December 16, 2005

To the Graduate School:

This thesis entitled “Motion Segmentation over Image Sequences Using Multiway Cuts and Affine Transformations” and written by Bragadeeshwaran Natarajan is presented to the Graduate School of Clemson University. I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Electrical Engineering.

Dr. Stanley T. Birchfield, Advisor

We have reviewed this thesis
and recommend its acceptance:

Dr. Adam W. Hoover

Dr. John N. Gowdy

Accepted for the Graduate School:

MOTION SEGMENTATION OVER IMAGE SEQUENCES USING MULTIWAY CUTS AND AFFINE TRANSFORMATIONS

A Thesis

Presented to

the Graduate School of

Clemson University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Electrical Engineering

---

by

Bragadeeshwaran Natarajan

December 2005

Advisor: Dr. Stanley T. Birchfield

ABSTRACT

Motion segmentation is an important process in computer vision. Graph cut based tools exist for segmenting moving objects in images using frame-to-frame correspondence. Also graph cut tools have been shown to perform efficiently for stereo correspondence. In this thesis, multiway graph cut techniques and affine transformations for stereo and motion by Birchfield and Tomasi are analyzed. This work was primarily intended for stereo pair of images and two frame motion with slanted surfaces. In this thesis, two extensions to this technique are explored. One is using multiway cuts and affine motion model to perform motion segmentation over a sequence of images. This is achieved by performing frame-to-frame motion segmentation and predicting an estimate of motion segments in the next image. This estimate is used to initialize the labels for multiway cuts in the next frame, thereby reducing the number of labels and increasing computational efficiency. The other extension is on stereo, where hard constraint points are used to prevent the algorithm from smoothing out small or thin long objects in the depth maps. In addition, an interactive technique for selective occlusion detection is also presented. Results for all proposed extensions are given for real images.

DEDICATION

To my mom and dad.

ACKNOWLEDGMENTS

I am indebted to my advisor, Stanley Birchfield, a man of kindness, for his overall support and encouragement. It has been a wonderful experience learning from him and many of the technical discussions with him would always be memorable.

I express deep gratitude to Dr. Adam Hoover and Dr. John Gowdy for reviewing this thesis.

I am also thankful to my roommates, colleagues and friends for a truly enjoyable life at Clemson.

TABLE OF CONTENTS

Table of Contents (Continued)

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

List of Figures (Continued)

Figure Page

Chapter 1

Introduction

With the advent of the video camera, the way we see things in this world has changed significantly. Images and video are becoming a part of everyday life. Large amounts of video data are being handled in the technological field. New applications are emerging in many commercial and industrial sectors that demand efficient processing of video information without any human intervention. The field of computer vision has been able to provide myriad computational tools to analyze and process these video and image data. In broad terms, object recognition, shape representation, classification, tracking, segmentation/grouping, matching/correspondence and 3D reconstruction are some of the high level goals of this field. In this thesis, the specific problem of segmenting image sequences primarily using motion information is studied.

Motion acts as an important cue in visual understanding of real world scenes. In the analysis of image sequences, the importance of motion information cannot be overemphasized as it is an underlying factor that helps in segmenting an image into regions that are homogenous in aspects such as color, texture and semantically meaningful objects. Motion segmentation is a fundamental technique in analyzing image sequences and its output can be used for further processing. For example, if object recognition is to be performed on a complex video sequence exhibiting motion, then it would be much easier to perform motion

segmentation first and process the resulting segments. By this way recognition algorithms can concentrate on important regions in the image and ignore uninteresting details such as background. Motion analysis and associated video processing techniques are also used in robotic navigation, video compression, video indexing and retrieval, object tracking, surveillance and numerous industrial applications.

1.1 Motion Segmentation

Motion segmentation is basically defined as grouping of pixels that are associated with a smooth and uniform motion profile. The segmentation of an image based on motion is a problem that is loosely defined and ambiguous in certain ways. Though the definition says that regions with coherent motion are to be grouped, the resulting segments may not conform to meaningful object regions in the image. To alleviate this problem, the motion segmentation problem is placed at two levels namely, low level and high level. Low level motion segmentation tries to group pixels with homogeneous motion vectors without taking any other image information such as color or shape. This is formally defined as carving out differently moving disjoint regions in an image in a sequence [4]. High level motion segmentation divides the image into regions that exhibit coherent motion and also uses other image clues to give image plane segments that are projections of real semantic objects.

In this thesis, motion sequences that contain a small number of 3D moving objects are considered for processing. High level motion segmentation of these sequences is performed by carrying out a low level motion segmentation task initially and then by using appropriate motion models. Traditionally, the motion segmentation task uses two consecutive image frames in discerning motion regions in the image. There are also approaches to motion segmentation where a sequence of image frames is taken and processed. This kind of processing uses information from both the spatial and the temporal domain. This is termed as spatiotemporal segmentation of image sequences. Spatiotemporal segmentation of video

is a general concept that is very popular in the area of video coding and video compression. In this type of space-time processing, the video 'volume' is separated into 3D blobs in such a way that each of these blobs represent some definite entity. For example, a video showing a single person moving may be analyzed and only the volume corresponding to that person may be extracted. This can be achieved using a variety of information such as object shape, color [12] and motion over time. In this thesis, work on extending a specific two frame motion segmentation algorithm to image sequences is performed. This also comes under the category of spatiotemporal segmentation, since spatial relationships are captured in frame-to-frame correspondence and the temporal relationship exploited by estimating segments from one frame to the next.

1.2 Outline of Proposed Algorithms

In this thesis, the problem of dividing images into distinct non-overlapping 2D segments mainly using motion information is studied. Also, the inter-frame dependence of motion segmentation in image sequences is utilized. The segmentation at any frame in the sequence is driven by motion segments obtained in the previous frame. An algorithm is proposed for motion segmentation over image sequences that uses powerful combinatorial mathematical tools. Specifically, the multiway cut technique for frame-to-frame correspondence developed by [11] is used. This technique was extended in [6] using the affine transformation. This paper is implemented from ground up and analyzed extensively in this thesis. Here multiway cuts and affine transformation techniques are used for stereo correspondence and two frame correspondence in motion. This technique is extended to perform motion segmentation on a sequence of images rather than just two frames.

The exact algorithm is detailed in Chapter 5 and it is briefly explained here. The multiway cut tool with affine displacement functions is run on the first frame of the image sequence. This produces a segmented image with each region having some calculated affine

parameters. The regions are warped based on these parameters to give an estimate of motion regions for the next frame. This is used to initialize the multiway cuts for this frame. The process is then continued for all the frames in the sequence to extract the motion layers. Another technique for extending the stereo correspondence work of [6] is presented. Here hard constraint points, for the disparity map, obtained using normalized correlation are used to initialize the multiway cuts. This is done to prevent the algorithm from removing small and thin long objects, essential for representing the scene structure, from the disparity map. A simple interactive technique for determining occluded regions in images is also proposed. This is performed by residue computation using affine parameters.

The following goals are used to make the algorithm comparable to some degree with existing state-of-art methods.

1. Accuracy of motion segments - results of the algorithms must obey ground truth to a reasonable extent.

2. Computational time - computer calculations must be feasible and processing time must be minimal.

3. Robustness - must be stable and able to handle different image sequences.

1.3 Overview of Thesis

The thesis is organized into six different chapters. Each of these chapters presents theoretical and practical research material regarding the problem of motion segmentation as well as details of the proposed algorithms, comparative analysis and experimental results.

The next chapter is a review of current and past research work on motion analysis and motion segmentation. This chapter gives a summary of different algorithms that deals with image motion and its use in segmentation. The algorithms summarized in this section

belong to several different areas inside the mathematical domain ranging from probability and statistics to level sets and partial differential equations.

The chapter on image motion and motion models talks specifically about mathematical computation of motion information and two different motion models. Theory on optical flow is discussed, and its importance on motion understanding is given. Translation motion model is briefly considered, and affine motion model is dealt in detail. The procedure to extract region based affine parameters is also addressed.

The graph cuts and energy minimization chapter talks about specific graph-based algorithms for various computer vision tasks. Topics such as binary segmentation using maximum flow, formulation of segmentation as energy minimization, and multiway graph cut techniques are explained. The performance analysis of these systems on some real sequences is also examined.

The next chapter contains all the algorithms experimented and their corresponding results. This chapter talks about how the motion segmentation problem for image sequences can be solved using existing multiway cut formulations. Extensions to improve current stereo correspondence results are discussed. Occlusion detection is investigated, and results for selective regions are given.

The final chapter summarizes the work put forth and discusses future possible extensions to the current techniques. The direction of subsequent work for improving results and increasing efficiency is also expressed.

Chapter 2

Motion Analysis and Segmentation: A Review

Motion in the image plane is a problem that has been studied quite extensively by researchers in computer vision. Analysis of motion is a critical step before motion information can be used for any image processing task. The computation of 2D motion vectors across an image either densely or sparsely is a difficult task that proceeds with several assumptions. Any motion estimation technique assumes certain image criteria to be true, and the efficiency of motion analysis and motion segmentation tools depends on how well these assumptions are true and how best these algorithms fit all those assumptions into a proper framework.

In this section, several existing algorithms that perform motion estimation and motion segmentation are discussed. Some of these algorithms are landmark works in image motion analysis and they carry out these tasks remarkably in practice. These algorithms give an overview of the problem of image motion estimation, theoretical difficulties with computation, and mathematical scope.

2.1 Lucas-Kanade Feature Tracking

Here we briefly explore the pioneering work done in the problem of sparse correspondence in images called feature tracking. It was introduced first by Lucas and Kanade in 1981 and then later extended by Kanade and Tomasi [28] and Shi and Tomasi [27]. Features are points or small regions in the image. In [27] optimal features are selected from an image and tracked over the sequence. Since feature selection was performed from the point of view of tracking, both the selection and the tracking tasks are optimized. Feature tracking uses two motion models, the translation motion model and the affine motion model. The translational motion model was used to perform frame-to-frame tracking and the affine motion model is used for consistency check of features tracked over longer sequences. The primary equation in feature tracking is given below.

$$\epsilon = \int\int_W [J(A\mathbf{x} + \mathbf{d}) - I(\mathbf{u})]^2 w(\mathbf{x}) d\mathbf{x} \tag{2.1}$$

This equation tracks a feature window from one image to another by finding a matching region by minimizing the sum of squared differences, ϵ, of the warped window and the original window. In this equation, I is the current image, J is the next image and A and $\mathbf{d}$ refer to affine motion parameters. $w(\mathbf{x})$ can be set to Gaussian function to give importance to pixels in the center of the window, or to all ones. The estimation of all the parameters involves minimizing ϵ, the residual error. For this, the equation is differentiated with respect to the unknown parameters and set to zero. After linearization by truncated Taylor series and manipulation, we get the well known T matrix of the equation that is shown below.

$$T\mathbf{z} = \mathbf{e} \tag{2.2}$$

In this equation, T captures the spatial variation of image gradients inside the window and e denotes the error between the matching windows in the two images. The eigenvalues of the T matrix give an idea of the quality of the feature window. If both eigenvalues

are small, the window is of fairly uniform intensity, if one is small and other is large, the window has a steep gradient in one direction and if both are large then the window has good texture and it can be easily tracked. But even if a window has good texture it may not be a good one to track because it might not represent a real point in the world, for example it may be over a depth discontinuity. To avoid such features, dissimilarity measures of the feature windows tracked over the frames are calculated and features whose dissimilarity goes out of limit are abandoned.

Once the problem of selecting and tracking of features is solved, we are faced with the problem of grouping the features to get an estimate of coherently moving objects in the image. Grouping of features is one way to get the motion information in the image. To avoid the heavy computational burden of calculating dense motion fields, motion vectors of large number of sparse feature points spread over the image can be calculated and those features could be grouped [4]. Here, only features that are successfully tracked from one frame to the other are considered. Also the frames are to be sufficiently apart in time for motion of different features to be distinguishable. Then the three-step algorithm, which involves the growth of group of features starting from the center one, is implemented to attain partitioning of the image into groups of features based on different motion profiles.

2.2 Segmentation and Tracking by Normalized Cuts

Segmentation and tracking can be achieved using normalized cuts, introduced by Shi and Malik [26]. Motion segmentation was performed by computing the normalized cut of a weighted graph obtained by estimating the motion profiles [25]. Normalized cuts come under the general category of graph-theoretic clustering. Since knowledge of segmentation in previous frames is incorporated, efficient grouping is attained at the current frame. Normalized cut technique is a global measure that reflects both the similarity within the segmented partitions, as well as the dissimilarity across the partitions. Normalized cut

techniques involve computation of eigenvectors y from the generalized eigenvalue system defined below.

$$(D - W)y = \lambda Dy \tag{2.3}$$

In this equation, D represents the connectivity information of all nodes in the graph that is being cut. The spatiotemporal affinity matrix W is obtained from the motion profile, which is a measure of the probability distribution of the image velocity at each pixel. In assigning weights to the edges in the graph, similarity measure between image patches based on SSD difference is used. After W and D matrix are formed the eigenvectors corresponding to first few eigenvalues are computed and using these, a recursive repartition algorithm is run to segment the image. When the image sequence is long, only a fixed number of image frames centered on each incoming image frame in the time domain is used to avoid computational complexity.

2.3 Layered Representation of Moving Images

A system to represent moving images with sets of overlapping layers is proposed in [29]. Here the task is to decompose the given image into a set of depth-ordered layers with each layer corresponding to different semantic objects in the image. This representation defines three maps, namely intensity map, alpha map and velocity map for compositing layers ordered in depth. The intensity map gives color information about the layer, the alpha map serves to define the opacity or transparency of the layer at each point and velocity map describes the warping over time.

The layered representation actually involves two steps. The first step is the segmentation step followed by a layer synthesis step. Segmentation step involves partitioning the image into non overlapping regions. This involves optical flow based motion analysis and the segmentation by affine model fitting. k-means clustering method has been used to

achieve this segmentation. Synthesis of layers is arrived at after considering the motion information of layers accumulated over time. Corresponding regions in two image frames are allowed to differ only by affine transformation and bicubic interpolation is used for motion compensation. Alpha and intensity maps are then estimated. Median filtering operation is applied on all data points over a layer to preserve edge information and finally foreground-background relationships are determined.

2.4 EM Algorithm for Motion Segmentation

This section summarizes the work done by Weiss and Adelson [30] in incorporating spatial coherence in the mixture framework, a probabilistic approach, for motion segmentation. Here, the addition of a spatial constraint to the mixture formulations and the use of a variant of the EM algorithm using both the form and motion constraints have been performed. Mixture estimation performs estimation of parameters given data that was generated by multiple processes. The EM algorithm is a special case of mixture estimation in a way that there is incomplete data. E stands for estimation and M for maximization.

Estimation step refers to assignment of data points to the appropriate models given the parameters of the models and Maximization refers to estimation of parameters given the association of data points to the models. In motion analysis, parameters are the ones that describe motion predicted by the model. The E step works to minimize the residue or the error between the observed motion and the motion predicted by model parameters. The M step then updates the model parameters based on data point assignments. To add spatial constraint, static intensity cues are used for model prediction for pixels in a fragment. The likelihood of nearby pixels belonging to the same model is also exploited. The performance of the algorithm for different video sequences is presented. It is observed that future investigations are possible in adding advanced static form constraints to achieve robust scene segmentation.

2.5 Space-Time Algorithms

Spatiotemporal segmentation is one of many ways to content-based analysis of image sequences. In [24] segmentation is performed in space-time using hierarchical mean shift analysis. In this method, all the frames in an image sequence are stacked up into a 3D block and for every pixel in this block, a 7D feature vector is computed that takes local optical flow, color and position information. Once all the pixels are taken to this higher dimensional space they are grouped in that dimension using clustering techniques of mean shift segmentation. The hierarchical clustering method works by the repeated application of mean shift analysis over increasingly large ranges. Mean shift analysis is a good clustering/grouping tool that is exploited here with the motion segmentation problem getting solved in the spatiotemporal domain.

This paper also has a good classification scheme of all segmentation methods that uses motion information. They claim that all algorithms for motion based image and video segmentation fall into the following general categories. 1. Image features such as motion, color and texture are used to perform a 2D grouping and these groups are grown in the temporal direction, 2. Discrete features or interest regions are first tracked temporally and then those features are grouped based on motion trajectories. 3. Segments or interest regions are simultaneously grouped spatially and temporally.

2.6 Graph Algorithms

Graph based algorithms [10, 21, 9, 19, 17] for motion estimation are robust and efficient. These algorithms construct a graph of nodes and links with usually nodes representing pixels in the image and links enforcing local connectedness of these pixels. The graph is then processed using algorithms such as maximum flow and multiway cuts to achieve tasks such as correspondence and grouping. For example, in the stereo correspondence problem, each node in the graph will be labeled with particular disparity based on the penalty that it

receives for any given label and labeling of neighborhood pixels. This local computation is put into a global energy minimization framework [11] and the multiway cut engine finds a good local minimum, if not global minimum. Graph based algorithms are explained in more detail in Chapter 4.

Chapter 3

Image Motion and Motion Models

The world, as we perceive through our eyes, is constantly changing. These changes are caused due to (a) the movement of objects in the scene, (b) the movement of eyes to focus a different scene and (c) changes in lighting across the scene. Two of these occurrences result due to some physical movement and the third one is the manifestation of variation in illumination. This visual motion helps us in understanding what is going on in the real world. In the same way, images of the real world captured by camera are analyzed by machines automatically to perform various tasks. In order to do this, a representation of the visual motion is desired and this is where low level motion analysis techniques such as optical flow, translation motion model and affine motion models are used. This section explains about these motion models and their parameters. The affine motion model is described in detail as it is used in most of the algorithms discussed in this thesis.

3.1 Optical Flow

Optical flow is the 2D velocity field, describing the apparent motion in the image that results from independently moving objects in the scene or from observer motion [7]. It is a low level image processing tool that is utilized by many high level computer vision techniques. The real motion in the world can be represented as a set of 3D vectors and the

projection of these vectors on to the image plane gives rise to 2D image motion field. The apparent motion in the image or the optical flow field is theoretically different from the image motion field. Although estimation of the motion field is what is required most of the time, only optical flow field can be measured. But it is a very good approximation of the motion field.

Optical flow equations are derived using the brightness constancy assumption, given below.

$$I(x, y, t) - I(x + dx, y + dy, t + dt) = 0 \tag{3.1}$$

This equation states that the light intensity value of any 3D world point projected on the image plane does not change value due to motion. This is used to derive the optical flow equation shown below.

$$I_x u + I_y v + I_t = 0 \tag{3.2}$$

In this equation, I_x and I_y are spatial image gradients at (x, y) and I_t is the temporal gradient. The image velocity at (x, y) is represented as (u, v). An equivalent representation of the optical flow is given below.

$$(\nabla I)^T \mathbf{u} + \frac{\partial I}{\partial t} = 0 \tag{3.3}$$

In this equation, ∇I captures the spatial gradient in the image and $\frac{\partial I}{\partial t}$ stands for temporal gradient. $\mathbf{u}$ is the 2D optical flow vector, which is the same as image velocity (u, v) as denoted before. The optical flow equation is an under-constrained equation that has two unknowns. All algorithms that compute motion using optical flow employ additional constraints to solve the equation. If pixel (x_1, y_1) in frame t and pixel (x_2, y_2) in frame $t + 1$ are estimated to correspond to the same world point, then the optical flow at (x_1, y_1) in frame t is the 2D vector $(x_2 - x_1, y_2 - y_1)$. This is (u, v) as in Equation 3.2. The optical flow equa-

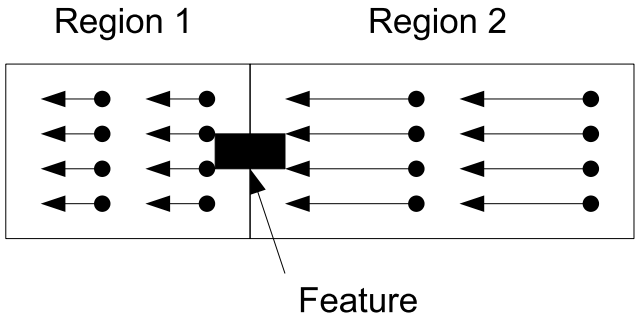


Figure 3.1: Feature straddling a motion discontinuity.

tion is an underlying equation to many motion models. Almost all of the motion models can be proven to support the optical flow equation. We will show this using the translation motion model described in the next section.

3.2 Translation Motion Model

In this section, we discuss about feature tracking, specifically that of Shi-Tomasi [27] tracking, developed from the original work of Lucas and Kanade [23]. Here small rectangular features are tracked from frame to frame. Good features are chosen using the amount of texture present inside each feature. Feature tracking proceeds using the simple translational motion model. This motion model assumes that all pixels inside a small rectangular support region in the image are exhibiting constant translational motion. The validity of this assumption depends on inter-frame motion or the sampling rate, the size of the rectangular region and whether that feature is lying inside a homogenous motion region or straddling a motion discontinuity. The specific case of a feature window straddling a motion discontinuity is shown in Figure 3.1. In these cases, assumption on constant motion inside the feature window fails.

The Shi-Tomasi translation based feature tracking equations [3] are obtained by minimizing the error, ϵ, given by

$$\epsilon = \int\int_W [J(\mathbf{x}+\mathbf{d}) - I(\mathbf{x})]^2 w(\mathbf{x}) d\mathbf{x} \tag{3.4}$$

where,

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}, \text{ and } \mathbf{d} = \begin{bmatrix} dx \\ dy \end{bmatrix}. \tag{3.5}$$

This equation can be rearranged using Taylor series expansion to give,

$$T\mathbf{d} = \mathbf{e} \tag{3.6}$$

where,

$$T = \int\int_W (\nabla I)^T (\nabla I) w(\mathbf{x}) d\mathbf{x}, \text{ and} \tag{3.7}$$

$$\mathbf{e} = \int\int_W [I(\mathbf{x}) - J(\mathbf{x})](\nabla I)^T w(\mathbf{x}) d\mathbf{x}. \tag{3.8}$$

The same translation equation can be derived using a series of optical flow equations for the same rectangular feature in the image. Using optical flow equation from 3.3, and changing notation, we obtain

$$\mathbf{g}^T\mathbf{d} = -I_t. \tag{3.9}$$

Applying this equation for a set of pixels, we get

$$\mathbf{g}_1^T\mathbf{d} = -I_{t1}, \;\; \mathbf{g}_2^T\mathbf{d} = -I_{t2}, \; \ldots \; \mathbf{g}_n^T\mathbf{d} = -I_{tn}. \tag{3.10}$$

Writing in matrix form,

$$\begin{bmatrix} \mathbf{g}_1^T \\ \mathbf{g}_2^T \\ \dots \\ \mathbf{g}_n^T \end{bmatrix} \mathbf{d} = - \begin{bmatrix} I_{t1} \\ I_{t2} \\ \dots \\ I_{tn} \end{bmatrix}. \tag{3.11}$$

This can be written as,

$$A\mathbf{d} = \mathbf{b}. \tag{3.12}$$

The Equation 3.12 and Equation 3.6 are equivalent and hence the feature tracking equations have their base in optical flow. This shows the importance of optical flow and shows that it forms the seed for many motion models. The translational motion model is simple, computationally efficient and very effective when the assumptions discussed above hold good. But when dense motion estimation is required or when those assumptions are violated translation motion model simply fails. A more robust motion model is the affine motion model and it is discussed in the next section.

3.3 Affine Motion Model

The affine motion model is a popular technique because it is a better approximation, than the translation motion model, in representing the motion of a region in an image. The projection of the real world onto the image plane introduces perspective distortions. These changes can be perfectly modeled by a mathematical equation if the geometry of the scene is known. If we have a good approximation to these equations, constructing those equations is not needed, even if we know the scene geometry. The affine motion model is one that can approximate a variety of image motion by employing a linear combination of translation, rotation and scaling operations. The important advantage of affine transformation is that it is a balance between lower order motion models like translation and higher order models

such as projective and B-spline [22]. Lower order models are computational efficient but poor in accuracy and higher order models are more accurate but involve estimation of lot of parameters. Affine model is a good trade-off between accuracy and computational efficiency. The affine motion model can support a bigger region undergoing coherent motion than the translation motion model.

3.3.1 Smoothing and Gradient computation

Before we discuss affine transformation, smoothing and gradient computation of images must be explained. The affine computations require the smoothing of images to make the computed parameters less sensitive to the noise in the image. Gaussian based smoothing of images have been proved to be the most elegant way because the Gaussian function can conclusively model most of the naturally occurring noise in the image. Also the Gaussian function given below has the separability property that makes the smoothing computation easier. Two 1D Gaussian kernels can be computed and successively convolved with the original image to get the smoothed output. The standard deviation of the kernel determines the extent to which noise is smoothed. The sigma can neither be too low nor be too high as low values would not eliminate noise and high values would flat out important intensity edges. The Gaussian functions and the derivative of Gaussian functions are given in Equations 3.13, 3.14, and 3.15.

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \quad G(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{y^2}{2\sigma^2}} \tag{3.13}$$

$$G(x,y) = \frac{1}{\sigma^2 2\pi} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.14}$$

$$\frac{\partial G}{\partial x} = -\frac{1}{\sigma^3\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \quad \frac{\partial G}{\partial y} = -\frac{1}{\sigma^3\sqrt{2\pi}} e^{-\frac{y^2}{2\sigma^2}} \tag{3.15}$$

The next step would be to compute the gradient or the first derivative of the image. The image gradients can be computed using a number of operators such as Sobel and Prewitt. But usually image gradients are achieved by the computation of the gradient of the Gaussian kernel and subsequently convolving the image with the computed kernels. From the equation it can be seen that the derivative of Gaussian also enjoys the separability property and hence two separate 1D kernels can be evaluated and convolved separately. This produces horizontal and vertical gradient images. Both discrete approximations of the Gaussian kernel and the derivative of Gaussian kernels are normalized before convolution.

3.3.2 Affine Parameters Estimation Procedure

The affine motion model comprises of six parameters, conventionally represented using two matrices *A*, **d**. *A* is a 2×2 matrix whose parameters define the amount of scaling, rotation and shearing of the region and **d** is the translation component that describes the vertical and horizontal linear motion in the image. Hence **d** is a 2D vector and there are a total of 6 parameters for the affine model. Before beginning the estimation of affine parameters the origin for all the location of pixels can be shifted to the centroid of that particular region to get a better approximation. This is especially true with a fixed camera and different objects in the scene undergoing dissimilar motion.

Affine computation procedure is iterative, that is an equation is solved to get the incremental addition to each of the six parameters. The process is repeated till convergence. The convergence can be checked by two different ways. One is by checking if the residue keeps decreasing in every iteration and the iteration can be stopped when residue starts to increase or the incremental addition to all the six parameters becomes insignificant and goes below a threshold. The validity of the computed parameters can be checked using the residue level for all regions for which affine parameters need to be computed. The residue level threshold will depend on the amount of noise present in the original image.

The six parameters can be estimated using equations of [27], which are explained here with respect to the computation procedure. The matching error, ϵ, can be expressed as

$$\epsilon = \sum \sum_{W} [J(A\mathbf{x} + \mathbf{d}) - I(\mathbf{x})]^2 w(\mathbf{x}) d\mathbf{x} \tag{3.16}$$

where, I is the reference image, J is the given image,

$$A = \begin{bmatrix} (1 + d_{xx}) & d_{xy} \\ d_{yx} & (1 + d_{yy}) \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}, \text{ and } \mathbf{d} = \begin{bmatrix} dx \\ dy \end{bmatrix}. \tag{3.17}$$

Using Taylor series approximation,

$$J(A\mathbf{x} + \mathbf{d}) = J(\mathbf{x}) + (d_{xx}x + d_{xy}y + d_x)g_x + (d_{yx}x + d_{yy}y + d_y)g_y \tag{3.18}$$

where,

$$g_x = \frac{\partial J(\mathbf{x})}{\partial x}, \text{ and } g_y = \frac{\partial J(\mathbf{x})}{\partial y}. \tag{3.19}$$

Differentiating 3.16 with respect to $\mathbf{z} = [d_{xx}\ d_{xy}\ d_x\ d_{yx}\ d_{yy}\ d_y]$,

$$\frac{\partial \epsilon}{\partial \mathbf{z}} = 2 \sum \sum_{W} [J(A\mathbf{x} + \mathbf{d}) - I(\mathbf{x})] \begin{bmatrix} xg_x \\ yg_x \\ g_x \\ xg_y \\ yg_y \\ g_y \end{bmatrix} w(\mathbf{x}) d\mathbf{x} = 0. \tag{3.20}$$

Rearranging terms,

$$T\mathbf{z} = \mathbf{e} \tag{3.21}$$

where,

$$T = \sum\sum_{W} \begin{bmatrix} x^2{g_x}^2 & xy{g_x}^2 & x{g_x}^2 & x^2g_xg_y & xyg_xg_y & xg_xg_y \\ xy{g_x}^2 & y^2{g_x}^2 & y{g_x}^2 & xyg_xg_y & y^2g_xg_y & yg_xg_y \\ x{g_x}^2 & y{g_x}^2 & {g_x}^2 & xg_xg_y & yg_xg_y & g_xg_y \\ x^2g_xg_y & xyg_xg_y & xg_xg_y & x^2{g_y}^2 & xy{g_y}^2 & x{g_y}^2 \\ xyg_xg_y & y^2g_xg_y & yg_xg_y & xy{g_y}^2 & y^2{g_y}^2 & y{g_y}^2 \\ xg_xg_y & yg_xg_y & g_xg_y & x{g_y}^2 & y{g_y}^2 & {g_y}^2 \end{bmatrix}, \text{ and} \tag{3.22}$$

$$\mathbf{e} = \sum\sum_{W} [I(\mathbf{x}) - J(\mathbf{x})] \begin{bmatrix} xg_x \\ yg_x \\ g_x \\ xg_y \\ yg_y \\ g_y \end{bmatrix} w(\mathbf{x}) d\mathbf{x}. \tag{3.23}$$

The affine equations derived above is classified as the forwards-additive method in the paper by Baker-Matthews [2]. Though this is the conventional type of affine estimation so far they claim that inverse compositional method of computation is the most effective for any parametric motion model. The exact algorithm for affine computation of an arbitrary shaped region is given here.

1. Smooth the images *I*,*J* using Gaussian and compute gradients of *J* using Laplacian of Gaussian.

2. Find the centroid of the region and convert all pixel locations in the region to centroid based values.

3. Initialize *A* and **d** and compute matrices *T* and **e**. Solve $T\mathbf{z} = \mathbf{e}$.

4. Incrementally add values in **z** vector to corresponding values in *A* and **d**.

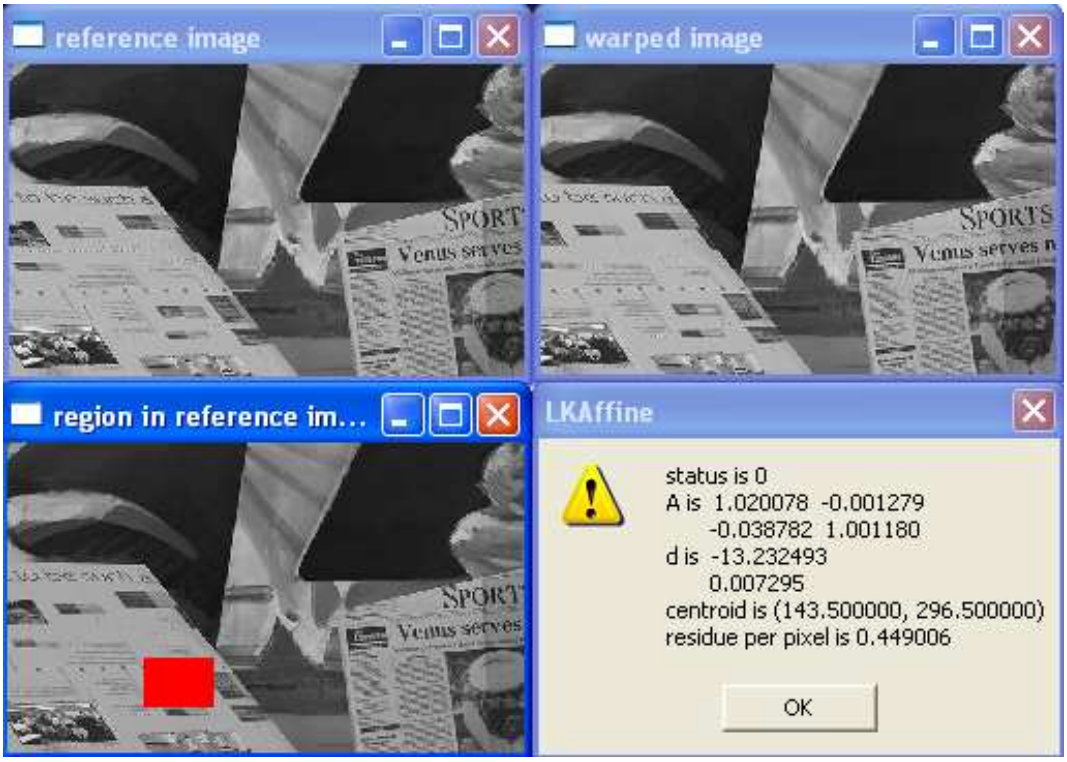


Figure 3.2: Affine parameter estimation of a region for a stereo pair of images.

5. Repeat process from step 3 till **z** less than threshold.

6. Check the validity of the parameters by residue computation.

The screen shot of Figure 3.2 shows affine computation for a stereo pair of images. A rectangular region in the reference image, left image here, is chosen and affine parameters are fitted to the warped image, right image. The parameters show that the region's displacement between the left and the right image is about 13 pixels. The screen shot of Figure 3.3 shows affine computation for consecutive frames of an image sequence. The region is on the vehicle and the parameters have clearly captured the region's translation and enlargement, as the vehicle approaches the camera.

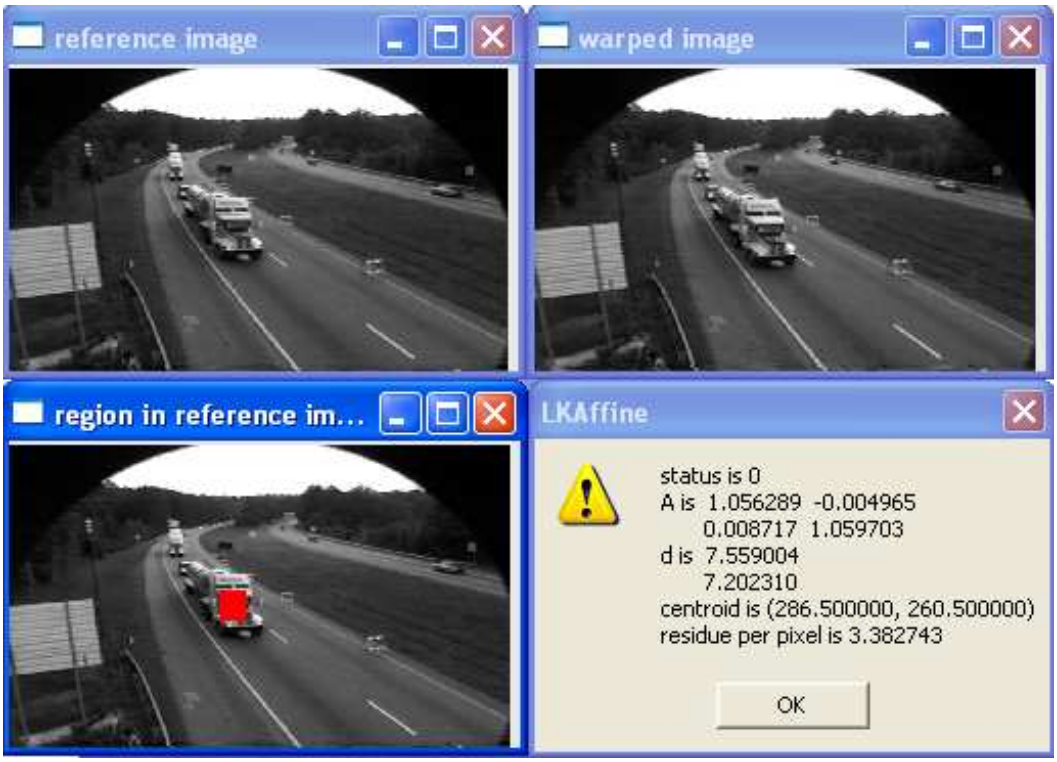


Figure 3.3: Affine parameter estimation of a region for motion images.

Chapter 4

Graph Cuts and Energy Minimization

Many problems in computer vision can be solved by energy minimization. The problem is usually formulated by the construction of an energy functional and the global minimum function of this energy functional is sought. There are two issues involved with this approach. One is designing the problem conditions into convenient energy terms and the other is choosing an appropriate minimization tool that can work with the designed energy. The disadvantage with this is that when the results are not as expected it would be difficult to point out where the algorithm failed. It may be because the energy functional did not represent the problem very well or the minimization mechanism faulted. But the advantage with this approach is that it provides a way for combining different heuristics to solve the problem into one clean framework.

In this chapter, we specifically discuss energy minimization for image segmentation using powerful graph based algorithms [14]. An important drawback with energy minimization algorithm is the computational complexity of the minimization procedure. One popular tool that is used for energy minimization is simulated annealing [15]. This is a stochastic method that can accurately find the global minimum of any energy functional but takes unreasonable amount of time for computation. Graph cut based tools for energy minimization are superior to other comparable methods in terms of computational speed.

They are fast but the trade-off is that they operate on restricted energy functionals and they are not bound to calculate the global minimum as the problem is NP-complete, but nevertheless finds a good local minimum. We have used graph based algorithms for motion segmentation. The motion segmentation issue is treated as an energy minimization problem and graph cuts minimizes the corresponding energy. This section lays down material on binary and multi-level motion segmentation, equivalently binary and multiway cuts on graphs.

4.1 Graph Constructions

All graph based algorithms construct a graph of nodes and edges, with the nodes and edges representing some meaningful entity related to the problem. For example, the recent graph based techniques [11] for image correspondence problems, especially stereo, construct graphs with nodes that represent pixels in the image. The edges on the graph forms links between nodes and these edges carry some weight. Usually edges are present only between nodes that represent neighboring pixels in the image. The neighborhood is typically either 4 or 8 surrounding pixels in the case of a normal 2D image. Graph constructions for a small image showing 4 and 8 neighborhood is shown in 4.1. The neighborhood edges are used to enforce the connectedness of a pixel with its neighbors. If we are labeling disparities in the case of stereo or evaluating displacement vectors in the case of motion, for every pixel it is usual to assume that neighboring pixels will get the same label, at least for most part of the image. This heuristic is enforced using neighborhood edges.

The cases discussed above are 2D images where all nodes and edges are within two dimensions. Images can also be stacked up and processed as a bundle at a time rather than processing two frames at a time. In this case we have 3D graphs that extend along horizontal, vertical and ‘time’ directions which are geometrically represented using width height and depth. In this case again, pixels are represented as nodes but pixels from all the

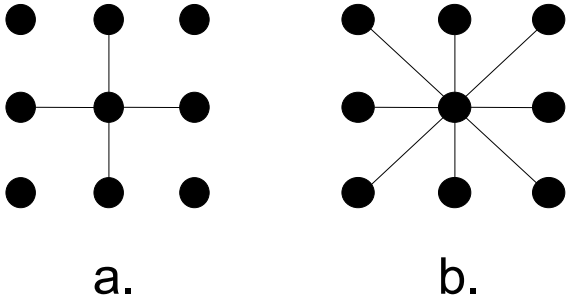


Figure 4.1: Neighborhoods in graph constructions. a. 4-connected and b. 8-connected.

frames are taken and a single big 3D graph is created. As we are dealing with graphs from space-time, edges in the graph are split into two different types, spatial edges and temporal edges. Spatial edges are equivalent to ones in the 2D case and temporal edges are edges that run between nodes belonging to different frames. The connectedness neighborhood in the 3D graph is usually 6 or 26 surrounding pixels.

In the case of graph technique [18] graphs are formed whose nodes represent not pixels but pairs of matching pixels. In this technique, a node means a pair of potentially corresponding pixels. This kind of treatment was needed to label pixels that don't end up getting matched and these are labeled as the occluded pixels. Again enforcing connectedness with a graph of this kind is by checking whether two matching pairs of pixels carry the same disparity if pixels in either of the set are neighbors. Illustration of this graph construction is given in Figure 4.2.

4.2 Energy Functions

When solving a problem using graph based methods, measurements are made on the graph after the construction is completed. There are two kinds of measurements that are computed at each node in the graph, one is the local data measurement at that node and the other is the smoothness measurement based on statuses of neighboring nodes. This can

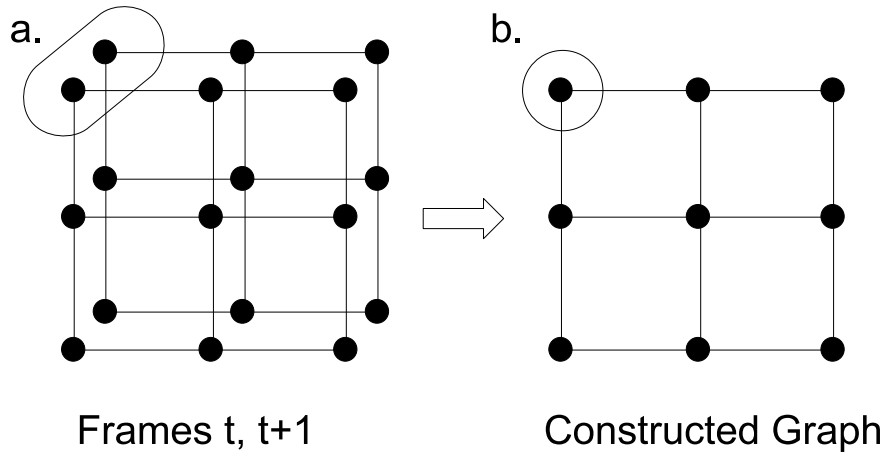


Figure 4.2: A different graph construction method. a. potentially matching pixels and b. corresponding node.

be explained with the help of the stereo example. In stereo, it is common to compute the intensity difference between a pixel in the left image and a pixel in the right image given a disparity value. This kind of computation of intensity difference is the local data measurement that we perform at every node, equivalently at every pixel. Also typically neighboring pixels of similar intensity carry the same disparity level. This kind of checking of intensity levels of neighboring pixels is the smoothness measurement that is performed. The data measurements along with the smoothness measurement at all nodes in the graph dictate the final result of processing or labeling that graph. These measurements in a graph are expressed using energy functions.

In traditional energy minimization techniques associated with the image correspondence problem, energy functions are written to map all possible solution to a specific real valued cost. The mapping of the solution space to the energy cost space enables algorithms to search for global maximum or minimum and correspondingly identify the right solution. Usually energy function terms are formulated in such a way that the ideal expected solution produces the least energy cost. Hence the energy cost for any solution measures the badness [17] of that particular result. Graph cut techniques discussed in this section operate on

specific types of energy functionals to find the least possible cost. The energy functionals take the form expressed below.

$$E = \sum_{\forall p} E_d(p) + E_s(p) \tag{4.1}$$

where,

$$E_d(p) = D(l_p), \text{ and} \tag{4.2}$$

$$E_s(p) = \sum_{q \in N} S(p, l_p, q, l_q), \quad N = \text{neighborhood of } p. \tag{4.3}$$

In Equation 4.1, E_d represents the cost of assigning labels to pixels and E_s enforces piecewise continuity along the spatial dimensions (x, y). p is any pixel in the image. The Equation 4.2 and Equation 4.3 show data and smoothness costs respectively given the labels of pixels. l_p and l_q denote the labels of pixels p and q. Generally, graph based algorithms discussed here handle only pair-wise smoothness cost to make the algorithm computationally faster, although they can manage a larger collection of interacting pixels at the expense of speed of operation [11].

The first term in the energy function, which is the data cost term, can be designed to suit the problem at hand. In the case of stereo and motion, the inherent task is to match pixels that correspond to the same world point. In case of stereo, the displacement between a pixel and its matching pair is a 1D value. This is called 'disparity' that depends on the depth of the pixel's world point from the camera. In the case of motion, displacement is a 2D vector and it can be in any of the four quadrants. In both cases, matching pairs of pixels have similar intensity values if effects such as noise and occlusions are not considered. Hence the data term is designed to be any one of the following ways. D_1 in Equation 4.4 and Equation 4.8 represent dissimilarity of intensities by squared difference and D_2

in Equation 4.5 and Equation 4.9 gives dissimilarity of intensities by absolute difference. $D(l_p)$ could be either $D_1(l_p)$ or $D_2(l_p)$.

For stereo,

$$D_1(l_p) = [I_L(p) - I_R(l_p(p))]^2, \text{ and} \tag{4.4}$$

$$D_2(l_p) = \mid I_L(p) - I_R(l_p(p)) \mid \tag{4.5}$$

where,

$$p = (x^l, y^l), \;\; l_p(p) = (x^r, y^r), \tag{4.6}$$

$$(x^r - x^l) = \delta = \text{disparity, and} \;\; (y^l - y^r) = 0. \tag{4.7}$$

For motion,

$$D_1(l_p) = [I_t(p) - I_{t+1}(l_p(p))]^2, \text{ and} \tag{4.8}$$

$$D_2(l_p) = \mid I_t(p) - I_{t+1}(l_p(p)) \mid \tag{4.9}$$

where,

$$p = (x^t, y^t), \;\; l_p(p) = (x^{t+1}, y^{t+1}), \text{ and} \tag{4.10}$$

$$(x^t - x^{t+1}, y^t - y^{t+1}) \in \text{ all 4 quadrants.} \tag{4.11}$$

If all the pixels in the image get the right disparity or the right displacement vector then the summation of data cost for all the pixels in the image will be at a minimum, provided there is no noise or occlusions in the image. Now if noise and occlusions are present, then the data cost for the actual displacement vector at a pixel may not be a minimum. To elegantly handle this problem, we need a smoothness cost that will pull noisy pixels from

getting incorrectly labeled. Also to handle problems of sampling from affecting the result we compute the intensity difference, using [5], as shown below.

$$D(l_p) = \min(d_R, d_L) \tag{4.12}$$

where,

$$d_R = \min_{x^r-\frac{1}{2}\leq u\leq x^r+\frac{1}{2}} \mid I_L(x^l) - \hat{I}_R(u) \mid \text{, and} \tag{4.13}$$

$$d_L = \min_{x^l-\frac{1}{2}\leq u\leq x^l+\frac{1}{2}} \mid \hat{I}_L(u) - I_R(x^r) \mid . \tag{4.14}$$

This dissimilarity compares the linearly interpolated region around the pixel in the left image and the interpolated region around the matching pixel in the right image. Also, bilinear interpolation can be used to compute the intensity value given a floating point location. This could be used to compute the dissimilarity. This dissimilarity for data cost is more appropriate especially for motion and affine fitting.

The smoothness term is a nontrivial term that depends on image noise. Intuitively, to design smoothness cost, any pair of pixels in the image that are next to each other and have similar intensity values are expected to have the same disparity or get the same displacement vector. This is violated only when similar colored objects are involved in occlusion-disocclusion. But this happens rarely. So when two neighboring pixels are labeled with same displacement vector then we set the smoothness cost to a minimum, typically 0, as shown below.

$$S(p, q) = 0, \text{if } l_p = l_q \tag{4.15}$$

If neighboring pixels are not labeled with the same labels then we measure the gradient at that point and set the smoothness cost based on the gradient value as shown in Equation 4.16.

$$S(p,q) = \alpha(\nabla I)_p, \text{if } l_p \neq l_q \tag{4.16}$$

In this equation, α is a positive constant. Intuitively, if the gradient is high then it is more likely that there is a discontinuity and we can set the smoothness cost to a low value and otherwise the cost to a high one. This maximizes piecewise smoothness across the image while penalizing motion and depth discontinuities. To make the computation faster, instead of exact gradient computation at that point, we can measure the difference between intensity values of neighboring pixels and set the costs based on that as given in the equations shown below.

$$S(p,q) = \lambda_1, \text{ if } \mid I_p - I_q \mid \leq I_{th}, \text{ and} \tag{4.17}$$

$$S(p,q) = \lambda_2, \text{ if } \mid I_p - I_q \mid > I_{th} \tag{4.18}$$

where,

$$\lambda_1 > \lambda_2 \,. \tag{4.19}$$

For energy function of Equation 4.20, we have a term for occlusion energy [18]. Here an additional term E_{occ} is introduced to handle occlusions. This energy function denotes that there is a constant occlusion data cost for pixels that are occluded. In this type of energy functionals, with a constant cost for occluded pixels, we minimize the number of occluded pixels in the image. In this method, there is no separate label for occlusion, all pixels in one image that does not have a matching pair in the other image get labeled occluded. Another way to handle occlusions is to have a separate label for occlusion. In this case, we have to devise a term to manage interaction penalties between normal and occluded pixels. This

could have similar numerical values as compared to the smoothness term discussed above or different.

$$E = \sum_{\forall p} E_d(p) + E_s(p) + E_{occ}(p) \tag{4.20}$$

$$E_{occ}(p) = C_p, \text{ if } p \text{ occluded, and} \tag{4.21}$$

$$E_{occ}(p) = 0, \text{ otherwise.} \tag{4.22}$$

4.3 Maximum Flow – Minimum Cut Algorithm

Two graph based algorithms that are extensively used and investigated in this work are binary and multiway cuts. These combinatorial tools are developed from the seminal work of Ford - Fulkerson, who devised optimal algorithms for graph based applications. They developed the famous maximum flow minimum cut theorem for network flows and this has been used in a variety of fields. The algorithm gives a method to calculate the value of maximum flow between two vertices of a network graph. Technically, given a flow network with all edges having non negative capacities and a source and sink vertex, then this method determines the path(s) in the graph that can accommodate the maximum flow between the source and sink vertex. Alternately, it identifies the edges in that graph, whose total capacities are at a minimum, that have to be removed so as to eliminate a direct path between the source and the sink. The total amount of capacities of edges that are cut is called the minimum cut. It turns out that the maximum flow and minimum cut are numerically equal and hence if one problem is solved then we have the solution for the other one also.

The algorithm is based on finding an augmenting path connecting source and sink where more flow can be pushed without exceeding capacity constraints on edges between any

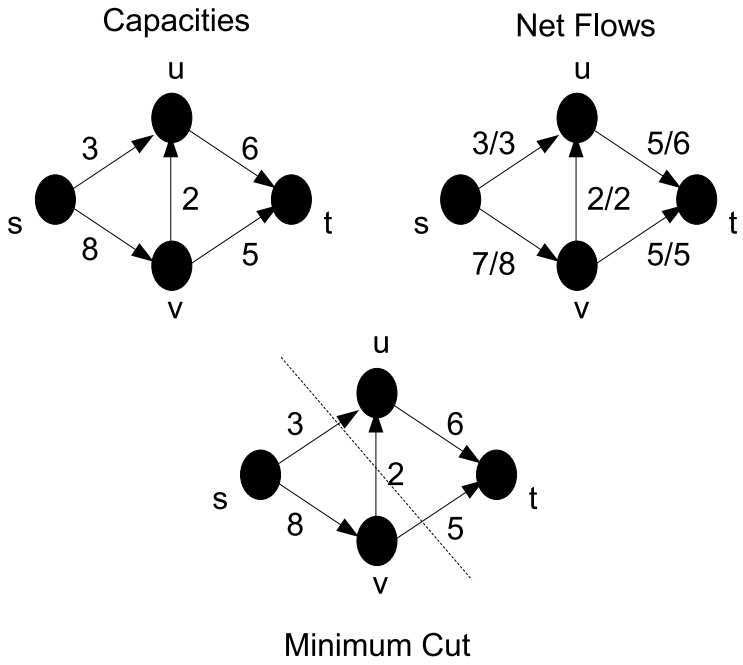


Figure 4.3: Maximum flow and minimum cut.

two nodes. The method is iterative and terminates only when no augmenting path could be found. This algorithm is guaranteed to converge when the capacities in the graph are integers or rational number. If capacities are irrational numbers, then there is no certainty that the algorithm would find the correct maximum flow and it may iterate forever. Given a flow network $G(V, E)$ with source *s* and sink *t*, then the algorithm can be summarized [13] as follows.

1. Initialize flow f to 0.
2. if an augmenting path *p* exists between *s* and *t*.
3. increment flow f along *p* by unit flow.
4. repeat 2-3 and return f.

The maximum flow - minimum cut in a graph is illustrated using the Figure 4.3. Here, there are 4 nodes in the graph *s, t, u* and *v*. All edges between the nodes and the corre-

sponding capacities are shown. The maximum flow between *s* and *t* is found to be 10 and maximum net flow path values are shown in the figure. The minimum cut on the graph is also shown and the value of minimum cut is the sum of capacities of edges that are severed, which is $3 + 2 + 5 = 10$. This is same as the maximum flow. The maximum flow is useful in computing what are called s-t cuts discussed in the next section.

4.4 Binary Segmentation

A fundamental problem in computer vision is the identification of foreground and background regions in an image. This is a typical binary classification problem where pixels in the image have to be labeled as foreground or background. Here we see how graph based methods are used to solve this kind of two-level segmentation. In [16] a single graph cut was shown to find the global minimum of energy functions. In [9] graph cuts are used to do foreground-background segmentation of images interactively. The maximum flow - minimum cut theorem discussed in the previous section can be used to globally minimize energy functions that deal with two labels.

In [21] it is shown how graph cuts can be used to minimize energy functions and what energy functions can be minimized using graph cuts. Especially, it shows energy functions of binary variables can be precisely minimized using graph cuts. For the binary segmentation of images, graphs can be constructed with nodes related to the formulation of the labeling problem. Then two special vertices are introduced in this graph that corresponds to the binary labels 0 and 1, for foreground and background or vice versa. Similarly, the labels 0 and 1 can be termed as source and sink or vice verse. Let us denote the graph by $G(V, E)$ where V stands for the nodes and E for edges. Let s and t be the source and the sink in this graph. $V - \{s, t\}$ denotes all the nodes in the graph excluding the source and sink, which would be called terminals rather than vertices or nodes. After the construction of nodes, the source and sink terminal edges are placed in the graph. These edges are

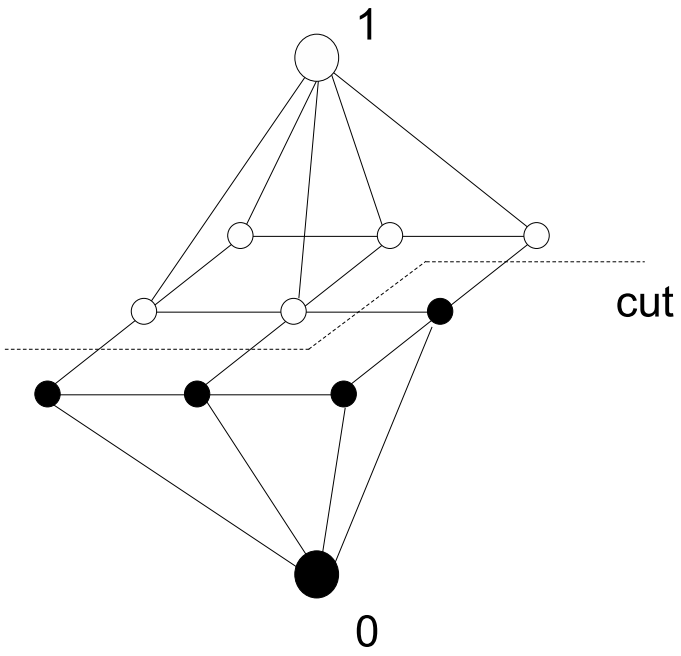


Figure 4.4: Binary cut. Cut terminal links are not shown.

called the t-links for terminal links. Also there are edges between neighbor nodes which are called n-links for neighbor links. The weights on these edges are based on the energy function. There are edges between every node and the source terminal and every node and the sink terminal. The data term in the energy function contributes primarily to the weights of t-links and smoothness cost determines edge weights between nodes. Since it is shown that link weights have to conform to the metric or the semi metric condition, n-link edge weights would be augmented using data costs.

Now, with the graph constructed, the minimum s-t cut on this graph is performed to divide the graph into two parts in such a way that there is no path between the *s* and *t* terminals. This kind of binary cut is shown in Figure 4.4. The combined weight of all edges that are removed to achieve this is minimum is equal to the maximum flow between source and sink. Now, if a t-link between a node *n* and source (s, n) is severed then the node may be labeled as 0. Correspondingly if its sink t-link (n, t) is broken the node would

link	weight
$\{p,s\}$	$D_s + I^p$
$\{p,t\}$	$\mid D_t - I^p \mid$
$\{p,q\}$	$B_s - \mid I^p - I^q \mid$

Table 4.1: Weight assignments for 'bunny' sequence.

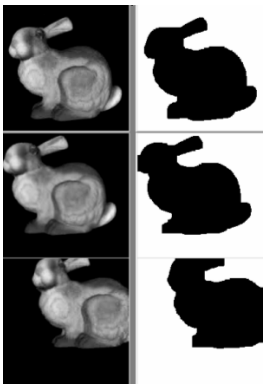


Figure 4.5: Intensity based binary segmentation. Frames 1, 17, and 95 of 'bunny' sequence are shown.

get the label 1. The edge weights in the graph can be reversed to do the opposite where cutting (s,n) might label n as 1 and then cutting (n,t) would give n label 0.

4.4.1 Intensity Based

Here we illustrate the binary segmentation problem using a pseudo sequence. This sequence shows a 'bunny' translating over a dark background. Our goal is to precisely carve out the bunny from the background. This is a simple sequence to process but simple methods like thresholding may not work because there are dark regions inside the bunny with intensity values close to the back ground. Here a simple binary segmentation of the sequence can be achieved using graph cuts. If we devise the energy functions to be based on intensities then the smoothness term will help in retrieving dark regions from the bunny which would not have been possible by thresholding. Also due to rapid frame-to-frame motion in this sequence 2D motion displacement vectors cannot be exhaustively listed and multiway cuts performed as this would be a waste of computation.

The min-cut/max-flow graph cut codes of [17] were obtained to generate the necessary max-flow algorithm library files. As a first step graph construction functions were written specifically for processing this pseudo image sequence exhibiting simple translation motion. The weight assignments for the graph generated for the 'bunny' sequence is given in Table 4.1. The *s* and *t* denote the source and the link terminals for the minimum cut algorithm. The I_p denotes the intensity value for the pixel *p*. The segmentation results for the 'bunny translation' video are shown for frames 1,17 and 95 in Figure 4.5. The values of the parameters were set at $D_s = 20$ and $D_t = 75$ and $B_s = 150$. The results show excellent segmentation for this synthetic sequence. If the link weights to the source terminal and the sink terminal are approximately same for all the nodes, then the segmentation is completely governed by the node- node links, which relate to the boundary term in the energy functional equation. This term works for smoothing out all the regions except for discontinuities. Since motion discontinuities in the bunny image relate clearly with intensity gradients, the weight assignments that were discussed above perform good segmentation.

4.4.2 Optical Flow Based

The minimum cut algorithm can be used to minimize energy functions based on optical flow. Given a sequence of images, frame-to-frame optical flow can be computed as discussed in Chapter 3. The 'Hamburg' taxi sequence is used here to illustrate graph cuts using optical flow information. Here we just perform a binary segmentation of the sequence splitting the image into regions that are moving and regions that are stationary. Optical flow fields are calculated and the labels are initialized based on the motion vectors. To compute the optical flow, dense Lucas-Kanade feature tracking is employed. The translation model is only used to track features from one frame to next.

Translation parameter **d** is estimated for every pixel in the image. The resultant motion vectors are not accurate because of the effects of lack of texture and motion discontinuity edges. Hence the values obtained are used just to initialize the graph cut algorithm. A

Figure 4.6: Optical flow based binary segmentation. Frame 1 of ‘taxi’ sequence is shown.

window size of 15×15 was used and the tracking was done only for those windows for which the determinant value of *T* exceeded 5×10^8. The segmentation output on the taxi sequence with the optical flow initialization is given in Figure 4.6. This figure shows that only the moving taxi is extracted whereas the other two vehicles are not recognized. This is due to the high threshold set for the determinant. High threshold was needed to make the optical flow vectors reliable. This effect causes the optical flow to remain at zero on the regions over the other two vehicles. Also the taxi borders are not crisp and they are smoothed very much due to a large window size. The window size and texture present inside the window plays a crucial role in the calculation of motion vectors.

4.5 Multiway Cuts

In this section, we discuss the important graph cut algorithm that is extensively used throughout this work and that is the multiway cut algorithm. It was proposed in [11] and was shown to produce excellent results for stereo pair of images. This method is a generalization of the binary cuts discussed in the previous section. The multiway cuts approach works by repeatedly performing minimum s-t cuts of binary segmentation. This method has been improved in [20, 19, 18] and made to work for a variety of applications. Unlike the binary segmentation where a single graph cut produces the global minimum of the energy function, multiway cuts are NP-hard and cannot be guaranteed to find the global

minimum. But the approaches, namely alpha-expansion and alpha-beta swap mentioned in this section, can find a good local minimum.

In binary segmentation of images, there are only two labels and we want all the pixels to get either one of the labels. But in multiway cuts, there are multiple labels that the pixels need to acquire and so there has to be clear way to remodel the minimum cut framework to handle more than two labels. For example in stereo, pixels need to be classified based on disparities and in motion on the basis of displacement vectors. This kind of multi-level segmentation can be performed using the minimum s-t cut technique by the divide and conquer methodology. A pair of labels is considered at any given time and a graph is constructed with those two labels of the pair forming the source and sink terminal. Then the minimum cut of the graph is obtained and pixels labeled with either of the labels. This is then repeated for every pair of labels present. Another way of divide and conquer methodology suggests that one of the terminals can be made to correspond to a particular label and the other terminal could just represent the previous existing label of the pixel. These two methods are called the alpha-beta swap and the alpha-expansion correspondingly and discussed subsequently.

4.5.1 Alpha-Expansion and Alpha-Beta Swap

The multiway cut algorithm is efficiently implemented using two methods and they are alpha-expansion and alpha-beta swap. Let P be the set of pixels in the image and L be the set of labels under consideration, then the segmentation of image into different regions is defined as: for all $p \in P$, $L(p) = l_p \in L$. Let us start from an arbitrary initial labeling where all pixels get some random labels from the label set. This initial labeling of the images would have very high energy, computed using the energy functions of Equation 4.1. We seek the labeling of the image that has the least energy. This involves finding the right labels to all the pixels from the initial labeling. The alpha-expansion and the alpha-beta swap can solve these using minimum s-t cuts.

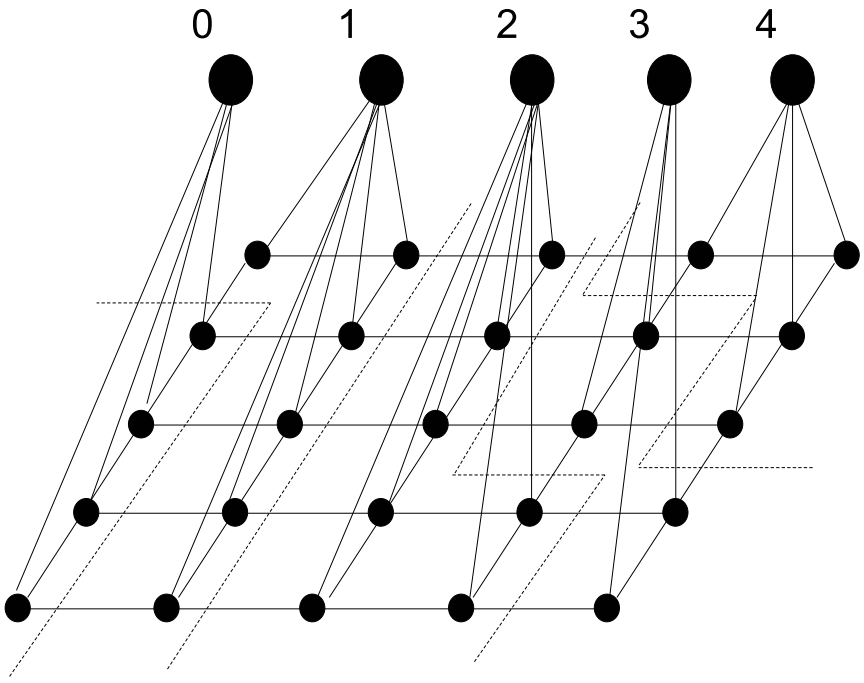


Figure 4.7: Schematic of multiway cuts. Cut terminal links are not shown.

The two algorithms involve move spaces and partitions of graph theory. The alpha-expansion algorithm is defined as follows, given a label α, then the change in labeling of the image is within a single alpha-expansion move if some pixels in the current labeling acquire the new label α, with all other labels of pixels remaining the same. That is P_α(old) $\subset P_\alpha$(new) and P_l(new) $\subset P_l$(old) for any label $l \neq \alpha$. P_α and P_l are sets of pixels with label α and l respectively. The alpha-beta swap algorithm is defined as follows, given two labels α and β then the change in the labeling of the image is within a single alpha-beta swap move if some pixels currently labeled α get the new label of β and some pixels currently labeled β get the new label α. That is $P_{\alpha\beta}$(old) $= P_{\alpha\beta}$(new) and P_α(old) $\subset P_\alpha$(new) and P_β(new) $\subset P_\beta$(old) or P_α(new) $\subset P_\alpha$(old) and P_β(old) $\subset P_\beta$(new). P_α and P_β are sets of pixels with label α and β respectively and $P_{\alpha\beta}$ is the union of P_α and P_β.

Let us denote the graph by $G = (V, E)$, consisting of nodes V and edges E, and assign edge weights based on the energy functionals. The two graph cut algorithms discussed

above can be used to find the least energy function. The segmentation result can be defined by the equation shown below.

$$L(x,y) = L(p) = l^m \tag{4.23}$$

where,

$$l^m \in \{l^1, l^2, ..., l^M\}. \tag{4.24}$$

Thus all the points in the graph gets associated with one of the M labels. At each iteration of the alpha-expansion or alpha-beta swap algorithms, a graph is constructed with source and sink terminals as in binary segmentation. The minimum cut on this graph is obtained and the pixels are labeled accordingly. The final assignment of labels by multiway cut can be demonstrated using Figure 4.7.

4.5.2 Labels and Weight Assignments

The multiway cut algorithm is a segmentation tool that is used to subdivide the image into partitions of uniform labels. These labels have different meanings based on the application. For stereo correspondence, the labels correspond to 1D displacement vectors called disparities, for motion segmentation, they are 2D vectors, and for image restoration they correspond to intensity levels. In this research, the application of multiway cuts for stereo and motion has been studied and hence the labels are disparities and displacement vectors. The multiway cut algorithm's computational time depends on the number of labels that we are working with and hence there is a strong desire to keep the number of labels at a minimum.

The labels for stereo are limited to a finite set of disparities denoted by $0 \leq \delta \leq \Delta$. It is assumed that closest object from the camera is at a distance Z greater than $\frac{f_b}{\Delta}$ where f_b is the baseline width for the stereo pair of cameras. In the case of motion, the labels are limited

link	weight
$\{s,p\}$	$D(l_p)$
$\{p,t\}$	$D(l_\alpha)$
$\{p,q\}$	$S(l_p,\alpha_q)+S(\alpha_p,l_q)-S_p(l_p,l_q)$

Table 4.2: Weight assignments in graph for alpha-expansion.

by $(\Delta_{x_-},\Delta_{y_-}) \leq (dx,dy) \leq (\Delta_{x_+},\Delta_{y_+})$, where (dx,dy) denotes the displacement of any pixel in the image. The Δ_x and Δ_y terms here depends on the maximum frame-to-frame motion of any object which again depends on the sampling rate. Usually the disparities and displacement vectors are assumed to be integral numbers even though in reality they are not. In stereo, if the baseline is large, then the disparities are larger numbers and hence we would have to work with a larger set of labels. If baseline is small, then there would not be a good number of disparities to split the image into distinct regions and this would give poor results. Similarly, for motion, sampling rate high or low determines the amount of labels and the ability to demarcate two differently moving regions.

As discussed in the previous sections, the energy function terms are used for assigning weights to graph edges. The data term for a pixel calculates the data penalty incurred if that pixel acquires a particular label. In the case of stereo, the data term measures the cost defined by Equation 4.4 or Equation 4.5. Each label *l* in the label set of disparities correspond to a particular disparity. Usually labels $(0$ to $l)$ correspond to disparities $(0$ to $-\Delta)$. The data penalty calculated is used to assign the weights for t-links. The smoothness penalty measures the amount of disagreement of labeling of neighboring pixels based on Equation 4.16. This cost assigns weights for n-links. The assignment of edge weights for motion is also similar. For motion, the labels are little complex because the labels capture 2D information. Generally, the labels $(0$ to $l)$ would correspond to $(x0,y0)$,$(x1,y0)$...$(xl,y0)$, $(x0,y1)$,$(x1,y1)$...$(xl,y1)$,..... $(x0,yl)$,$(x1,yl)$...(xl,yl) which is nothing but the combination of all vertical and horizontal displacements. The weight assignments of links, after the graph construction by an iteration in the alpha-expansion algorithm, is shown in Table 4.2.

4.5.3 Translation and Affine Displacements

Generally, multiway cut algorithms have been formulated to handle translation based frame-to-frame displacement for both stereo and motion. In [6], the multiway cuts were used with the affine based frame-to-frame transformation and excellent results have been obtained for both motion and stereo. This paper addresses the fundamental problem with the original multiway cut formulations that assume piecewise constancy of disparities instead of piecewise continuity. This leads to poor segmentation of images with non fronto-parallel surfaces. If instead of piecewise constant disparities assumption, affine parameters are fit into the regions then we can retrieve the smooth continuous variation of disparities on slanted surfaces. This paper forms the basis for all the segmentation algorithms discussed in this thesis. This algorithm would be referred to as the parent algorithm in this thesis. The affine based displacement function, for a label, can be defined using the equation,

$$l_p(p) = A_l p + \mathbf{d}_l. \tag{4.25}$$

In this equation, A_l and $\mathbf{d}_l$ define the affine displacement parameters for label l. l_p is the label of pixel p and $l_p(p)$ is the warped location of p. Figure 4.8 shows the flowchart for implementing multiway cuts with affine based displacement functions. The first step is similar to regular multiway cuts, where a set of labels corresponding to translational displacement functions are proposed. Using these labels, multiway cuts is performed. The resulting label image's connected components is computed and after removing small regions, affine parameters are fitted to the regions and the displacement functions are refined. If energy of the new configuration is less, then the process is repeated from the multiway cut step. If not, a final oversegmentation step is carried out to combine missed neighboring regions. This step pulls the result from getting caught in local minimum.

Here we analyze the differences between multiway cut results using translation based disparities and affine based disparities for stereo pair of images. The Figure 4.9 shows the

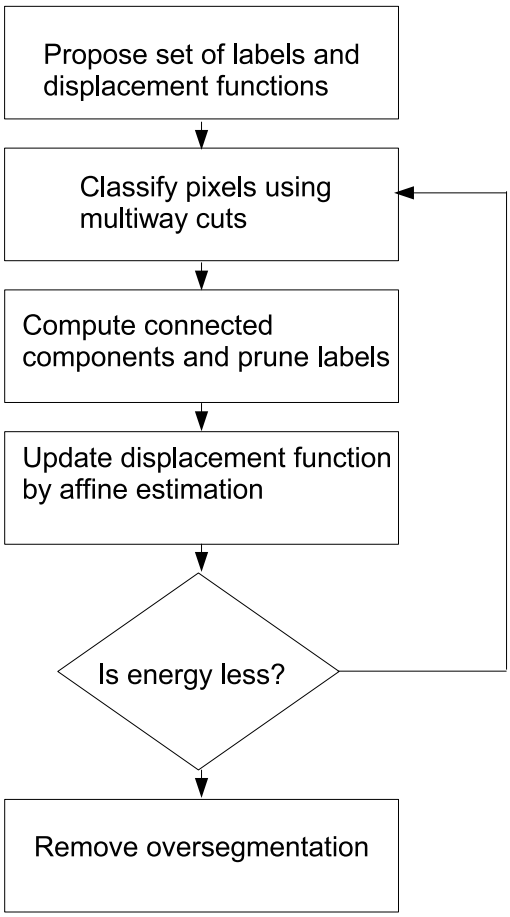


Figure 4.8: Flowchart for affine-based multiway cuts.

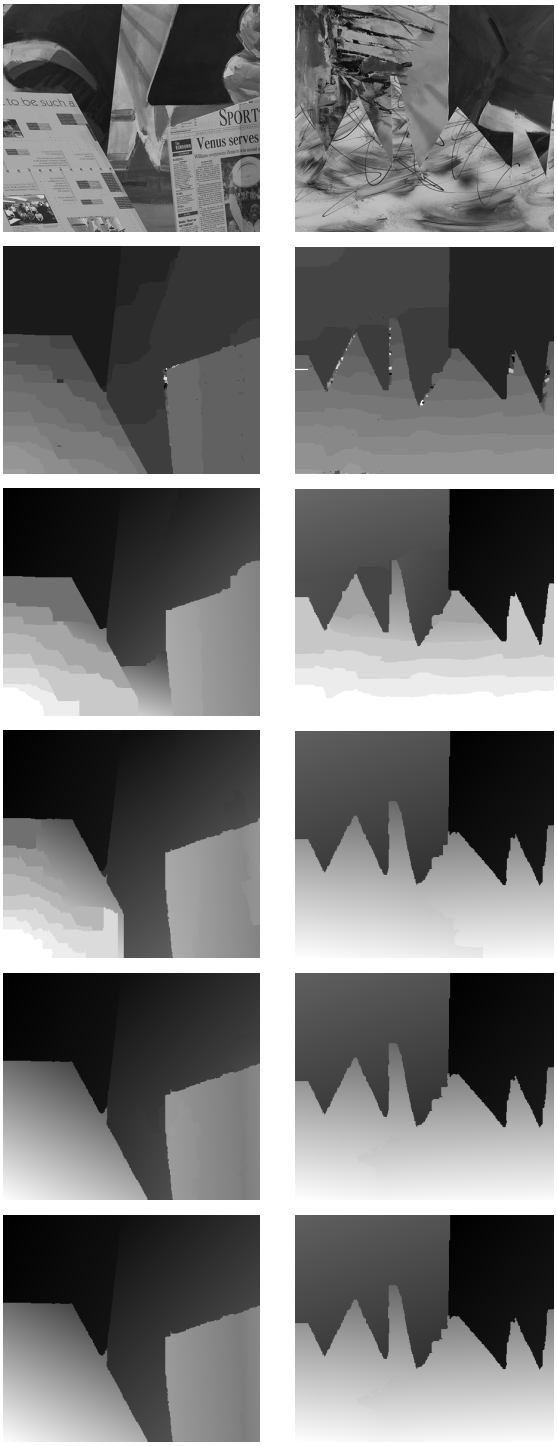


Figure 4.9: Left: ‘venus’ stereo pair’s left image, its translation disparity map and iterations of parent algorithm; right: ‘sawtooth’ stereo pair’s left image, its translation disparity map and iterations of parent algorithm.

marked differences between the two algorithms. There are totally four layers in the original image of the ‘venus’ stereo pair. The disparity map due to multiway cuts with translational disparities shows the discrete depths on some of the layers. The affine based multiway cut iterations shows the smooth disparity maps found. Similar results with the ‘sawtooth’ stereo pair is also demonstrated. In this pair, there are totally three layers and the affine based multiway cut shows its ability to retrieve all the three different layers.

Many real stereo pairs have slanted surfaces and it would be desirable to retrieve the entire surface as a single layer with continuously changing values of disparities rather than bits of regions having constant disparities. Also, the disparities of split regions on the slanted surfaces and how these regions are divided depend heavily on the numerical values of smoothness costs as they determine how regions get smoothed out in the absence of high local intensity gradients. The parent algorithm on the other hand can retrieve slanted surfaces as a single smooth region. The only disadvantage with the parent algorithm is that it works in an iterative style and hence the computational time is higher. But nevertheless, it produces reliable and reasonably good results when compared with other state of the art methods.

Chapter 5

Algorithms and Results

This chapter details all the algorithms that were experimented in this thesis. Initially, the main algorithm of this work is presented. Here, the multiway cut formulation using affine transformation is used for motion segmentation of a sequence of images. Existing techniques that use graph cuts for motion segmentation handles the problem as a two-frame correspondence. Other research work that uses graph cuts [31] for motion extraction in image sequences does not employ graph cuts to handle majority of the segmentation. Also, only the maximum flow binary cut is used. In the algorithm that is presented here, multiway cuts handle all the segmentation work from frame to frame. The next algorithm is for stereo correspondence where we extend the current algorithm to handle correspondence of important small regions in the scene that gets gobbled by neighboring big regions. Subsequently, we present the work done on detecting occlusions interactively by computation of affine residuals for border regions of two neighboring motion segments.

The results for all the algorithms on real sequences and images are presented. For the main algorithm, that works on image sequences, we have experimented with three simple motion sequences, namely the ‘pepsi’, ‘Hamburg’ and ‘flower garden’ sequences. For the stereo case, the ‘parking meter’ and ‘tsukuba’ pair of images are used for testing. The occlusion detection technique uses images from the motion sequences mentioned above.

All the algorithms that are mentioned here is implemented from ground up using C++. The only external code that is used is the maximum flow graph cuts library of [17]. For image manipulation tasks, ‘Blepo’ computer vision library [1] was used.

5.1 Motion Segmentation of Image Sequences

In [6], the basic assumption of piecewise constant disparity for stereo correspondence, in multiway graph cuts of [11], was replaced with the assumption of piecewise continuity. Stereo pair of images and two-frame motion images with slanted surfaces cannot be properly represented using discrete levels of disparity as for slanted surfaces, the disparity varies smoothly across the surface. This was solved using the piecewise continuity assumption for disparities by estimating affine parameters for every connected region in the output of the discrete disparity maps. These parameters are then used to refine the displacement functions and the multiway cuts algorithm is executed again with affine displacement functions. The process is repeated to get cleaner disparity maps. The same problem was applied to motion images with slanted surfaces. Affine transformation can represent a huge class of object motion and the algorithm produced good results on motion images. But the algorithm for motion images was implemented only as a two-frame correspondence. There has been no work in the literature that extended this technique to perform motion segmentation over a sequence of images.

Here, we perform motion segmentation over a sequence of images using the technique mentioned above. The motivation to do this is that the existing algorithm is a clean framework that combines the power of recent graph based techniques and the abilities of the affine motion model. Also, motion segmentation of image sequences is a problem where there is dependence between the segmentation results of consecutive frames. This inter-frame dependence is exploited in the algorithm that is explained here.

1. Initialize all pixels in the label image to zeros. All labels here represent translational displacement functions.

2. Perform multiway cuts. This gives segmentation (label) image.

3. Compute connected components of the label image. Merge small regions (percentage area $<$ threshold a_{th}) with neighboring large regions.

4. Estimate affine parameters for all connected regions. If affine estimation is successful, update displacement function of the label with new affine parameters.

5. Run multiway cuts with pruned labels and updated displacement functions. Compute energy. If energy has reduced, repeat steps 2 to 5.

6. Remove oversegmentation by merging neighboring regions.

These steps come from the parent algorithm [6]. For segmentation over image sequences, these steps are to be done for the first frame and the number of motion layers obtained. After that the following steps are carried out.

7. Obtain label image of the previous frame and warp every region based on the affine parameters represented by their labels.

8. If apriori knowledge of foreground-background relationship is known, warp foreground regions first, followed by background regions.

9. Again estimate affine parameters for the warped label image and update displacement functions. This gives the estimate of motion segments for the current frame.

10. Perform steps 2 to 6 for the current frame and get the label image.

11. If number of motion segments in the label image is known to be constant, execute parameterized affine merging, explained below, to get the merged label image.

12. Repeat steps 7 to 11 from frame to frame till the end of the sequence.

The parameterized affine merging step is explained here. The goal of this step is given a label image; it employs an iterative parameterized affine based merging to reduce the number of regions in the label image to an expected value. The expected value is the number of motion segments we wish to keep constant throughout a sequence. The technique is explained below.

1. Compare the label image, at the current frame, with its estimated label image computed by the previous frame for number of motion segments.

2. Establish correspondence between regions in the estimated label image and regions in the actual label image based on a voting scheme.

3. For every major region, if affine parameters of its neighbors are within a threshold η_i, merge the neighbor if affine parameter estimation for the combined region is successful.

4. If number of segments greater than expected, repeat step 3 after automatically adjusting threshold η_i.

The automatic adjustment of affine parameter thresholds proceeds as follows. Initially, the η_i is stringent, with values kept at minimum, and as iteration progresses the η_i is made lenient by multiplying the values by a constant factor of 1.5. The parameterized affine merging step is similar to the oversegmentation of the parent algorithm but differs in the way that the later works by measuring energy costs after every iteration whereas the former is based on thresholds, which are parameters. The parameters are empirical and have to be changed from one sequence to another. The voting scheme that is mentioned in step 2, compares the area of overlap between a given region in the estimated label image and all regions in the actual label image and picks the maximum one.

In step 8 of the main algorithm, it was mentioned that warping order can be adjusted based on the apriori knowledge of foreground-background relationships. This is because warping a pixel location based on affine parameters gives a floating point location. All the four integral neighbors of the floating point location get the label of the pixel that got warped. Therefore if foregrounds are warped first followed by background, then near the border regions of foreground and background, it is likely that background pixels will get proper labels than foreground. This enforces multiway cuts to make good decisions at the border. This is because the multiway cut algorithm usually retrieves foregrounds clearly and makes mistake on background pixels. The order of warping mentioned here is just necessary to improve the results at border regions between neighbors and is not essential for multiway cuts.

The parameterized affine merging step can be executed only for simple sequences and when the number of motion layers is known. For cases when new objects enter the scene, the number of regions has to be manually changed for the technique to work. But nevertheless it achieves some degree of automation for a sequence of images. The estimated label image serves two purposes. One is that it initializes the label image for the next frame so that multiway cuts for the next frame has a good starting point and the other one is the speedup achieved by avoiding exhaustive search of many translation displacement functions. The motion segmentation that is performed here can be compared to tracking of multiple regions through the sequence. The number of multiway cut loops for every frame is set to a constant, to avoid over smoothing. This is a variant of the parent algorithm.

5.1.1 Results for ‘Pepsi Can’ Sequence

In this section, we give the results and analysis of our algorithm for the pepsi can sequence. In this sequence, a pepsi can placed on table is shot using a slowly moving camera. The camera motion is from right to left and the sequence contains 11 frames. Our algorithm was run on this sequence and the goal was to retrieve the foreground pepsi can and the

Parameters	Value
D_p	$(I^p - I^q)^2$
I_{th}	5.0
λ_1	20
λ_2	10
r_{th}	2.0
a_{th}	0.75
N_{itr}	2
$\eta_i(A)$	0.06
$\eta_i(d)$	0.6
Δ_{x_-}	-4
Δ_{x_+}	0
Δ_{y_-}	0
Δ_{y_+}	0

Table 5.1: Parameters for 'pepsi can' sequence.

background wall as two separate motion segments. Since the discontinuity of motion vectors between the can and the wall is marked, the segmentation is good on top. Since there is no real motion discontinuity between the can and the table on which it is placed, the table gets labeled as foreground as the motion of table and the can are almost similar from the camera's point of view.

The parameters used for the sequence are given in Table 5.1. Most of the parameters mentioned here were discussed in Chapter 4. They are briefly mentioned here. D_p is the data penalty. λ_1, λ_2 and I_{th} are used to compute the smoothness penalty. r_{th} is the residue threshold for affine parameter computation. a_{th} is the percentage of area of the image, that gets labeled as small region. N_{itr} defines the number of iterations of the parent algorithm. $\eta_i(A)$ is the starting threshold for affine parameters of the A matrix and $\eta_i(d)$ is the starting threshold for affine parameters of the d vector. Δ_{x_-}, Δ_{x_+}, Δ_{y_-} and Δ_{y_+} define the initial translational displacement functions for the labels.

The progression of segmentation for a particular frame is shown in figure 5.2. The number of iterations was two and the segmentation results show that in the first frame while most of the pepsi can was recovered there were errors in the top. The second iteration has corrected this and the entire pepsi can is recovered. This was a simple sequence and the

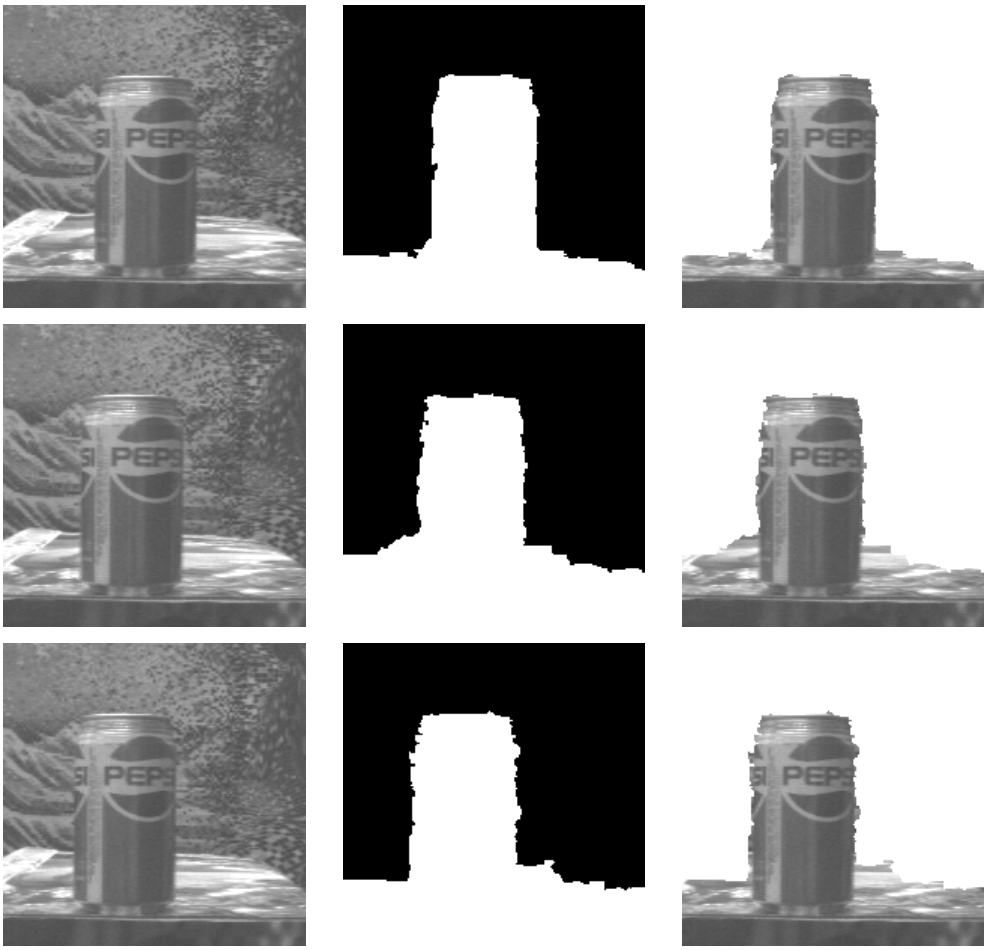


Figure 5.1: Segmentation results for ‘pepsi’ sequence. Frames 0, 6, and 9 are shown. Left: original image, center: segmented image, and right: foreground region.

Figure 5.2: Parent algorithm's progression for frame1. Iterations 1 and 2 are shown.

parameterized affine merging routine was not called since the number of segments stayed constant, at two, for all frames. The results are shown in Figure 5.1. The computational time for running the parent algorithm on every pair of frames for the entire sequence is 30.2 seconds. This algorithm took only 15.1 seconds on the same 10 frame-pairs. Hence a speed up of about 2 was achieved. A Pentium 4 machine with 2.80Ghz was used to run the algorithms.

5.1.2 Results for 'Flower Garden' Sequence

In this section, we present the output of our algorithm for the flower garden sequence. In this sequence, the camera captures a flower garden with a tree in the center. Also, the flower garden gradually slopes toward the horizon showing the sky and far objects. Semantically, this sequence has four layers. They are the tree, flower garden, house and sky. This sequence contains 29 frames. Here since the frame-to-frame motion of the sequence is small, every frame of the sequence is compared not with the next frame but one frame after. This enables the multiway cut algorithm to distinguish between motion segments. The parameters used for this sequence is given in Table 5.2.

Here, we compare the results of this algorithm on frame 1 and frame 2 with the parent algorithm, shown in Figure 5.3. The result for frame 1 is same for both the algorithms. But when the parent algorithm is run on the second frame with the same parameters, it is not successful in merging the two segments on the tree. For some frames there is over smooth-

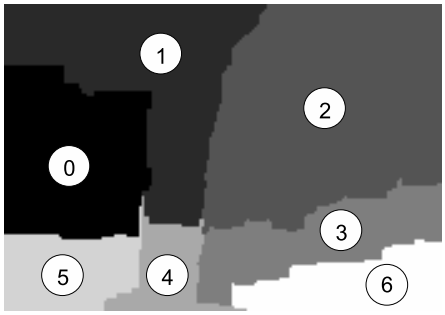
Parameters	Value
D_p	$(I^p - I^q)^2$
I_{th}	5.0
λ_1	30
λ_2	15
r_{th}	2.5
a_{th}	0.75
N_{itr}	2
$\eta_i(A)$	0.06
$\eta_i(d)$	0.6
Δ_{x_-}	-8
Δ_{x_+}	0
Δ_{y_-}	-1
Δ_{y_+}	1

Table 5.2: Parameters for 'flower garden' sequence.

Figure 5.3: Left: frame 1 segmentation result, common to both algorithms, center: frame 2 segmentation result of parent algorithm, and right: frame 2 segmentation result of this algorithm.

ing and two distinct motion segments, such as parts of the tree and the garden are combined. Hence running the parent algorithm independently for every pair of frames will not produce consistent results. This algorithm maintains the number of motion segments found on the first frame throughout the sequence. The number of motion segments maintained is 5.

The working of parameterized affine merging is shown for frame 18. Figure 5.4 shows that at the end of the multiway cuts, 7 segments are given. Two segments on the tree and two segments on the right side garden are not properly merged. The parameterized affine merging step fixes this problem by merging the two segments on the tree in the first iteration shown in Figure 5.5 and merging the two segments of garden in the second iteration shown in Figure 5.6. This merging step is highly dependent on the threshold, η_i. But the algorithm requires the threshold to be set only once for the entire sequence. The estimated affine



Label	$A(0,0)$	$A(0,1)$	$A(1,0)$	$A(1,1)$	$d(0)$	$d(1)$
0	$+0.98767$	-0.00182	-0.00104	$+1.00185$	-0.8797	$+0.1946$
1	$+0.97759$	-0.01166	$+0.00966$	$+0.99587$	-4.4247	$+0.9700$
2	$+0.99686$	$+0.00111$	$+0.00141$	$+1.00396$	-1.0762	$+0.1923$
3	$+0.99196$	-0.03646	-0.00004	$+0.99783$	-2.0760	$+0.2661$
4	$+0.94732$	-0.01755	-0.01170	$+0.98243$	-4.9808	$+0.3899$
5	$+0.99598$	-0.02896	-0.00053	$+0.99962$	-2.0109	$+0.3109$
6	$+0.99193$	-0.04205	-0.00302	$+0.99026$	-2.9296	$+0.2025$

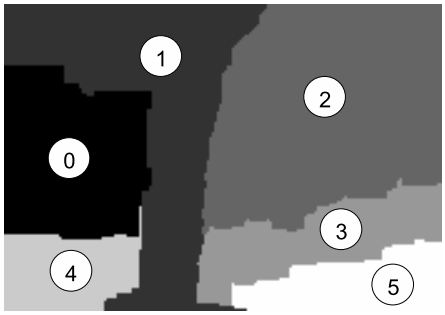
Figure 5.4: Parameterized affine merging; initial segmentation for frame 18.

parameters at each step of the algorithm are given under each figure. The label numbers are also given. The time taken for the parent algorithm is 97.1 seconds for 27 frames. This algorithm executed in 71.2 seconds.

The results of the algorithm for frames 2, 10, 19, and 25 are shown in Figure 5.7. The original images and the obtained segments are shown. The tree and the house regions of the sequence is separately shown in Figure 5.8. The results show that the segmentation is good, except for small sky regions getting associated with the tree and the far end of the garden getting the same label as the house.

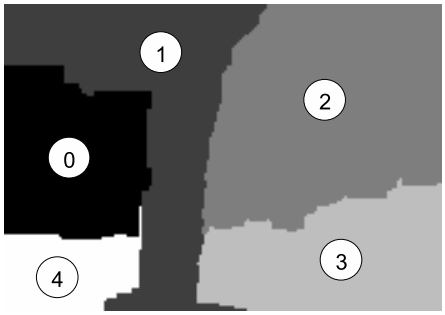
5.1.3 Results for 'Hamburg Taxi' Sequence

Here, we employ our algorithm on the taxi sequence. This sequence is recorded by a fixed camera looking on a road from the top. There are 41 frames in this sequence and 3 moving vehicles. The taxi at the center moves at around 1 pixel/frame to the left and about 0.5 pixels/frame to the top. The vehicle on the left moves at approximately 3 pixels/frame to the right. The vehicle on the right also moves at around 3 pixels/frame but to the left. This



Label	$A(0,0)$	$A(0,1)$	$A(1,0)$	$A(1,1)$	$d(0)$	$d(1)$
0	$+0.98767$	-0.00182	-0.00104	$+1.00185$	-0.8797	$+0.1946$
1	$+0.97664$	-0.00910	$+0.01037$	$+0.99479$	-4.5544	$+0.9130$
2	$+0.99686$	$+0.00111$	$+0.00141$	$+1.00396$	-1.0762	$+0.1923$
3	$+0.99196$	-0.03646	-0.00004	$+0.99783$	-2.0760	$+0.2661$
4	$+0.99598$	-0.02896	-0.00053	$+0.99962$	-2.0109	$+0.3109$
5	$+0.99193$	-0.04205	-0.00302	$+0.99026$	-2.9296	$+0.2025$

Figure 5.5: Parameterized affine merging; intermediate segmentation for frame 18.



Label	$A(0,0)$	$A(0,1)$	$A(1,0)$	$A(1,1)$	$d(0)$	$d(1)$
0	$+0.98767$	-0.00182	-0.00104	$+1.00185$	-0.8797	$+0.1946$
1	$+0.97664$	-0.00910	$+0.01037$	$+0.99479$	-4.5544	$+0.9130$
2	$+0.99686$	$+0.00111$	$+0.00141$	$+1.00396$	-1.0762	$+0.1923$
3	$+0.99039$	-0.04217	-0.00083	$+0.99647$	-2.3767	$+0.2482$
4	$+0.99598$	-0.02896	-0.00053	$+0.99962$	-2.0109	$+0.3109$

Figure 5.6: Parameterized affine merging; final segmentation for frame 18.

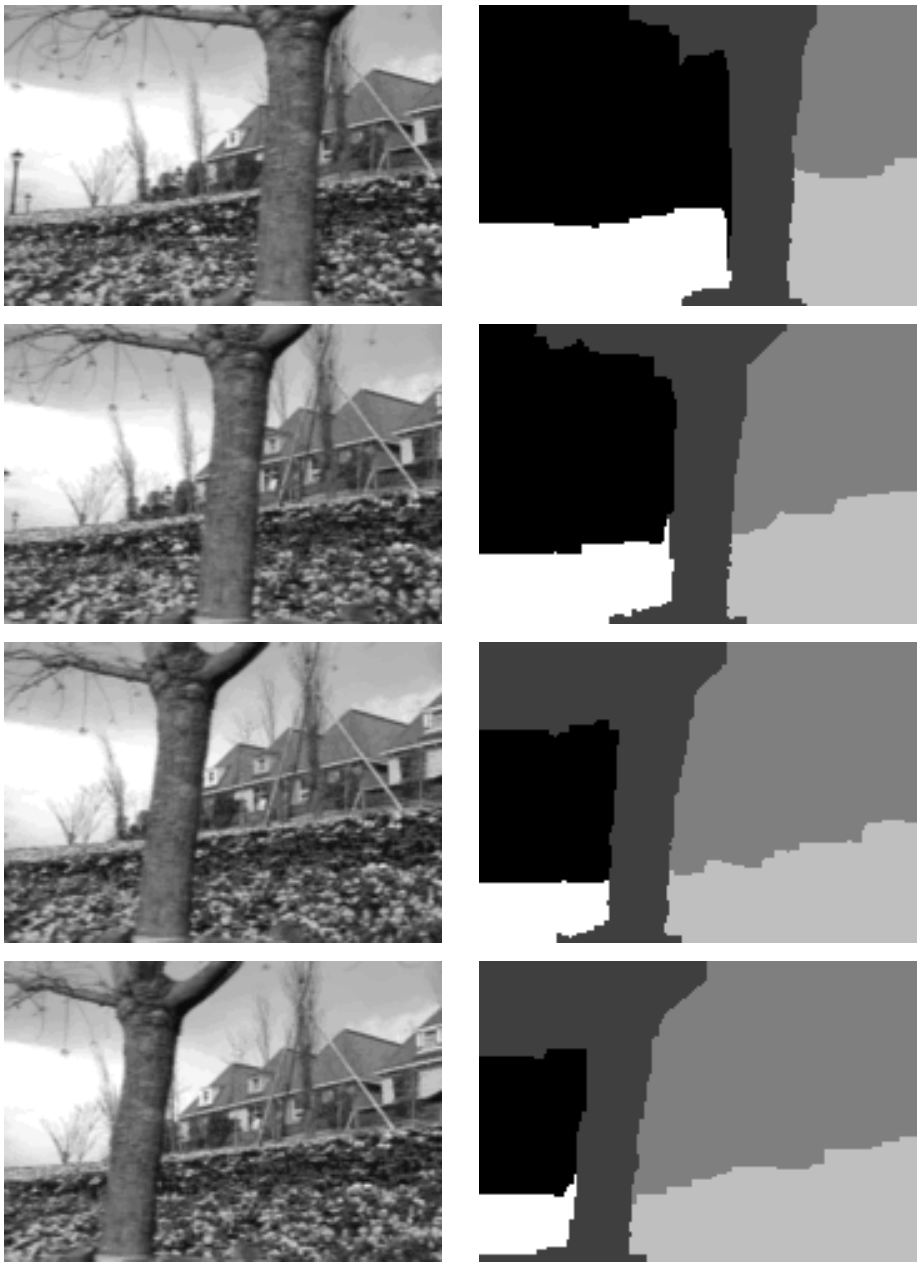


Figure 5.7: Segmentation results for ‘flower garden’ sequence. Frames 2,10,19, and 25 are shown. Left: original image, and right: segmented image.

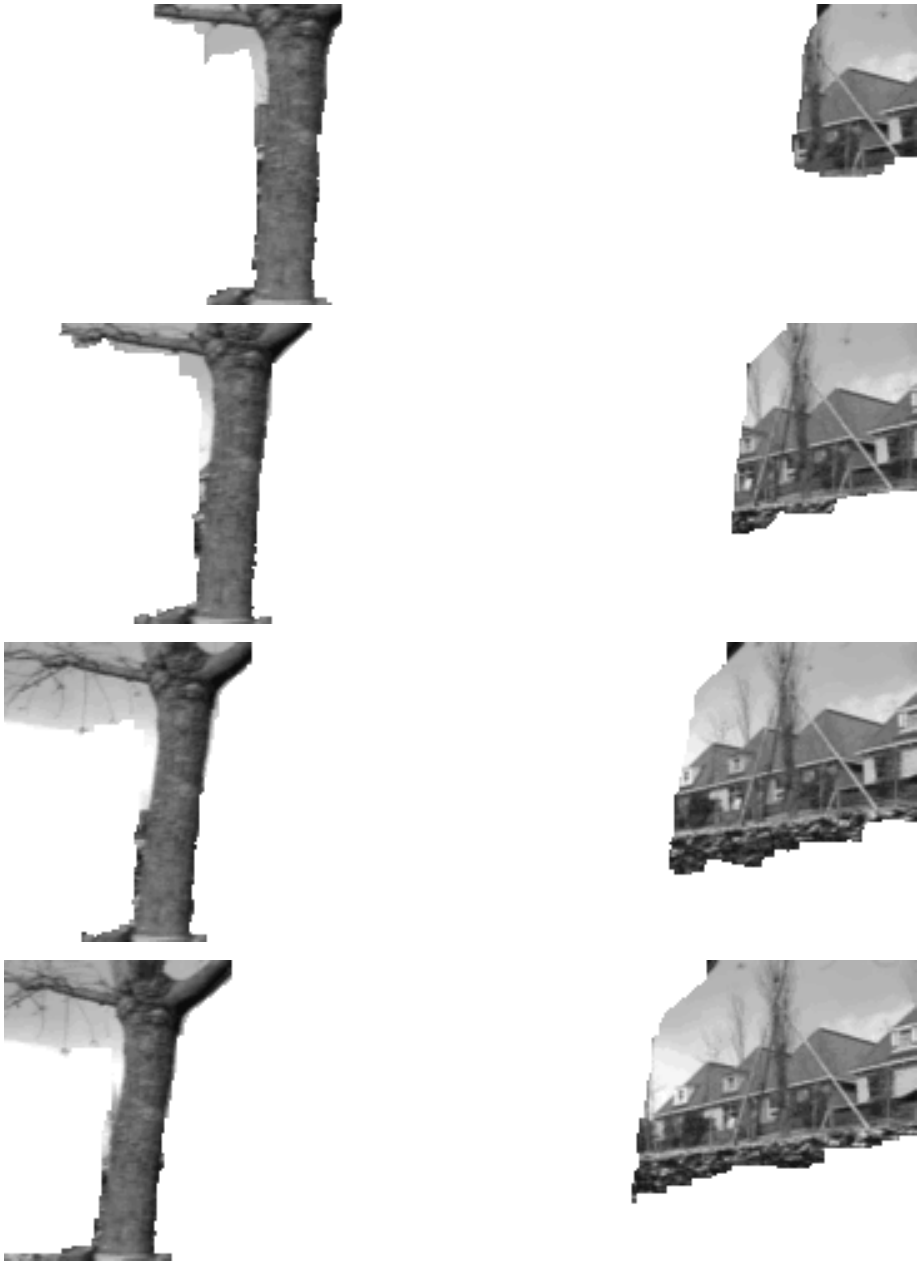


Figure 5.8: ‘Flower garden’ sequence; ‘tree’ and ‘house’ regions for frames 2,10,19, and 25 are shown.

Parameters	Value
D_p	$(I^p - I^q)^2$
I_{th}	5.0
λ_1	100
λ_2	40
r_{th}	2.5
a_{th}	0.5
N_{itr}	2
$\eta_i(A)$	0.06
$\eta_i(d)$	0.6
Δ_{x_-}	-3
Δ_{x_+}	3
Δ_{y_-}	-3
Δ_{y_+}	3

Table 5.3: Parameters for ‘taxi’ sequence.

sequence has poor lighting conditions. The left and the right vehicles are almost of the color of the road. Also, the right vehicle gets occluded by a tree in most of the frames. The parameters for this sequence is given in Table 5.3.

For this sequence, the number of segments was manually fixed to be at 4. This is to accommodate for the right vehicle that enters the scene in the 3^{rd} frame. The affine merging step works well for the sequence from frame 1 to frame 36. After that it fails. The failure is shown for frame 40 in Figure 5.9. This is because two disconnected regions are obtained for the right vehicle due to occlusion. The affine merging step merges the taxi with the background before it merges one of the segments on the right vehicle with the background. This is because the motion of taxi becomes very small near the end of the sequence and affine estimation produces closer values for the taxi and the background.

The results of the algorithm on frames 1, 5, 18, 22, and 36 are shown in Figure 5.10. This shows the original images and the obtained segmentation images. The regions of taxi, left vehicle and right vehicle are separately shown in Figure 5.11. The algorithm works reasonably for most of the frames. The segmentation on the right vehicle is incomplete in frames 22 and 36 due to occlusion by the tree and the resemblance of its texture with the background. Also the taxi segment of frame 36 shows that most of the background

Figure 5.9: Affine merge failure for frame 40.

gets joined with it due to little motion of the taxi and the estimated label image from the previous frame provides a bad starting point for the multiway cuts. The running time of this sequence with the parent algorithm is 366.5 seconds and with this algorithm it is 195.8 seconds.

5.2 Hard Constraint Points for Stereo

The human eye and brain is able to perceive depth information from a scene because of its stereo vision. The left and the right eye independently record the scene and the brain compares these two images in order to retrieve the depth map. Stereo pair of cameras is similar to the human stereo vision in the aspect that it also captures two different versions of the scene using a left and a right camera at the same instant of time. Once we have the left and the right image, we are faced with the problem of correspondence of pixels in these images that came from the same world point. If this correspondence is solved, then we have the solution for the depth information in the scene. Stereo correspondence is one of the many problems in computer vision that is intensely researched. The problem is hard due to various reasons such as occluded pixels, noise in the image, non Lambertian surfaces in the scene, and untextured regions.

In the case of rectified stereo, any pixel in the left image can have a matching pixel in the right image only along the scanline at the same vertical height. This is known as the epipolar constraint. If assuming fb is the baseline width of the stereo pair of cameras and Z

Figure 5.10: Segmentation results for ‘taxi’ sequence. Frames 1,5,18,22, and 36 are shown. Left: original image, and right: segmented image.

Figure 5.11: ‘Taxi’ sequence; ‘left vehicle’, ‘taxi’, and ‘right vehicle’ regions for frames 1,5,18,22, and 36 are shown.

the distance of any world point from the cameras, then the displacement difference between the point's projection on the left image and the point's projection on the right image is given by $\frac{fb}{Z}$. This is the disparity and we wish to compute disparity at every pixel in the pair of images.

One serious disadvantage with the multiway cuts with affine based displacement functions for stereo, discussed in Section 4.5.3, is that it over-smoothes small regions and thin long regions. Basically, this formulation works very well for large roughly round regions and almost completely eliminates small or thin long regions. This is because the connected components stage after the initial multiway cuts throws away small regions and even if some small or thin true regions manage to appear, the affine step over fits surrounding regions and the small region gets lost. Hence this is a problem with the energy functional that needs to be corrected using some other methods. One way to avoid this problem is to feed hard constraint points, points in the image where disparities can be computed accurately, to the multiway cuts. Finding hard constraint points has to be simple and fast and hence the obvious choice is to use normalized correlation. The correlation data curve, which gives the residue for all disparities at any pixel, can be analyzed and if some disparity value has a clear minimum without any ambiguity, then that pixel can be chosen as a hard constraint point. If all such pixels and their corresponding disparities are used to initialize and constrain multiway cuts, then small and thin regions can be segmented.

The expression for normalized correlation value at a pixel p, given a disparity δ, is shown below.

$$C(\delta, p) = \frac{1}{N} \sum \sum_{W} [I_L(p) - I_R(\delta(p))]^2 \tag{5.1}$$

where,

$$p = (x, y), \text{ and } \delta(p) = (x + \delta, y). \tag{5.2}$$

In the Equation 5.1, *W* is the correlation window, usually of size 5×5. *N* is the number of pixels in the window. The chosen disparity value, δ_p^{min}, for the pixel *p* is the one that gives the least correlation error. This can be expressed as

$$\delta_p^{min} = \arg \min_{0 \leq \delta \leq \Delta} C(\delta, p). \tag{5.3}$$

The necessary condition for a pixel to be labeled as a hard constraint point is given below.

$$p \in P_{hard} \text{ iff } C(\delta_p^{min}, p) < \gamma C(\delta, p) \quad \forall\, \delta \neq \delta_p^{min} \tag{5.4}$$

Here, γ is a constant set at 0.4. P_{hard} is the set of all hard constraint points or pixels. The normalized correlation based choice of hard constraint points suffer from its own difficulties because usually if the threshold, γ, used for selection of hard constraint points is made small then only very few pixels would be obtained and it is less likely that these pixels would lie on small and ambiguous regions. If the threshold is increased, then correlation would render many false positive hard constraint points which could prove detrimental to the operation of multiway cuts. Hence optimal threshold is needed and an empirical value of 0.4 is chosen.

The result of feeding hard constraint points to the algorithm as starting points is discussed. The number of iterations of the parent algorithm was constrained and set at 3. This is because the algorithm, if allowed to run normally, will eventually smooth out small regions. This is because the cost functional of the multiway cuts involve affine fitting and as iteration progresses, the affine updating step slowly over fits all small regions and only preserves the prominent structures in the scene. The results for the 'parking meter' stereo pair is shown in Figure 5.12. The results due to hard constraint points show that there are more details in the image. Here the disparity difference near the first parking meter is recovered.

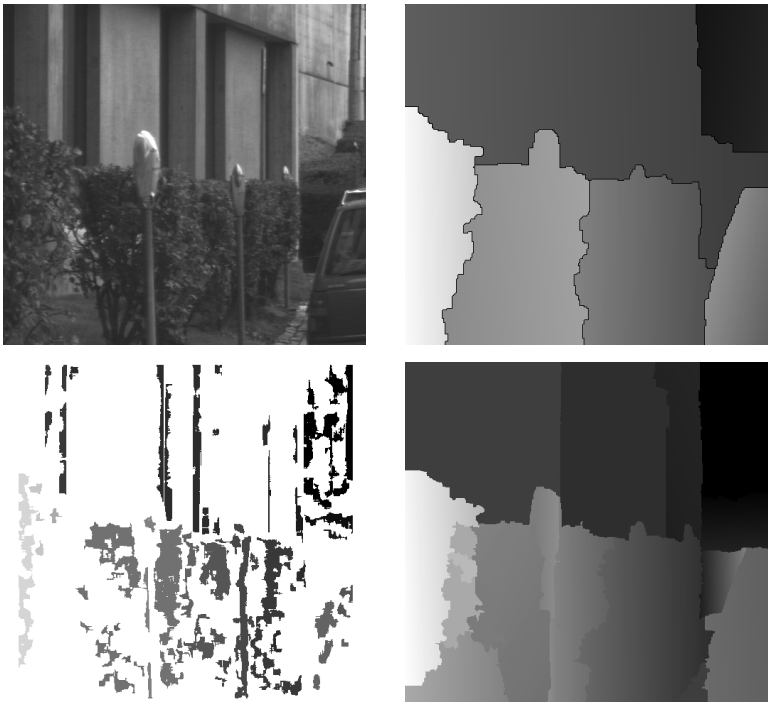


Figure 5.12: ‘Parking meter’ results; left-top: left image, right-top: parent algorithm’s result, left-bottom: hard constraint points, right-bottom: results due to hard constraint points.

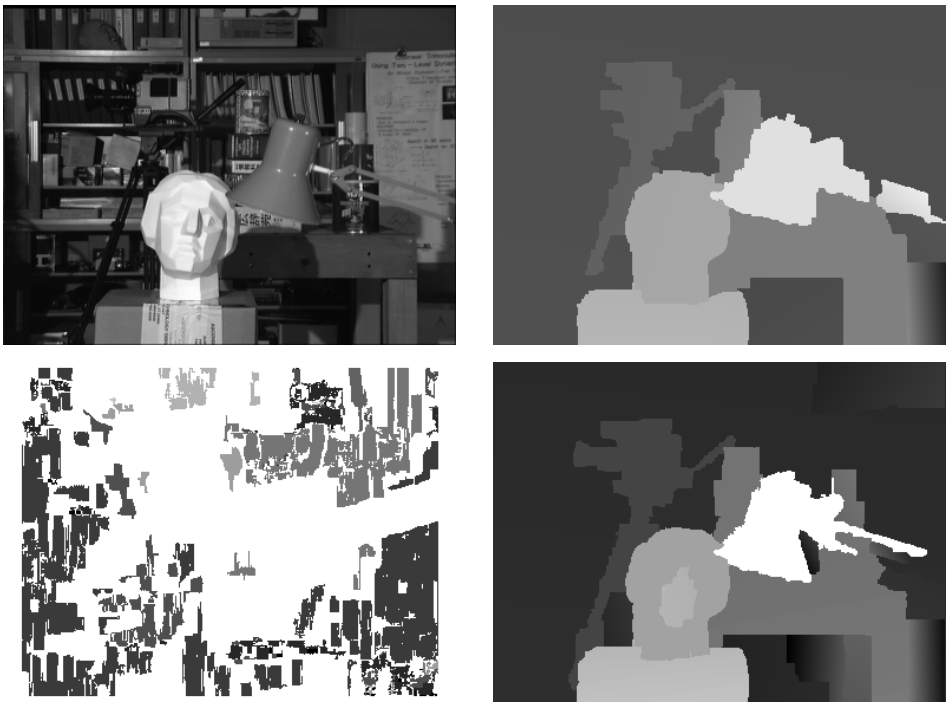


Figure 5.13: ‘Tsukuba’ results; left-top: left image, right-top: parent algorithm’s result, left-bottom: hard constraint points, right-bottom: results due to hard constraint points.

Figure 5.13 shows the results for the 'tsukuba' stereo pair. There are errors near the table but the tripod camera is fully recovered. Also, the pile of cans behind the lamp is retrieved.

5.3 Occlusion Detection

Occlusions are regions in the image, between relatively moving objects, that 'disappear' from frame-to-frame due to motion of the foreground object over the background. Similarly, disocclusion can be defined as those regions that 'appear' from frame-to-frame due to motion of objects. When foreground moves 'away', some areas of background becomes visible and this area is called disocclusion. Occlusions and disocclusions need to be handled to get crisp segmentation along borders of objects. Since occlusion and disocclusion areas are always sandwiched between two true layers, occlusions can be predicted based on the displacement parameters of the two neighboring motion layers.

Occluded regions are selectively detected here in the method that is proposed. The occluded region between two motion layers is predicted based on the affine parameters of these regions. This is explained using Figure 5.14. Region 1 and region 2 are two neighboring motion layers with affine parameters $\{A_1, \mathbf{d_1}\}$ and $\{A_2, \mathbf{d_2}\}$ respectively. Using interactive methods, when region 1 and region 2 are clicked, two regions a. and b. as shown are computed. Region a is inside region 1 and region b is inside region 2. The equations concerning the computation of regions a and b are discussed. Let P^1 be the set of pixels in region 1 and P^2 be the set of pixels in region 2. P^1_{occ} is set of all $p^1 \in P^1$ that satisfies the equation shown below.

$$A_2^{-1}(A_1 p^1 + \overline{d}_1 - \overline{d}_2) \in P^2 \tag{5.5}$$

The meaning of this equation is explained here. When a pixel p^1 in region 1 is warped by its affine parameters, we get a location. This location is inverse warped by region 2's affine parameters to get another pixel location. Now if this location is in region 2, then p^1

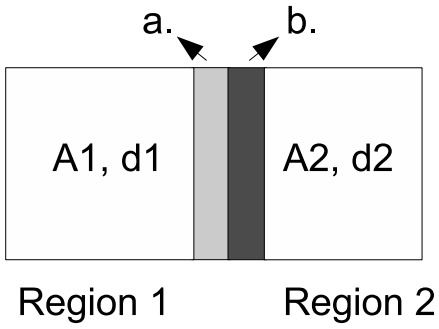


Figure 5.14: Occlusion prediction by affine warpings.

belongs to region a. All such p^1's form P^1_{occ}, which is region a. Similarly, P^2_{occ} is region b and it is the set of all pixels $p^2 \in P^2$ that satisfies the equation

$$A_1^{-1}(A_2 p^2 + \overline{d}_2 - \overline{d}_1) \in P^1. \tag{5.6}$$

Usually multiway cuts label occluded pixels with the foreground's label. Assuming all occluded regions got the foreground label, meaning of region a and region b can be explained. Intuitively, region a. is the predicted occlusion region assuming region 1 is the foreground and Region b. is the predicted occlusion region assuming region 2 is the foreground. But only one of these two regions is actually occluded and this is determined by calculating the affine residue for both of these regions. Region a and region b are warped by both region 1 and region 2 affine parameters and the residues for the warped regions are determined. Here we get 4 residues, due to two regions and two affine parameters. The maximum residue value is chosen. This residue's corresponding region is the occluded region and the corresponding affine parameters belong to the background. This is based on the fact that occluded region in one frame does not have a matching region in the next frame and hence their affine residual is bound to be higher.

The results due to this technique are demonstrated. The results for the 'pepsi' can sequence is shown in Figure 5.15. Occlusions are computed between pepsi can and the background. The results show that most of the occlusion pixels, on the occluding direc-

Figure 5.15: ‘Pepsi’ occlusion results; left-to-right: original image, segmented image, predicted occlusion between foreground pepsi can and background, occlusion overlaid on original image.

Figure 5.16: ‘Flower garden’ occlusion results; left-to-right: original image, segmented image, predicted occlusion between tree and left sky, occlusion overlaid on original image.

tion, are obtained. Figure 5.16 shows the results for the ‘flower garden’ sequence where selective occlusion detection between the tree segment and the left sky segment was done. Since this technique assumes segmentation by multiway cuts places just the occluded pixels with the foreground, the prediction is not accurate. Nevertheless, it finds most of the occluded regions to the left of the tree and just under the tree branch. The results for the taxi sequence is given in Figure 5.17. Occlusions are predicted between the taxi and the background. Most of the occlusions are predicted as taxi moves in the left-top direction. There are some errors under the taxi. This is due to inexact representation of taxi motion by affine parameters. The occlusion prediction between the left vehicle and the road is almost accurate as most of the pixels on the road that got labeled as foreground is retrieved.

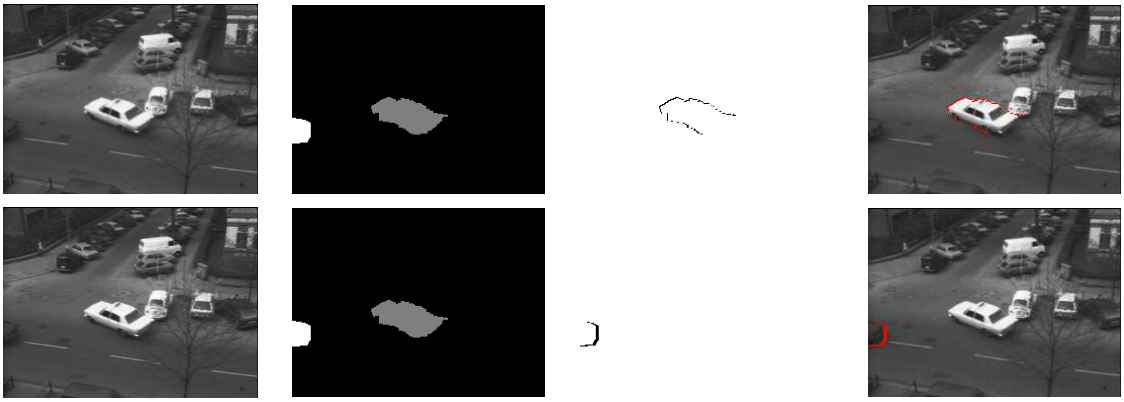


Figure 5.17: ‘Taxi’ occlusion results; left-to-right: original image, segmented image, predicted occlusions, occlusion overlaid on original image; top: occlusion between taxi and background, bottom: occlusion between left vehicle and background.

Chapter 6

Conclusions and Future Work

The goal of the research work summarized in this thesis is motion segmentation of image sequences. Many classical algorithms in the literature perform motion segmentation using just two frame correspondence. These algorithms hold great potential for extension to the spatiotemporal domain of image sequences. With this thought, motion segmentation for image sequences has been carried out using an algorithm that involves graph-based methods and geometric operations. Particularly, multiway cut tools with the Lucas-Kanade affine transformation for two frame motion and stereo correspondence has been studied, analyzed, and implemented. This idea has been extended to perform motion segmentation over image sequences.

In this thesis, relevant background work on all the ideas discussed was presented. Motion segmentation techniques from important papers in the literature were discussed. These papers talked about how image motion can be analyzed and used for correspondence and segmentation. Geometric models for representing pixel and region motion in images were described, as they are fundamental to this research work. A detailed procedure for Lucas-Kanade affine parameters estimation was given. Motion segmentation by graph cut based tools forms the core of this thesis and hence important fundamentals for graph-based methods used in computer vision were presented. Multiway cut algorithms were explained and

results on stereo correspondence were described. All algorithms for extending existing implementations were put forward. Motion segmentation over image sequences was the primary extension that exploited the power of multiway cuts and affine transformations. The secondary extension was hard constraint points for stereo. The results of these techniques on image sequences and stereo pairs were demonstrated.

There are number of future works to be mentioned. The idea of combining image sequences into one spatiotemporal volume was proposed in [8]. With ever increasing computational power, people have started processing image sequences as a single spatiotemporal block for various applications. Frame-to-frame methods are being replaced by batch processing techniques where a collection of frames is handled simultaneously. Multiway cuts can also be extended to analyze blocks of frames. Since multiway cuts are segmentation tools, all frames in an image sequences can be represented as a 3D graph and multiway cuts employed on this graph to get segmentation volumes rather than 2D regions. The algorithm proposed for getting motion segments over image sequences can be performed using this idea. The occlusion detection technique could be integrated with multiway cuts to get cleaner segmentation along borders of regions. Also, recent image segmentation methods [12] could be combined with the algorithms discussed in this thesis, to harness the power of statistical, combinatorial, and geometric concepts.

Bibliography

[1] S. T. Birchfield. http://www.ces.clemson.edu/∼stb/blepo/.

[2] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221 – 255, March 2004.

[3] S. T. Birchfield. Derivation of Kanade-Lucas-Tomasi tracking equation, 1996. http://www.ces.clemson.edu/∼stb/klt/birchfield-klt-derivation.pdf.

[4] S. T. Birchfield. *Depth and Motion Discontinuities*. PhD thesis, Dept. of Electrical Engineering, Stanford University, 1999.

[5] S. T. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406, April 1998.

[6] S. T. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *Proceedings of the Seventh International Conference on Computer Vision*, pages 489–495, September 1999.

[7] M. J. Black. *Robust Incremental Optical Flow*. PhD thesis, Yale University, 1992.

[8] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar image analysis: An approach to determining structure from motion. *IEEE International Journal of Computer Vision*, 1:7–56, 1987.

[9] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proceedings of the Eighth IEEE International Conference on Computer Vision*, volume 1, pages 105–112, July 2001.

[10] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 377–384, September 1999.

[11] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[12] D. Comaniciu and P. Meer. Robust analysis of feature spaces: Color image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 750–755, June 1997.

[13] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. The Massachusetts Institute of Technology Press, 2001.

[14] S. Dickinson, M. Pelillo, and R. Zabih. Introduction to the special section on graph algorithms in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1049–1052, 2001.

[15] S. Geman and D. Geman. Stochastic relaxation, gibbs distribution, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[16] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B*, 51:271–279, 1989.

[17] V. Kolmogorov. *Graph based Algorithms for Scene Reconstruction from Two or More Views*. PhD thesis, Cornell University, 2004.

[18] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *Proceedings of the IEEE International Conference on Computer Vision*, 2001.

[19] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision*, 2002.

[20] V. Kolmogorov and R. Zabih. Spatially coherent clustering using graph cuts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 437–444, 2004.

[21] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts?. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):147–159, February 2004.

[22] M. H. Lin and C. Tomasi. Surfaces with occlusions from layered stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[23] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1981.

[24] R. Megret and D. DeMenthon. A survey of spatio-temporal grouping techniques. Technical Report LAMP-TR-094,CS-TR-4403,UMIACS-TR-2002-83,CAR-TR-979, University of Maryland, College Park, 2002.

[25] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *Proceedings of the Sixth IEEE International Conference on Computer Vision*, pages 1154–1160, 1998.

[26] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[27] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, June 1994.

[28] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[29] J. Wang and E. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3:625–638, 1994.

[30] Y. Weiss and E. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 321–326, 1996.

[31] J. Xiao and M. Shah. Motion layer extraction in the presence of occlusion using graph cut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004.