ABSTRACT

Title of dissertation: THE COMPOSITIONAL CHARACTER OF VISUAL CORRESPONDENCE

Abhijit S. Ogale, Doctor of Philosophy, 2004

Dissertation directed by: Professor Yiannis Aloimonos, Department of Computer Science

Given two images of a scene, the problem of finding a map relating the points in the two images is known as the correspondence problem. Stereo correspondence is a special case in which corresponding points lie on the same row in the two images; optical flow is the general case. In this thesis, we argue that correspondence is inextricably linked to other problems such as depth segmentation, occlusion detection and shape estimation, and cannot be solved in isolation without solving each of these problems concurrently within a compositional framework. We first demonstrate the relationship between correspondence and segmentation in a world devoid of shape, and propose an algorithm based on connected components which solves these two problems simultaneously by matching image pixels. Occlusions are found by using the uniqueness constraint, which forces one pixel in the first image to match exactly one pixel in the second image. Shape is then introduced into the picture, and it is revealed that a horizontally slanted surface is sampled differently by the two cameras of a stereo pair, creating images of different width. In this scenario, we show that pixel matching must be replaced by interval matching, to allow intervals of different width in the two images to correspond. A new interval uniqueness constraint is proposed to detect occlusions. Vertical slant is shown to have a qualitatively different character than horizontal slant, requiring the role of vertical consistency constraints based on

non-horizontal edges. Complexities which arise in optical flow estimation in the presence of slant are also examined. For greater robustness and flexibility, the algorithm based on connected components is generalized into a diffusion-like process, which allows the use of new local matching metrics which we have developed in order to create contrast invariant and noise resistant correspondence algorithms. Ultimately, it is shown that temporal information can be used to assign correspondences to occluded areas, which also yields ordinal depth information about the scene, even in the presence of independently moving objects. This information can be used for motion segmentation to detect new types of independently moving objects, which are missed by state-of-the-art methods.

THE COMPOSITIONAL CHARACTER OF VISUAL CORRESPONDENCE

by

Abhijit Satishchandra Ogale

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2004

Advisory Committee

Professor Yiannis Aloimonos, Chair
Professor Larry Davis
Professor Steven Zucker (Yale University)
Professor David Jacobs
Professor Cynthia Moss, Dean's representative

Acknowledgements

I would like to thank my advisor, Professor Yiannis Aloimonos, for giving me an opportunity to explore the exciting world of vision. His deep understanding of the issues in vision, genuine scientific curiosity, and open-mindedness towards new ideas has gone a long way in making my research experience a rewarding one. I thank him for encouraging me and giving me the freedom to explore even the wildest of ideas. His warm personality has made working with him a very pleasant experience.

I would also like to thank Dr. Cornelia Fermüller for discussing various problems from time to time and sharing her valuable insights. My fellow graduate students and friends Jan Neumann, Patrick Baker, Hui Ji and Gutemberg Bezerra have all contributed towards creating an interactive and pleasant atmosphere in the lab.

I would like to thank the members of my examining committee, Professor Larry Davis, Professor David Jacobs, Professor Steven Zucker and Professor Cynthia Moss for their comments and suggestions for improving this thesis. I would also like to thank the late Professor Azriel Rosenfeld, who was a member of my proposal committee, for his help and advice in finding valuable references. I thank the members of the staff at the Center for Automation Research, UMIACS and the Department of Computer Science for their help with many administrative matters.

This thesis is dedicated to my parents for their unending love and support, without which none of this would be possible. My father's academic achievements have been a great source of inspiration over the years, and I am grateful to him for introducing me to the pleasures of scientific curiosity and understanding. My mother has played a pivotal role in shaping my career and shaping me as a person, and I owe her a great debt of gratitude. I would also like to thank my sister Deepti and my grandparents for their support and encouragement. Last but not the least, I thank my wife Neeti.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The field of Computational Vision has experienced tremendous growth in the past thirty years. This was due in part to the wide range of applications that may be realized if one understands to some degree how vision works, and partly due to the desire to understand how the brains of biological systems are designed. The field has seen several novel theories regarding the solution to core problems such as stereo matching [SS02], computation of image motion (optical flow) [BFB94, BB95, MB96, LHH$^+$98, GMN$^+$98], shape from texture [BL76, Wit81, Alo88, BA89, SB95], structure from motion [Fau93, HN94, MSKS04], and so on (the references given here are mostly surveys and books; see thesis chapters for detailed references). There has also been significant progress in understanding the geometric constraints underlying multiview vision [HZ00]. Despite the tremendous progress, we are still unable to effectively deal with a variety of inputs which are encountered in the real world. For example, although the field has advanced a great deal in the estimation of image motion, state of the art algorithms have difficulty finding occlusions, i.e. places in the scene that were visible at some instant of time and became invisible at the next time instant, or vice versa. Similarly, many stereo algorithms fail when presented with an input where there are few features (i.e. large untextured areas), strongly slanted surfaces (e.g. the corridor walls in the top row of Figure 1.1), when the contrast of one image differs significantly from another (e.g. bottom row of Figure 1.1), or when noise is present in one or more frequency channels. Motion segmentation algorithms, i.e. techniques for finding independently moving objects in a video taken by a camera moving in an unrestricted manner, fail miserably when the background motion is similar to the independent motion.

One may consider these cases as exceptional and hardly an obstacle to the realization of practical robust systems. This is, however, not true. Consider the following application of extreme relevance to the automotive industry. Imagine that a pair of cameras (a stereo system) is installed on a car with the goal of finding a spatial layout of the environment in front of the car, as well as the location of independently moving objects, i.e. other vehicles or humans and animals. Imagine that we are driving towards the North in the morning

Figure 1.1: Top row: stereo images of a scene with slanted untextured surfaces. Bottom row: stereo pair with mismatched contrast

hours, with the sun to our left. The stereo pair of images that the system will acquire will probably be such that the left image will be much brighter than the right one. Unequal contrast may also result due to differences in aperture and exposure for the two cameras. Matters are complicated further by the presence of large untextured slanted regions such as roads, walls of buildings or surfaces of other vehicles. Most stereo algorithms will simply fail in this case. Imagine further (see Figure 1.2), that as we drive along, on the periphery of the left camera which is not visible in the right camera, a car appears from behind a truck, both moving in the opposite direction as our own. Existing motion segmentation algorithms will miss such a moving object, classifying it as part of the background, which also moves in the same direction.

Thus, it becomes essential to reexamine basic visual processes that lie at the heart of low or intermediate level vision. These are the well known processes of image correspondence including stereo matching and optical flow, shape from X, structure from motion, motion segmentation, and the like. They are the processes responsible for creating de-

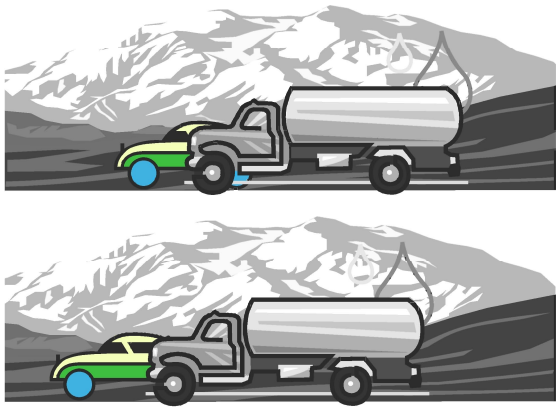

Figure 1.2: Two images at successive time instants of a car emerging from behind a truck. All the objects including the background move to the left.

scriptions of the surfaces surrounding us, their boundaries and discontinuities, as well as descriptions of the movements of the observer or of other objects. These are descriptions of a non-cognitive nature and their recovery depends on the appropriate utilization of the underlying geometry and physics. The central question is: what are the major issues which are preventing us from making these processes more robust and accurate?

1.1 Modularity vs. Compositionality

Given a complex multifaceted problem, we often use a modular approach which breaks up the larger problem into multiple modules or black boxes which are then connected together to solve the original problem. Like much of engineering, contemporary computer vision often works in such a modular manner. Consider the example of structure from motion. We have as an input at least two images of a scene taken from two different viewpoints (e.g. a video taken by a moving camera), and would like to develop a three dimensional model of the scene in view. First, the images are matched, by solving the correspondence problem. After this, the 3D camera motion (or the rigid transformation between the views) is estimated using the correspondence and geometric constraints. After this step, one can place the cameras in the world, i.e. we know the viewpoints from which the images were taken. The last step amounts to a triangulation: knowing the correspondence and the rigid transformation between the views, we can compute the location of each point of the scene in view and build a 3D model of the scene structure. One notices immediately that the

processing has a modular character, and information is passed from one module to the next in a feedforward manner.

Consider now the same problem in the presence of independently moving objects, the so-called motion segmentation problem. Finding the scene structure requires that we know the camera motion (the egomotion). Computing the camera motion requires us to first find the background, i.e. parts of the scene which are not independently moving. Finding the background, however, is equivalent to locating the independently moving objects, which cannot be done without first knowing the camera motion and the structure. The three problems of egomotion estimation, structure estimation and moving object detection are completely entangled in a chicken-and-egg loop.

Now let us focus our attention on the correspondence problem, which is required to solve the motion problem mentioned above. Local properties such as color or intensity alone are not sufficient to solve the point correspondence problem between two images, since matching based on these properties alone yields a large set of possibilities. Additional assumptions about scene smoothness are necessary to obtain unique solutions. However, in order to obtain the best results, we need to use smoothness constraints but respect depth discontinuities at the same time. Hence, depth segmentation (in the image space, not 3D) is needed in order to solve the correspondence problem accurately. On the other hand, we do not know the segmentation beforehand, but we could compute it if we knew the correspondence map by searching for discontinuities. Thus, we have another set of chicken-and-egg problems whose solutions depend on each other.

Now assume that we have a stereo system viewing a planar patch in the scene which is horizontally slanted. The slant causes the patch to be sampled differently by the two cameras, creating images of different width (notice the width of the slanted left wall of the corridor shown in Figure 1.1). If we knew the slant of the patch, we could stretch (or unwarp) one of the images to equalize the sampling, so that we could solve for the correspondence accurately, perhaps by applying signal processing operations for accurate matching. But we do not know the slant beforehand, so how can we find it? We can find it if we knew the correspondence. Thus, estimation of shape is also a part of the chicken-and-egg loop containing correspondence and segmentation. Another example of a chicken-and-egg pair are the processes of texture segmentation and shape from texture. The list goes on and on. If you consider a process from the low and intermediate level

vision repertoire, the chances are that you will find that it depends on another process, which in turn depends on the original process and perhaps other related processes.

It appears then that vision, at least at the low and intermediate level, is a compositional problem, consisting of several subprocesses which are parts of an elaborate set of feedback loops. Our role as computational vision theorists is then to discover and implement such loops. The word discover is used since vision is a physical process which exists in the biological world. One of the remarkable discoveries in neuroscience is the presence of extensive feedback between different parts of the visual system. Many details are known about this architecture, but we are far from a complete understanding of how the system functions as a whole. Thus, a theory of vision considered as a compositional process will not only contribute to robust artificial vision systems, but can also serve as a source of hypotheses that can be addressed with specific experiments in the neurosciences.

Although there exist many interdependent processes in single image analysis as well, in this dissertation, we shall concentrate on processes that involve more than one image. This is because the positions of corresponding image points in different images are governed by well known geometric principles, which makes problem formulation easier, and better reveals some of the compositional relations involved as compared to the harder problem of analyzing a single image. As far as low and intermediate level vision is concerned, the problem of correspondence, i.e. matching images is considered one of the most basic problems, as it represents the foundation of a large number of higher level processes. In this dissertation, we shall concentrate on the relationships between the correspondence problem and well known problems such as segmentation, shape estimation and occlusion detection. (By segmentation, we mean the process which detects discontinuities in well-defined properties such as disparity, flow or 3D motion.) We shall develop solutions which extend the state of the art to deal with inputs which currently pose serious problems. This dissertation shows how the new framework allows us to build a number of retinotopic maps for a binocular observer in motion, including stereo disparity, image motion (optical flow), occlusions, shape, ordinal depth, and motion segmentation. In the next section, we shall describe how the compositionality framework is progressively developed in the dissertation, and provide a road map of the ideas which will be introduced.

1.2 Organization: a road map of the thesis

Given two images of the same scene from different viewpoints, the dense point correspondence problem deals with finding a piecewise continuous map which relates points in one image to points in the other image. We begin in Chapter 2 by examining the problem in a world which contains no slanted surfaces. In such a fronto-parallel Flatland, two stereo images of the same surface have the same width, which allows us to solve the pixel correspondence problem instead of the point correspondence problem. In this shapeless world, we show how correspondence (stereo and optical flow) and segmentation can be solved together in a compositional fashion by using connected components. Occlusions are also found using the pixel uniqueness constraint. In Chapter 3, we introduce shape into the picture, to show how a horizontally slanted object projects onto stereo images of different width, causing problems due to uneven sampling and lack of a pixel uniqueness constraint for finding occlusions. These problems are addressed by presenting a compositional algorithm which solves for correspondence, shape and slant simultaneously. We also address problems caused by the presence of untextured regions and vertically slanted surfaces, and discuss the difficulties involved in formulating smoothness constraints for optical flow in the presence of slant. In Chapter 4, we generalize our algorithm based on connected components into a diffusion formulation, which increases robustness, and allows for a broader selection of local matching metrics. In Chapter 5, we develop local metrics which perform matching in a manner which is invariant to the contrast of the two images. One of these local metrics is biologically motivated and uses multiple spatial frequency channels to improve robustness with respect to noise in one of the channels. In Chapter 6, we demonstrate how occluded regions can be filled by optical flow values from neighboring regions even in the presence of independently moving objects. The underlying method uses segmentation obtained from previous frames, thereby presenting another case for the presence of feedback over time. We also show that if the occlusions can be filled, then ordinal depth relations can easily be derived between different regions of the scene, even if the scene contains independent motion. In Chapter 7, occlusion and ordinal depth information obtained in previous chapters is used to find a novel solution to the motion segmentation problem. In fact, if motion information is available, then the problems of formulating smoothness constraints for optical flow are alleviated. Hence, knowledge of

the egomotion can actually help us find the correct correspondence, which testifies to the feedback of information all the way back to the roots. In this thesis, we shall present robust correspondence algorithms which find discontinuities, deal with untextured regions, handle slanted surfaces, find occlusions and ordinal depth, and are able to perform in the presence of large non-uniform changes in contrast between two images. Finally Chapter 8 explores open problems and possible avenues of further research.

Chapter 2

Correspondence and Segmentation in Flatland

The dense correspondence problem (also referred to as the *matching* problem) consists of finding a unique mapping between the points belonging to two images of the same scene. If the camera geometry is known, the images can be rectified ([Fau93, HZ00]), and the problem reduces to the stereo correspondence problem, where points in one image can correspond only to points along the same horizontal line in the other image. If the geometry is unknown, then we have the optical flow estimation problem. In both cases, regions in one image which have no counterparts in the other image are referred to as occlusions (or more correctly as *half-occlusions*).

2.1 Previous work

There exists a considerable body of work on the dense stereo correspondence problem. Scharstein and Szeliski [SS02] have provided an exhaustive review and comparison of dense stereo correspondence algorithms. Dense matching algorithms generally utilize local measurements such as image intensity (or color) and phase, and aggregate information from multiple pixels using smoothness constraints. The simplest method of aggregation is to minimize the matching error within rectangular windows of fixed size [OK93]. Better approaches utilize multiple windows [GLY92, FRT97], adaptive windows [KO94] which change their size in order to minimize the error, shiftable windows [BI99, TSK01], or predicted windows [MD00], all of which give performance improvements at discontinuities.

Global approaches to solving the stereo correspondence problem rely on the extremization of a global cost function or energy. The energy functions which are used include terms for local property matching ('data term'), additional smoothness terms, and in some cases, penalties for occlusions. Depending on the form of the energy function, the most efficient energy extremization scheme can be chosen. These include dynamic programming [OK85], simulated annealing [GG84, Bar89], relaxation labeling [Sze90], non-linear diffusion [SS98], maximum flow [RC98] and graph cuts [BVZ01, KZ01]. Maximum flow

and graph cut methods provide better computational efficiency than simulated annealing for energy functions which possess a certain set of properties. Some of these algorithms treat the images symmetrically and explicitly deal with occlusions (eg. [KZ01]). The uniqueness constraint [MP79] is often used to find regions of occlusion. Egnal and Wildes [EW02] provide comparisons of various approaches for finding occlusions. Recently, some algorithms [BT99] have explicitly incorporated the estimation of slant while performing the estimation of horizontal disparity. Lin and Tomasi [LT03] explicitly model the scene using smooth surface patches and also find occlusions; they initialize their disparity map with integer disparities obtained using graph cuts, after which surface fitting and segmentation are performed repeatedly.

As is the case with stereo correspondence, there exists a large body of literature devoted to the understanding of the optical flow problem, including the dense flow estimation problem. Beauchemin et al [BB95] and Mitiche et al [MB96] provide surveys of the various techniques for optical flow estimation, while Barron et al [BFBB94], and more recently, Galvin et al [GMN$^+$98] and Liu et al [LHH$^+$98], provide performance comparisons between various optical flow algorithms. Mitiche et al [MB96] also discuss the problems of finding motion based segmentation and occlusions, and survey related approaches. There also exists some very interesting recent work on explicitly finding occlusions and motion discontinuities [BF00, ADPS02], and also regarding the spectral properties of occlusions [BB00] in the context of optical flow.

2.2 Flatland

When we compute the disparity map or the optical flow from a given pair of images, the desirable property of such a map is that it should explain the observed images while minimizing the number of discontinuities. In other words, we would like to model the disparity (or the optical flow) as a *piecewise continuous* function which is consistent with the observed images and has the minimum possible number of pieces. To simplify the problem computationally, we often choose more restrictive versions of the general model of a piecewise continuous function. The simplest but most restrictive version models the disparity map as a *piecewise constant* function. An obvious improvement is to model the depth map as a *piecewise linear* (ie. planar) function. We can proceed in this manner towards progressively

complex models in an attempt to get closer to the true property of piecewise continuity.

One aspect to keep in mind is that when we speak of segmentation, we mean depth segmentation on the image. However, surfaces which are connected in the 3D scene may project to multiple disconnected regions due to self overlap. We are not attempting to find the 3D continuity of such surfaces, which may involve other principles.

In this chapter, we shall begin with the simplest but most restrictive case of trying to model the disparity as a piecewise constant function. This assumption models the scene as a shapeless world consisting of a collection of flat fronto-parallel surfaces, hence the name *Flatland*. We assign unique correspondences to each pixel and do not deal with transparent surfaces. In this world, we show that correspondence and segmentation are chicken-and-egg problems which can only be solved simultaneously.

2.3 Chicken-and egg problems

Establishing correspondence between two images of a scene involves first selecting a local metric, such as the intensity (gray level) or color of a pixel which forms the basis for local comparisons. However, matching on the basis of such local information alone is almost impossible since many pixels have similar intensity or color. To reduce the correspondence possibilities for a pixel to a single possibility, regions around that pixel must be used along with additional continuity or smoothness assumptions about the scene. Thus, information around a pixel must be *aggregated* to obtain a unique match. Enforcing smoothness without a prior knowledge of depth discontinuities (segmentation) will inevitably lead to errors, especially near the discontinuities. Hence, prior knowledge of the segmentation is essential in order to correctly define regions around a pixel for information aggregation. Conversely, if exact correspondence is known, the segmentation may be easily deduced.

Thus, if we knew the segmentation, then we could better estimate the correspondence. But we need correspondence in order to achieve segmentation. Correspondence and segmentation are chicken-and-egg problems: we need one in order to solve the other. Any recipe for solving such cyclic problems must involve feedback, either implicitly or explicitly. In the following section, we present a method which does not separate the problem of finding correspondence from the problem of finding the segmentation at any stage, and explicitly demonstrates the interdependence of correspondence and segmentation. Recently,

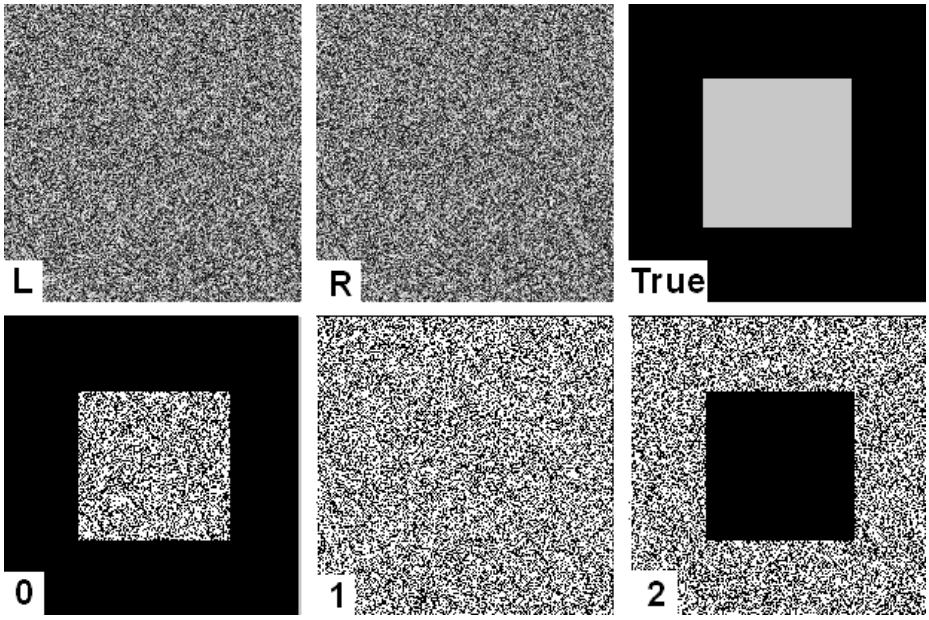

Figure 2.1: Top row: Left image, Right image, True disparity (black denotes 0, gray denotes 2). Bottom row: Absolute intensity difference of left and right images for horizontal shift $\delta_x = 0, 1, 2$

we have found that a method similar to our own but using different vertical consistency constraints also exists in the stereo literature [BVZ98].

2.4 Connected matching regions

Let $I_1(x, y)$ and $I_2(x, y)$ be a given pair of rectified stereo images. The absolute intensity difference between the two images is found using equation 2.1, where δ_x denotes the relative horizontal shift between the two input images. For the case of optical flow, the shifts will be two dimensional.

$$\Delta I(x, y, \delta_x) = |I_1(x, y) - I_2(x + \delta_x, y)| \tag{2.1}$$

The first row of Figure 2.1 shows a random dot pair of stereo images and the true disparity map. The second row shows the absolute intensity difference images for three horizontal shifts $\delta_x = 0, 1, 2$. If we observe the intensity difference images (second row), we notice that large connected regions of matching pixels (shown in black) appear for certain values of the shift. By the word 'match', we mean that the absolute intensity difference is below a certain threshold t.

$$\Delta I(x, y, \delta_x) < t \tag{2.2}$$

The appearance of large connected sets of matching pixels is the first observation of interest.

In the case of the random-dot pair, the background matches for $\delta_x = 0$ forming a large connected region, and the central square matches for $\delta_x = 2$. We know from the true disparity map that these shifts correspond to the correct disparities of the background and the square. However, we also notice that some pixels in the central square will match and form smaller connected regions even when the shift is wrong (not equal to 2). The same is true for the background pixels. *Thus, a pixel may form a part of a connected matching region even when the shift does not correspond to the true shift.* So how do we choose the correct shift for a pixel?

2.5 Boundaries and connectivity maximization

Recall our definition of Flatland - a world containing only fronto-parallel surfaces. Consider a uniformly colored region R_1 in image I_1 having a disparity δ_x. It corresponds to a region R_2 in the image I_2. Thus, if we shift image I_1 by δ_x and overlay it on I_2, then regions R_1 and R_2 will overlap and match perfectly and yield a connected region having an area equal to the size of R_1(which is the same as R_2). However, if the shift is not δ_x but has some other value, parts of R_1 and R_2 may still overlap and yield some connected matching region.

The area of overlap will be maximum only when the boundaries of R_1 and R_2 match perfectly, which occurs only for the true shift δ_x.

For all other shifts, the connected matching area will be less. Similarly, in case the region R_1 (and hence R_2) is textured, connected matching regions will also be obtained for shifts other than the true shift δ_x. For example, if the regions contain a periodic texture such as a checkerboard, with square size λ, then we will obtain connected matching regions even if the shifts are $\delta_x + 2m\lambda$, where m is an integer. Even if this happens, the largest connected region will occur only when the boundaries of R_1 and R_2 match, which happens only if the shift equals the true shift δ_x. It is clear that only the knowledge of region boundaries allows us to assign correct shifts to the interior pixels. Maximizing the area of connected matching regions around a pixel is intimately related to the matching of

region boundaries.

From another perspective, we can see that maximizing the size of matching regions is an attempt to minimize the segmentation. This is similar in spirit to the idea of a Minimum Description Length (MDL). As we had discussed earlier, our aim in Flatland is to find a piecewise constant disparity map which is consistent with the input images, and which has the *least number of pieces*. In order to get the least number of pieces, each piece must be as large as possible, which provides justification for the connectivity maximization criterion introduced above. Thus, in Flatland, we can assert that *for any image pixel* (x, y)*, the correct disparity* δ_x *maximizes the area* $A(x, y, \delta_x)$ *of the connected matching region containing that pixel.*

2.6 Vertical connectivity

In Figure 2.2, we see a stereo pair of images having a gray background which has zero disparity, and a white square in front which has a non-zero disparity. On the right of the figure, we show the absolute intensity difference between the two images for zero relative shift. The black portion of this image indicates regions whose intensities match perfectly for zero relative shift. Notice that a part of the white foreground object also matches for zero shift, and is connected to the large matching background region. This will cause the entire black portion visible in the right hand side image to be labeled with zero disparity, which is clearly an incorrect result.

The problem lies in the propagation of connectivity vertically across horizontal edges. In a stereo pair, if two single colored regions with different disparity are separated by an horizontal edge, then as we try out different horizontal shifts, points on both sides of the horizontal edge will always match regardless of the shift being tried. Thus, when we build connected components, regions on the two sides of the horizontal edge will form part of the same connected component, which causes both sides to be eventually assigned the same disparity.

Thus, in an image, if two single colored regions are separated by a horizontal intensity edge, then we must treat them as potentially having different disparities. Hence, before we build connected components on our thresholded intensity difference images, we must explicitly sever connections across horizontal edges. In the case of optical flow, the shifts we shall try will be two dimensional and the corresponding definition of a horizontal edge

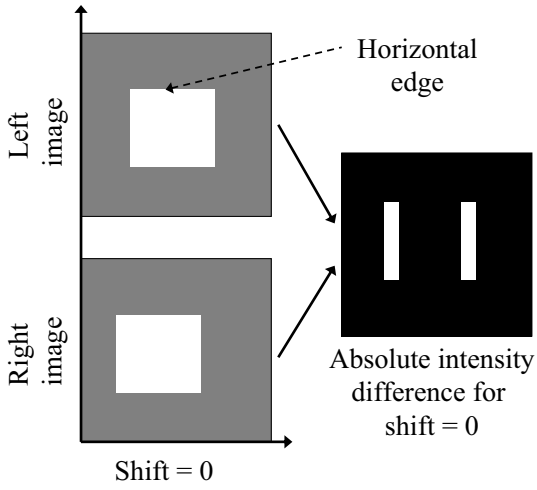

Figure 2.2: Vertical connectivity must not be established across horizontal edges

is different for each shift. Thus, if we are considering shift (δ_x, δ_y), then edges parallel to this direction are the *horizontal* edges.

In Chapter 3, we shall examine vertical connectivity in an even broader context which deals with the effects of shape on correspondence. We shall show that vertical connectivity must only be established across non-horizontal edges. This is more restrictive than the condition described above which severs connections across pixels separated by horizontal edges, because it also precludes the establishment of connections between a pixel and its vertical neighbor if they have the same intensity/color.

2.7 Occlusions and uniqueness

The *uniqueness constraint* enforces a one-to-one correspondence between pixels in the two images. Hence, a pixel in one image may match exactly one pixel in the other image and vice-versa. There exists a competition between pairs of pixels (p_L, p_R), where p_L is a pixel in the left image, and p_R is a pixel in the right image. If a pair (p_L, p_R) wins, it automatically excludes the existence of all pairs of the form $(p_L, p_{R'})$ and $(p_{L'}, p_R)$, such that $p_{L'} \neq p_L$ and $p_{R'} \neq p_R$. Some pixels, which do not form a part of any of the winning pairs, are the occlusions. It is possible to enforce the uniqueness constraint within the correspondence search itself as it progresses: whenever we assign a new partner to a given pixel, we make sure that it's previous partner (if it was previously paired) is marked as unpaired.

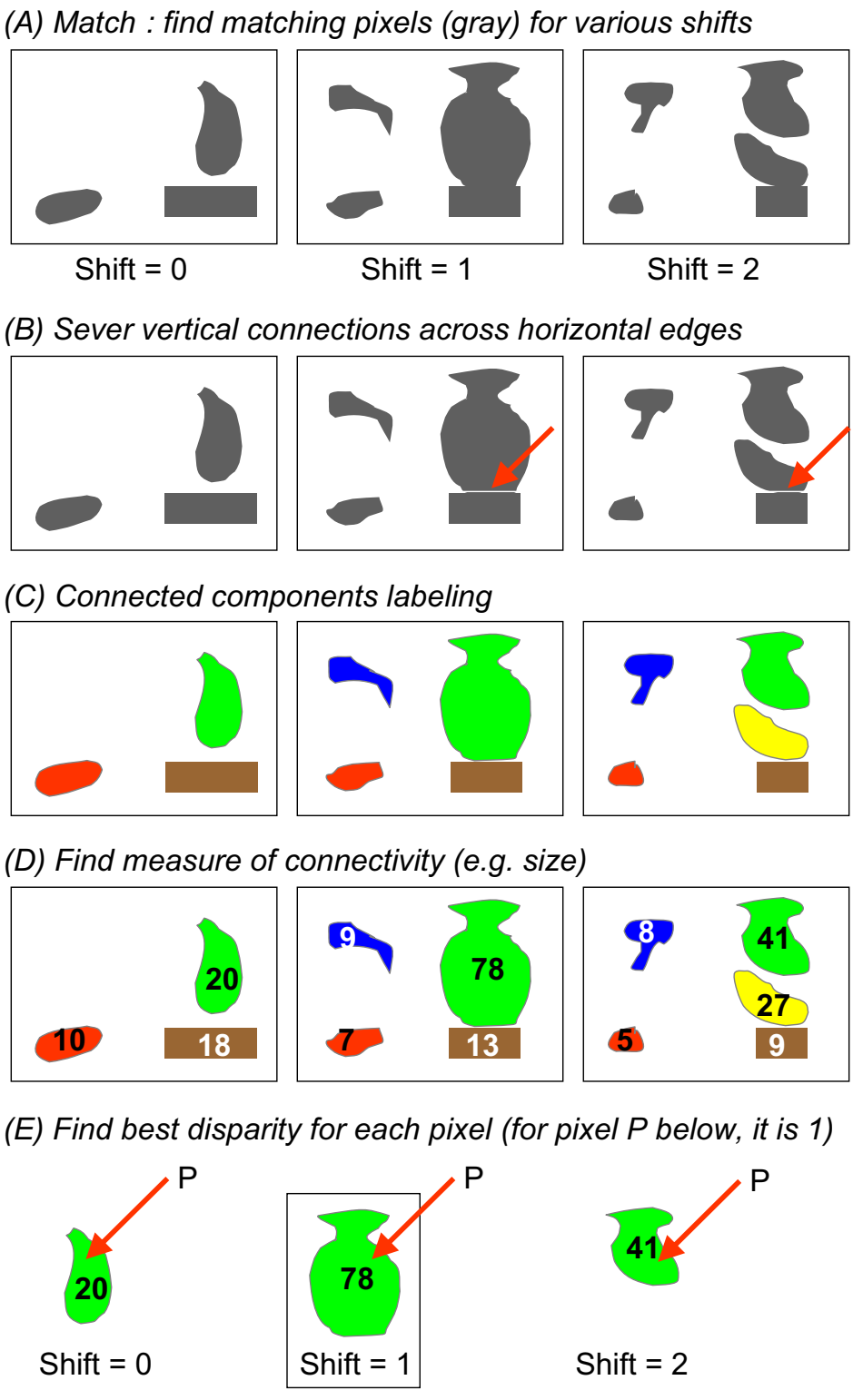

Figure 2.3: An Algorithm for Flatland. Besides these steps, the uniqueness constraint is enforced to find occlusions.

2.8 An algorithm for Flatland

Our algorithm for Flatland is outlined in Figure 2.3. Basically, the algorithm consists of the following steps:

1. For every shift $\delta_x \in \{\delta_1, \delta_2, ..., \delta_k\}$, do
   (a) Shift the left image $I_L$horizontally by $\delta_x$to get I'_L
   (b) Match $I'_L$with I_R
   (c) Sever vertical connections across horizontal edges
   (d) Build connected components and compute a connectivity metric
   (e) For each pixel, if the connectivity increases, update left and right disparity maps, preserving uniqueness.

For edge detection, we use the Canny edge detector. For the case of optical flow, the only difference is that the shifts are two dimensional, and in step (c), we sever connections using the appropriate definition of a horizontal edge for each shift.

For d possible shifts and an image with N pixels, the total running time is $\Theta(Nd)$. In our implementation, we use the technique of Birchfield and Tomasi [BT98] to calculate the absolute intensity differences for matching. For color images, matching two pixels implies matching all their color components. Measures of connectivity other than the area may also be used leading to minor improvements; for example, we may use a combination of the area of the connected component and the total intensity difference inside the connected component, to reduce the sensitivity to threshold selection. Also, pixels which locally match for k shifts out of d possible shifts can be assigned to have an area of $1/k$; this ensures that pixels which match frequently do not dominate the estimation.

In Chapter 4, we show that connectivity is merely a mechanism for propagating the influence of one pixel to another, which can be generalized to a diffusion process for greater robustness and wider choice of non-binary matching metrics.

2.9 Experimental Results

Figure 2.4 shows the disparity maps for four standard test sequences (obtained from the website *www.middlebury.edu/stereo*), created by Scharstein and Szeliski [SS02]. Figure 2.5

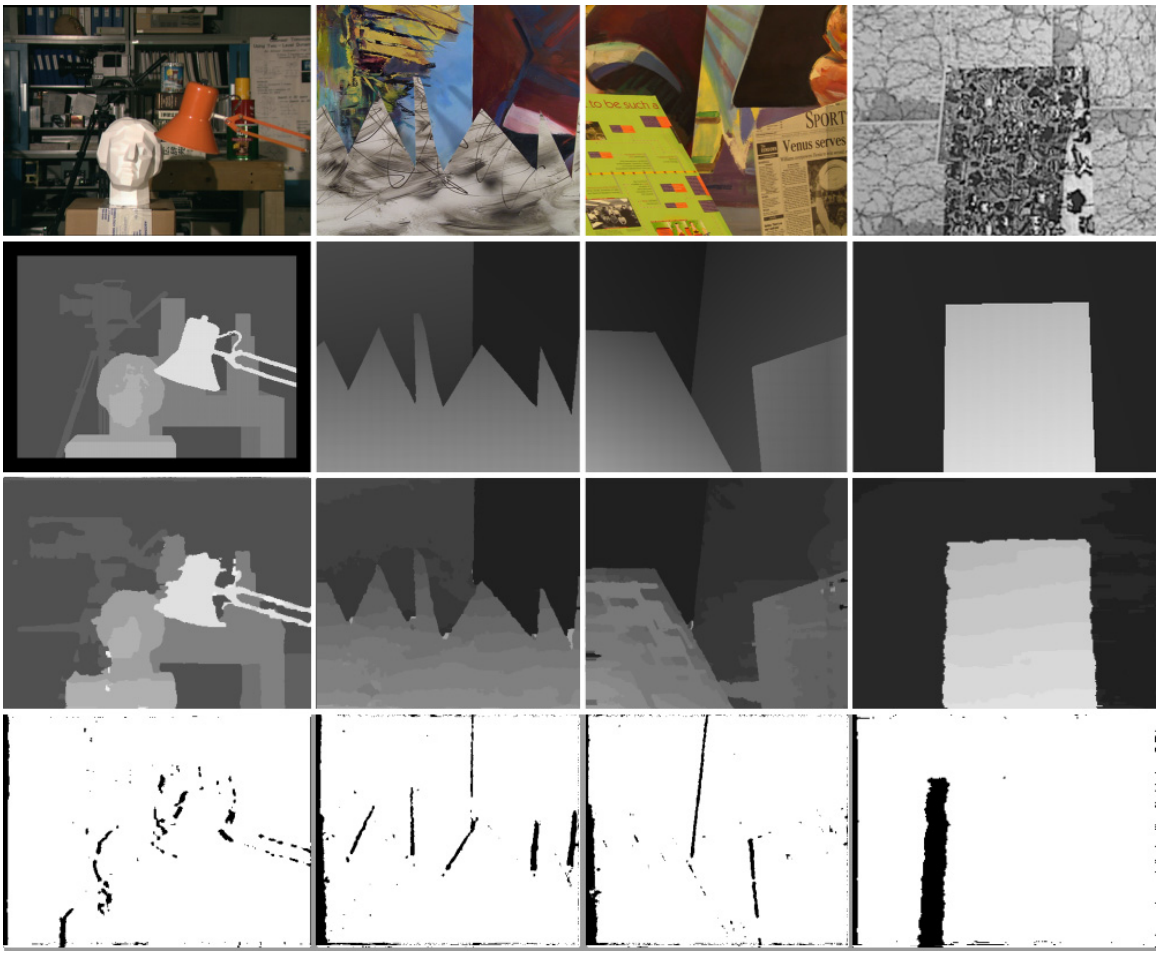

Figure 2.4: Top row: left images from four stereo pairs *tsukuba, sawtooth, venus and map.* Second row: true disparity maps. Third row: our results with occlusions filled in as required by the Scharstein and Szeliski evaluation. Bottom row: the detected occlusions are shown separately.

shows the results of using the above algorithm to compute optical flow on two frames of the *table-vase* sequence, while Figure 2.6 shows the results on two standard test sequences: the *yosemite* sequence, and the *sofa* sequence, which shows a rotating object.

Real world scenes are quite unlike Flatland, since they rarely consist of fronto-parallel surfaces. In the next chapter, we shall identify new issues which arise in the presence of slanted surfaces, which require us to alter our definitions of correspondence and introduce novel constraints for detecting occlusions. In Figure 2.4, notice the errors in the computed disparity in the *venus* sequence for the untextured regions of the slanted plane on the left. In the next chapter, we shall how untextured vertically slanted surfaces require us to modify our vertical consistency constraints in order to obtain the correct disparity.

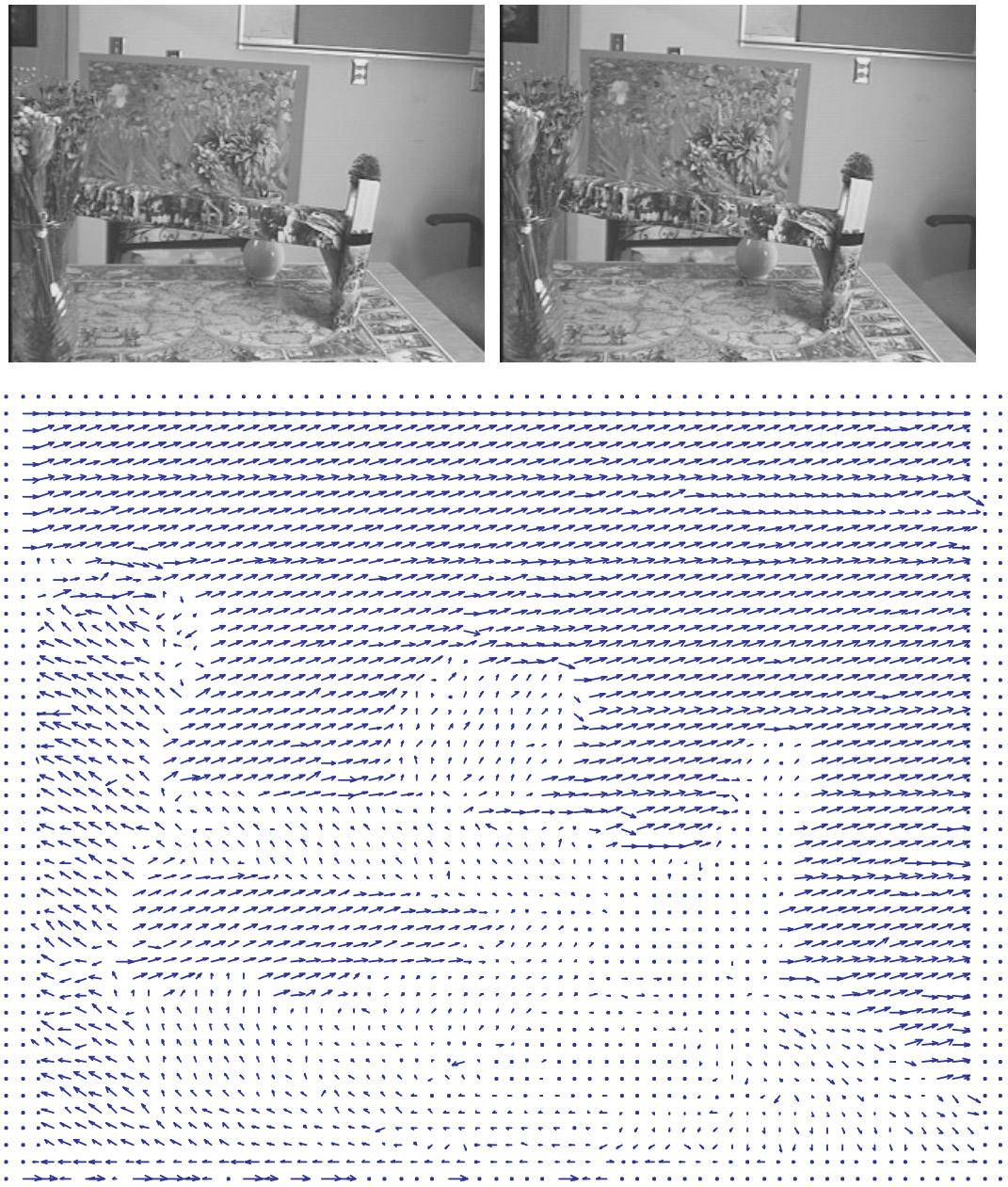

Figure 2.5: Top row: two frames of the *table-vase* sequence. Second row: computed optical flow field.

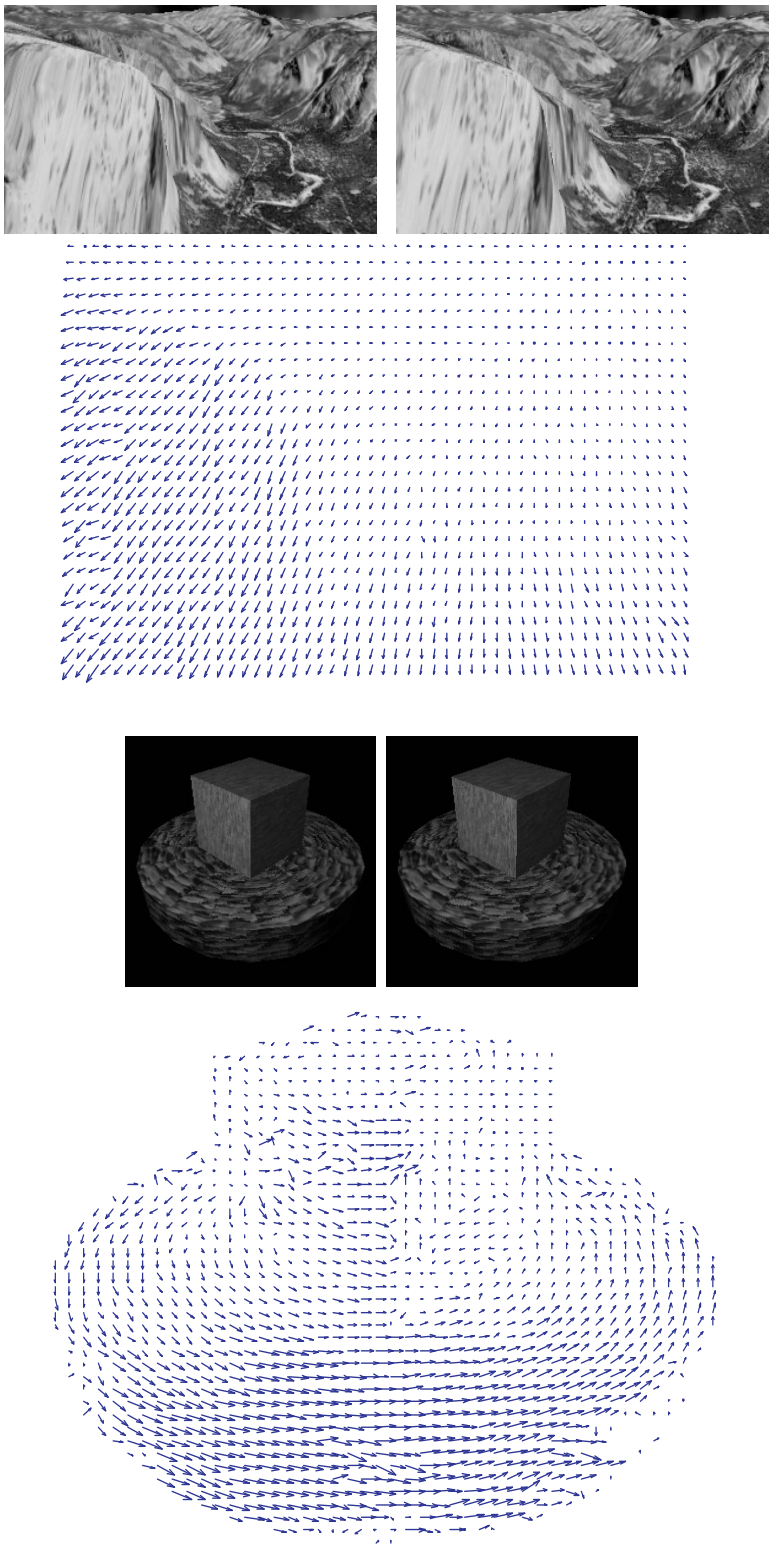

Figure 2.6: First and second rows: Two frames of the *yosemite* sequence and the computed optical flow. Third and fourth rows: Two frames of the *sofa* sequence and the computed optical flow.

Chapter 3

Shape and Correspondence

Previously, we mentioned that the desirable property of a disparity map (or optical flow) is that it should explain the observed images while minimizing the number of discontinuities. Ideally, we would like to model the disparity (or the optical flow) as a *piecewise continuous* function which is consistent with the observed images and has the minimum possible number of pieces. In the previous chapter, we chose the simplest but most restrictive approximation to piecewise continuity, which modeled the disparity map as a *piecewise constant* function. We termed this world as Flatland, since the scene was modeled as a collection of flat fronto-parallel surfaces. In this chapter, we leave behind the shapeless Flatland, and introduce slant into the picture, thereby progressing to a *piecewise linear* model.

We begin by highlighting a known but unexploited geometric fact: a horizontally slanted surface (ie. having depth variation in the direction of the separation of the two cameras) will appear horizontally stretched in one image as compared to the other image. Thus, while corresponding two images, N pixels on a scanline in one image may correspond to a different number of pixels M in the other image. This leads to three important modifications to existing stereo algorithms: (a) due to unequal sampling, existing intensity matching metrics must be modified, (b) unequal numbers of pixels in the two images must be allowed to correspond to each other, and (c) the uniqueness constraint, which is often used for detecting occlusions, must be changed to an interval uniqueness constraint. These ideas on horizontal slant appeared recently in our paper [OA04] in CVPR 2004. We also discuss the asymmetry between vertical and horizontal slant, and the central role of non-horizontal edges in the context of vertical slant. Using experiments and comparisons, we discuss cases where existing algorithms fail, and how the incorporation of new constraints provides correct results.

In this context, it is important to mention recent efforts [BT99, LT03] to explicitly incorporate the estimation of slant while performing the estimation of horizontal disparity. In particular, Lin and Tomasi [LT03] explicitly model the scene using smooth surface

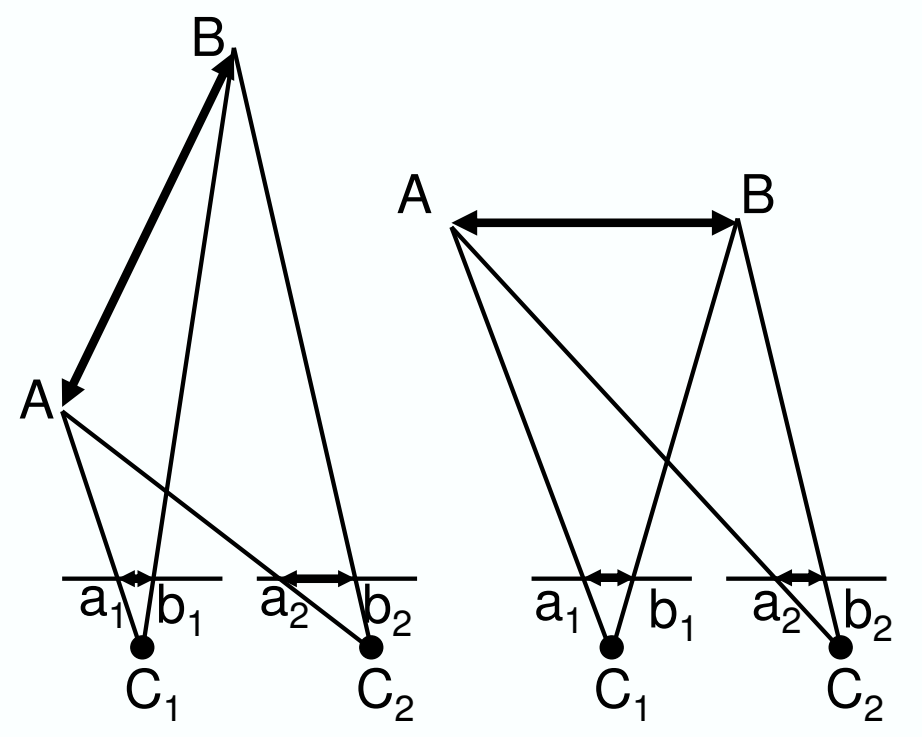

Figure 3.1: (a) unequal projection lengths of a horizontally slanted line (b) equal projection lengths of a fronto-parallel line

patches and also find occlusions; they initialize their disparity map with integer disparities obtained using graph cuts, after which surface fitting and segmentation are performed repeatedly.

3.1 Horizontal slant

Let us begin by introducing horizontally slanted surfaces into the scene, ie. the depth on such surfaces changes as we move along the X-axis (horizontally), and does not change if we move along the Y-axis. Let us also assume for simplicity that we are using a stereo system with a parallel viewing geometry and in which the cameras are separated only by a translation along the X-axis. Our system therefore provides us with rectified images.

3.1.1 Unequal projection lengths and interval matching

Figure 3.1(a) shows that a horizontally slanted line AB in the scene projects onto the line segment a_1b_1 in camera C_1, and a_2b_2 in camera C_2. Clearly, the lengths of a_1b_1 and a_2b_2 are not equal. Assume that the cameras have focal length equal to 1. Let the point A have

coordinates (X_A, Z_A) in space with respect to camera 1, and point B have coordinates (X_B, Z_B), where the X-axis is along the scanline, and the Z-axis is normal to the scanline. Then, if the cameras are separated by a translation t, we can immediately find the lengths L_1 and L_2 of the projected line segments in the two cameras.

$$\begin{aligned} L_1 &= X_B/Z_B - X_A/Z_A \\ L_2 &= (X_B - t)/Z_B - (X_A - t)/Z_A \end{aligned} \tag{3.1}$$

Clearly, in general, $L_1$and L_2 are not equal. For the fronto-parallel line shown in Figure 3.1(b), $Z_A = Z_B = Z$, hence

$$L_1 = L_2 = (X_B - X_A)/Z \tag{3.2}$$

Thus, we have the following:

- *Except for the fronto-parallel case, horizontally slanted line segments in space will always project onto segments of different lengths in the two cameras.*
- *Consequently, N pixels on a scanline in one image can correspond to a different number of pixels M on a scanline in the other image.*

We must ensure that our stereo algorithms permit unequal correspondences of this nature; hence, an interval on a scanline in one image must be matched to an interval on a scanline in the other image, where the two intervals being matched may have different lengths. Note that the scanline is treated as a continuous entity rather than a discrete pixelized entity.

Conclusion: We must perform interval matching instead of pixel matching.

3.1.2 Slant affects Sampling

Since a horizontally slanted line segment in space has different projection lengths in the two cameras, it's intensity function is also sampled differently by the two cameras as shown in Figure 3.2. Birchfield and Tomasi [BT98] have provided a very useful method for matching pixel intensities, which is used by many of the best performing stereo algorithms. Let us briefly describe what this procedure does: Given two scanlines $I_L(x)$ and $I_R(x)$, we have to find the absolute intensity difference between pixel x_L in the left scanline

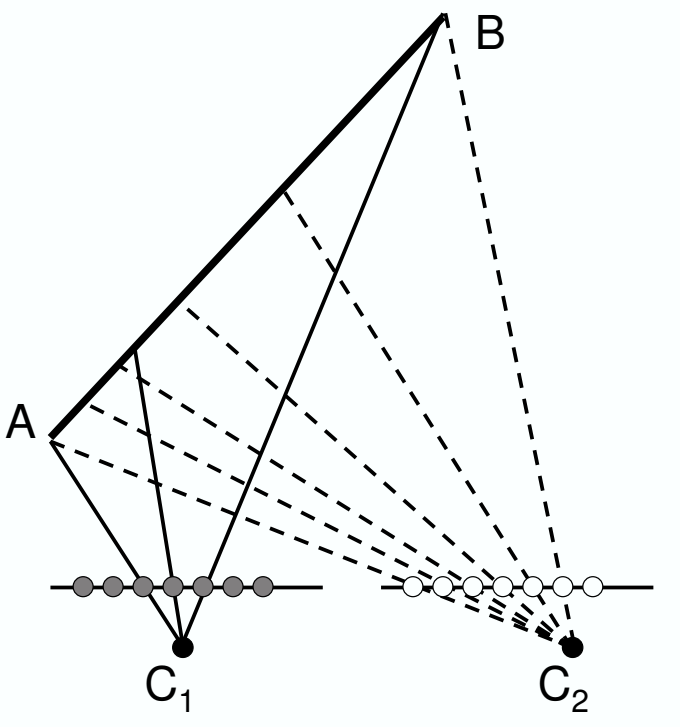

Figure 3.2: Sampling problem for a horizontally slanted line

and pixel x_R in the right scanline. We first find $I_L(x_L - 1/2)$, $I_L(x_L + 1/2)$, $I_R(x_R - 1/2)$ and $I_R(x_R + 1/2)$ by a simple linear interpolation. These values are used to find $I_L^{min} = min\{I_L(x_L - 1/2), I_L(x_L), I_L(x_L + 1/2)\}$, $I_L^{max} = max\{I_L(x_L - 1/2), I_L(x_L), I_L(x_L + 1/2)\}$, and similarly I_R^{min} and I_R^{max}. The left difference is $d_L = max\{0, I_L(x_L) - I_R^{max}, I_R^{min} - I_L(x_L)\}$ and the right difference is $d_R = max\{0, I_R(x_R) - I_L^{max}, I_L^{min} - I_R(x_R)\}$. Finally, the absolute intensity difference between the pixels is $d = min\{d_L, d_R\}$. The procedure is therefore symmetric and linearly interpolates the intensity function between neighboring pixels. Such a matching procedure cannot be applied directly in the presence of horizontal slant, due to the unequal sampling.

We must first resample each scanline correctly, and then apply the Birchfield-Tomasi matching method. In other words, we first stretch one of the scanlines, by an amount related to the horizontal slant we are considering, and then match this stretched scanline with the other unstretched scanline using the Birchfield-Tomasi matching method as usual. For example, if we are considering the linear correspondence function $x_R = mx_L + d$ between points of camera L and R, then we must stretch the image of camera L by a factor m before performing the intensity based matching. Thus, we first compute I_L^{min} and I_L^{max}

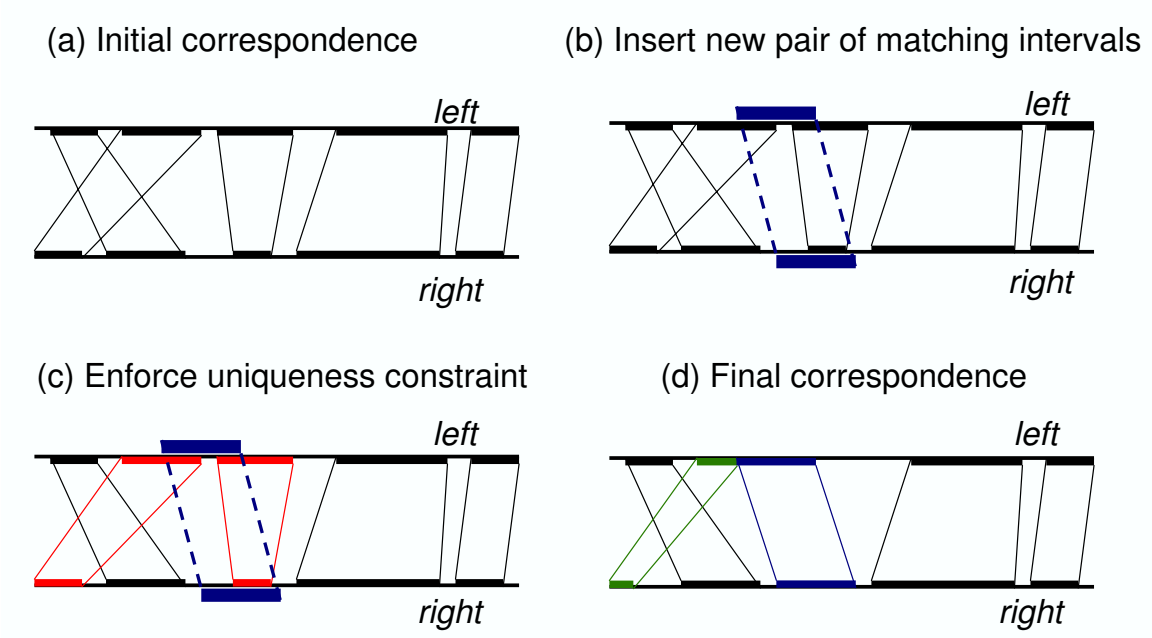

Figure 3.3: The modified uniqueness constraint operates by preserving a one-to-one correspondence between intervals on the left and right scanlines, instead of pixels.

for the unstretched scanline I_L, then stretch all three by a factor m, and then apply the remainder of the Birchfield-Tomasi method. As we try various values of the slant, we appropriately resample the scanlines before matching. Figure 3.4 shows how corresponding line segments of unequal length attain the same length after stretching one of the images.

Conclusion: Stretch one of the images first and then match .

3.1.3 Occlusions and the new interval uniqueness constraint

The uniqueness constraint [MP79] is often used to find occlusions. In its present form, the uniqueness constraint forces a one-to-one correspondence between pixels in the two images. In the end, the unpaired pixels are the occlusions. However, since horizontal slant allows N pixels in one image to match with a different number of pixels M in the other image, we can no longer impose a one-to-one correspondence for finding occlusions. We propose a new uniqueness constraint which enforces *a one-to-one mapping between continuous intervals* (line segments) in the two scanlines, instead of pixels. An interval in one scanline may correspond to an interval of a different length in the other scanline, as long as the correspondence is unique. This is equivalent to enforcing uniqueness in the scene space instead of the image space, hence we may also refer to this constraint as the 3D

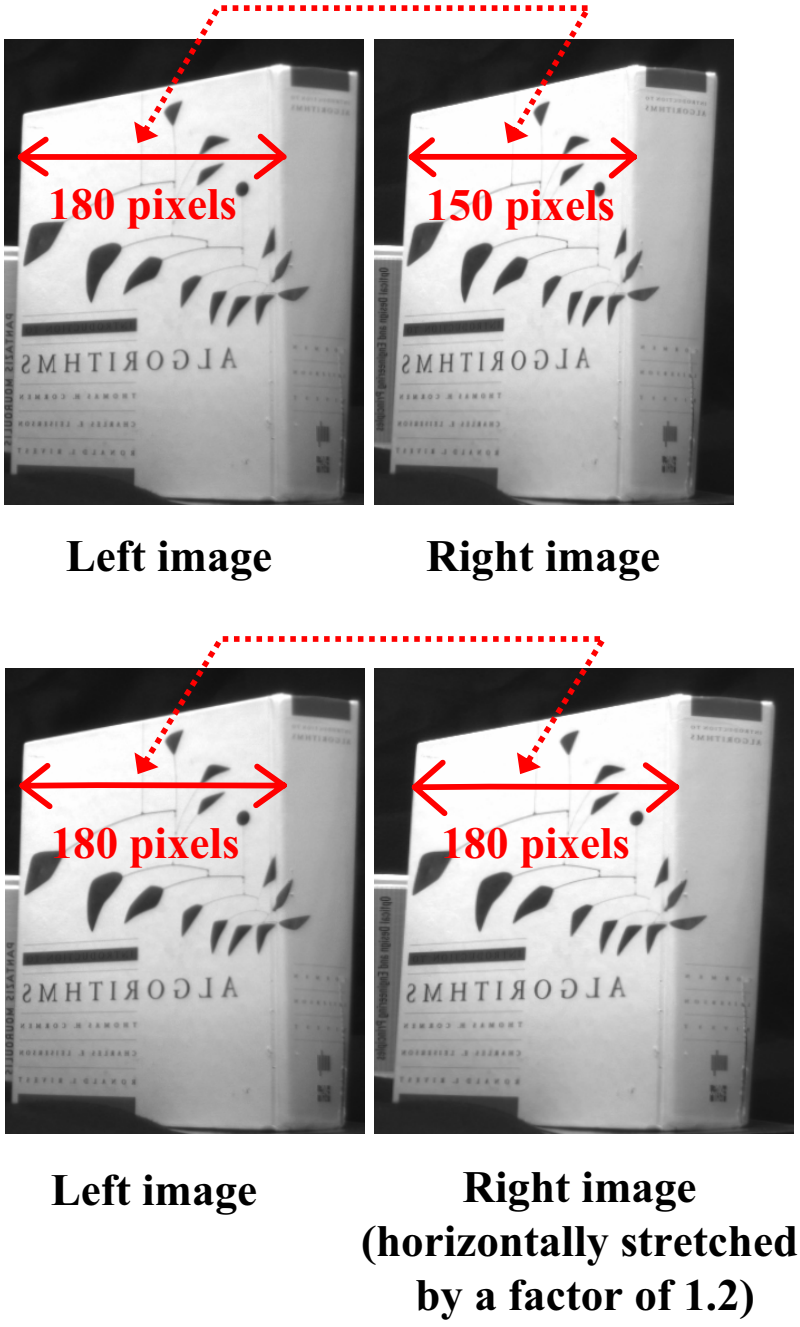

Figure 3.4: Stretch and match

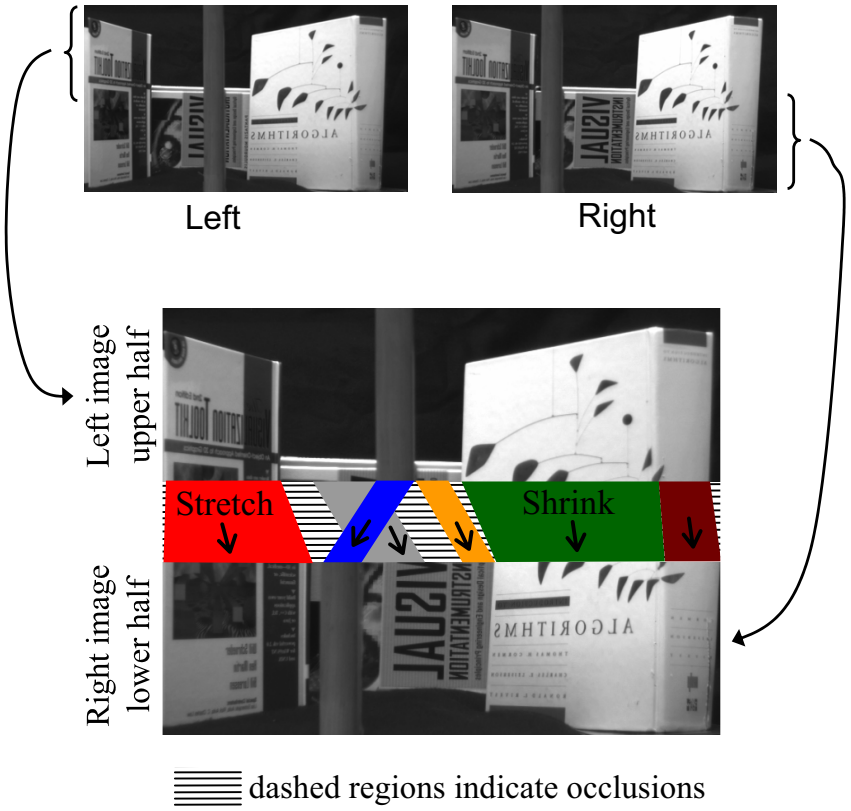

Figure 3.5: Top: stereo pair of images. Bottom: Corresponding intervals on the left and right scanlines can have different length. The order (left-right) of matching intervals can also change (see the blue and gray intervals).

uniqueness constraint. Figure 3.5 shows using a real example how intervals of different length correspond to each other, leaving behind the occlusions.

Figure 3.3 shows how the new uniqueness constraint can be used. Part (a) shows an existing one-to-one correspondence between intervals on the left and right scanlines. This denotes an intermediate state in the progress of a stereo matching and segmentation algorithm. Notice that the intervals may correspond in any order (ie. the ordering constraint is not needed). Now, in part (b), we wish to insert a new pair of corresponding intervals, shown in blue. (This new pair of matching intervals improves upon the existing matches according to some energy metric which depends on the stereo algorithm being used). In part (c), we see that the insertion of this pair of intervals conflicts with existing intervals (shown in red). In order to enforce uniqueness, the red pair of intervals on the right must be removed, while the red pair of intervals on the left must be resized. In part (d), we see the new correspondences. The interval pair which was resized is shown in green, and the

newly inserted pair is shown in blue.

Figure 3.5 shows the idea of interval mapping and occlusions detection using a real example. In this figure, we see how intervals of unequal length can correspond uniquely to yield occlusions.

Conclusion: There exists a one-to-one mapping between intervals (possibly having unequal lengths), and not between pixels.

3.1.4 An algorithm to deal with horizontal slant

We now describe a simple scanline algorithm which implements the ideas presented above; this algorithm also uses the concept of connectivity maximization presented in Chapter 2 along scanlines instead of the whole image, and simultaneously searches the space of possible disparities and horizontal slants. It processes a pair of scanlines $I_L(x)$ and $I_R(x)$ at a time without using any vertical consistency constraints. Horizontal disparities $\Delta_L(x)$ are assigned to the left scanline within a given range $[\Delta_1, \Delta_2]$, and $\Delta_R(x)$ to the right scanline in the range $[-\Delta_2, -\Delta_1]$. The disparities are not assigned to pixels, but continuously over the whole scanline. The disparities are not directly estimated, but instead, we search for functions $m_L(x)$ and $d_L(x)$ for the left scanline, and $m_R(x)$ and $d_R(x)$ for the right scanline, such that given a point x_L on the left scanline, its corresponding point x_R in the right scanline would be

$$x_R = m_L(x_L) \cdot x_L + d_L(x_L) \tag{3.3}$$

and reciprocally:

$$x_L = m_R(x_R) \cdot x_R + d_R(x_R)$$

Clearly,

$$\begin{aligned} m_R(x_R) &= 1/m_L(x_L) \\ d_R(x_R) &= -d_L(x_L)/m_L(x_L) \end{aligned}$$

The disparities are then computed as:

$$\begin{aligned} \Delta_L(x_L) &= x_R - x_L = (m_L(x_L) - 1) \cdot x_L + d_L(x_L) \\ \Delta_R(x_R) &= x_L - x_R = (m_R(x_R) - 1) \cdot x_R + d_R(x_R) \end{aligned} \tag{3.4}$$

The functions m_L and m_R are the horizontal slants, which allow line segments of differ-

ent length in the two scanlines to correspond. The scanlines are represented continuously by linearly interpolating intensities between pixel locations. Thus, if $m_L = 2$, then the left scanline is stretched (resampled) by a factor of 2, and then matched with the unstretched right scanline using the Birchfield-Tomasi method. Due to the stretching of one scanline before performing the intensity based matching, we are automatically modifying the traditional Birchfield-Tomasi method to properly deal with horizontal slant. For each possible m_L and d_L, absolute intensity differences between corresponding points are computed, and thresholded by a threshold t. The best value of m_L and d_L for a point is chosen such that it maximizes the size of the matching line segment containing that point (ie. the maximum connectivity approach of Chapter 2 applied to one dimension).

The values of the horizontal slant which are to be examined are provided as inputs, ie. $m_L, m_R \in M$, where $M = \{m_1, m_2..., m_k\}$, such that $m_1, m_2, ..., m_k \geq 1$. Since $m_L = 1/m_R$, when we try a value $m_L = m_i$, we are implicitly trying $m_R = 1/m_i$. Hence, the specified set of slants M contains only values greater than 1, since the reciprocal values are implicitly tried. The disparity search range $[\Delta_1, \Delta_2]$ is also provided as an input. In order to find the occlusions, we enforce the uniqueness constraint in its modified form as shown in Figure 3.3. We maintain a one-to-one correspondence between intervals in the two scanlines. Hence, at any stage of the process, we have a set S_L of non-overlapping intervals in the left scanline and a set S_R of non-overlapping intervals in the right scanline. An interval i is of the form $[x_1, x_2)$. The uniqueness constraint enforces a one-to-one mapping U between the elements of S_L and the elements of S_R. When a new corresponding pair of intervals i_L and i_R is found, the segment previously corresponding to i_L is removed if present, and the same is done for i_R. Then, i_L is added to S_L, and i_R to S_R, and the one-to-one mapping in U is updated. Thus, we always ensure that a line segment in the left scanline uniquely maps to a line segment in the right scanline. In the end, line segments which remain unmapped are the occlusions. In our implementation, we have used hash-tables to maintain the interval information and detect overlaps. The skeleton of the algorithm is shown below.

1. For all $m_L \in M$, $\Delta_L \in [\Delta_1, \Delta_2]$, do

   (a) stretch I_L by m_L to get I'_L

      i. find range for d_L using given range for Δ_L and eqn. 3.4

ii. for every d_L, match I'_L and I_R and find connected matching segments and their sizes; update correspondence map while enforcing the uniqueness constraint.

2. For all $m_R \in M$, $\Delta_R \in [-\Delta_2, -\Delta_1]$, do

   (a) stretch I_R by m_R to get I'_R

      i. find range for d_R using given range for Δ_R and eqn. 3.4

      ii. for every d_R, match I'_R and I_L and find connected matching segments and their sizes; update correspondence map while enforcing the uniqueness constraint

3. $m_L = m_R = 1$

   (a) for every $d_L \in [\Delta_1, \Delta_2]$, match I_R and I_L and find connected matching segments and their sizes; update correspondence map while enforcing the uniqueness constraint

3.2 Vertical slant

3.2.1 Fundamental differences between vertical and horizontal slant

Assume that we are given a rectified stereo pair of images. Due to the discrete (pixelized) nature of the images, changes in disparity as we move from left to right along a scanline may be caused by one of two factors:

- There exists a depth discontinuity (as seen in Figure 3.6(a)), or
- The pixels form a part of a horizontally slanted surface (Figure 3.6(b)).

In this case, we can distinguish between these two possibilities because only a true depth discontinuity will cause an occlusion to appear (as seen in Figure 3.6(a)). However, if we move vertically and find a disparity change (as seen in Figure 3.6(c)), we have no way of distinguishing whether the vertical change is caused by a discontinuity or by a vertically slanted surface, since neither causes occlusions to appear. Thus, there is a fundamental difference between horizontal and vertical slant.

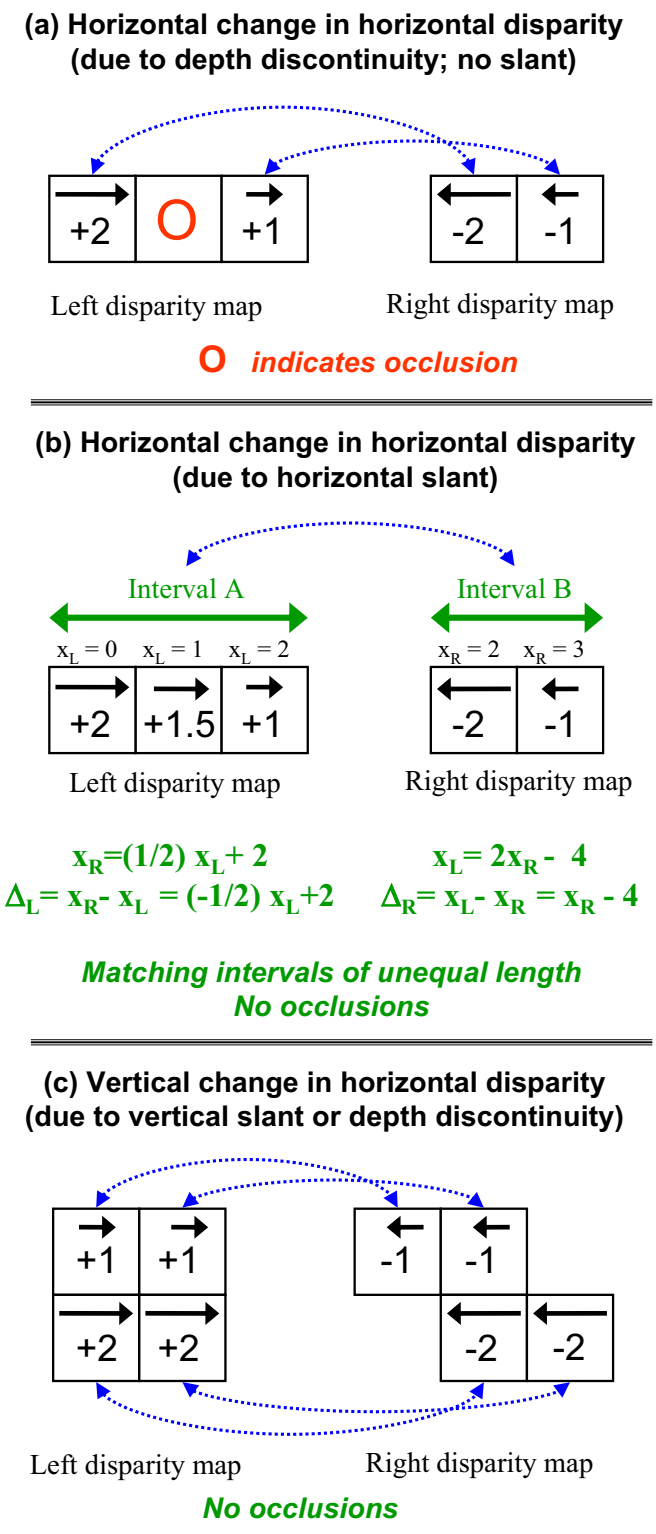

Figure 3.6: Top: Horizontal changes in horizontal disparity due to a discontinuity create an occlusion. Middle: Horizontal changes in horizontal disparity due to horizontal slant lead to stretching/shrinking but no occlusions. Bottom: Vertical changes in horizontal disparity due to discontinuity and vertical slant cannot be distinguished (since no occlusions occur in either case).

3.2.2 Vertical connectivity and non-horizontal edges

Since we cannot distinguish whether purely vertical changes in disparity are due to a true discontinuity or due to a vertically slanted surface, additional assumptions must be employed if we are to enforce any vertical consistency constraints on the disparity map. Consider this example: in the image, if we have a horizontal intensity edge (a horizontal line), we have two possibilities (a) this edge corresponds to a depth discontinuity, and pixels separated by it do not lie on the same surface, or (b) the edge is just an intensity edge, and pixels separated by it lie on the same surface. If we commit the error of assuming that the pixels separated by the horizontal edge are connected and it happens to be a discontinuity, our solution will yield a disparity map without the depth discontinuity, which is clearly incorrect. It is safer to assume that such pixels are *not* connected, to allow the possibility that there may be a depth discontinuity.

Therefore, vertical neighbors separated by a horizontal edge or no edge at all should not be connected.

Also, as shown in Figure 3.7, if we have two images of a single-colored object, and we assume vertical connections in the interior, then we will get a single disparity in the entire interior (when maximum overlap of the images takes place) instead of a vertical gradient. Thus, we cannot assume that disparity is vertically constant even if two vertical neighbors have the same color/intensity. *Disparity can change even when there is no change in color or intensity.* (Note that we *can* assume that disparity is *continuous*, but not necessarily *constant*, if intensity or color do not vary in a region.)

However, if we have a non-horizontal edge running across the image, it will cause occlusions to appear if it is a discontinuity, and no occlusions will appear if both sides of it lie on the same surface. This distinguishing ability allows us to make the assumption that:

Vertical neighbors lying on non-horizontal edges should be connected (Figure 3.8).

3.2.3 Cue integration along the vertical direction

We have examined the differences in the character of vertical and horizontal slant in the previous sections. It is clear that if there are vertical changes in the horizontal disparity, we cannot distinguish whether we have a discontinuity or a vertically slanted surface. Hence, the computation of smooth vertical slant may be performed after disparities have

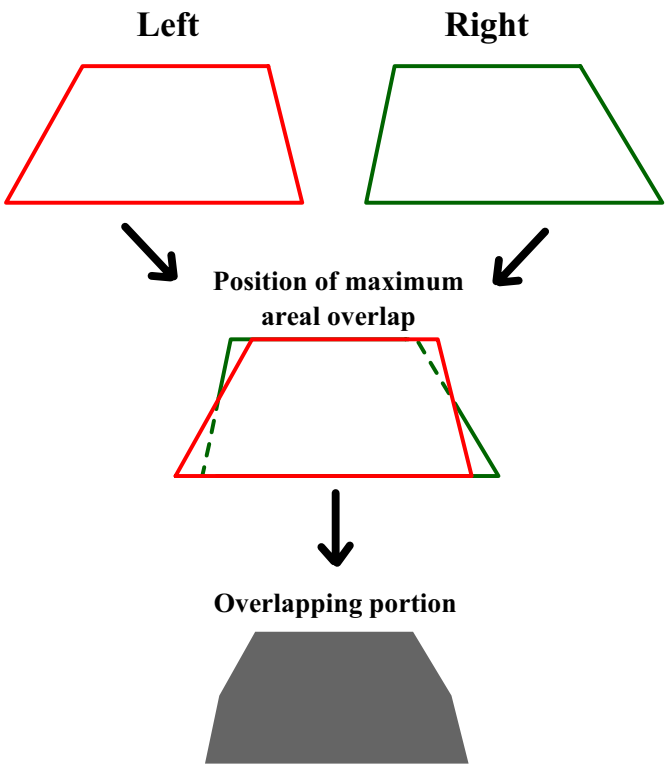

Figure 3.7: Top: images of a vertically slanted plane. Middle: images overlaid to maximize overlap. Bottom: area of largest overlap

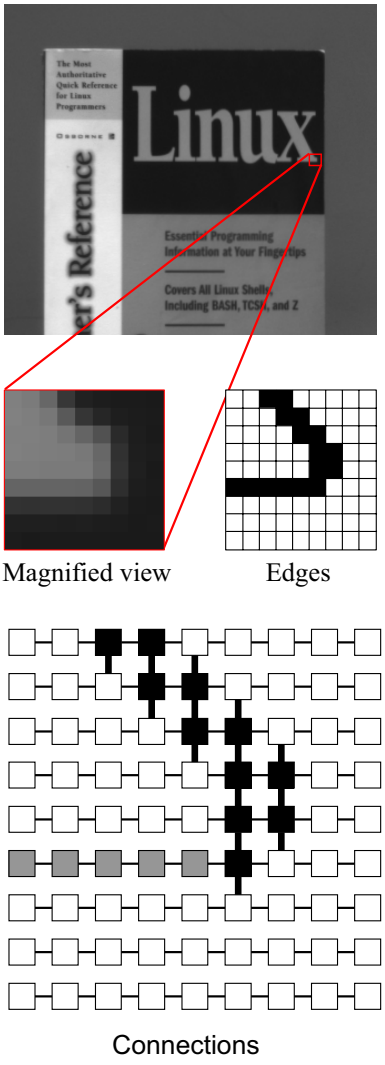

Figure 3.8: Vertical connections between pixels are established only along non-horizontal edges

been computed by a smoothing or fitting process. Another promising avenue for computing smooth vertical shape is by using edge based methods which rely on orientation disparities and contour continuity of non-horizontal edges [LZ03]. The resulting method would be a combination of area based and edge based methods. In this situation, it is also conceivable that inputs from 'Shape from X' modules (eg. shape from texture, shape from shading) are critical to establish vertical consistency and construct vertically smooth models of the scene shape and structure. We believe that such other cues may strongly influence the estimation of vertical slant in the human visual system, although not so much the horizontal slant. There exists some support for this idea in studies dealing with the perception of slanted surfaces by humans, which conclude that there is an anisotropy in the perception of stereoscopic slant [RG83, MM90, GR92, CR93, RG94], ie. a horizontally slanted surface and a vertically slanted surface having the same slant are perceived differently. If shape from disparity is being integrated with other cues such as shape from texture more strongly in the vertical direction than the horizontal, such an anisotropy in slant perception may arise.

3.3 Revisiting the principle of minimum segmentation

Let us return once more to the principle of minimum segmentation first introduced in Chapter 2. When we compute the depth map of a real scene, the desirable property of such a depth map is that it should explain the observed images while minimizing the number of discontinuities. In other words, we would like to model the depth map as a *piecewise continuous* function which is consistent with the observed images and has the minimum possible number of pieces. This *principle of minimum segmentation* implies that we desire each segment to have the maximum possible size, which is consistent with the ideas on connectivity maximization presented previously.

To simplify the problem computationally, we often choose more restrictive versions of the general model of a piecewise continuous depth map. We have already discussed the *piecewise constant* approximation and the *piecewise linear* (ie. planar) approximation. We can proceed in this manner towards more complex models of the depth and shape, such as quadratic and cubic models, in an attempt to get closer to the true property of piecewise continuity. However, these models progressively increase the dimensionality of the search

space.

3.4 Shape and motion: problems with optical flow

We have discussed the effects of vertical and horizontal slant on the stereo correspondence problem. Firstly, we have seen why disparity continuity constraints cannot be enforced across horizontal edges, since we do not know *a priori* if the regions on either side of the horizontal edge constitute a single smooth surface. In order to generalize this idea to optical flow, we must closely examine the definition of a horizontal edge in the case of the stereo problem. As seen in Figure 3.9(a), a horizontal edge is an edge where the disparity change across the edge is parallel to the edge. Figure 3.9(b) shows a vertical edge where the disparity change is perpendicular to the edge. Note that the definition of a horizontal or vertical edge is actually related to the *flow difference* across it, rather than the flow itself.

But in the case of stereo problem, the disparity difference is always parallel to the disparity itself (since both are horizontal).

Hence, for the case of stereo, we can define the edge relative to the direction of the disparity instead of the disparity difference. But as shown in Figure 3.9(c), in the case of optical flow, *the flow difference is parallel to the edge but the direction of the flow itself is not.*

In the case of stereo, we severed connections across horizontal edges. But for the case of flow, if we are trying a candidate shift (δ_x, δ_y), since the flow difference may be unrelated to this direction, we cannot be certain which connections to sever. Thus, there is an additional ambiguity in the case of optical flow which was not present for the case of stereo.

Figure 3.9(d) shows a shear in the flow parallel to the edge, which may be due to vertical slant or a vertical discontinuity. Figure 3.9(e) shows a change in the flow component perpendicular to the edge, which may be due to horizontal slant or a horizontal discontinuity, distinguishable by the presence of occlusion. Thus, although the effects of slant are identical to the stereo case, distinguishing slant from discontinuities is more difficult in the case of optical flow.

3.5 Experimental results

We have shown in the previous sections that horizontal and vertical slant play a critical role in the estimation of correspondence and occlusions. The first row of Figure 3.10 shows a

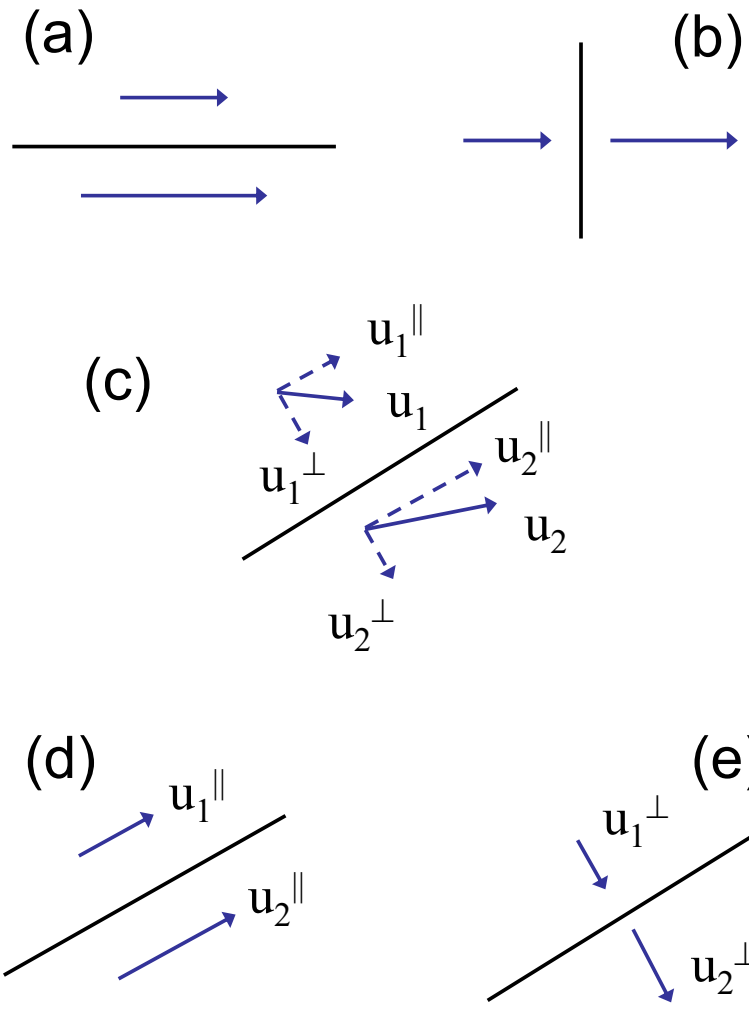

Figure 3.9: Stereo: disparity difference on two sides of (a) horizontal edge, and (b) vertical edge. Optical flow: (c) components of optical flow parallel and perpendicular to an edge. (d) change in parallel flow component across the edge. (e) change in perpendicular flow component across the edge.

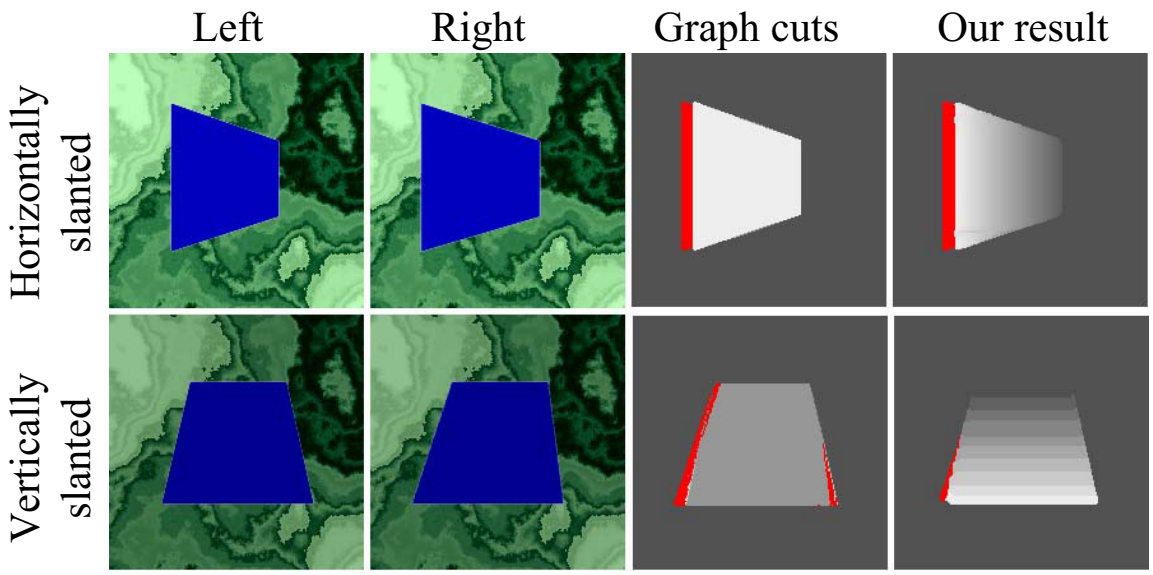

Figure 3.10: Columns 1 to 4: Left image, right image, graph cuts result for the left disparity map, our result for left disparity map. Row 1: horizontally slanted object, Row 2: vertically slanted object. Occlusions are shown in red.

stereo pair of images in which the blue object is horizontally slanted (ie. depth varies from left to right), and the second row shows a stereo pair in which the blue object is vertically slanted. The third column of this figure shows the results of the graph cuts [KZ01] algorithm, while the fourth column shows our results for each of these stereo pairs. In these results, occlusions are shown in red. The graph cuts result was obtained using software kindly provided by the authors (*www.cs.cornell.edu/People/vnk/software.html*). It is clear that for both these stereo pairs, the graph cuts result gives a constant disparity for the blue object, while our result correctly shows the slant and still finds the occlusions.

We expect that the constraints presented above will improve the results of many existing dense stereo algorithms in both qualitative and quantitative ways. However, for the sake of completeness, we compare our results with other algorithms using the test procedure created by Scharstein and Szeliski [SS02] available at *www.middlebury.edu/stereo.* They compare the disparity map d_{out} generated by an algorithm to the true disparity d_{true}, and the pixels which deviate by more than 1 unit from their true disparity are labeled as 'bad' pixels. The percentage of bad pixels in the entire image, in the untextured regions and near depth discontinuities are used to compare the results of various algorithms. The percentages of bad pixels are reported in Table 3.1, which was generated by submitting our disparity maps (Figure 3.12) to the web-based evaluation program created by Scharstein

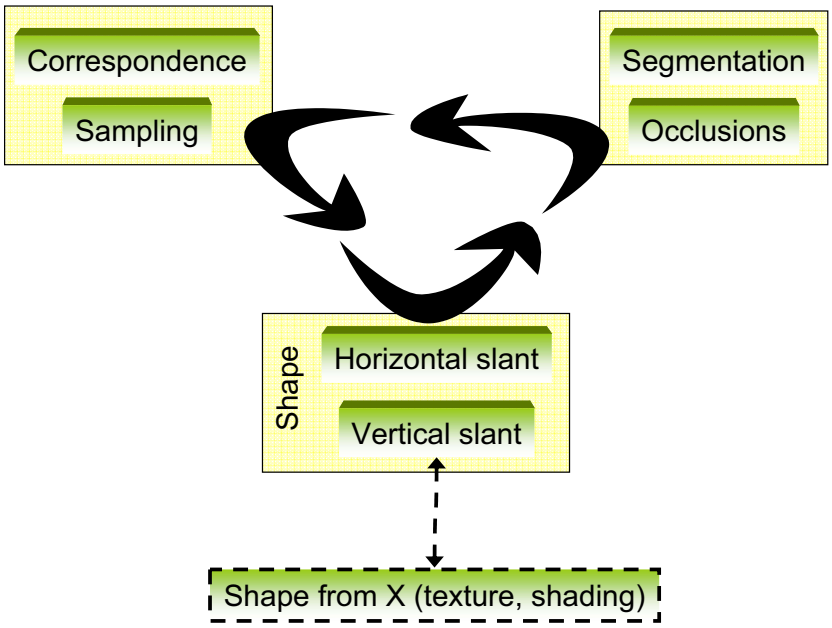

Figure 3.11: Visual problems such as correspondence, depth segmentation, and shape estimation must be solved simultaneously

and Szeliski. Our algorithm ('connectivity-slant') ranks sixth overall, while the ranks in each column are showed in brackets, below the error percentages. For the bottom left of the Venus sequence, it is not possible to assign correct disparities, since the corresponding points in the second image lie outside the image. Scharstein and Szeliski exclude a ten pixel boundary before evaluation, but it is not adequate to remedy this situation (a twenty pixel left boundary will suffice).

Figure 3.13 shows the results for two more stereo pairs: the tree branch pair and the corridor pair. The tree branch illustrates the ability of the algorithm to correctly handle thin overlapping objects. The corridor scene contains many untextured surfaces which are strongly slanted and specular. Note the correctness of the results for the walls, and especially for the left wall, which has a very large slant.

We have analyzed the effects of shape in establishing dense point correspondence. As shown in Figure 3.11, we have shown that correspondence, segmentation, occlusion detection, and shape estimation influence each other and have to be solved in concert, with other modalities such as 'Shape from X' or orientation disparity possibly influencing the computation of vertical shape.

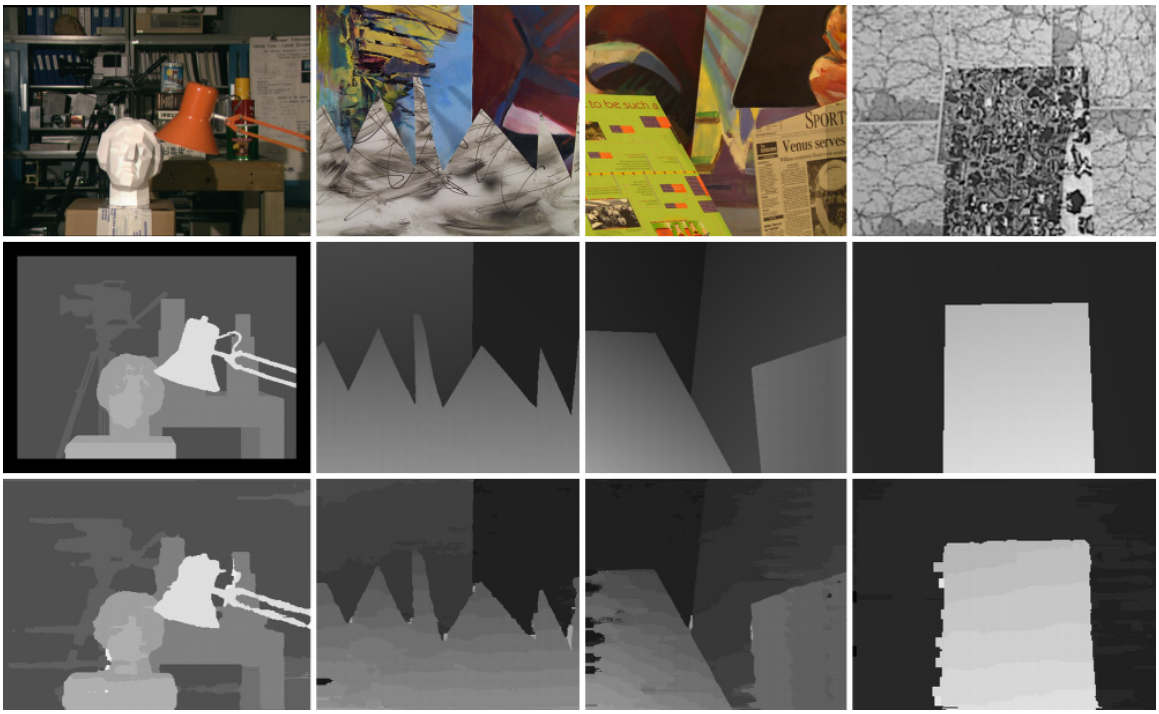

Figure 3.12: Top row (Left frames), Middle row (ground truth), Bottom row (our results). Occlusions were filled in as required by the evaluation procedure.

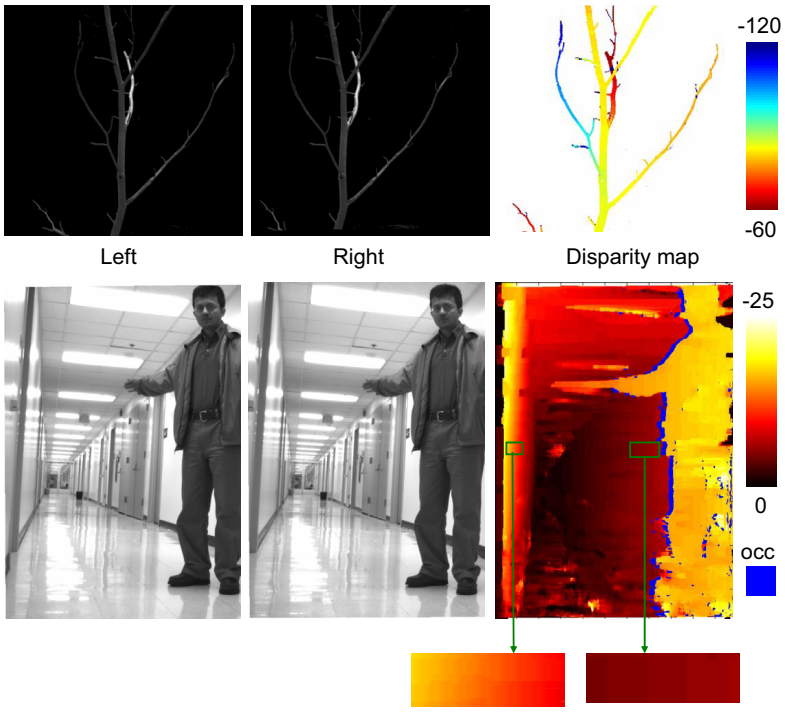

Figure 3.13: Top row: tree stereo pair and disparity map. Bottom row: Corridor stereo pair with the disparity map and occlusions (blue regions). Note the disparity variation for the left and right walls of the corridor.

Table 3.1: Performance comparison from the Middlebury Stereo Vision Page (overall rank is 6′th among 28 algorithms). The table shows only the top ten algorithms. Error percentages and rank (in brackets) in each column is shown.

Rank	Algorithm	Tsukuba			Sawtooth			Venus			Map	
		all	untex	disc.	**all**	untex	disc.	**all**	untex	disc.	**all**	disc.
1	Layered	1.58 (4)	1.06 (7)	8.82 (5)	0.34 (1)	0.00 (1)	3.35 (1)	1.52 (8)	2.96 (17)	2.62 (2)	0.37 (11)	5.24 (11)
2	Belief prop	1.15 (1)	0.42 (2)	6.31 (1)	0.98 (8)	0.30 (13)	4.83 (5)	1.00 (4)	0.76 (4)	9.13 (12)	0.84 (18)	5.27 (12)
3	Multcam GC	1.85 (8)	1.94 (13)	6.99 (4)	0.62 (6)	0.00 (1)	6.86 (10)	1.21 (6)	1.96 (8)	5.71 (6)	0.31 (8)	4.34 (10)
4	GC+occl.	1.19 (2)	0.23 (1)	6.71 (2)	0.73 (7)	0.11 (7)	5.71 (8)	1.64 (11)	2.75 (15)	5.41 (5)	0.61 (14)	6.05 (13)
5	Impr.Coop.	1.67 (5)	0.77 (4)	9.67 (9)	1.21 (12)	0.17 (10)	6.90 (11)	1.04 (5)	1.07 (5)	13.68 (17)	0.29 (6)	3.65 (7)
→ ***6***	***connectivity -slant***	**1.77 (6)**	**0.95 (5)**	**9.48 (7)**	**0.61 (4)**	**0.17 (11)**	**5.05 (6)**	**3.00 (20)**	**5.22 (20)**	**7.63 (8)**	**0.21 (2)**	**3.01 (4)**
7	GC+occl.	1.27 (3)	0.43 (3)	6.90 (3)	0.36 (2)	0.00 (1)	3.65 (2)	2.79 (19)	5.39 (21)	2.54 (1)	1.79 (21)	10.08 (20)
8	Disc. pres.	1.78 (7)	1.22 (9)	9.71 (10)	1.17 (10)	0.08 (6)	5.55 (7)	1.61 (10)	2.25 (11)	9.06 (11)	0.32 (9)	3.33 (6)
9	Graph cuts	1.94 (10)	1.09 (8)	9.49 (8)	1.30 (14)	0.06 (5)	6.34 (9)	1.79 (14)	2.61 (14)	6.91 (7)	0.31 (7)	3.88 (8)
10	Symbiotic	2.87 (13)	1.71 (11)	11.90 (11)	1.04 (9)	0.13 (8)	7.32 (13)	0.51 (2)	0.23 (2)	7.88 (9)	0.50 (13)	6.54 (14)
	⇩				⇩						⇩	
28	Max. surf.	11.10 (28)	10.70 (26)	41.99 (28)	5.51 (28)	5.56 (28)	27.39 (27)	4.36 (23)	4.78 (19)	41.13 (27)	4.17 (27)	27.88 (27)

Chapter 4

Connectivity and Diffusion

In chapter 2, we have seen how the connectivity between pixels proves to be an effective measure for deciding the assignment of correspondences. We have presented an algorithm which uses connected components labeling on thresholded absolute intensity difference images. The problem lies in the use of hard thresholds on the absolute intensity differences; changing the threshold by small amounts can alter the connectivity of pixels and change the connected components labeling significantly. In this chapter, we explore an alternate definition of connectivity, based on diffusion, which operates directly on the intensity differences, without binarizing them first with a threshold. We also show how this new representation of connectivity, which is more general, reduces to the connected components model from the earlier chapter.

4.1 Connectivity becomes diffusion

Let us recap our approach to finding the best correspondence. Given two images I_1 and I_2, we seek to assign disparities or flows to the pixels in these images. Candidate disparities or flows are assigned from a given set S of candidates. For each shift $\vec{\delta}$ in the set S, we find the absolute intensity difference

$$\Delta I_{\vec{\delta}}(\vec{x}) = \left| I_1(\vec{x}) - I_2(\vec{x} - \vec{\delta}) \right| \tag{4.1}$$

and decide which pixels are matching by thresholding with a value t. We then build connected components on these binary images, and for each pixel, the best shift $\vec{\delta}$ is the one which maximizes the size of the connected component containing it. Thus, the size of the connected component is the measure which we use to decide the assignment of correspondences. Let us then ask ourselves: *what exactly is the size of the connected component measuring?*

To answer this question, let us take a closer look at the matching process. For a given shift $\vec{\delta}$, we are using the absolute intensity differences and thresholding to create a binary

image which says whether a pair of pixels from the two images $I_1(\vec{x})$ and $I_2(\vec{x}-\vec{\delta})$ matches (indicated by the binary value 1) or not (indicated by the binary value 0). Then, on this binary image, we are performing connected components labeling and for each pixel, we find the size of the connected component containing it. Equivalently, this means that if a pixel p and a pixel q both have the label 1, and if there exists a path of matching pixels connecting pixel p to pixel q, then pixel p contributes the value 1 to the pixel q and vice-versa. Thus, a pixel p in the image which matches for shift $\vec{\delta}$ supports the assignment of this shift $\vec{\delta}$ to all other matching pixels in the image which are connected to it. This *influence* of the matching pixel p on other pixels can be thought of as being propagated by surrounding matching pixels until all possible pixels have been reached. Thus, for any pixel p, the connected component size is equivalent to the sum of the *influence* of all other matching pixels in the image on it, after *diffusing through* the intermediate pixels.

Connected components are therefore a special case of diffusion of information from one pixel to another. In the more general case, we can use diffusion to facilitate the *conduction* of information from one pixel to another, where the information being propagated is not necessarily binary as in the case of the connected components (*match* or *no match*). This generalization to diffusion allows us to use a non-binary continuous measure of matching (such as the absolute intensity difference values themselves *without thresholding*).

4.2 Fast diffusion along scanlines

In this section, we examine in greater detail the generalization of connected components to diffusion by examining the one dimensional problem of matching pixels in two corresponding scanlines from a stereo pair of images. The one dimensional problem allows us to directly establish the link between diffusion and connected components using examples.

Given two scanlines $I_1(x)$ and $I_2(x)$, assume that we wish to compute a goodness measure $G(x, \delta)$ for each pixel x and each shift δ, such that for each x, we can choose the shift δ which maximizes $G(x, \delta)$. First, we need a method of performing local matching, so that we can extract a local measure $M(x, \delta)$ which tells us how well pixels $I_1(x)$ and $I_2(x - \delta)$ match. This measure $M(x, \delta)$ may be $I_1(x)I_2(x - \delta)$ (correlation), $|I_1(x) - I_2(x - \delta)| < t$ (thresholded absolute differences), phase differences, or any suitable local measure. We must also define a conductivity $C(x, \delta)$ at each pixel, which allows us to propagate the

influence of a pixel to its neighbors in the following manner:

1. A pixel is influenced by pixels on its left (via G_{Left}) and pixels on its right (via G_{Right}).

$$G(x,\delta) = G_{Left}(x,\delta) + G_{Right}(x,\delta) - M(x,\delta) \tag{4.2}$$

2. From the left to the right, we compute G_{Left} as follows:

$$G_{Left}(x,\delta) = G_{Left}(x-1,\delta)C(x,\delta) + M(x,\delta) \tag{4.3}$$

3. Similarly, from the right to the left,

$$G_{Right}(x,\delta) = G_{Right}(x+1,\delta)C(x,\delta) + M(x,\delta) \tag{4.4}$$

Note that in $G(x,\delta)$, we subtract $M(x,\delta)$ to avoid counting it twice. Note that like $M(x,\delta)$, the conductivity $C(x,\delta)$ is also a local metric.

Figure 4.1 shows that this diffusion process reduces to the connected components matching by choosing:

$$M(x,\delta) = |I_1(x) - I_2(x-\delta)| < t \quad ; \quad C(x,\delta) = M(x,\delta) \tag{4.5}$$

The top of the figure shows the absolute intensity differences for some δ. The left hand side shows how $G(x,\delta)$ is computed using the diffusion process, and the right hand side shows how the same result is obtained using the connected components labeling (In the connected components approach, $G(x,\delta)$ is the size of the connected component).

In the one dimensional case, diffusion is able to improve the robustness of the matching process, without any change in the computational efficiency. In the two dimensional case, such an efficient non-iterative metamorphosis of connected components into diffusion has not been discovered, although restricted definitions of connectivity permit efficient extensions.

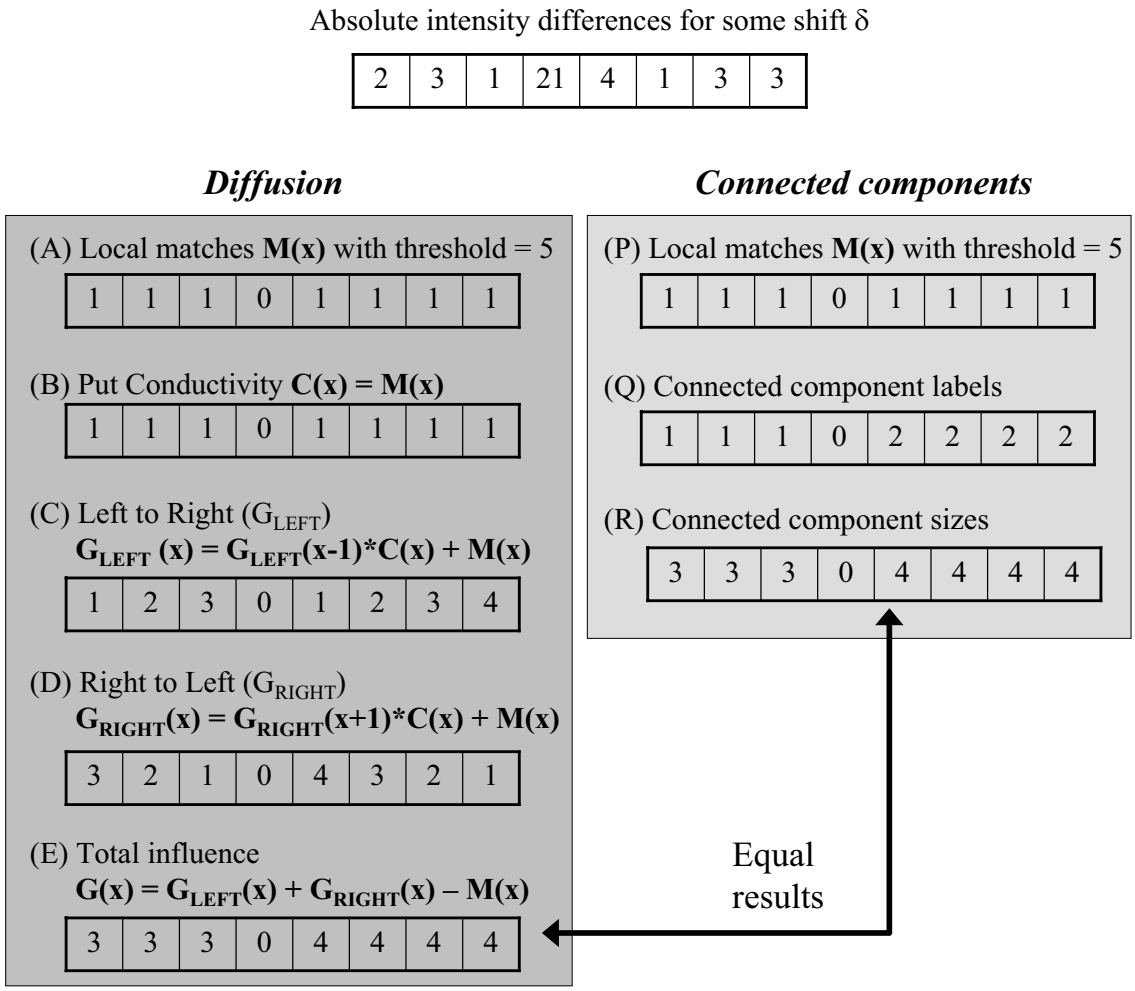

Figure 4.1: Connected components is a special case of diffusion. On the left (A to E), we show how a measure of the influence of matching pixels on each other is computed using diffusion. On the right, we see how the same measure is obtained using connected component sizes. Note that $M(x)$ is used to denote $M(x, \delta)$, $C(x)$ for $C(x, \delta)$, and so on, for the sake of brevity.

4.3 An algorithm for matching by using diffuse connectivity

In this section, we shall describe a stereo matching algorithm based on the notion of diffuse connectivity outlined in the previous section. The input to the algorithm consists of two image scanlines $I_1(x)$ and $I_2(x)$ and a set S of possible shifts.

Algorithm DiffuseMatching (I_1, I_2, S) returns shifts d

1. for each shift $\delta \in S$, do

   (a) $M(x, \delta) = m(I_1(x), I_2(x - \delta))$ where m is a local matching function

   (b) $C(x, \delta) = c(I_1(x), I_2(x - \delta))$ where c is a local function taking on values from 0 to 1

   (c) From left to right, find $G_{left}(x, \delta)$ using $G_{Left}(x, \delta) = G_{Left}(x - 1, \delta)C(x, \delta) + M(x, \delta)$

   (d) From right to left find $G_{right}(x, \delta)$ using $G_{Right}(x, \delta) = G_{Right}(x + 1, \delta)C(x, \delta) + M(x, \delta)$

   (e) Compute $G(x, \delta) = G_{Left}(x, \delta) + G_{Right}(x, \delta) - M(x, \delta)$

2. For each pixel x, find $d(x) = \underset{\delta}{argmax}[G(x, \delta)]$

In the actual implementation, step 2 of the above algorithm also implements the uniqueness constraint, which enforces a one-to-one matching between pixels in the two scanlines, in order to find occlusions.

In the experimental results shown in the next section, we use the functions

$$m(I_1(x), I_2(x - \delta)) = c(I_1(x), I_2(x - \delta)) = exp\left(-\lambda \left| I_1(x) - I_2(x - \delta) \right|\right) \tag{4.6}$$

The absolute intensity differences are normalized to lie in the range $[0, 1]$. The parameter λ can be chosen to alter the smoothing effect of the diffusion. It is also possible to use a sigmoid function which can mimic the effect of thresholding the intensity differences. In a later chapter on contrast invariance, we shall develop local metrics which are invariant to the contrast of the images.

Table 4.1: Performance of scanline diffusion. Numbers indicate percentage of bad pixels overall, in untextured regions and at discontinuities. Numbers in brackets indicate rank in each column (overall rank is 14).

Algorithm	Tsukuba			Sawtooth			Venus			Map	
	all	untex	disc.	**all**	untex	disc.	**all**	untex	disc.	**all**	disc.
Scanline Diffusion	2.13 (10)	1.52 (9)	10.57 (10)	1.11 (9)	1.27 (19)	8.68 (15)	3.31 (20)	5.99 (21)	12.81 (13)	0.26 (4)	3.65 (6)

4.4 Experimental results

Figure 4.2 shows the disparity maps for four standard test sequences (obtained from the website *www.middlebury.edu/stereo*) computed with the parameter value $\lambda = 20$ used in equation 4.6. In the test procedure devised by Scharstein and Szeliski [SS02], the disparity map d_{out} generated by an algorithm is compared to the true disparity d_{true}, and the pixels which deviate by more than 1 unit from their true disparity are labeled as 'bad' pixels. The percentages of bad pixels in the entire image, in the untextured regions and near depth discontinuities are used as a measure of performance of an algorithm. In Table 4.1, we provide the error percentages for the results shown in Figure 4.2.

The connected components algorithm presented in Chapter 2 seems to perform slightly better than the scanline diffusion algorithm with the above choice of matching functions, however, from an application standpoint, the diffusion algorithm has the advantage of being able to work with a broader class of continuous matching metrics. Moreover, it's results are not sensitive to the choice of the input parameter λ, for this particular choice of matching functions.

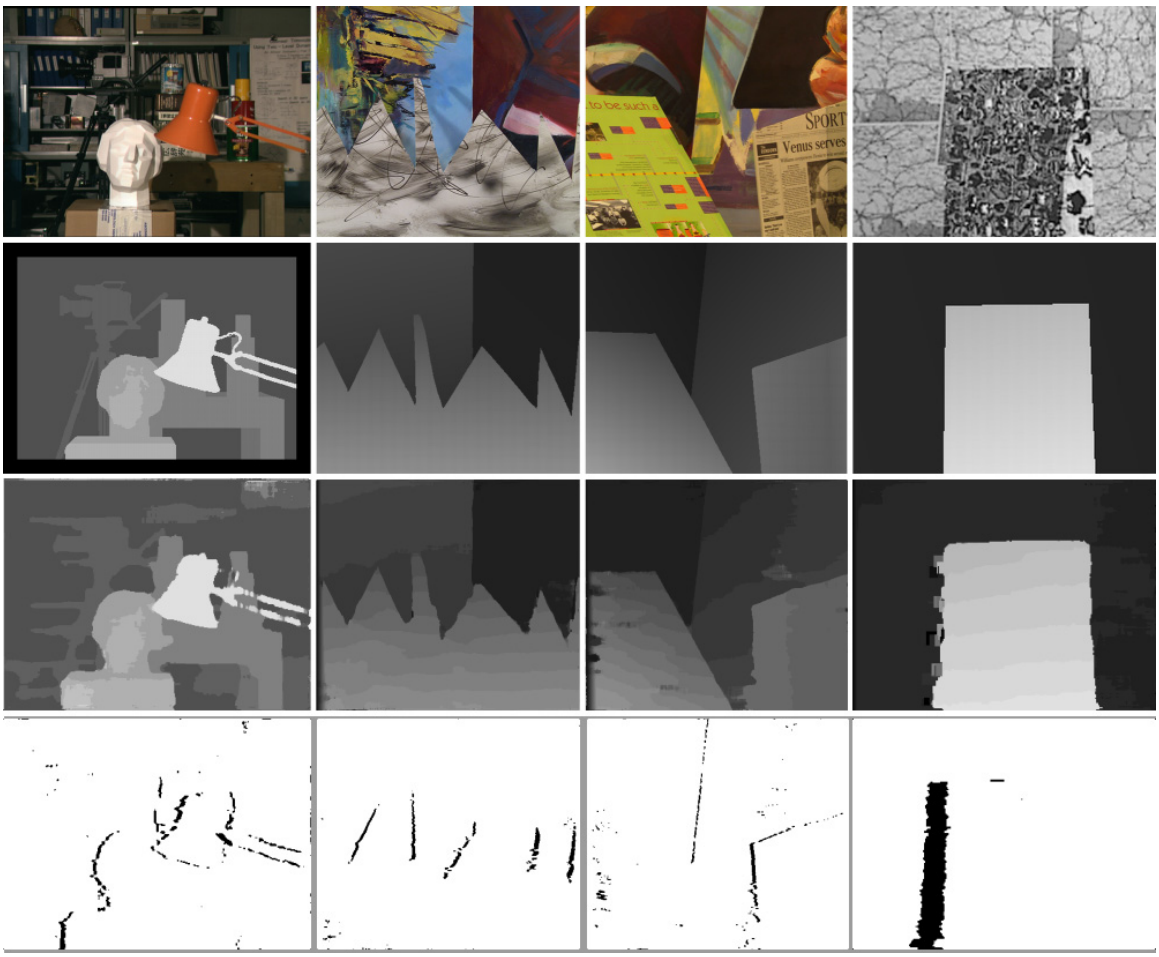

Figure 4.2: Top row: left images from four stereo pairs *tsukuba, sawtooth, venus and map.* Second row: true disparity maps. Third row: results of scanline diffusion with occlusions filled in as required by the Scharstein and Szeliski evaluation. Bottom row: the detected occlusions are shown separately.

Chapter 5

Contrast Invariance

In the previous chapters, we have used absolute intensity differences as a measure of local similarity between two pixels. Intensity difference based matching metrics are popular local matching metrics which are often used inside a global extremization framework with additional smoothness constraints to resolve the best correspondence. The sum of squared intensity differences (SSD) in a small window is another example of such an intensity difference measure. However, any change in the contrast of the images leads to poor or incorrect matching using these difference metrics. (see Scharstein et al [SS02] for a summary of matching metrics commonly used in machine vision).

The intensity of corresponding points in two images obtained from a stereo pair of cameras may not necessarily match even if identical hardware is used; the reasons for this include differences in internal parameters of the cameras and also the specular properties of scene surfaces. The internal camera parameters such as aperture, exposure time and gain are often dynamically adjusted depending on the view, hence different viewpoints can lead to different settings for the two cameras. Due to the difficulty of obtaining and maintaining precise intensity or color calibration between the two cameras, contrast invariance becomes an extremely desirable property of correspondence algorithms.

In this chapter, we shall discuss two approaches which have been developed after a preliminary investigation into the issue of contrast invariance; the first approach is closer in spirit to local metrics commonly used in machine vision, while the second approach is motivated by biological vision.

5.1 Local linear fitting - a first step towards contrast invariance

Given two images, the left image $I_l(x)$ and the right image $I_r(x)$, we seek a local matching function $F(x, d)$, which determines if the function $I_l(x)$ matches $I_r(x + d)$ at the position x. In order to obtain a contrast invariant matching function, let us *locally* allow a linear relationship between the left and right intensities. Hence, if we are examining the pixel at

location x for a candidate disparity d, then we can write

$$\begin{aligned} a\, I_l(x-1) + b\, I_r(x-1+d) + c &= 0 \\ a\, I_l(x) + b\, I_r(x+d) + c &= 0 \\ a\, I_l(x+1) + b\, I_r(x+1+d) + c &= 0 \end{aligned} \tag{5.1}$$

The above linear system can be solved to compute the values of a, b and c upto a constant multiplying factor. Here, a, b and c are local constants whose values are different for various positions and disparities, i.e.

$$\begin{aligned} a &\equiv a(x,d) \\ b &\equiv b(x,d) \\ c &\equiv c(x,d) \end{aligned} \tag{5.2}$$

We choose the constant multiplying factor such that

$$a(x,d)^2 + b(x,d)^2 = 1 \tag{5.3}$$

In order to obtain the local matching at position x for a disparity d, we can use a simple criterion that the numbers $a(x,d)$ and $b(x,d)$ must have opposite sign. This is because if $a(x,d)$ and $b(x,d)$ have the opposite sign, then if I_l increases, I_r will also increase, and vice-versa, which indicates a similarity in the local intensity variations in the two images. Otherwise, if they have the same sign, then an increase in I_l is accompanied by a decrease in I_r and vice-versa, and there is no match. See Figure 5.1 to see a pictoral depiction of the above argument. The following function $S(x,d)$ measures the degree to which the signs of $a(x,d)$ and $b(x,d)$ are opposite:

$$S(x,d) = -a(x,d) \cdot b(x,d) \tag{5.4}$$

Note that we can use equation 5.3 to show that $S(x,d)$ lies in the range $[-0.5, 0.5]$. In this context, it is important to mention that the residual of the linear fit can also be used as

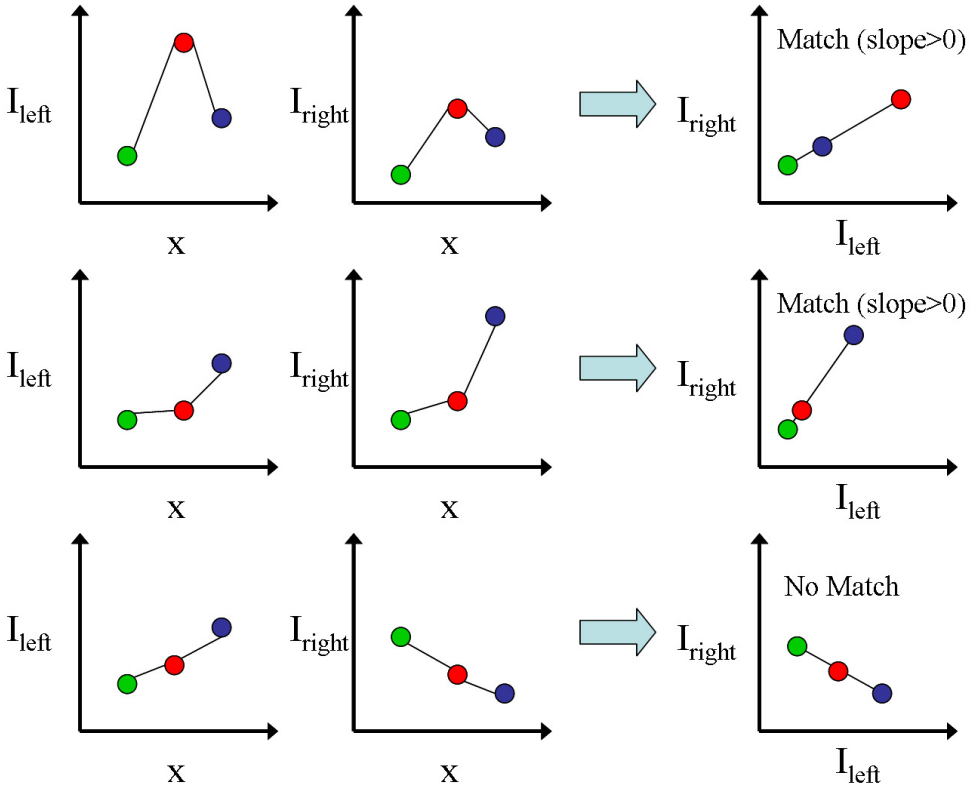

Figure 5.1: Top row: the local intensity around a pixel in the left image and the right image is shown. The plot on the extreme right is an intensity-intensity plot, which is a straight line with positive slope, indicating a match. Middle row: another pair of left and right intensity profiles with similar variations also yields an intensity-intensity plot with positive slope. Bottom row: a mismatched pair of intensity profiles is shown.

a goodness measure for matching, but it is not used since it does not distinguish between the sign of the contrast in the two images (i.e. it will indicate good matching even if the local intensities are anti-correlated).

In chapter 4, we have defined the conductivity $C(x, d)$ for position x and disparity d, such that $C(x, d)$ had values in the range $[0, 1]$. We have also defined a local measure of matching $M(x, d)$, which describes how well the left image $I_l(x)$ matches the shifted right image $I_r(x + d)$. To obtain a contrast invariant matching algorithm, we can choose

$$C(x, d) = 0.5 + S(x, d) \tag{5.5}$$

$$M(x, d) = C(x, d) \tag{5.6}$$

In our implementation, we follow the method of Birchfield and Tomasi [BT98] to allow a pixel's intensity value to vary in a range determined by interpolating its left and right neighbors, and perform a weighted least squares solution to the equation 5.1, instead of an ordinary least squares solution. Figures 5.2 and 5.3 show the results of the above algorithm on a few stereo pairs with different left and right image contrasts.

5.2 Contrast invariance in biological vision

If we observe the stereo pairs given in Figures 5.2, 5.3, 5.4 and 5.5 by fusing the image pairs (by parallel viewing), we can immediately perceive depth, even though the two images have vastly different contrast. It is a well known fact that human observers can still perceive depth even when the contrast of the image seen by one eye is quite different from the contrast of the image seen by the other eye [Jul71]. Hubel and Wiesel [HW68] were the first to describe binocular interactions exhibited by simple cells in the cat's visual cortex, followed by an abundance of literature on physiological investigations into the mechanisms underlying stereopsis (see [FO90] for a review). One of the most interesting results is that Gabor filters can be used to adequately describe the properties of simple cells in the visual cortex [Mar80, Dau85, JP87].

Studies performed by Freeman and his co-workers [FO90, ODF90, DOF91] on the primary visual cortex of the cat revealed [FO90, ODF90] that the response of a binocular simple cell can, to a good approximation, be described by the following equation:

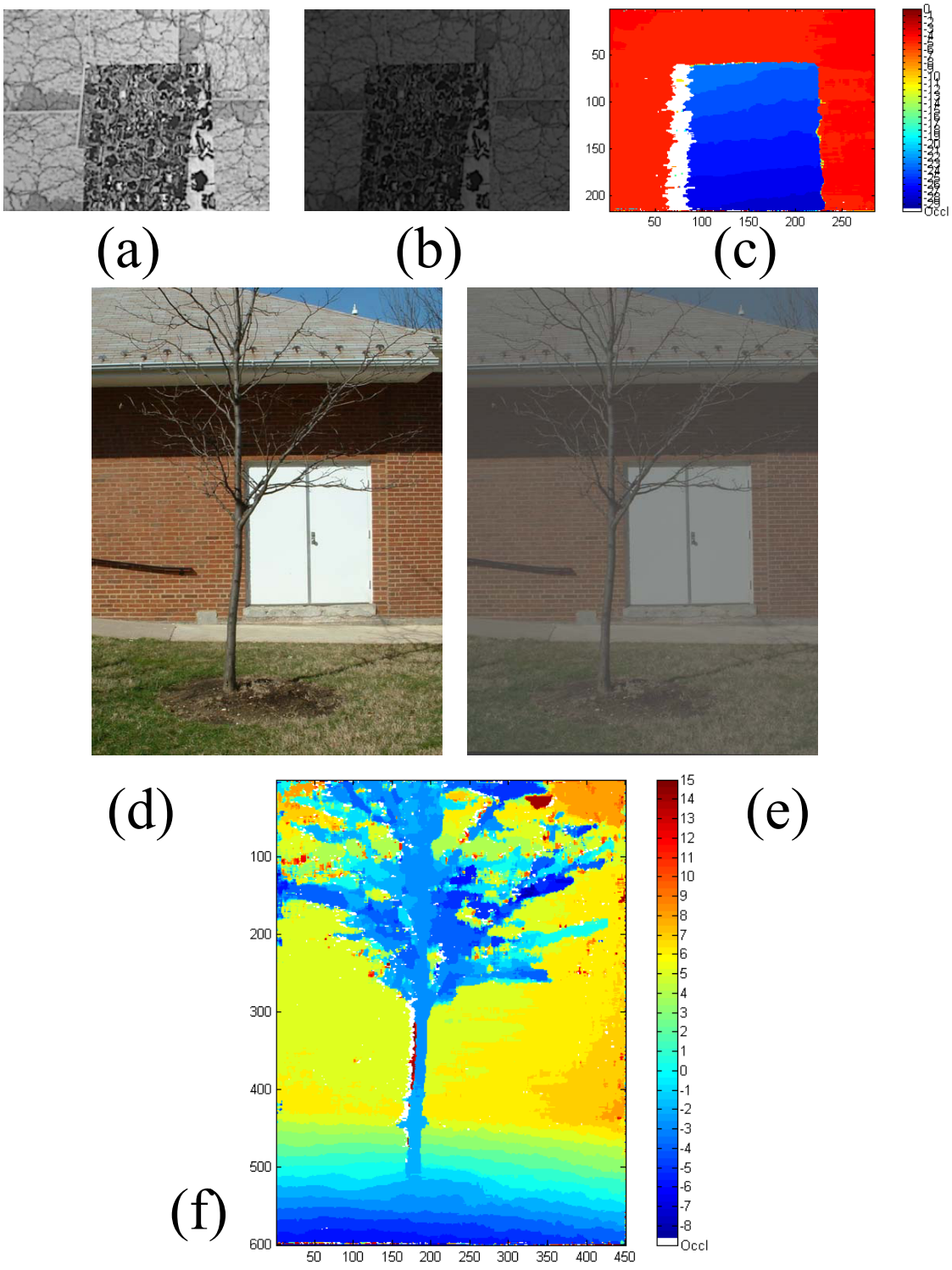

Figure 5.2: (a) and (b) denote a left and right image from the *map* sequence with different contrasts. (c) shows the result of the *linear fit* algorithm. (d) and (e) are a stereo pair of images from the *tree* sequence, and (f) shows the resulting disparity map. Note that occlusions are colored white in the disparity maps.

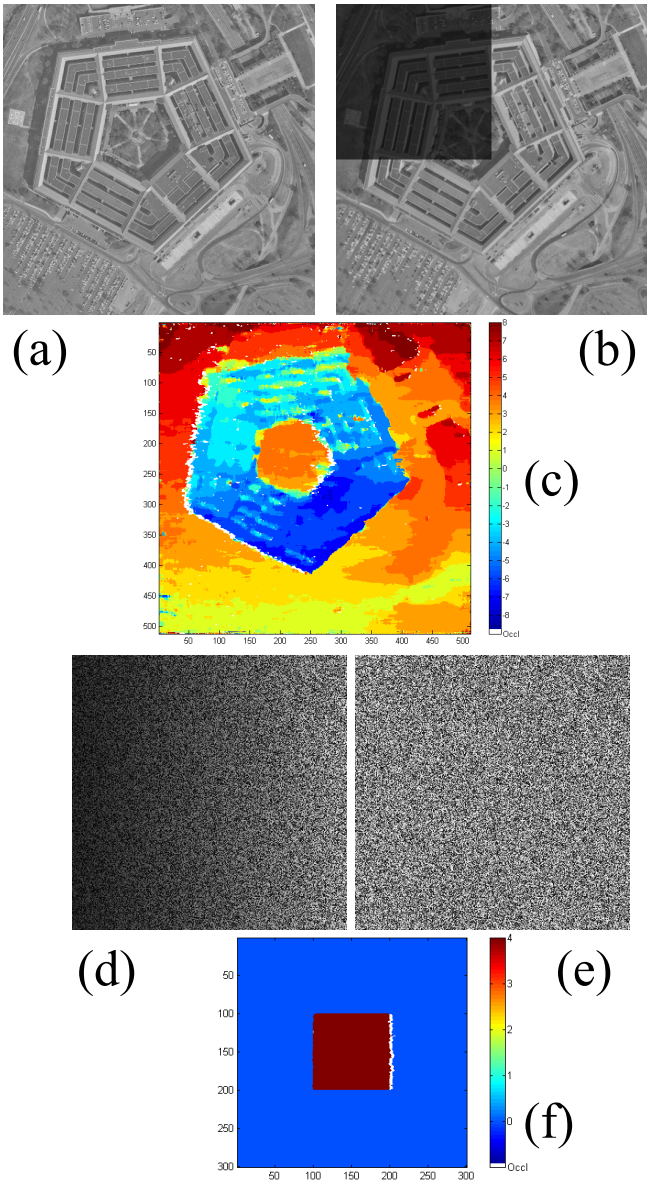

Figure 5.3: (a) and (b) denote a left and right image from the *pentagon* sequence, with only a part of the right image having different contrast. (c) shows the result of the *linear fit* algorithm. (d) and (e) are a random dot stereo pair of images, with a quadratic variation in contrast across the left image, (f) shows the resulting disparity map. Occlusions are colored white in the disparity maps.

$$r_s = \int_{-\infty}^{+\infty} dx \left(f_l(x) I_l(x) + f_r(x) I_r(x) \right) \tag{5.7}$$

where $I_l(x)$ and $I_r(x)$ are the left and right images respectively. The receptive field profiles of the cell for the left and right eye are denoted by $f_l(x)$ and $f_r(x)$, and are well described by Gabor functions (see Nomura et al [NMF90]). The equations below give the example of Gabor functions centered at $x = 0$.

$$f_l(x) = e^{-x^2/2\sigma^2} cos(\omega x + \phi_l) \tag{5.8}$$

$$f_r(x) = e^{-x^2/2\sigma^2} cos(\omega x + \phi_r) \tag{5.9}$$

They also showed that the response of a complex cell can be modeled by summing the squared responses of two simple cells in quadrature (see [AB85, WA85, ODF90, Qia94]) as follows:

$$r_c = (r_{s,1})^2 + (r_{s,2})^2 \tag{5.10}$$

Qian [Qia94] further showed that if the stimulus disparity D is much smaller than the receptive field width, then the response of a complex cell becomes

$$r_c \approx (1-\gamma)^2 \rho^2 + 4\gamma\rho^2 cos^2 \left(\frac{\phi_l - \phi_r}{2} + \frac{\omega D}{2} \right) \tag{5.11}$$

where ρ^2 is the power (amplitude squared) in the frequency channel and γ is the contrast ratio between the two images. Qian further argues that as little as two complex cells with different $(\phi_l - \phi_r)$ are sufficient to determine the disparity at any location using a particular frequency channel. This basically amounts to the separation of the amplitude of the response (i.e. the power) from the phase difference, thereby allowing the use of the phase difference for disparity computation. Hence, although the Gabor convolution itself does not yield contrast invariance, using the phase does achieve this goal. Thus, the underlying theme of using phase differences in various frequency channels is of primary interest to us when we attempt to design a biologically motivated contrast invariant local matching metric. There exist a large number of techniques in the literature which compute disparity

by making use of phase differences explicitly [San88, JJ88, FJJ91, Wen94] or implicitly via phase correlation [KH75, GK89, OP89]. Of particular interest to us in the next section is the method of Fleet [Fle94], which uses phase correlation to compute disparity maps.

Large disparities cannot be encoded by high frequency channels in a pure phase shift model. On the other hand, high frequency channels are clearly essential in order to obtain good localization of the depth boundaries. Hence, a hybrid model involving position and phase shifts needs to be used. Anzai et al. [AOF97] have discussed the experimental observations vis-a-vis phase and positional shifts, and have concluded that although binocular disparity is mainly encoded through the phase disparity, position disparity may play a significant role in the case of high frequency channels. The model which we discuss in the next section goes to the other extreme of using position shifts alone.

5.3 Multiple spatial frequency channels

In this section, we discuss our approach for finding correspondence in a contrast invariant manner, which is inspired by the ideas presented in the previous section on biological vision. We use pure positional shifts, maintaining our approach in previous chapters of searching over a disparity space. However, we use phase differences from various frequency channels to determine the degree of local matching.

Let us examine the problem in one dimension for purposes of explanation. Given two scanlines $I_l(x)$ and $I_r(x)$ from the left and right images, we apply complex valued Gabor filters of the form $G_{x_0,\omega}(x)$, where the filter is centered at x_0 in space, and at ω in the frequency domain. For example, the filter centered at $x = 0$ having a frequency ω is given by:

$$G_{0,\omega}(x) = e^{-x^2/2\sigma^2} e^{i\omega x} \tag{5.12}$$

The real and imaginary part of this complex valued filter form a quadrature pair. We select σ to ensure a constant one octave bandwidth in the frequency domain. The space frequency domain can be sampled by a complete set of functions obtained by translation and scaling of this basic filter. In the two dimensionsal case, rotation is also present, since the filters are oriented. Let us denote the output of the filter $G_{x_0,\omega}(x)$ at position x with $x_0 = x$ on the left image by

$$L_\omega(x) = G_{x,\omega}(x) \otimes I_l(x) \tag{5.13}$$

and on the right image by

$$R_\omega(x) = G_{x,\omega}(x) \otimes I_r(x) \tag{5.14}$$

If the phase difference between the right and left response at position x and disparity d is denoted by $\Delta\phi_{\omega,d}(x)$, then

$$e^{i\Delta\phi_{\omega,d}(x)} = \frac{L_\omega(x)R^*_\omega(x+d)}{|L_\omega(x)R^*_\omega(x+d)|} \tag{5.15}$$

Notice that we are explicitly shifting the right image by the candidate disparity d before taking the product, and hence if the left and right images locally match in this frequency channel, we would expect the phase difference to be zero. As we shall see below, the deviation of the phase difference from zero can be implicitly used as a measure of local matching.

Phase correlation can be thought of as a voting scheme [Fle94], such that when we take the inverse Fourier transform, each channel casts a vote in a sinusoidal manner in the spatial domain. The inverse Fourier transform using the outputs of filters centered at a spatial position x_0 is given by (ignoring the spatial extents of the filters for the moment):

$$G_{x_0 d}(x) = \int e^{i\Delta\phi_{\omega,d}(x_0)} e^{i\omega(x-x_0)} d\omega \tag{5.16}$$

Ideally, the real parts of all the sinusoids would sum to create a single peak at a certain position, and the imaginary parts would all cancel out. To find the degree of local matching, we want to measure the likelihood that this peak lies at the center position of the applied filters, i.e. at $x = x_0$. To achieve this, we can simply use the real part of the function $G_{x_0,d}(x_0)$ at $x = x_0$ as a measure of the likelihood that the peak lies at $x = x_0$.

Thus, if we are applying N filters to the images, and the phase difference at location x from channel i for disparity d is denoted by $\Delta\phi_{i,d}(x)$, then as per the above discussion, we can define a function which sums the real parts of the inverse Fourier transform in the discrete case:

$$H(x,d) = \frac{1}{N} \sum_{N\ channels} cos(\Delta\phi_{\omega,d}(x)) \tag{5.17}$$

Note that the factor $1/N$ is used to ensure that $H(x,d)$ has the same range as the cosine function, i.e. $[-1,1]$. Since phase relationships can become unreliable if the power in the selected frequency channels is close to zero, we define another function $G(x,d)$ as follows:

$$\begin{aligned} G(x,d) &= W(x,d) \cdot H(x,d) + (1 - W(x,d)) \cdot (1) && (5.18)\\ W(x,d) &= exp(-\alpha P(x,d)) && (5.19)\\ P(x,d) &= \sum_{\omega} |L_{\omega}(x) R^{*}_{\omega}(x+d)| && (5.20) \end{aligned}$$

Here, $P(x,d)$ denotes the sum of the magnitudes (power) of all the filter responses, and $W(x,d)$ is a weight which exponentially decays to zero as $P(x,d)$ increases. Hence, as $P(x,d)$ tends to zero, the weighting function reduces the importance of the phase difference function $H(x,d)$ and forces $G(x,d)$ closer to the value 1, which indicates good matching. Thus, if the phase responses are unreliable due to the nonexistence of local variations in the intensity, we take the default position that there exists a local match. Note that $G(x,d)$ also lies in the range $[-1,1]$.

In chapter 4, we have defined the conductivity $C(x,d)$ for position x and disparity d, such that $C(x,d)$ had values in the range $[0,1]$. We have also defined a local measure of matching $M(x,d)$, which describes how well the left image $I_l(x)$ matches the shifted right image $I_r(x+d)$. To obtain a contrast invariant matching algorithm, we can choose

$$C(x,d) = \frac{G(x,d)+1}{2} \tag{5.21}$$

$$M(x,d) = C(x,d) \tag{5.22}$$

Other choices for $C(x,d)$ and $M(x,d)$ are certainly possible, but the above choice is arguably the simplest. Figure 5.4 shows the results of the above algorithm on a few stereo pairs with different left and right image contrasts. We perform the Gabor convolutions

in two dimensions using an efficient implementation by Nestares et al [NNPT98] at four different scales and four different orientations.

In Figure 5.5, we have added noise in the high frequency region of the right image. The linear fit algorithm discussed in Section 5.1 uses only the high frequency channel, leading to large errors in the computed disparity map. The algorithm discussed in this section uses multiple frequency channels, and is relatively robust to the addition of noise in some of the channels.

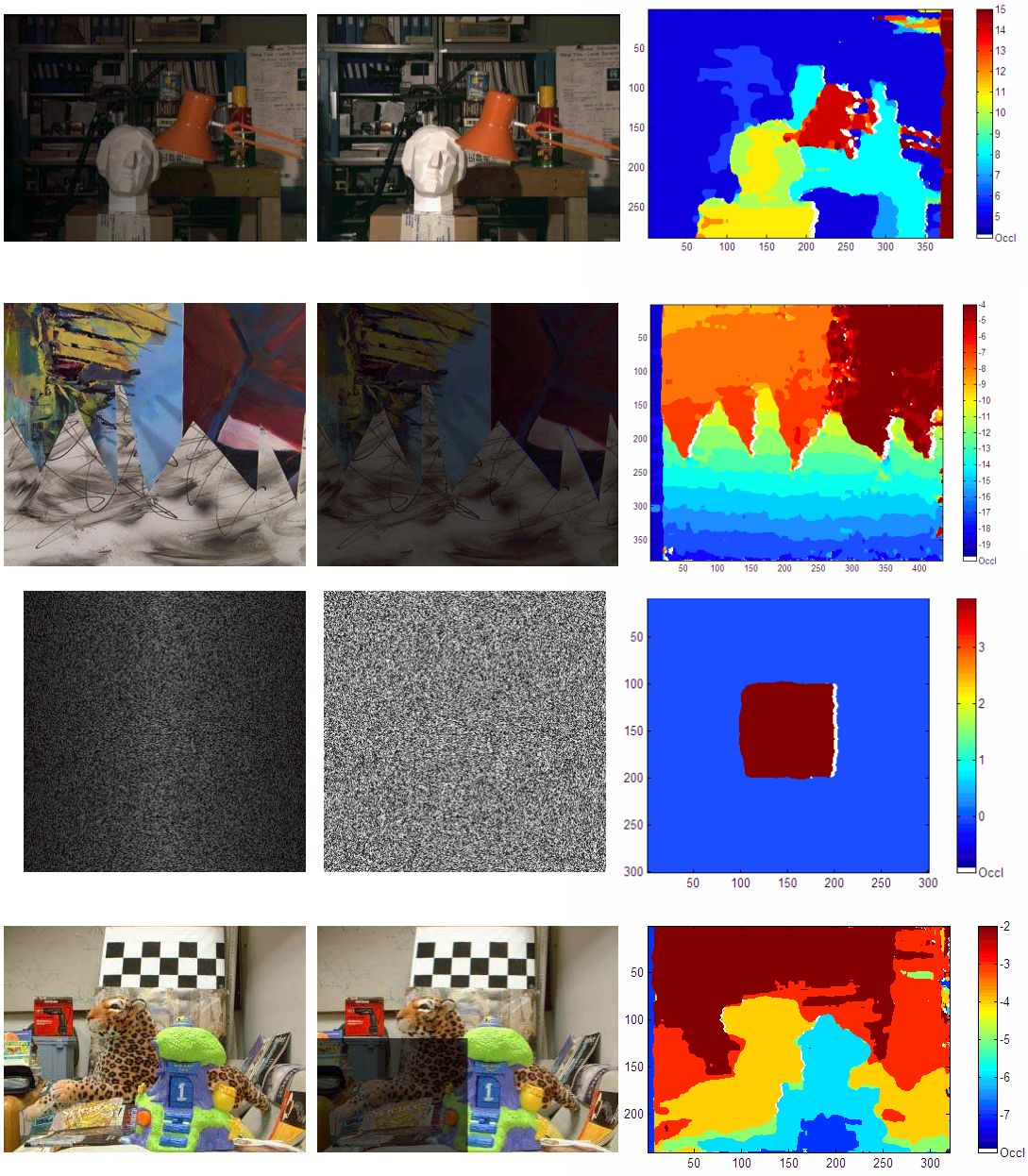

Figure 5.4: Row 1: *Tsukuba* stereo pair with a quadratic contrast variation across the left image. The disparity map is shown on the right. Row 2: *Sawtooth* stereo pair with different image contrasts. Row 3: Random dot pair with a Gaussian contrast variation across the left image. Row 4: *Leopard* stereo pair with different contrast in a square patch in the right image.

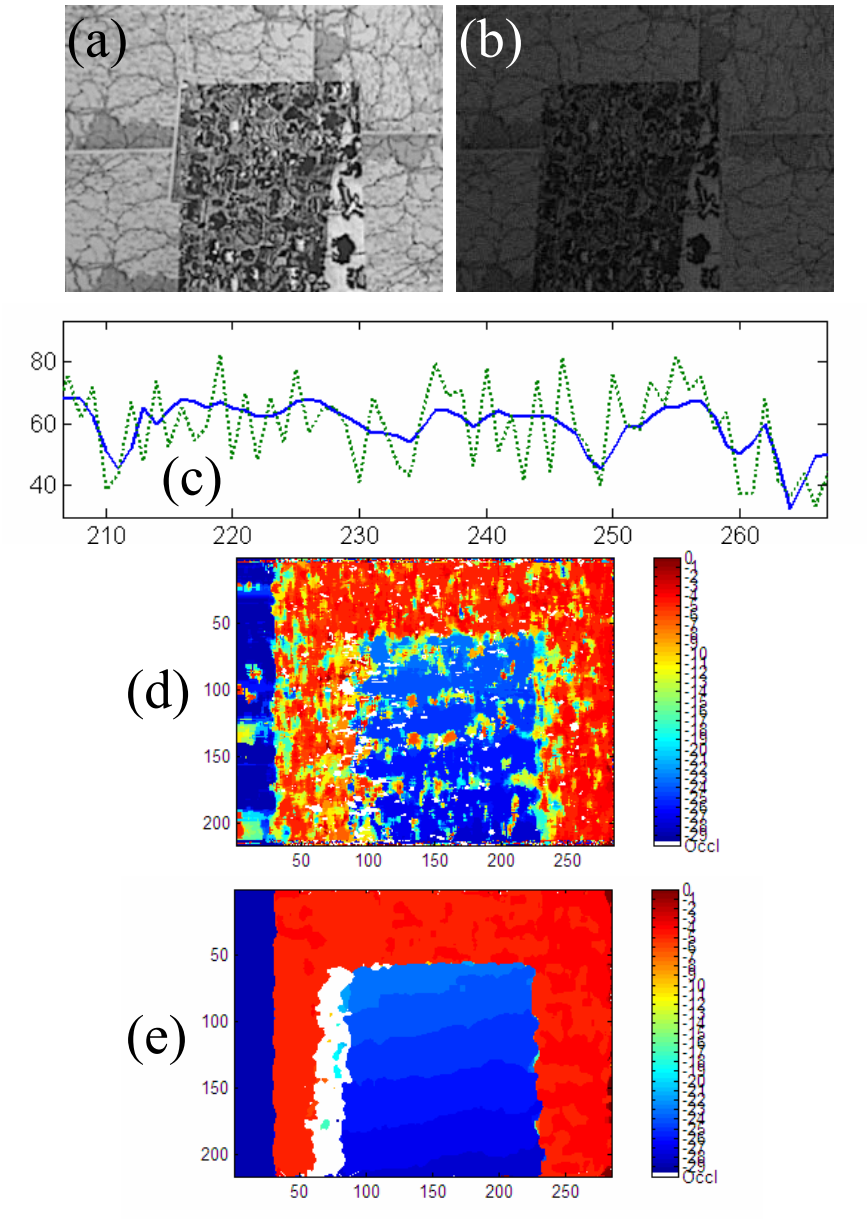

Figure 5.5: (a) Left image from the *map* sequence. (b) Right image with lower contrast and the addition of noise in the high frequency channel. The noise causes upto 25% variation in the original intensity. (c) shows a portion of a scanline in the right image, where the solid line shows the intensity values before addition of the noise, and the dotted line shows the values after addition of the noise. (d) shows the results of the *linear fit* algorithm. (e) shows the results of the *multiple frequency channel* algorithm.

Chapter 6

Occlusions and ordinal depth

In the previous chapters, we have explored methods of performing stereo disparity estimation and optical flow estimation concurrently with segmentation and occlusion detection. In this chapter, we show how our knowledge of occlusions can be used in order to find an ordinal depth order between different regions of an image. We will demonstrate that in order to find ordinal depth, occlusions must not only be known, but they must also be filled. We present a novel algorithm for occlusion filling and deducing ordinal depth using three frames of a video sequence. This algorithm functions even in the presence of independently moving objects. This ordinal depth computed by this algorithm will be put to use in Chapter 7 to facilitate the detection of new types of independently moving objects.

6.1 Why occlusions must be *filled*?

Given two frames from a video, occlusions are points in one frame which have no corresponding point in the other frame. However, merely knowing the occluded regions is not sufficient to deduce ordinal depth. In Figure 6.1, we show a situation where an occluded region O is surrounded by two regions R_1 and R_2 which are visible in both frames.

If the occluded region O belongs to region R_1, the we know that R_1 must be behind R_2, and vice-versa.

This statement is extremely significant, since it holds true even when the camera undergoes general motion, and even when we have independently moving objects in the scene! Thus, we need to know *'who occluded what'* as opposed to merely knowing *'what was occluded'*. Since optical flow estimation provides us with a segmentation of the scene (regions of continuous flow), we now have to *assign flows to the occluded regions*, and *merge* them with existing segments. Having done this, we can then deduce ordinal depth relations between segments.

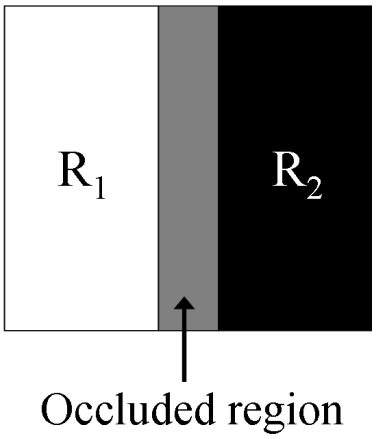

Figure 6.1: If the occluded region belongs to R_1, then R_1 is behind R_2 and vice-versa.

6.2 Occlusion filling (rigid scene, no independently moving objects)

Let us first consider how occlusion filling may be performed using two frames, if we know that there are no independently moving objects. Here is a simple example: if the camera translates horizontally to the right, then all the scene objects will move to the left. Hence, as shown in Figure 6.2(a), if object A is in front of object B, then object A moves more to the left than B, causing a part of B on the left of A to become occluded. It is easy to see that in the case where the camera translates horizontally to the right, occluded parts in the first frame always belong to segments on their left. In the case of generalized rotation and translation, if the focus of expansion (FOE) or contraction (FOC) is known, occlusions are filled in as follows: first, draw a line L from the FOE/FOC to an occluded pixel O, then:

(A) as shown in Figure 6.2(b), if we have a focus of *expansion*, the flow of the occluded pixel is obtained from the flow at the nearest visible pixel P on this line L, such that O lies between P and the FOE.

(B) As shown in Figure 6.2(c), if we have a focus of *contraction*, then fill in with the nearest pixel Q on line L, such that Q lies between O and the FOC.

Thus, the knowledge of the FOE helps us devise a simple occlusion filling strategy. *An important note in this regard is that camera rotation does not change visibility of objects and cannot cause occlusions* [SSV01], *hence knowing the FOE is enough*. (Also, knowledge of the occlusions can be used to reduce the motion valley). But what do we do when the FOE is unknown and we have independently moving objects?

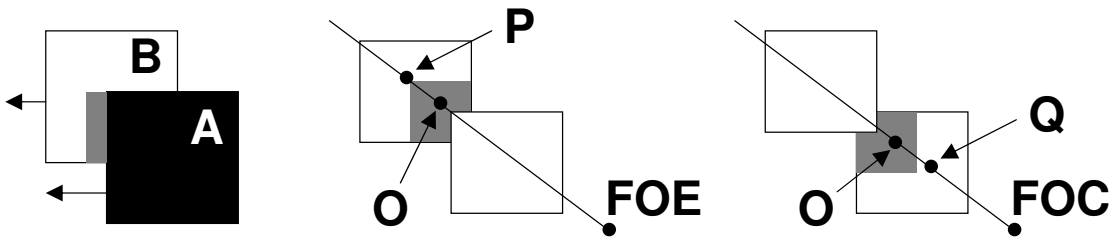

Figure 6.2: Occlusion Filling: from left (a) to (c). Gray regions indicate occlusions (portions which disappear in the next frame)

6.3 Generalized occlusion filling (in the presence of moving objects)

In the presence of moving objects, even the knowledge of the FOE will provide little assistance in the filling of occlusions, since the occlusions will no longer obey the criteria presented in the previous section. A more general strategy must be devised to deal with occlusion filling in the presence of moving objects. The simplest idea which comes to mind is the following: if an occluded region O lies between regions R_1 and R_2, then we can decide how to fill O based on its *similarity* with R_1 and R_2. However, *similarity* is an ill-defined notion in general, since it may mean similarity of gray value, color, texture or some other feature. Thus, using a similarity measure to decide how to fill occlusions will create many failure modes. We shall now present a novel and robust strategy for filling occlusions in the general case using three frames instead of two, without relying on similarity.

Given three consecutive frames F_1, F_2, F_3 of a video sequence, we use our optical flow algorithm (see Chapter 2) to compute the following:

1. Using F_1 and F_2, we find flow $\vec{u}_{12}$ from frame F_1 to F_2, the reverse flow $\vec{u}_{21}$ from frame F_2 to F_1. The algorithm also gives us occlusions O_{12} which are regions of frame F_1 which are not visible in frame F_2. Similarly, we also have O_{21}.

2. Using frames F_2 and F_3, we find $\vec{u}_{23}$ and $\vec{u}_{32}$, and O_{23} and O_{32}.

Our objective is to fill the occlusions O_{21} and O_{23} in frame F_2 to deduce ordinal depth. The idea is simple:

O_{23} denotes areas of F_2 which have no correspondence in F_3. However, these areas were visible in both F_1 and F_2, hence in $\vec{u}_{21}$, these areas have already been grouped with their neighboring

regions. Therefore, we can use the segmentation of flow $\vec{u}_{21}$ to fill the occluded areas O_{23} in the flow field $\vec{u}_{23}$.

Similarly, we can use the segmentation of $\vec{u}_{23}$ to fill the occluded areas O_{21} in the flow field $\vec{u}_{21}$. This method of occlusion filling is robustly able to fill the occlusions. After filling, deducing ordinal depth is straightforward: if an occlusion is bounded by R_1 and R_2 and if $R_1$was used to fill it, then R_1 is below R_2.

If our flow segmentation consists of N regions, then we can create an ordinal depth matrix $OrdM$ of size $N \times N$, such that if $OrdM(i,j) = 1$, then region i is above region j. $OrdM(i,j) = 0$ denotes that no relation was found between regions i and j. If this matrix is treated as the adjacency matrix of a directed acyclic graph (DAG) of N nodes, we can deduce secondary relations between regions by using a method functionally similar to topological sort. This method infers relations like: *if region i is above j, and j is above k, then i is above k.*

The occlusion filling strategy mentioned above functions even in the presence of independently moving objects. It can also be used in conjunction with a different optical flow algorithm which deals with non-rigid objects, to handle such cases as well.

6.4 Experiments

Figure 6.3 shows an example of ordinal depth estimation using three frames of the *flower-garden* video sequence. The top of the figure shows three frames of the sequence, in which a region marked by a yellow color is visible in frames F_1 and F_2, and becomes occluded by the tree in frame F_3. Since the yellow region is visible in both frames F_1 and F_2, it has already been segmented in flow $\vec{u}_{21}$ as part of the background, and not the tree. This segmentation can be used in conjuction with the flow values in $\vec{u}_{23}$ to fill in the occluded region, and deduce ordinal depth relations. The last row of the figure shows one of the deduced relations, in which the tree trunk (colored green) is found to be in front of the red background region.

In the next chapter, we shall use our ability to deduce ordinal depth from occlusions to help us detect new types of independently moving objects.

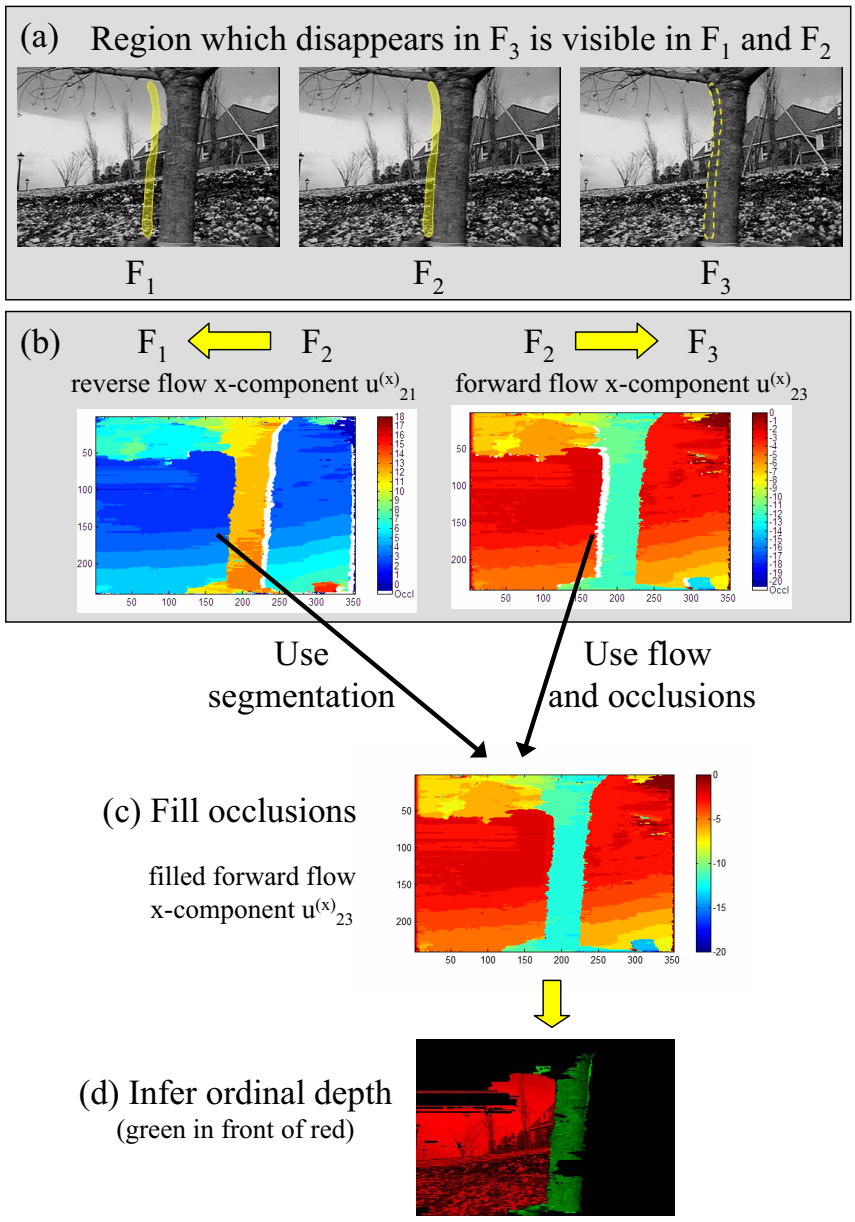

Figure 6.3: Generalized occlusion filling and ordinal depth estimation. (a) Three frames of a video sequence. The yellow region which is visible in F_1 and F_2 disappears behind the tree in F_3. (b) Forward and reverse flow (only the x-components are shown). Occlusions are colored white. (c) Occlusions in $\vec{u}_{23}$ are filled using the segmentation of $\vec{u}_{21}$. Note that the white areas have disappeared. (d) Deduced ordinal depth relation. In a similar manner, we can also fill occlusions in $\vec{u}_{21}$ using the segmentation of $\vec{u}_{23}$ to deduce ordinal depth relations for the right side of the tree.

Chapter 7

Motion segmentation

Motion segmentation is the problem of finding independently moving objects in a video. This process is conceptually simple when the camera is stationary, and a variety of solutions exist in today's literature under the general heading of background subtraction. However, if the camera itself is moving, a general and robust solution is still elusive. In the case of the stationary camera, the motion field on the image is only caused by moving objects, but in the case of a moving camera, the motion field is generated by the combined effect of camera motion, structure and the independent motion of objects. Isolating the contribution of each of these three factors is needed to solve the independent motion problem completely.

7.1 Previous work

Prior research can mostly be classified into two groups: (a) the approaches relying, prior to 3D motion estimation, on 2D motion field measurements only [BK94, BBH+89, OB95, Wei97]. The limitations of these techniques are well understood. Depth discontinuities and independently moving objects both cause discontinuities in the 2D optical flow, and it is not possible to separate these factors without 3D motion estimation. (b) approaches which assume that partial or full information about egomotion is available or can be recovered. Adiv [Adi85] first segments on the basis of optical flow, and then groups the segments by searching for agreeable 3D motion parameters. Zhang et al [ZFA88] utilize rigidity constraints on a sequence of stereo images to find egomotion and moving objects. Thompson and Pong's [TP90] first method finds inconsistencies between the egomotion and the flow field by using the motion epipolar constraint, while the second method relies on external depth information. Nelson [Nel91] discusses two approaches, the first of which is similar to Thompson and Pong, while the second relies on acceleration detection. Sinclair [Sin93] uses the angular velocity field and the premise that independently moving objects violate the epipolar constraint. Torr and Murray [TM94] find a set of fundamental matrices to

describe the observed correspondences by hypothesizing clusters using robust statistics. Costeira and Kanade [CK95] use the factorization method along with a feature grouping step (block diagonalization of the shape interaction matrix).

Some techniques, such as [WM97], which address both 3D motion estimation and moving object detection, are based on alternate models of image formation, such as weak perspective. Such additional constraints can be justified for domains such as aerial imagery. In this case, the planarity of the scene allows a registration process [TPHA00, ASB94, WB95, ZC93], and uncompensated regions correspond to independent movement. This idea has been extended to cope with general scenes by selecting models depending on the scene complexity [Tor98], or by fitting multiple planes using the plane plus parallax constraint [IA98, SGK00]. The former [IA98] uses the best of 2D and 3D techniques, progressively increasing the complexity based on the situation. The latter [SGK00] also develops constraints on the structure using three frames. Clearly, improvement in motion detection can be gained using temporal integration. Yet questions related to the integration of 3D motion and scene structure are not yet well understood, as the extension of the rigidity constraint to multiple frames is non-trivial.

Most of the techniques presented above detect independently moving objects based on the 3D motion estimates, either explicitly or implicitly. Some utilize inconsistencies between egomotion estimates and the observed flow field. Some techniques utilize external information such as depth from stereo, or partial egomotion from other sensors. Nevertheless, the central problem faced by all motion based techniques is that in general, it is nearly impossible to uniquely estimate 3D motion. Due to the confusion between rotation and translation for a camera with a restricted field of view (FOV) undergoing a small motion between frames, a set of egomotion solutions (translation-rotation values) instead of a unique solution is consistent with the observed (noisy) flow field. The occurrence of disjoint sets of motion solutions, which permits explicit or implicit motion based clustering, is required by many of the above algorithms.

In this chapter, we show that this is not often possible, primarily due to inherent ambiguities in 3D motion estimation. Instead, we turn the problem on its head by *assuming* the existence of ambiguities in the 3D motion estimation, and demonstrate the existence of different categories of moving objects which are classified on the basis of the constraints required for their detection. In Section 7.2, we discuss the ambiguities involved in estimating

the egomotion for a camera with a small FOV undergoing a small motion between frames. Section 7.3 describes the different categories of moving objects along with the constraints required for their detection. We emphasize the important role of occlusions in motion segmentation and motion estimation, and utilize the methods of Chapter 6 to deduce ordinal depth from occlusions. Section 7.4 discusses our method of detecting moving objects based on motion clustering. Section 7.5 describes the complete algorithm for detecting moving objects. Finally, in Section 7.6, we present experiments with a real scene to illustrate the presence of each type of moving object and how it is detected.

7.2 Ambiguities in 3D motion estimation

Given the knowledge of optical flow, there exist many techniques in the literature [HZ00] which estimate the 3D motion. Several studies have addressed the issue of noise sensitivity in structure from motion. In particular, it is known that for a moving camera with a small field of view viewing a scene with insufficient depth variation, translation and rotation are easily confused [Adi89, DS97]. This can be intuitively understood by examining the differential flow equation:

$$\begin{aligned} u &= \frac{-t_x f + x t_z}{Z} + \alpha \frac{xy}{f} - \beta \left(\frac{x^2}{f} + f \right) + \gamma y \\ v &= \frac{-t_y f + y t_z}{Z} + \alpha \left(\frac{y^2}{f} + f \right) - \beta \frac{xy}{f} - \gamma x \end{aligned} \tag{7.1}$$

In the above equation, (u, v) is the optical flow, (t_x, t_y, t_z) is the translation, (α, β, γ) is the rotation and $Z(x, y)$ is the depth map. Notice that for a planar scene, upto zeroeth order, we have $u \approx -t_x f/Z - \beta f$ and $v \approx -t_y f/Z + \alpha f$. Intuitively, we can see how translation along the x-axis t_x can be confused with rotation β along the y-axis, and t_y with α for a small field of view. Furthermore, Maybank [May87] and Heeger and Jepson [HJ92] show that if the scene is sufficiently nonplanar, then the minima of the cost function resulting from the epipolar constraint lie along a line in the space of translation directions, which passes through the true translation direction and the viewing direction. These ideas have been formalized in [FA00] and an algorithm independent stability analysis of the structure from motion problem has been carried out.

Given a noisy flow field, any motion estimation technique will yield a region of solutions in the space of translations, instead of a unique solution. We refer to this region of

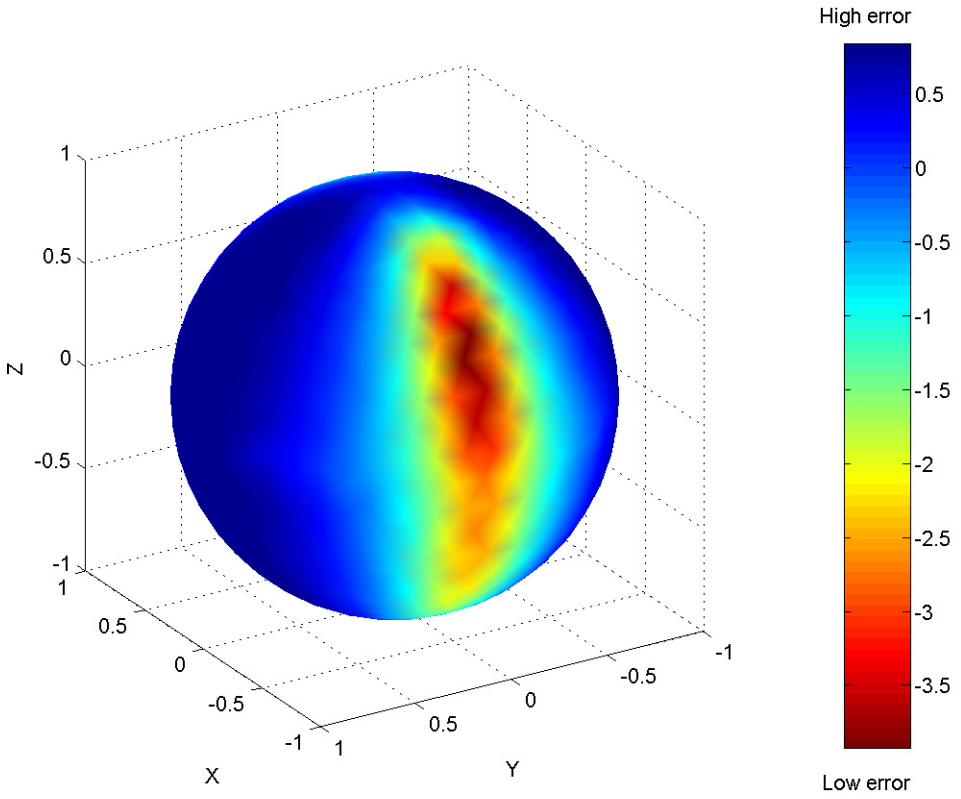

Figure 7.1: Motion valley (red) visualized as an error surface in the 2D space of directions of translation. The error is found after finding the optimal rotation and structure for each translation direction.

solutions as the *motion valley.* Each translation direction in the motion valley, along with its best corresponding rotation and structure estimate, will agree with the observed noisy flow field. Figure 7.1 shows a typical error function obtained using the method of Brodsky et al [BFA00] plotted on the 2D spherical surface of translational directions. The best solutions lie in the red area of the surface. The error surface makes it evident that attempting to pick a single solution in this valley is futile. Since such valleys are ubiquitous, motion based clustering can only succeed if a scene entity has a motion which lies in a valley which is separate from the valley containing the background motion. In the next section, we go beyond motion based clustering to show that even if motion estimation yields a single valley, there are certain types of moving objects which can still be detected.

7.3 Types of independently moving objects

We now discuss three distinct classes of independently moving objects; the moving objects belonging to *Class 1* can be detected using motion based clustering, the objects in *Class 2* are detected by detecting conflicts between depth from motion and ordinal depth from occlusions, and objects in *Class 3* are detected by finding conflicts between depth from motion and depth from another source (such as stereo). Any specific case will consist of a combination of objects from these three classes. Figure 7.2 shows illustrative examples of the three classes.

7.3.1 Class 1: 3D motion based clustering

The first row of Figure 7.2 shows a situation in which the background objects (non-independently moving) are translating horizontally, while the red object is moving vertically. In this scenario, motion based clustering approaches will be successful, since the motion of the red object is not contained in the motion valley of the background. Thus, *Class 1* objects can be detected using motion alone. Our strategy for quickly performing motion based clustering and detecting *Class 1* objects is discussed in Section 7.4.

7.3.2 Class 2: Ordinal depth conflict between occlusions and structure from motion

The second row of Figure 7.2 shows a situation in which the background objects are translating horizontally to the right, and the red object also moves towards the right. In this

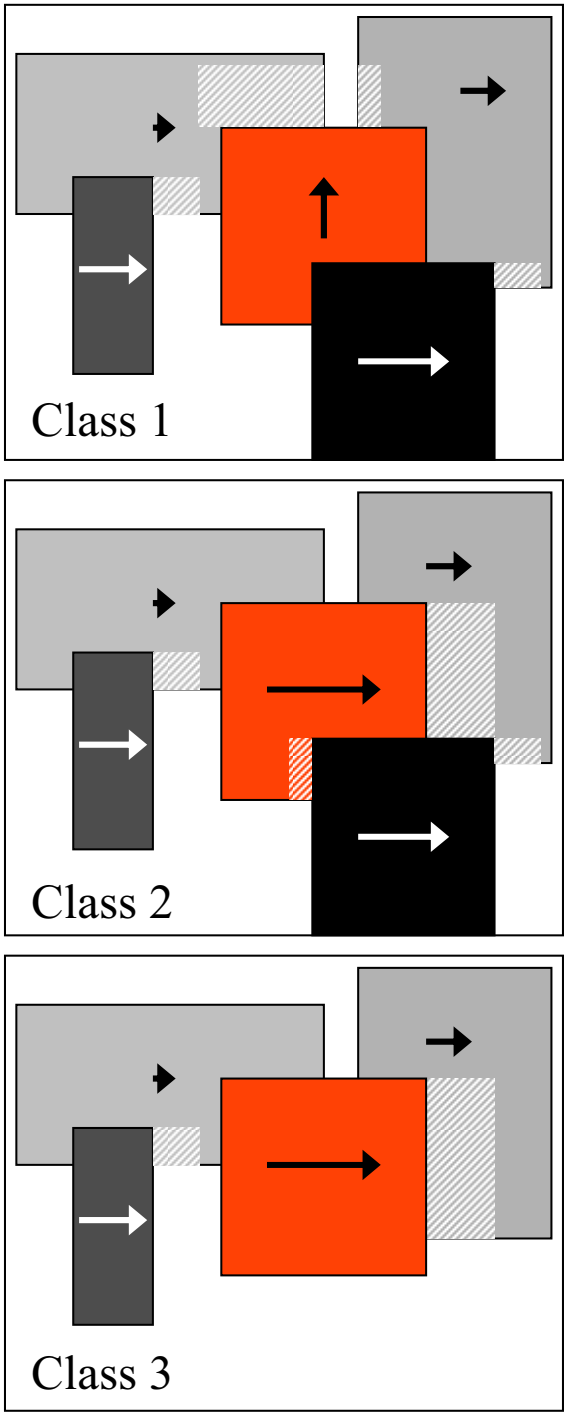

Figure 7.2: Toy examples of three classes of moving objects. In each case, the independently moving object is red colored. Portions of objects which disappear in the next frame (i.e. occlusions) are shown in a dashed texture.

scenario, motion estimation will not be sufficient to detect the independently moving object, since motion estimation yields a single valley of solutions. An additional constraint, which may be termed as the *ordinal depth conflict* or the *occlusion-structure from motion (SFM) conflict* needs to be used to detect the moving object.

Notice the occluded areas in the figure: we can use our knowledge of these occlusions to develop ordinal depth (ie. *front/back*) relationships between regions of the scene (see Chapter 6). In this example, the occlusions tell us that the red object is *behind* the black object. However, if we compute structure from motion, since the motion is predominantly a translation, the result would indicate that the red object is in front of the black object (since the red object moves faster). This conflict between ordinal depth from occlusions and structure from motion permits the detection of *Class 2* moving objects. Note that the given example is one of the simplest cases of *Class 2* moving objects, and that more complicated examples involving large camera rotation can also be devised.

7.3.3 Class 3: Cardinal depth conflict

The third row of Figure 7.2 shows a situation similar to the second row, except that the black object which was in front of the red object has been removed. Due to this situation, the ordinal depth conflict which helped us detect the red object in the earlier scenario is no longer present. In order to detect the moving object in this case, we must employ cardinal comparisons between structure from motion, and structure from another source (such as stereo) to identify deviant regions as *Class 3* moving objects. In our experiments, we have used a calibrated stereo pair of cameras to detect objects of *Class 3.* The calibration allows us to compare the depth from motion directly with the depth from stereo upto a scale. We use k-means clustering (with k = 3) on the depth ratios to detect the background (the largest cluster). The reason for using k=3 is to allow us to find three groups: the background, pixels with depth ratio greater than the background, and pixels with depth ratio less than the background. Pixels not belonging to the background cluster are the *Class 3* moving objects. At this point, it may be noted that alternative methods exist in the literature (e.g. [ZFA88]) for performing motion segmentation on stereo images, which can also be used to detect *Class 3* moving objects.

7.4 Motion based clustering

Motion based clustering is in itself a difficult problem, since the process of finding the background motion and finding the independently moving clusters has to performed concurrently. A chicken-and-egg aspect of this problem is as follows: if we knew the background pixels, we could find the background motion accurately. Similarly, knowing the background motion accurately would enable us to better detect independently moving regions. The dilemma is in choosing how to begin.

The camera motion is described by five parameters, two for the direction of translation and three for the rotation. To bootstrap the process and find a subset of the background that will allow 3D motion estimation, we use a signal processing technique that will uncover four parameters, not directly related to the five parameters of the camera motion. The technique amounts to performing phase correlation in successive video frames, both in the cartesian and the logpolar domain. The technique is described in detail in Appendix A. Not only does it allow us to start the process, but it also stabilizes the video.

Thus, our method consists of two simple steps:

1. Use phase correlation [RC96] on the two images in the cartesian representation (to find 2D translation t_x, t_y) and in the logpolar representation (to find scale S and z-rotation γ), giving us a four parameter transformation. Phase correlation can be thought of as a voting approach [Fle94], and hence we find that these four parameters depend primarily on the dominant background motion even in the presence of moving objects. This assumption is true as long as the background texture dominates the textures on the moving objects. This four parameter transform predicts a flow direction at every point in the image. We select points in the image whose true flow direction lies within a cone of 45 degrees about the predicted direction or its exact opposite direction.

2. The optical flow at the points selected using the earlier procedure are used to estimate the background motion valley using the technique of Brodsky et al [BFA00]. In addition, the positive depth constraint is used to reduce the size of the valley by only keeping translations for which the computed depth at all points is positive. Since all points in the valley predict similar flows on the image (that is why the valley exists in the first place), we can pick any solution in the valley and compare the reprojected

flow with the true flow. Regions where the two flows are not within 45 degrees of each other are considered to be *Class 1* independently moving objects.

The first step helps us to quickly select a subset of the image which contains mostly background pixels, so that accurate egomotion estimation can be performed without iterative processes. The second step then uses the egomotion to find the moving objects. The voting nature of phase correlation helps us to get around the chicken-and-egg aspect of the problem. It may be noted that other algorithms for motion based clustering (discussed in Section 7.1) can also be used to realise this motion clustering step.

7.5 Algorithm summary

1. Input video sequence: $V = \{F_1, F_2, ..., F_n\}$
2. foreach $F_i \in V$, do
   (a) find forward $\vec{u}_{i,i+1}$ and reverse $\vec{u}_{i,i-1}$ flows with occlusions $O_{i,i+1}$ and $O_{i,i-1}$
   (b) select a set S of pixels using phase correlation between F_i and F_{i+1}
   (c) find background motion valley using the flows for pixels in S
   (d) detect *Class 1* moving objects and background B_1
   (e) find ordinal depth relations using results of step (a)
   (f) for pixels in B_1, detect *Class 2* moving objects, and new background B_2
   (g) if depth from stereo is available, detect *Class 3* objects present in B_2

7.6 Experimental results

Figure 7.3 shows a situation in which the background is translating horizontally, while a teabox is moved vertically. In this scenario, since the teabox is not contained in the motion valley of the background, it is detected as a *Class 1* moving object.

Figure 7.4 shows a situation in which the background is translating horizontally to the right, and the leopard is dragged (by a person who is not in the frame) towards the right. There is a red box in front of the leopard, which is a part of the static (non independently

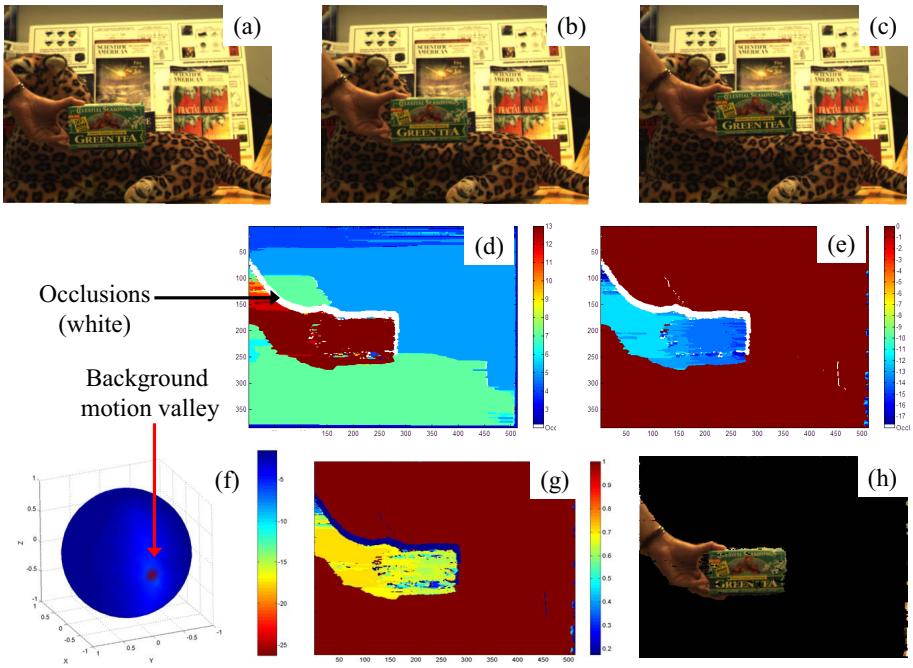

Figure 7.3: Class 1: (a,b,c) shows three frames of the *teabox* sequence. (d,e) show X and Y components of the optical flow using frames (b) and (c). Occlusions are colored white. (f) shows the computed motion valley for the background. (g) shows the cosine of the angular error between the reprojected flow (using the background motion) and the true flow. (h) shows detected *Class 1* moving object after thresholding angular error (greater than 45 degrees).

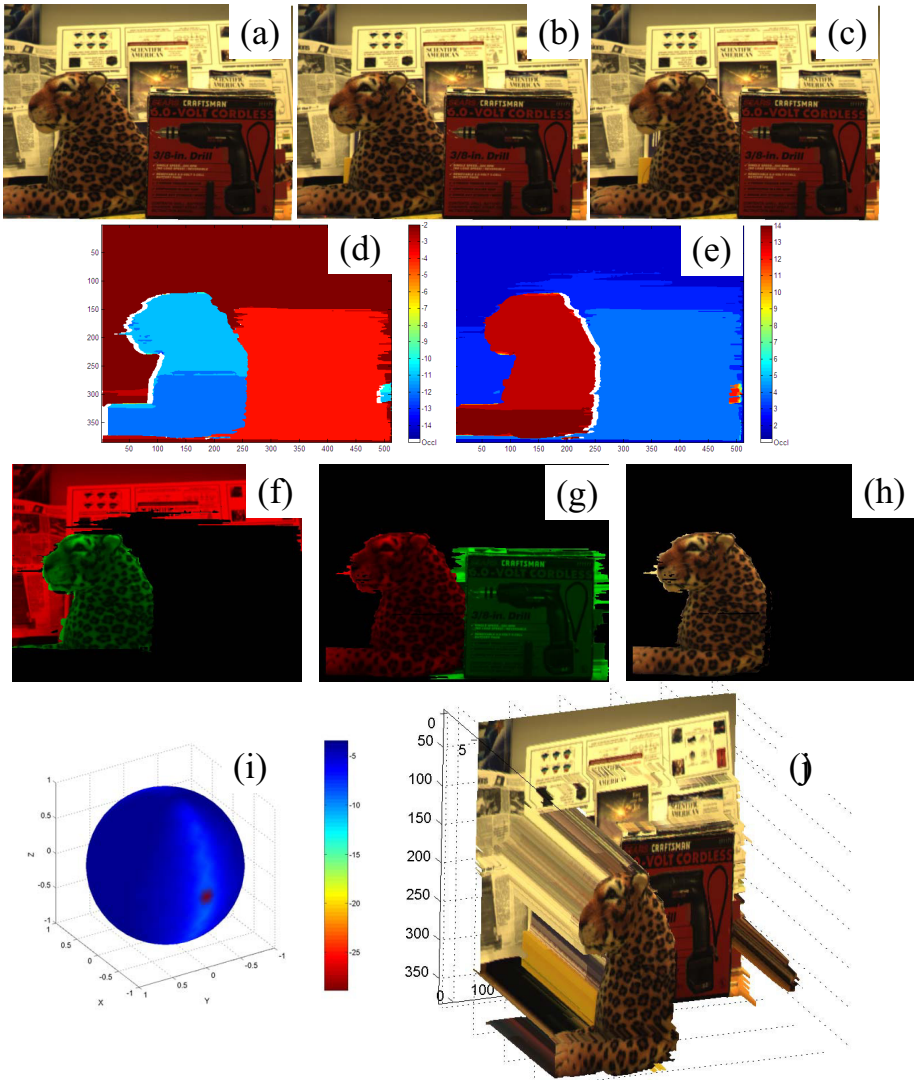

Figure 7.4: Class 2: (a,b,c) show three frames F_1, F_2, F_3 of the *leopardA* sequence. (d) shows X-component of the optical flow $\vec{u}_{21}$ from frame F_2 to F_1. Y-component (not shown) is zero. White regions denote occlusions (e) shows X-component of the optical flow $\vec{u}_{23}$ from frame F_2 to F_3. Y-component (not shown) is zero. (f) shows an example of ordinal depth (green in front, red behind) obtained by filling $\vec{u_{21}}$ using the segmentation of $\vec{u}_{23}$. (g) shows another example of ordinal depth (green in front, red behind) obtained by filling $\vec{u_{23}}$ using the segmentation of $\vec{u}_{21}$. (h) shows the *Class 2* moving object detected using the ordinal depth conflict. (i) shows the computed motion valley. (j) shows the structure from motion which puts the leopard in front of the red box, whereas occlusions (see (g)) show that the leopard is behind the box.

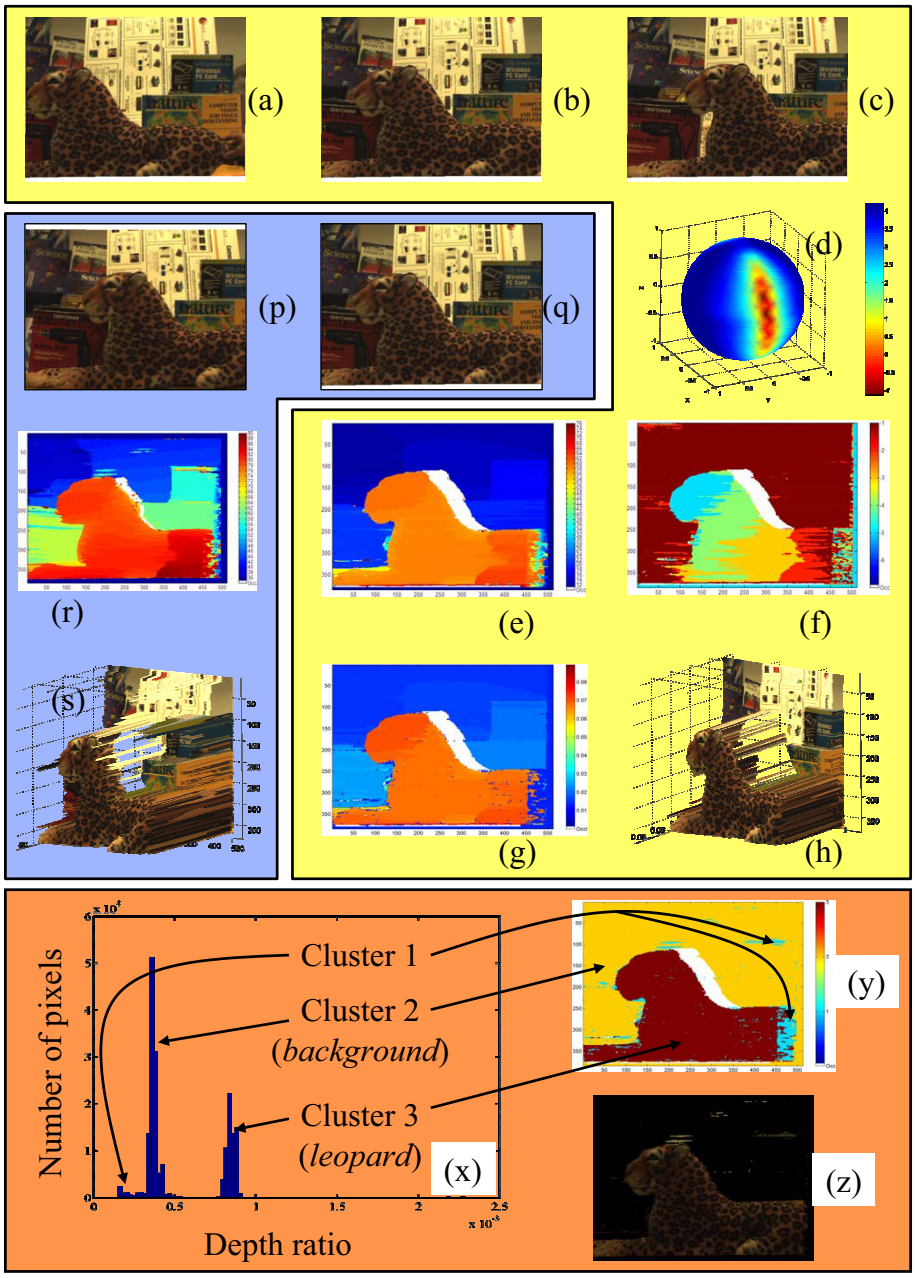

Figure 7.5: Class 3: (a,b,c) show three frames F_1, F_2, F_3 of the *leopardB* sequence. (d) shows computed motion valley. (e,f) show X and Y components of the flow $\vec{u}_{23}$ between F_2 and F_3. White regions denote occlusions (g) shows inverse depth from motion. (h) shows 3D structure from motion.
(p,q) show rectified stereo pair of images. (q) is the same as (b). (r) shows inverse depth from stereo. (s) shows 3D structure from stereo. Compare (s) with (h) to see how the background objects appear closer to the leopard in (s) than in (h).
(x) shows the histogram of depth ratios and clusters detected by k-means (k=3). (y) shows cluster labels: cluster 2 (yellow) is the background, cluster 3 (red) is the leopard, cluster 1 (light blue) is mostly due to errors in the disparity. (z) shows the moving objects of *Class 3* (clusters other than 2).

moving) background. Since the motion of the leopard is in the same valley as the background, *Class 1* detection (motion-based clustering) cannot succeed. Our method is able to detect the leopard as a *Class 2* moving object using conflicts between occlusions and structure from motion. A handwaving analysis indicates that the leopard is moving faster than the red box, hence the computed motion (which is predominantly a translation) naturally suggests that the leopard is in front of the box. However, the occlusions clearly suggest that the leopard is behind the box, thereby generating a conflict.

Finally, Figure 7.5 shows a situation in which the background is translating horizontally to the right, and the leopard is dragged horizontally towards the right. In this case, a single motion valley is found, the depth estimates are all positive, and no ordinal depth conflicts are present. (Although this case shows the simplest situation, we can also imagine the same situation as Figure 7.4, with the exception that the leopard does not move fast enough to the right so as to appear in front of the red box and generate an occlusion-motion conflict). In this case, depth information from stereo (obtained using a calibrated stereo pair of cameras) was compared with depth information from motion. k-means clustering (with k = 3) of the depth ratios was used to detect the background (the largest cluster). The pixels which did not belong to the background cluster are labeled as *Class 3* moving objects.

For the interested reader, Appendix B provides details about our mobile experimental setup which was designed to capture synchronized video from multiple cameras in the lab and outdoors.

7.7 Summary

We have classified moving objects into three classes, and discussed constraints for detecting each class of objects: *Class 1* is detected using motion alone, *Class 2* is detected using conflicts between ordinal depth from occlusions and depth from motion, while *Class 3* requires cardinal comparisons between depth from motion and depth from another source. The key contribution is the detection of *Class* 2 moving objects using the ordinal depth conflict. This tool is of general use in video processing and can be used in a variety of applications including video compression.

Chapter 8

Conclusion and Future Work

This thesis represents an attempt to highlight the interdependence of low level visual problems, such as correspondence, segmentation, occlusions and shape. We have argued that the solution of any one problem requires us to solve a host of related problems as well. In fact, such strong relationships seem to be a general feature of problems in artificial intelligence, which makes these problems difficult or sometimes impervious to modular analysis. Under these circumstances, modularity is often the enemy; compositionality, which seeks to mix problems and solve them together, seems to be the right approach. In the preceding chapters, we have discussed compositional solutions and algorithms for image correspondence and related problems. Let us briefly review the main ideas that were presented earlier, along with possible directions for future research.

The aim of establishing dense correspondence between the points in two images of the same scene is to obtain a piecewise continuous map relating the two point sets. In Chapter 2, we examined the dense correspondence problem in a world without slanted surfaces, in which it was sufficient to compute a piecewise constant map. In this Flatland of frontoparallel surfaces, we showed that correspondence and segmentation are chicken-and-egg problems which can be solved together using a compositional approach. This approach relied on matching the two images for different shifts and identifying and maximizing connected regions of matching pixels, subject to vertical connectivity restrictions. In this world, horizontal distances remained constant from one image to the next, thereby allowing us to correspond pixels instead of points. We used the uniqueness constraint to enforce a one-to-one matching between image pixels to find the occlusions.

In Chapter 3, we introduced shape into the problem; in particular, we showed that horizontally slanted surfaces project onto images of different width in the two cameras of a stereo system. This stretching and shrinking of horizontal line segments from one image to the other required us to change course from one-to-one pixel matching to matching intervals on corresponding rows, in order to allow N pixels in one image to match M pixels in the other image. Using unique interval matching, we were able to solve problems

concerning unequal sampling and lack of pixel uniqueness for finding occlusions. The resulting algorithm is a compositional effort to solve for image correspondence, segmentation and horizontal slant. Further extensions would include the use of a more complex model of shape than simple planes in order to model curved surfaces. On the experimental side, if biological vision systems also address the problems with horizontal slant, especially sampling differences, then we would expect to find binocular (simple) cells with left and right receptive fields tuned to different spatial frequencies and scales, and not just be phase shifted filters with the same spatial frequency. This may be an interesting avenue for further exploration by experimentalists.

In this chapter, we also showed how horizontal slant and horizontal depth discontinuities can be distinguished using occlusions, unlike vertical slant and vertical discontinuities. This forced us to alter our vertical consistency constraints, and argue about the difficulties of computing vertically smooth shape. Our approach thus far was able to explicitly model and estimate horizontal slant, but not vertical slant. Scene surfaces which are vertically slanted appear as discrete steps in the result instead of a smooth surface. We believe that future efforts need to focus on constraints regarding vertical slant; in this context, we find orientation disparities and contour continuity of non-horizontal edges to be promising candidates for obtaining vertically smooth shape. The resulting method would be a union of area based and edge based estimation methods. Shape from X methods such as shape from texture can also be integrated with stereo disparity in the vertical direction in order to obtain smooth surfaces. In Chapter 3, we also discussed the effects of shape on optical flow estimation and presented arguments about problems in finding the correct flow continuity constraints across edges. Much further investigation will be needed in order to truly understand the effects of shape in establishing correspondence in the presence of general motion. Knowledge of the egomotion will help considerably in enforcing the right continuity constraints, since the problem becomes similar to the stereo problem in this case. Therefore, feedback from motion estimation is an important direction for further exploration.

Until Chapter 4, we were using thresholded absolute intensity differences between the two images for various relative shifts to enable us to build connected components of matching regions, and using their size to select the correct correspondence for each pixel. However, in this chapter, we showed that connectivity between pixels is only a quick way

of diffusing local matching information about one pixel to other image pixels such that this information influences the decision of selecting the correct correspondence for these other pixels. We presented a general diffusion model which allowed us to use non-binary local matching metrics instead of binary thresholded intensity differences, and was relatively robust to parameter selection. Further work is needed to find a fast two dimensional generalization of the diffusion model. Another possible direction of research is to recast the diffusion model into a differential framework, thereby bridging the gap between differential and discrete methods. This would require a reformulation of the continuity constraints arising from shape in the differential context. From a larger perspective, we see that optical flow magnitudes can be broadly divided into three regimes: differential (subpixel), small (few pixels), and large (tens of pixels). In the differential case, we can use efficient energy extremization techniques to search our energy surface. The small regime forces us to search our energy surface exhaustively to avoid local extrema and find the best correspondences, while in the large regime, the size of the search space leaves us with no option but to compute gross quantities only by using techniques such as phase correlation. Future efforts must be directed at finding techniques which are able to deal with all the three regimes, instead of having different unrelated techniques for each case.

In Chapter 5, we used the diffusion model with alternative local matching metrics which were invariant to the contrast of the two images. One of the methods for contrast invariance performed local matching using filters tuned to different spatial frequency channels, thereby transforming the correspondence problem to a space-frequency problem, and adding signal processing to the compositional framework. Further investigation is needed to obtain improvements at depth discontinuities, especially in the presence of a noisy high frequency channel. Furthermore, the metric we developed combined the results of various frequency channels to estimate a single disparity at every point. However, it is possible to have multiple disparities at every point, which leads to the phenomenon of transparency. To model transparency, each frequency channel must be treated separately to compute different disparities at the same point. One possibility is to compute a disparity for each frequency channel and combine the results by using a voting mechanism. The votes would not only weigh the perceived disparities, but also the texture perceived to be present at each disparity. We have also found that the problem of transparency is further complicated by the presence of shape, leading to multiple scene models (one with trans-

parency, the other with slant) which explain the same scene. Preliminary experiments by presenting such ambiguous scenes to human observers seem to indicate that the human observer prefers the solution which has a lesser degree of segmentation (usually the transparent solution), and if both explanations yield the same segmentation, then the shape model is preferred. However, much further work is needed to fully understand the complexities associated with transparency.

In Chapter 6, we demonstrated a method of filling occluded regions with flow values from neighboring regions by using segmentation information from previous frames. This method can successfully fill occlusions even in the presence of independently moving objects in the scene. We then showed that if occlusions can be filled, the ordinal depth order of neighboring scene regions can be obtained. This method showed us the need for maintaining knowledge of the segmentation across time in order to compute dense correspondence in the entire image, including the occluded regions. In fact, future research also needs to focus on ways of solving for the optical flow across longer time intervals by further developing such segmentation consistency constraints. From an application standpoint, knowing the disparities or the flow in occluded regions is especially important for applications such as image based rendering, which interpolate two views based on the correspondence. It is also important for motion estimation modules in video compression algorithms, which generally face problems near discontinuities and occlusions. Knowledge of ordinal depth in a general motion scenario can prove to be very useful for video manipulation applications which add virtual objects inside a real scene.

In Chapter 7, we used ordinal depth from occlusions to help solve another compositional set of problems - moving object detection, egomotion estimation and structure estimation. We have discussed the ambiguities involved in egomotion estimation. Knowledge of ordinal depth provides an additional constraint which can be used to narrow the ambiguity considerably. It also allows us to detect a new category of moving objects based on conflicts between structure from motion and ordinal depth from occlusions. This new category of moving objects is frequently encountered in scenarios such as automotive vision, and cannot be detected by using traditional motion clustering methods.

Feedback from motion estimation to optical flow computation is an important avenue for future research. In fact, as we have mentioned earlier, if the egomotion is known, the difficulties arising in the computation of optical flow in the presence of shape are allevi-

ated, since the problem becomes similar to stereo matching and the ambiguity in defining flow continuity constraints across edges is removed. Hence, it is not just that optical flow and occlusions are used to solve the motion segmentation and egomotion problem, but knowledge of the egomotion also lets us impose correct continuity constraints during optical flow estimation. Thus, the dependence exists in both directions.

All these chapters together paint a picture which reveals some of the relationships and dependencies between correspondence, segmentation, shape, occlusions, signals, structure and 3D motion. We have presented robust correspondence algorithms which find discontinuities, deal with untextured regions, handle slanted surfaces, find occlusions, and are able to perform in the presence of large non-uniform changes in contrast between two images. Possible immediate avenues for further research have been described above. The compositional philosophy can be extended and applied to many other sets of related problems. In fact, even in the most abstract sense, the sensation, cognition and action faculties of living things appear to be intimately related and perhaps inseparable entities. In other words, feedback and compositionality seem to be omnipresent in the brain.

Appendix A

Phase correlation

In this appendix, we review the technique of phase correlation discussed in [RC96], which is generally used for video stabilization, and has also been used by us in Chapter 7 for initializing motion segmentation.

A.1 Basic process

Consider an image which is moving in plane, i.e. every point on the image has the same flow. Thus, if the image $I_2(x, y)$ is such a translated version of the original image $I_1(x, y)$, then we can use phase correlation to recover the translation in the following manner:

If $I_2(x, y) = I_1(x + t_x, y + t_y)$, then their Fourier transforms are related by:

$$f_2(\omega_x, \omega_y) = f_1(\omega_x, \omega_y) e^{-i(\omega_x t_x + \omega_y t_y)} \tag{A.1}$$

The phase correlation $PC(x, y)$ of the two images is then given by:

$$PC(x, y) = F^{-1}\left[\frac{f_1^* \cdot f_2}{|f_1^* \cdot f_2|}\right] = F^{-1}\left[e^{-i(\omega_x t_x + \omega_y t_y)}\right] \tag{A.2}$$

$$PC(x, y) = \delta(x - t_x, y - t_y) \tag{A.3}$$

Note that F^{-1} is the inverse Fourier transform, and δ is the delta function. Thus, if we use phase correlation, we can recover this global image translation (t_x, t_y) since we get a peak at this position (see example in Figure A.1).

A.2 Logpolar coordinates

The logpolar coordinate system (s, γ) is related to the cartesian coordinates (x, y) by the following transformation:

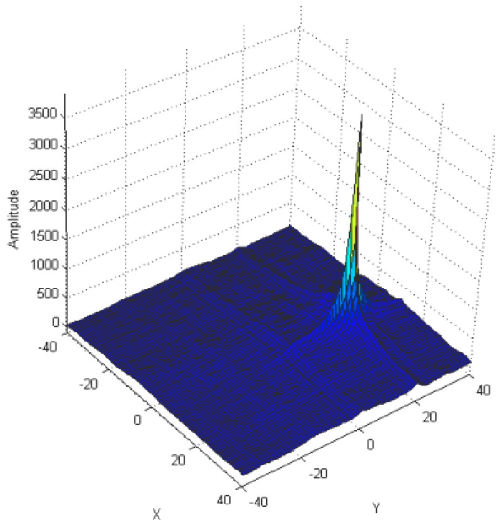

Figure A.1: An example of a peak generated by the phase correlation method.

$$x = e^{\rho} cos(\gamma) \tag{A.4}$$

$$y = e^{\rho} sin(\gamma) \tag{A.5}$$

If the image is transformed into the logpolar coordinate system (see Figure A.2), then changes in scale and rotation about the image center in the cartesian coordinates are transformed into translations in the logpolar coordinates. Hence, if we perform the phase correlation procedure mentioned above in the logpolar domain, we can also recover the scale change s, and a rotation about the center γ, between two images.

A.3 Four parameter estimation

Given two images which are related by 2D translation, rotation about the center and scaling all together, we can perform phase correlation both the cartesian domain and logpolar domains to compute a four parameter transformation *T* between the two images:

$$T = \begin{bmatrix} s \cdot cos(\gamma) & s \cdot sin(\gamma) & t_x \\ -s \cdot sin(\gamma) & s \cdot cos(\gamma) & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{A.6}$$

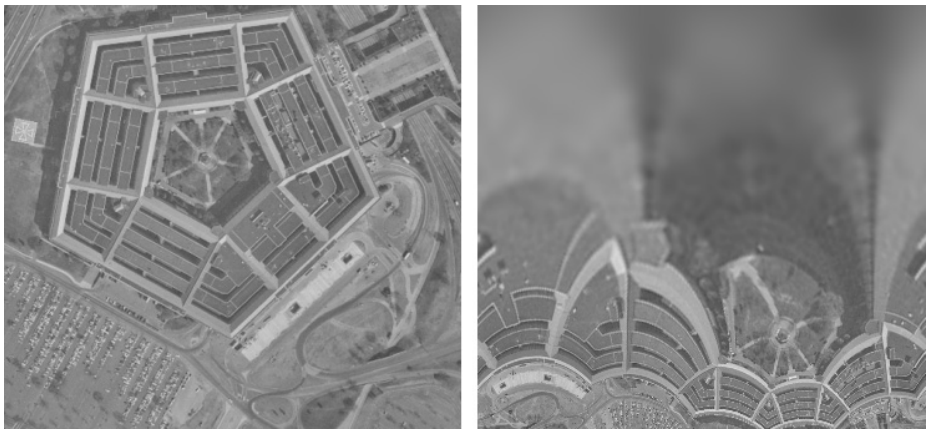

Figure A.2: An image in cartesian coordinates (left) and its logpolar representation (right).

In practice, initializing this process is tricky, since dominant 2D translation will cause problems in the logpolar phase correlation by introducing many large additional peaks, and similarly, dominant scaling and rotation will cause problems in the cartesian phase correlation.

To address this, we first perform phase correlation in both the cartesian and logpolar representations on the original images. Then, for each of the results, we find the ratio of the magnitude of the tallest peak to the overall median peak amplitude. If this ratio is greater for the cartesian computation, it means that translation is dominant over scaling and rotation, and must be removed first. Then we can estimate scaling and rotation again on the corrected images. Similarly, if the ratio is greater for the logpolar computation, we perform the correction the other way around.

A.4 Results

Figure A.3 shows results on a pair of test images which are related by significantly large values of translation, rotation and scaling. These results can be improved to subpixel accuracy by using the method of Foroosh et al [FZB02]. We have applied the method mentioned above to video sequences and have achieved good stabilization over long durations, even in the presence of independently moving objects. Some results can be found at *www.cfar.umd.edu/users/ogale/research/phase/phase.html.*

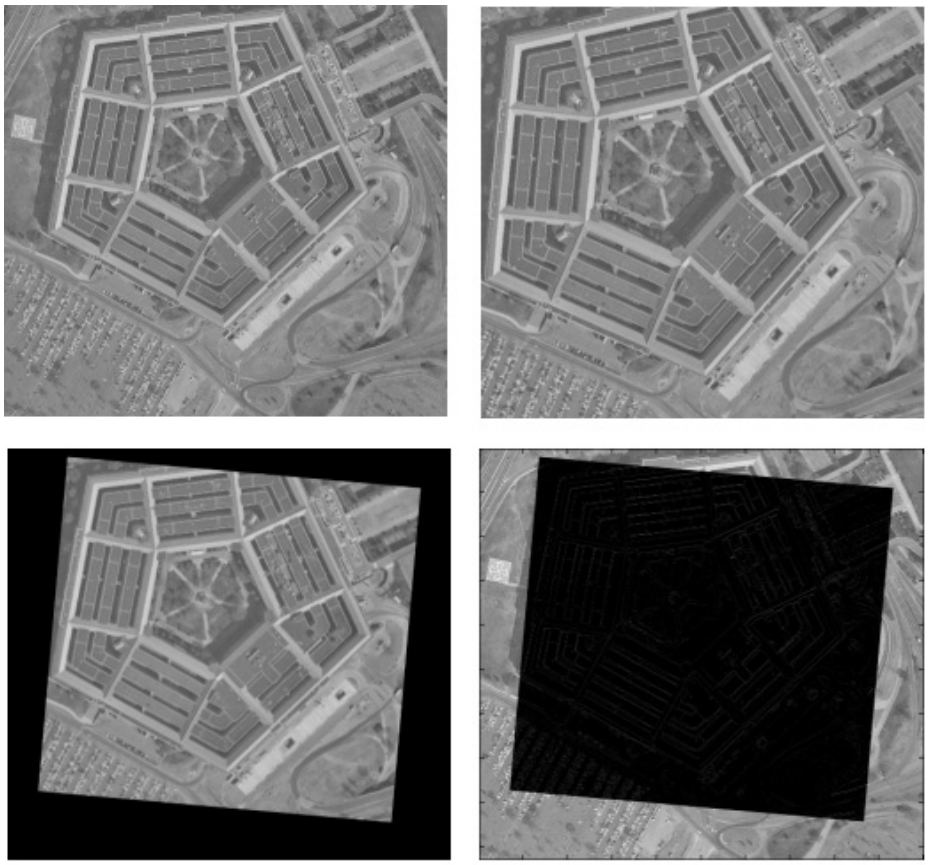

Figure A.3: Top row shows two input images I_1 and I_2. Image I_2 was created from I_1 by rotating by 5 degrees, scaling by a factor of 1.2, and translating by (-10,20) pixels. Bottom row: The left image shows image I_2' obtained by unwarping I_2 using the results of the phase correlation. The right hand side shows the absolute intensity difference between I_1 and the unwarped image I_2' to reveal the accuracy of the registration.

Appendix B

Design of a mobile Argus Eye with nine synchronized cameras

Camera networks are becoming an important tool for use in computer vision. In order to study these networks, it is necessary to engineer systems which can handle the high bandwidths of multiple cameras with inexpensive off-the-shelf hardware and open source software. This chapter explains the engineering of a mobile system designed to synchronously capture nine IEEE1394 digital cameras at a resolution of 1024x768 at 15 fps. This work was performed in collaboration with Patrick Baker (*pbaker@cfar.umd.edu*). My contribution to this work is the development of the complete capture software system, the details of which are described below.

B.1 Introduction

Camera networks have great potential for applications of computer vision to surveillance, model building, navigation, and many others. Before they become widespread, the problems of data acquisition, calibration, synchronization, and image processing must be solved in a way that allows for easy construction, use, and maintenance. This paper addresses these issues and provides new solutions for use by the vision community.

The IEEE-1394 (firewire) bus is fast becoming the standard for tranferring data from a camera to the computer because it can transfer up to 400 Mb/s, and also suuply power over the same cable. We describe here a system engineered to capture data from many synchronized firewire cameras directly to hard drives, which fully utilizes the bus bandwidth, while using virtually no CPU, due to the bus mastering capabilities of the firewire and SCSI boards. The code runs under Linux with the standard libraries `video1394` and `dc`, the camera control library.

B.2 Project Requirements

The theory behind the Argus eye, which is an omnidirectional camera, and the motivation for building such a camera system for accurate egomotion estimation can be found in our

recently published work [BOFA03]. The camera system for this purpose was required to facilitate the synchronized capture of different parts of the visual sphere using multiple cameras. It was also required to function outdoors to capture natural scenes, and therefore could not use multiple computers or other cumbersome hardware. Since we intended to capture sequences of durations of the order of 1 minute, we could not design a system which captured data to RAM, and then store it on the disk after capture; the data had to be written directly to disk.

Additionally, we would need to have at least nine cameras pointing in different directions, with fields of view not necessarily overlapping. We needed a camera which could achieve at least 25 fps so that we could avoid temporal aliasing when taking motion video. More cameras were desirable, if the hardware platform could support it. We desired to be able to carry the network of cameras on a human being, to facilitate the capture of data in an unconstrained manner. For that reason the cameras had to be light and relatively free of cables.

For the computation of egomotion, the cameras needed to have a reliable syncrhonization system so that we could control the capture of the data. We also desired a flexible (in the figurative sense) and rigid (in the mechanical sense) mounting system so that different configurations could be tested without much camera movement.

B.3 Hardware Configuration

B.3.1 Cameras

We chose the Sony XCD-X700 camera, which is a `1024X768` 8-bit black and white camera with digital output and control through the firewire port. The camera has an external sync BNC connector to provide an external synchronization signal. The camera unfortunately is only rated for 15 fps at the maximum resolution. However, the camera has a partial scan function which allows an increase over the nominal 15 fps maximum in inverse proportion to the fraction of scan lines captured, and could yield a maximum framerate of upto 60 fps. This camera therefore was a good compromise among portability, cost, resolution, and framerate.

There is one detail about the camera which was not apparent from a cursory overview of its features. The external trigger reduces the framerate of the camera depending on

the exposure setting. When the external trigger is activated, the camera does not send data during its exposure, so that when using long exposure settings, the 15 fps rate is not achievable. So for a 10 ms exposure, only 13 fps will be achievable. One can bring up the gain in order to compensate for low light, but in that case noise will be increased. However, under natural illumination conditions available outdoors, the exposure can be set to a fairly small value.

When using the partial scan function, the external trigger is required, so you cannot achieve 60 fps. There are additional timing requirements for partial scan which reduce the framerate even more, so that the fastest we could get the cameras to run was 48 fps when scanning only the top quarter of the image. Another notable factor is that the partial scan function also only increases the framerate according to the number of scan lines which are output, without regard to the image width.

B.3.2 Computer configuration

The bandwidth requirements for this computer are substantial, and dominated our design decisions. A quick calculation of the bandwidth required for each camera running at 1024x768 and 15 fps results in a figure of 11.25 MB/s, which means that nine cameras would require 101.25 MB/s of bandwidth from the cameras to the disks. Because of the external trigger limitations, the likely framerate would be closer to 14 fps, so that our bandwidth requirements will be 94.5 MB/s.

A 32-bit PCI bus running at 33 Mhz has a maximum bandwidth of 133 MB/s, so that was barely sufficient for transferring the data from the cameras to memory. In order to get the data from memory to disk, we needed another PCI bus to host the disk controllers. The processor speed is not an essential parameter, because we used DMA (Direct Memory Access) for all data transfers and are doing no real-time image processing.

The Dell Precision 620 is an 866 Mhz Pentium III system with two PCI buses, one 32-bit running at 33 Mhz/5V and one 64-bit, running at 66 Mhz/3.3V. This system has sufficient bandwidth to move the data from the cameras to the disks. We chose four Seagate SCSI 15000 RPM disks in order to write the data. We have benchmarked these disks to have a maximum write throughput of about 33 MB/s, so that four disks were sufficient to write the data. As an engineering note, if one writes to the disks efficiently, by making sure there is no extraneous disk head movement and the writes are all consecutive and cached, the

seek speed of the disks should not make a difference, and a less expensive SCSI disk with the same write throughput should be sufficient.

Although the Precision 620 has an onboard Ultra 160 SCSI controller, with a stated burst throughput of 160 MB/s, it is attached to the 32-bit PCI bus. This means that we are limited to the 133 MB/s of the PCI bus, and we only managed to get 80 MB/s sustained throughput. Even if we could use this controller, this would mean that we could only mount firewire cards in the two 64-bit PCI slots. And since each firewire card has only three free DMA channels, this means that we would only be able to use DMA with six cameras, which would not meet our system requirement.

We chose two Ultra-2 SCSI controllers running on the 64-bit bus in order to control our disks. Though these controllers have a stated burst capacity of 80 MB/s, we could get only a maximum of 45 MB/s from them, when writing to two or more of the Seagate disks. The IDE disk which hosted the operating system provided some extra disk bandwidth which we have utilized in the finished system when capturing with all nine cameras.

B.3.3 Sync Network

The BNC network to synchronize the cameras is attached to the parallel port of the computer with an amplifier to make sure the cameras do not draw too much current. It was desireable for some of our experiements to be able to run different cameras at different frame rates, so that we used multiple pins of the parallel port. The camera trigger control is discussed more extensively in the software configuration section of the code. The BNC cabling is always somewhat problematic, because the connections between cables are inconsistent, but we found no other triggering solution which is easier and economical.

B.4 Why Linux?

Generally, other synchronized camera systems utilize a trigger generator which is external to the PC, and hence the triggering process in not harmonious with the camera capture process of the PC. If we try to extract the best performance from such a system, it results in intermittent dropped frames for some of the cameras. We were previously experimenting with such a system, which utilized an external trigger generator, and where the grabbing process was performed using the GraphEdit tool in the DirectX SDK from Microsoft. We

tested the SCSI disks in two write modes: as a single striped volume, and as raw devices (which performed better). We were able to grab a maximum of eight cameras with this system, at full resolution at about 7-8 fps. Thus, we were nowhere near the performance level which we knew this system was capable of producing. This system also performed very poorly with regards to synchronization and resulted in dropped frames for some of the cameras. The DirectX architecture did not offer us sufficient flexibility to incorporate the trigger signals into our code directly, and was overall very restrictive in allowing us to control our hardware at a low level. The final limitation was that the DirectX system was unable to use the XCD-X700 cameras in partial mode, and we were always restricted to grabbing at full resolution. Partial mode capture was essential for application to ego-motion problems due to the higher framerates (30fps, 45fps) which were achievable in this mode. After experiencing all these limitations in a Windows based approach, it was decided to turn to Linux, which offered a significant increase in the level of control over the hardware. Moreover, the open source nature of the system allowed us to make critical changes (fix some bugs) to the dc1394 libraries, and adapt the sg scsi writing code to suit our particular purposes.

B.5 Capture Software design

B.5.1 Hardware detection and initialisation

At the outset, all the firewire cards attached to the system are detected, and the number of cameras on each card and their capabilities are assessed. Each camera has a unique ID stored in it which can be associated with a user-assigned label to be used to identify and sort the cameras for the convenience of the user. The user supplies an initialisation (INI) file to the software, which specifies camera configurations, capture modes, trigger frequencies, and many other critical paramters which will be discussed in a subsequent section. Camera parameters, such as the shutter, gain, capture mode (full/partial), and optional partial mode parameters (left, top, height, width) are set for each camera by using the values in the INI file. The trigger mode (internal/external) is also set during initialisation. Finally, the cameras are assigned DMA channels and buffers for transmitting their data, and are then put into isochronous transmission mode. The cameras are now ready to start capturing the data.

The capture process itself consists of three basic modules:

1. Trigger: send trigger signals to the cameras
2. Grabber: Transfer captured images to memory
3. Writer: Transfer images from memory to the disks

To obtain the maximum efficiency, the Writer module functions in parallel with the first two modules. Thus, when the Grabber is transferring new frames from the cameras to memory, the Writer is simultaneously writing the previously captured frames to the disks, which makes it possible to transfer data at about 100 MB/s from the cameras to the disks. All the modules have been implemented as C++ classes.

B.5.2 Trigger

Triggers were to be sent to the parallel port of the PC, which provides eight independent pins. Each pin can be sent a separate trigger signal. In order to design a system with the maximum flexibility, it was essential that each camera be allowed to function in any one of the three partial capture modes: full height, 1/2 height, and 1/4 height. In such a scenario, the 1/4 height cameras would need triggers at 4 times the frequency of the triggers being sent to the full height cameras. If we just choose to send a single high frequency trigger to all the cameras, it would result in synchronization errors for the low frequency, full height cameras. Therefore, we had to make provisions to allow separate triggers to be sent to each of the three potential groups of cameras: full, 1/2 and 1/4 height. Moreover, the triggers must be synchronized, eg. every fourth trigger for the 1/4 height cameras must coincide with a trigger sent to the full height cameras. In order to achieve this, we allowed the user to select the trigger frequency to be sent to each pin of the parallel port. The user also specifies which pin of the parallel port is connected to which camera, and the system is then able to verify whether the correct pins have been paired with the correct camera modes. Before sending the trigger signal, the Grabber initiates a separate processing thread for each camera which listens for interrupts from that camera. After sending the trigger signal, the cameras begin to send interrupts, which are captured by the listening threads previously created by the Grabber.

B.5.3 Grabber

The Grabber module utilizes the dc1394 library to perform the camera capture process. The library had to be slightly modified in order to correctly perform partial mode capture. The Grabber itself is designed as a multithreaded module, which creates separate threads for capturing from each camera. Such a multithreaded capture process is much more efficient than the native single-threaded multi-camera capture function implemented in the dc1394 library. The multithreaded method permits simultaneous DMA transfers from many cameras, and is able to fully utilize the available bandwidth on the bus without consuming any CPU resources. Although we have allowed each camera to function in one of three partial modes: full, 1/2 and 1/4 height, all the cameras are on the average sending data at the same rate, because if full frame cameras run at X triggers/second, then 1/2 and 1/4 height cameras run at 2X and 4X respectively.

B.5.4 Writer

The Writer module is designed to write the output of two cameras to each SCSI disk, and the output of one camera to an IDE disk, giving a net throughput of about 108 MB/s, which is close to the maximum possible efficiency for this system. Since we were using two Ultra80 SCSI cards which actually yield a throughput of 45 MB/s each, it was necessary to use the IDE disk to scavenge the additional bandwidth. If Ultra160 cards were used, the IDE disk would not be required. We have used small portions of code from the sg driver in linux, and adapted them to write raw data to the SCSI disks using DMA. The Writer module is also multithreaded, and uses a separate thread for writing to each disk, which ensures that we obtain the maximum write performance without using any CPU resources. Since the data is captured directly to disk, our system can capture about 15 minutes of data from 9 cameras running at 1024x768, 15 fps.

Let us now discuss how the writes are scheduled, if different cameras are capturing at different rates and different image heights. If a camera capturing full height frames sends X bytes of data every T seconds, then

1. cameras capturing at 1/2 height send (X/2 bytes of data, 2 times, in T seconds) = (X bytes of data in T seconds)

2. cameras capturing at 1/4 height send (X/4 bytes of data, 4 times, in T seconds) = (X bytes of data in T seconds)

Thus, irrespective of the capture height, each camera sends X bytes of data in T seconds. This uniformity makes it possible to write data to the disks in an orderly fashion: every T seconds, 2X bytes of data (from 2 cameras) is written to one scsi disk. Thus, if a camera was running at 1/4 height, then 4 frames from that camera are written as one block to the scsi disk. In this manner, the write scheduling process is greatly simplified.

B.5.5 User input

The user can specify the details of the entire capture process by providing an initialisation (INI) file. We will only mention some of the important parameters which are to be specified in this file. The CaptureTime, which specifies the grabbing time in seconds, and Trigger, which sets the trigger mode to external or internal, are the important common parameters specified in the Common section. In the Trigger section, the frequencies to be sent to each pin of the parallel port are specified as multiples of the base frequency (which is taken to be 1). These two sections are followed by sections for each camera, which specify the Gain, Shutter, and partial mode parameters (top, height) for each camera. An example INI file for a stereo pair of cameras is shown in Figure B.1.

The INI file is also used by the capture software to record the results of the capture process in a section known as Results. This section contains information regarding how the data was saved to the raw disks and the number of frames with their sizes captured by each camera. This information is used by the accompanying utilities, such as scsi2pgm, which converts the raw captured frames to PGM files.

B.5.6 Utilities for focusing, calibration, viewing and frame extraction

The software contains a very useful tool called *focuscams*, which displays the real-time video streams from all the cameras (upto nine cameras) in a tiled format on the screen. The user can also choose to view the video from any one of the cameras in full screen mode, in order to focus the cameras one by one. The tiled display is useful to examine the fields of view being seen by the cameras, and to equalise parameters such as the exposure of all the cameras. This utility can also be used for calibration purposes to grab single frames

```
; Sample camera capture file

[Common]
SequenceName=test
CaptureTime=120 ; this is the capture time in seconds
NumCards=4 ; number of firewire cards to scan (0-10)
Trigger=External ;
DisablePartialMode=0; to force full mode for all cams
ExtraDelay=0;
BeepOnTrigger=0;
AutoOffset=1 ; whether to use AutoOffsetFile
AutoOffsetFile=offsets.dir
Debug=0 ; to cross check input parameters
PinCheck=1 ; to check if pin frequencies are compatible with cameras

[Trigger]
Pin0=1
Pin1=1
Pin2=1
Pin3=1
Pin4=1
Pin5=1
Pin6=1
Pin7=1

[Camera 0]
Enabled=1
Gain=800
Shutter=A80
UsePartialMode=1
partialTop=0
partialHeight=4
PinConnect=0

[Camera 1]
Enabled=1
Gain=800
Shutter=A80
UsePartialMode=1
partialTop=0
partialHeight=4
PinConnect=0
```

Figure B.1: Sample INI file for 2 cameras

from the cameras by pressing a key. Another tool, known as *viewscsi*, allows the user to view the raw results of a capture directly from the SCSI disks, without first converting the data to PGM files. This is useful in order to immediately examine if a captured sequence is satisfactory, and if not, it can be immediately deleted, so that more space is freed up for capture. Finally, *scsi2pgm* is another utility which converts the captured data into a set of PGM/PPM files, which can then be offloaded for further processing.

B.6 Applications and extensions

This system has been used by Patrick Baker and myself to acquire indoor and outdoor data for egomotion estimation using an omnidirectional camera (see Figure B.2). Although the system was designed for the Argus problem, it was also used to gather other types of data by graduate students from our lab. It has been used by me for obtaining sequences for testing correspondence and motion segmentation algorithms, and by Jan Neumann (*jneumann@cfar.umd.edu*) to simulate a plenoptic camera in a hexagonal configuration. It is currently being used by Gutemberg Bezerra (*guerra@cs.umd.edu*) for human motion capture from multiple views, and by Justin Domke (*domke@cs.umd.edu*) for collecting Argus data. This system is currently designed to run on a single computer, but can be easily extended to multiple computers by communicating the trigger over the parallel port from a master computer to other machines. This is, however, in the domain of future work.

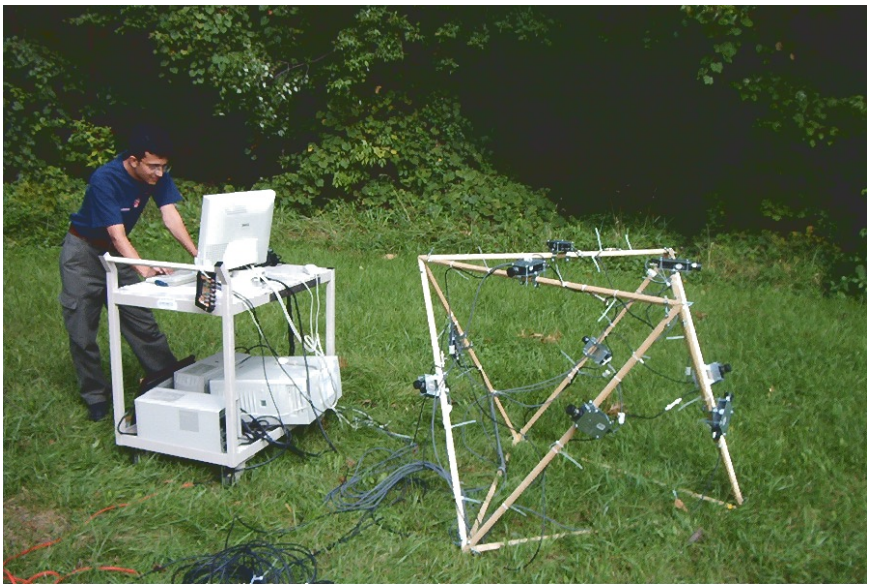

Figure B.2: Argus eye system being used outdoors to collect data. The cameras are mounted on the octahedral frame, and carried around by a person who stands in the middle holding the frame.

Bibliography

[AB85] E.H. Adelson and J.R. Bergen. Spatiotemporal energy models for the perception of motion. *J. Opt. Soc. Am.*, 2(2):284–299, 1985.

[Adi85] G. Adiv. Determining 3D motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:384–401, 1985.

[Adi89] G. Adiv. Inherent ambiguities in recovering 3-D motion and structure from a noisy flow field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:477–489, 1989.

[ADPS02] L. Alvarez, R. Deriche, T. Papadapoulo, and J. Sanchez. Symmetrical dense optical flow estimation with occlusion detection. *ECCV*, page I: 721 ff., 2002.

[Alo88] J. Aloimonos. Shape from texture. *Biological Cybernetics*, 58:345–360, 1988.

[AOF97] A. Anzai, I. Ohzawa, and R.D. Freeman. Neural mechanisms underlying binocular fusion and stereopsis: position vs. phase. *Proc. Natl. Acad. Sci. USA, Neurobiology*, 94:5438–5443, May 1997.

[ASB94] S. Ayer, P. Schroeter, and J. Bigün. Segmentation of moving objects by robust motion parameter estimation over multiple frames. In *Proc. Third European Conference on Computer Vision*, pages 316–327. Springer-Verlag, 1994.

[BA89] D. Blostein and N. Ahuja. Shape from texture: integrating texture-element extraction and surface estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1233–1251, 1989.

[Bar89] S. T. Barnard. Stochastic stereo matching over scale. *IJCV*, 3(1):17–32, 1989.

[BB95] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–467, 1995.

[BB00] S.S. Beauchemin and J.L. Barron. On the fourier properties of discontinuous visual motion. *Journal of Mathematical Imaging and Vision*, 13(3):155–172, 2000.

[BBH+89] P. J. Burt, J. R. Bergen, R. Hingorani, R. Kolczynski, W. A. Lee, A. Leung, J. Lubin, and H. Shvaytser. Object tracking with a moving camera. In *Proc. IEEE Workshop on Visual Motion*, pages 2–12, 1989.

[BF00] M.J. Black and D.J. Fleet. Probabilistic detection and tracking of motion discontinuities. *IJCV*, 38(3):231–245, 2000.

[BFA00] T. Brodský, C. Fermüller, and Y. Aloimonos. Structure from motion: Beyond the epipolar constraint. *International Journal of Computer Vision*, 37:231–258, 2000.

[BFB94] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77, 1994.

[BFBB94] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt. Performance of optical flow techniques. *CVPR*, 92:236–242, 1994.

[BI99] A. F. Bobick and S. S. Intille. Large occlusion stereo. *IJCV*, 33(3):181–200, Sept 1999.

[BK94] M. Bober and J. Kittler. Robust motion analysis. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 947–952, 1994.

[BL76] R. Bajcsy and L. Lieberman. Texture gradient as a depth cue. *Computer Graphics and Image Processing*, 5:52–67, 1976.

[BOFA03] P. Baker, A.S. Ogale, C. Fermuller, and Y. Aloimonos. New eyes for robotics. *International Conference on Intelligent Robots and Systems*, 1:1018–1023, Oct 2003.

[BT98] Stan Birchfield and Carlo Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Trans. PAMI*, 20(4):401–406, 1998.

[BT99] Stan Birchfield and Carlo Tomasi. Multiway cut for stereo and motion with slanted surfaces. *ICCV*, 1:489–495, 1999.

[BVZ98] Y. Boykov, O. Veksler, and R. Zabih. A variable window approach to early vision. *IEEE Trans. PAMI*, 20(12):1283–1294, Dec 1998.

[BVZ01] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. PAMI*, 23(11):1222–1239, Nov 2001.

[CK95] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *Proc. International Conference on Computer Vision*, pages 1071–1076, 1995.

[CR93] R. Caganello and B.J. Rogers. Anisotropies in the perception of sterescopic surfaces: the role of orientation disparity. *Vision Research*, 33(16):2189–2201, 1993.

[Dau85] J.G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J. Opt. Soc. Am. A*, 2:1160–1169, 1985.

[DOF91] G.C. DeAngelis, I. Ohzawa, and R.D. Freeman. Depth is encoded in the visual cortex by a specialized receptive field structure. *Nature*, 352:156–159, 1991.

[DS97] K. Daniilidis and M. E. Spetsakis. Understanding noise sensitivity in structure from motion. In Y. Aloimonos, editor, *Visual Navigation: From Biological Systems to Unmanned Ground Vehicles*, Advances in Computer Vision, chapter 4. Lawrence Erlbaum Associates, Mahwah, NJ, 1997.

[EW02] G. Egnal and R.P. Wildes. Detecting binocular half-occlusions: empirical comparisons of five approaches. *IEEE Trans. PAMI*, 24(8):1127–1133, Aug 2002.

[FA00] C. Fermüller and Y. Aloimonos. Observability of 3D motion. *International Journal of Computer Vision*, 37:43–63, 2000.

[Fau93] O. D. Faugeras. *Three-Dimensional Computer Vision*. MIT Press, Cambridge, MA, Nov 1993.

[FJJ91] D. Fleet, A. Jepson, and M. Jenkin. Phase-based disparity measurement. *CVGIP: Image Understanding*, 53:198–210, 1991.

[Fle94] D.J. Fleet. Disparity from local weighted phase-correlation. *IEEE International Conference on SMC*, pages 48–56, October 1994.

[FO90] R.D. Freeman and I. Ohzawa. On the neurophysiological organization of binocular vision. *Vision Research*, 30:1661–1676, 1990.

[FRT97] A. Fusiello, V. Roberto, and E. Trucco. Efficient stereo with multiple windowing. *CVPR*, pages 858–863, June 1997.

[FZB02] H. Foroosh, J.B. Zerubia, and M. Berthod. Extension of phase correlation to subpixel registration. *IEEE Transactions on Image Processing*, 11(3):188–200, Mar 2002.

[GG84] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. PAMI*, 6(6):721–741, Nov 1984.

[GK89] B. Girod and D. Kuo. Direct estimation of displacement histograms. *OSA Meeting: Image Understanding and Machine Vision*, pages 73–76, 1989.

[GLY92] D. Geiger, B. Ladendorf, and A. Yuille. Occlusions and binocular stereo. *ECCV*, pages 425–433, 1992.

[GMN+98] Ben Galvin, Brendan McCane, Kevin Novins, David Mason, and Steven Mills. Recovering motion fields: An evaluation of eight optical flow algorithms. *Proceedings of the British Machine Vision Converence*, Sept 1998.

[GR92] B. Gilliam and C. Ryan. Perspective, orientation disparity, and anisotropy in sterescopic slant perception. *Perception*, 21:427–439, 1992.

[HJ92] D. J. Heeger and A. D. Jepson. Subspace methods for recovering rigid motion I: Algorithm and implementation. *International Journal of Computer Vision*, 7:95–117, 1992.

[HN94] T.S. Huang and A.N. Netravali. Motion and structure from feature correspondences: a review. *Proceedings of the IEEE*, 82(2):252–268, Feb 1994.

[HW68] D. Hubel and T. Wiesel. Receptive fields and functional architecture of the monkey striate cortex. *Journal of Physiology*, 195:215–243, 1968.

[HZ00] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[IA98] M. Irani and P. Anandan. A unified approach to moving object detection in 2d and 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:577–589, 1998.

[JJ88] M. Jenkin and A. Jepson. *Computational process in Human Vision*, chapter The measurement of binocular disparity. (ed.) Z. Pylyshn, Ablex Press, NJ, 1988.

[JP87] J.P.Jones and L.A. Palmer. The two-dimensional spatial structure of simple receptive fields in the cat striate cortex. *J. Neurophysiology*, 58:1187–1211, 1987.

[Jul71] B. Julesz. *Foundations of Cyclopean Perception*. University of Chicago Press, Chicago., 1971.

[KH75] C. Kuglin and D. Hines. The phase correlation image alignment method. *Proc. IEEE Int. Conf. Cyber. Soc*, pages 163–165, 1975.

[KO94] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: theory and experiment. *IEEE Trans. PAMI*, 16(9):920–932, 1994.

[KZ01] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. *ICCV*, pages 508–515, July 2001.

[LHH+98] H. Liu, T.H. Hong, M. Herman, T. Camus, and R. Chellappa. Accuracy vs efficiency trade-offs in optical flow algorithms. *Computer Vision and Image Understanding*, 72(3):271–286, 1998.

[LT03] Michael Lin and Carlo Tomasi. Surfaces with occlusions from layered stereo. *CVPR*, 1:I–710–I–717, June 2003.

[LZ03] G. Li and S. Zucker. A differential geometrical model for contour based stereo correspondence. *IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision, Nice, France*, October 2003.

[Mar80] S. Marcelja. Mathematical description of the responses of simple cortical cells. *J. Opt. Soc. Am. A*, 70:1297–1300, 1980.

[May87] S. J. Maybank. *A Theoretical Study of Optical Flow*. PhD thesis, University of London, 1987.

[MB96] A. Mitiche and P. Bouthemy. Computation and analysis of image motion: a synopsis of current problems and methods. *IJCV*, 19(1):29–55, July 1996.

[MD00] Jane Mulligan and Kostas Daniilidis. Predicting disparity windows for real-time stereo. *Lecture Notes in Computer Science*, 1842:220–235, 2000.

[MM90] G.J. Mitchison and S.P. McKee. Mechanisms underlying the anisotropy of sterescopic tilt perception. *Vision Research*, 30(11):1781–1791, 1990.

[MP79] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proc. Royal Soc. London B*, 204:301–328, 1979.

[MSKS04] Y. Ma, S. Soatto, J. Kosecká, and S.S. Sastry. *An Invitation to 3-D Vision*, volume 26 of *Interdisciplinary Applied Mathematics*. Springer, 2004.

[Nel91] R. C. Nelson. Qualitative detection of motion by a moving observer. *International Journal of Computer Vision*, 7:33–46, 1991.

[NMF90] M. Nomura, G. Matsumoto, and S. Fujiwara. A binocular model for the simple cell. *Biological Cybernetics*, 63:237–242, 1990.

[NNPT98] O. Nestares, R. Navarro, J. Portilla, and A. Tabernero. Efficient spatial-domain implementation of a multiscale image representation based on gabor functions. *J. Electronic Imaging*, 7:166–173, 1998.

[OA04] Abhijit S. Ogale and Yiannis Aloimonos. Stereo correspondence with slanted surfaces: critical implications of horizontal slant. *CVPR*, June 2004.

[OB95] J.-M. Odobez and P. Bouthemy. MRF-based motion segmentation exploiting a 2D motion model and robust estimation. In *Proc. International Conference on Image Processing*, volume III, pages 628–631, 1995.

[ODF90] I. Ohzawa, G.C. DeAngelis, and R.D. Freeman. Stereoscopic depth discrimination in the visual cortex: neurons ideally suited as disparity detectors. *Science*, 249:1037–1041, 1990.

[OK85] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Trans. PAMI*, 7(2):139–154, March 1985.

[OK93] M. Okutomi and T. Kanade. A multiple baseline stereo. *IEEE Trans. PAMI*, 15(4):353–363, April 1993.

[OP89] T. Olson and R. Potter. Real-time vergence control. *Proc. IEEE CVPR*, pages 404–409, 1989.

[Qia94] N. Qian. Computing stereo disparity and motion with known binocular cell properties. *Neural Computation*, 6:390–404, 1994.

[RC96] B. Srinivasa Reddy and B.N. Chatterji. An fft-based technique for translation, rotation and scale-invariant image registration. *IEEE Transactions on Image Processing*, 5(8):1266–1271, August 1996.

[RC98] S. Roy and I. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. *ICCV*, pages 492–499, 1998.

[RG83] B. Rogers and M. Graham. Anisotropies in the perception of three-dimensional surfaces. *Science*, 221:1409–1411, 1983.

[RG94] C. Ryan and B. Gilliam. Cue conflict and sterescopic surface slant about horizontal and vertical axes. *Perception*, 23:645–658, 1994.

[San88] T.D. Sanger. Stereo disparity computation using gabor filters. *Biological Cybernetics*, 59:405–418, 1988.

[SB95] B. J. Super and A. C. Bovik. Shape from texture using local spectral moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(4):333–343, 1995.

[SGK00] H.S. Sawhney, Y. Guo, and R. Kumar. Independent motion detection in 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1191–1199, 2000.

[Sin93] D. Sinclair. Motion segmentation and local structure. In *Proc. Fourth International Conference on Computer Vision*, pages 366–373, 1993.

[SS98] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *IJCV*, 28(2):155–174, 1998.

[SS02] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1):7 – 42, April 2002.

[SSV01] César Silva and José Santos-Victor. Motion from occlusions. *Robotics and Autonomous Systems*, 35(3-4):153–162, June 2001.

[Sze90] R. Szeliski. Bayesian modeling of uncertainty in low-level vision. *IJCV*, 5(3):271–302, Dec 1990.

[TM94] P. H. S. Torr and D. W. Murray. Stochastic motion clustering. In *Proc. Third European Conference on Computer Vision*, pages 328–337. Springer-Verlag, 1994.

[Tor98] P. H. S. Torr. Geometric motion segmentation and model selection. In J. Lasenby, A. Zisserman, R. Cipolla, and H. Longuet-Higgins, editors, *Philosophical Transactions of the Royal Society A*, pages 1321–1340. Roy Soc, 1998.

[TP90] W. B. Thompson and T.-C. Pong. Detecting moving objects. *International Journal of Computer Vision*, 4:39–57, 1990.

[TPHA00] B. Triggs, P.McLauchlan, R. Hartley, and A.Fitzgibbon. *Vision Algorithms: Theory and Practice*, chapter Bundle adjustment - a modern synthesis. Springer Verlag, 2000.

[TSK01] H. Tao, H.S. Sawhney, and R. Kumar. A global matching framework for stereo computation. *ICCV*, 1:532–539, July 2001.

[WA85] A.B. Watson and A.J. Ahumada. Model of human visual-motion sensing. *J. Opt. Soc. Am. A*, 2:322–342, 1985.

[WB95] C. S. Wiles and M. Brady. Closing the loop on multiple motions. In *Proc. Fifth International Conference on Computer Vision*, pages 308–313, 1995.

[Wei97] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 520–526, 1997.

[Wen94] J. Weng. Image matching using windowed fourier phase. *IJCV*, 11:211–236, 1994.

[Wit81] A. P. Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17:17–45, 1981.

[WM97] J. Weber and J. Malik. Rigid body segmentation and shape description from dense optical flow under weak perspective. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2), 1997.

[ZC93] Q. F. Zheng and R. Chellappa. Motion detection in image sequences acquired from a moving platform. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 201–204, 1993.

[ZFA88] Z. Zhang, O. D. Faugeras, and N. Ayache. Analysis of a sequence of stereo scenes containing multiple moving objects using rigidity constraints. In *Proc. Second International Conference on Computer Vision*, pages 177–186, 1988.