A Study of Fast, Robust Stereo-Matching Algorithms

by

Wenxian Hong

S.B. Mechanical Engineering
Massachusetts Institute of Technology (2009)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2010

Author ....[illegible]..........................
Department of Mechanical Engineering
May 21, 2010

Certified by ....[illegible]..................................
Douglas P. Hart
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by ....[illegible]..................
David E. Hardt
Chairman, Department Committee on Graduate Students

A Study of Fast, Robust Stereo-Matching Algorithms

by

Wenxian Hong

Submitted to the Department of Mechanical Engineering
on May 21, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Stereo matching is an actively researched topic in computer vision. The goal is to recover quantitative depth information from a set of input images, based on the visual disparity between corresponding points. This thesis investigates several fast and robust techniques for the task. First, multiple stereo pairs with different baselines may be meaningfully combined to improve the accuracy of depth estimates. In multi-baseline stereo, individual pairwise similarity measures are aggregated into a single evaluation function. We propose the novel product-of-error-correlation function as an effective example of this aggregate function. By imposing a common variable, inverse distance, across all stereo pairs, the correct values are reinforced while false matches are eliminated. Next, in a two-view stereo context, the depth estimates may also be made more robust by accounting for foreshortening effects. We propose an algorithm that allows a matching window to locally deform according to the surface orientation of the imaged point. The algorithm then performs correlation in multiple dimensions to simultaneously determine the most probable depth and tilt. The 2D surface orientation search may be made more efficient and robust by separating into two 1D searches along the epipolar lines of two stereo pairs. Moreover, by organizing the multi-dimensional correlation to avoid redundant pixel comparisons or using numerical minimization methods, greater efficiency may be achieved. Finally, we propose an iterative, randomized algorithm which can significantly speed up the matching process. The key insights behind the algorithm are that random guesses for correspondences can often produce some good disparity matches, and that these matches may then be propagated to nearby pixels assuming that disparity maps are piecewise smooth. Such a randomized algorithm converges within a small number of iterations and accurately recovers disparity values with relative computational efficiency. All three techniques developed are described analytically and evaluated empirically using synthetic or real image datasets to demonstrate their superior performance.

Thesis Supervisor: Douglas P. Hart
Title: Professor of Mechanical Engineering

Acknowledgments

I would like to thank my advisor, Prof. Douglas Hart, for his support throughout my research. His brilliant ideas have led me to dream big, and without his kind understanding, I would not have been able to finish this work. Coming from a mechanical engineering background, I am also indebted to Profs. Berthold Horn, Ramesh Raskar and Bill Freeman for showing me the ropes in computer vision and computational photography. Thanks also to Federico Frigerio, Tom Milnes and Danny Hernandez-Stewart for many helpful discussions. Lastly, I am grateful to Leslie Regan and the ME Graduate Office for always extending a helping hand.

On a personal note, I would like to thank my friends, especially those in the Sport Taekwondo team, the Singapore Students Society and my fellow classmates, for making these past four years a true pleasure amidst the occasional pain. And thanks to my parents for their constant encouragement and love.

Contents

List of Figures

Chapter 1

Introduction

Since the earliest studies in visual perception, it is well known that humans use the difference between the images in our left and right eyes to judge depth. Intuitively, objects that are closer to the viewer exhibit a larger displacement between images than objects that are further away. This visual disparity provides powerful depth cues which shape our understanding of the world around us and guide the way in which we respond to it.

Yet, while biological systems can easily interpret visual stimuli to perceive three-dimensional structure, it has been challenging to mimic this behavior in computer vision systems. Stereo matching is the process of computing a three-dimensional reconstruction of the scene, given a set of images taken from different viewpoints. This task involves automatically finding matching pixels between images and then converting the measured disparity into 3D depths based on the scene and camera geometry. The desired output is typically a dense depth map which assigns accurate depths to each pixel in the input images.

Early work on stereo matching was motivated by applications in photogrammetry, where the goal is to determine the shape of an object surface from a series of calibrated images. For instance, topographic maps of the earth's surface may be generated from overlapping aerial or satellite images. Before automatic stereo matching algorithms were developed, operators would use a manual device known as a stereo plotter (Fig. 1-1). Adjacent pairs of photographs are shown to each eye, allowing the operator

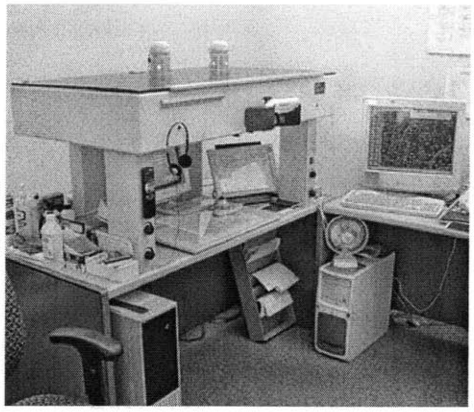


Figure 1-1: Alpha 2000 analytical stereo plotter *(Source: Wikipedia)*

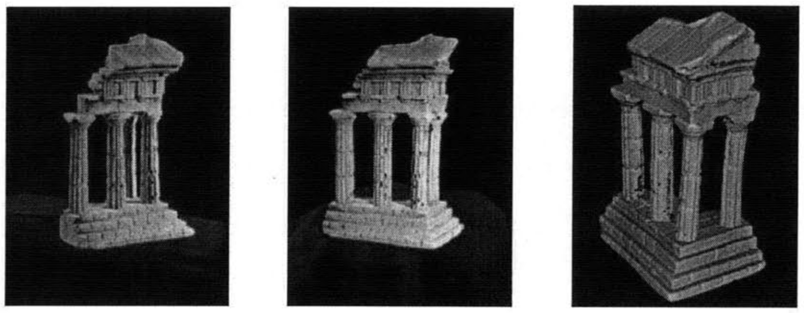


Figure 1-2: Modern stereo techniques can reconstruct a dense 3D model (right) from a set of images (left and middle). [28]

to measure the apparent change in positions of surface features and trace constant elevation contours using an artificial floating dot. Other previous research on stereo matching has focused on robotic navigation and object recognition.

More recently, however, the trend has been towards creating realistic object models for computer graphics and metrology applications (Fig. 1-2), as well as image-based rendering. Reliable algorithms have been developed to reconstruct the 3D geometry of a complex scene from thousands of partially overlapping images, or even generate compelling 3D models of famous landmarks from online community photograph collections [3]. Today, research on stereo matching has regained momentum as a result of publicly available performance evaluations such as the Middlebury library [1, 2], which allow researchers to compare new algorithms with the current state of the art.

This thesis explores different methods of quickly and robustly recovering quantitative depth information from a set of stereo images, as outlined below:

In Chap. 2, we first study the specific geometry of stereo systems and present a framework for understanding existing algorithms. Dense correspondence algorithms generally work by computing and aggregating matching costs, from which disparity estimates may then be obtained and refined. Depending on the exact mechanism, these algorithms may be classified as local or global methods.

Chap. 3 presents a technique known as multi-baseline stereo, which combines multiple stereo pairs together in order to obtain precise, unambiguous depth estimates. In multi-baseline stereo, individual pairwise similarity measures are aggregated into a single function, allowing reliable matches to reinforce each other while eliminating false matches. The method is described analytically, and experiments on synthetic and real image datasets demonstrate its effectiveness and robustness.

Chap. 4 presents another method that improves the accuracy of depth estimates, while at the same time recovering additional information about the surface properties. Conventional stereo algorithms use fixed windows based on the implicit assumption that the intensity patterns around corresponding points in different images remain constant. In actual fact, however, foreshortening effects between different viewpoints lead to local image deformations which give rise to inaccurate measurements. To overcome this issue, we propose an algorithm that allows the matching windows to deform according to the local surface orientation. Unlike conventional methods which only consider depth, our proposed method performs correlation over multiple dimensions of depth and local tilt. Efficient methods of performing multi-dimensional correlation are also proposed and evaluated.

In Chap. 5, we shift our focus to the task of increasing the speed of stereo matching. We introduce an iterative, randomized algorithm that efficiently computes dense disparity maps. The algorithm begins with random guesses for correspondences, which are often likely to be wrong. However, over the entire image, a few lucky disparity guesses will almost give the correct match. The algorithm then propagates these good matches to nearby pixels, based on the assumption that disparity maps are piecewise smooth. An iterative scheme is employed to refine the disparity estimates. Our theoretical analysis and experimental results show that such an algorithm typically converges within a small number of iterations, and potentially brings about significant savings in computation time and memory.

Finally, in Chap. 6, we summarize our work and indicate directions for future work.

Chapter 2

Basics of Stereo Matching

In this chapter, we describe the fundamental principles behind stereo matching. First, we study the geometry of stereo matching in order to understand how points in a scene are imaged at different positions and orientations based on their distances from the viewer. We then review techniques for finding a set of corresponding points between two images. This task involves selecting a suitable similarity measure and then employing local or global methods to identify correct matches.

2.1 Epipolar Geometry

The main task of an automatic stereo matching algorithm is to match a given point in one image with its corresponding point in the other image. At first, this task of establishing correspondences seems to require a search through the whole image. However, the *epipolar constraint* specific to stereo systems reduces this search to a single line. Consider a 3D point $\mathbf{p}$ being viewed from two cameras, as shown in Fig. 2-1. The point $\mathbf{p}$ projects onto the location $\mathbf{x_0}$ in the left image with camera center $\mathbf{c_0}$. Given that the point $\mathbf{p}$ lies somewhere along this viewing ray to infinity, the pixel $\mathbf{x_0}$ in the left image projects onto a line segment in the right image, called the epipolar line. The epipolar line is bounded on one end by the projection of the left camera center $\mathbf{c_0}$ in the right image, called the epipole $\mathbf{e_1}$, and on the other end by the vanishing point of the viewing ray from the left camera to $\mathbf{p}$. A similar line

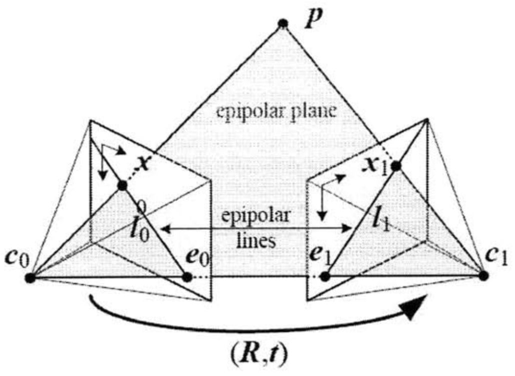


Figure 2-1: Epipolar geometry of binocular stereo systems [28]. A pixel in the left image is constrained to lie along the corresponding epipolar line in the right image.

is obtained by projecting the epipolar line in the right image onto the left image. These form a pair of corresponding epipolar lines, which are obtained by cutting the image planes with the epipolar plane containing the point $\mathbf{p}$ and the camera centers $\mathbf{c_0}$ and $\mathbf{c_1}$. An object imaged on the epipolar line in the left image can only be imaged on the corresponding epipolar line in the right image, and thus the search for correspondences is reduced to a line.

In the case of calibrated cameras whose relative position is represented by a rotation $\mathbf{R}$ and translation $\mathbf{t}$, we may express the epipolar constraint more formally. A pixel $\mathbf{x_0}$ in the left image is then mapped onto the right image at location $\mathbf{x_1}$ by the transformation

$$\mathbf{x_1} = \mathbf{R}\mathbf{x_0} + \mathbf{t} \tag{2.1}$$

Given that the rays $\mathbf{x_0}$ and $\mathbf{x_1}$ intersect at the point $\mathbf{p}$, the vectors describing these rays, namely $\mathbf{x_1}$ and $\mathbf{R}\mathbf{x_0}$, and the vector connecting the two camera centers $\mathbf{c_1} - \mathbf{c_0} = \mathbf{t}$ must be coplanar. Hence, the triple product is equal to zero, viz.

$$\mathbf{x_1} \cdot (\mathbf{t} \times \mathbf{R}\mathbf{x_0}) = 0. \tag{2.2}$$

We thus derive the epipolar constraint

$$\mathbf{x_1^T E x_0} = 0, \tag{2.3}$$

where

$$\mathbf{E} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \mathbf{R} \tag{2.4}$$

is known as the essential matrix. The essential matrix $\mathbf{E}$ maps a point $\mathbf{x_0}$ in the left image to a line $\mathbf{l_1} = \mathbf{Ex_0}$ in the right image. Since $\mathbf{E}$ depends only on the extrinsic camera parameters (i.e. $\mathbf{R}$ and $\mathbf{t}$), the epipolar geometry may be computed for a pair of calibrated cameras. In the case of uncalibrated cameras, a similar quantity to the essential matrix, known as the fundamental matrix, may also be computed from seven or more point matches. The fundamental matrix captures both the intrinsic and extrinsic parameters of the system of cameras, but serves the same function of mapping points in the left image to lines in the right image.

Even though the epipolar geometry constrains the search for potential correspondences along epipolar lines, the arbitrary orientations of these lines makes it inconvenient for algorithms to compare pixels. To overcome this issue, the input images are commonly warped or *rectified* so that the epipolar lines reduce to corresponding horizontal scanlines. Rectification may be achieved by first rotating both cameras such that their optical axes are perpendicular to the line joining the two camera centers, or the baseline. Next, the vertical y-axis of the camera is rotated to become perpendicular to the baseline. Finally, the images are rescaled, if necessary, to compensate for differences in focal lengths. In practice it would be difficult to arrange the two camera optical axes to be exactly parallel to each other and perpendicular to the baseline. Moreover, the cameras are often verged or tilted inwards so as to adequately cover the region of interest. Hence, rectification is often carried out in software rather than hardware [4].

After rectification, the camera geometry is transformed into a canonical form.

By considering similar triangles in the ray diagram, we may derive a simple inverse relationship between depth Z and disparity d between corresponding pixels in the two images, viz.

$$d = \frac{Bf}{Z}. \tag{2.5}$$

In this equation, f is the focal length (measured in pixels) and B is the baseline. The disparity d describes the difference in location between corresponding pixels in the left and right images, i.e. (x_0, y_0) and $(x_1, y_1) = (x_0 + d, y_0)$. By estimating the disparity for every pixel in a reference image, we obtain a disparity map $d(x, y)$, from which the object depth may then be computed.

2.2 Finding Correspondences

Over the years, numerous algorithms have been proposed to find correspondences in a set of images. These may be broadly classified into two categories.

The first approach extracts features of interest from the images, such as edges or contours, and matches them in two or more views. These methods are fast because they only use a small subset of pixels, but they also yield only sparse correspondences. The resulting sparse depth map or disparity map may then be interpolated using surface fitting algorithms. Early work in finding sparse correspondences was motivated partly by the limited computational resources at the time, but also by the observation that certain features in an image produce more reliable matches than others. Such features include edge segments and profile curves, which occur along the occluding boundaries. Unfortunately, these early algorithms require several closely-spaced camera viewpoints in order to stably recover features.

Recent research has focused on extracting features with greater robustness and repeatability, and thereafter using these features to grow into missing regions. To date, the most successful technique for detecting and describing features is the Scale Invariant Feature Transform (SIFT) keypoint detector [5]. The algorithm locates interest points at the extrema of a Difference-of-Gaussian function in scale-space.

Each feature point also has an associated orientation, as determined by the peak of a histogram of local orientations. Hence, SIFT features are robust to changes in scale, rotation and illumination that may occur between different images. Using these SIFT features, one may apply robust structure-from-motion algorithms to determine image correspondences, compute the fundamental matrix with associated intrinsic and extinsic camera parameters, as well as generate a sparse 3D reconstruction of the scene. Such methods have been employed to reconstruct 3D geometry in situations where the input images are uncalibrated [6] or from Internet photograph collections [3].

The second approach seeks to find a dense set of correspondences between two or more images, since recovering a smooth and detailed depth map is more useful for 3D modeling and rendering applications. Based on the taxonomy of dense correspondence algorithms proposed by Scharstein and Szeliski [1], such techniques typically involve the calculation and aggregation of matching costs, from which disparity values may then be computed and refined. Dense stereo techniques may also be further subdivided into two main strategies: local and global. Local approaches determine the correspondence of a point by selecting the candidate point along the epipolar lines that minimizes a cost function. To reduce matching ambiguity, the matching costs are aggregated over a support window rather than computed point-wise. On the other hand, global methods generally do not aggregate the matching costs, but instead rely on explicit smoothness assumptions to ensure accuracy. The objective of global stereo algorithms is to find the disparity assignment which minimizes an energy function that includes both data (cost) and smoothness terms.

In this thesis, we deal primarily with the problem of finding dense correspondences. Subsequent sections will also explain both local and global methods in greater detail.

2.3 Similarity Measures

Regardless of the type of stereo algorithm used, a means of determining the similarity between pixels in different images is key to disambiguating potential matches and finding correspondences. In order to find where a given pixel $\mathbf{x} = (x, y)$ in the left image I_0 appears in the right image I_1, we use a fixed support window sampled at locations $\{\mathbf{x}_i = (x_i, y_i)\}$ around the pixel and measure the degree of similarity with I_1 at different disparities. A basic measure is the sum-of-squared differences (SSD) function, given by

$$C_{SSD}(\mathbf{x}, \mathbf{d}) = \sum_i \left[I_1(\mathbf{x}_i + \mathbf{d}) - I_0(\mathbf{x}_i)\right]^2. \tag{2.6}$$

This similarity measure implicitly assumes that the pixel intensities around corresponding points in the two images remain constant, such that the SSD cost is minimized at the correct offset. In the presence of image noise, the similarity measure may be made more robust to outliers within the window by imposing a penalty that grows less quickly than the quadratic SSD term. A common example of such a robust measure is the sum-of-absolute-differences (SAD) function, given by

$$C_{SAD}(\mathbf{x}, \mathbf{d}) = \sum_i \left|I_1(\mathbf{x}_i + \mathbf{d}) - I_0(\mathbf{x}_i)\right|. \tag{2.7}$$

In this case, the cost function grows linearly with the residual error between the windows in the two images, thus reducing the influence of mismatches when the matching cost is aggregated. Other robust similarity measures such as truncated quadratics or contaminated Gaussians may also be used [28].

Furthermore, it is not uncommon for the two images being compared to be taken at different exposures. To compensate for this, cost functions which are invariant to inter-image intensity differences may be used. A simple model of linear intensity variation between two images may be described by

$$I_1(\mathbf{x} + \mathbf{d}) = (1 + \alpha) I_0(\mathbf{x}) + \beta, \tag{2.8}$$

where α is the gain and β is the bias. The SSD cost function may then be modified to take the intensity variation into account, viz.

$$\begin{aligned} C_{SSD}(\mathbf{x}, \mathbf{d}) &= \sum_i \left[I_1(\mathbf{x}_i + \mathbf{d}) - (1+\alpha) I_0(\mathbf{x}) - \beta\right]^2 && (2.9)\\ &= \sum_i \left[\alpha I_0(\mathbf{x}) + \beta - (I_1(\mathbf{x}_i + \mathbf{d}) - I_0(\mathbf{x}))\right]^2 . && (2.10)\end{aligned}$$

Hence, it is possible to correct for intensity variations across images by performing a linear regression to recover the bias-gain parameters, albeit at a higher computational cost. More sophisticated models have also been proposed, which can account for spatially-varying intensity differences due to vignetting. Alternatively, cost functions that match gradients rather than intensities or subtract some window average from the pixel intensities have also been shown to be robust to changes in exposure and illumination.

Besides measuring the residual error in window intensities, another commonly used similarity measure is the cross-correlation of the two displaced windows, given by

$$C_{CC}(\mathbf{x}, \mathbf{d}) = \sum_i I_0(\mathbf{x}_i) I_1(\mathbf{x}_i + \mathbf{d}). \tag{2.11}$$

The maximum correlation value among all the candidate disparities corresponds to the most probable match. It is also worth noting that Eq. 2.12 is the spatial convolution of the signal in I_0 with the complex conjugate of the signal in I_1. Since convolution in the spatial domain is equivalent to multiplication in the Fourier domain, the cross-correlation function may thus be computed efficiently by multiplying the Fourier transforms of the two image windows and taking the inverse transform of the result. Mathematically, this is expressed as

$$C_{CC}(\mathbf{x}, \mathbf{d}) = \mathcal{F}^{-1}\{I_0(\mathbf{x}) * I_1(-\mathbf{x})\} = \mathcal{F}^{-1}\{\mathcal{I}_0(\omega)\mathcal{I}_1^*(\omega)\}, \tag{2.12}$$

where the asterisk superscript denotes the complex conjugate, ω is the angular frequency of the Fourier transforms (in calligraphic symbols) and $\mathcal{F}^{-1}$ denotes the inverse Fourier transform.

However, matching using cross-correlation can fail if the images have a large dynamic range. For instance, the correlation between a given pixel in the left image and an exactly matching region in the right image may be less than the correlation between the same pixel and a bright patch. Moreover, Eq. 2.12 is sensitive to changes in illumination or exposures across images. To circumvent these limitations, the normalized cross-correlation function is typically used, viz.

$$C_{NCC}(\mathbf{x},\mathbf{d}) = \frac{\sum_i \left[I_0(\mathbf{x}_i) - \overline{I_0(\mathbf{x}_i)}\right]\left[I_1(\mathbf{x}_i+\mathbf{d}) - \overline{I_1(\mathbf{x}_i+\mathbf{d})}\right]}{\sqrt{\sum_i \left[I_0(\mathbf{x}_i) - \overline{I_0(\mathbf{x}_i)}\right]^2 \left[I_1(\mathbf{x}_i+\mathbf{d}) - \overline{I_1(\mathbf{x}_i+\mathbf{d})}\right]^2}}, \tag{2.13}$$

where $\overline{I_0}$ and $\overline{I_1}$ are the mean intensities of the corresponding windows. The resulting correlation table is normalized such that a value of 1 indicates perfect correlation.

Finally, we describe a related form of correlation, known as phase correlation. Phase correlation is based on the observation that the magnitudes of the Fourier transform of two displaced images remain constant and only the phase in the transform domain changes. The phase correlation function thus divides the frequency spectrums of the two image windows by their respective Fourier magnitudes before taking the inverse transform, as given by

$$C_{PC}(\mathbf{x},\mathbf{d}) = \mathcal{F}^{-1}\{\frac{\mathcal{I}_0(\omega)\mathcal{I}_1^*(\omega)}{\|\mathcal{I}_0(\omega)\|\,\|\mathcal{I}_1^*(\omega)\|}\}. \tag{2.14}$$

Consider the ideal case where the two image windows are simply displaced, i.e. $I_1(\mathbf{x}_i + \mathbf{d}) = I_0(\mathbf{x}_i)$. The Fourier transforms of each image window may be found using the shift theorem, viz.

$$\mathcal{F}\{I_1(\mathbf{x}_i + \mathbf{d})\} = \mathcal{I}_1(\omega)e^{-2\pi j\mathbf{d}\cdot\omega} = \mathcal{I}_0(\omega) \tag{2.15}$$

Hence, the phase correlation function is given by

$$C_{PC}(\mathbf{x},\mathbf{d}) = \mathcal{F}^{-1}\{e^{-2\pi j\mathbf{d}\cdot\omega}\}, \tag{2.16}$$

which is a Dirac delta function at the correct disparity value $\mathbf{d}$. In theory, phase correlation thus gives a sharper peak, which facilitates the disambiguation of potential matches. A further advantage of working with Fourier magnitudes of the displaced windows is the potential to compensate for rotations between the two images. Since the magnitudes of Fourier transforms are relatively insensitive to translations, the magnitude images may be rotated and aligned in Fourier space so as to guide the alignment of the images in the spatial domain. The aligned images may then be compared using a conventional translational similarity measure to recover the matching disparity.

2.4 Local Methods

Local area-based stereo methods match pixel values between images by aggregating the matching costs over a support window. In order to achieve a smooth and detailed disparity map, the selection of an appropriate window is critical. The optimal window should be large enough to contain enough intensity variation for reliable matching, particularly in areas of low texture. On the other hand, it should also be small enough to minimize disparity variation within the window due to tilted surfaces or depth discontinuities. Using too large a window can lead to undesirable smoothing and the 'fattening' or 'shrinkage' of edges, as the signal from a highly textured surface may affect less-textured surfaces nearby across occluding boundaries. While in the simplest case the support window can be a fixed square window, several techniques have been proposed to balance the trade-offs involved in window selection. Generally, these methods use a variable support which dynamically adapts itself depending on the surroundings of the pixels being matched, such as windows with adaptive sizes and shapes [7], shiftable windows [8], and segmentation-based windows which only consider the contributions of pixels at the same disparity [9].

After computing and aggregating the matching costs, the disparity of a given pixel may be easily computed by performing a Winner-Takes-All (WTA) optimization, i.e. choose the disparity value associated with the minimum matching cost. Repeating the process for every pixel in the image produces a disparity map accurate to the nearest pixel.

Nonetheless, in some applications such as image-based rendering or 3D modelling, a pixel-wise disparity map is often not accurate enough and produces implausible object models or view synthesis results. Hence, many local stereo algorithms also employ an additional sub-pixel refinement step. Several techniques may be used to obtain better sub-pixel disparity estimates at little additional computational cost. One may fit a curve to the matching costs evaluated at discrete integer values around the best existing disparity and then interpolate to find a more precise minimum. Another possibility is to perform iterative gradient descent on the SSD cost function, as proposed by Lucas and Kanade [10]. This approach relies on the assumption of brightness constancy in the local spatial and temporal neighborhoods of the displaced frames, i.e.

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t), \tag{2.17}$$

where $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$ are the x- and y- velocities of the image window, also known as the optical flow components. Setting the total derivative of the image brightness (Eq. 2.17) to zero then yields the optical flow constraint equation

$$I_x u + I_y v + I_t = 0, \tag{2.18}$$

where the spatial and temporal partial derivatives I_x, I_y and I_t may be estimated from the image. Since Eq. 2.20 applies for all pixels within the image window, we can solve an overdetermined system of equations for the optical flow components and determine the subpixel disparity.

Apart from increasing the resolution of the disparity map, other possible post-processing steps include cross-checking between the left-to-right and right-to-left disparity maps to detect occluded areas, applying a median filter to eliminate mismatched outliers, and surface-fitting to fill in holes due to occlusion.

2.5 Global Methods

Unlike local methods which compute the disparity of each pixel fairly independently, global methods typically perform an iterative optimization over the whole image [1]. The task is to label each pixel with the disparity solution d that minimizes a global energy function

$$E(d) = E_d(d) + \lambda E_s(d), \tag{2.19}$$

where $E_d(d)$ and $E_s(d)$ represent the data and smoothness terms respectively, and the parameter λ controls the influence of the smoothness term [1]. The data term $E_d(d)$ measures the degree of similarity between each pixel in the reference image and its corresponding pixel in the other image at the current disparity assignment d, and may thus be defined as

$$E_d(d) = \sum_{x,y} C(x, y, d(x, y)), \tag{2.20}$$

where $C(x, y, d(x, y))$ is the point-wise or aggregated matching cost. The smoothness term $E_s(d)$ performs a similar function to cost aggregation in minimizing spurious matches and making the disparity map as smooth as possible. A simple way to mathematically codify the smoothness assumption is to minimize the difference in disparities between neighboring pixels, viz.

$$E_s(d) = \sum_{x,y} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1)), \tag{2.21}$$

where ρ is some monotonically increasing function of the disparity differences. However, this simplistic smoothness term tends to make the disparity map smooth throughout and thus breaks down at occluding boundaries. Intuitively, depth discontinuities tend to occur along intensity edges. Hence, a modified smoothness term can be made to depend on the intensity differences on top of disparity differences, as given by

$$E_s(d) = \sum_{x,y} \rho_d(d(x,y) - d(x+1,y)) \cdot \rho_I(\|I(x,y) - I(x,y+1)\|). \tag{2.22}$$

The solution of the global energy minimization problem is a maximum a posteriori estimate of a Markov Random Field (MRF). Early methods, such as iterated conditional modes (ICM) or simulated annealing, were inaccurate or very inefficient. Recently, however, high-performance algorithms such as graph cuts, loopy belief propagation (LBP) and tree-reweighted message passing have been developed, forming the basis for many of the top-performing stereo methods today. An MRF optimization library is also available online [11].

Finally, another noteworthy category of global methods makes use of dynamic programming. These techniques consider two corresponding scanlines and compute the minimum-cost path through the 2D matrix of all pairwise matching costs functions along those lines. Each entry of the matrix is computed by combining its cost value with one of its previous entries. As the optimization solution is one-dimensional, dynamic programming can produce extremely fast results and has been employed in real-time stereo applications. Unfortunately, consistency between scanlines cannot be well-enforced as scanlines are considered individually, often leading to streaking effects in the resulting disparity map.

2.6 Summary

While many variants of stereo matching algorithms exist, these may be broadly classified into local and global methods. Although global methods currently produce some of the best stereo matching results (according to the Middlebury evaluation website [1]), they are still usually much slower than local methods, since global optimization is an NP-hard problem while local matching runs in polynomial time. In the rest of this thesis, we will primarily consider local window-based methods, while proposing novel refinements to increase the robustness and efficiency of disparity computation.

Chapter 3

Multiple-Baseline Stereo

3.1 Introduction

Having discussed the fundamentals of binocular stereo matching, we now extend the principles to stereo systems with multiple cameras. In multi-view stereo, the goal is to recover quantitative depth information from a set of multiple calibrated input images. Since more information is available from different perspectives of the scene, dense high-quality depth maps as well as realistic 3D object models are often possible with multi-view stereo.

Many multi-view stereo algorithms have their roots in traditional area-based binocular stereo, which relies on the observation that objects in a scene are imaged at different locations depending on their distance from each camera. Given a point in a reference image, the stereo matching process searches for the corresponding point along the epipolar line of another image. The best match is decided using a criterion that measures similarity between shifted image windows, such as by minimizing the sum of squared differences (SSD). Finally, the relative locations of the matched point pair are triangulated to determine depth.

For the canonical stereoscopic system with two cameras positioned such that their optical axes are parallel and x-axes are aligned, the epipolar lines conveniently reduce to scanlines. In this case, the disparity d between corresponding points in the left and right images is related to the distance z of the object by

$$d = \frac{BF}{z}, \tag{3.1}$$

where B and F are the baseline and focal length respectively. Eq. 3.1 shows that disparity is directly proportional to baseline, and hence the precision of depth calculation increases with a longer baseline. However, a longer baseline requires the search to be done over a larger disparity range and increases the chance of occlusion, thereby making false matches more likely. Presumably, multiple images with different baselines may be used to disambiguate potential matches and reduce errors in depth estimates.

Okutomi and Kanade propose one such method to integrate multiple baseline stereo pairs into an accurate and precise depth estimate [12]. Their key insight is that since disparity is inversely proportional to distance (according to Eq. 3.1), the calculation of the similarity criterion (chosen to be SSD) may be carried out in terms of inverse distance rather than disparity, as is typically the case. Computing SSD values using inverse distance is advantageous because unlike disparity which varies across baselines, inverse distance is a common measure throughout every stereo pair. Hence, there is a single distance (i.e. the true distance) at which the SSD values from all stereo pairs are minimized. Summing these SSD values into a single function, called SSSD-in-inverse-distance, then gives an unambiguous and precise minimum at the correct matching position. In effect, the SSSD-in-inverse-distance function meaningfully combines different stereo pairs to produce a depth estimate that is more refined than that from any single pair. Such a approach is robust to noise and can handle challenging scenes with periodic patterns where traditional stereo methods would otherwise fail.

In this chapter, we shall study the multi-baseline stereo approach in detail. Section 2 provides a theoretical analysis of the method, while Section 3 describes our software implementation. In particular, we propose the novel product-of-error-correlation (PEC) function as an alternative function to aggregate individual stereo pairs. In Section 4, we present experimental results with synthetic and real scenes to demonstrate the utility of the algorithm. Finally, we conclude by evaluating the advantages and limitations of this method.

3.2 Theoretical Analysis

In this section, we present the mathematical justifications behind the multi-baseline stereo method in [12] and demonstrate how it reduces ambiguity and increases precision of measurements.

Consider a row of $(n+1)$ cameras $C_0, C_1, ..., C_n$ whose optical axes are perpendicular to the line, thus resulting in a set of n stereo pairs with baselines $B_1, B_2, ..., B_n$ relative to a base view C_0. Following Eq. 3.2, the correct disparity $d_{r,i}$ between C_0 and an arbitrary view C_i for an object at distance z is

$$d_{r,i} = \frac{B_i F}{z}. \tag{3.2}$$

Without loss of generality, we consider only an x-z world plane, such that the images are 1D functions. Assuming noise may be neglected, the image intensity functions $f_0(x)$ and $f_i(x)$ near the correct matching position may then be written as

$$\begin{aligned} f_0(x) &= f(x) \\ f_i(x) &= f(x - d_{r,i}) \end{aligned} \tag{3.3}$$

For a pixel at position x on the image $f_0(x)$, the SSD cost function for the candidate disparity d_i over a window W is defined to be

$$\begin{aligned} E_d(x, d_i) &= \sum_{j \in W} (f_0(x+j) - f_i(x + d_i + j))^2 \\ &= \sum_{j \in W} (f(x+j) - f(x + d_i - d_{r,i} + j))^2 \end{aligned} \tag{3.4}$$

The matching window is identified when the SSD function $E_d(x, d_i)$ reaches a minimum. Clearly, the disparity d_i that minimizes $E_d(x, d_i)$ occurs at $d_i = d_{r,i}$, and is taken to be the disparity estimate at x.

Observe that in the presence of periodicity in the image, the SSD function (as defined in Eq. 3.4 with respect to disparity) exhibits multiple minima and thus leads to ambiguity in matching. For instance, if the image $f(x)$ appears the same at pixel positions x and $x + a$, $a \neq 0$

$$f(x + j) = f(x + a + j), \tag{3.5}$$

then from Eq. 3.4,

$$E_d(x, d_{r,i}) = E_d(x, d_{r,i} + a) = 0, \tag{3.6}$$

thereby resulting in two possible minima and a false match at $d_{r,i} + a$. Since this false match occurs at the same disparity location for all stereo pairs, the inclusion of multiple baselines does not disambiguate the potential matches.

Guided by our earlier intuition, we now rewrite disparity in Eq. 3.2 in terms of inverse distance $\zeta = \frac{1}{z}$, viz.

$$\begin{aligned} d_{r,i} &= B_i F \zeta_r & (3.7) \\ d_i &= B_i F \zeta & (3.8) \end{aligned}$$

Using Eqs. 3.7 & 3.8, the SSD function may also be expressed in terms of a candidate inverse distance ζ, so that Eq. 3.4 becomes

$$E_{\zeta_i}(x, \zeta) = \sum_{j \in W} \left(f(x + j) - f(x + B_i F(\zeta - \zeta_r) + j)\right)^2. \tag{3.9}$$

Lastly, we define the SSSD-in-inverse-distance function $E_{\zeta(12\ldots n)}(x, \zeta)$ by taking the sum of SSD functions with respect to inverse distance for the n stereo pairs:

$$\begin{aligned} E_{\zeta(12\ldots n)}(x,\zeta) &= \sum_{i=1}^{n} E_{\zeta_i}(x,\zeta) \\ &= \sum_{i=1}^{n}\sum_{j\in W}\left(f(x+j) - f(x+B_iF(\zeta-\zeta_r)+j)\right)^2 \qquad (3.10) \end{aligned}$$

Now, we reconsider the problem of ambiguity in the input images, as stated in Eq. 3.5. Although each SSD function with respect to inverse distance still exhibits ambiguous minima, i.e.

$$E_\zeta(x,\zeta_r) = E_\zeta(x,\zeta_r + \frac{a}{B_iF}) = 0, \qquad (3.11)$$

we can show that the SSSD-in-inverse-distance function reaches a minimum only at the correct ζ_r, i.e.

$$\begin{aligned} E_{\zeta(12\ldots n)}(x,\zeta) &= \sum_{i=1}^{n}\sum_{j\in W}\left(f(x+j) - f(x+B_iF(\zeta-\zeta_r)+j)\right)^2 \\ &> 0 = E_{\zeta(12\ldots n)}(x,\zeta_r)\ \forall\zeta \neq \zeta_r \qquad (3.12) \end{aligned}$$

To see this, suppose for a moment that

$$\sum_{i=1}^{n}\sum_{j\in W}\left(f(x+j) - f(x+B_iF(\zeta-\zeta_r)+j)\right)^2 = 0. \qquad (3.13)$$

For the case of Eq. 3.5, Eq. 3.13 attains equality only when

$$B_iF(\zeta-\zeta_r) = a,\ \forall i = 1,2,\ldots,n \qquad (3.14)$$

However, if $B_1 \neq B_2 \neq \ldots \neq B_n$ and $\zeta \neq \zeta_r$, Eq. 3.14 obviously cannot hold for all i. Consequently, Eq. 3.13 also does not hold and its left-hand side must be positive, thus validating Eq. 3.12. In other words, the SSSD-in-inverse-distance function produces a clear, unique minimum at the correct matching position and thus eliminates ambiguity arising from patterns in the input images.

This disambiguation of potential matches is best illustrated graphically. The authors of [12] took nine images of a highly periodic grid pattern at various camera positions separated by lateral displacement b. Fig. 3-1 shows the SSD values against normalized inverse distance for various baseline stereo pairs. Shorter baselines exhibit fewer minima but with less precision, while longer baselines produce sharper minima but with greater ambiguity. Evidently, any single SSD matching function will not yield a depth estimate that is both accurate and precise. However, summing these SSD functions in succession gives rise to a minimum at the correct position that becomes sharper as more stereo pairs are used, as shown in Fig. 3-2. Therefore, the effect of using the SSSD-in-inverse-distance function with enough baselines is to resolve any matching ambiguities and to increase precision of the depth estimate.

Finally, it is worth noting that while the above theoretical analysis applies to the SSD cost function, the multi-baseline method does not place restrictions on the choice of similarity criterion. The above theoretical analysis is thus valid for other cost functions which may be more robust than SSD.

3.3 Implementation

We implement the Okutomi-Kanade multi-baseline stereo algorithm with two main modifications, as described below.

3.3.1 Correlation-based Similarity Measures

First, we use Normalized Cross-Correlation (NCC) instead of Sum of Squared Differences as the measure of similarity, since NCC is functionally equivalent to SSD except that the NCC function should be maximized rather than minimized (see Sect. 2.3). Moreover, NCC has the added advantage of being invariant to changes in the mean intensity value and the dynamic range of the windows.

The NCC function between two images I_0 and I_i for a window (W_x, W_y) centered at a pixel (x, y) is defined to be

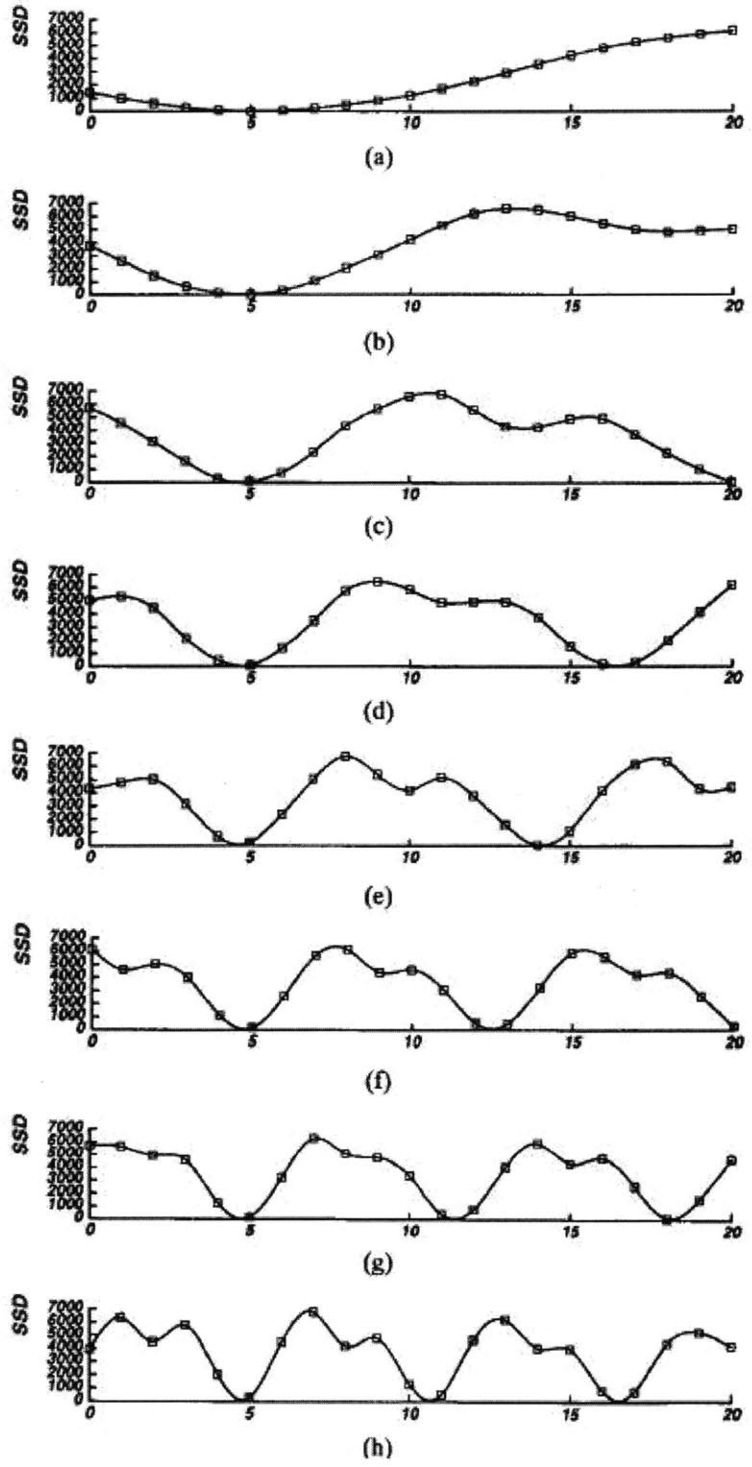


Figure 3-1: SSD as functions of normalized inverse distance for various baselines: (a) $B = b$, (b) $B = 2b$, (c) $B = 3b$, (d) $B = 4b$, (e) $B = 5b$, (f) $B = 6b$, (g) $B = 7b$, (h) $B = 8b$. The horizontal axis is normalized such that $8bF = 1$. [12]

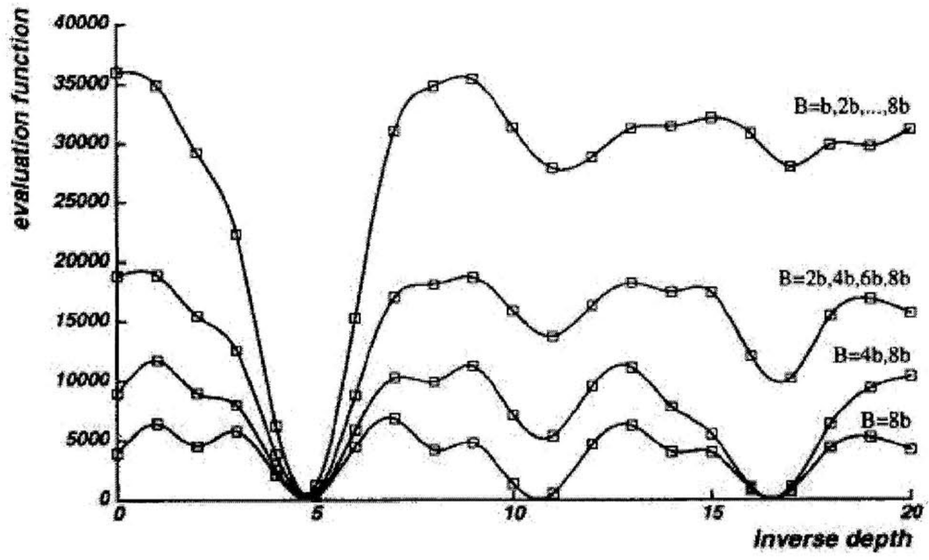


Figure 3-2: Effect of combining multiple baseline stereo pairs. Every time more stereo pairs are introduced by successively halving the baselines, the minima of the SSSD-in-inverse-distance function become fewer and sharper. When all the images are used, there is a clear minimum at the correct inverse depth. [12]

$$\Phi(x,y,d_x,d_y) = \frac{\sum_{x\in W_x}\sum_{y\in W_y}\left(I_0(x,y)-\overline{I_0(x,y)}\right)\cdot\left(I_i(x+d_x,y+d_y)-\overline{I_i(x_i+d)}\right)}{\sqrt{\sum_{x\in W_x}\sum_{y\in W_y}\left(I_0(x,y)-\overline{I_0(x,y)}\right)^2\cdot\sum_{x\in W_x}\sum_{y\in W_y}\left(I_i(x+d_x,y+d_y)-\overline{I_i(x+d_x,y+d_y)}\right)^2}}. \tag{3.15}$$

Each element of the resulting correlation table gives the degree of similarity between image windows for a candidate disparity (d_x, d_y), so that the peak represents the best disparity estimate of the corresponding point in the second image. To further improve processing speed, we perform an alternative form of correlation, known as error correlation (EC) [13]. The EC function may be expressed as

$$\Phi(x,y,d_x,d_y) = \frac{\sum_{x\in W_x}\sum_{y\in W_y}[I_0(x,y)+I_i(x+d_x,y+d_y)-|I_0(x,y)-I_i(x+d_x,y+d_y)|]}{\sum_{x\in W_x}\sum_{y\in W_y}[I_0(x,y)+I_i(x+d_x,y+d_y)]}. \tag{3.16}$$

As before, a correlation table is generated whose peak corresponds to the disparity estimate, albeit using only cheap addition operations rather than costly multiplications (in either the spatial or Fourier domain) for the standard NCC function. Like

Eq. 3.15, Eq. 3.16 also does not give unfair preference to high intensity pixels and is normalized so that a computed value of 1 indicates perfect correlation while a value of 0 indicates no correlation.

3.3.2 Product of EC-in-inverse-distance (PEC)

The second modification pertains to the method of aggregating similarity measures. Following the Okutomi-Kanade approach, we may re-express the NCC or EC functions (Eqs. 3.15 and 3.16) in terms of inverse-distance rather than disparity. However, instead of taking the sum of correlation tables, we multiply the correlation table from one stereo pair, element-by-element, with those generated from all other stereo pairs to obtain a single "product of EC-in-inverse distance" (PEC) table. Although multiplication is computationally more expensive than addition, combining the correlation tables via multiplication reduces ambiguity with fewer images and produces a sharper peak compared to summation. Disambiguation is made more efficient because any correlation value that does not appear in each of the individual correlation tables is automatically eliminated from the resulting PEC table instead of being retained, which is the case if the tables are summed or averaged. Conversely, correlation values that are identical in location are amplified exponentially rather than linearly due to the multiplicative operation. By simultaneously eliminating false correlation peaks and amplifying potential true peaks, the peak of the resulting PEC table then becomes more well-defined.

We illustrate this effect graphically using a set of five synthetic images of a ramp function illuminated by a random speckle pattern, taken from equally-spaced positions (the central view is shown in Fig. 3-5). Using a small window size of 3-by-3-pixels, the disparity for a point in the reference view (Image 3) is not discernable in any of the individual correlation tables. However, when combined together (see Sect. 3.3.3 for details), the peak of the resulting PEC table is easily resolved and may be measured to sub-pixel precision.

In effect, the PEC-in-inverse-distance table is a zero-dimensional correlation of multiple correlation tables. It merges individual correlations computed for different

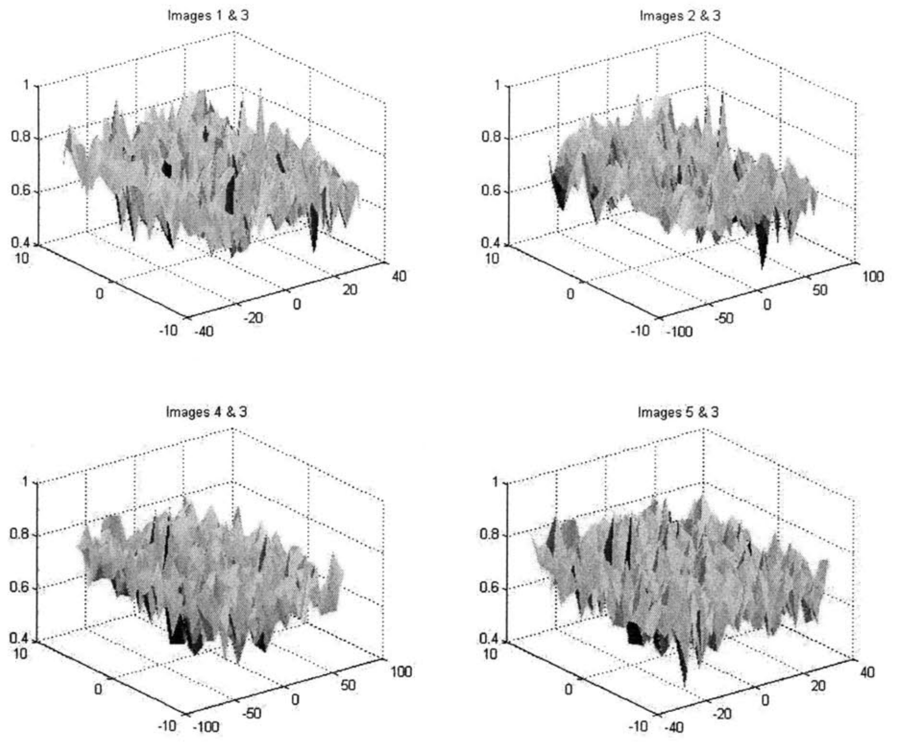


Figure 3-3: Individual correlation tables for each stereo pair in the synthetic image dataset. The true matching peak is not discernable in any of the tables alone.

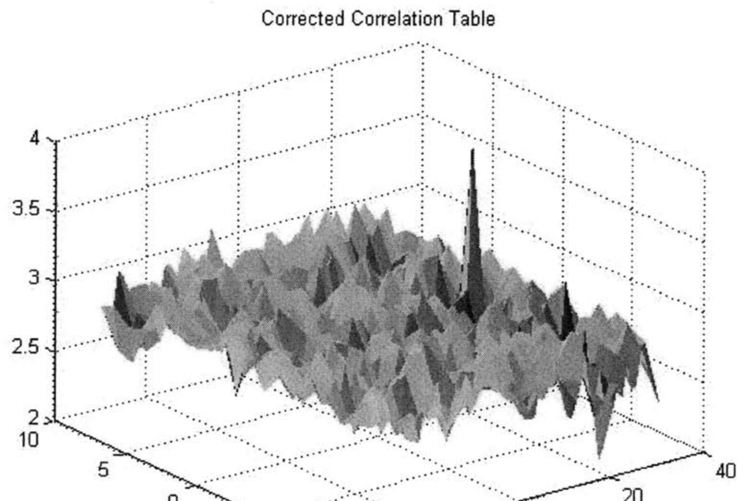


Figure 3-4: Corrected PEC table using element-by-element multiplication of the individual correlation tables in Fig.3-3. The true matching peak is easily resolved and measured.

stereo pairs into a common reference system, and allows us to find the best candidate depth based not only on the correlation values, but also on the coherence between the correlation functions.

3.3.3 Description of Algorithm

With the above modifications in mind, the multi-baseline stereo algorithm seeks to find the inverse distance ζ that maximizes the PEC-in-inverse distance function:

$$\begin{aligned} PEC(x,y,\zeta) &= \prod_{i=1}^{n} \Phi_i(x,y,\zeta) && (3.17) \\ &= \prod_{i=1}^{n} \frac{\sum_{x\in W_x}\sum_{y\in W_y}[I_0(x,y)+I_i(x+B_iF\zeta,y)-|I_0(x,y)-I_i(x+B_iF\zeta,y)|]}{\sum_{x\in W_x}\sum_{y\in W_y}[I_0(x,y)+I_i(x+B_iF\zeta,y)]}, && (3.18) \end{aligned}$$

where we have simplified the expression by noting that $d_y = 0\,\forall x, y$ for a set of n stereo pairs laterally displaced in only the x-direction. For ease of implementation, however, we normalize the disparity values for each stereo pair with respect to a reference baseline B_{ref}, and use this normalized disparity as the common variable for optimization instead of using inverse distance. In mathematical terms,

$$d_i = B_iF\zeta = \frac{B_i}{B_{ref}}B_{ref}F\zeta = R_id_{ref}, \tag{3.19}$$

where $R_i = \frac{B_i}{B_{ref}}$ is the baseline ratio for the i-th stereo pair. Eq. 3.19 implies that each element of a correlation table (representing some candidate disparity d_i for that particular pair) can be mapped to a location in the correlation table of a reference stereo pair, given the geometric constraints that the points must satisfy. Once this is done for all tables, the reference disparity d_{ref} is effectively equivalent to inverse distance, since the entries of each transformed table now indicate the same depth.

Substituting Eq. 3.19 into Eq. 3.20 yields the revised multi-baseline stereo optimization evaluation function with respect to a reference disparity d_{ref}:

$$PEC(x, y, d_{ref}) = \prod_{i=1}^{n} \frac{\sum_{x \in W_x} \sum_{y \in W_y} \left[I(x,y) + I(x + R_i d_{ref}, y) - \left| I(x,y) - I(x + R_i d_{ref}, y) \right| \right]}{\sum_{x \in W_x} \sum_{y \in W_y} \left[I(x,y) + I(x + R_i d_{ref}, y) \right]}. \qquad (3.20)$$

In practice, we maximize Eq. 3.20 using two approaches that differ only in algorithmic implementation. The first and more straightforward approach performs pairwise correlation on a set of images with respect to a point on a reference image. The individual correlation tables are then mapped onto the correlation table for a reference stereo pair using Eq. 3.19, and multiplied element-wise to obtain the PEC table. Since pixel resolution for longer baselines correspond to subpixel resolution for shorter baselines, we interpolate the correlation tables accordingly during the mapping step. Interpolated entries that lie outside the range of the individual pairwise correlation tables are assigned a value of 1 so as to effectively ignore them when multiplying the tables together. However, this might unduly weight those entries with terms from only a subset of all possible pairwise tables. To ensure that the shape of the PEC table is preserved, we then weight each PEC entry by the squared number of individual correlation tables that contributed to its product. The resulting peak represents the true disparity for the reference stereo pair, which corresponds to a unique depth.

While the first method closely follows the reasoning presented in Sect.3.3.2, it requires n correlation tables to be generated and stored, which slows down processing. The second method hence aims to build the PEC table in a single step, thus reducing the amount of memory and memory accesses by a factor of n. For each entry in the correlation table for a reference stereo pair, the corresponding element in every other pairwise correlation table is located based on the geometric constraint (Eq. 3.19) and multiplied directly into the reference table entry. In this way, pairwise correlation tables do not have to be separately generated and the PEC table is built more efficiently. From several experimental trials, we have found the second method to be 10-20 times faster than the first method.

Given a set of $n + 1$ images, we select a reference image and apply the efficient

PEC optimization algorithm on a sliding window of user-defined size, with the image dataset and baselines as inputs. For a given window, the disparity corresponding to the peak of the resulting PEC table is determined and stored in a disparity map. To further refine the disparity measurements, we may also fit a Gaussian curve to the peak of the PEC table to obtain subpixel disparity estimates.

Finally, we filter the disparity estimates to keep only high-confidence values. Two filtering criteria are used. First, we recall that $d_y = 0\,\forall x, y$ for a set of stereo pairs laterally displaced in only the x-direction. Hence, any disparity estimate with a y-coordinate exceeding a threshold close to zero is deemed faulty and filtered out. Second, we deem a disparity (or depth) estimate valid only if the peak PEC value exceeds a certain threshold. Too low a PEC value indicates that not all images are correlating well with the window in the reference view, perhaps due to the lack of visibility or specularities, and thus the corresponding disparity estimate is eliminated.

3.4 Results

To demonstrate the effectiveness of the PEC-based multi-baseline stereo algorithm, we have applied it on both synthetic and real stereo images. We also compare its performance to that of conventional binocular stereo.

3.4.1 Synthetic Images

As mentioned earlier, the synthetic image dataset consists of 5 images of a ramp function taken at equally spaced positions along the x-axis. A speckle illumination pattern is artificially projected to increase signal-to-noise ratio and avoid problems arising from insufficient texture or varying brightness. The central view (Image 3) is taken to be the reference view (Fig. 3-5a), and the other views are generated by shifting each point by the projected disparity within a range of 15-25 pixels using Fourier-based translation. Fig. 3-5c shows the disparity map for the Image 1-3 stereo pair, recovered using the proposed multi-baseline stereo algorithm with a 5-by-5-pixel window. The kinks in the disparity map are due to the limited pixel resolution in

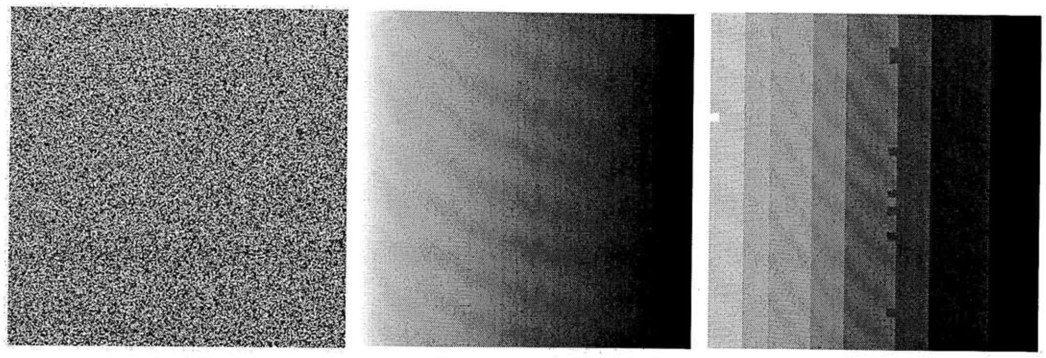


Figure 3-5: (a) Central view of synthetic image of ramp function. (b) Grouth truth disparity map. (c) Disparity map obtained using the PEC-based multi-baseline stereo method with a 5-by-5 pixel window.

estimating the peak of PEC tables. Visual comparison with ground truth (Fig. 3-5b), however, reveals the accuracy and robustness of the algorithm in reconstructing the shape of the ramp, even with a small correlation window.

3.4.2 Real Images

Encouraged by these results, we tested the algorithm on a real-world scene of an approximately-Lambertian carton, taken again at 5 equally locations along the x-axis and cropped to include only the corner of the carton (Fig. 3-6a). The whole process took roughly 2 minutes in MATLAB on a 2GHz Core 2 Duo computer. Fig. 3-6b shows the 3D surface reconstruction of the carton from the perspective of the central reference image. Majority of the points have been reconstructed accurately, in spite of the large disparities exceeding 40 pixels for the closest objects, thereby attesting to the robustness of PEC-based correlation correction. However, the algorithm performs poorly for regions with low texture, which is expected since the signal-to-noise ratio is too low for correlation to be carried out accurately. After filtering, this results in holes in the disparity map and 3D surface. Furthermore, the disparity map is especially noisy along the edges of the central image, roughly corresponding to the edges of the carton. These erratic spikes occur due to the effect of occlusions along the boundaries of the carton's silhouette. Images taken with large baselines are much less likely to see points on the carton's edge and thus no match can be found. As

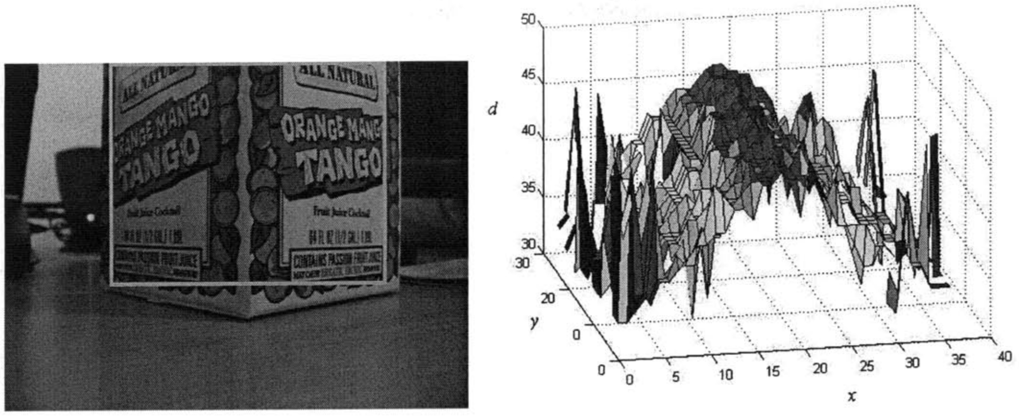


Figure 3-6: (a) Central reference view of carton dataset. The orange rectangle outlines the region of interest for correlation. (b) 3D surface reconstruction of carton. Holes and inaccuracies arise for regions with low texture, while disparity estimates along the right and left edges are noisy due to occlusion.

the baselines become longer, the effect of geometric aberrations also becomes more severe. Both effects substantially increase the probability of false matches and thus diminish the accuracy of the algorithm.

Nonetheless, in general, the PEC-based multi-baseline stereo algorithm manages to obtain relatively accurate distance estimates for highly-textured Lambertian surfaces. Since it accumulates multiple images, sufficient precision is still achievable while avoiding matching ambiguities.

3.4.3 Performance Comparison

Lastly, we compare the performance of the multi-baseline algorithm with that of conventional binocular stereo. Fig. 3-7 shows the percentage of correct disparities recovered using both methods for the synthetic random-dot image dataset. A pixel is defined to have a correct disparity if its value differs from the ground truth by less than 1, and occluded pixels are also ignored in calculating percentages. For window sizes larger than 7×7, both the multi-baseline and binocular stereo methods are able to accurately recover the disparity of the image pair, because the larger windows contain sufficient intensity variation to achieve reliable matching. However, at a small support window size of 3×3, binocular stereo using a single baseline only recovers

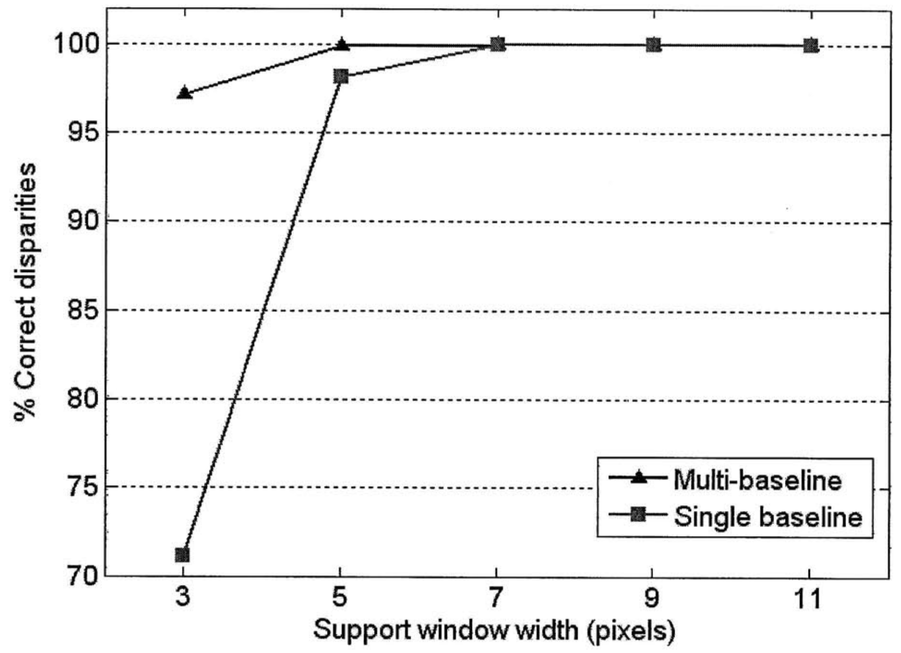


Figure 3-7: Plot of percentage of correct disparities against the width of the support window, for the multi-baseline method (blue) and conventional binocular stereo (red) applied to the synthetic image dataset. The multi-baseline method is robust against different sizes of support window due to the elimination of spurious matches through cost function aggregation.

about 70% of correct disparity, compared to 97% for the multi-baseline method. This shows that the multi-baseline algorithm is fairly robust and insensitive to different support window sizes. Even when there is insufficient signal for conventional binocular stereo to select the true cost minimum, the aggregation of signal content from other baselines and the elimination of false minima through element-wise multiplication enables the multi-baseline approach to still perform well.

3.5 Discussion

Having seen the utility of multi-baseline stereo, we shall now briefly discuss its advantages and limitations.

One key advantage of multi-baseline stereo is that information from all the stereo pairs are integrated into a final decision of the true disparity, as opposed to being used sequentially to refine correspondence estimates. Using the SSSD-in-inverse distance or PEC evaluation functions, geometric consistency across stereo pairs is elegantly enforced all at once without relying on the accuracy of previous correspondence measurements that may be noisy and faulty.

Furthermore, the multi-baseline stereo approach does not restrict the choice of similarity criterion, and thus the cost function may be freely changed depending on the application. In our implementation, we employed error correlation, which is well-suited for real-time compressed image processing. Other sophisticated similarity measures such as shiftable windows [8], adaptive windows [7] or locally adaptive support weights [9] may also be used to better preserve depth discontinuities. In particular, using correlation functions opens up possibilities for employing hierarchical coarse-to-fine processing, since correlation values obtained at different resolutions may be summed together.

On the other hand, a major drawback to the multi-baseline stereo method is that it only gives a depth map from a reference perspective, thus unduly weighting the significance of one image over the rest. Parts of the scene that are occluded in the reference image cannot be reconstructed using this method.

Also, while the algorithm uses multiple stereo pairs, not all possible pairs of images are exploited and more information (even if redundant) may presumably be extracted by extending the algorithm. However, using all possible pairs causes the algorithmic complexity to scale with $O(n^2)$ in the number of images, instead of $O(n)$, making the extended method unsuitable for large image datasets.

Another limitation of the multi-baseline stereo technique is that it reconstructs surfaces using a fronto-parallel approximation (i.e. each point in a scene is modelled as a flat plane perpendicular to the optical axis) and is thus sensitive to differences in foreshortening. This is not optimal when reconstructing slanted surfaces, such as that of the drink carton. Warped windows might be used to overcome this problem. In Chap. 4, we also describe a method to account for surface orientations when performing stereo matching.

Finally, this method implicitly requires the object to be visible in almost all views.

Hence, it performs poorly for widely-distributed views, because the effect of occlusions becomes so significant that good matches between image windows cannot be found. More sophisticated occlusion reasoning may then have to be incorporated into the optimization cost function for larger baselines.

Chapter 4

Multi-Dimensional Correlation

4.1 Introduction

The central task of stereo matching is finding corresponding points between different images. To determine a match, conventional area-based techniques use a fixed support window to measure the degree of similarity between two image regions. This implicitly assumes that all points within the window are locally fronto-parallel so that the intensity patterns around corresponding points are identical and only displaced from each other. Although this assumption is valid for small window sizes, in general, the viewing geometry and local surface orientation with respect to the camera cause pixel displacements within the support window to vary. Hence, correlation using fixed windows may produce an incorrect peak value, resulting in poor depth estimates. Instead, support windows should be deformed based on the surface orientations of the enclosed regions to compensate for this foreshortening effect.

To describe three-dimensional shape, the local surface normals and curvatures may be computed by fitting a spline function over the resulting point cloud after triangulation. However, this is not ideal because fixed window correlation incurs errors due to foreshortening or noise and thus the fitted surface is also likely to be inaccurate. Moreover, the original images which inherently contain information about the surface properties are not fully utilized. A stereo matching algorithm that directly considers surface orientation while searching for the best depth is thus desirable.

This chapter presents an algorithm that simultaneously recovers both depth and surface orientation from a set of images. The imaged surface is locally approximated by a plane and the shape of the support windows is altered based on the plane's orientation. Unlike conventional methods which only consider depth, the proposed method performs correlation over multiple dimensions: one for depth and two for surface orientation. Alternatively, the use of three cameras enables the 2D surface orientation search to be reduced to two 1D searches along epipolar lines. Finally, to reduce computational cost, we propose an efficient way of organizing the correlation process by building the correlation table after each pixel comparison, thereby minimizing the number of redundant comparisons.

4.2 Related Work

Numerous local stereo matching techniques have been proposed to counter the effects of projective distortion. An early approach was proposed by Kanade and Okutomi [7] to adaptively change the shape and size of the support window for each pixel. Their algorithm models the uncertainty of disparity estimates within a window based on the local statistics of intensity and disparity. In so doing, one can find the optimal window that estimates disparity with the lowest uncertainty by balancing the trade-off between signal-to-noise ratio and depth variation. However, the adaptive window algorithm requires an iterative search for the best windows for every pixel, which is usually very computationally expensive. Moreover, the algorithm does not explicitly extract the local surface orientations and only gives a depth estimate as its output.

Other methods compute the surface orientation by treating the local image distortions from surface tilt as useful signal rather than noise to be minimized. Jones and Malik used a set of linear oriented filters at different scales to directly recover local surface orientations [16]. Their approach is inspired by how the human visual system makes use of orientation and spatial frequency differences between the left and right views to perceive 3D shape. Robert and Hebert search for the best homography that maps a window in the left image to a skewed window in the right image [17].

The most likely orientation for a given window is found by minimizing the matching cost over the space of all possible window skewing parameters, as determined from knowledge of epipolar geometry and intrinsic camera parameters. While these methods directly measure local surface orientation, they unfortunately treat the problem of finding surface orientations as separate from that of depth estimation by assuming that point correspondences have already been correctly and precisely found. Hence, these methods do not take into account the local window deformations needed to refine depth estimates.

Devernay and Faugeras propose a fine correlation algorithm that allows the window to locally deform between image pairs [18]. For a given pixel (x, y) in the left image with disparity $d(x, y)$, nearby pixels have a disparity close to $d(x, y)$ but distorted by higher order terms in the local Taylor expansion of disparity. Hence, a square window in the left image appears as a sheared and slanted rectangle in the right image according to the local variation of disparity. The fine correlation algorithm first treats the surface as locally fronto-parallel, and then estimates the disparity and its derivatives that maximize the correlation between both image regions. From the derivatives of disparity, it is then possible to determine the surface differential properties such as orientation and curvatures from the image data. Although the method of Devernay and Faugeras overcomes the limitations of the fronto-parallel assumption, it however represents the local window deformations in terms of disparity derivatives rather than surface orientations. Therefore, since different image pairs have different disparities and thus derivatives, information from multiple images cannot be easily combined.

Hattori and Maki also employ an iterative framework that initially assumes the disparity to be locally constant and then accounts for surface orientation to refine the depth estimate [19]. However, instead of doing an exhaustive search for the parameters of image distortion, their algorithm directly computes surface orientations from intensity gradients within the support window, assuming an affine intensity variation model to compensate for differences in image contrast. With a similar intent of increasing computational efficiency, Xu and Ahuja propose a three-camera configuration whereby the brute-force search for the surface orientation parameters is broken down into two simpler 1D searches along two pairs of epipolar lines [20].

The work presented in this chapter belongs to the same category of techniques which allow the matching window to locally deform according to the surface orientation. We first formulate a theoretical model for local image deformation in both the general case and the standard rectified camera geometry. Based on this model, we then explore three methods of performing multi-dimensional correlation. The first method naively builds the correlation table along all three dimensions of depth and local tilt. The other two methods seek to increase computational efficiency by noting redundant terms in the computation of correlation values and by using a classical Levenberg-Marquardt minimization algorithm.

4.3 Local Image Deformation

We may approximate the local surface within an image window as a plane in 3D space. Strictly speaking, the left and right images of a given plane are related by a 3×3 homography matrix, causing the local intensity patterns to be warped between images. However, since the depth variation within the window is small, we may approximate the exact image deformation by a simpler 2×2 affine transformation matrix. Furthermore, we note from the epipolar geometry that any image deformation must occur along corresponding epipolar lines. Hence, using the extrinsic and intrinsic camera parameters, it is possible to model the local image deformation along only a 1D epipolar line instead of the 2D image plane. In this section, we present a theoretical analysis of these deformations.

4.3.1 Cameras In Generic Positions

Fig. 4-1a shows the general viewing geometry within the epipolar plane of a given point $\mathbf{P}$ on the object surface. Without loss of generality, we set the 2D world origin at the left camera center $\mathbf{C}_1$, and encode the relative position of the right camera $\mathbf{C}_2$

by a rotation $\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$ and a translation $\mathbf{T} = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}$. The matrices $\mathbf{R}$ and $\mathbf{T}$ may be obtained from extrinsic camera parameters. Now, notice that the epipolar plane C_1C_2P intersects the object surface along a curve. We approximate a short segment along this curve PP' as a line, with $\mathbf{P} = (X, Z)^T$ and $\mathbf{P}' = (X + \Delta X, Z + \Delta Z)^T$. The left camera projects the 3D points $\mathbf{P}$ and $\mathbf{P}'$ onto the left epipolar line at locations x and $x + \delta x$ respectively. From the perspective projection model, we obtain

$$\begin{aligned} x_1 &= \frac{f_1}{Z} X && (4.1) \\ x_1 + \delta x_1 &= \frac{f_1}{Z + \Delta Z}(X + \Delta X), && (4.2) \end{aligned}$$

where f_1 is the focal length of the left camera. Similarly, the right camera projects the points $\mathbf{P}$ and $\mathbf{P}'$ onto the right epipolar line at coordinates x_2 and $x_2 + \delta x_2$, given by

$$\begin{aligned} x_2 &= \frac{f_2}{Z'} X' = \frac{f_2}{Z'}(R_{11}X + R_{12}Z + T_1) && (4.3) \\ x_2 + \delta x_2 &= \frac{f_2}{Z' + \Delta Z'}(X' + \Delta X') && (4.4) \\ &= \frac{f_2}{Z' + \Delta Z'}[R_{11}(X + \Delta X) + R_{12}(Z + \Delta Z) + T_1], && (4.5) \end{aligned}$$

where f_2 is the focal length of the right camera, and $(X', Z')^T$ and $(X' + \Delta X', Z' + \Delta Z')^T$ represent the respective coordinates of $\mathbf{P}$ and $\mathbf{P}'$ as viewed from the right camera frame. Let the equation of the line segment PP' be $Z = pX + r$, such that $\Delta Z = p\Delta X$. Since the depth variation within the short segment PP' is small, we can further assume that $\Delta Z \ll Z$ and $\Delta Z' \ll Z'$. Combining Eqs. 4.1-4.5 then yields

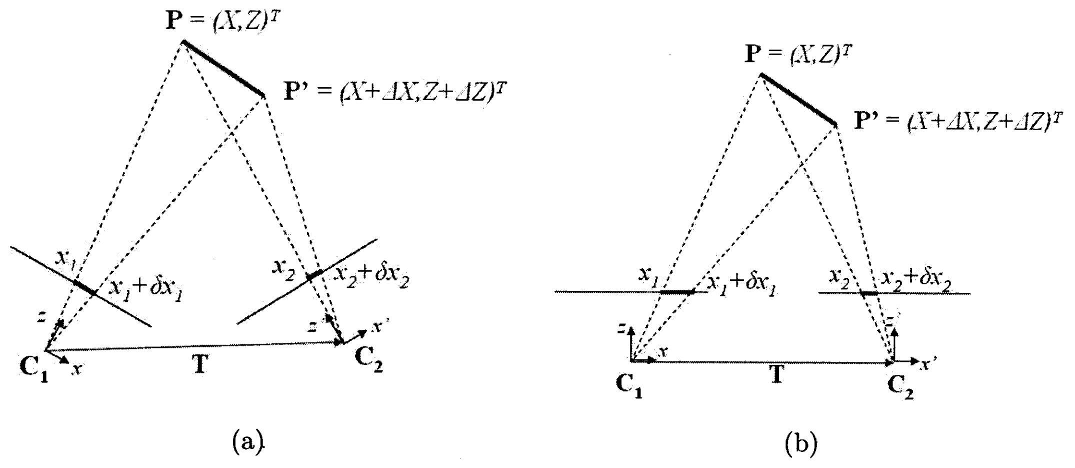


Figure 4-1: (a) Generic camera geometry within the epipolar plane. PP' is a short segment along the intersection of the epipolar plane with the object surface. (b) Standard rectified geometry.

$$\begin{aligned} \delta x_1 &= \frac{f_1}{Z}\Delta X && (4.6) \\ \delta x_2 &= \frac{f_2}{Z'}(R_{11}\Delta X + R_{12}\Delta Z) && (4.7) \\ &= \frac{f_2}{Z'}(R_{11} + R_{12}p)\Delta X, && (4.8) \end{aligned}$$

thereby giving the following relationship between δx_1 and δx_2

$$\delta x_2 = \frac{f_2}{f_1}\frac{Z}{Z'}(R_{11} + R_{12}p)\delta x_1. \quad (4.9)$$

Eq. 4.11 represents the local image deformation between two windows centered at x_1 and x_2 in the left and right images respectively. A pixel at $x_1 + \delta x_1$ in the left image window corresponds to the pixel $x_2 + \delta x_2$ in the right image window, for a given depth Z and surface orientation p. By finding the best combination of Z and p that maximizes the correlation between the two image windows, we may thus simultaneously recover both the depth and surface orientation of the point $\mathbf{P}$ within the epipolar plane. This forms the basis of multi-dimensional correlation.

4.3.2 Standard Geometry

Notice, however, that in the standard rectified geometry where the optical axes of the two cameras are perpendicular to the baseline (Fig. 4-1b), Eq. 4.11 unfortunately does not allow us to determine the surface orientation p. In this case, the relative rotation is simply $\mathbf{R} = \mathbf{I}$ and the two cameras are only displaced from each other by the translation vector $\mathbf{T}$. The depth coordinates Z and Z' are also equal. Hence, Eq. 4.11 reduces to

$$\delta x_2 = \frac{f_2}{f_1}\delta x_1, \tag{4.10}$$

which only accounts for the scaling effect from the difference in focal lengths and is clearly inadequate for our purposes. Nonetheless, our intuition tells us that some degree of foreshortening is present even in the standard rectified geometry, especially for large baselines. Hence, we revisit our earlier approximation that $Z \approx Z + \Delta Z$ and $Z' \approx Z' + \Delta Z'$ so as to derive the image deformation equation exactly. For simplicity, we assume that $f_1 = f_2 = f$ and note that $R_{11} = 1$, $R_{12} = 0$ and $Z = Z'$ under this configuration. The ratio between δx_1 and δx_2 is then given by

$$\begin{aligned} \frac{\delta x_2}{\delta x_1} &= \frac{(X + \Delta X + T)Z - (X + T)(Z + \Delta Z)}{(X + \Delta X)Z - X(Z + \Delta Z)} && (4.11) \\ &= 1 - \frac{T\Delta Z}{Z\Delta X - X\Delta Z} && (4.12) \\ &= 1 - \frac{Tp}{Z - pX}. && (4.13) \end{aligned}$$

Therefore, we see that the exact local image deformation differs from Eq. 4.10 by a term that depends on the baseline T. By recalling from the perspective projection model that $X = \frac{Zx_1}{f}$ and substituting into Eq. 4.13, we further obtain

$$\frac{\delta x_2}{\delta x_1} = 1 - \frac{Tf}{Z}\frac{p}{f - px_1}. \tag{4.14}$$

Hence, in the neighborhood of a given pixel x_1 in the left image, the local deformation around the corresponding pixel x_2 in the right image may be determined based on the depth Z and tilt p of the imaged point. More precisely, by noting that disparity is inversely related to depth via the expression $x_2 - x_1 = \frac{Tf}{Z}$, we may rewrite Eq. 4.14 as

$$x_2 + \delta x_2 = (x_1 + \delta x_1) + \left(\frac{Tf}{Z}\right) \frac{f - p(x_1 + \delta x_1)}{f - px_1}. \tag{4.15}$$

Eq. 4.15 explicitly expresses the relationship between the left image window (i.e. the set of neighboring pixels $\{x_1 + \delta x_1\}$) and the corresponding (deformed) window in the right image $\{x_2 + \delta x_2\}$. Each pixel in the image window is mapped to the same (Z, p) location. In particular, when $p = 0$ (i.e. the surface is fronto-parallel), Eq. 4.15 indicates that all points within the window will have a constant disparity $\frac{Tf}{Z}$, as expected. Also note that in the rectified camera geometry, the epipolar lines are horizontal scanlines, and thus x_1 and x_2 are the pixel coordinates along the x-axis. Using Eq. 4.15, the best depth and surface orientation may then be found from the peak of the resulting multi-dimensional correlation table in Z and p.

4.3.3 Multiple Views

So far, we have only described the local image deformation within the epipolar plane, allowing us to obtain the depth and surface orientation within this plane. A minimum of three calibrated cameras spaced in different epipolar planes must be used in order to recover the same parameters in 3D space. First, for a given pixel in a reference view, the corresponding pixels in the other images is calculated. For a range of candidate depths and surface orientations, the corresponding image windows are deformed according to Eqs. 4.9 or 4.15. The correlation values from each stereo pair may then be summed up (using the multi-baseline method described in Chap. 3), and the depth estimate is obtained from the largest sum of correlation values. At this depth, the best-matching surface orientations in each pairwise multi-dimensional correlation table is found. Each surface orientation estimate corresponds to a vector tangential to the object surface that lies in the respective epipolar plane. Finally, we may estimate the local surface normal of the object by simply taking the cross-product of any two tangential vectors.

4.4 Evaluation

For simplicity, we have chosen to implement the multi-dimensional correlation (MDC) scheme using only two views in a standard rectified geometry. As such, we only need to consider image deformations in the horizontal x-direction. To evaluate the effectiveness of our method, we generated synthetic stereo pairs by simulating the perspective projection of two parallel cameras. Both cameras have a simulated focal length of 1000 pixels and are displaced from each other by 5 a.u. Based on the known geometry of the synthetic scene, we may also obtain the ground truth depth maps and surface orientation maps.

Fig. 4-2 shows a pair of synthetic images of a convex corner with flat sides. The object has a depth range of 80-100 a.u., where the apex is the closest to the camera origin. The sides are planes with a constant tilt of $p = \pm 2$ in the x- direction and no tilt in the y- direction. This dataset enabled us to test the consistency of the multi-dimensional correlation algorithm for different points with the same surface tilt.

Fig. 4-3 shows another synthetic image pair of a convex parabolic surface, using the same simulated cameras as before. The object has a depth range of 90-100 a.u., with its middle being closer to the camera center than the sides. The parabolic dataset allowed us to test the algorithm on varying surface orientations within the image.

In the rest of this chapter, we will explain and evaluate the three different implementations of multi-dimensional correlation mentioned previously, namely the basic method, efficient method, and minimization approach.

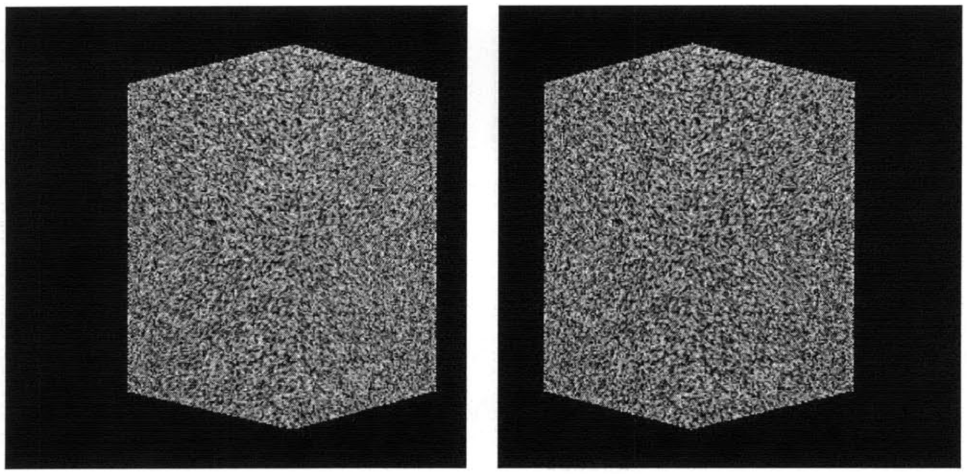

(a) (b)

Figure 4-2: Left and right images of a simulated convex corner with flat sides. The object has a depth range of 80-100 a.u. and the sides are flat planes with a constant tilt of $p = \pm 2$ in the x- direction.

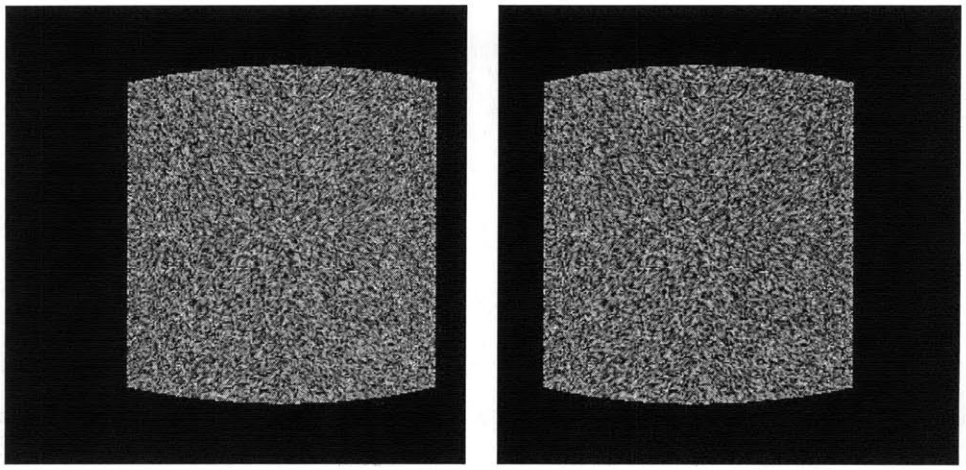

(a) (b)

Figure 4-3: Left and right images of a simulated convex parabolic surface. The object has a depth range of 90-100 a.u. and is only tilted in the x- direction.

4.5 Basic Multi-Dimensional Correlation

The most straightforward method of performing multi-dimensional correlation is to calculate the correlation value for every possible (Z, p) in the correlation table. Since a search along multiple dimensions is computationally costly, we increase efficiency by adopting a coarse-to-fine hierarchical approach. An initial depth estimate is obtained by assuming that the surface is locally fronto-parallel and performing fixed window correlation. We then build the correlation table by evaluating depth values within ± 1 of the initial estimate and surface orientations p in the range of $[-3, 3]$, at a user-defined resolution of 0.1 in both dimensions. For each table entry, we select a square or rectangular window around the current pixel in the left image and compute the disparities $(d_x, d_y) = (d, 0)$ in the corresponding right image window using the image deformation equation (Eq. 4.15). Since the windows being compared are typically of different shapes, we interpolate the right image at the calculated locations before summing the correlation criterion over the window.

While a variety of correlation measures may be used, we have selected the error correlation function [13] that was also used in Chap. 3, reproduced here for convenience:

$$\Phi(x, y, d_x, d_y) = \frac{\sum_{x \in W_x} \sum_{y \in W_y} [I_1(x, y) + I_2(x + d_x, y + d_y) - |I_1(x, y) - I_2(x + d_x, y + d_y)|]}{\sum_{x \in W_x} \sum_{y \in W_y} [I_1(x, y) + I_2(x + d_x, y + d_y)]} \tag{4.16}$$

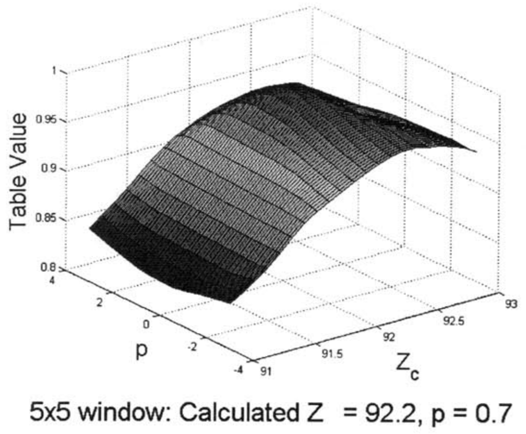
The error correlation function efficiently measures the degree of similarity, while also normalizing the values to lie within 0 and 1. Once all the entries of the MDC table are filled, the resulting peak gives the most probable depth Z and local tilt p for the current pixel of interest.

Although this basic implementation may be inefficient (typical processing times in MATLAB are on the order of 10 seconds per pixel), it does allow us to easily visualize the correlation table. From numerous trials, we have discovered that the accuracy of the algorithm is fairly sensitive to the size of the support window. Fig. 4-4 shows typical correlation tables obtained using window sizes of 5×5, 15×15 and 35×35 for the same pixel in the parabolic dataset. For small image windows, the correlation table peak is not distinct nor accurate due to the lack of signal. Conversely, at large window sizes, the local planar approximation begins to break down as the window contains a curved surface, and the peak again becomes less discernible. While small windows are generally ideal to avoid straddling depth discontinuities, we have found larger window sizes to be necessary in order to obtain accurate depth and tilt values. This is unfortunately due to the limited foreshortening effect in the standard rectified geometry, such that substantial local image deformation is only present in larger windows. Presumably, a verged stereo configuration will allow the use of smaller windows, leading to greater robustness in correlation.

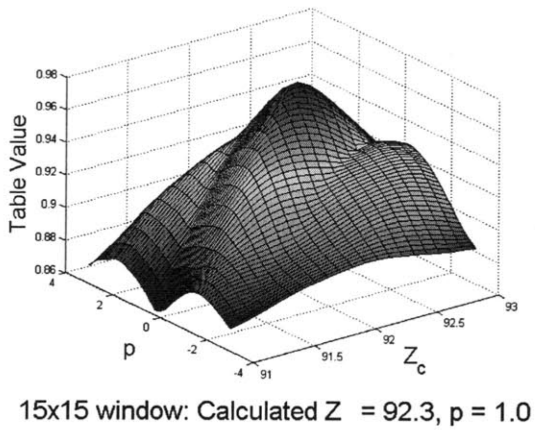
4.6 Efficient Multi-Dimensional Correlation

In the basic implementation of multi-dimensional correlation, the calculation of multiple entries in the correlation table involve the same pixel comparisons between the left and right images. As such, a more efficient method would be to fill in all the table entries corresponding to each pixel comparison made, instead of building the table entry by entry. Below we describe the methodology.

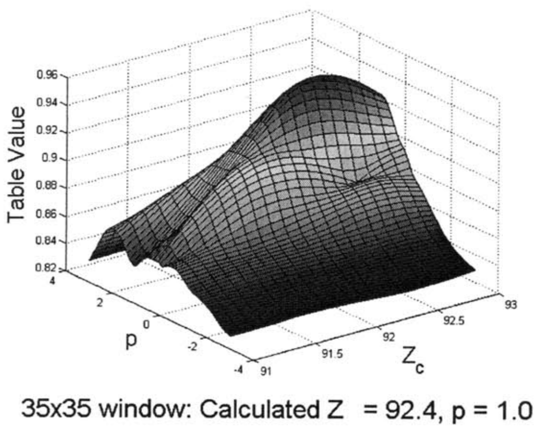
From fixed window correlation, we obtain an initial estimate of where the pixels in the left image window would be in the right image. Due to local window deformation, however, the actual points in the right image corresponding to those in the left image window will likely deviate from this estimated location. We arbitrarily assume that any local deformation occurs within a small neighborhood of ± 1 pixel from the initial correspondence, and interpolate the right image around this region to a finer resolution of 0.05 pixel. Each pixel in the left image window is then compared with points in its corresponding fine grid. For every point-wise comparison, the pixel intensities are multiplied together and the resulting value is added into every possible entry in the MDC table for which the given comparison is made. These entries lie along some



(a)



(b)



(c)

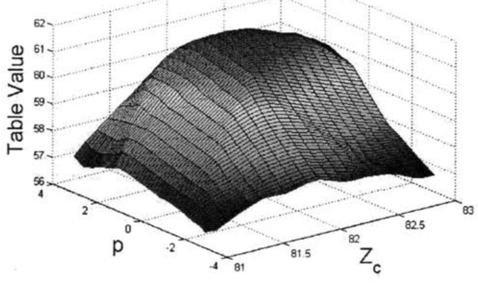
Figure 4-4: Typical MDC tables obtained using window sizes of 5×5, 15×15 and 35×35 pixels, as applied to the parabola dataset. Based on ground truth, the actual depth is $Z = 92.3$ and the actual local tilt is $p = 1.0$. The small 5×5 window produces a non-distinct and inaccurate peak, while the large 35×35 window also begins to show signs of inaccuracy. An intermediate window size between 15×15 to 25×25 tends to perform well.

non-linear function $Z = \phi(p)$, as obtained from Eq. 4.15 for a given point $x_1 + \delta x_1$ in the left window being compared to another point $x_2 + \delta x_2$ in the right window. The entries are also quantized according to the user-defined resolution of the table. This process of multiplying pixel intensities and simultaneously adding into multiple table entries is then repeated for every pixel within the left image window to populate the MDC table. The table may then be searched for the peak value representing the most likely values of depth Z and surface orientation p.

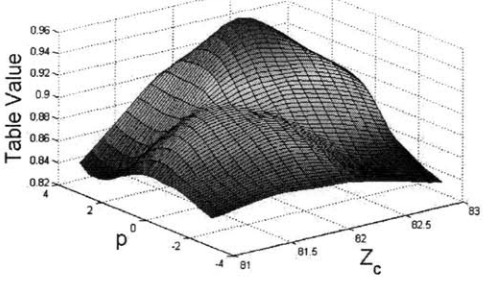
Essentially, this efficient method accumulates terms in the correlation sum as new pixel comparisons are added, rather than summing over the regions being compared as in standard correlation. Another way to think about the efficient method is that each pointwise comparison is placed into bins in the MDC table lattice and the final sum of elements in each bin gives the correlation value. In the limit where every pointwise comparison for each table entry has been included, the efficient MDC table is equivalent to that of the basic implementation.

The key advantage of the efficient method is that duplicate pixel comparisons may be avoided, thus improving computational efficiency. The basic MDC scheme has a computational intensity of $O(m^2R^N)$, where m is the width of the image window, R is the number of entries in any given dimension and N is the number of dimensions. Conversely, the efficient method only requires a computational load of $O(m^2Q)$, where Q is the number of points in the fine interrogation grid for the image window, assuming that the binning cost is negligible. As such, significant computational savings are theoretically realizable.

However, by only considering each pixel comparison, the efficient method cannot account for the characteristics of the entire matching window so as to normalize the correlation table entries. In effect, the multiplication of pixel intensities is equivalent to standard unnormalized cross-correlation. The lack of normalization allows certain high intensity pixels to dominate in the correlation sum, thereby giving rise to inaccuracies in the calculation of the table peak. This effect is best demonstrated by comparing the normalized and unnormalized tables obtained using the basic implementation. As shown in Fig. 4-5, the unnormalized table produces a flat peak with a



(a)



(b)

Figure 4-5: Comparison of typical (a) unnormalized and (b) normalized MDC tables, for the same pixel in the corner dataset and using a constant window size of 15×15. For this example, the flat peak in the unnormalized table yields inaccurate values of Z and p, as opposed to the accurate and well-defined peak in the normalized table.

range of possible depths and local tilts, instead of a single value with high statistical significance. In contrast, the normalized table for the same pixel and window size produces an accurate and well-defined peak. A rough way to achieve some degree of normalization in the efficient method is to take the mean of all the correlation values within a particular (Z, p) bin. However, without keeping track of the local means of the matching windows, exact normalization cannot be achieved.

In practice, the shape of the efficient MDC table is also fairly sensitive to the resolution at which the table values are binned. Too coarse a resolution in either dimension (i.e. too few bins) causes the combined correlation sum of adjacent (Z, p) entries to become lumped together, while too fine a resolution (i.e. too many bins) prevents values from individual pixel comparisons from being meaningfully aggregated into a single table entry. In our implementation, we have found that using the same resolution as that at which the disparity and surface tilt were evaluated tends to work well.

Fig. 4-6 shows a typical histogram of the number of pointwise comparisons belonging to all (Z, p) bins. Each bin represents a particular combination of correspondences between the left and right image windows, or alternatively, the locations at which the non-linear functions $Z = \phi(p)$ intersect. A high number of correspondences in a given bin is analogous to using a larger image window, and thus the MDC value is taken to be more reliable. As expected, along the $p = 0$ plane (i.e. flat window), equal numbers of pointwise comparisons at found at different depths corresponding to the candidate disparities sampled. This is equivalent to matching the left image window with the undeformed right image window at the candidate disparity, and placing the correlation sum at the appropriate table entry. Furthermore, we observe that the region closest to the initial depth estimate and around small tilt values produces MDC values that are most reliable, while the rest of the table contains too few correspondences for meaningful statistical conclusions to be drawn. This is unavoidable because for a given search range in the right image (in our case, ± 1 pixel), there are only limited (Z, p) combinations such that every point in the left image window also appears in the right image window. However, this observation has two key implications. First, although efficient MDC minimizes the number of redundant pointwise comparisons between the two images, it also performs a large number of comparisons that do not yield reliable correlation values. After all, correlation is an area-based similarity measure, and individual pointwise comparisons alone are not useful. Ideally, we would like to populate the entire correlation table with equally reliable measurements, as in basic MDC. Second, since the efficient MDC table is heavily populated by correlation values from only a few correspondences, some thresholding is required to remove unreliable table entries. In our implementation, we require at least 50% of the left image window to find correspondences in the right image search region before the MDC entry is considered valid.

After thresholding the efficient MDC table, we obtain a sparse set of reliable MDC values in the Z-p space, as shown in Fig. 4-7a. In this example, we use a 7×19 window on the corner dataset. To fill in holes in the efficient MDC table, we may also fit a surface over the reliable points, similar to that in Fig. 4-7b. From visual inspection, the shape of the fitted efficient MDC function is similar to that from basic MDC (Fig. 4-7c). However, due to the sparsity of the efficient MDC table, the shape of the fitted curve is highly sensitive to the grid resolution and often incurs errors from overfitting. Moreover, the efficient MDC table undergoes a further loss of resolution after curve-fitting. For the example in Fig. 4-7, the peak of the fitted curve gives

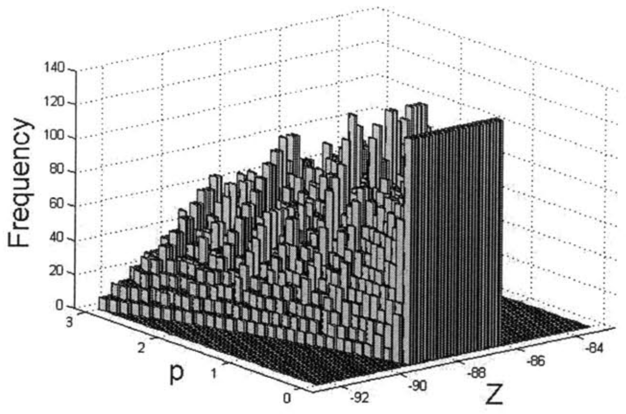
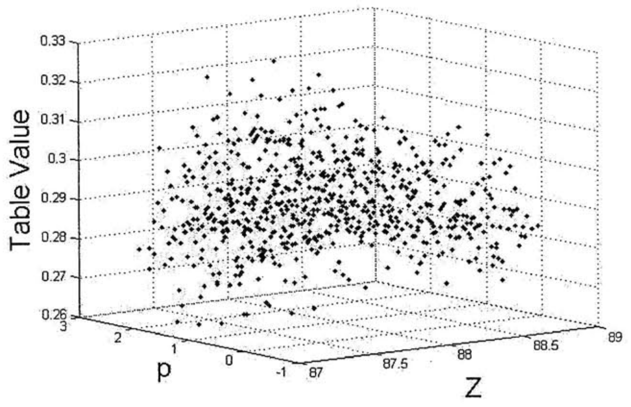


Figure 4-6: Typical histogram of the number of pointwise comparisons for each (Z, p) bin. Notice that a higher number of correspondences are usually found at tilt values and are thus more reliable, while the rest of the table is sparsely populated.

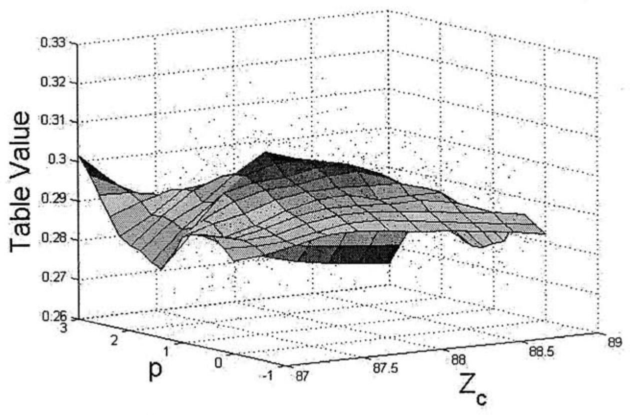
an estimate of $Z = 87.8$ and $p = 1.8$, while the ground truth value is $Z = 88.1$ and $p = 2.0$.

A final note concerns the unexpectedly slow processing speed of the efficient MDC method. Unfortunately, the pointwise comparisons and binning operations require a large number of memory accesses, which is computationally intensive in MATLAB. We leave a more efficient implementation in another programming environment as future work.

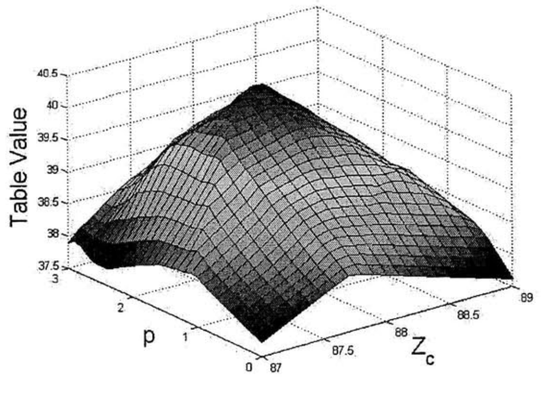
In summary, we have described the pipeline to perform efficient MDC on a set of images. While a theoretical analysis of the method predicts significant computational savings, our experiments have identified numerous tradeoffs in the reliability of MDC values and the resolution of the resulting MDC table. A potentially better implementation may be to build the MDC table entry-by-entry, but keep track of the individual pointwise comparisons to populate the rest of the table. As each entry is scanned, only the missing comparisons for the matching window points are made. Such a method reduces the processing load at the expense of memory usage, since each entry now stores both the pointwise correlation value as well as the location of the points.



(a)



(b)



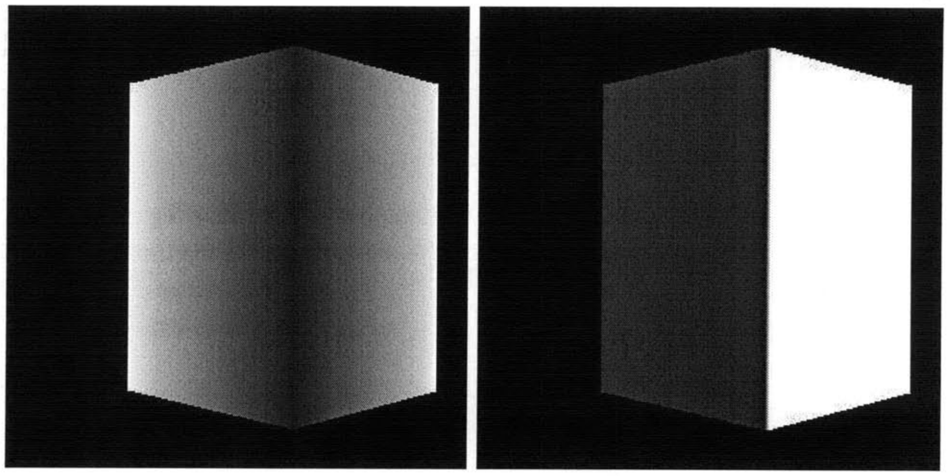
(c)

Figure 4-7: Typical MDC tables for a 7×19 window applied to the corner dataset. (a) Sparse point cloud representing the reliable values in the efficient MDC table after thresholding. (b) Efficient MDC function obtained by fitting a surface over the reliable points. (c) Basic (unnormalized) MDC table for the same pixel and window size.

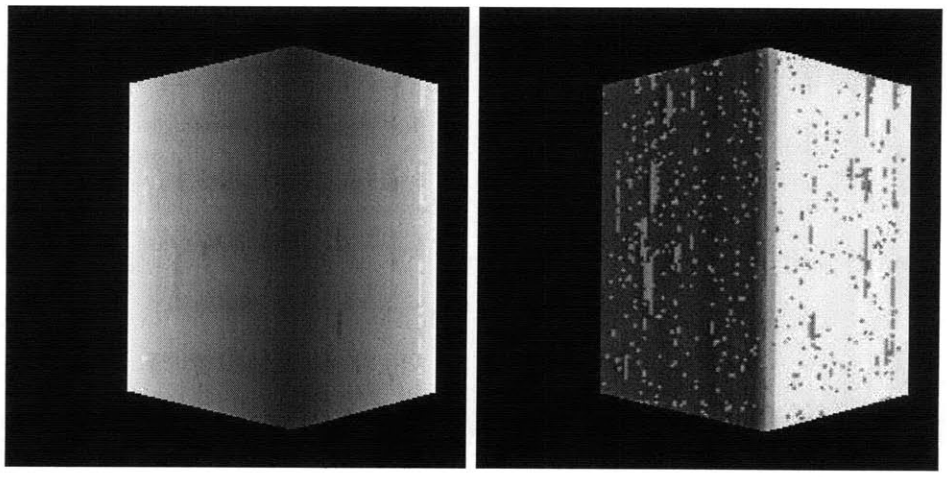
4.7 Numerical Minimization

In the third implementation of multi-dimensional correlation, we use a classical Levenberg-Marquardt minimization technique to determine the peak of the correlation curve (or more precisely, the minimum of the inverted correlation function). Again, we seek the values of depth Z and surface tilt p that maximize the correlation between the left and right image windows, as determined using Eq. 4.15. We initialize the numerical minimization step using the initial disparity estimate computed by fixed window correlation as the first component of the initialization vector, and a locally flat surface orientation $p = 0$ for the second component. Assuming the correlation function is locally well-behaved and unimodal, the output vector of the minimization technique then gives the most probable depth Z and surface orientation p of the imaged surface. The advantage of such a method is that not all entries of the MDC table have to be calculated, thus significantly cutting down on processing time.

Fig. 4-8 shows the resulting depth map and surface orientation map obtained by applying the MDC minimization approach to the corner dataset with a window size of 25×25. Indeed, by allowing the matching window to locally deform according to the surface orientation, the depth and surface orientation of each pixel may be obtained simultaneously and accurately. At this window size, 81.4% of surface orientations have been recovered correctly, where a correct value is taken to have an absolute error of less than 0.2. This result is promising, given the limited foreshortening effects and lack of deformation signal in the standard camera geometry.



(a)



(b)

Figure 4-8: (a) Ground truth depth map and surface orientation map of corner dataset. (b) Calculated depth map and surface orientation map by using the MDC minimization approach. At a window size of 25×25, 81.4% of the surface orientations have been recovered correctly.

Chapter 5

Randomized Stereo Matching

5.1 Introduction

Although the topic of stereo matching has been widely studied in computer vision, most research has focused on improving its accuracy and robustness, rather than processing speed. However, in certain real-time or data-intensive applications such as medical imaging or Particle Image Velocimetry (PIV), a computationally efficient block-matching method is desirable. This chapter presents a new algorithm that uses randomized sampling to quickly find correspondences across multiple frames and produce dense disparity maps.

The task of estimating a set of pixel-wise correspondences involves locating, for each block in one image region, the closest match in another image region based on a similarity measure. Conventional techniques typically carry out a brute-force search of all blocks within a search window of the second image. In the spatial domain, this is computationally expensive and scales as $O(BWS)$ for images and blocks of size S and B pixels, and a search window size of W pixels. Most of these comparisons, however, are redundant as we are ultimately interested in only the best matching one. Presumably, if we knew where to look, we could then narrow our search by focusing on the regions with high matching probabilities.

The inherent problem of conventional stereo methods is that they only make decisions locally and ignore the global structure of images. Marr and Poggio [21] propose that stereo disparity maps should be continuous almost everywhere and have unique values, since most scenes contain smooth and opaque surfaces, and discontinuities only occur along object boundaries. The continuity assumption implies that disparity values across adjacent pixels are relatively consistent. Hence, we may reasonably predict the disparity of a given pixel by referring to its neighbors' values, in so doing improving the efficiency of stereo matching.

The question remains: how do we set an initial disparity estimate? Given no prior information about the scene, at best we may randomly assign a disparity value to each pixel. While any single random disparity value is likely to be wrong, a small proportion of random assignments will also likely be correct, by virtue of the large number of assignments. We may thus exploit the large number of image pixels to obtain some accurate matches, and then use these as seeds to grow additional matches.

Based on these observations, we propose an iterative, randomized algorithm to efficiently compute dense disparity maps. The algorithm starts from an initial disparity estimate, which may be randomly assigned or obtained from prior information. In each iteration, we first enforce the continuity assumption by updating each pixel with the best disparity estimate in its neighborhood, and then randomly search around the current disparity to refine the estimate. We present a theoretical analysis and experimental results to demonstrate the convergence properties of this algorithm. Furthermore, since the randomized algorithm only considers a subset of all candidate blocks when searching for matches, it is expected to take less time and memory than traditional methods. We show empirically the potential savings in computation time by using our algorithm.

5.2 Related Work

Fast block-matching algorithms have found interest not only in the field of stereo matching, but also in the video compression community, where active research has been done over the last two decades to devise video coding standards. In video compression, block-matching algorithms are used to estimate the motion of a block between the current frame and the previous frame. Given that typically only some fraction of a video changes from frame to frame, we may encode and transmit a motion-compensated frame consisting of blocks from the previous frame, thereby reducing the number of bits transmitted and eliminating the spatial and temporal redundancy in video sequences.

To this end, the most straightforward approach is to exhaustively compare each block in the current frame with all blocks within a fixed search window of the previous frame, known as a Full Search (FS). The best match is determined by minimizing a cost function such as SSD or SAD. While FS guarantees an optimal compression quality, it is computationally intensive and thus unsuitable to deal with large amounts of video data. For instance, a 15×15 search window requires 225 block comparisons. As such, a plethora of sub-optimal algorithms have been designed to achieve a high compression quality in as few computation steps as possible.

An early attempt at fast block-matching is the Three Step Search (TSS) [22], shown in Fig. 5-1. Instead of searching every location within a search window, TSS searches at the central location $(0, 0)$ and the eight locations spaced around it at a step size of 4. From these nine locations, the algorithm selects the point with the lowest cost as the new search origin, at the same time halving the step size to 2. It repeats the process once more until the step size is 1 and selects the lowest cost location as the best match. Since TSS only makes a maximum of 25 comparisons, it requires 9 times fewer computation steps than FS, but trades off matching accuracy for efficiency. An enhanced version of TSS, known as the New Three Step Search (NTSS) [23], includes eight more blocks closer to the central region for comparison, so as to account for small motions near the center.

In the same vein as NTSS, the Four Step Search (FSS) [24] searches sequentially but with a smaller and fixed step size of 2 for the first three steps. At the fourth step, the step size is reduced to 1. At any stage, if the lowest cost is found at the central point, the algorithm transitions to the fourth step to quickly end the search. Furthermore, the shape of a search pattern may be changed from square to diamond,

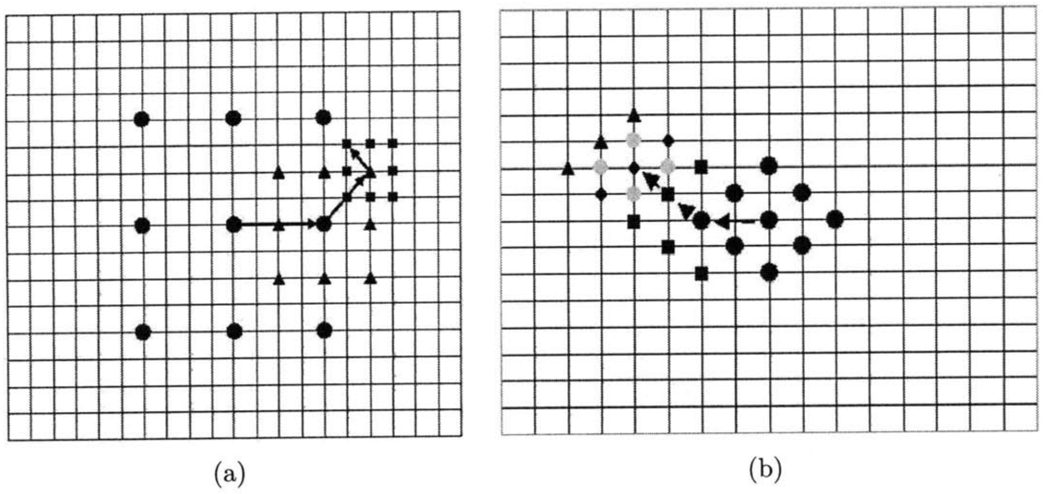


Figure 5-1: (a) Typical Three Step Search (TSS) procedure. (b) Typical Diamond Search procedure. The algorithm searches in five search steps using a diamond pattern of step size 2, except for the last step where the step size is reduced to 1.

and the limit on the number of steps may also be lifted to allow the global minimum to be found accurately [25]. Fig. 5-1 depicts the procedure for such a Diamond Search (DS).

While the above methods lead to considerable computational savings by reducing the number of block comparisons made, they do not exploit the observation that the motions of adjacent blocks are highly correlated (similar to our continuity assumption for stereo disparities). The class of algorithms that most resembles our randomized algorithm in this aspect is the Adaptive Rood Pattern Search (ARPS) [26]. ARPS seeks to predict the motion of a block by referring to its surrounding blocks across space and time. Besides searching at four equally-spaced locations about the center point of the current block, ARPS also searches at the location given by the motion vector of its left neighbor block. Because the likelihood of finding the correct match at the sampled locations is high, a small pattern search is sufficient to quickly complete the remaining search.

Unfortunately, all these fast block-matching algorithms implicitly assume that the cost function surface is unimodal, such that the costs at each search location increase monotonically from the global minimum. Hence, they run the risk of being trapped in local minima over a long search path, particularly when the image contains periodic patterns. In the case of ARPS, such errors may even be propagated to other parts of the image or to other frames. We thus opt for an iterative scheme that casts a wide search net during each iteration so as to escape from local minima. We also reduce propagation error by refining the disparity estimates over several iterations.

Finally, we note that the randomized algorithm presented here bears great similarity to the work by Barnes *et al.* on PatchMatch [27]. While the theoretical foundations are mostly identical, Barnes et al. focus on the algorithm's applicability to image editing, reshuffling and retargeting. In this chapter, however, we demonstrate the utility of the algorithm in a stereo matching framework.

5.3 Overview of Algorithm

In this section, we give an overview of the randomized stereo matching algorithm. Unlike other local window-based methods, the algorithm is *cooperative*, because it aggregates information from neighboring pixels in an interdependent manner so as to search more efficiently. Matching is also done at *random*, keeping in mind that using a sufficiently large number of random disparity assignments ensures that some of them would be accurate.

The algorithm comprises three main stages: Initialization, Propagation and Search. Each step is depicted in Fig. 5-2 and described in greater detail below.

5.3.1 Initialization

First, we define a disparity field $\mathbf{u}(x, y)$, where $\mathbf{u} = (u, v)$ is the displacement between a block centered at (x, y) in image I_1 and its closest match in image I_2. The disparity field $\mathbf{u}(x, y)$ is an array of the same size as I_1. We initialize $\mathbf{u}(x, y)$ by independently assigning uniformly distributed values to each (x, y) coordinate. The random displacements span a given search parameter, but do not exceed the boundaries of I_2. Once we have initialized the disparity field, we may store and use the same values for multiple frames, instead of recomputing a new random prior for every stereo pair.

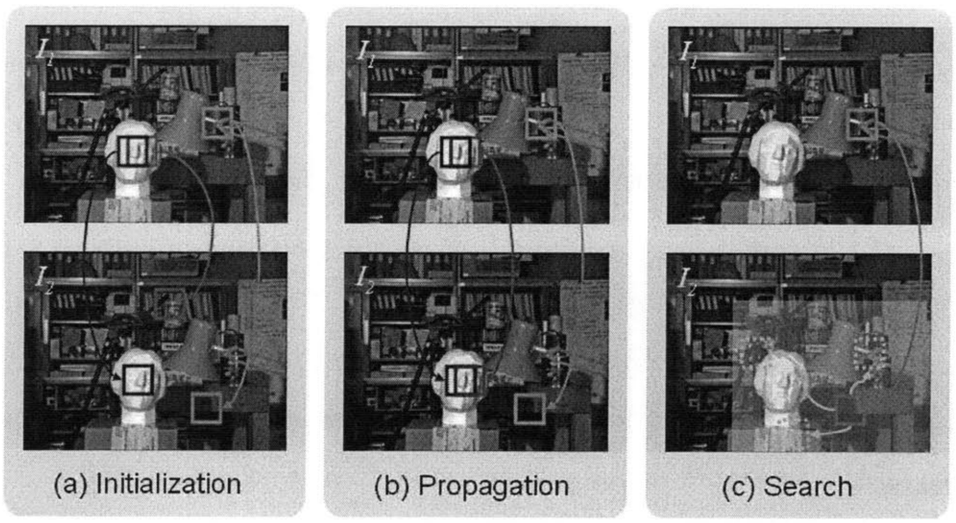


Figure 5-2: Stages of the randomized stereo matching algorithm: (a) Pixels are initially assigned random disparities; (b) Good matches (blue) are disseminated to neighbors (red); (c) Random local search to improve estimate.

To speed up convergence, we may also use prior knowledge of the scene or camera geometry to limit the range of disparities and impose ordering constraints. For instance, for a rectified left-right stereo pair, the horizontal disparity field for the left image may be confined to negative values, while the vertical disparity field may be set to zero assuming no image distortion.

Furthermore, we may use a hierarchical coarse-to-fine pyramid approach to obtain a more accurate initial disparity field. In our implementation, we use three pyramid levels, where the image at level l is obtained by downsampling the image at level $l-1$ by a factor of 0.5. Coarse block-matching is first performed on low-resolution images using a fixed block size and a search parameter of $2^{-l}w$, where w is the desired search width at the original resolution. The estimated disparity field from the top level of the pyramid is then used to initialize a narrower search at the next finer level, after multiplying and resizing by a factor of 2. Such a hierarchical procedure allows us to save processing time by performing most of the refinement at coarser and faster levels and reducing the number of iterations at finer and slower levels.

5.3.2 Iteration

After initialization, we iteratively update the disparity field. In each iteration, we cycle through every point in the disparity field, first propagating good matches from adjacent pixels and then carrying out a random search, before proceeding onto the next pixel.

5.3.2.1 Propagation

Since adjacent pixels belonging to the same object surface are likely to share similar disparity values, we seek to improve the current pixel's disparity estimate by referring to the neighbors immediately above and to its left. We thus compare the block at (x, y) with interrogation blocks centered at its current disparity $\mathbf{u}(x, y)$, as well as candidate disparities $\mathbf{u}(x-1, y)$ and $\mathbf{u}(x, y-1)$ for the left and top neighbors respectively. Let the degree of similarity between the block at (x, y) in image I_1 and the block at $(x, y) + \mathbf{u}$ in image I_2 be measured by a cost function $C(\mathbf{u})$. We then replace the current pixel's disparity with the value that minimizes $C(\mathbf{u})$ for the candidate disparities. Mathematically, this disparity update may be expressed as:

$$\mathbf{u}(x, y) = \arg\min\{C(\mathbf{u}(x, y)), C(\mathbf{u}(x-1, y)), C(\mathbf{u}(x, y-1))\}. \qquad (5.1)$$

Care is taken to ensure that the candidate disparities taken from surrounding pixels do not exceed the boundaries of I_2. For pixels on the edges of image I_1 which do not have neighbors to the left or above, we also modify Eq. 5.4 to make use of the best available information. Through the process of propagation, if the disparity at (x, y) is found correctly, then all pixels below and to the right of (x, y) lying in the same contiguous region will also converge to the correct value within a single iteration. The search is therefore highly efficient because it skips the computation of other less probable matches.

Furthermore, to allow good disparity values to spread to all parts of the image, we alternate propagation directions between iterations. On odd iterations, we scan through the disparity field from left to right, top to bottom, thereby propagating information downwards and to the right using Eq. 5.1. Conversely, on even iterations, we scan in reverse order from right to left, bottom to top, so as to propagate upwards and to the left. Consequently, the disparity update equation for even iterations is rewritten as:

$$\mathbf{u}(x, y) = \arg\min\{C(\mathbf{u}(x, y)), C(\mathbf{u}(x + 1, y)), C(\mathbf{u}(x, y + 1))\}. \tag{5.2}$$

5.3.2.2 Search

Next, we search around the current best disparity to further improve our estimate. This search may be carried out sequentially at equally spaced points from the current disparity vector, resembling the regular pattern of DS, TSS or FSS. However, following the implementation of [27], we only search at a small number of randomly spaced points to increase efficiency. While a single random search pattern is unlikely to track the correct disparity by itself, we again use the fact that many random searches across neighboring pixels will corroborate with each other through the process of propagation. For a given pixel at (x, y) with current disparity $\mathbf{u}(x, y)$, the set of disparities for random search $U = \{\mathbf{u}_i\}$ may be expressed as:

$$\mathbf{u}_i = \mathbf{u}(x, y) + \alpha^i \begin{pmatrix} w_x r_x \\ w_y r_y \end{pmatrix}, \tag{5.3}$$

where w_x and w_y are the maximum search radii in the x and y directions, and r_x and r_y are i.i.d. uniform random variables in the interval $[-1, 1]$. α is a scaling factor that controls the size of the search window between candidate disparities; typically α is set at 0.5. We populate $\{\mathbf{u}_i\}$ for $i = 0, 1, 2, \ldots$ until the current search width is less than 1 pixel. Note also that the search window should be bounded by the image size of I_2.

After computing the cost values at each candidate disparity, the new disparity of pixel $\mathbf{p}$ is selected in a Winner-Takes-All (WTA) fashion, as given by

$$\mathbf{u}(x, y) = \arg\min\{C(\mathbf{u}_i)\,|\mathbf{u}_i \in U\}. \tag{5.4}$$

To further increase processing speed and avoid spurious matches over long search distances, the search radii, w_x and w_y, may be limited by the user based on prior knowledge of the images being compared. An adaptive scheme may also be used, where the search range for a given pixel is proportional to the variance of its neighboring pixels. The intuition for such an adaptive search is to widen the search window in earlier iterations when the disparity field is ill-defined, and narrow the search subsequently when the disparity field becomes more accurate. Nonetheless, including an adaptive search width introduces additional processing complexities which may not warrant its usage.

5.3.2.3 Similarity Measure

Since our randomized algorithm is inherently a local technique, we may employ any arbitrary window-based error metric $C(\mathbf{u})$ to compare the two blocks in images I_1 and I_2. As described in Chap. 2, a common similarity measure is the Sum of Absolute Differences (SAD). Given a template block $I_1(\mathbf{q})$ consisting of pixels in the neighborhood of $\mathbf{p} = (x, y)$, the SAD cost function is given by

$$C_{SAD}(\mathbf{u}) = \sum_{\mathbf{q} \in N_p} |I_2(\mathbf{q} + \mathbf{u}) - I_1(\mathbf{q})|, \tag{5.5}$$

where $\mathbf{u}$ is the displacement of the block in I_2. N_p is the $b_x \times b_y$ rectangular neighborhood centered at $\mathbf{p}$, where b_x and b_y are the block width and height respectively. Minimizing the SAD matching cost then gives the disparity of pixel $\mathbf{p}$. While the SAD cost function is relatively robust to outliers, it unfortunately assumes that all pixels within the support window are at the same depth as it implicitly weights all of them equally. When a block straddles a depth discontinuity, the SAD metric tends to give an inaccurate disparity since neighboring pixels at different depths bring about matching ambiguity. As illustrated in Fig. 5-3, the effect is a rounding of corners and edges, which becomes more pronounced as the block size increases.

To overcome this problem, we use the Locally Adaptive Support Weight (LASW) approach proposed by Yoon and Kweon [9]. Instead of using square windows of

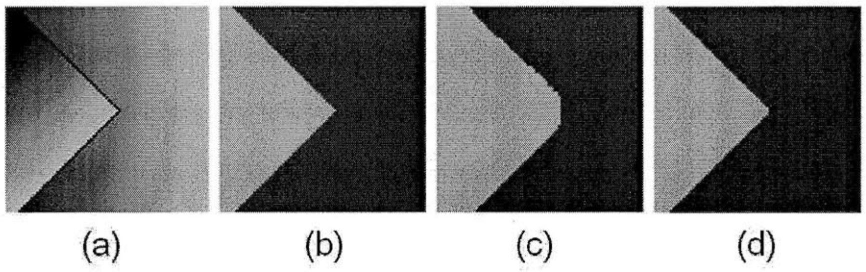


Figure 5-3: Dense disparity map for a synthetic image [9]. (a) Left image. (b) Ground truth. (c) Disparity map using naive SAD. Notice that the corners are rounded as the uniformly weighted support window smooths over depth discontinuities. (d) Disparity map using locally adaptive support window. In this case, corners and edges are preserved.

uniform weights, each pixel in the support window affects the matching cost differently depending on their color similarity and proximity to the pixel under consideration. The idea is that pixels which are close to the reference pixel and of similar color will likely be at the same depth and should thus be weighted more heavily in the cost aggregation, while the remaining pixels should have less effect. Consequently, arbitrarily-shaped depth continuities are better preserved, as illustrated in Fig. 5-3d.

Formally, the local support window is associated with a spatially varying weighting function $w(\mathbf{p}, \mathbf{q})$ that is a combination of two Laplacian kernels, viz.

$$w(\mathbf{p}, \mathbf{q}) = \exp\left(-\frac{\|I(\mathbf{q}) - I(\mathbf{p})\|}{\lambda_c}\right) \exp\left(-\frac{\|\mathbf{q} - \mathbf{p}\|}{\lambda_d}\right). \tag{5.6}$$

The first term of the expression defines the influence of neighboring pixels $\mathbf{q}$ according to their color difference from the center pixel $\mathbf{p}$, where $\|I(\mathbf{q}) - I(\mathbf{p})\|$ represents the Euclidean distance between two pixel colors $I(\mathbf{q})$ and $I(\mathbf{p})$ and the parameter λ_c controls the level of influence. Although Yoon and Kweon first convert colors to CIELab space in order to enhance any perceptual color difference, we measure the Euclidean distance in RGB space to simplify matters. The second term defines the influence of neighboring pixels based on their spatial distance from $\mathbf{p}$, as controlled by the parameter λ_d. The weighting function $w(\mathbf{p}, \mathbf{q})$ is zero outside of the neighborhood N_p.

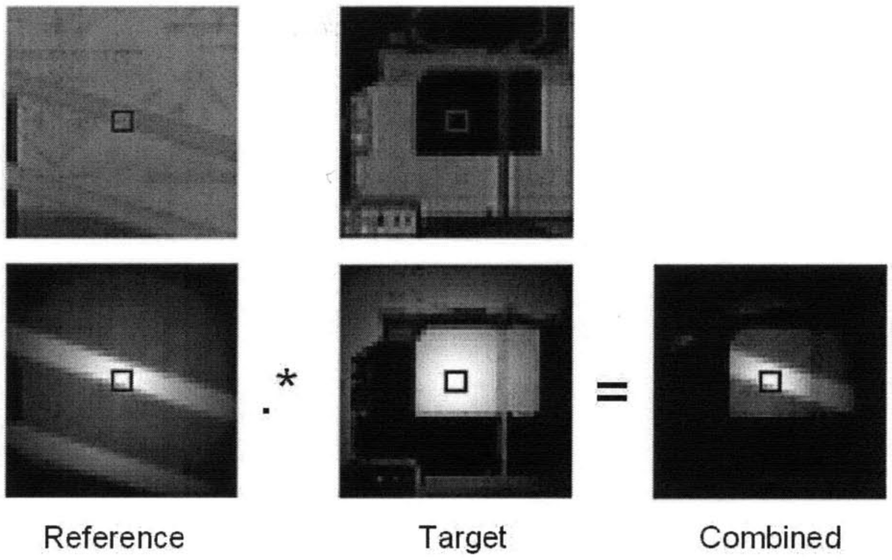


Figure 5-4: Locally adaptive support weight computation [9]. The top row shows the reference and target blocks, and the bottom row shows the respective support weights computed using Eq.5.6. The blue square denotes the reference pixel. Pixels that are similar in color and close to the reference pixel have larger support weights and appear brighter. After computing the support weights for each block, we combine them by pixel-wise multiplication to aggregate support only from similar neighboring pixels.

After applying the local weighting function separately to both the reference block in I_1 and the target block in I_2 (yielding w_1 and w_2 respectively), the cost function for pixel $\mathbf{p}$ becomes a weighted average of SAD costs in the support window:

$$C_{LASW}(\mathbf{u}) = \frac{1}{K}\sum_{\mathbf{q}\in N_p} w_1(\mathbf{p},\mathbf{q})w_2(\mathbf{p}+\mathbf{u},\mathbf{q}+\mathbf{u})\left|I_2(\mathbf{q}+\mathbf{u}) - I_1(\mathbf{q})\right|. \qquad (5.7)$$

The normalization term K is defined as follows:

$$K = \sum_{\mathbf{q}\in N_p} w_1(\mathbf{p},\mathbf{q})w_2(\mathbf{p}+\mathbf{u},\mathbf{q}+\mathbf{u}), \qquad (5.8)$$

and ensures that all the weights sum to unity. Interestingly, the operation in Eq. 5.7 is similar to applying a bilateral filter jointly to the reference and target blocks before comparing them. A graphical depiction of this procedure is shown in Fig. 5-4.

5.3.3 Termination

The randomized algorithm terminates after the disparity field has converged. While various methods may be used to determine convergence and adapt the number of iterations accordingly, we show in Sect. 5.4 that the expected number of iterations to convergence remains constant for large image resolutions. In practice, we use 4-5 iterations. In the hierarchical pyramid scheme, we set the number of iterations at each level l as 2^{l+1}, so that coarser levels (large l) use more iterations than finer and slower levels (small l) .

5.4 Theoretical Analysis

In this section, we present the theoretical justifications for our algorithm. From a probabilistic assessment, we shall show that random initialization often assigns at least one correct disparity, and that the disparity field quickly converges to the exact solution within a few iterations. We will also discuss the potential computational savings of our algorithm relative to other methods.

5.4.1 Random Initialization

Consider two images I_1 and I_2, each of size S pixels. Our task is to find, for every pixel in I_1, the best matching pixel in I_2 within a search window of size W pixels. As described in Sect. 5.3.1, the initialization procedure assigns each pixel in $I_1$with a random disparity $\mathbf{u}(x, y)$. The probability of a given pixel having the right disparity on the initial assignment is extremely small (i.e. $\frac{1}{W}$). However, the probability that all the pixels in I_1 are assigned a wrong disparity also becomes exponentially smaller with increasing image size (i.e. $(1 - \frac{1}{W})^S$). Hence, the chance of at least one pixel in I_1 having the right disparity is relatively high (i.e. $1 - (1 - \frac{1}{W})^S$). Let the ratio between the size of the search window and image size be $\beta = \frac{W}{S}$. Then in the limit of large S, the probability of at least one pixel disparity being correctly assigned is approximately $1 - \exp(-\frac{1}{\beta})$. As β tends to 0 (i.e. the window where the match is guaranteed to be found is much smaller than the image), this probability naturally goes to 1. Conversely, as β tends to 1, this probability goes to $1 - \frac{1}{e}$. Therefore, even in the worst case when the search window is as large as the image, our random initialization still has at least 63% chance of producing a correct match prior to any iterative refinement.

5.4.2 Convergence

Now, we consider the likelihood of convergence at each iteration. Suppose that after initialization, a contiguous region of size R pixels has yet to converge and its pixels were all assigned incorrect disparities. Our analysis is simplified by making three observations. First, we may think of each random search iteration as consisting of a new random disparity assignment within the maximum search window size W, as well as a series of dense local searches around the current disparity $\mathbf{u}(x, y)$. Second, we may also treat a given disparity as correct if it falls within a neighborhood of Q pixels from the actual disparity, since the dense local searches will very likely detect the correct match in one or two iterations. Lastly, we note that once a single pixel within the region of interest has been correctly matched, then all R pixels will also rapidly converge to their correct disparities due to propagation. Therefore, our criteria for convergence is when at least one pixel in the region has been assigned an almost correct disparity within an error margin of Q pixels.

Right after initialization, the probability of convergence prior to iteration is

$$p = 1 - (1 - \frac{Q}{W})^R. \tag{5.9}$$

The probability of convergence at each subsequent iteration is also p, based on our first observation. Since the random search iterations are independent, the probability of the region not converging on iterations $0, 1, 2, \ldots, n-1$ but converging on iteration t is given by $p(1-p)^n$. In other words, the number of iterations for convergence belongs to a geometric distribution, with expected value $E(n)$, viz.

$$E(n) = \frac{1}{p} - 1 = \left[1 - (1 - \frac{Q}{W})^R\right]^{-1} - 1 \tag{5.10}$$

Let the ratio between the size of the region of interest and the image size be $\gamma = \frac{R}{S}$. For large image sizes, Eq. 5.10 may then be rewritten as

$$\begin{aligned} E(n) &= \left[1 - (1 - \frac{Q}{\beta S})^{\gamma S}\right]^{-1} - 1 & (5.11) \\ \lim_{S \to \infty} E(n) &= \left[1 - \exp(-\frac{Q\gamma}{\beta})\right]^{-1} - 1 & (5.12) \end{aligned}$$

Taking the Taylor series expansion of Eq. 5.12 for small γ/β, we obtain

$$\lim_{S \to \infty} E(n) = (\frac{Q\gamma}{\beta})^{-1} - 1 = \frac{W}{QR} - 1. \tag{5.13}$$

Hence, the expected number of iterations for convergence is constant for a small region relative to the search window (i.e. small γ/β) and a large image size S. This result is significant because it allows us to fix the number of iterations n for large images. Typical values for the search window size W, feature size R and neighboring region Q may be 50×50, 3×3 and 10×10 pixels respectively. For this hypothetical case, Eq. 5.13 estimates convergence within about 2 iterations, not including a few additional iterations required for propagation and refinement. In practice, we have found that our algorithm almost fully converges after $n = 4$ iterations, even at sub-Megapixel image resolutions.

5.4.3 Computational Complexity

In this section, we compare the computational complexity of randomized matching to other methods so as to demonstrate its efficiency. Suppose we want to perform stereo matching on two images of size $s \times s$ pixels, using a block size of $b \times b$ pixels and a pre-defined search window of size $w \times w$ pixels. Assuming that initialization carries little computational burden, we only consider the computational load of iterative refinement. First, during propagation, we update each pixel with the best disparity estimate among two of its neighbors and itself. This step involves $3b^2$ multiplications for each pixel. Next, we do a series of m random searches at fractional multiples of the search radius $\frac{w}{2}$ from the current disparity, where $m = \left\lfloor -\frac{\log(w/2)}{\log \alpha} \right\rfloor$. This step involves another mb^2 multiplications for each pixel. We repeat this process for every pixel in the reference image. Thus a single iteration involves $O((m+3)b^2s^2)$ operations. Assuming the algorithm converges (or terminates) after n iterations, in total we require a computational complexity of $O(n(m+3)b^2s^2)$ for randomized stereo matching.

In contrast, conventional stereo matching techniques usually involve a full search in the spatial domain and have a computational complexity of $O(w^2b^2s^2)$. Since the search window size w^2 is typically large ($w^2 \gg n(m+3)$), randomized stereo matching is thus far superior to spatial block-matching in terms of efficiency, barring the computational overhead in random number generation and memory reads. For instance, a randomized algorithm using $n = 4$ iterations and $m = 4$ random samples per iteration can theoretically achieve a 90-fold increase in processing speed over a full search with a typical search parameter of $w = 50$.

Furthermore, it is worth noting that the computational complexity of the randomized algorithm does not depend on the size of the search window (up to a first approximation). This allows the randomized algorithm to efficiently measure large displacements in both horizontal and vertical directions, making it well suited for wide-baseline stereo matching and PIV applications requiring high correlation speeds.

Now, we consider another common method of stereo matching using spectral correlation. Correlation may be implemented efficiently in the Fourier domain by multiplying the 2D Fast Fourier Transform (FFT) of the search window with the complex conjugate of the 2D FFT of the block, before taking the inverse transform. The peak of the resulting correlation table then corresponds to the best disparity estimate. Spectral correlation is known to have a computational complexity of $O(4w^2 \log w + w^2)$ per pixel, or $O(s^2w^2(4\log w + 1))$ for the whole image. Hence, the randomized algorithm is faster than FFT correlation if $n(m+3)b^2 < w^2(4\log w + 1)$. Again, for a large search parameter w and a small block height b, randomized stereo matching results in significant computational savings.

5.5 Results

To demonstrate the effectiveness of the randomized algorithm, we have applied it to several real and synthetic images in MATLAB. For comparison, we have also implemented a conventional full search algorithm using the adaptively weighted SAD cost function. Experimental results for the algorithm's accuracy, convergence and processing time are presented below.

5.5.1 Synthetic Random Dot Images

Figs. 5-5 and 5-6 show the synthetic random dot image pairs generated to evaluate the algorithm, along with the corresponding disparity maps for the reference view. Each image has a size of 200×200 pixels and consists of uniformly distributed random noise on the interval $[0, 1]$. In the first dataset (called *synthetic1*), we shift portions of the reference image by up to 20 pixels left and right to generate the second image. As seen from the checkerboard pattern in the disparity map, small square regions were assigned large disparities in opposite directions to test if the algorithm could effectively handle closely spaced discontinuities. A periodic sinusoidal pattern with a linear offset was also inserted into the image to make it difficult for the randomized algorithm to find the true global minimum. In the second dataset (called *synthetic2*), we simulate a tilted plane imaged by two cameras with parallel optical axes, where the left side of the reference image is closer to the image plane than the right side. In this simple stereo configuration, corresponding points lie on horizontal epipolar lines. Hence, to generate the right image, we simply shift each column of the reference image by the calculated subpixel disparity using Fourier-based translation and sum the shifted columns. Note also that the *synthetic2* dataset is a subset of the multi-image dataset used in Chap. 3.

We ran both the randomized and full-search algorithms on the two synthetic

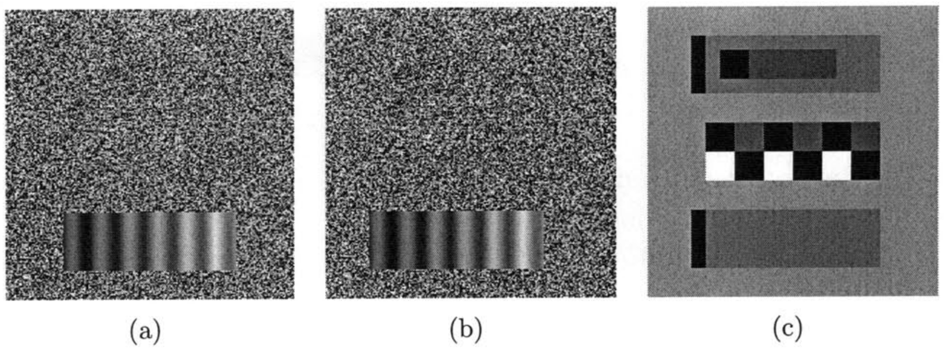


Figure 5-5: Image dataset *synthetic1*, consisting of shifted rectangular blocks. (a) Reference (left) image. (b) Right image. (c) Ground truth disparity map. Brighter regions have larger disparities (positive to the right), and occluded areas are marked in black.

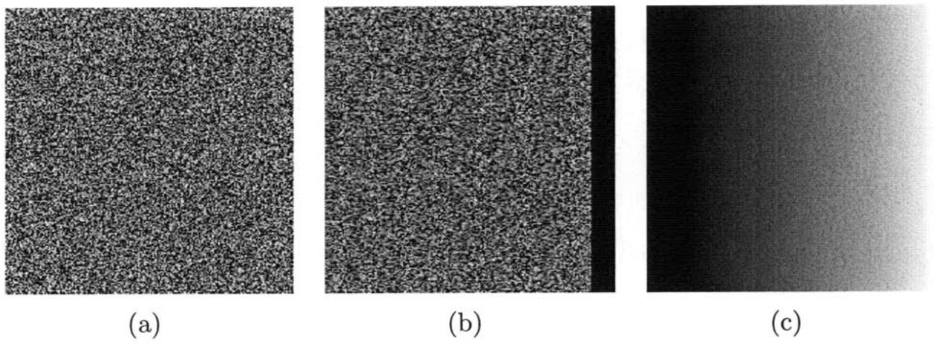


Figure 5-6: Image dataset *synthetic2* of a simulated tilted plane. (a) Reference (left) image. (b) Right image. (c) Ground truth disparity map. All pixels have negative disparity values (i.e. shifted left), but brighter regions have less negative disparities. Occluded areas are marked in black.

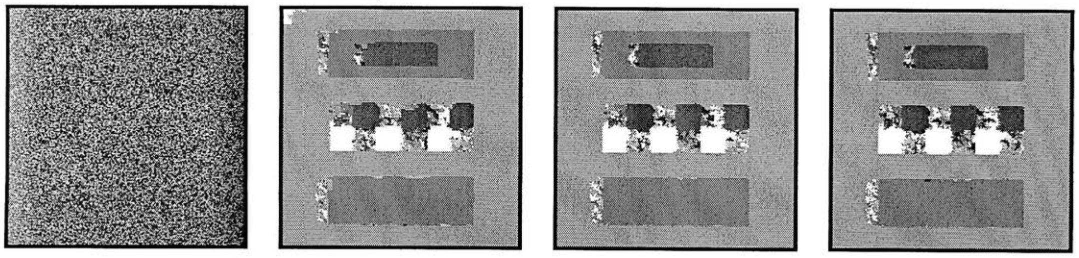

(a) Initial (b) 1 iteration (c) 2 iterations (d) 5 iterations

Figure 5-7: Convergence of randomized algorithm. (a) Initial random disparity field, with brightness indicating disparity values assigned to each pixel (positive to the right). (b) End of first iteration. Notice that the top left portions of homogeneous regions usually have incorrect disparities, until a correct seed causes the rest of the region below and to the right to be filled with the correct values. (c) Two iterations completed. The opposite propagation directions eliminate incorrect disparities in coherent regions. (d) By the end of iteration 5, almost all the pixels have stopped changing values, except for occluded areas which are inherently unstable due to the lack of matching correspondences.

datasets, using a block size of 7×7, search window radius $w_x = 25$, and adaptive support weight parameters $\lambda_c = 0.3$ and $\lambda_d = 3.5$ (radius of block). To speed up processing, we do not search in the y-direction (i.e. $w_y = 0$), noting that the images are already rectified and any disparity vector only has a x-component (i.e. $\mathbf{u}(x, y) = (d_x, 0)$). All disparity maps presented henceforth thus depict only horizontal displacements.

Fig. 5-7 illustrates the convergence of the randomized algorithm over 5 iterations, as applied to the *synthetic1* dataset. At iteration 0, the initial disparity field is fully random. However, even by the end of the first and second iterations, good disparity matches have been propagated throughout the image. After 5 iterations, almost all the pixels have stopped changing disparity values, attesting to the stability and rapid convergence of the randomized algorithm.

In Fig. 5-8, we compare the disparity maps obtained by randomized search (after 4 iterations) and full search with the ground truth for the *synthetic1* dataset. Due to the usage of locally adaptive support windows, the disparity maps for both the randomized and full search algorithms are very accurate for unoccluded regions. Depth discontinuities are preserved well and the corners and edges of the shifted rectangular portions remain sharply defined in the disparity map. Nonetheless, some errors do occur on the edges of discontinuous regions, partly due to the highly textured nature of the random-dot images. Paradoxically in highly textured regions, the amount of aggregated support for each pixel is reduced as immediately adjacent pixels may be of dissimilar colors. Hence, the locally adaptive similarity measure may produce false matches due to insufficient signal. Experimentally, we have also observed this erroneous effect when applying adaptively weighted support windows to the *synthetic2* dataset. For this reason, we instead use the regular SAD cost function for the *synthetic2* dataset, which produces clean disparity maps correct to pixel-wise resolution, as shown in Fig. 5-9.

More importantly, we observe that the randomized algorithm has converged to the exact solution obtained by full search after a small number of iterations. In fact, for the *synthetic2* dataset, the disparity maps between randomized search and full search are exactly identical. For the *synthetic1* dataset, homogeneous regions in the checkerboard disparity pattern were able to arrive at the correct disparity values, in spite of the lower probability of a correct seed occuring within the smaller regions. Moreover, the randomized algorithm could detect the correct global minimum for the sinusoidal pattern without falling into local minima along its search path, thus demonstrating the robustness of random search.

Having seen an illustration of the randomized algorithm's convergence, we now attempt to quantify its convergence rate using the percentage of correct disparities relative to the ground truth as an approximate metric. Intuitively, a low percentage across multiple iterations indicates poor convergence, while a value close to that of the full search indicates nearly complete convergence. Fig. 5-10 shows the percentage of correct disparities over several iterations for both the *synthetic1* and *synthetic2* datasets. A correct disparity is defined as having an absolute error of less than 1, and occluded regions are discounted in the calculation of statistics. Moreover, the percentages are averaged over 3 trials to minimize statistical anomalies. As expected, the randomized algorithm is least accurate after the first iteration. However, it is worth noting that at the beginning of iteration 1, only 6.5 percent of all disparities

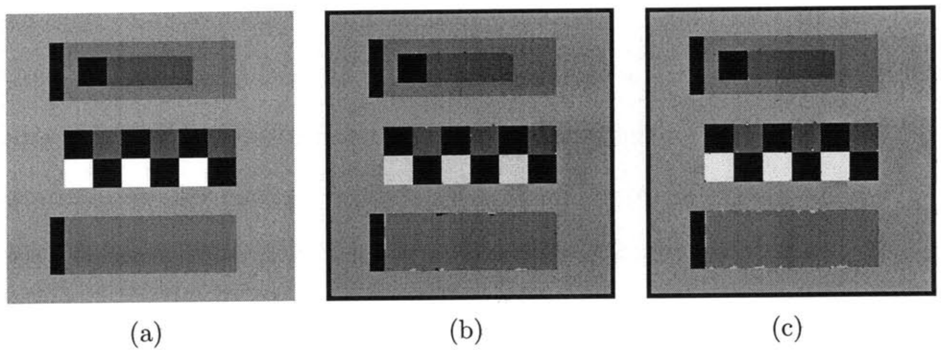


Figure 5-8: Results for *synthetic1* dataset. (a) Ground truth. (b) Disparity map produced by randomized algorithm at the end of 4 iterations. (c) Disparity map obtained from full search.

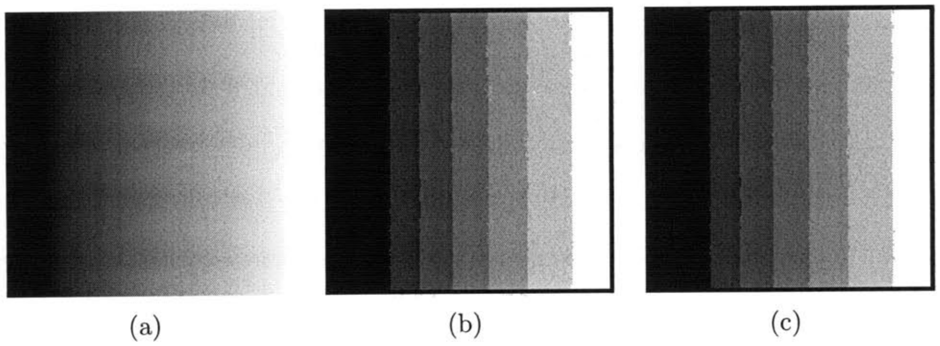


Figure 5-9: Results for *synthetic2* dataset. (a) Ground truth. (b) Disparity map produced by randomized algorithm at the end of 4 iterations. (c) Disparity map obtained from full search.

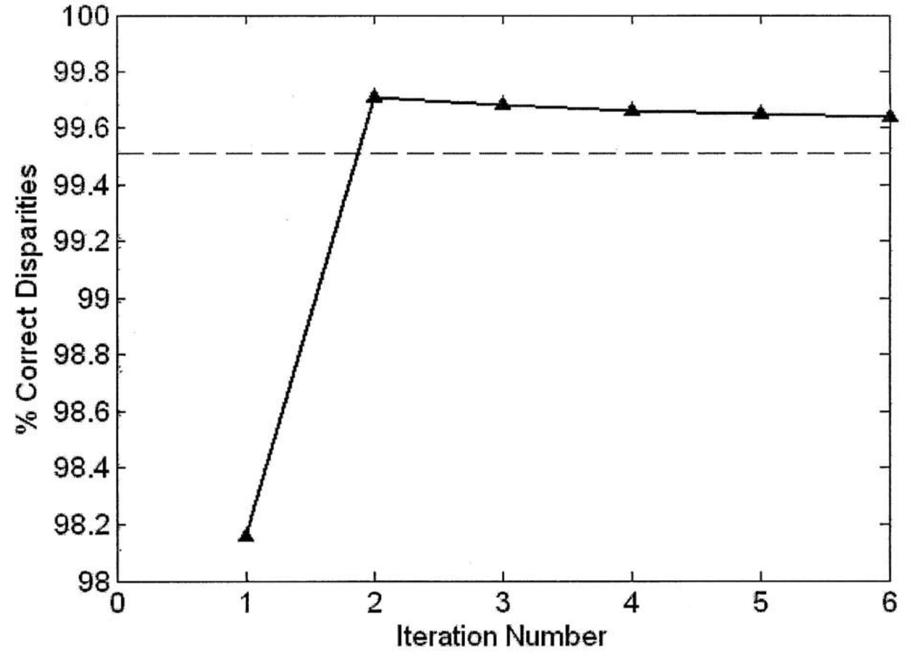
were correctly assigned on average. Thus it is impressive that the algorithm could quickly achieve a high accuracy of at least 98 percent at such an early stage. Theoretically, the randomized algorithm attains the same accuracy as the full search in the limit of having randomly searched all possible disparity values. For the *synthetic2* dataset, the algorithm produces correct disparities for the entire image from the end of the second iteration onwards. Interestingly however, for the *synthetic1* dataset, the randomized algorithm yields a higher percentage of correct disparities than the full search in the second iteration before gradually declining to the same accuracy in subsequent iterations. This is due to the randomized algorithm landing in local minima that coincidentally correspond to the correct disparity, and not because of any additional accuracy afforded by random search. Regardless of the search method used, the final accuracy is limited by the discriminative power of the chosen similarity measure.

Based on our insights from Figs. 5-7 and 5-10, it is clear that the randomized algorithm converges quickly for synthetic datasets within a small number of iterations. Hence, we have empirically verified our theoretical predictions for convergence, as described in Sect. 5.4.2.

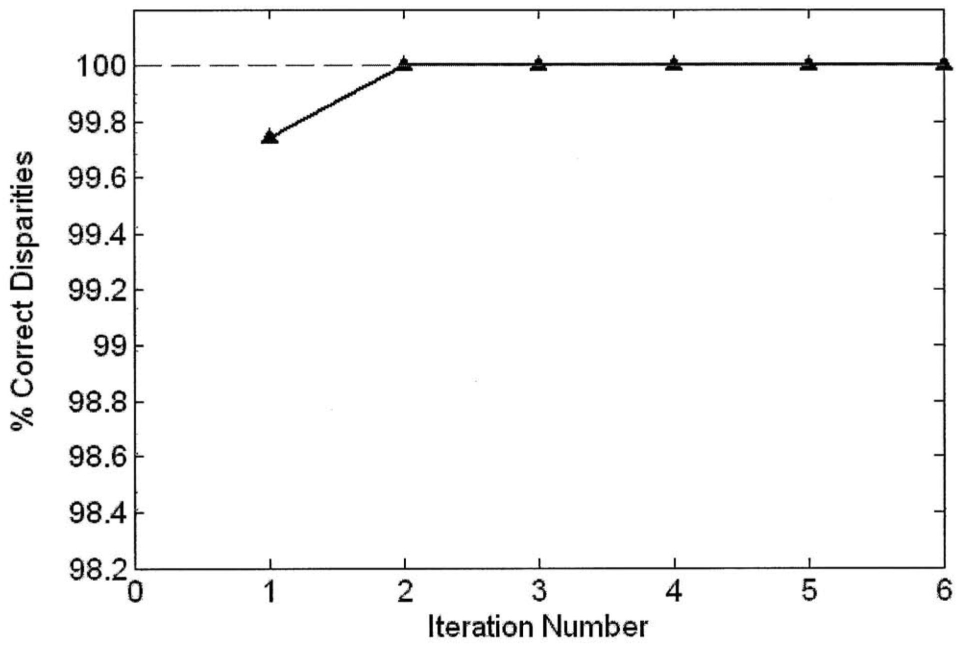
5.5.2 Real Images

We further evaluated the performance of the randomized algorithm using the Middlebury stereo datasets, which are commonly used to compare different stereo techniques [1]. We selected two datasets, named *Tsukuba* and *Sawtooth*, as shown in Figs. 5-11 and 5-12 together with their ground truth disparity maps. In running our algorithms on the real image datasets, we used a block size of 25×25, search radii $w_x = 25$ and and $w_y = 0$, and adaptive support weight parameters $\lambda_c = 0.3$ and $\lambda_d = 12.5$ (radius of block).

Fig. 5-13 illustrates the convergence of the randomized algorithm over 5 iterations for the *Tsukuba* dataset. As before, the algorithm quickly transitions from an initial random disparity field to an accurate and stable version. Such rapid convergence makes the algorithm useful for processing real images.



(a)



(b)

Figure 5-10: Illustration of convergence rate. (a) Percentage of correct disparities over 6 iterations for (a) the *synthetic1* dataset and (b) the *synthetic2* dataset, represented in blue solid lines. The green dotted lines denote the accuracy of the full search method, which the randomized algorithm should converge to in the limit. The percentages of correct disparities are averaged over three trials and do not account for occluded regions.

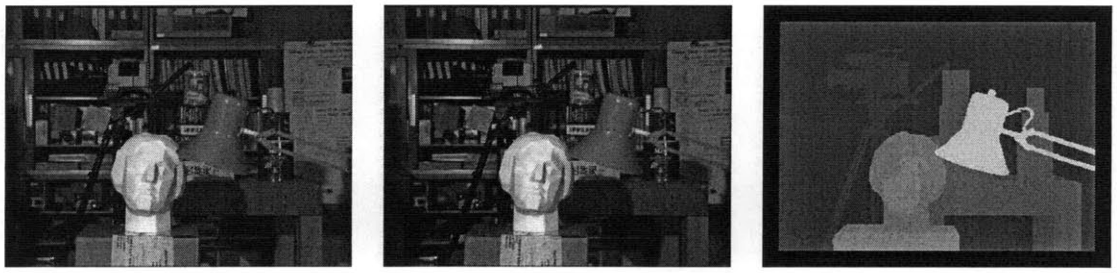

(a) (b) (c)

Figure 5-11: Real image dataset *Tsukuba.* (a) Reference (left) image. (b) Right image. (c) Ground truth disparity map, with bright regions representing positive disparities (to the right).

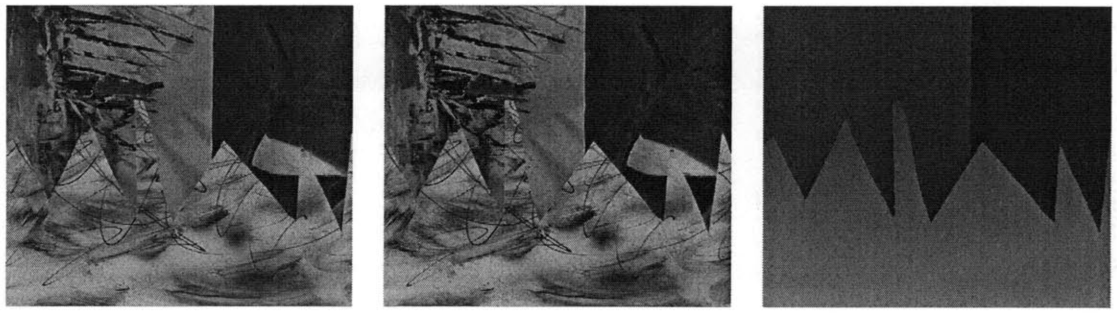
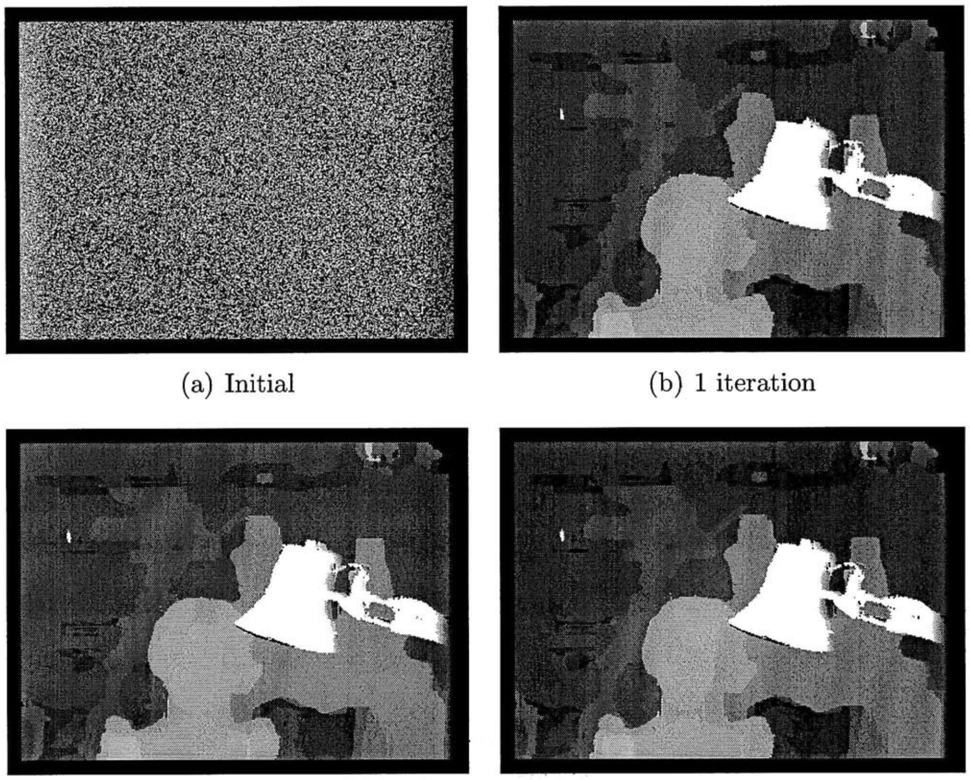

(a) (b) (c)

Figure 5-12: Real image dataset *Sawtooth.* (a) Reference (left) image. (b) Right image. (c) Ground truth disparity map. Note that the creators of this dataset use brighter regions to denote larger disparities to the left.


(a) Initial (b) 1 iteration

(c) 2 iterations (d) 5 iterations

Figure 5-13: Convergence of randomized algorithm for *tsukuba* dataset. (a) Initial random disparity field. (b) End of first iteration. Note that majority of disparity values have already been found accurately. (c) End of second iteration. (d) After 5 iterations, the disparity field has converged.

Figs. 5-14 and 5-15 show matching results from the randomized algorithm (after 4 iterations) and full search for the *Tsukuba* and *Sawtooth* datasets respectively. The use of locally adaptive support windows works especially well for images of real scenes as there is usually sufficient aggregated support for each pixel compared to the synthetic datasets. In both datasets, the locally adaptive similarity measure produces accurate results at the depth discontinuities as well as in coherent regions. In particular, for the *Tsukuba* dataset, disparity values for narrow objects, such as the lamp stem and the tripod, are found correctly, and for the *Sawtooth* dataset, the sharp corners and edges in the foreground are preserved and not rounded in spite of the large block size. However, errors are still present especially in textureless regions (e.g. the gray back-wall of the *Tsukuba* dataset), along occluding boundaries and at the image borders where points may not be visible to both cameras.

Furthermore, we note from Figs. 5-14 and 5-15 that the disparity maps from the randomized algorithm have converged to the exact solution produced by full search. Again, we quantify this convergence by measuring the percentage of correct disparities relative to ground truth during each iteration. As before, a correct disparity has an absolute error of less than 1, and the percentages for each iteration are averaged over 3 trials. However, for real datasets, we include occluded pixels in computing the disparity errors as occluded regions are not marked on the ground truth disparity maps, thus resulting in slightly lower accuracies. Nonetheless, in these comparisons, the rate of convergence is of greater interest than the actual performance of the chosen similarity measure. We should expect the randomized algorithm to eventually converge to the same accuracy as that of the full search, given enough iterations.

Fig. 5-16 shows the percentage of correct disparities over several iterations for the *Tsukuba* dataset. At first glance, it might appear that the randomized algorithm has yielded the same disparity field as full search by the end of the first iteration, judging by the similar percentages of correct disparities. However, a closer inspection of the error map between the full search solution and the randomized algorithm output after 1 iteration reveals substantial errors. While the percentage of correct disparities remains relatively constant over several iterations, it is not immediately clear how the

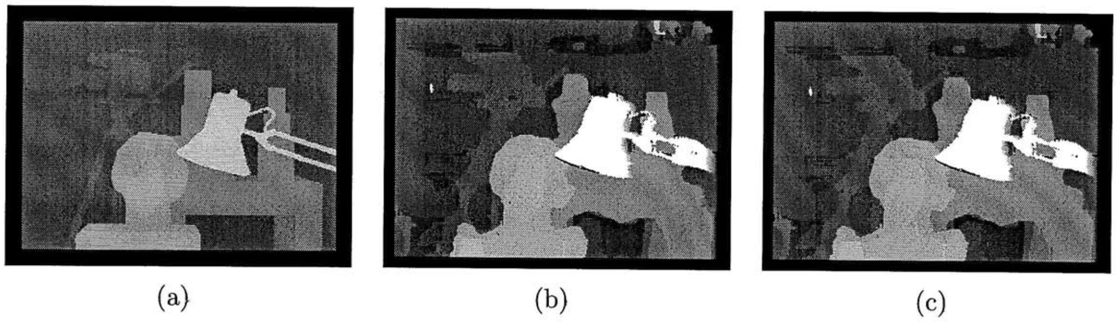


Figure 5-14: Results for *Tsukuba* dataset. (a) Ground truth. (b) Disparity map produced by randomized algorithm at the end of 4 iterations. (c) Disparity map obtained from full search. Note that in (b) and (c), negative values and values above a threshold of 20 have been removed from the disparity maps.

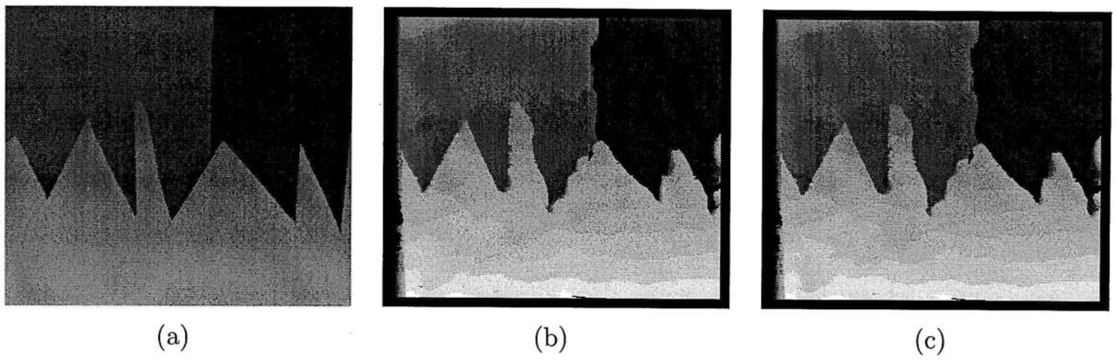


Figure 5-15: Results for *Sawtooth* dataset. (a) Ground truth. (b) Disparity map produced by randomized algorithm at the end of 4 iterations. (c) Disparity map obtained from full search. Note that in (b) and (c), negative values and values above a threshold of 20 have been removed from the disparity maps.

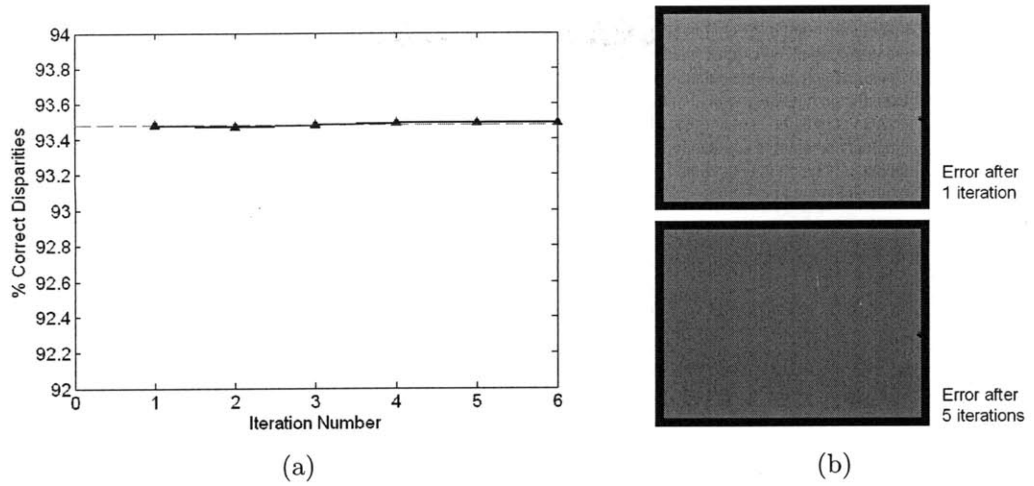


Figure 5-16: Illustration of convergence rate for the *Tsukuba* dataset. (a) Percentage of correct disparities over 6 iterations. The blue line represents the randomized algorithm, while the green line denotes the accuracy of the exact full search solution. (b) Error maps between the full search and randomized outputs after 1 iteration and 5 iterations. While the percentage of correct disparities seems fairly constant over 6 iterations, the error map after 1 iteration reveals incomplete convergence. After 5 iterations, however, most errors have been eliminated and the randomized algorithm almost fully converges.

distribution of errors changes throughout. By the fifth iteration, however, the error map is almost empty, indicating that the randomized algorithm has indeed converged to the exact solution. Fig. 5-17 presents similar findings for the *Sawtooth* dataset. Though the randomized algorithm coincidentally finds a better solution than full search in earlier iterations, it eventually converges to the same accuracy subsequently. These results for the two real image datasets reinforce our earlier statement that the randomized algorithm converges completely within 4-5 iterations.

5.5.3 Processing Time

Lastly, we empirically compare the processing speeds of the randomized algorithm and the full search approach. Although Sect. 5.4.3 predicts significant computational savings by using the randomized algorithm, our MATLAB implementation actually takes substantially longer than the full search algorithm to complete 4 iterations. This difference is likely due to the inherent inefficiencies of MATLAB when reading

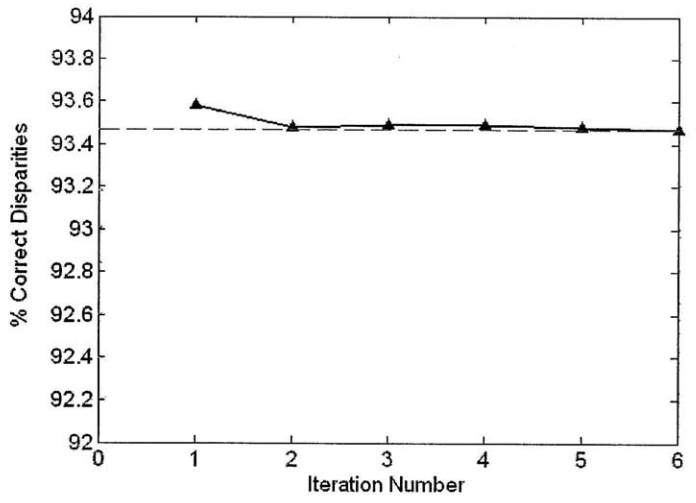


Figure 5-17: Illustration of convergence rate for the *Sawtooth* dataset, showing the percentage of correct disparities over 6 iterations. The blue line represents the randomized algorithm, while the green line denotes the accuracy of the exact full search solution.

large arrays element-by-element. Conceivably, an implementation in C is expected to be much faster and will better demonstrate the predicted computational savings from the randomized algorithm.

Fig. 5-18 shows the processing time taken to run the randomized and full search algorithms as a function of block width b. We measure all times on a Intel Core 2 Duo 2GHz processor using the *synthetic1* dataset and a constant search width of 50. As expected, the processing time for both methods increases with larger block sizes, since both their computational complexities scale quadratically with b. However, we also observe that as the block size increases, the randomized algorithm has a smaller fractional increase in processing time compared to the full search method. To better visualize this difference, we divide our data by a reference time for a 3×3 block to nondimensionalize the processing times. At a block size of 19×19, the full search method takes nearly 2.8 times the reference time, while the randomized algorithm only requires about 1.8 times the reference time. Ideally, scaling the data in this manner should produce similar-looking curves because the dependence on image size, search width, and number of iterations is eliminated. As such, the reason that the full search has a steeper dimensionless time curve than the randomized algorithm is probably due to the greater number of memory reads that it requires, thereby incurring a larger overhead in MATLAB which was not factored in the theoretical predictions of computational complexity.

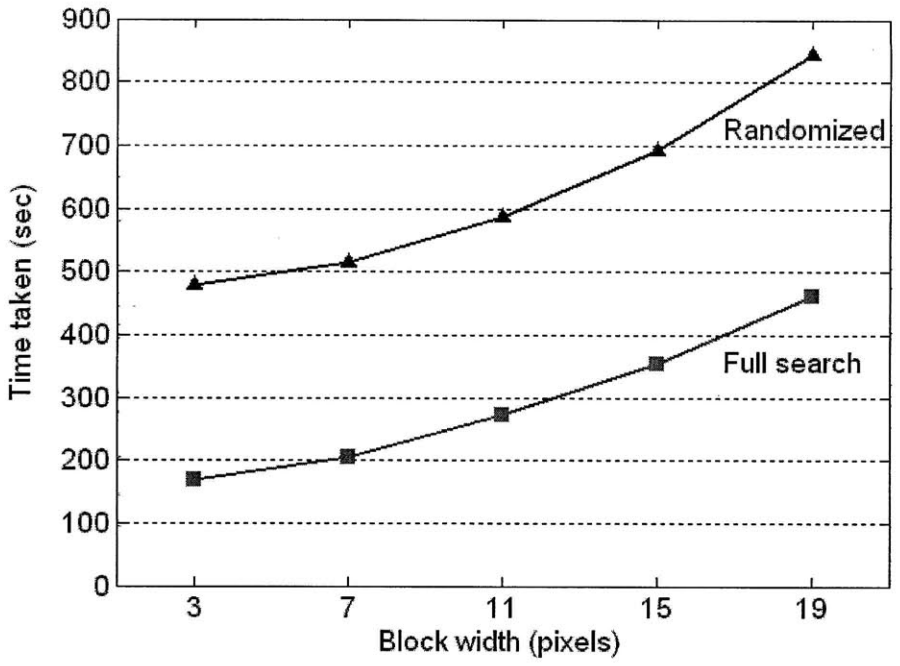
Next, Fig. 5-19 shows the processing time taken to run both algorithms as a function of search window width w, again using the *synthetic1* dataset and a constant block size of 7×7. As the search width increases from 50 to 100 pixels, the processing time for the full search rises significantly. In contrast, the time taken for the randomized search remains relatively constant across a large range of search widths, with the exception of a kink at 70 pixels due to an additional random match being made within the search window (recall that the number of random searches per iteration is $m = \left\lfloor -\frac{\log(w/2)}{\log \alpha} \right\rfloor$). This finding agrees with our earlier prediction that the computational complexity of the randomized algorithm does not depend on the size of the search window. For large search widths, the randomized algorithm is thus much more efficient than a full search.

5.6 Discussion

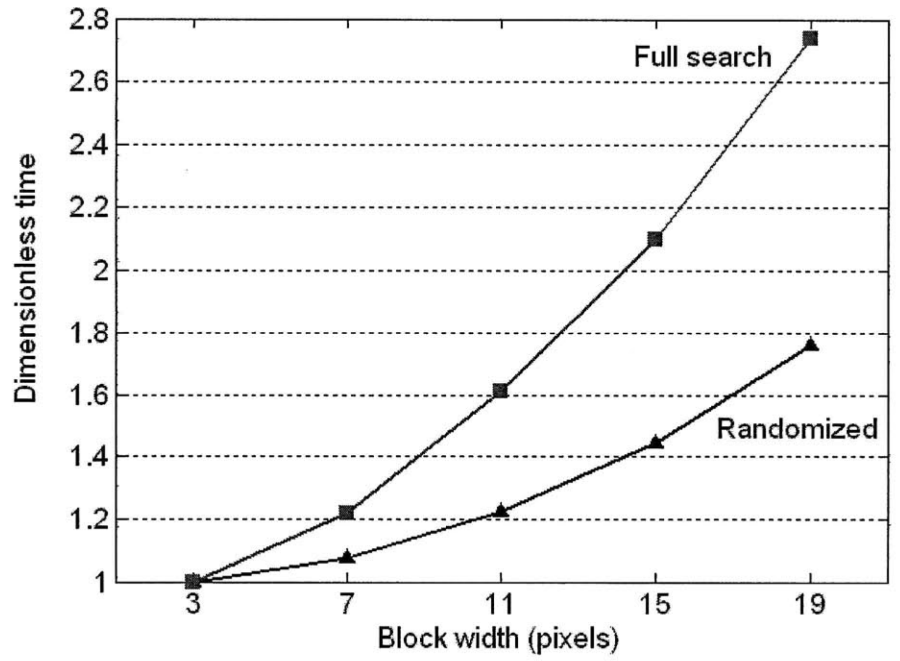
5.6.1 Strengths and Limitations

In summary, the above results demonstrate that the randomized algorithm produces an accurate disparity map when applied to both synthetic random-dot and real image datasets. The algorithm uses coherency in the image data to focus on likely match locations and minimize redundant block-wise comparisons. In so doing, it achieves a very low computational complexity that is independent of search width. Furthermore, unlike other fast block-matching methods which assume a unimodal cost function surface and thus fail in the presence of periodicity, our iterative random search procedure allows the algorithm to robustly identify true global minima with high probability. Hence, in just a small number of iterations, the algorithm converges to the exact solution obtained from a full search.

A key advantage of the randomized algorithm is its versatility in supporting any



(a)



(b)

Figure 5-18: (a) Plot of actual processing time taken against block width b, for 4 iterations of the randomized algorithm (in blue) and the full search method (in red). (b) Plot of dimensionless time as a function of block width, scaled against the reference time for a 3×3 block. Notice that although the actual time for the randomized algorithm is higher than that for the conventional full search, it incurs a smaller fractional increase in processing time than the full search.

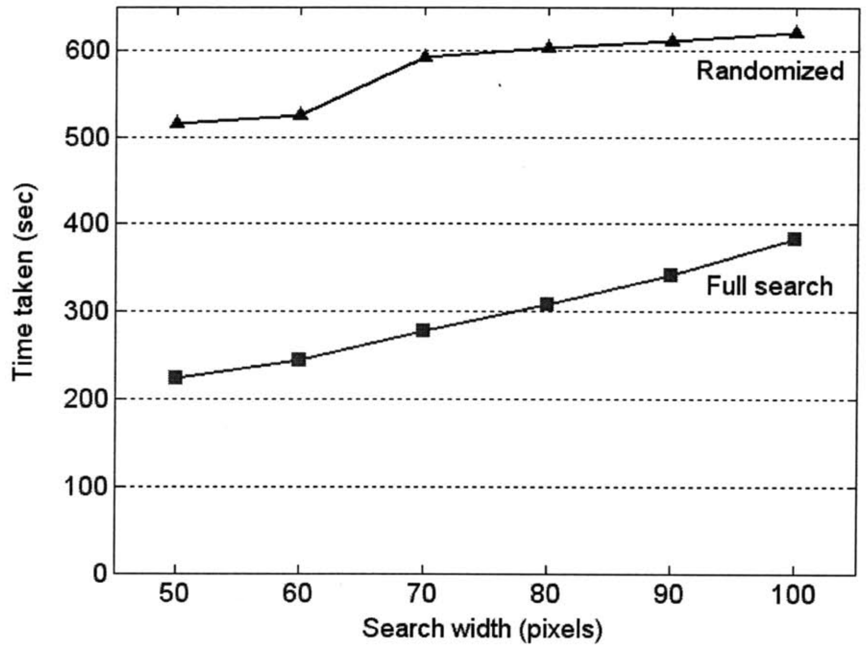


Figure 5-19: Plot of processing time taken against search window width w, for the 4 iterations of the randomized algorithm (in blue) and the full search method (in red). Notice that the time taken for full search increases with larger search widths, while the randomized algorithm is relatively independent of the search parameter.

similarity measure. In our implementation, we have chosen to use locally adaptive support weights to aggregate neighboring pixels according to their color similarity and spatial proximity. Our experiments show that this method preserves arbitrarily shaped depth discontinuities well. Using locally adaptive windows in conjunction with randomized correlation is also especially advantageous because of the smaller computational load compared to other methods. A full search using locally adaptive windows is extremely computationally intensive as it requires a new Laplacian filter kernel to be calculated for every candidate disparity. FFT-based correlation is also difficult to implement because the locally adaptive windowing function is not constant for different displacements.

As the randomized algorithm is inherently a local block-matching technique, other similarity measures may also be used, such as the standard Normalized Cross-Correlation (NCC) function that compensates well for different exposure levels. Moreover, for multiple-image datasets, the multiple-baseline approach described in Chap. 3 may be adopted. The random search procedure can be easily modified so that for each candidate disparity with respect to a reference stereo pair, the corresponding disparities for the other stereo pairs are also computed. All the matching costs may then be summed or multiplied together to reinforce reliable matches while eliminating faulty information.

Nonetheless, there are also limitations to the randomized algorithm. The main disadvantage is that the overall shape of the cost function surface is not available due to the sparse, random sampling of candidate disparities. As such, while the randomized algorithm is able to find the integer displacement of a given block, incremental refinements cannot be made immediately. Furthermore, techniques which rely on the local variation of matching score, such as adaptive windows [7] or shiftable windows [8], cannot be applied. A simple solution would be to evaluate several disparity values around the best value found so far and fit a Gaussian or parabolic curve in order to find the subpixel displacement via interpolation. Alternatively, one may use the Lucas-Kanade optical flow method to perform gradient descent on the matching cost function around the best disparity estimate [10]. In both cases, subpixel refinement should be made after the pixel-resolution disparity field from the randomized algorithm has converged, so as to speed up processing.

Second, although the randomized algorithm makes use of the assumption that the disparity map is mostly continuous to propagate good matching values, it does not explicity enforce smoothness in the output. Hence, in the presence of noise, the localized randomized algorithm may not only produce spurious matches, but may also take longer to converge to the exact solution. Noisy image data distort the cost function surface and potentially introduce many more local minima basins. Propagating noisy matches may also cause the disparity field in earlier iterations to be unstable as the random search is targeted around incorrect regions. A similar problem occurs for pathological cases where periodicity in the image leads to multiple global minima (e.g. an image of a picket fence). Since the local similarity measure cannot disambiguate the potential matches and as there is no term penalizing discontinuous disparities, the resulting disparity field is likely to be noisy.

Finally, the randomized algorithm does not include a neighborhood term that regularizes optimizations over occluded regions. In these regions, there is no corresponding point in the right image, and thus the local similarity measure returns a meaningless disparity value. Ideally, we should label and penalize these occluded regions so that incorrect disparities are not propagated to regions with accurate matches. If possible, we can do even better by filling in the missing values for these occluded regions so as to form a complete 3D representation of the scene. In the next section, we present some ideas to incorporate occlusion reasoning into the existing randomized algorithm framework.

5.6.2 Extensions

To conclude our analysis of the randomized algorithm, we suggest ways to improve its effectiveness as a stereo matching technique and further extend its usage to other applications.

5.6.2.1 Occlusion Reasoning

As mentioned previously, the ability to account for occlusions is central to accurate stereo matching. Such occluded pixels are often associated with a high matching cost and can be detected during postprocessing using a threshold, above which they are deemed as occluded. However, due to the randomized nature of sampling, not all non-occluded pixels will necessarily have low matching costs at a given iteration. Care should thus be taken to ensure that pixels which have not yet found their true global minimum are not accidentally marked as occluded. Instead of applying a naive threshold, we propose an adaptive threshold that decreases with the current iteration number, such that earlier iterations utilize a higher threshold while later iterations are more stringent in filtering out occlusions.

Alternatively, occluded areas may be detected using a consistency check between the left-to-right and right-to-left disparity maps. A given pixel in the left image has a correct match if its corresponding pixel in the right image has the same disparity value (albeit in the opposite direction). However, if the two disparity values do not fall within a small error margin, then the point is either mismatched or occluded in the right image. At first glance, performing this cross-checking step appears to double the processing load, since two disparity maps are computed instead of one during each iteration. We note however that the calculation of the left-to-right and right-to-left disparity maps can be done in parallel. Moreover, we can speed up convergence by initializing the disparity field for the right-to-left comparison using the left-to-right disparity field from the previous iteration. The resulting right-to-left disparity field may in turn be used to refine the left-to-right disparity field during the next iteration, where accurate right-to-left disparity values are also propagated on top of the candidate left-to-right disparities from the two adjacent pixel neighbors.

While labelling occluded pixels on the fly prevents them from adversely affecting other pixels via propagation, the tradeoff is that the operation is irreversible. A better approach may then be to associate a confidence level to the disparity estimate of each pixel. To this end, we propose using the ratio of the two lowest matching costs out of all the candidate disparities as a rough measure of confidence, i.e. how much better is the winning (lowest cost) disparity compared to the next best disparity estimate? Intuitively, high-confidence matches have a well-defined global cost minimum, while low-confidence matches might be trapped in noisy or wide local minima basins. At the end of all iterations, the confidence values for each iteration may then be aggregated. If a pixel has a confidence value higher than a certain threshold, then its disparity is taken as correct; otherwise it is classified as occluded.

5.6.2.2 Efficiency

The efficiency of the randomized algorithm may also be improved in several ways. First, when comparing blocks during the propagation and random search phases, we can stop calculating the matching cost for the current candidate disparity if its partial sum exceeds that of the best disparity estimate so far. This allows us to move on to the next candidate disparity more quickly. Second, we note that the computation of matching costs for adjacent pixels often involves the same pixel comparisons within the overlapping regions of their blocks. Hence, instead of computing a new sum for each adjacent pixel, we may retrieve the redundant pixel comparisons from memory and sum only the missing terms.

Finally, the computational complexity of the algorithm may be greatly lowered by compressing the images beforehand. One such compression scheme, known as compressed image correlation, has been commonly used in Particle Image Velocimetry (PIV) [14]. In PIV, a fluid is seeded with tracer particles, illuminated with a laser sheet, and imaged using a high-speed camera. The fluid flow is then visualized by tracking the particle displacements using cross-correlation. Prior to correlation, however, each frame may be compressed by keeping pixels with high intensity gradients while removing smoother regions with low information content. The resulting image is a sparse array containing only strong features, thereby reducing the number of pixel comparisons and memory reads required for block-matching. Compressed image correlation has also been tested on real images with promising results [15]. Presumably, a combination of compressed image correlation and randomized matching should give rise to even higher processing speeds.

Mathematically, a full search using compressed image correlation has a computational intensity of $O(\epsilon^2 w^2 b^2 s^2)$, where ϵ is the compression ratio, w is the search window width, b is the block width, and s is the image width. Since the compression ratio ϵ is typically about $\frac{1}{30}$, significant computational savings over a regular full search are already possible. Suppose instead that randomized matching is implemented in place of full search. Referring to Sect. 5.4.3, the computational complexity of the randomized algorithm without compression is $O(n(m+3)b^2 s^2)$, where n is the number of iterations and m is the number of random searches per iteration. If compressed image correlation is also employed, the complexity drops to $O(\epsilon^2 n(m+3)b^2 s^2)$, leading to a dramatic improvement in processing speed by a factor of over 10000. Randomized compressed image correlation thus has the potential to accurately process stereo image pairs at tremendously high speeds.

5.6.2.3 Future Directions

Due to its low computational intensity, the randomized algorithm may be applied in a myriad of applications requiring fast block-matching. For instance, Barnes *et al.* have explored the use of fast randomized matching in the context of creative image manipulation, with the aim of creating a real-time user interface for interactive image editing [27]. Particle Image Velocimetry is another domain where the randomized algorithm might be directly applicable. Typically, PIV processing involves performing block-matching on a sequence of images using a cross-correlation similarity measure. Since the tracer particles are free to move in any direction across different frames, PIV correlation uses a 2D search window, in contrast to a 1D search strip for stereo matching when the epipolar constraint is invoked. Unfortunately, the two-dimensionality of the search window creates a substantial computational burden, especially when performing a full search in the spatial domain. Conversely, the computational complexity of the randomized algorithm does not depend on the size of the search window, and is thus well-suited to speed up PIV processing. Furthermore, the displacement field from a previous frame may be used to initialize the next frame by assuming that a given block in the current frame retains the same displacement in the corresponding block of the subsequent frame. In so doing, the number of iterations required for convergence may be reduced.

Future work may be done to implement the randomized algorithm in C/C++, so as to realize the true performance enhancements. The randomized algorithm could also be compared to other fast block-matching algorithms or nearest neighbor search methods, and evaluated in terms of its accuracy and speed.

Chapter 6

Conclusions

This thesis has described several methods for quickly and robustly performing stereo matching on a set of images. The overall objective is to accurately recover a dense disparity map indicating the correspondences of each point. In this final chapter, we will summarize the key takeaways and discuss potential directions for future work.

6.1 Stereo Matching Basics

The theoretical underpinnings of stereo matching were first discussed in Chap. 2. Due to the specific geometry of stereo matching, the search for correspondences may be constrained along corresponding epipolar lines instead of the entire image. A variety of techniques already exist to perform stereo matching. Some techniques seek to match reliable features so as to generate a sparse scene reconstruction, while the more common approach involves finding dense correspondences between images. Typically, these dense correspondence algorithms may be further categorized as local and global methods. Local approaches determine point correspondences by choosing the candidate which minimizes a matching cost function. The selection of a suitable support window for aggregating matching costs ensures reliable estimation. Global methods, on the other hand, seek to minimize an energy function over the entire image using a Markov Random Field (MRF) model, and explicitly enforce smoothness assumptions about the disparity map. Although the problem is computationally challenging, efficient strategies such as graph cuts and belief propagation have recently been developed to produce some of the best results yet. Nonetheless, the processing times of global methods are usually far longer than those of local methods, and thus local methods are still frequently used in many real-time stereo applications.

6.2 Multiple-Baseline Stereo

In Chap. 3, we have presented an early approach to multi-view stereo, called multi-baseline stereo. Multi-baseline stereo uses multiple stereo pairs with various baselines to obtain precise, unambiguous depth estimates. Matching is performed by computing either the sum of SSD-in-inverse-distance function or the product of EC-in-inverse-distance table, whereby individual pairwise similarity measures have been aggregated into a single function. The use of inverse distance as the common variable across all stereo pairs allows reliable evidence to reinforce each other and faulty information to be eliminated, thus reducing ambiguity and increasing the accuracy of depth estimates. This technique is also particularly useful because it can accommodate any cost function. Experimental results for the algorithm have shown its effectiveness in recovering quantitative depth information from both synthetic and real image datasets.

Once multiple viewpoints can be combined in an efficient and robust manner, other post-processing techniques such as optical aberration correction, super-resolution, and accurate subpixel interpolation become possible. These extensions are worth exploring when dealing with multi-view stereo systems and light fields. The multi-baseline approach thus lays the foundation for reconstructing dense, high-accuracy 3D models of objects and scenes.

6.3 Multi-Dimensional Correlation

In Chap. 4, we have proposed a stereo matching algorithm which takes into account foreshortening effects so as to achieve robust stereo matching. Unlike existing methods which first determine depth before estimating surface orientation (thus relying on the accuracy of the initial depth estimate), our method simultaneously recovers the depth and surface orientation of an imaged point by performing correlation in multiple dimensions. The imaged surface is approximated as a plane and locally deformed according to the surface orientation and camera geometry. The peak of the resulting multi-dimensional correlation table then gives the most probable depth and tilt of the point. The 2D surface orientation parameters may also be efficiently found by using at least two 1D searches along corresponding epipolar lines and then taking the cross-product of the tangential vectors to obtain the surface normal.

Three methods for performing multi-dimensional correlation were proposed and evaluated. The first method is an exhaustive search for the parameters which maximize the correlation between the deformed image windows. While computationally intensive, this basic method allows us to easily visualize the behavior of the correlation tables under different conditions. An intermediate window size which contained sufficient intensity variation without excessive depth variation was found to work well. The second method seeks to increase computational efficiency by noting redundant terms in the correlation summation. Each pixel in the left image window is compared to the right image window and the point-wise correlation value is added to multiple table entries at a time, thereby avoiding duplicate comparisons. While this method leads to computational savings in theory, our experiments unfortunately highlight numerous limitations with the method. These include the lack of normalization, sensitivity to the histogram binning resolution, the lack of reliable entries in the resulting correlation table, and errors incurred after fitting a surface over the sparse table. Further research is required in order to realize the predicted performance enhancements of the efficient method. Finally, the third method uses a classical Levenberg-Marquardt minimization algorithm to find the peak value of the multi-dimensional correlation table. Experiments on synthetic datasets demonstrate the algorithm's ability to accurately extract both depth and surface orientation even from stereo images with limited foreshortening.

Future work could explore coarse-to-fine hierarchical approaches to improve the efficiency of multi-dimensional stereo matching. Using at least two stereo pairs, it is also possible to account of depth discontinuities near the pixel of interest. If the peak correlation values for the two stereo pairs differ, it is likely that the point is close to a depth edge. Based on this reasoning, the support windows may be better chosen to accurately recover smooth surfaces without crossing over discontinuities.

6.4 Randomized Stereo Matching

Finally, Chap. 5 presented a new iterative, randomized algorithm that could significantly speed up stereo matching. The key insights behind the algorithm are that random guesses for correspondences can often produce some good disparity matches, and that these matches may be propagated to surrounding pixels based on the coherence of the disparity map. The algorithm starts from an initial random disparity estimate, and then iteratively refines the disparity of each pixel by considering its local neighborhood as well as randomly searching around the best existing disparity. Our theoretical analysis and experimental results demonstrate that the algorithm rapidly converges within a small number of iterations. To increase the accuracy of disparity estimates, we employed locally adaptive support weights based on color segmentation to successfully preserve arbitrarily-shaped depth discontinuities. Furthermore, since only a fraction of all possible block comparisons are made, the randomized algorithm has a far lower computational complexity than a conventional full search. Experimental results on synthetic and real images confirm its relative efficiency under different block sizes and search widths.

As extensions, we have proposed a hierarchical scheme as well as subpixel interpolation to refine disparity estimates. Several methods of occlusion reasoning were also suggested, such as implementing an adaptive threshold, left-right cross-checking, and measuring the confidence of candidate matches. The possibility of using compressed image correlation in conjunction with the randomized algorithm was also discussed, potentially leading to tremendous computational savings. Lastly, other promising applications of such a fast and versatile technique were envisioned.

Bibliography

[1] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-frame Stereo Correspondence Algorithms," *Int'l Journal of Computer Vision*, vol. 47, no. 1, pp.7-42, 2002.

[2] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *Proc. Conf. Computer Vision and Pattern Recognition*, pages 519– 528, 2006.

[3] M. Goesele, N. Snavely, B. Curless, H. Hoppe and S. M. Seitz, "Multi-View Stereo for Community Photo Collections," *IEEE Int'l Conf. on Computer Vision*, pp. 1-8, 2007.

[4] C. Loop, Z. Zhang, "Computing Rectifying Homographies for Stereo Vision," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, pp. 125-131, 1999.

[5] D. G. Lowe, "Distinctive Image Features from Scale-invariant Keypoints," *Int'l. Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[6] M. Brown and D. G. Lowe, "Unsupervised 3D Object Recognition and Reconstruction in Unordered Datasets," *Int'l Conf. on 3-D Digital Imaging and Modeling (3DIM 2005)*, pp. 56-63, 2005.

[7] T. Kanade and M. Okutomi, "A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 9, pp. 920-932, 1994.

[8] S. B. Kang, R. Szeliski and J. Chai, "Handling Occlusions in Dense Multi-View Stereo," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 103-110, 2001.

[9] K. J. Yoon and I. S. Kweon, "Adaptive Support-Weight Approach for Correspondence Search," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 650-656, 2006.

[10] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proc. DARPA Imaging Understanding Workshop*, pp. 121-130, 1981.

[11] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen and C. Rother, "A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 30, no. 6, pp. 1068-1080, 2008.

[12] M. Okutomi and T. Kanade, "A Multiple-Baseline Stereo," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 4, pp. 353-363, 1993.

[13] G. Roth, D. P. Hart and J. Katz, "Feasibility of Using the L64720 Video Motion Estimation Processor (MEP) to Increase Efficiency of Velocity Map Generation for Particle Image Velocimetry (PIV)," *ASME/JSME Fluids Engineering and Laser Anemometry Conference*, 1995.

[14] D. P. Hart, "High-Speed PIV Analysis Using Compressed Image Correlation," *Journal of Fluids Engineering*, vol. 120, pp. 463-470, 1998.

[15] S. S. Tan and D. P. Hart, "A Fast and Robust Feature-based 3D Algorithm Using Compressed Image Correlation," *Pattern Recognition Letters*, vol. 26, pp. 1620-1631, 2005.

[16] D. G. Jones and J. Malik, "Determining Three-dimensional Shape from Orientation and Spatial Frequency Disparities," *Proc. European Conference on Computer Vision*, pp. 661-669, 1992.

[17] J. Robert and M. Hebert, "Deriving Orientation Cues from Stereo Images," *Proc. European Conference on Computer Vision*, pp. 377-388, 1994.

[18] F. Devernay and O. Faugeras, "Computing Differential Properties of 3-D Shapes from Stereoscopic Images Without 3-D Models," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 208-213, 1994.

[19] H. Hattori and A. Makim "Stereo Matching with Direct Surface Orientation Recovery," *Proc. British Machine Vision Conference*, pp. 356–366, 1998.

[20] N. Xu and N. Ahuja, "A Three-view Matching Algorithm Considering Foreshortening Effects," *Proc. Computer Vision, Pattern Recognition and Image Processing*, pp. 635-638, 2003.

[21] D. Marr and T. Poggio, "Cooperative Computation of Stereo Disparity," *Science*, vol. 194, pp. 209-236, 1976.

[22] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated Interframe Coding for Video Conferencing," *Proc. NTC 81*, pp. C9.6.1-9.6.5, 1981.

[23] R. Li, B. Zeng, and M. L. Liou, "A New Three-step Search Algorithm for Block Motion Estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438-442, 1994.

[24] L. M. Po and W. C. Ma, "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation," *IEEE Trans. Circuits And Systems For Video Technology*, vol. 6, no. 3, pp. 313-317, 1996.

[25] S. Zhu and K. K. Ma, "A New Diamond Search Algorithm for Fast Block-matching Motion Estimation," *IEEE Trans. Image Processing*, vol. 9, no. 2, pp. 287-290, 2000.

[26] Y. Nie and K. K. Ma, "Adaptive Rood Pattern Search for Fast Block-matching Motion Estimation," *IEEE Trans. Image Processing*, vol. 11, no. 12, pp. 1442-1448, 2002.

[27] C. Barnes, E. Shechtman, A. Finkelstein and D. B. Goldman, "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), 2009.

[28] R. Szeliski, *Computer Vision: Algorithms and Applications.* Microsoft Research, 2010. [Online Draft]. Available at: http://szeliski.org/Book/