motion, we can always convert any dynamic scene to the fixed camera model by subtracting out the camera displacement.

If the orientation of the camera is changed during time Δt, the case will go beyond the translation model. We will discuss it in the next section.

5.3.2 DYNAMIC VIEW INTERPOLATION WITH ROTATION

Similarly, we consider the fixed camera model for the rotation case as Figure 5.6.a. In this model, the object rotates by angle θ around axis ω with a constant speed, the camera C is fixed, and two pictures I_1 and I_2 are taken at time t_1 and t_2.

This original rotation scenario can also be converted into an equivalent case (Figure 5.6.b), such that the object is fixed and the camera rotates by $-\theta$ around axis ω with a constant speed, and two pictures I'_1 and I'_2 are taken at time t_1 and t_2. Therefore, the objects also have the same location and shape in the images I_i and I'_i.

If the cameras are not at the same location, the motion of camera can be separated into rotation ψ and translation q. First, we consider the rotation angle, which can also be converted to the object's rotation by adding a rotation $-\psi$ and translation v_j to each object j in the scene. After that, the translation $-q$ can be applied on the objects as mentioned in the previous section.

Based on the above analysis, it can be stated that any dynamic scene intrinsically comprises several static scenes. Each static scene only includes one rigid object, and its fundamental matrix is determined by using the object's points. Therefore, there are several different fundamental matrices corresponding to several objects, including the background. Theoretically, if we can segment the scene into several layers and each of them only contain one object, we can apply the view morphing algorithm to the dynamic scene with the rigid objects.

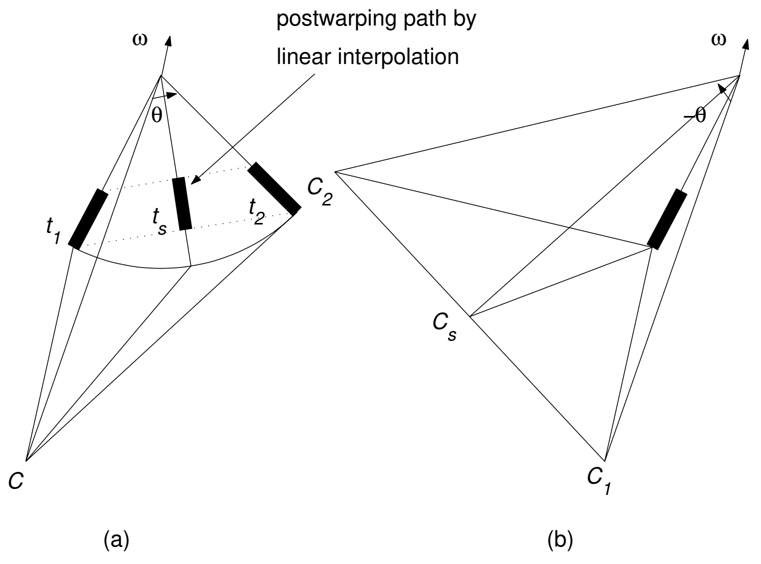


Figure 5.6: Rotation case. (a) Original scenario, the object rotates by θ around axis ω and camera C is fixed. (b) Equivalent scenario, the camera rotates by $-\theta$ around axis ω and the object is fixed. The camera centers C_1 at t_1, C_s at t_s, C_2 at t_2.

However, due to the rotation, the object shrinks and its shape cannot be maintained when the linear interpolation is used for postwarping (the dark intermediate shape in Figure 5.6.a). The distortion becomes bigger when the rotation angle increases.

In order to reduce the distortion and recover the correct postwarping path as much as possible, we use the least distortion method to minimize the error.

5.4 POSTWARPING BY USING LEAST DISTORTION METHOD

In shape blending or view morphing area, the task to determine the in-between shape or path is still a difficult problem. In this section, we introduce the use of the least distortion method to determine the postwarping path.

5.4.1 3D ROTATION

Since we only have two images without depth information for the pixels, it is impossible to recover the exact 3D rotation of the objects, even though the object's translation can be obtained accurately by the linear interpolation. From geometric property, given a projection plane ρ, the 3D rotation of an object can be decomposed into three components α, β, γ around the axis X, Y, and Z respectively. X and Y are parallel to x and y on the plane ρ, and Z is perpendicular to the plane ρ. The rotation transformation can be represented as

$$\begin{aligned} R &= R_z \cdot R_y \cdot R_x \\ &= \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \\ &= R_z \cdot B, \end{aligned} \tag{5.10}$$

where $B = R_y \cdot R_x$. Due to the loss of depth information, the rotation R_y and R_x cannot be recovered by using two images. But the rotation R_z around axis Z can be obtained by decomposing affine transformation using SVD.

5.4.2 LINEAR TRANSFORMATION

For any pair of three corresponding points $P_1\{p_{11}, p_{12}, p_{13}\}$ and $P_2\{p_{21}, p_{22}, p_{23}\}$ not on the same line, we can form a triangle for each set of points. There is an affine transformation represented by a matrix A and a vector T between two original shapes such that $P_2 = AP_1 + T$. Since the translation doesn't change the shape of the triangle, the shape will be determined only by the property of A.

If we change the parameters of A by a coefficient $s (s \in [0,1])$, we can get a different intermediate shape

$$P_s = A(s)P_1 \tag{5.11}$$

where $A(s)$ is the corresponding intermediate transformation at s. One possible representations for $A(s)$ based on linear interpolation can be expressed in Equation 5.12.

$$A(s) = (1-s)I + sA \tag{5.12}$$

However, using this linear transformation, the shape and area of the triangle cannot be preserved very well (Figure 5.7). In order to maintain the shape and area of the triangle

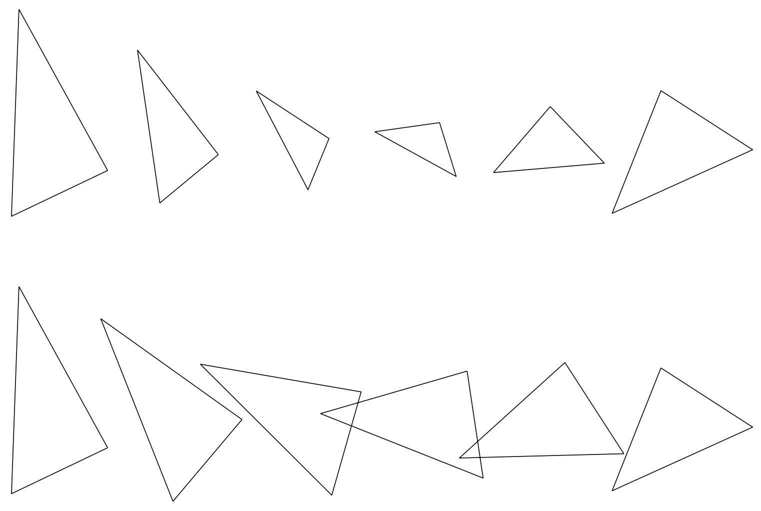


Figure 5.7: Two affine transformations. Bottom: using the least distortion method (Equation B.1), the triangle's area is kept constant. Top: using linear transformation (Equation 5.12), the triangle shrinks in the middle position. The original triangles are at the left and right sides.

during morphing, we use another representation of $A(s)$, which can be implemented by the least distortion method [1].

5.4.3 THE LEAST DISTORTION TRANSFORMATION

The least distortion method is based on the decomposition of an affine transformation. The basic idea is that an affine transformation can be decomposed into rotation matrix $R(\alpha)$ and scaling-shearing matrix C by singular value decomposition (SVD) [30].

$$A = SVD = S \begin{bmatrix} v_1 & 0 \\ 0 & v_2 \end{bmatrix} D, \tag{5.13}$$

where S and D are rotation matrices, and V is scaling matrix. Then, we can denote rotation matrix $R(\alpha) = SD$, and the scaling-shearing matrix $C = D^{-1}VD$.

$$\begin{aligned} A &= SVD = S(DD^{-1})VD = (SD)(D^{-1}VD) \\ &= R(\alpha)C = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} v_1 & v_h \\ v_h & v_2 \end{bmatrix}. \end{aligned} \tag{5.14}$$

After the decomposition, an affine transformation can be rewritten by multiplication of a rotation matrix R and a scaling-shearing matrix C. Comparing to Equation 5.14 and 5.10, we can find the rotation matrix R extracted from A, which is essentially R_z, one component of rotation matrix R. The remaining part C is a combination of R_x, R_y, and scaling components of the 3D transformation of the object. Based on Equation 5.14 , we can construct a new intermediate transformation A(s) as below.

$$\begin{aligned} A(s) &= R(s)C(s) = R(s)((1-s)I + sC) \\ &= \begin{bmatrix} \cos(s\alpha) & -\sin(s\alpha) \\ \sin(s\alpha) & \cos(s\alpha) \end{bmatrix} ((1-s)I + sC), \end{aligned} \tag{5.15}$$

where rotation matrix $R(s)$ is linearly changed by rotation angle $s\alpha$, and scaling-shearing matrix $C(s)$ is also controlled by scale s linearly.

Since the rotation component R_z of the object is directly projected on the projection plane, it is easier for the human eyes to detect. Using Equation 5.15 to separate R_z, and linearly interpolating this angle can efficiently improve the visual effect compared to other methods as shown in Figure 5.7.

Comparing Equation 5.15 and Equation 5.12, the linear interpolation is the special case of the least distortion transformation when $R(s) = I$. The least distortion transformation can linearly change the affine transformation not only by scale-shear matrix, but also by the rotation angle. Therefore, the triangle can control its orientation by rotation angle and shape by scaling-shearing matrix separately during transformation. Figure 5.7 compares the intermediate shapes obtained using the two transformations. Using the least distortion method, the shape and area of the triangle can be kept steady, and the distortion is minimized during the postwarping.

5.5 DYNAMIC VIEW SYNTHESIS WITH NON-RIGID OBJECTS

For non-rigid objects, we assume these objects can be separated into several parts, where each part can be considered as an approximate rigid object. Particularly, the human body can be reasonably assumed as an articulated object. Therefore, the segmentation of a human body or an articulated object into rigid parts is a preliminary step for view synthesis of the scenes containing non-rigid objects.

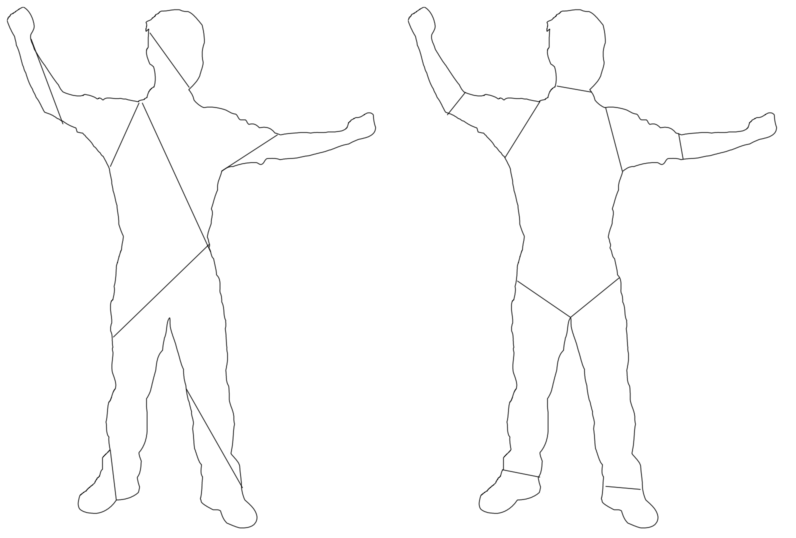


Figure 5.8: Decomposition for a human body. (a) random segmentation. (b) nature segmentation.

5.5.1 SEGMENTATION

Segmentation doesn't only break the non-rigid object into small rigid pieces, which can be aligned effectively by our approach, but also it provides the boundary connection constraint, which is crucial to determine the postwarping path.

A non-rigid object can be decomposed in many different ways as shown in Figure 5.8. However, the ideal decomposition for non-rigid object is the natural segmentation as shown in Figure 5.8.b. Since we focus on humans as the non-rigid objects in my thesis, this assumption is basic and important. In order to obtain an ideal decomposition of the non-rigid objects, some requirements must be satisfied:

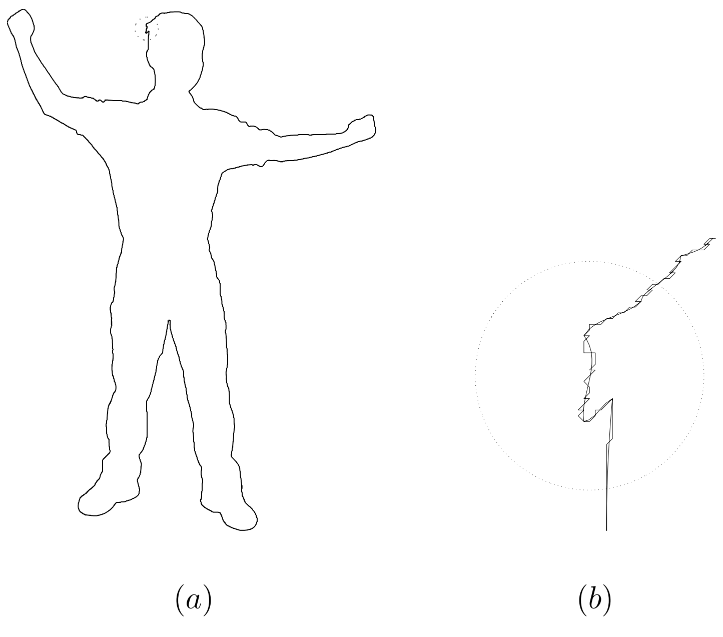


Figure 5.9: The boundary points are fitted by Hermit cubic curve.

- the segmentation should correspond to the natural body parts of an object.
- the decomposition should be invariant under translation, rotation, and scaling.
- the decomposition should be computable.

In Zhao's dissertation [114], she suggested to use some constraints to decompose the shape by body properties, such as negative curvature minima, symmetry, etc [35].

In our approach, we use a Hermit cubic curve to fit the boundary and smooth the noise as shown in Figure 5.9. Next, the negative curvature minima are selected as candidates for cutting points. Using a similar symmetric algorithm, the human body can be segmented into nature parts. Finally, we also apply a refinement step to use the information of two images to improve the quality of the segmentation.

5.5.2 BOUNDARY CONNECTION CONSTRAINT

After segmentation, the object is separated into several parts, which are connected by a new boundary between the parts. In order to preserve the connection, we select four or more control points to outline each rigid part from the boundary points.

Figure 5.11 shows that an arm is segmented into three parts: lower arm, upper arm, and shoulder. The selection of control points is determined by the following conditions:

- The points are selected close to the object's boundary.
- The points can be located at the joint between the parts.
- The polygon constructed by the points can describe the part's shape effectively.

Each set of control points can form a polygon, and any pair of neighboring parts can share several points at the boundary. First, we consider a simple case: using four points to outline a rigid part, which can form a quadrilateral (Figure 5.10). The quadrilateral can be decomposed into two triangles along one diagonal, which share the points p_1 and p_2 as shown in Figure 5.10.a. If we apply the least distortion transformation to the two triangles independently, the two triangles don't share the boundary point any more. In order to maintain boundary connection property, we merge them to generate new points by averaging the points from two triangles as shown in Figure 5.10.b. However, this results in some shrinking along the other diagonal. Therefore, we decompose the quadrilateral along the two diagonals, and get four triangles instead. Then, we apply the least distortion

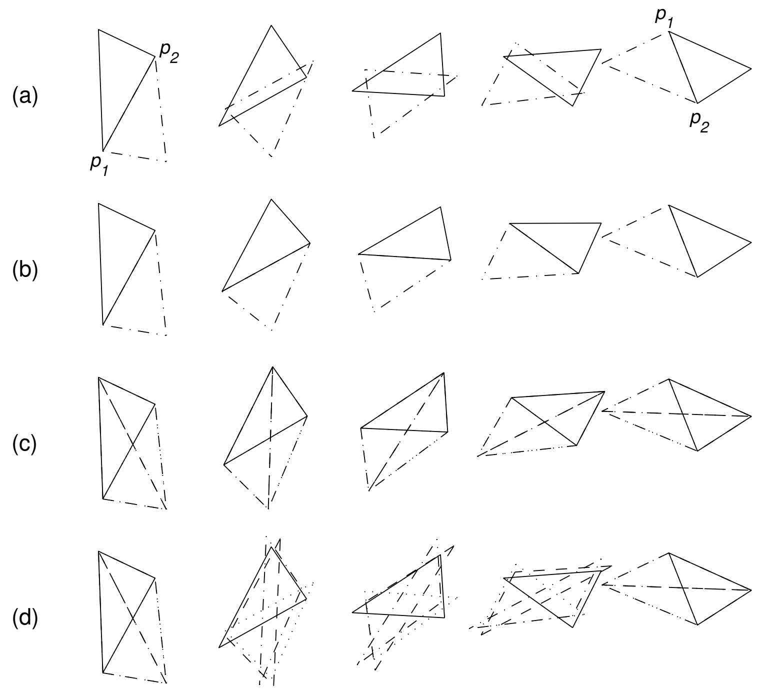
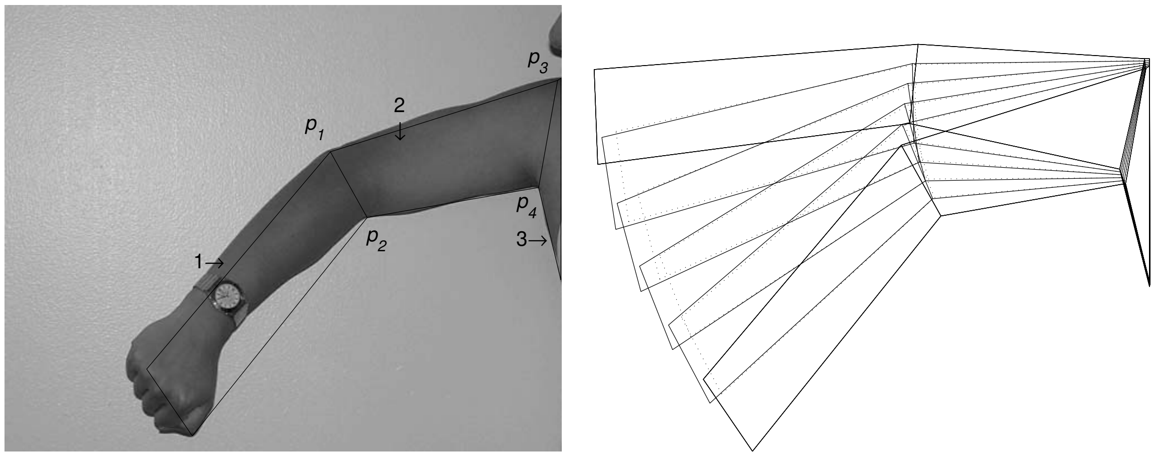


Figure 5.10: Boundary connection constraint for a quadrilateral. (a) Using least distortion method only. (b) Using boundary connection constraint along one diagonal. (c-d) Using boundary connection constraint along two diagonals.

transformation to each pair of triangles independently (Figure 5.10.d), and merge the points from the same source into one point by averaging (Figure 5.10.c).

For more complicated cases, the points are not only shared in one rigid part, but are also shared between two neighboring parts. Figure 5.11.a shows a complicated case for a non-rigid object: a human arm moves from lower to upper part of the image. In this procedure, there are two rotations, one at the elbow and the other at the shoulder, and some apparent deformations also happen at the joints. We separate the images into lower arm (part 1),



(a) (b)

Figure 5.11: Arm morphing. (a) Boundary connection constraint. (b) Postwarping path by least distortion method (solid line) and using linear interpolation (dot line).

upper arm (part 2) and body (part 3), and assume each part is rigid object to some extent. The points p_1 and p_2 are shared by part 1 and 2; p_3 and p_4 are shared by part 2 and 3. In order to get new point p'_i, first we transform all triangles $T_{i_1}, T_{i_2}, \cdots, T_{i_n}$ including point p_i, and then calculate corresponding $p_{i_1}, p_{i_2}, \cdots, p_{i_n}$ for each triangle.

$$p'_i = \frac{1}{n} \sum_{k=1}^{n} p_{i_k}$$

Figure 5.11.b shows the interpolation path obtained by two algorithms. The dashed lines show the paths generated by linear algorithm; the solid lines show the paths generated by the least distortion algorithm using connection constraint. It is obvious that the linear algorithm causes the shrinking of the arm during the interpolation when the object has rotating motion. Figure 5.12 shows the in-between views generated by the two reference views.

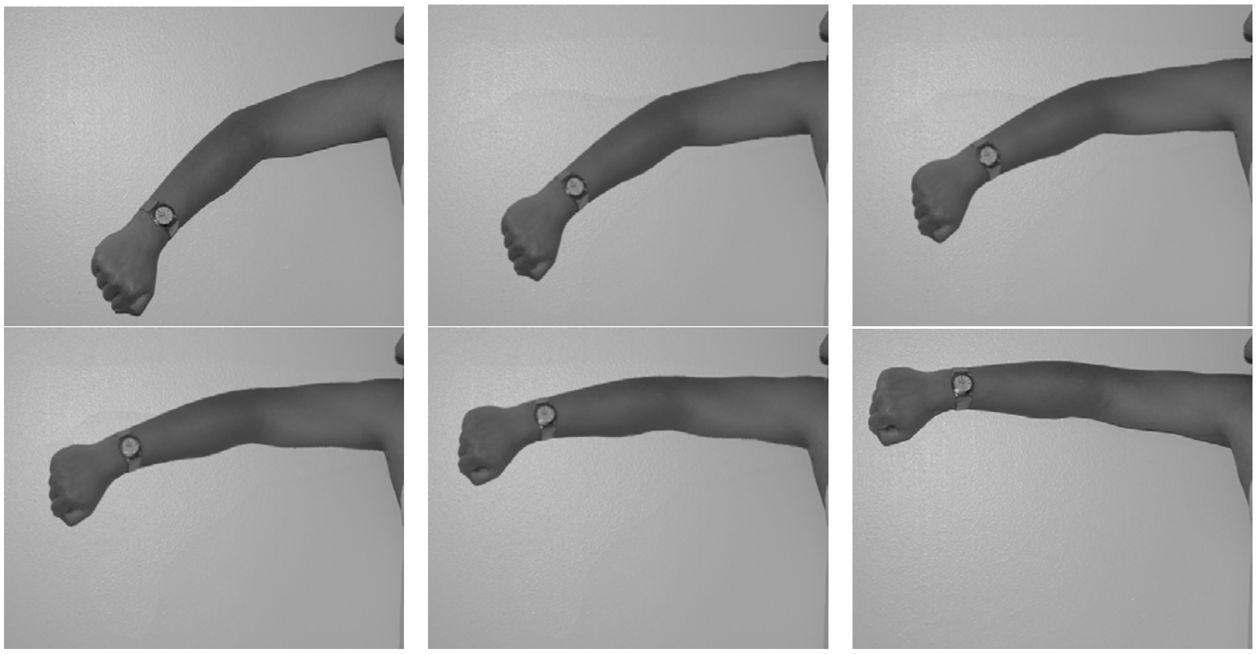


Figure 5.12: Arm morphing. The lower arm rotates round the elbow, the upper arm rotates round the shoulder, and some apparent deformations can be seen at the joints. The first and last images are reference views; the middle images are in-between views.

5.6 EXPERIMENTS

Figure 5.13 shows the images of a box object, which is rotating around its symmetric axis and translating from the right to left. We segment the image into several layers; each layer contains one moving object. Since the object has a big rotation motion, some parts are visible in both views, other parts are only visible in the left view, and the rest are only visible in the right view. So we segment the object into different views, even though they have the same motion. In order to restore the relationship of these parts, we use their common boundary points to merge part 5.13.d with 5.13.c, and part 5.13.h with 5.13.g. Since the control points

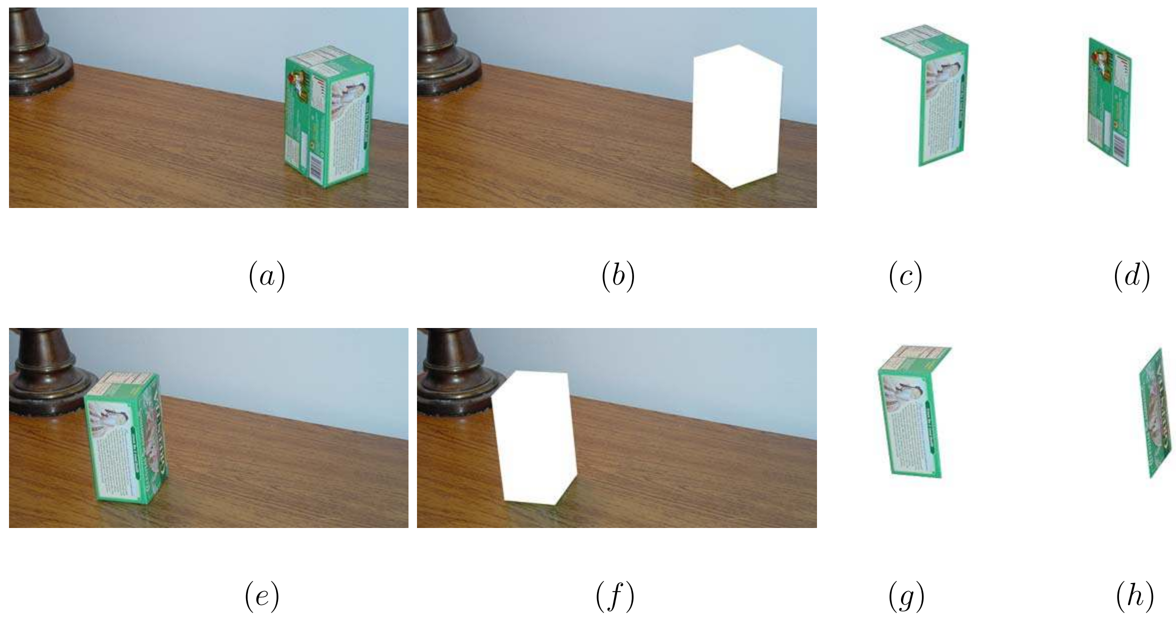


Figure 5.13: Segmentation of two original views of a box(a and e). (b)(f) background, (c)(g) the first part of the rotating object, (d) the second part of the rotating object, (h) the third part of the rotating object.

at the boundary can form a polygon or triangle, we use its normal to determine the visibility of the parts. The final morphing results are shown in Figure 5.14. Due to segmentation, a blank area is left on the background. The blank area of one image can be filled by the pixels of the other image from the same source. However, if illumination conditions for two reference views are different, the shadows will be observed in the intermediate views.

Figure 5.15 illustrates a dynamic view synthesis for human movement. In this experiment, a person is moving left to right with different postures, and the reference pictures are taken from two view positions. For every part of the person, there are different rotations and

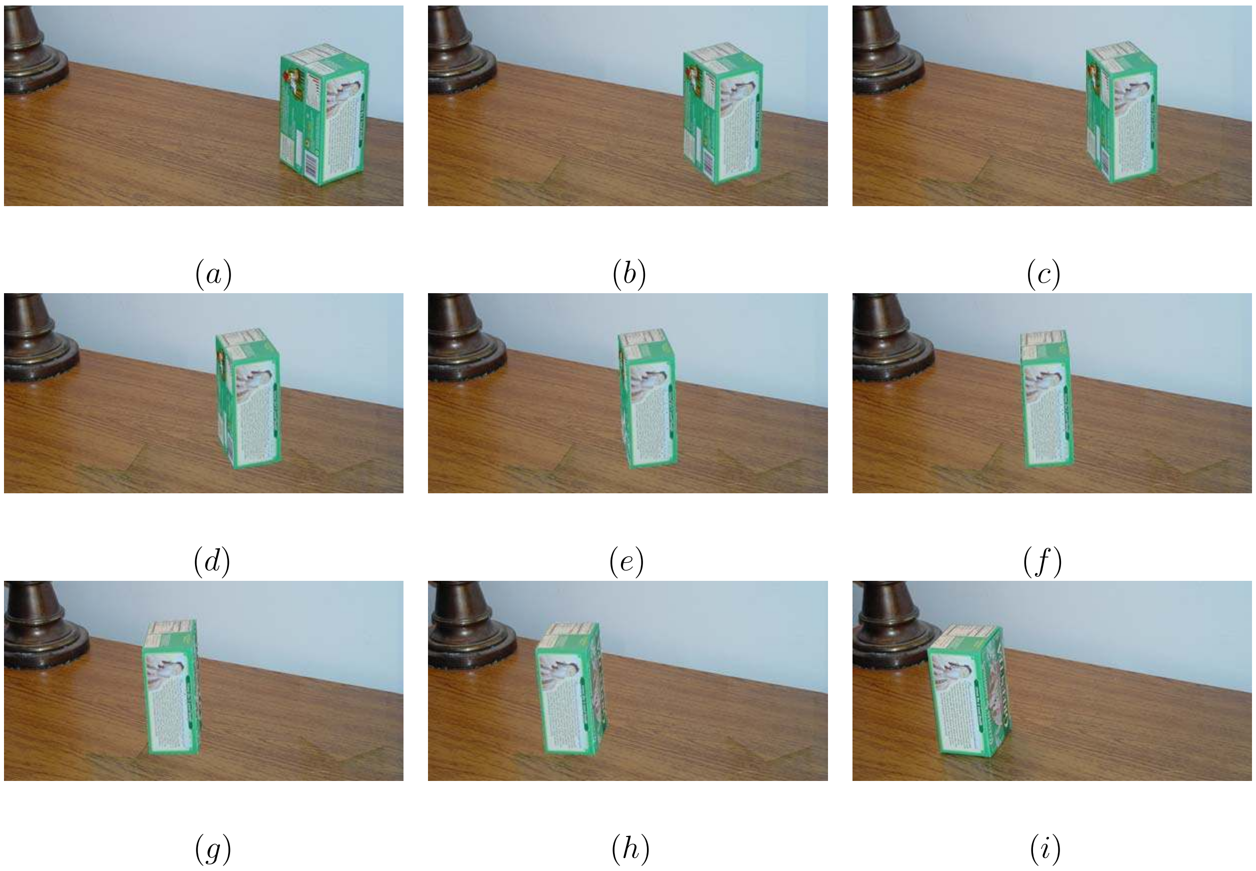


Figure 5.14: Dynamic view morphing with rotation. (a) and (i) are reference images. Since the object has rotation motion, one side of box is occluded by itself in (a) and (f), another side of box is occluded by itself in (g) and (i). Our approach can correctly restore the geometric property in the intermediate views (b)-(h).

translation. In order to obtain better results, we segment the person into eight layers: head, body, two upper arms, two lower arms, and two legs (left and right). For each part, we use view morphing algorithm to obtain a new rectified image, and then use least distortion method and boundary connection constraint to determine the postwarping path to get the intermediate view as shown in Figure 5.15. The results are also available at our web site [38].

5.7 SUMMARY

In this chapter, we successfully extend dynamic scene synthesis into the rotation case, where the rigid objects can move in different directions and orientations. Since the problem of 3D reconstruction from two 2D images without depth information is quite complex, it is almost impossible to exactly recover the object's position for postwarping due to rotation. However, our approach provides a unique and reasonable solution for the dynamic scene. The least distortion method fully recovers one rotational component, R_z, and partially compensates the remaining two rotational components R_x and R_y by scaling-shearing matrix. It efficiently avoids the shrinking problem at view morphing in the presence of rotation motion, and generate more realistic viewing results.

Furthermore, we generalized dynamic scene synthesis for scenes containing non-rigid objects. Our method can handle these cases mixed with dynamic rigid or non-rigid objects, including complicated objects such as humans. In our approach, if the non-rigid object can be segmented into several rigid parts, we can use the boundary information to connect adjacent parts during morphing. From only two reference views, we generate a series of continuous and realistic intermediate views without 3D knowledge.

After combining the least distortion method and boundary connection constraints with centroid adjustment, we can effectively compensate for the displacement due to rotation, and the amount of shape shrinking can be reduced. Moreover, due to strength of the view morphing algorithm, our approach can preserve the realistic 3D effect for each layer.

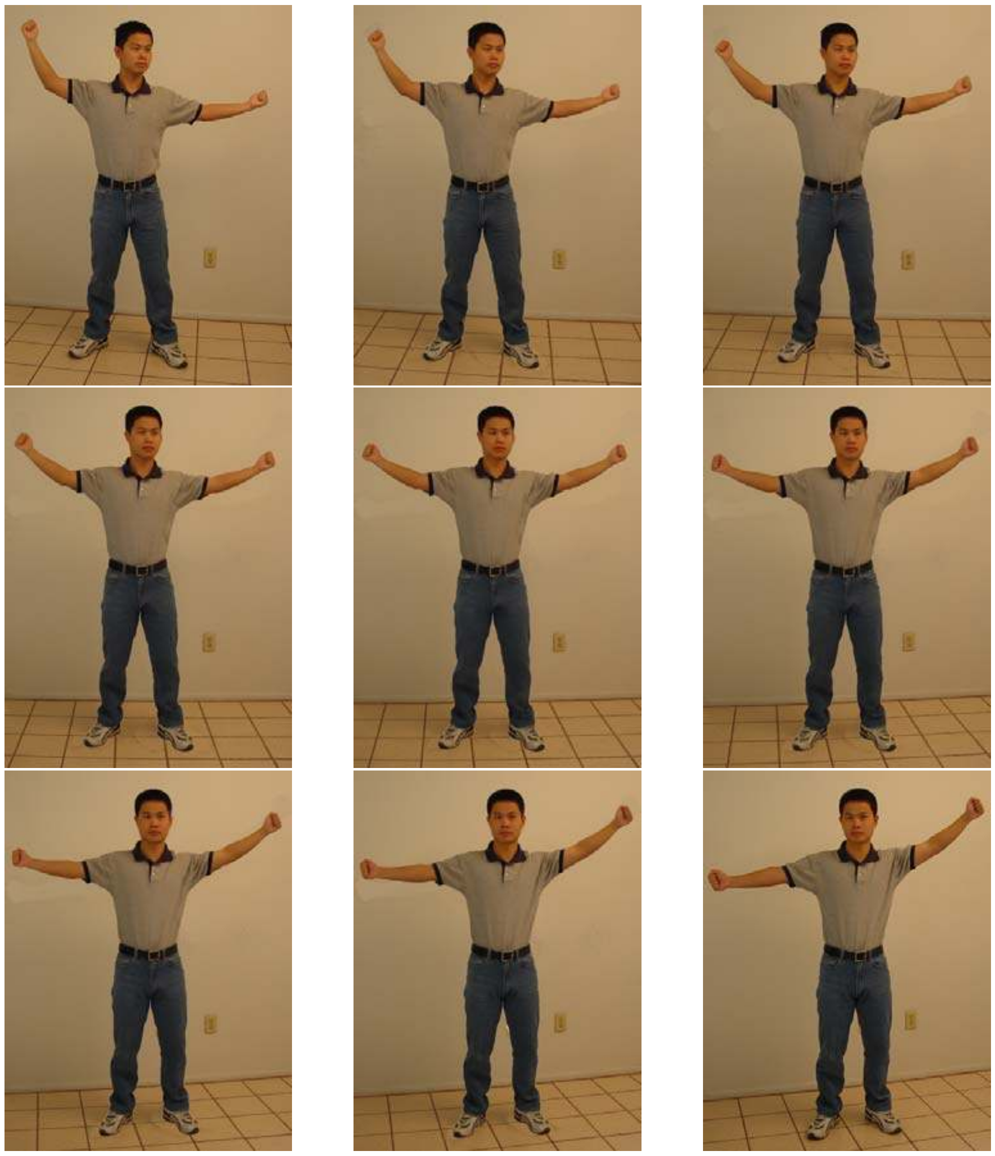


Figure 5.15: Dynamic view morphing for human motion. The first and last images are reference views. The person is moving from right to left with different postures, and the reference images are taken from different view points. Since the motion of the person is combined with multiple rotations and translations, we segment the person into several parts based on boundary connection constraint, and use our method to restore physically correct motion for the man between two reference views.

CHAPTER 6

TRI-VIEW MORPHING

In this chapter, we propose a novel technique to synthesize a virtual view in a 2D space. First, using three wide-baseline uncalibrated images, we automatically recover corresponding feature points and determine the epipolar geometry between each pair of images. Second, we refine these correspondences and extract the trifocal plane by trifocal tensor computation. Third, employing the user friendly GUI, a small number of feature lines are easily marked. Then, we automatically compute the disparity maps using our trinocular-stereo algorithm. As a result, we generate an arbitrary novel view located in the trifocal plane and easily navigate through the scene over a 2D space. After self-calibration of these three cameras, we can also accurately augment 3D objects in a virtual scene.

We demonstrate three applications of the tri-view morphing algorithm. The first one is 4D video synthesis, which can be used to fill in the gap between a few sparsely located video cameras to synthetically generate a video from a virtual moving camera. This synthetic camera can be used to view the dynamic scene from a novel view instead of the original static camera views. The second application is multi-view morphing, where we can seamlessly fly through the scene over a 2D space constructed by more than three cameras. The last one is dynamic scene synthesis using three still images, where rigid objects may move in any orientation or direction. After segmenting three reference frames into several layers, the novel views in the dynamic scene can be generated by applying our algorithm.

Our approach overcomes the limitation of previous view morphing methods, which are subject to a linear transformation between source and target views (or objects) and can generate only one visualization trajectory connecting these original frames. In particular, our approach makes the following contributions: First, we show that a virtual environment can be generated for 2D navigation based on only a few (three or four) wide baseline reference images. Second, using the trifocal plane extracted by the trifocal tensor, our morphing procedure can maintain geometrical correctness of synthesized novel views, which can also correctly augment with 3D objects. Third, we introduce three novel applications–4D video synthesis, multi-view morphing, and dynamic tri-view morphing, which can be efficiently implemented by using our tri-view morphing algorithm.

6.1 ALGORITHM OVERVIEW

The inputs to tri-view morphing are three wide-baseline uncalibrated reference images as shown in Figure 6.1.a-c. A series of novel virtual views (Figure 6.1.d-f) from any arbitrary position in the trifocal plane can be synthesized to navigate through the scene based on the

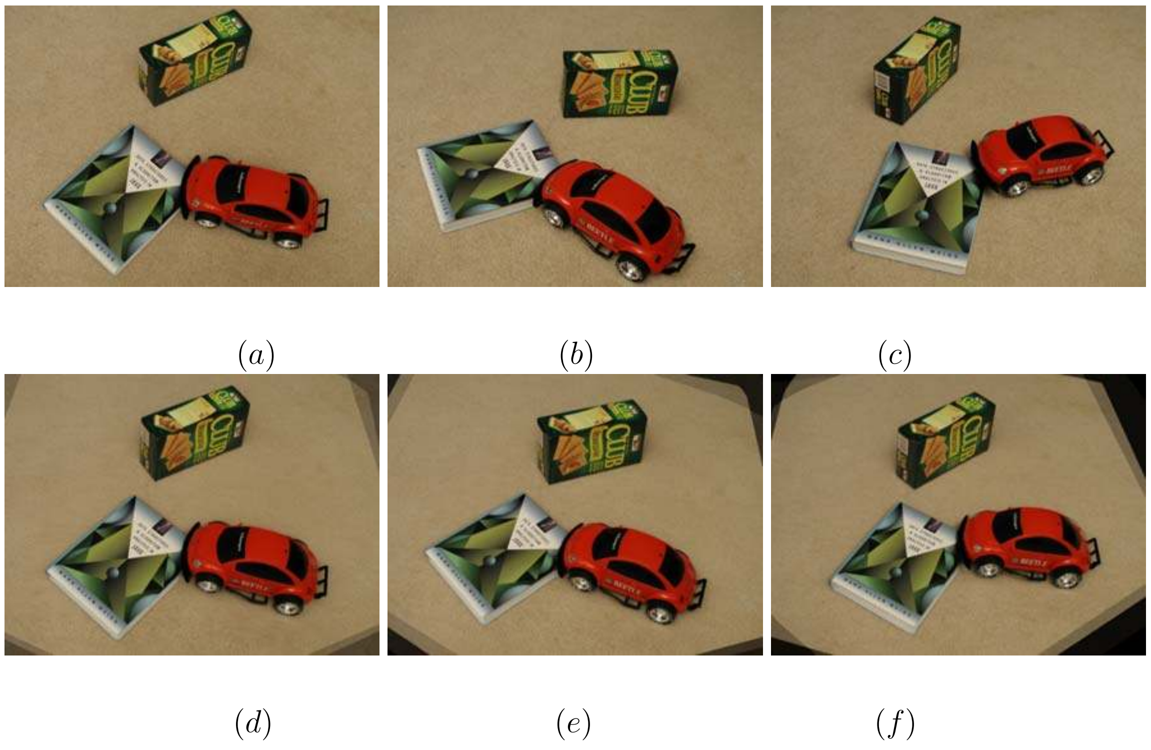


Figure 6.1: A typical tri-view morphing scenario. (a), (b) and (c) are three uncalibrated wide baseline reference images. (d-f) show a series of synthesized virtual views.

three original images without any knowledge of the scene. Our tri-view morphing algorithm is implemented using the following steps:

First, using a two-stage wide baseline matching algorithm method using edge corners [107], a number of corresponding points are automatically recovered to compute the fundamental matrix and epipolar geometry for each pair of reference images.

Second, a unique trifocal plane E is determined using their epipoles e_{ij} ($i, j \in \{1, 2, 3\}$ and $i \neq j$) as shown in Figure 6.2. Then, the three original images I_1, I_2 and I_3 are warped into a plane parallel to the trifocal plane to obtain rectified images $\hat{I}_1$, $\hat{I}_2$, and $\hat{I}_3$.

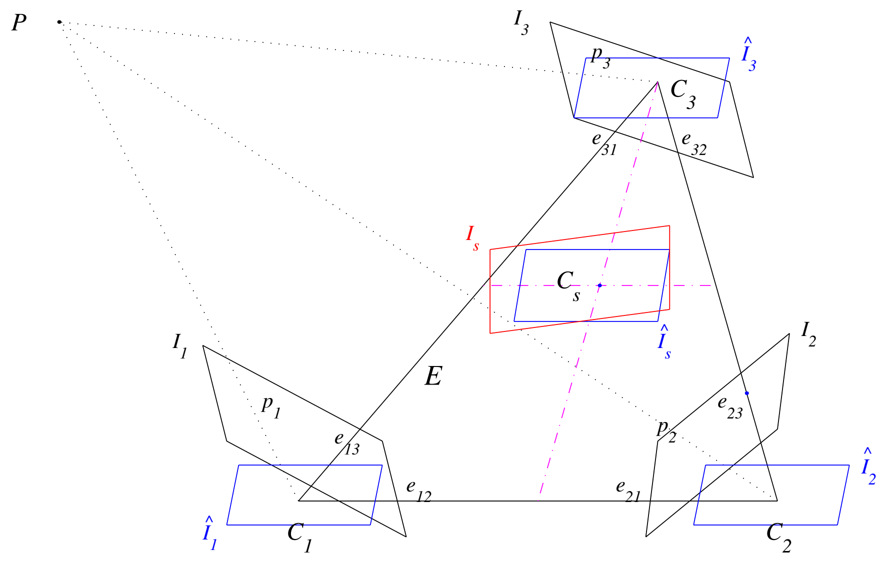


Figure 6.2: Tri-view morphing procedure. After automatically determining a focal plane E, which is constructed by three camera centers C_1, C_2, and C_3, three original images I_1, I_2, and I_3 are warped into parallel views $\hat{I}_1$, $\hat{I}_2$, and $\hat{I}_3$. The morphing image, $\hat{I}_s$, is blended by using the rectified images with correct disparity maps. The final image I_s at C_s is postwarped from $\hat{I}_s$ by using the 5-point postwarping scheme.

Third, based on the feature lines obtained by GUI, our feature-based trinocular-stereo algorithm is used to automatically compute the correct disparity map between each pair of rectified images.

Finally, a tri-view blending function is determined according to the viewpoint position. Following the perspective geometrical principles, the morphing image, $\hat{I}_s$, is obtained by combining the blending function with the disparity maps. Then, a 5-point postwarping scheme is used to project the morphing image to a proper final position (Appendix B).

6.2 CORRESPONDING POINTS AND TRIFOCAL PLANE

6.2.1 DETERMINING CORRESPONDING POINTS

In this chapter, we use our wide baseline matching method to determine corresponding points between each pair of images [107]. By decomposing the affine model into rotation matrix $R(\alpha)$, scaling matrix $S(\kappa)$, and stretch-shearing matrix E, the minimal image residue between two corner windows can be computed by a two-stage algorithm.

For each pair of images, first, we determine a large number of corners by edge-corner detector in two images respectively. The edge-corner detector can efficiently detect the corners located at the intersection of multiple edges. Then, a set of reliable corresponding points are found using the two-stage matching algorithm [107]. Figure 6.3 shows the corresponding points and several epipolar lines obtained by our algorithm for a pair of images.

6.2.2 DETERMINING THE TRIFOCAL PLANE

In order to get geometrically correct morphing (Figure 6.2), we first need to determine a trifocal plane E and warp the original images into parallel views. The focal plane is defined by the three camera centers. The epipole $e_{ij} = P_i C_j$, where P_i is the projection matrix of camera i, and C_j is the optical center of camera j. After computing the corresponding corners m'_1 and m'_2 between image I_1 and I_2, and corresponding corners m''_1 and m''_3 between

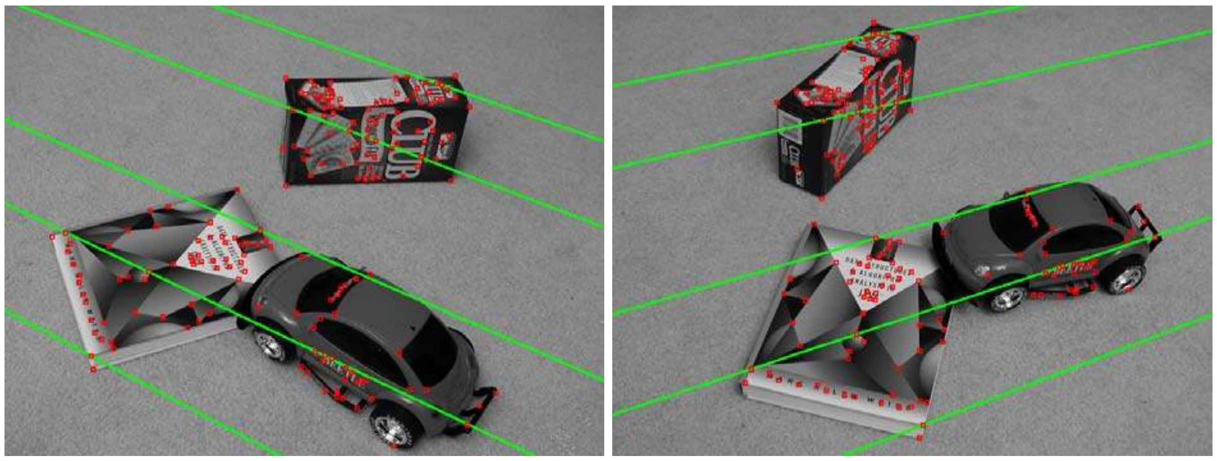


Figure 6.3: 182 corresponding points between two car images ((b) and (c) in Figure 6.1) were found. The green lines are epipolar lines.

image I_2 and I_3, we merge these two groups of corresponding points into one group and obtain new correspondences m_1, m_2, m_3, where $m_1 \in (m'_1 \cap m''_1)$.

Using Hartley and Zisserman's robust method, the outliers are eliminated and the trifocal tensor $T = [\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3]$ can be determined [41]. Then, the fundamental matrices F_{12}, F_{23}, and F_{31} can be extracted from the tensor. As a result, we can compute the trifocal plane by using the cross products of epipoles. For each camera, the trifocal plane normal is different. In camera C_1, the plane normal $N_{E1} = e_{12} \times e_{13}$; in camera C_2, $N_{E2} = e_{23} \times e_{21}$; in camera C_3, $N_{E3} = e_{31} \times e_{32}$.

After the trifocal plane is determined, the three original images are warped into parallel views using the prewarping algorithm (Appendix A) and employing the computed N_{E1}, N_{E2}, and N_{E3}. Also, the epipoles are projected into infinity. As a result of this warping,

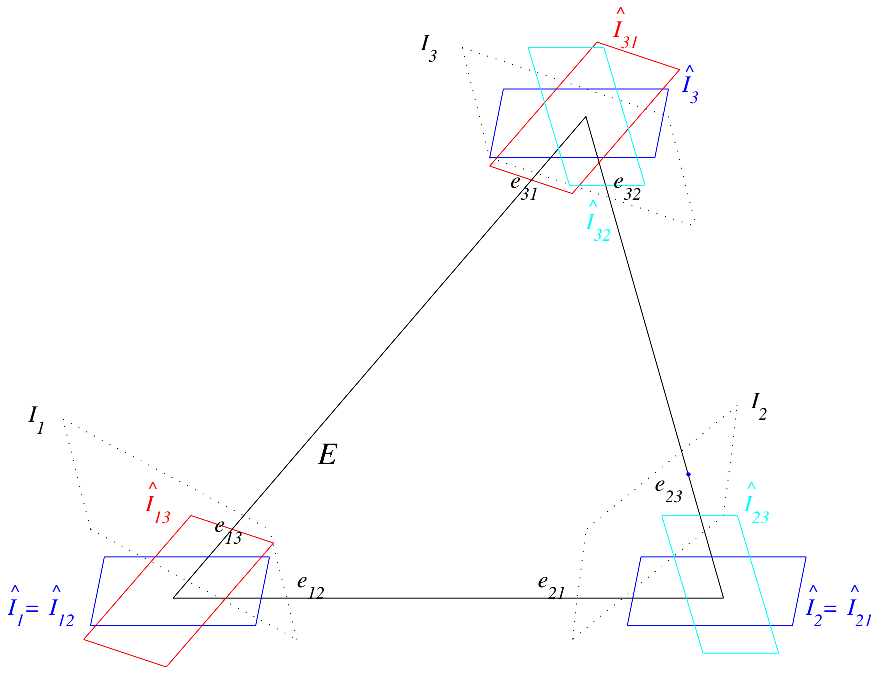


Figure 6.4: The x axis of rectified image $\hat{I}_i$ can be rotated to make it parallel to different epipolar directions. $\hat{I}_{12}$ and $\hat{I}_{21}$, $\hat{I}_{23}$ and $\hat{I}_{32}$, $\hat{I}_{31}$ and $\hat{I}_{13}$ are three corresponding rectified pairs.

all epipolar lines in the three rectified images are pairwisely parallel. Next, for each pair of rectified images, corresponding epipolar lines are rotated to make them parallel to scanline directions.

6.3 TRINOCULAR-STEREO

In this section, we propose a trinocular-stereo algorithm to compute the disparity map between each rectified image pair, which is based on a pixel-to-pixel dynamic scanline algorithm [62, 13]. In our algorithm, the dissimilarity function uses three image intensity differences (SAD: Sum of Absolute Difference) of corresponding pixels in three rectified pairs of images (Eq. 6.1).

Figure 6.4 shows that the rectified images can be rotated to make scanlines parallel to different epipolar lines. For example, image $\hat{I}_{12}$ is obtained when the x axis of $\hat{I}_1$ is rotated to make it parallel to e_{12}, and $\hat{I}_{13}$ is obtained when the x axis of $\hat{I}_1$ is rotated to make it parallel to e_{13}. Images $\hat{I}_{ij}$ and $\hat{I}_{ji}$ are called a corresponding rectified pair, since the correspondences of these two images are always located on the same scanline.

Consider the corresponding scanlines L and R in $\hat{I}_{12}$ and $\hat{I}_{21}$, which start from L_s and R_s, and end at L_e and R_e respectively. If the two pixels x_{12} and x_{21} match, the corresponding pixel, x_3, in original image I_3 is given as the intersection of the two epipolar lines:

$$\mathrm{x}_3 = (F_{31}\mathrm{x}_1) \times (F_{32}\mathrm{x}_2),$$

where x_1 is the coordinates of the pixel x_{12} in the original image I_1, and x_2 is the coordinates of the pixel x_{21} in the original image I_2. Let x_{13} be the projection of x_1 on $\hat{I}_{13}$, x_{23} be the projection of x_2 on $\hat{I}_{23}$, and x_{31} and x_{13} be the projections of x_3 on $\hat{I}_{31}$ and $\hat{I}_{13}$ respectively.

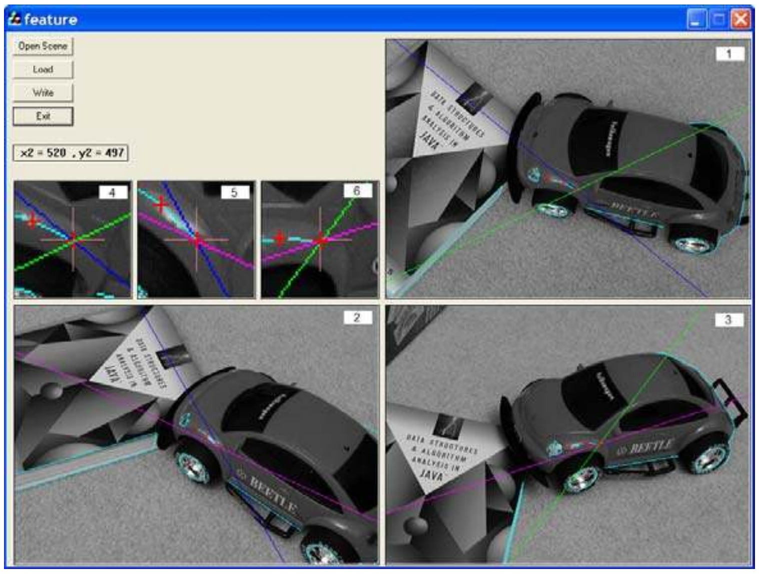


Figure 6.5: GUI to mark feature lines for car frames. (1), (2), and (3) are three reference image windows. (4), (5), and (6) are zoomed windows for current corresponding feature points. The cyan lines are feature lines. The current feature points are located at the intersection of two epipolar lines with red '+'. The magenta, blue and green lines are epipolar lines. After clicking in the first window, the UI will give two epipolar lines (blue and green) in this window, and one corresponding epipolar line (blue) in window (2), and one (green) in window (3). Then, the magenta lines will show the two corresponding points (the intersections with blue and green lines) in window (2) and (3). The user can move the mouse to drag the magenta lines and easily find these two corresponding feature points simultaneously with the second click.

The dissimilarity function $d(x_{12}, x_{21})$ is computed by using three SADs over 3×3 window N.

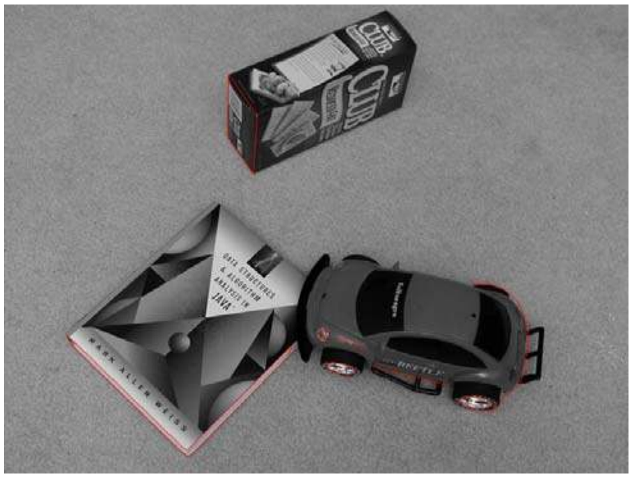
$$d(x_{12}, x_{21}) = \sum_N |\hat{I}_{12}(x_{12}) - \hat{I}_{21}(x_{21})| + \sum_N |\hat{I}_{23}(x_{23}) - \hat{I}_{32}(x_{32})| + \sum_N |\hat{I}_{31}(x_{31}) - \hat{I}_{13}(x_{13})|$$

Next, we use a pixel-to-pixel dynamic-scanline algorithm which uses an inter-scanline penalty to compute a dense disparity map for each corresponding rectified pair.

This stereo algorithm is reliable for the images with small occlusions and diffuse reflections. On the other hand, the stereo algorithm is not robust enough to handle some areas with large occlusion or specular reflection. Therefore, we have developed a user interface (GUI) to get some additional features to improve the disparity maps. This tool is based on trifocal geometry, which guarantees that the three corresponding feature lines match simultaneously. Once a user clicks a point in the first window, two epipolar lines will appear in the second and third window to guide the user to get the correct corresponding points. Figure 6.5 shows how to find three corresponding feature points by only two clicks.

After marking the feature lines, there may be several corresponding feature pixels at scanlines L and R with certain order, such as $(L_s, L_1, \cdots, L_i, \cdots, L_e)$ and $(R_s, R_1, \cdots, R_i, \cdots, R_e)$. For each corresponding marked interval, L_iL_{i+1} and R_iR_{i+1}, we apply the previous trinocular-stereo algorithm to compute the dense disparity inside this interval. Next, we link these intervals together and get the complete disparity map for these scanlines.

Using this tool, the geometrically correct features can be obtained easily. As a result, the dense disparities in these areas with specular reflections or occlusions can be correctly computed and blended. Figure 6.6.(b-c) shows a comparison of the results with and without the GUI. In car images, 8 feature lines are marked on the area with occlusions or specular reflection, which required about less than ten minutes of manual work by employing the GUI.


(*a*)

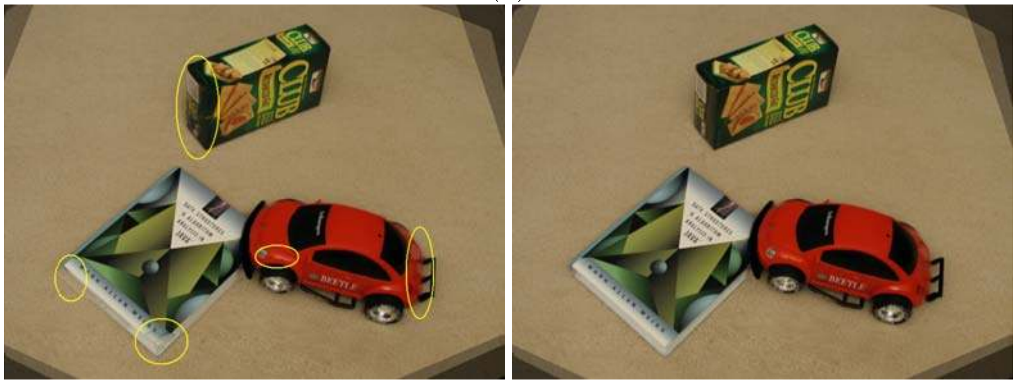

(*b*) (*c*)

Figure 6.6: Comparison of the morphing results using two different disparity maps. (b) is the result obtained without using feature lines. Note: there are some severe artifacts at the left most edge of the box, headlight of the car, book boundaries (yellow circles), etc. (c) is the result obtained with feature lines (red lines in (a)).

6.4 VIEW BLENDING FUNCTION

The view blending function determines the contribution of each reference image to the morphed images. In traditional image morphing, the blending function works on the original images such as I_1 and I_2. Usually, it cannot maintain geometric properties as strictly as view morphing, which interpolates the rectified images, $\hat{I}_1$ and $\hat{I}_2$, according to perspective geometry principles. Following this direction, we first show that any linear combination of three parallel views satisfies the perspective geometry property.

Suppose that the trifocal plane is located at $Z = 0$, and the camera center $C_i = [C_{ix}\ C_{iy}\ 0]^T$. The projection matrices for the parallel view C_i can be represented as:

$$\Pi_i = \begin{bmatrix} f_i & 0 & 0 & -f_i C_{ix} \\ 0 & f_i & 0 & -f_i C_{iy} \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

where f_i is focal length of C_i. The new point $\hat{p}_s$ can be obtained by linearly blending the three corresponding points $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$, which are respectively the projections of a 3D point $P = [X\ Y\ Z\ 1]^T$ on images $\hat{I}_1$, $\hat{I}_2$, and $\hat{I}_3$.

$$\begin{aligned} \hat{p}_s = \lambda_1\hat{p}_1 + \lambda_2\hat{p}_2 + \lambda_3\hat{p}_3 &= \frac{1}{Z}(\lambda_1\Pi_1 + \lambda_2\Pi_2 + \lambda_3\Pi_3)P \\ &= \frac{1}{Z}\Pi_s P, \end{aligned}$$

where Π_s is the linear interpolation of Π_1, Π_2, and Π_3, its focal length $f_s = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3$, and the camera center $C_s = \lambda_1 C_1 + \lambda_2 C_2 + \lambda_3 C_3$. Therefore, any linear combination of three parallel views satisfies the perspective geometry property when $\sum_{i=1}^{3} \lambda_i = 1$.

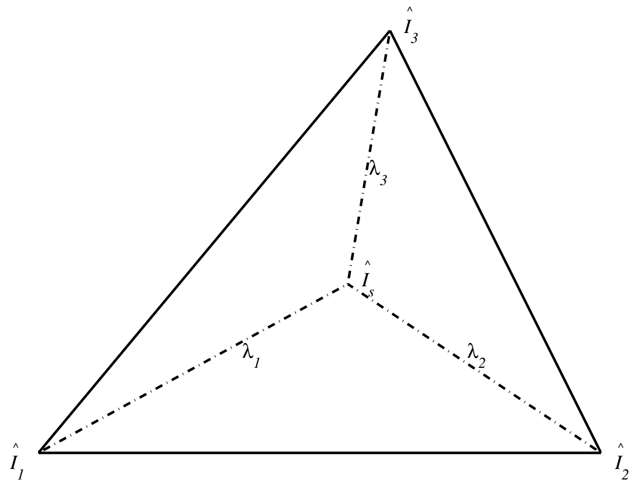


Figure 6.7: Blending by barycentric coefficients λ_1, λ_2, and λ_3. $\hat{I}_1$, $\hat{I}_2$ and $\hat{I}_3$ are rectified images, $\hat{I}_s$ is the desired image morphed using these three images.

Let W_{ij} be the warping function between images $\hat{I}_i$ and $\hat{I}_j$, which specifies the correspondences between $\hat{I}_i$ and $\hat{I}_j$.

$$W_{ij} = \hat{p}_i + d_{ij},$$

where d_{ij} can be obtained using the dense disparity map in the previous section when $i \neq j$, and $d_{ij} = 0$ when $i = j$. Next, we create a new warping function B_i to warp each of the images $\hat{I}_i$ to $\hat{I}_s$ and blend them together.

$$B_i = \Sigma_{i=1}^{3} \lambda_i W_{ij}, \tag{6.1}$$

$$\hat{I}_s = \Sigma_{i=1}^{3} \lambda_i B_i(\hat{I}_i), \tag{6.2}$$

where $\lambda_i B_i(\hat{I}_i)$ represents the warped image $\hat{I}_i$ into $\hat{I}_s$ with opacity value λ_i. It is easily verified that Eq. 6.1 and 6.2 have the linear blending property $\hat{p}_s = \sum_{i=1}^{3} \lambda_i \hat{p}_i$.

Based on this result, we present a scheme to blend three rectified images, which is based on image $\hat{I}_s$'s barycentric coordinates $\lambda = (\lambda_1, \lambda_2, \lambda_3)$, subject to $\lambda_i \geq 0$ and $\sum_{i=1}^{3} \lambda_i = 1$ (Figure 6.7). Along each edge of the triangle, one of the three coordinates (corresponding to the opposite vertex) is zero. This property is useful for continuous interpolation across the edges of a triangulation in multi-view morphing, such as Figure 6.11. After blending the rectified image pairs, we reproject the morphing images to the final position by 5-point postwarping algorithm (Appendix B) and obtain the final novel views as shown in Figure 6.1.

6.5 AUGMENTING 3D OBJECTS

In order to augment 3D objects in the virtual morphing environment, first we self-calibrate three cameras, and obtain the intrinsic and external parameters of these cameras. Then, for any novel view, the new camera matrix can be interpolated using blending coefficient λ. Based on this camera model, the 3D objects can be correctly rendered and augmented into the scene during the scene navigation, which is a prerequisite for the interaction with the synthesized environment.

Since we have three images, it become possible to perform self-calibration by assuming that aspect ratio = 1, skew ratio = 0, and principal points are located at the image center. This assumption is valid for most digital cameras. Therefore, we only try to recover the unknown focal length using three images. In our experiments, we also ignore the lens distortion which can be compensated by a radial distortion model[1]. In our experiment, we assume that the camera is distortion-free in the field of view. If the camera has a large lens distortion, we have to remove it before we perform the self-calibration or view morphing.

From the recovered trifocal tensor T, the projective camera matrices Π_{p_1}, Π_{p_2}, and Π_{p_3}, can be extracted as follows:

$$\begin{aligned}
\Pi_{p_1} &= [\mathrm{I} \mid 0], \\
\Pi_{p_2} &= [[\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3]e_{31} \mid e_{21}], \\
\Pi_{p_3} &= [(e_{31}e_{31}^T - \mathrm{I})[\mathrm{T}_1^T, \mathrm{T}_2^T, \mathrm{T}_3^T]e_{21} \mid e_{31}].
\end{aligned}$$

Since we used one digital camera to capture these images, we can safely assume that only the focal length in the camera's intrinsic parameters was changed during taking the pictures. Then, we employ the self-calibration method [41] to recover a 3D homography H and obtain a metric 3D construction of the sparse corresponding points, where the metric camera matrix $\Pi_{m_i} = \Pi_{p_i}H$. The metric camera matrix Π_{m_i} also can be represented as:

$$\begin{aligned}
\Pi_{m_i} &= K_i[R_i \mid R_i t_i] \\
&= K_i[R_z(\phi_i)R_y(\theta_i)R_x(\psi_i) \mid R_i t_i],
\end{aligned}$$

where R_i and t_i are the rotation and translation components of the camera's external parameters. Each rotation matrix R_i can be decomposed into three components $R_z(\phi_i)$, $R_y(\theta_i)$,

[1]In order to remove lens distortion, a precise calibration pattern is usually required.

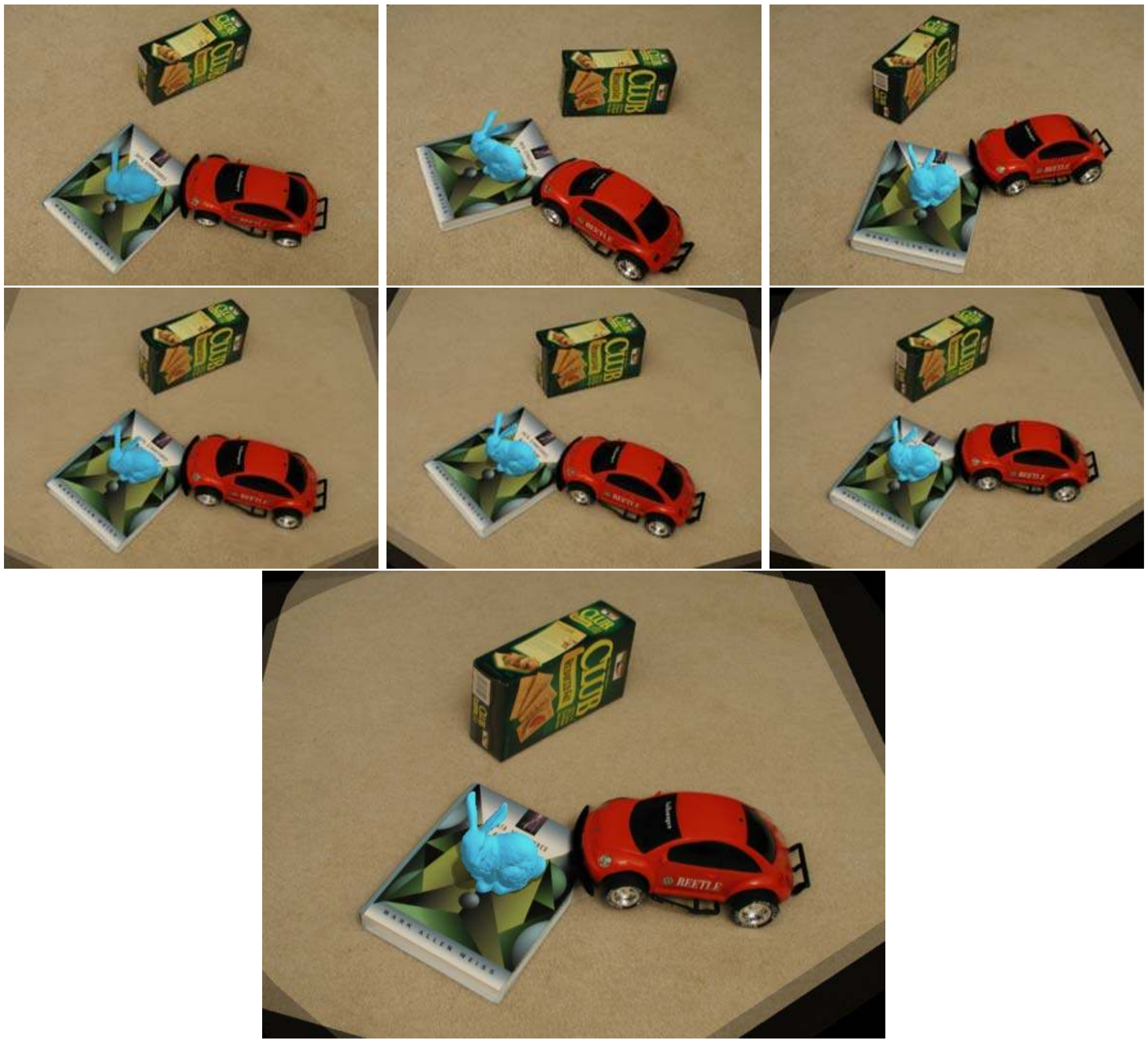


Figure 6.8: Augmenting 3D objects in tri-view morphing. Top: three original images with the augmented object (“Bunny” model from Stanford University). The 3D model is accurately augmented on the top of the book. Middle: a series of synthesized virtual views with the augmented object can maintain geometric correctness in the morphing procedure. We also generate the shadow of model and cast it on the book by assuming a light source. The bottom image is an enlarged result.

and $R_x(\psi_i)$. K_i is intrinsic parameter of camera, Π_{m_i}, such that

$$\Pi_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where f_i is the focal length of camera i.

Therefore, the parameters of the new virtual camera can be easily linearly interpolated by multiplying the coefficient $\lambda = (\lambda_1, \lambda_2, \lambda_3)$ respectively.

$$\begin{aligned} K_s &= \lambda_1 K_1 + \lambda_2 K_2 + \lambda_3 K_3, \\ t_s &= \lambda_1 t_1 + \lambda_2 t_2 + \lambda_3 t_3, \\ \phi_s &= \lambda_1 \phi_1 + \lambda_2 \phi_2 + \lambda_3 \phi_3, \\ \theta_s &= \lambda_1 \theta_1 + \lambda_2 \theta_2 + \lambda_3 \theta_3, \\ \psi_s &= \lambda_1 \psi_1 + \lambda_2 \psi_2 + \lambda_3 \psi_3, \end{aligned}$$

The new camera matrix is given by

$$\Pi_{m_s} = K_s[R_z(\phi_s)R_y(\theta_s)R_x(\psi_s) \mid R_s t_s].$$

This new camera model is guaranteed to move on the trifocal plane due to the linear interpolation of the translation components of the cameras. Based on this new camera matrix Π_{m_s}, we use OpenGL to render the 3D object and augment the object into the view generated from the same viewpoint. In Figure 6.8, we augment a 3D “Bunny” model on the top of the book. In order to generate the shadow, we simply recover a plane of book using sparse 3D points reconstructed from self-calibration method. Then, after assuming an approximate light source, the shadow is generated and projected on this plane and “Bunny” model by multiple pass rendering.

6.6 APPLICATIONS AND RESULTS

The tri-view morphing framework can be extended into different applications such as 4D video synthesis [110], multi-view morphing, dynamic tri-view morphing [106], automatic target recognition [108]. In this section, we demonstrate the first three applications and illustrate examples for each one. In our experiments, all of the static reference images were captured by a hand held Olympus digital camera.

6.6.1 4D VIDEO SYNTHESIS

Our tri-view morphing algorithm can be used for either static scenes or dynamic scenes with rigid objects. If the scene contains a non-rigid object with a large deformation, it is difficult to segment the object into several rigid parts as discussed in [105]. In order to navigate through a real dynamic scene, we can set up several static uncalibrated video cameras to capture the scene over time. For each time, T_i, using the tri-view morphing algorithm, we can morph multiple frames and navigate through the virtual scene. Furthermore, we can

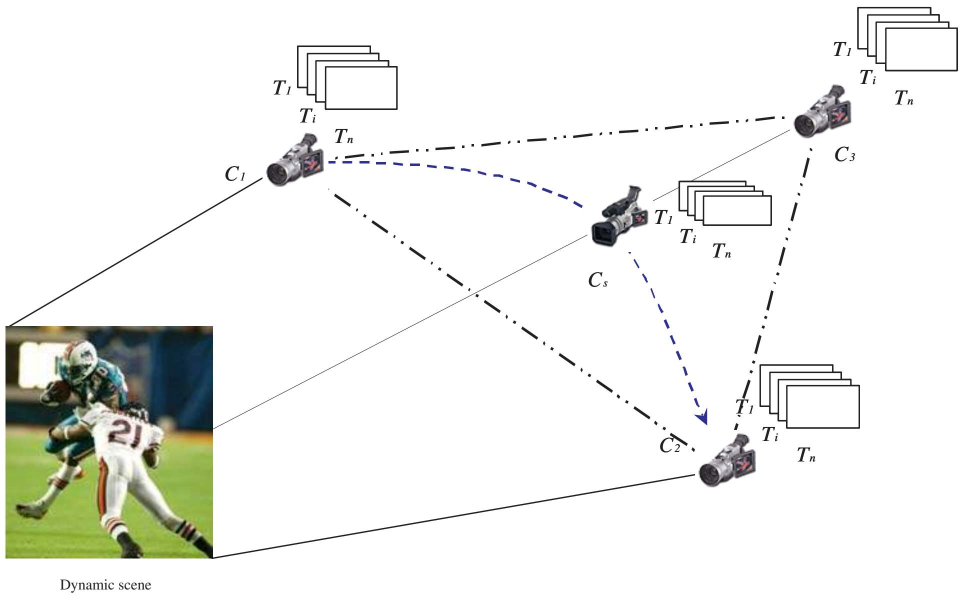


Figure 6.9: 4D video synthesis. A dynamic scene is captured by three video cameras. For each time T_i, tri-view morphing can be used to generate a virtual video camera C_s to navigate through the scene.

generate a moving camera to navigate through a 4D space (3D Euclidean space + 1D time space), which is called 4D video synthesis as shown in Figure 6.9.

Vedula, et. al. [96] proposed view interpolation over spatio-temporal domain, which is similar to our application. In their application, 14-17 fully calibrated cameras (with small baseline) were used on the one side of the actor/actress to capture the events. They used voxel coloring, 3D scene flow, and ray-casting algorithm to synthesize the novel view over these original image sequences. In order to simplify the rendering procedure, they removed the background layer and only rendered the actor/actress layers.

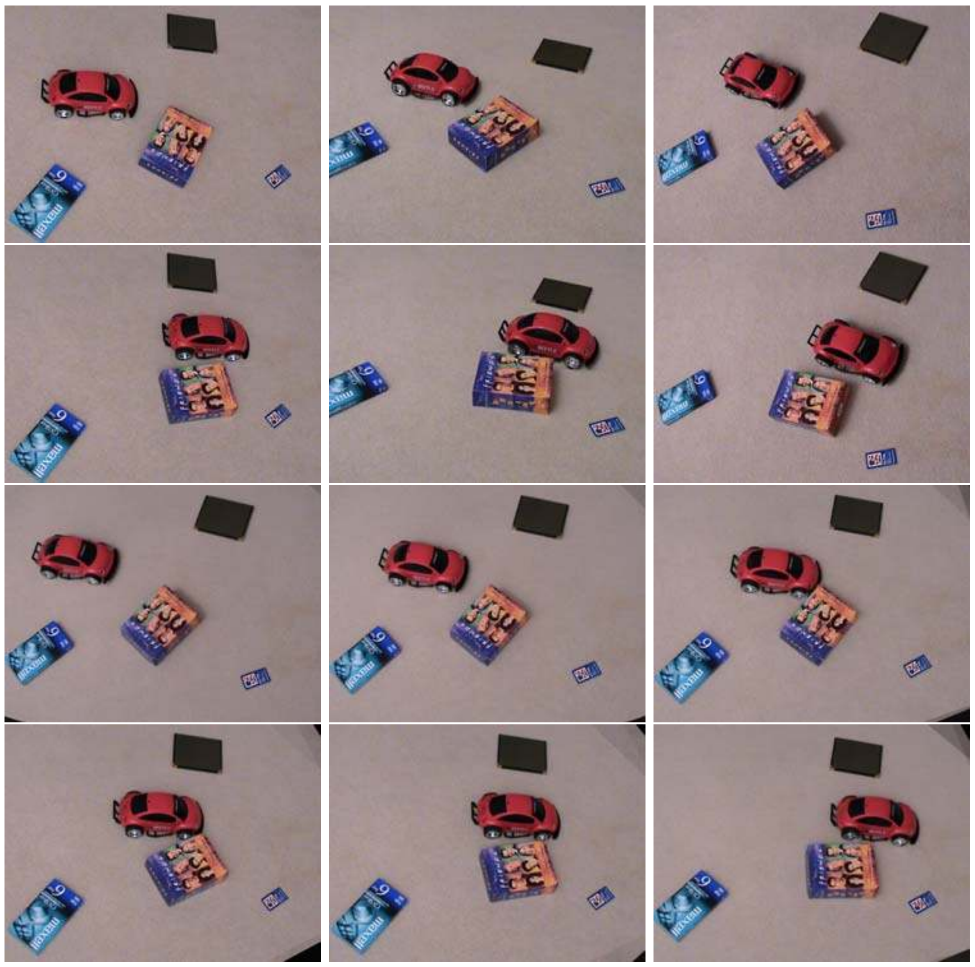


Figure 6.10: 4D video synthesis for car collision. The first row is the images captured at time T_1 by three video cameras. The second row is the images captured at T_2 by these video cameras. All of the original cameras were fixed during the video capture. The last two rows show images synthesized by a virtual moving video camera to navigate through the dynamic event of a car collision.

In our 4D video synthesis, we only use three uncalibrated wide baseline video cameras to acquire original image sequences, and blend the foreground (moving objects) and background simultaneously during the view interpolation. Our approach can be implemented in three steps. First, the video frames amongst the three cameras are synchronized. For each time, T_i, we get three images taken at the same time. Second, applying the tri-view morphing algorithm, we compute dense disparity between each pair of images and generate an arbitrary novel view in this trifocal plane. Third, if the trajectory for the motion of the virtual camera is specified, we can even link the frames over time T_i and generate a moving video camera instead of the original static video cameras.

Figure 6.10 shows a 4D video synthesis for a video depicting collision of a car with a box. We placed three video cameras at three different locations and in different orientations to record a car collision event. The car was controlled by a remote controller to impact the box. For each video camera, we obtained 26 frames. The first frame of each camera was used to determine the trifocal plane using the algorithm proposed in this thesis. Next, for each time T_i, disparity maps were computed between rectified images using our trinocular stereo algorithm. After an arbitrary trajectory for a virtual moving video camera is specified, a novel dynamic video is synthesized. A few images of video for one specified trajectory are shown in the second row of Figure 6.10.

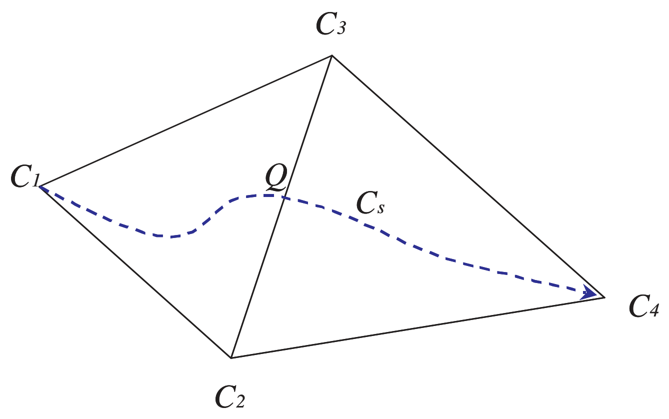


Figure 6.11: Four view morphing. A virtual camera C_s can seamlessly navigate the scene covered by four cameras over two triangles.

6.6.2 MULTI-VIEW MORPHING USING TRIANGLE TESSELLATION

If a scene is captured by more than three cameras (the cameras do not necessarily have to lie in the same plane), we can use triangle tessellation to group each triple of neighboring cameras and generate a new view to seamlessly navigate through the scene. Figure 6.11 shows a triangle tessellation for four view morphing. The four cameras are grouped into two triples (C_1, C_2, C_3) and (C_4, C_3, C_2). In Section 6.4, we introduced barycentric blending schemes to blend a new view, which can make a seamless connection over the triangle boundaries.

Figure 6.12 shows the morphing results using four images of a skateboarding doll. These four images were acquired by a hand held digital camera without any camera calibration. The surface of the doll is quite complicated. It is difficult to model it using a 3D textured model

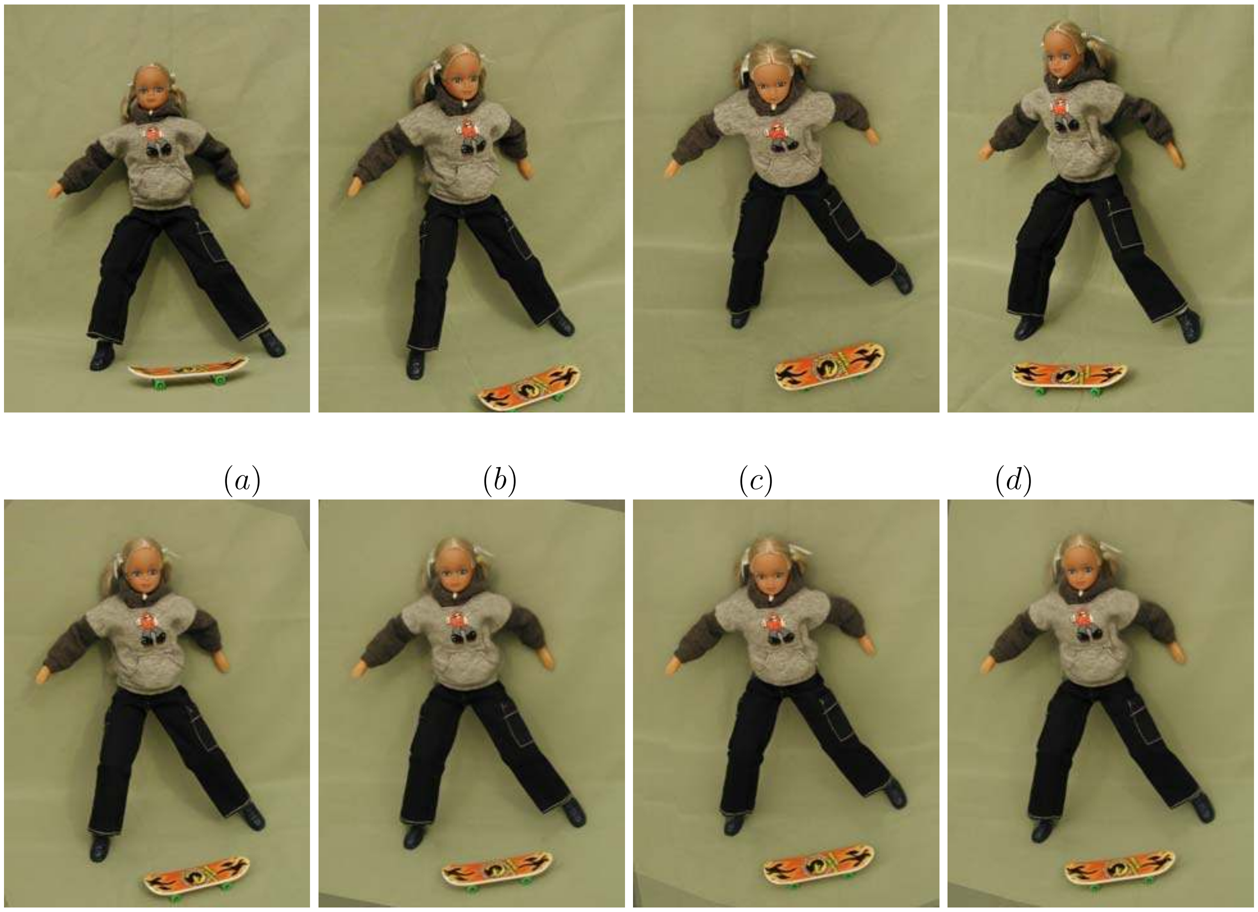


Figure 6.12: Four view morphing of a doll. (a-d) are four original uncalibrated images taken by a hand held camera. The second row shows a series of synthesized virtual views.

employing only these four images. We tessellated the four views into two triangles as shown in Figure 6.11, and generated novel views using the view independent blending scheme. In this case, a novel camera generates a continuous trajectory, which passes through the triangle boundaries, resulting in a seamless navigation.

Figure 6.13 shows the morphing results using four images of a Mickey Mouse (copyright of Disney). We put Mickey Mouse, a remote control and a star in the scene to increase the

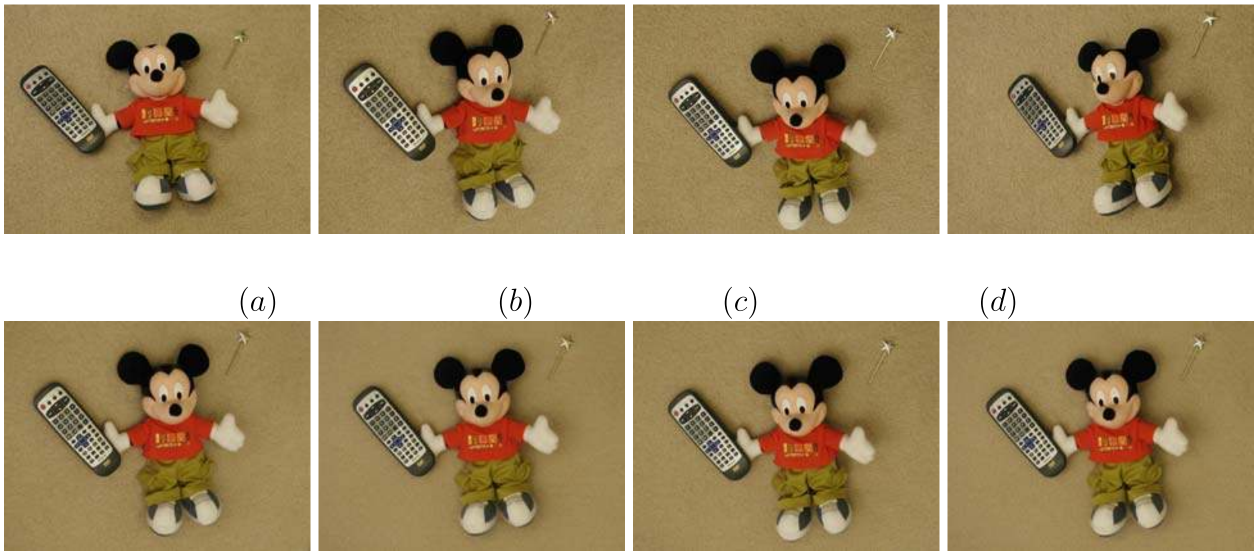


Figure 6.13: Four view morphing for Mickey Mouse. (a-d) are four original uncalibrated images taken by a hand held camera. The second row shows a series of synthesized virtual views.

variety of material properties (metal, plastic, and cloth, etc.). Since Mickey Mouse has a big nose, it occluded large area of the face (eye or mouth) in different views, and there are some small artifacts around the nose on the rendering images.

6.6.3 DYNAMIC TRI-VIEW MORPHING

Our dynamic tri-view morphing is focused on a dynamic scene containing only rigid moving objects, which may rotate or translate. Xiao et. al. [105] have shown that novel views can be generated for such dynamic scenes by using a separate fundamental matrix for each

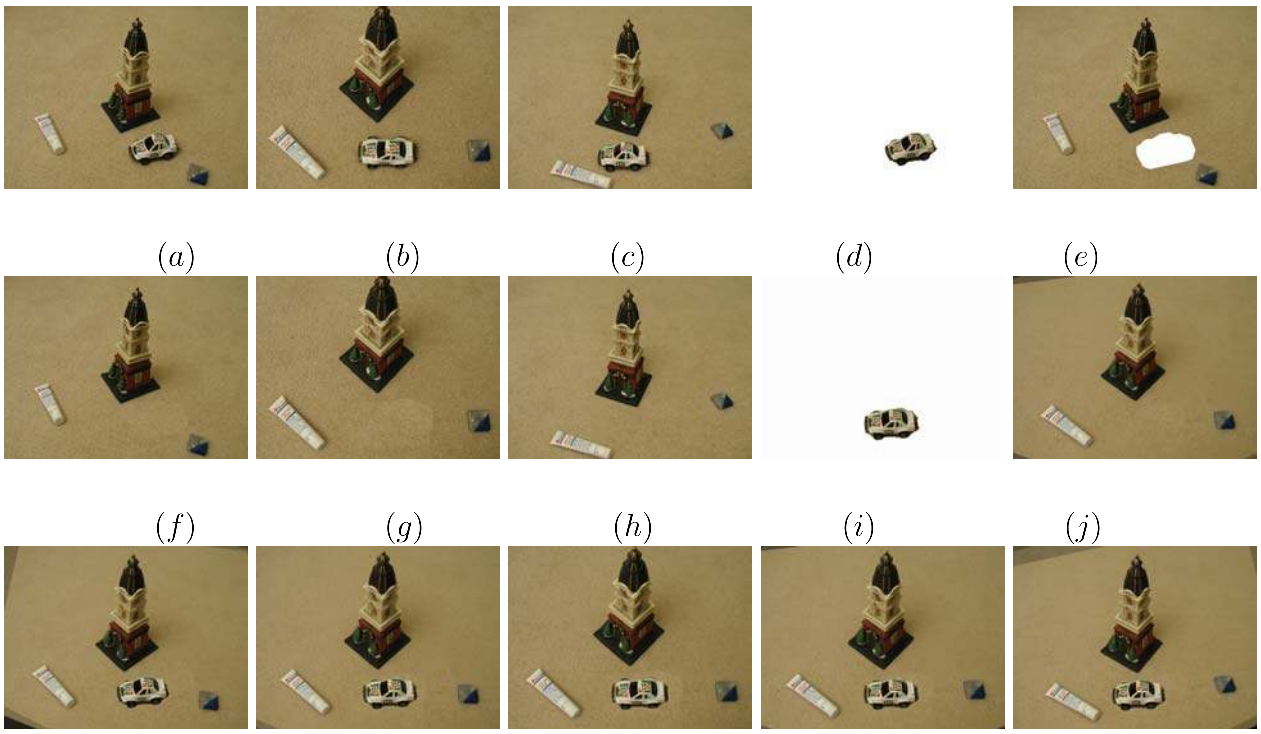


Figure 6.14: A typical dynamic tri-view morphing scenario. (a), (b) and (c) are three uncalibrated wide-baseline reference images. (d) and (e) are segmentation results of the first frame. (f-h) are the background images corresponding to the three original views after filling in the gaps occupied by the car. (i) and (j) are morphing results of the car and background layers respectively. The last row shows the virtual views obtained after compositing the car and background layers.

rigid object (layer) based on the relative motion between two views. In this thesis, we extend this idea to dynamic tri-view morphing. Our algorithm consists of three main steps: segmentation, tri-view morphing for each layer and multiple layer blending.

Figure 6.14 shows a dynamic scene with a moving car. Three reference images were taken at different times and at different positions and orientations while the car was moving. We put an eye-drops tube in the scene to introduce a transparent material. Since the moving object (car) and the background (the rest of the image) have different epipolar geometries, we segment the scene into two layers: background (Figure 6.14.e) and car (Figure 6.14.d). Next, we compute a homography to fill in the hole in Figure 6.14.e using the pixels from the other two images, in order to obtain the background image (Figure 6.14.f) for the first frame. Then, the background (Figure 6.14.i) and car (Figure 6.14.j) are blended separately by the tri-view morphing algorithm. Finally, these two images are composited together and the final results are obtained as shown in the last row of Figure 6.14. As a result, the car is moving along its own trajectory during the scene navigation.

Note: The true strength of the proposed approach can only be realized by looking at a video sequence. The high resolution video sequences of the above experiments are available on our web site [37].

6.7 SUMMARY

Using our proposed image-based approach, we can interactively navigate through a scene based on only three wide-baseline uncalibrated images without the explicit use of a 3D model. Based on the decomposition of the affine matrix, a large number of corresponding points among the original images are automatically recovered to determine an accurate trifocal geometric relationship between three reference images. The disparity maps are computed to construct the blending functions using our feature-based trinocular-stereo algorithm. An arbitrary novel view located in the trifocal plane is blended to obtain a photo-realistic image, which can be correctly augmented by 3D objects after camera self-calibration.

We demonstrated three applications of tri-view morphing: 4D video synthesis, multi-view morphing, and dynamic view morphing. All of the applications are useful for filling in gaps in multiple images, movie-making, video compression, etc. Our results successfully show that even using few images (3-4), we can generate visually realistic images to navigate the virtual environment, where the virtual objects can interact with the 3D scene.

One limitation of our work is that our navigation currently is restricted in a 2D space constructed by three images. Even though an arbitrary novel view at any position can be synthesized from two or three images as claimed by Avidan [83], it usually cannot preserve the good-quality rendering results due to texture missing from the new viewpoint. Hence, the novel view along the baseline of two images or along the tri-focal plane of three images can avoid this problem and can provide the higher quality results. In order to navigate a 3D volume and obtain a realistic visual effect, four images captured by the non-coplanar cameras are minimal requirements. In the future, we will investigate this area and extend our method to 3D volume navigation. On the other hand, if the strong specular reflections are present in the images, it is hard to determine disparities only by the stereo algorithm without feature marks. How to remove the specular reflection and obtain stable disparity is another direction with good research potential.

CHAPTER 7

CONCLUSION

In this dissertation, we successfully showed that using the sparse images, we can efficiently synthesize a virtual scene by using an image-based approach without the use of 3D model.

We developed two powerful tools for the preliminary steps: first is the two-stage, wide baseline matching algorithm that can recover feature corresponding points and obtain epipolar geometry implied in the images; second is the layer-based video segmentation algorithm which makes it possible to accurately extract layers or objects from a video sequence. After the preliminary steps, we can estimate density correspondences by using stereo algorithms under projective geometry. Finally, based on the estimated disparity maps or depth, a final synthetic view is correctly blended by a linear interpolation scheme.

This work addresses the following problems in different areas: wide baseline matching, layer-based video segmentation, dynamic view synthesis with non-rigid objects, tri-view and multi-view synthesis, 4D video synthesis. However, all of these problems are closely related and demonstrate the important fact that image based rendering technique will be useful and attractive if we successfully resolve the related vision problems.

APPENDIX A

PREWARPING

The purpose of the prewarping procedure is to warp the three reference images into the trifocal plane and to obtain rectified images, $\hat{I}_1$, $\hat{I}_2$, and $\hat{I}_3$, which are used to compute dense disparity maps. This procedure is implemented in three steps.

Note that $N_{E1} = e_{12} \times e_{13}$ in camera C_1, $N_{E2} = e_{23} \times e_{21}$ in camera C_2, and $N_{E3} = e_{31} \times e_{32}$ in camera C_3.

First, we rotate the original images to a plane parallel to the trifocal plane. Let R_1, R_2, and R_3 be the rotation matrices for images I_1, I_2, and I_3 respectively.

$$R_1^{-1} = \begin{bmatrix} r_{11}^T \\ r_{12}^T \\ r_{13}^T \end{bmatrix}, \; R_2^{-1} = \begin{bmatrix} r_{21}^T \\ r_{22}^T \\ r_{23}^T \end{bmatrix}, \; R_3^{-1} = \begin{bmatrix} r_{31}^T \\ r_{32}^T \\ r_{33}^T \end{bmatrix},$$

where $r_{11} = \frac{e_{12}}{|e_{12}|}$, $r_{13} = \frac{N_{E1}}{|N_{E1}|}$, $r_{12} = r_{11} \times r_{13}$, $r_{21} = \frac{-e_{21}}{|e_{21}|}$, $r_{23} = \frac{-N_{E2}}{|N_{E2}|}$, $r_{22} = r_{21} \times r_{23}$, $r_{31} = \frac{e_{31}}{|e_{31}|}$, $r_{33} = \frac{-N_{E3}}{|N_{E3}|}$, $r_{32} = r_{31} \times r_{33}$. After this step, all of the epipoles are projected onto infinity, and the epipolar lines passing through e_{12} and e_{21} are rotated so they are parallel to the horizontal. Now the new F_{12} can be represented as:

$$\tilde{F}_{12} = R_2 F_{12} R_1^{-1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & a \\ 0 & b & c \end{bmatrix}.$$

The image $\hat{I}_2$ is adjusted by vertical scaling and translation by a matrix T [75, 73]. The prewarping homographies for images I_1, I_2 are $H_1 = R_1$, $H_2 = TR_2^{-1}$ respectively. Next, we rotate $\hat{I}_3$ by an angle ϕ and adjust the image by a scaling and shearing matrix Q, which guarantees that the sum of the three internal angles between projected epipoles equals $180°$. The images $\hat{I}_1$ and $\hat{I}_2$ are equivalent to the corresponding rectified pair $\hat{I}_{12}$ and $\hat{I}_{21}$.

Similarly, we can rotate images to make different epipolar lines (passing e_{ij} and e_{ji}) horizontal and get the other two corresponding rectified pairs, $\hat{I}_{23}$ and $\hat{I}_{32}$, $\hat{I}_{31}$ and $\hat{I}_{13}$.

APPENDIX B

POSTWARPING

The postwarping procedure deals with the reprojection of the morphing results into a proper final position. In the original view morphing algorithm, this step was implemented by manually determining the final position of four control points or performing linear interpolation (which usually causes shrinking [73, 105]).

If the cameras are self-calibrated using the computed correspondences and the metric sparse 3D points are recovered (the case for augmenting 3D objects), the position of these points in the final view can be computed by transforming 3D points using the new interpolated camera matrix. Then, the reprojection can be implemented by simply finding a homography between the morphed image and the final view.

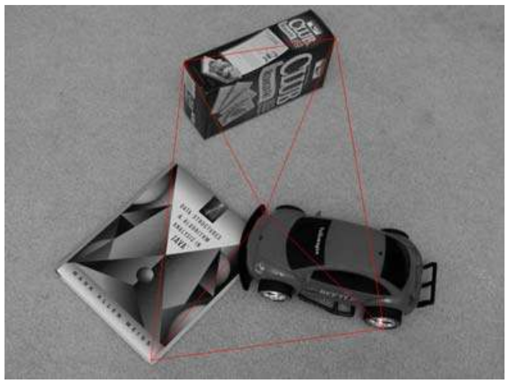
For the uncalibrated camera case, we can use the least distortion method [110, 105, 1] to approximate the final postwarping shape position to obtain smooth views transformation. First, we automatically select 5 corresponding points such that one is close to the centroid, and the other four points are located in four different directions near the image boundary in original images I_1, I_2 and I_3 as shown in Figure B.1. Then, based on the assumption that the centroid of the scene p_{cs} moves following the linear combination of p_{c1}, p_{c2}, and p_{c3},

$$p_{cs} = \sum_{i=1}^{3} \lambda_i p_{ci},$$

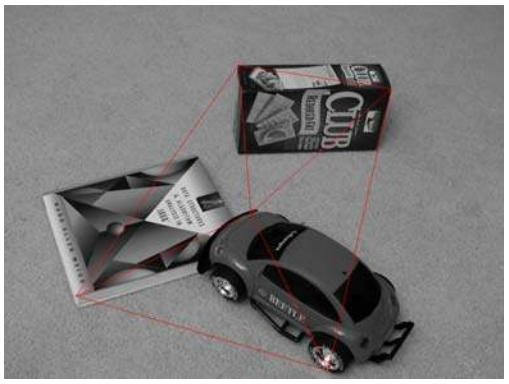
where p_{ci} is the center point position in image I_i (Figure B.1). In the 5-point scheme, the triangulation T includes 4 triangles t_i ($i = 1 \cdots 4$). The intermediate postwarping position (blue lines in Figure B.1.d) can be obtained by minimizing Eq. B.1 by the least squares method.

$$\varepsilon_V = \sum_{t_i \in T} \| A_{\{t_i\}} - B_{\{t_i\}} \|^2, \tag{B.1}$$

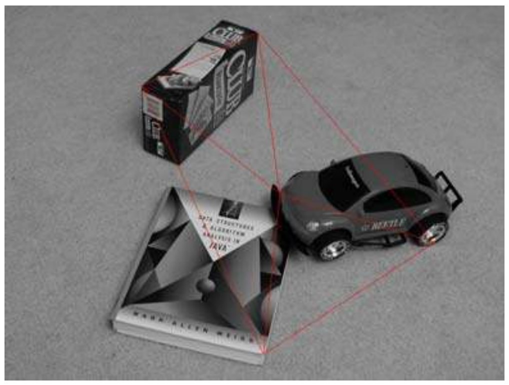
where $A_{\{t_i\}}$ is an ideal shape of triangle t_i obtained by the least distortion method over three images, $B_{\{t_i\}}$ is an actual shape of the triangle t_i, and V is a set of the five points.



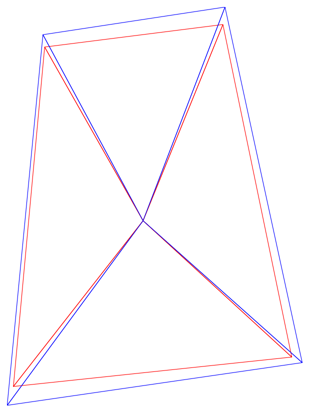
(*a*)



(*b*)



(*c*)



(*d*)

Figure B.1: 5-point postwarping. 5 corresponding points are selected to construct four triangles in reference images (a-c). (d) compares an intermediate shape generated by the least distortion method (blue) with linear method (red).

List of References

[1] M. Alexa, D. Cohen-Or, and D. Levin. "As-Rigid-As-Possible Shape Interpolation." *Proceedings of ACM SIGGRAPH*, pp. 157–164, 2000.

[2] G. Adiv. "Determining three-dimensional motion and structure from optical flow generated by several moving objects." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1985.

[3] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[4] S. Ayer and H. Sawhney. "Layered Representation of Motion Video using Robust Maximum-Likelihood Estimation of Mixture Models and MDL Encoding." *International Conference on Computer Vision*, 1995.

[5] A. Baumberg. "Reliable feature matching across widely separated views." *Proceedings of Computer Vision and Pattern Recognition*, 2000.

[6] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. "Unstructured Lumigraph Rendering." *Proceedings of ACM SIGGRAPH*, pp. 425–432, 2001.

[7] Y. Boykov and M. Jolly. "Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images." *International Conference on Computer Vision*, 2001.

[8] Y. Boykov and V. Kolmogorov. "Computing Geodesics and Minimal Surfaces via Graph Cuts." *International Conference on Computer Vision*, 2003.

[9] M. Brown and D. Lowe. "Invariant features from interest point groups." *British Machine Vision Conference*, pp. 656–665, 2002.

[10] L. Bergen and F. Meyer. "A Novel Approach to Depth Ordering in Monocular Image Sequences." *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[11] T. Beier and S. Neely. "Feature-Based Image Matamorphosis." *Proceedings of ACM SIGGRAPH*, pp. 35–42, 1992.

[12] K. Boyer and S. Sarkar. *Perceptual Organization for Artificial Vision Systems.* Kluwer Academic Publishers, 2000.

[13] S. Birchfield and C. Tomasi. "Depth Discontinuities by Pixel-to-Pixel Stereo." *International Journal of Computer Vision*, pp. 269–293, 1999.

[14] Y. Boykov, O. Veksler, and R. Zabih. "Markov Random Fields with efficient approximations." *IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[15] Y. Boykov, O. Veksler, and R. Zabih. "Fast Approximate Energy Minimization via Graph Cuts." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[16] W. Chen, J. Bouguet, M. Chu, and R. Grzeszczuk. "Light Field Mapping: Efficient Representation and Hardware Rendering of Surface Light Fields." *Proceedings of ACM SIGGRAP*, pp. 447–456, 2002.

[17] S. Cohen, G. Elber, and R Bar Yehuda. "Matching of Freeform curves." *CAD*, pp. 369–378, 1997.

[18] Y. Cheng. "Mean Shift, Mode Seeking, and Clustering." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[19] J. Costeira and T. Kanade. "A multibody factorization method for independently moving objects." *International Journal of Computer Vision*, 1998.

[20] D. Comaniciu and P.Meer. "Robust Analysis of Feature Spaces: Color Image Segmentation." *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[21] D. Comaniciu and P.Meer. "Mean Shift Analysis and Applications." *International Conference on Computer Vision*, 1999.

[22] D. Comaniciu and P.Meer. "Mean Shift: A Robust Approach toward Feature Space Analysis." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[23] S. Chen and L. William. "View Interpolation for Image Synthesis." *Proceedings of ACM SIGGRAPH*, pp. 279–288, 1993.

[24] E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour, and M. Yannakakis. "The Complexity of Multiway Cuts." *ACM Symp. Theory of Computing*, 1992.

[25] R. Deriche, Z. Zhang, Q. Luong, and O. Faugeras. "Robust recovery of the epipolar geometry for an uncalibrated stereo rig." *European Conference on Computer Vision*, 1994.

[26] O. Faugeras and Q. Luong. *The geometry of multiple images.* The MIT Press, 2001.

[27] X. Feng and P. Perona. "Scene segmentation from 3D motion." *IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[28] V. Ferrari, T. Tuytellars, and L. Van Gool. "Wide-Baseline Multiple-View Correspondences." *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[29] A. Fusiello, E. Trucco, T. Tommasini, and V. Roberto. "Improving features tracking with robust statistics." *Pattern Analysis & Applications*, pp. 312–320, 1999.

[30] J. Gallier. *Geometric Methods and Applications.* Springer Press, 2001.

[31] S. Gortler, R. Grzeszczuk, R. Szeliski, , and M. Cohen. "The Lumigraph." *Proceedings of ACM SIGGRAPH*, pp. 43–52, 1996.

[32] P. Giaccone and G. Jones. "Segmentation of Global Motion using Temporal Probabilistic Classification." *British Machine Vision Conference*, 1998.

[33] L. Gaucher and G. Medioni. "Accurate Motion Flow Estimation with Discontinuties." *IEEE International Conference on Computer Vision*, pp. 695–702, 1999.

[34] R. Hartley. "In defence of the 8-point algorithm." *IEEE International Conference on Computer Vision*, pp. 1064–1070, 1995.

[35] D. Hoffman and W. Richards. "Salience of Visual Parts." *Cognition*, pp. 29–78, 1997.

[36] http://www.cs.ucf.edu/∼vision/projects/motion_layer_extraction/.

[37] http://www.cs.ucf.edu/∼vision/projects/triview/.

[38] http://www.cs.ucf.edu/∼vision/projects/viewMorphing/.

[39] http://www.cs.ucf.edu/∼vision/projects/widematching/.

[40] R. Hartley and R. Vidal. "The Multibody Trifocal Tensor: Motion Segmentation from 3 Perspective Views." *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[41] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[42] H. Jin, P. Favaro, and S. Soatto. "Real-time teature tracking and outlier rejection with changes in illumination." *IEEE International Conference on Computer Vision*, 2001.

[43] H. Johan, Y. Koiso, and T. Nishita. "Morphing Using Curves and Shape Interpolation Techniques." *Proceedings of Pacific Graphics*, 2000.

[44] V. Kwatra, I. Essa, A. Schdl, G. Turk, and A. Bobick. "Graphcut Textures: Image and Video Synthesis Using Graph Cuts." *Proceedings of ACM SIGGRAPH*, 2003.

[45] Q. Ke and T. Kanade. "A Subspace Approach to Layer Extraction." *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[46] Q. Ke and T. Kanade. "A Robust Subspace Approach to Layer Extraction." *IEEE Workshop on Motion and Video Computing*, 2002.

[47] Q. Ke and T. Kanade. "Robust Subspace Clustering by Combined Use of kNND Metric and SVD Algorithm." *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[48] S. Khan and M. Shah. "Object Based Segmentation of Video Using Color, Motion and Spatial." *IEEE International Conference on Computer Vision and Pattern Recognition*, 2001.

[49] S. Kang, R. Szeliski, and J. Chai. "Handing Occlusions in Dense Multi-view Stereo." *IEEE Conferece on Computer Vision and Pattern Recognition*, 2001.

[50] V. Kolmogorov and R. Zabih. "Visual correspondece with occlusions using grpah cuts." *IEEE International Conference on Computer Vision*, 2001.

[51] V. Kolmogorov and R. Zabih. "Multi-camera Scene Reconstruction via Graph Cut." *European Conference on Computer Vision*, 2002.

[52] S. Lee, K. Chwa, J. Hahn, and S. Shin. "Image metamorphosis using snakes and free-form deformations." *Proceedings of ACM SIGGRAPH*, pp. 439–448, 1995.

[53] Q. Luong and O. Faugeras. "The Fundamental Matrix: Theory, Algorithms, and Stability Analysis." *International Journal of Computer Vision*, pp. 43–75, 1996.

[54] M. Levoy and P. Hanrahan. "Light Field Rendering." *Proceedings of ACM SIGGRAPH*, pp. 31–42, 1996.

[55] A. Leonardis, F. Solina, and R. Bajcsy. *Confluence of Computer Vision and Computer Graphics*. Kluwer Academic Publishers, 2000.

[56] S. Lee, G. Wolberg, and S. Shin. "Polymorph: Morphing Among Mutltiple Images." *IEEE Computer Graphics and Applications*, pp. 60–72, 1998.

[57] R. Manning and C. Dyer. "Interpolating View and Scene Motion by Dynamic View Morphing." *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 388–394, 1999.

[58] G. Medioni, M. Lee, and C. Tang. *A Computational Framework for Segmentation and Grouping.* Elsevier Science, 2000.

[59] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan. “mage-Based 3D Photography using Opacity Hulls.” *Proceedings of ACM SIGGRAPH*, pp. 427–437, 2002.

[60] K. Mikolajczyk and C. Schmid. “Indexing based on scale invariantinterest points.” *International Conference on Computer Vision*, 2001.

[61] K. Mikolajczyk and C. Schmid. “An affine invariant interest point detector.” *European Conference on Computer Vision*, 2002.

[62] G. Van Meerbergen, M. Vergauwen, M. Pollefeys, and L. Van Gool. “A Hierarchical Symmetric Stereo Algorithm Using Dynamic Programming.” *International Journal of Computer Vision*, 2002.

[63] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces.* The Springer-Verlag Press, 2003.

[64] M. Pollefeys and L. Van Gool. “Visual modeling: from images to images.” *Journal of Visualization and Computer Animation*, pp. 199–209, 2002.

[65] 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*, 2004.

[66] I. Patras, E. hendirks, and R. Lagendijk. “Video segmentation by MAP labeling of watershed segments.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[67] S. Pollard, M. Pilu, S. Hayes, and A. Lorusso. “View synthesis by edge matching and transfer.” *IEEE Workhop on Applications of Computer Vision*, 1998.

[68] P. Pritchett and A. Zisserman. “Wide baseline stereo matching.” *IEEE International Conference on Computer Vision*, pp. 754–760, 1998.

[69] K. Rangarajan, M. Shah, and D. Brackle. “Optimal corner detector.” *Journal of Computer Vision, Graphics and Image Processing*, pp. 230–245, 1989.

[70] S. Smith and M. Brady. “SUSAN-a new approach to low level image processing.” *International Journal on Computer Vision*, 1997.

[71] S. Seitz and C. Dyer. “Physically-valid view synthesis by image interpolation.” *IEEE Workshop on Representation of Visual Scenes*, 1995.

[72] S. Seitz and C. Dyer. "Toward image-based scene representation using view morphing." *International Conf. on Pattern Recognition*, 1996.

[73] S. Seitz and C. Dyer. "View Morphing." *Proceedings of ACM SIGGRAPH*, pp. 21–30, 1996.

[74] P. Smith, T. Drummond, and R. Cipolla. "Layered Motion Segmentation and Depth Ordering by Tracking Edges." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[75] S. Seitz. *Image-Based Transformation of viewpoint and scene appearance.* computer science, University of Wisconsin, 1997.

[76] J. Sethian. *Level set methods and fast marching methods.* Cambridge University Press, 1999.

[77] T. Sederberg and E. Greenwood. "A Physically Based Approach to 2-D Shape Blending." *Proceedings of ACM SIGGRAPH*, pp. 25–34, 1992.

[78] T. Sederberg, P. Gao, G. Wang, and H. Mu. "2-D Shape Blending: An Intrinsic Solution to the Vertex Path Problem." *Proceedings of ACM SIGGRAPH*, pp. 15–18, 1993.

[79] H. Shum and L. He. "Rendering with Concentric Mosaics." *Proceedings of ACM SIGGRAPH*, pp. 299–306, 1999.

[80] J. Shi and J. Malik. "Motion segmentation and tracking using normalized cuts." *International Conference on Computer Vision*, 1998.

[81] J. Shi and J. Malik. "Normalized Cuts and Image Segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 888–905, 2000.

[82] M. Shapira and A. Rappoport. "Shape Blending Using the Star-Skeleton Representation." *IEEE Computer Graphics and Application*, 1995.

[83] S.Avidan and A. Shashua. "Novel View Synthesis by Cascading Trilinear Tensors." *IEEE Transactions on Visualization and Computer Graphics*, pp. 293–306, 1998.

[84] D. Scharstein and R. Szeliski. "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms." *International Journal of Computer Vision*, 2002.

[85] J. Shi and C. Tomasi. "Good Features to Track." *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[86] F. Shen and H. Wang. "A local edge detector used for finding corners." *Proceedings of ICICS*, 2001.

[87] F. Schaffalitzky and A. Zisserman. “Viewpoint invariant texture matching and wide baseline stereo.” *International Conference on Computer Vision*, 2001.

[88] R. Szeliski. “Video Mosaics for Virtual Environments.” *IEEE Computer Graphics and Applications*, 1996.

[89] D. Tschumperle and R. Deriche. “Vector-Valued Image Regularization with PDE’s: A Common Framework for Different Applications.” *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[90] A. Tal and G. Elber. “Image Morphing with Feature Preserving Texture.” *Proceedings of EUROGRAPHICS*, pp. 339–348, 1999.

[91] T. Tuytelaars and L. Van Gool. “Wide baseline stereo matching based on local, affinely invariant regions.” *British Machine Vision Conference*, 2000.

[92] P. Torr and D. Murray. “Outlier detection and motion segmentation.” *SPIE Sensor Fusion Conference VI*, 1993.

[93] C. Tomasi and R. Manduchi. “Bilateral filtering for gray and color images.” *IEEE International Conference on Computer Vision*, 1998.

[94] H. Tao, H. Sawhney, and R. Kumar. “Object tracking with bayesian estimation of dynamic layer representations.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[95] E. Trucco and A. Verri. *Intorductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

[96] S. Vedula, S. Baker, and T. Kanade. “Spatio-Temporal View Interpolation.” *Eurographics Workshop on Rendering*, 2002.

[97] R. Vidal and Y. Ma. “A Unified Algebraic Approach to 2-D and 3-D Motion Segmentation.” *Eourpean Conference on Computer Vision*, 2004.

[98] R. Vidal, Y. Ma, S. Soatto, and S. Sastry. “Two-view multibody structure from motion.” *International Journal of Computer Vision*, 2004.

[99] J. Wang and E. Adelson. “Representing moving images with layers.” *IEEE Transaction on Image Processing*, pp. 625–638, 1994.

[100] J. Wills, S. Agarwal, and S. Belongie. “What Went Where.” *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[101] Y. Weiss. “Smoothness in Layers: Motion Segmentation using Nonparametric on Homographics.” *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[102] G. Wolberg. "Image Morphing: a Survey." *The visual computer*, 1998.

[103] Y. Wexler and A. Shashua. "On the synthesis of dynamic scenes from reference views." *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[104] N. Xu, R. Bansal, and N. Ahuja. "Object Segmentation Using Graph Cuts Based Active Contours." *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[105] J. Xiao, C. Rao, and M. Shah. "View Interpolation for Dynamic Scenes." *EUROGRAPHICS*, 2002.

[106] J. Xiao and M. Shah. "From Images to Video: View Morphing of Three Images." *Workshop of Vision, Modeling, and Visualization*, 2003.

[107] J. Xiao and M. Shah. "Two-Frame Wide Baseline Matching." *IEEE International Conference on Computer Vision*, 2003.

[108] J. Xiao and M. Shah. "Automatic Target Recognition Using Multi-View Morphing." *Conferences of SPIE Automatic Target Recognition XIV*, 2004.

[109] J. Xiao and M. Shah. "Motion Layer Extraction in the Presence of Occlusion using Graph Cut." *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[110] J. Xiao and M. Shah. "Tri-view Morphing." *Computer Vision and Image Understanding*, 2004.

[111] J. Xiao and M. Shah. "Layer Based Video Registration." *Journal of Machine Vision and Application*, 2005.

[112] J. Xiao, Y. Zhang, and M. Shah. "Adaptive Region-Based Video Registration." *IEEE Workshop on Motion*, 2005.

[113] Z. Zhang. "Determining the Epipolar Geometry and its Uncertainty: A Review." *International Journal of Computer Vision*, 1998.

[114] L. Zhao. *Dressed Human Modeling, Detection, and Parts Localization.* The Robotics Institue, Carnegie Mellon University, 2001.

[115] L. Zelnik-Manor and M. Irani. "Multi View Subspace Constraints on Homographies." *IEEE International Conference on Computer Vision*, 1999.

[116] Y. Zhou and H. Tao. "A Background Layer Model for Object Tracking through Occlusion." *International Conference on Computer Vision*, 2003.

[117] Z. Zhang, L. Wang, B. Guo, and H. Shum. “Feature-Based Light Field Morphing.” *Proceedings of ACM SIGGRAPH*, pp. 457–464, 2002.

[118] Y. Zhang, J. Xiao, and M. Shah. “Layer-Based Object Removal in Videos.” *Eurographics*, 2004.

[119] Y. Zhang, J. Xiao, and M. Shah. “Motion Layer Based Object Removal in Videos.” *IEEE Workshop on Application on Computer Vision*, 2004.