A PHASE BASED DENSE STEREO ALGORITHM IMPLEMENTED IN CUDA

A Thesis

by

BRENT DAVID MACOMBER

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2011

Major Subject: Aerospace Engineering

A PHASE BASED DENSE STEREO ALGORITHM IMPLEMENTED IN CUDA

A Thesis

by

BRENT DAVID MACOMBER

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	John Junkins
Committee Members,	John Hurtado
	Jim Ji
	James Turner
Head of Department,	Dimitris Lagoudas

May 2011

Major Subject: Aerospace Engineering

ABSTRACT

A Phase Based Dense Stereo Algorithm

Implemented in CUDA. (May 2011)

Brent Macomber, B.A., University of California at Berkeley

Chair of Advisory Committee: Dr. John Junkins

Stereo imaging is routinely used in Simultaneous Localization and Mapping (SLAM) systems for the navigation and control of autonomous spacecraft proximity operations, advanced robotics, and robotic mapping and surveying applications. A key step (and generally the most computationally expensive step) in the generation of high fidelity geometric environment models from image data is the solution of the dense stereo correspondence problem. A novel method for solving the stereo correspondence problem to sub-pixel accuracy in the Fourier frequency domain by exploiting the Convolution Theorem is developed. The method is tailored to challenging aerospace applications by incorporation of correction factors for common error sources. Error-checking metrics verify correspondence matches to ensure high quality depth reconstructions are generated. The effect of geometric foreshortening caused by the baseline displacement of the cameras is modeled and corrected, drastically improving correspondence matching on highly off-normal surfaces. A metric for quantifying the strength of correspondence matches is developed and implemented to recognize and reject weak correspondences, and a separate cross-check verification provides a final defense against erroneous matches. The core components of this phase based dense stereo algorithm are implemented and optimized in the Compute Unified Device Architecture (CUDA) parallel computation environment onboard an NVIDIA Graphics Processing Unit (GPU). Accurate dense stereo correspondence matching is performed on stereo image pairs at a rate of nearly 10Hz.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. John Junkins, for his guidance and his overwhelming amount of knowledge. He has served to inspire and motivate me to succeed and to always put in the extra effort, even when the problem at hand has been daunting. A huge amount of credit goes to Dr. Manoranjan Majji, who has served as a mentor and guide for me during my transition to the field of engineering. His patient explanations have helped me in all aspects of student/research life. Thank you to my committee members, Drs. John Hurtado, Jim Ji, and James Turner, for the constant motivation to finish this project, and the guidance to see it to completion. Thank you to everyone at LASR Lab: Drs. Jeremy Davis and Xiaoli Bai, and James Doebbler, Brien Flewelling, Drew Woodbury, Clark Moody, Kurt Cavalieri, and Andrei Kolomenski for their collaboration and moral support. Thank you to LASR Lab's collaborators, Drs. Jan-Michael Frahm and Pierre Georel, and Mr. Tim Johnson at University of North Carolina- Chapel Hill for hosting me for two weeks, and patiently training me in CUDA development and optimization. And finally, thank you to you, the reader who is taking the time to read this!

TABLE OF CONTENTS

CHAPTER Page

I INTRODUCTION 1
- A. Motivation 1
- B. Stereo Background 4
  - 1. Feature Based Stereo 5
  - 2. Dense Stereo 5
  - 3. Epipolar Geometry 7
  - 4. Challenges to Stereo Correspondence Matching 9

II PHASE BASED DENSE STEREO ALGORITHM 14
- A. Theory 14
  - 1. 1-Dimensional Example 14
  - 2. 2-Dimensional Example 19
- B. An Initial 2-Dimensional Implementation in MATLAB . . 24
- C. 1-Dimensional Implementation in C 29
- D. 1-Dimensional Implementation in MATLAB 36

III ALGORITHMIC IMPROVEMENTS, QUALITY METRICS, AND 3D RECONSTRUCTIONS 37
- A. Algorithmic Improvements 37
  - 1. Averaging Spatial Correlation Functions 37
  - 2. Sub-Pixel Disparity Calculation 38
  - 3. Geometric Foreshortening Correction 41
- B. Quality Control Metrics 50
  - 1. Left/Right Disparity Cross Checking 50
  - 2. Correlation Peak Strength 53
  - 3. Comparison of Quality Control Methods 57
- C. 3D Reconstructions from Dense Stereo 58

IV CUDA IMPLEMENTATION 61
- A. CUDA Background 61
  - 1. Program Flow 62
  - 2. Memory Structure and Kernel Optimization 64
  - 3. CUFFT Library 66

CHAPTER Page

B. Existing Parallel Dense Stereo Algorithms 66
C. 1-Dimensional Implementation in CUDA 67
1. Initialization 70
2. Padded Arrays 71
3. Fourier Transforms 71
4. Array Multiplication 72
5. Inverse FFT 72
6. Locate Maxima 72
7. Termination 73

V COMPARISON OF RESULTS 75

A. CUDA Implementation Results 75
B. Comparison to Ground Truth Disparity Maps 78
C. Comparison to Planar Lab Truth 82

VI CONCLUSION 89

REFERENCES 91

VITA 98

LIST OF FIGURES

FIGURE Page

1 Image pair of same scene taken from locations of arbitrary relative rotation and translation, illustrating correspondence problem. 4

2 Rectified stereo pair, illustrating epipolar geometry. 7

3 Left and right stereo image pair, illustrating challenges to stereo correspondence matching algorithms. 10

4 1-Dimensional phase correspondence example, constructed left and right signal. 15

5 1-Dimensional phase correspondence example, spatial correlation function. 18

6 1-Dimensional phase correspondence example, bounded spatial correlation function. 19

7 2-Dimensional phase correspondence example. 20

8 2-Dimensional phase correspondence example, padded images. 21

9 2-Dimensional phase correspondence example, plots of spatial correlation function. 23

10 Stereo image pair, rectified left and right image. 25

11 2-Dimensional implementation, object of interest and search space images. 26

12 2-Dimensional implementation, row plot from typical spatial correlation function. 27

13 2-Dimensional implementation, matched regions in left and right image. 27

14 2-Dimensional implementation, resulting raw disparity map. 29

FIGURE Page

15 2-Dimensional implementation, resulting median filtered disparity map. 30

16 1-Dimensional implementation, padded search space and object of interest arrays. 32

17 1-Dimensional implementation, typical spatial correlation function response for a single column. 33

18 1-Dimensional implementation, typical right referenced disparity map. 34

19 Averaging spatial correlation functions, effect on one dimensional spatial correlation function. 38

20 Disparity map of nearly fronto-parallel plane, discrete and continuous disparity versions. 42

21 Stereo image pair, exemplifying foreshortening effects. 43

22 Foreshortening effect geometry, stereo camera and off-normal rotated plane. 43

23 Left and right referenced disparity maps from Mayan blanket scene, uncorrected for geometric foreshortening effect. 47

24 Left and right referenced disparity maps from Mayan blanket scene, corrected for geometric foreshortening effect. 48

25 Typical row plot from left and right referenced disparity map of Mayan blanket scene, uncorrected versus foreshortening effect corrected. 49

26 Left and right referenced raw disparity maps from ditch scene. 52

27 Left and right referenced psuedo disparity maps, generated by transforming right and left raw disparity maps. 53

28 Left and right referenced verified disparity maps. 54

29 Ditch scene, right image of stereo pair and right referenced correlation peak height map. 56

30 Ditch scene, right referenced raw disparity map and right referenced filtered disparity map. 57

FIGURE Page

31 Three dimensional reconstruction of ditch scene from phase based dense stereo algorithm. 59

32 Flow chart of CUDA implementation of phase based dense stereo algorithm. 68

33 Resulting disparity map, CUDA versus MATLAB. 76

34 Column plot comparison of CUDA disparity map and MATLAB disparity map for nearly fronto-parallel planar scene. 77

35 Three dimensional reconstruction generated by CUDA phase based dense stereo algorithm, from nearly fronto-parallel Mayan blanket plane scene. 78

36 Left and right stereo image pair, rocks scene 79

37 Right referenced disparity maps from rocks scene, ground truth and phase based dense stereo algorithm output 80

38 Disparity error map for rocks scene compared to ground truth. 80

39 OpenCV open source block matching and graph cut dense stereo algorithms error maps for Middlebury rocks scene. 82

40 Statistics for plane fit to SURF points as laboratory truth. 84

41 Typical plane fit to SURF points as laboratory truth. 84

42 Mean absolute value of residual errors for phase based dense stereo algorithm and SAD algorithm for planar scenes. 86

43 Variance of residual errors for phase based dense stereo algorithm and SAD algorithm for planar scenes. 87

44 OpenCV block matching algorithm 3D reconstruction of off-normal planar scene at extreme angle. 87

CHAPTER I

INTRODUCTION

A. Motivation

As humanity continues to push the frontier of exploration into space, our species is developing more intelligent and observant robotic vehicles which can perceive and interact with their environment. More nations are becoming involved in space activities and each new launch adds to the collection of objects in orbit around our planet and beyond. Additionally, many missions are being planned to small solar system bodies for the purpose of surveying and possibly colonizing in the future. Near Earth Asteroids zip harmlessly by our planet, but one day the asteroid that will not zip by, but instead is on a collision course, will be discovered and will need to be surveyed and diverted or destroyed. For all of these reasons it is essential that the spacecraft that are launched from this point onward have the ability to accurately sense objects, both natural and man-made, in their immediate environment.

As the number of man made objects in orbit grows, the probability of collisions between these objects also grows, especially in densely populated orbits. Every collision causes more space debris, which in turn increases the likelihood of another collision. Kessler and Cour-Palais predict that the number of objects in orbital debris belts caused by collisions could increase exponentially if actions are not taken to avert the problem [1]. Many proposed solutions to the orbital debris problem involve sending custodial spacecraft to the vicinity of large disabled spacecraft and causing them to de-orbit by directly interacting with them ([2, 3] for instance). The custodial spacecraft sent to interact with disabled satellites obviously need to be keenly aware

The journal model is *IEEE Transactions on Automatic Control.*

of their surroundings and know their position relative to the disabled object to avoid causing an orbital collision of the type they are trying to prevent. They need to be able to see and understand their environment, and from the information gathered by their environmental sensors, be able to successfully maneuver in the vicinity the disabled spacecraft.

Robotic interaction and intervention with space objects is not limited to man made satellites. The day will someday come when mankind must face the challenge of asteroid mitigation. There are hundreds of asteroids which are classified by the International Astronomical Union as Potentially Hazardous, including the most well known, asteroid (99942) Apophis [4, 5]. Many asteroid mitigation strategies involve close proximity operations of spacecraft to the asteroid [6]. Missions of this type require a spacecraft to be able to “see” the asteroid and know its own position relative to it with a high accuracy. Indeed, the fate of the world could depend on it. A notable strategy for deflecting an asteroid is by exploiting the Yarkovsky effect [7] whereby a slight orbital perturbation of the asteroid could be achieved by modifying its surface albedo (i.e. painting it). In order to determine the proper location in which to apply the paint job, an accurate three dimensional model of the asteroid with its native texture is required. This information can only be acquired in sufficient detail by sensors onboard spacecraft in the near proximity of the asteroid.

The need for high fidelity sensing systems for robotic exploration is not restricted only to spacecraft operations. Planetary rovers need to be very aware of their surroundings at all times to avoid collisions with obstacles, or other hazards involved in navigation with a long time delay for instructions. There is a precedent of using vision based sensors to navigate planetary exploration vehicles. Goldberg, Maimone and Matthies [8] describe the stereo vision based rover navigation algorithm which has been successfully guiding the twin Mars rovers since 2004. Many others have also studied the problem of autonomous navigation and mapping using vision based sensors (see for instance [9, 10] and the references therein).

For all the applications above, vision based sensors are the mission critical piece of hardware that allows the spacecraft/rover to sense its surroundings and successfully interact with them. Every aerospace application for autonomous exploration has its own unique set of challenges for the vision based sensing system to overcome. For this reason, there are several classes of vision based sensors. LIDAR/LADAR systems are an example of an active sensing system. Their mode of operation is to emit a pulse of light which flies to the nearest surface along the scan direction and is reflected back to the sensor. The sensor measures the round trip time of flight of the light, and thus the distance to the object. Stereo vision is an example of a passive sensing system (usually, although some modified active stereo systems make use of a structured light projection system to paint surface texture onto the scene). In a stereo vision system, two cameras image the same scene from a slightly displaced relative location, or the same camera is displaced with time, and the three dimensionality of the scene is reconstructed by triangulation.

Stereo vision is an excellent choice as a vision sensor for any of the mission types described above. Because it is a nominally passive system (i.e. with no projected light or moving parts), there are few failure modes for the hardware involved in a stereo system. This is beneficial because equipment for space exploration takes a beating. Between acceleration, shock, and vibration during launch and planetary surface landing, and radiation exposure in the space environment, high precision sensing equipment for the space environment must also be designed to be robust. Additionally, stereo vision uses a low amount of power when compared to a LADAR/LIDAR system which must project beams of light out to the scene and recover the reflection, so a stereo system may be run in a space environment on a limited power budget such as that provided by solar panels at a large distance from the sun. Additionally, stereo vision has been proven to be able to perform high quality three dimensional reconstructions of scenes with relatively inexpensive equipment [11].

B. Stereo Background

Stereo vision is the process of solving the correspondence problem for points located in the overlapping region of a stereo pair of images, essentially locating the same point in two images which are taken from different locations. Fig. 1 shows a pair of images of the same space object taken from two different locations. The difference of the camera's position between images may be modeled as an arbitrary rotation matrix R and a translation vector $\boldsymbol{t}$.

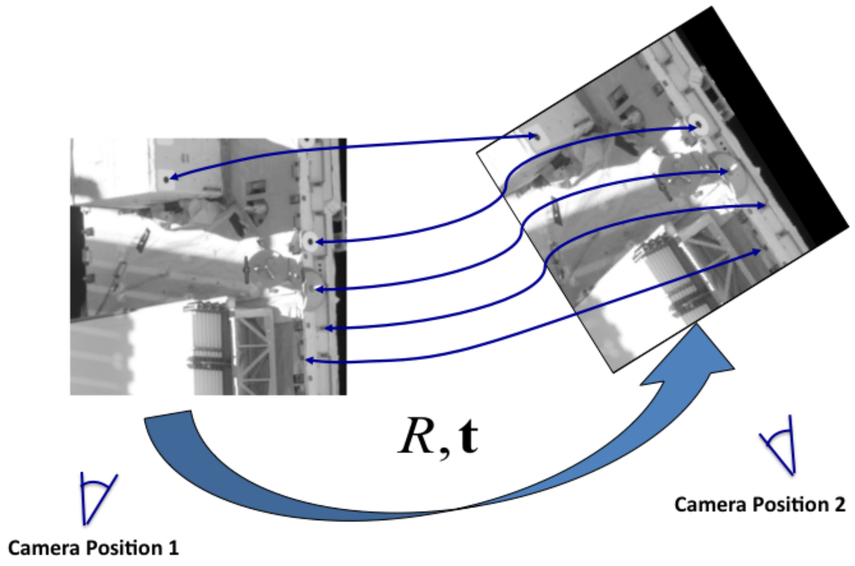


Fig. 1. Image pair of same scene taken from locations of arbitrary relative rotation and translation, illustrating correspondence problem.

1. Feature Based Stereo

Humans are excellent at pattern matching and can easily distinguish the same landmarks in pairs of images, such as the few that are connected by the arrows in Fig. 1. It is more difficult (but not impossible) to write an algorithm such that a computer can match landmark features in images that are rotated and translated by an arbitrary $(R, \boldsymbol{t})$. Since digital images have become available, people have been working on the problem of feature detection and feature based stereo correspondence (see for instance [12, 13]). This can be summarized as the process of identifying landmark locations in images, and by comparing the displaced locations of these landmarks in multiple images, calculating the depth to the object in the image whose surface contains the landmark feature. A common problem with feature matching is that as the images involved degrade in quality, lighting conditions change between images, or the viewpoint from which the images are acquired is significantly different, the feature match quality degrades. A new generation of feature matching algorithms has enabled more robust matches to be made despite all the challenges stated above [14, 15].

2. Dense Stereo

Dense stereo is the process of solving the stereo correspondence problem for every pixel in the overlapping region of a pair of stereo images, thus being able to generate a high confidence three dimensional reconstruction of the entire scene with no interpolation necessary. In contrast to sparse (feature based) stereo matching of a few highly distinct features (e.g. Fig. 1), humans are not capable of solving the dense stereo correspondence problem without the aid of computer based algorithms. There are, however, hundreds of dense stereo algorithms available for solving the correspondence problem. Scharstein and Szeliski provide an excellent survey of the methods available for dense stereo, as well as a comparison of the relative performance of the methods based upon several quality metrics [16]. They also provide datasets and metrics for evaluating dense stereo algorithms and comparing their relative performance [17].

An industry standard method for computing dense stereo correspondence matching is called Sum of Squared Differences (SSD). In this method a small mask of one image is compared across the search space for that mask in the other image. The best match is deemed to be the one that minimizes a cost function which is generated by comparing the square of the differences of the pixel intensity values [18]. A close cousin of this method is called Sum of Absolute Differences (SAD), whereby the cost function to be minimized is generated by the absolute value of the pixel intensity differences (see [19], for example).

In many dense stereo techniques, a small mask of one image is convolved with another image to compare how similar the two appear. Another method of convolution for performing dense stereo image correspondence matches is to exploit the Convolution Theorem, which may be stated as

$$f(x) * g(x) \leftrightarrow F(s)G(s) \tag{1.1}$$

where $*$ denotes a convolution operation, $f(x)$ and $g(x)$ are spatial domain functions, and $F(s)$ and $G(s)$ are the Fourier transformed frequency domain representations of the spatial functions f and g. Thus, the theorem says that a convolution in the spatial domain may be accomplished by an elementwise multiplication in the frequency domain, and vice versa [20]. Correspondence matching in the Fourier frequency domain is not a new concept (see for instance [21, 22, 23, 24] and the references therein). Other techniques use different operators such as Gabor filters or wavelet decompositions to transform to the frequency domain (for example [25, 26]) before perfoming correspondence matching.

3. Epipolar Geometry

Above it is stated that the relative position between cameras in a stereo pair of images may be described by a rotation matrix R and a translation vector $\boldsymbol{t}$. The line that connects the two camera centers, as shown in Fig. 2, is called the line of centers of the stereo pair (also called the baseline). For the case of fixed baseline stereo, the translation vector $\boldsymbol{t}$ is in a fixed direction and has a fixed length B, so the stereo image pair is generated by two cameras rigidly fixed side by side or one above the other. A plane containing the line of centers of the cameras and a point in object space is called an epipolar plane (there are an infinite number of them).

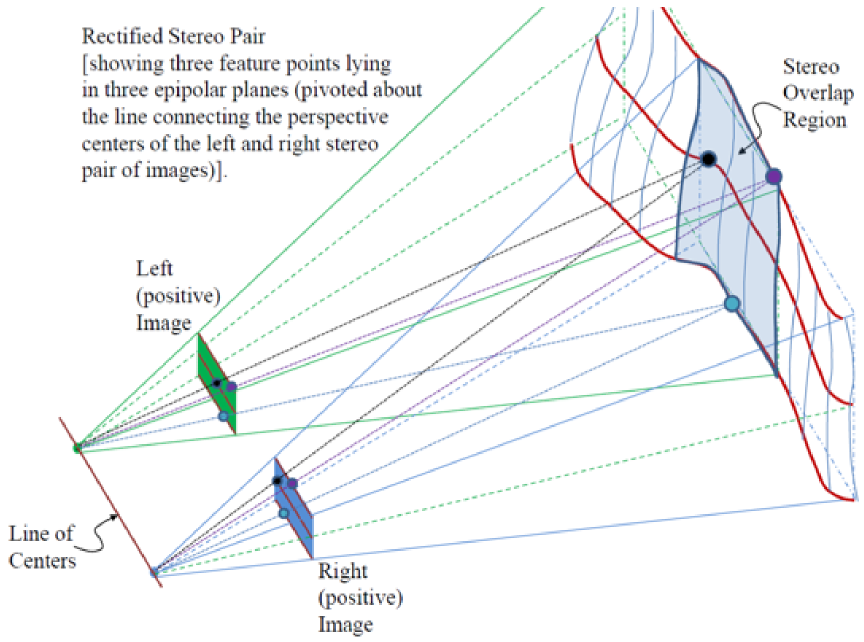

Rectified Stereo Pair
[showing three feature points lying in three epipolar planes (pivoted about the line connecting the perspective centers of the left and right stereo pair of images)].
Stereo Overlap Region
Left (positive) Image
Line of Centers
Right (positive) Image

Fig. 2. Rectified stereo pair, illustrating epipolar geometry.

A transformation process called stereo rectification may be applied to the images of a stereo pair that warps them such that they appear as they would if the focal planes of the cameras generating the images were co-planar. Mathematically this is equivalent to stating that the relative rotation matrix R becomes an identity matrix. The amount to warp the images, and in which direction, is a consequence of the relative R and $\boldsymbol{t}$ of the stereo imaging system, and also of the intrinsic parameters of the cameras in the stereo imager (focal length, and principle point offset). In the process of rectification, non-linear lens effects caused by the cameras are also removed (these effects must be quantified by calibration prior to applying the rectification process). The result of applying the rectification process on a stereo pair of images is that the intersection of an arbitrary epipolar plane with the image planes of the two cameras is horizontal, and is at the same vertical location on both image planes (for the case of a horizontal baseline camera system). The implication is that image features appear in the same row in both images, and the stereo correspondence problem is reduced to a one dimensional search problem. Thus with a bit of clever geometric insight, the search space for correspondence matching is reduced from containing the entire image to containing a single row of the image. For more information on epipolar geometry refer to Hartley and Zisserman [27].

All stereo images that are discussed in this document are assumed to be rectified, and thus the same object features occur on the same row in the left and right images. The problem of stereo correspondence matching is thus reduced to finding the one dimensional offset of the same object space point between the left and right image, a quantity which is called the disparity. The disparity of a pixel may be defined numerically as

$$d = x_L - x_R \tag{1.2}$$

where x_L is the column number of the location of an image feature in the left image, and x_R is the same feature's location column number in the right image. Solving the stereo correspondence problem for every single pixel in the overlapping region of the stereo image pair, and placing the solved disparity value into that pixel's location in a new image generates a construction called a disparity map. The disparity value of a point in two dimensional image space is used to reproject the location of the point back to three dimensional object space. The equations which map from disparity space (x, y, d) to three dimensional object space (X, Y, Z) assuming two identical ideal pinhole cameras are

$$\begin{aligned} Z &= \frac{Bf}{d} \\ X &= \frac{Zx}{f} = \frac{xB}{d} \\ Y &= \frac{Zy}{f} = \frac{yB}{d} \end{aligned} \tag{1.3}$$

where B is the baseline distance between the two cameras and f is the focal length [28]. Using the phase based dense stereo algorithm to generate three dimensional reconstructions of scenes is addressed in Section C of Chapter III.

4. Challenges to Stereo Correspondence Matching

The stereo correspondence problem is to find the same object space point in two images of the same scene captured from different locations. For a variety of reasons, the scene may not look the same from these two locations, or the region of the images may not contain enough information to match between images, and in these cases a stereo correspondence algorithm (even a perfect one) is destined to fail. By understanding situations where the algorithm is destined to fail, it can be made to fail more gracefully by knowing that it is in trouble and reporting that it has a low confidence in the reported best correspondence matches.

Fig. 3 shows a left and right stereo image pair of a scale model of the Hubble telescope which exemplify different challenges to stereo correspondence matching that are present in a space environment. The images were acquired with a horizontal baseline stereo camera system consisting of two identical cameras rigidly mounted approximately 24 centimeters apart. The differences in the images are caused by the fact that the cameras are located at geometrically different locations while they acquire the images, so the scene is bound to appear differently as seen from the location of each camera. The multi-colored circles are not present in object space, they are added to the images to denote regions of interest.

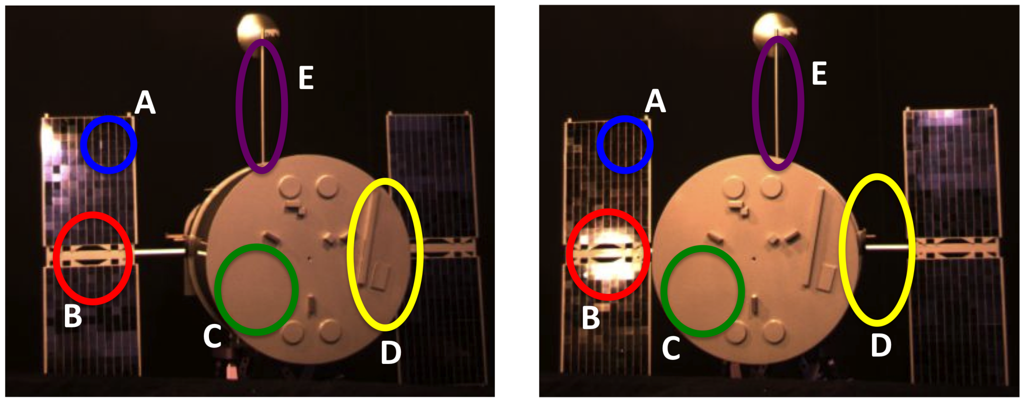


Fig. 3. Left and right stereo image pair, illustrating challenges to stereo correspondence matching algorithms.

The area in the left and right images that is circled and denoted by the letter A is an example of a region in the scene with periodic surface texture. An algorithm attempting to match a small chunk of the image within the circle on the left has several equally good choices as to where the chunk matches in the right circle, and thus several possible disparity values may be assigned. Only one of these equally appealing (to the algorithm) choices is correct. Space objects (and man-made objects in general) will frequently have many repeated textures, so this is bound to be a recurring problem that any algorithm must take into account.

The image regions denoted by the letter B are an example of a non-Lambertian surface texture causing the same location in object space to appear differently when viewed from two different angles. The solar panels are shiny, and the overhead laboratory lights are reflecting the light directly into the right camera but not the left. An algorithm attempting to match the circled region from the left camera will have no hope of locating the correct region in the right because the two appear entirely differently. Non-Lambertian surfaces may cause glare in one or both cameras, or they may cause reflections of one part of the scene to appear as surface texture in another part, both of which are challenging for a stereo correspondence matching program. This problem (like the repeated surface texture problem) is more prevalent in space objects than in natural objects because there are many shiny surfaces on spacecraft. Additionally, the lighting in space is never gentle, objects in space will either be exposed to direct sunlight or they will be in darkness.

The regions labeled C are an example of an area with bland surface texture. Any small chunk of texture from within the circle in the left image could equally likely match any region within the circle in the right image. Bland surface texture regions are a difficult problem for a stereo matching algorithm, but not a fatal one. The stereo imaging system can be equipped with a pattern projection device which projects a surface texture onto the scene at the same times as the cameras acquire images. By performing stereo correspondence matching using the projected texture, the scene may be successfully mapped. If this pattern projection system is not an option, a simple check may be performed on the object prior to choosing a correspondence match to see if there is enough resident texture available for successful matching. For the phase based dense stereo algorithm, this check is easily performed in frequency space by examining the Fourier coefficients and ensuring that enough of them are non-zero (there is interesting spatial frequency information at several scales) that a successful match may be made.

The regions marked by D are an example of the most general type of difficulty for a stereo matching algorithm, called occlusion. Notice in the left image the mounting bracket for the solar panel is visible, but comparing the similar region in the right image the bracket is not visible behind the endcap of the telescope. Because of the geometric displacement between the cameras, one part of the scene is causing another to not be visible to both cameras. Obviously something that cannot be seen by both cameras cannot be matched between the images.

A final challenge to dense stereo correspondence matches are sharp depth discontinuities, such as the antenna mast marked by the letter E. Discontinuities are challenging because dense stereo algorithms use a finite sized mask from one image to localize within the other image. If this mask happens to contain information from the front-most surface and the rear-most surface across a depth discontinuity, multiple correct disparity values could be assigned. Typically this leads to blurring across depth boundaries. The phase based dense stereo algorithm has an advantage in that the ultimate implementation uses one dimensional horizontal image strips for correspondence matching, so it is only susceptible to blurring across vertically orientated depth discontinuities. On the other hand, because the phase based algorithm uses one dimensional image strips, it must use relatively long image strips in order to contain as much information for matching purposes as an equivalent algorithm using a square mask. Thus, the region of blurring across vertically oriented depth discontinuities is wider than other algorithms.

By understanding the likely failure modes of a stereo correspondence matching algorithm such as the ones described above, software mechanisms may be put into place to detect failure of the algorithm and preventative measures may be taken. In the case that these failures are causing incorrect correspondence matches to occur, the affected regions may be flagged and removed from the disparity map. Removal from the disparity map is equivalent to not including those parts in the three dimensional reconstruction of the scene that the disparity map will be used to generate. Frequently, these gaps in the three dimensional reconstruction of the scene may be filled in by subsequent images from other more favorable vantage points. Chapter III discusses methods of quantifying the quality of disparity measurement and methods for removing incorrect correspondence matches.

CHAPTER II

PHASE BASED DENSE STEREO ALGORITHM[1]

In this chapter, the core components of the phase based dense stereo algorithm are presented. In Section A.1 a simple one dimensional example is derived that shows the underlying concepts of phase based correlation, and a dense stereo algorithm for performing stereo correspondence matching is described in Sections C and D using these concepts. This one dimensional implementation is based upon the developments of Shibahara et al. [24]. A similar process using two dimensional phase based correlation is proposed by Muquit et al. [23], and this is the basis for a simple example in Section A.2 and a two dimensional implementation that is described in Section B. Thus, this thesis is a direct extension of [23, 24].

A. Theory

1. 1-Dimensional Example

To illustrate 1-dimensional phase correspondence consider a simple sine wave of frequency ω_0, call this signal the "right signal." Also consider another sine wave of frequency ω_0, but shifted to the right by a distance x_0. Call this second signal the "left signal." Multiply the right signal by a step function whose value is one for x greater than zero, and zero for x less than zero. Multiply the left signal by a shifted step function whose value is one for x greater than x_0 and zero for x less than x_0. This construction is shown graphically in Fig. 4, and the left and right signals are

[1]Portions of this chapter are in preparation for publication [29].

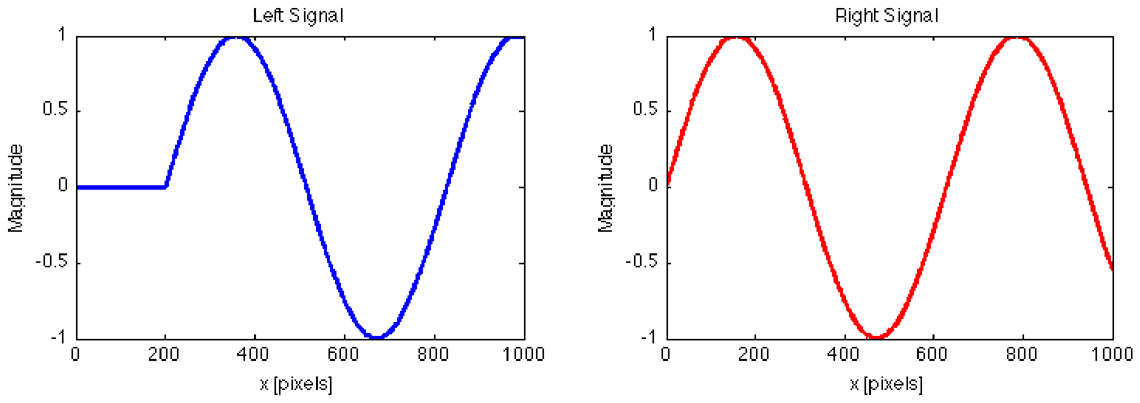


Fig. 4. 1-Dimensional phase correspondence example, constructed left and right signal.

given by:

$$\begin{aligned} r(x) &= \sin(\omega_0 x)u(x) \\ l(x) &= \sin(\omega_0(x-x_0))u(x-x_0) \\ u(x) &= \begin{cases} 0 & : x < 0 \\ 1 & : x > 0. \end{cases} \end{aligned} \tag{2.1}$$

Taking a 1-dimensional Fourier Transform of the right signal gives Eq. 2.2. The lower bound of integration is shifted to zero because the step function states that any contributions of the other terms for x less than zero are multiplied by zero.

$$\begin{aligned} \mathcal{F}\left\{r(x)\right\} &= \int_{-\infty}^{\infty} \sin(\omega_0 x)u(x)e^{-j\omega x}dx \\ &= \int_{0}^{\infty} \sin(\omega_0 x)e^{-j\omega x}dx. \end{aligned} \tag{2.2}$$

Taking a similar Fourier Transform of the left signal gives Eq. 2.3. As with the right signal the lower limit of integration is changed, but in this case the step function states that the contribution from any x less than x_0 is zero.

$$\begin{aligned} \mathcal{F}\left\{l(x)\right\} &= \int_{-\infty}^{\infty} \sin(\omega_0(x-x_0))u(x-x_0)e^{-j\omega x}dx \\ &= \int_{x_0}^{\infty} \sin(\omega_0(x-x_0))e^{-j\omega x}dx \end{aligned} \tag{2.3}$$

Introducing the substitution $X = x - x_0$, which also means $dX = dx$ (since x_0 is a constant), allows $\mathcal{F}\left\{l(x)\right\}$ to be simplified as shown in Eq. 2.4. Once again, the lower limit of integration may be changed. Because the step function ensures that there is no contribution from x less than x_0 in the integration, the lower limit is changed from x_0 to 0 with no change of meaning.

$$\begin{aligned} \mathcal{F}\left\{l(x)\right\} &= \int_{x_0}^{\infty} \sin(\omega_0 X)e^{-j\omega(X+x_0)}dX \\ &= e^{-j\omega x_0}\int_{0}^{\infty} \sin(\omega_0 X)e^{-j\omega X}dX \end{aligned} \tag{2.4}$$

Eq. 2.4 looks remarkably similar to Eq. 2.2, the only difference is the exponential term in the front. This implies that the left and right signals obey the relation:

$$\mathcal{F}\left\{l(x)\right\} = e^{-j\omega x_0}\mathcal{F}\left\{r(x)\right\}. \tag{2.5}$$

Multiplying the Fourier transformed left signal by the complex conjugate of the transformed right signal and dividing by the magnitude of that quantity forms the frequency domain correlation function $R(\omega)$, as given by Eq. 2.6. This quantity is called the cross-correlation, and is used in all branches of signal processing for measuring correlation between two signals. The particular implementation in this case is a normalized cross-correlation, ensuring that the correlation value is between 0 and 1. The final simplification in Eq. 2.6 is possible because the magnitude of a complex number $|Ae^{j\theta}|$ is A, which in the case of $e^{j\omega x_0}$ is one.

$$R(\omega) = \frac{\mathcal{F}\left\{r(x)\right\}\overline{\mathcal{F}\left\{l(x)\right\}}}{|\mathcal{F}\left\{r(x)\right\}\overline{\mathcal{F}\left\{l(x)\right\}}|} = \frac{e^{j\omega x_0}\mathcal{F}\left\{r(x)\right\}\overline{\mathcal{F}\left\{r(x)\right\}}}{|e^{j\omega x_0}||\mathcal{F}\left\{r(x)\right\}\overline{\mathcal{F}\left\{r(x)\right\}}|} = e^{j\omega x_0} \tag{2.6}$$

This quantity in Eq. 2.6 is the one dimensional frequency domain correlation function. It is a function of the frequency ω and the disparity x_0, which is the quantity of interest. To determine the value of the disparity from this complex, frequency domain representation, a one dimensional Inverse Fourier Transform is performed on $R(\omega)$, which leads to

$$\begin{aligned}\mathcal{F}^{-1}\left\{R(\omega)\right\} &= \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{j\omega x_0}e^{j\omega x}d\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{j\omega(x+x_0)}d\omega \\ &= A\delta(x-x_0) \qquad (2.7)\end{aligned}$$

Eq. 2.7 is the spatial correlation function. It is a delta function, shifted from the origin by the disparity x_0. Thus the theoretical disparity between the left and right signal is found by locating the peak of this delta function in the spatial correlation function.

Fig. 5 shows the spatial correlation function from Eq. 2.7, computed using Fast Fourier Transforms. It has a clear delta function peak centered at the expected value of x_0, which is 200 pixels in this example.

From Fig. 4, it is clear that the left signal is shifted by 200 pixels, and thus the delta function in Fig. 5 centered at $x_0 = 200$ pixels is marking the correct value of disparity. But somewhat less satisfactorily, there are other spikes and bumps in the correlation function, even for this perfectly noise free signal. Because the above derivation uses the formulae of Continuous Fourier Transformations, but the actual implementation is performed with Discrete Fourier Transforms (FFTs in MATLAB in this case), the result is not perfect. Additionally, the Fourier Transform treats a signal as continuous, and assumes it to wrap around the end back to the beginning. The evenly spaced extra spikes are due to this periodicity. In the presence of noise

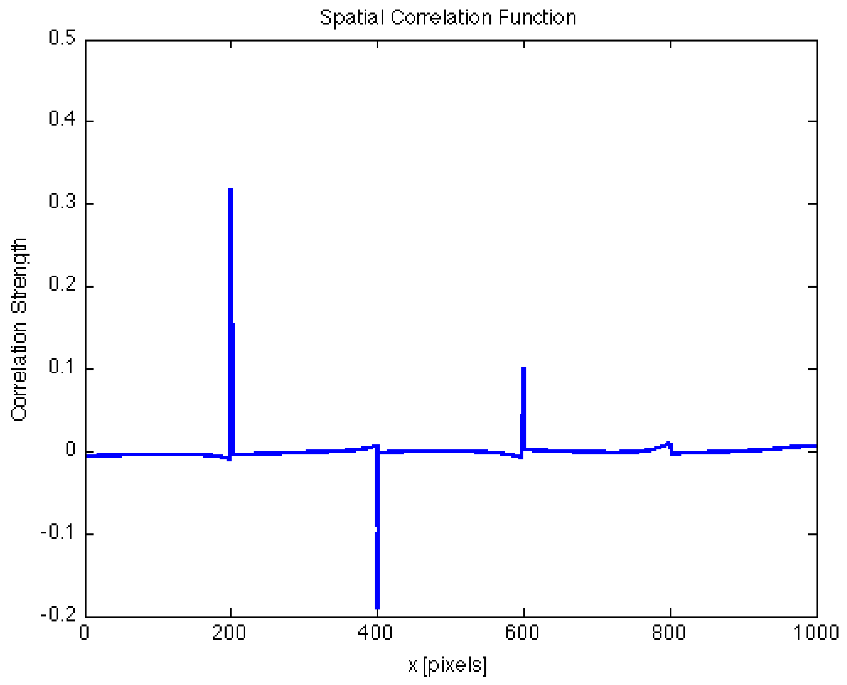


Fig. 5. 1-Dimensional phase correspondence example, spatial correlation function.

(as would be present in an actual image signal), noise-induced spikes could grow to the same size or even higher than the correct delta function spike, and in doing so, cause the algorithm to associate an incorrect value of disparity to the pixel of interest. Making the (not-unreasonable) assumption that it is known *a priori* the minimum and maximum possible disparity value in the scene, a spike that lies between the two bounds may be chosen as the correct delta function spike marking the disparity. If the dense stereo algorithm is being used in a SLAM system then it is a valid assumption that at least an estimate of the minimum and maximum possible disparity for the stereo pair are known quantities because a coarse estimate of the geometry is determined during the feature matching step when the relative movement of the camera system between frames is estimated. Fig. 6 shows an example of using knowledge of the upper and lower bounds of the possible solution set to eliminate the potential for false matches in the spatial correlation function.

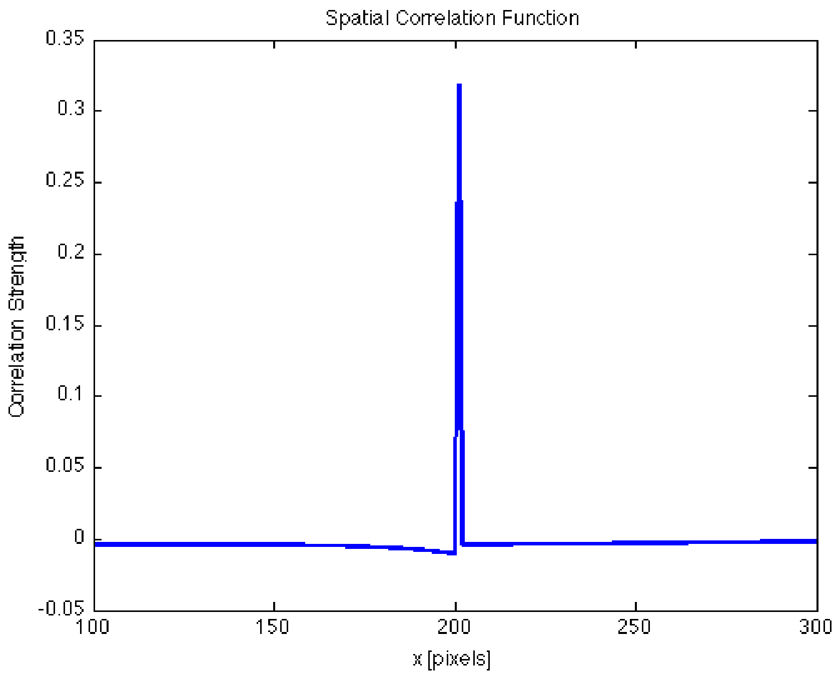


Fig. 6. 1-Dimensional phase correspondence example, bounded spatial correlation function.

2. 2-Dimensional Example

The example in Section A.1 calculates the disparity (shift between left and right signals) for a 1-dimensional signal. A similar technique can be applied in 2-dimensions by taking 2D patches of images as input signals and applying 2-dimensional Fourier Transforms to correlate in the frequency domain and 2-dimensional Inverse Fourier Transforms to retrieve the answer in the spatial domain. For example, consider the two images shown in Fig 7. Call the image in Fig. 7(a) the "search space," and the image in Fig. 7(b) the "object of interest." Knowing that the object of interest is contained within the search space, the 2D phase correspondence technique can be used to find the location.

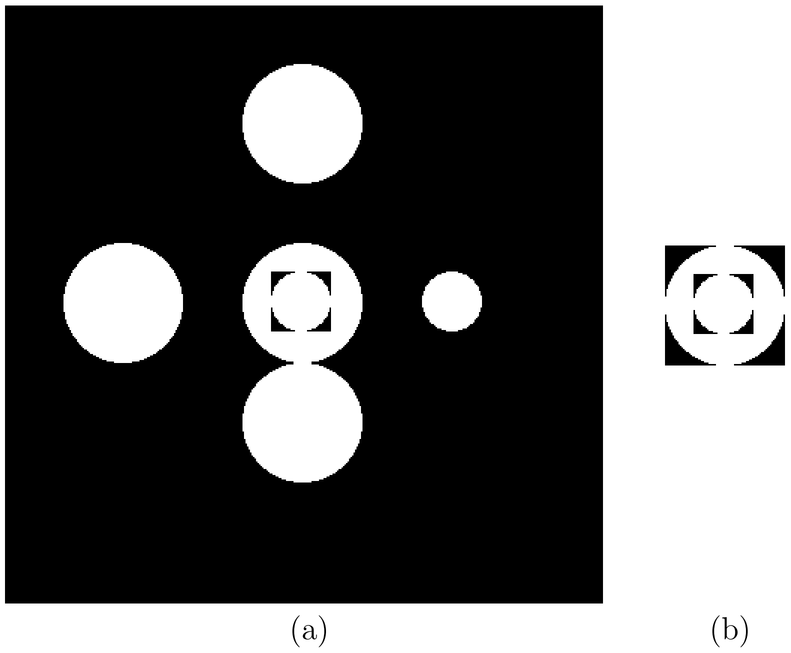


Fig. 7. 2-Dimensional phase correspondence example: (a) Search space image; (b) Object of interest image.

The first step is to pad the search space image with zeros, and then pad the object of interest image to the same size as the enlarged search space. The zero-padding is a widely used practice to mitigate edge effects at the borders of the image that result from the periodicity of the Fourier Transform, as is described above in Section A.1. The images must be padded to the same size so that Discrete Fourier Transforms of the same number of samples may be applied and array-wise multiplications of the transformed images may be performed. Fig. 8(a) shows the search space image which has been enlarged by padding it with zeros on the right and bottom edges, and

Fig. 8(b) shows the object of interest image which has been padded to the same size as the newly enlarged search space.

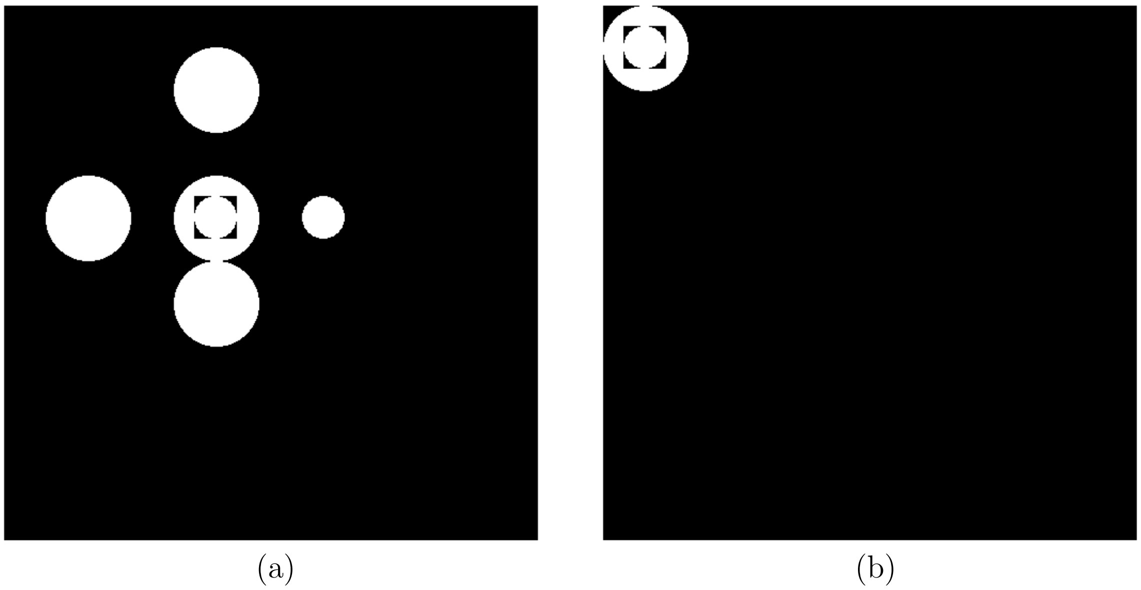
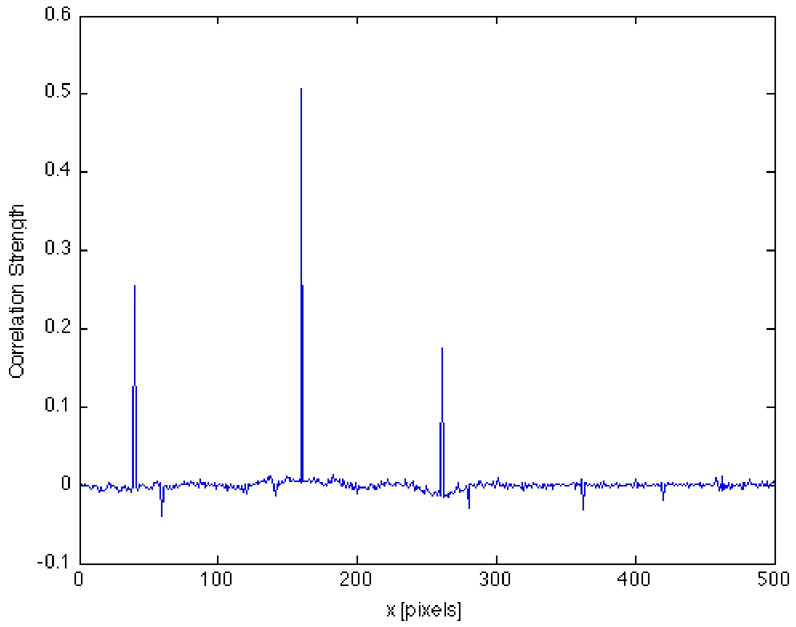


Fig. 8. 2-Dimensional phase correspondence example, padded images: (a) Padded search space image; (b) Padded object of interest image.

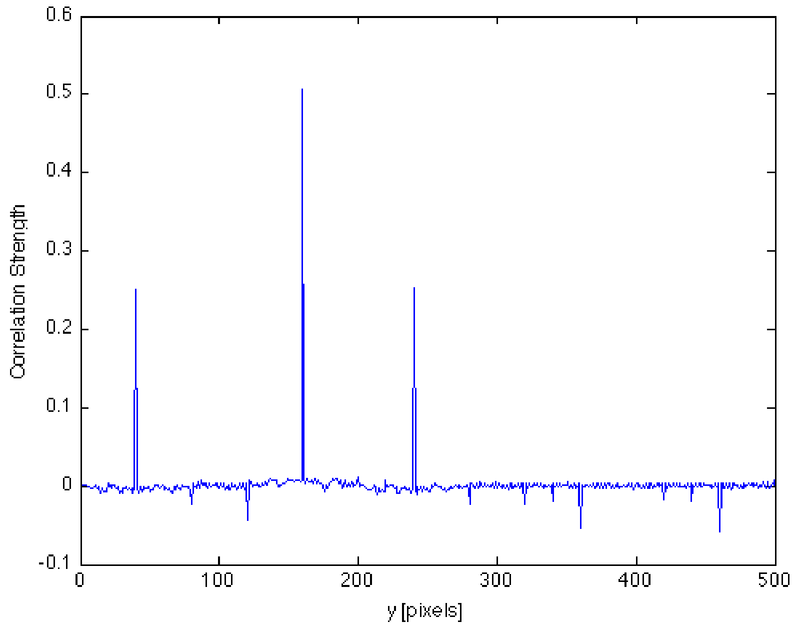
Taking a 2D Fourier Transform of the padded search space and object of interest images yield 2D complex arrays that contain 2D spectra of the spatial frequency information contained within the two images. Constructing the frequency domain correlation function follows a similar process as the one dimensional case given by Eq. 2.6, except in this case the multiplications and divisions are 2D element-wise operations. Performing a 2D Inverse Fourier Transform on the frequency correlation function gives the spatial correlation function, which is also a 2D function. Finding the maximum delta function peak of this 2D function gives the x and y coordinates of the object of interest image within the search space image. Searching for the maximum value of the array in this example yields $(x, y) = (160, 160)$. A plot of row number 160 of the spatial correlation function is shown in Fig. 9(a), and a plot of column number 160 of the spatial correlation function is shown in Fig. 9(b).

Intuitively this result makes sense. In Fig. 9(a) there are three spikes of correlation strength, corresponding to the three possible solutions of the three circles centered on row 160 of the search space image. The central peak is the highest because it is (by inspection) the correct answer, but mathematically it is higher because the regions of the image corresponding to the other two circles do not look as much like the object of interest as the correct region does. The peak on the left is taller than the peak on the right because it is a circle of the same diameter as the correct answer, but it lacks the distinguishing internal features of the central circle corresponding to the central peak. In Fig. 9(b) there are also three spikes corresponding to the three circles centered along column 160 in the search space image. The left and right peaks are the same height because the circles are identical to each other, but they are a less identical match than the central peak corresponding to the correct central circle.

This example shows a situation where the 2-dimensional method allows the proper correspondence matching to be realized, despite the fact that there is not one, and only one, clear cut correct answer. The 1-dimensional method would have more trouble choosing the correct match in this situation. To see why, consider a column plot of the column which passes through the center of the concentric circles of the object of interest image in Fig. 8(b). The central column does not contain any of the interior features of the object of interest, so the column plot is simply a step function which has the value one for $0 < y < D$ (choosing the top left corner of the image as the origin), and zero for $y > D$, where D is the diameter of the exterior white circle in the object of interest image. Considering a column plot through the centers of the three white circles in Fig. 8(a), it is intuitive to see that there will be three step functions of the same shape as the one in the column plot from the object



(a)



(b)

Fig. 9. 2-Dimensional phase correspondence example, plots of spatial correlation function: (a) Row plot of row number 160 of spatial correlation function; (b) Column plot of column number 160 of spatial correlation function.

of interest image. Looking at the problem from this 1-dimensional point of view, each of the three circles in Fig. 8(a) is equally as correct of an answer as the others because the centered 1-dimensional strip does not contain enough information to distinguish between them. This scenario is rather contrived to expose this issue, usually the 1D algorithm does not encounter a uniqueness problem of this degree. This situation falls under the category of repeated texture regions, as is described in Section 4 of Chapter I.

B. An Initial 2-Dimensional Implementation in MATLAB

This section presents a 2-dimensional stereo correspondence algorithm to test the power of phase based dense stereo in an easy to debug environment before migrating to C code and parallelization. It is an un-optimized implementation to match single pixels between left and right images of stereo image pairs from representative scenes, intended as a simple check of the robustness of the algorithm. Had the algorithm failed to satisfactorily find correspondence in real data, a new approach could have been chosen at this point. Consider the rectified left and right image from the stereo image pair shown in Fig. 10. Choosing a random point in the right image, it is desirable for the algorithm to be able to successfully find the same object space point in the left image. Selecting a small square of the right image centered around the point of interest (this square has an adjustable side length in pixels, which can be treated as a tuning parameter), the object of interest image is created. Because the input images are rectified, is is known that the corresponding point in the left image of the stereo pair lies in the same row as it is in the right image. Thus by taking a strip from the left image of the same height as the as the square from the right image, and which covers the entire width of the left image, the search space image is formed.

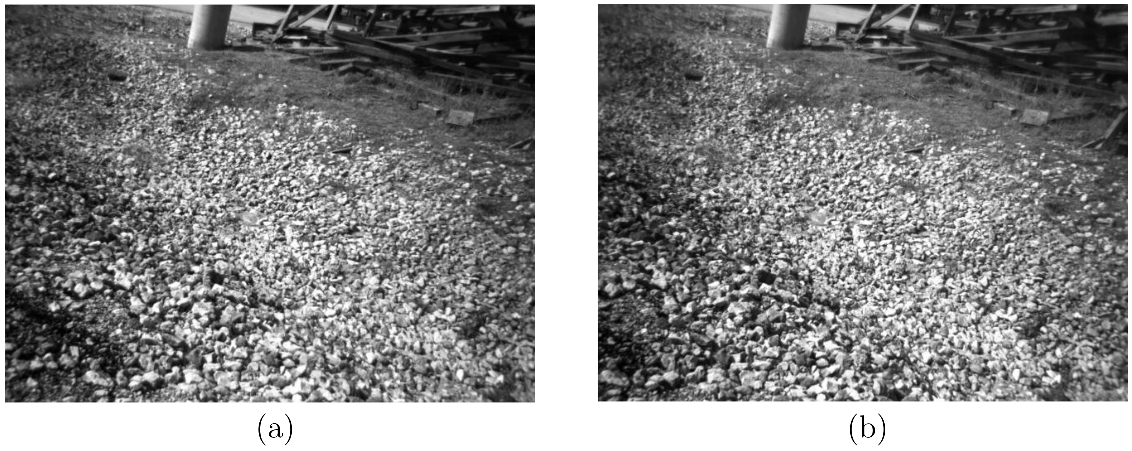


Fig. 10. Stereo image pair, rectified left and right image: (a) Left image; (b) Right image.

From the stereo geometry it is known that the corresponding point in the left image is located in a column number that is greater than or equal to the column number which contains it in the right image.

MATLAB's `fft2()` function takes a 1-dimensional Fast Fourier Transform first along the column direction of the input array and then takes another 1-D FFT along the row direction. Since the algorithm utilizes 2-dimensional Fourier Transforms, the search space and the object of interest images must be padded with zeros both in the horizontal and the vertical direction. Typical padded object of interest and search space images are shown in Fig. 11.

2-Dimensional FFTs are taken of the padded search space and the padded object of interest image, resulting in two arrays containing the complex 2-dimensional frequency components. Multiplying these together by the same process as in the 2-dimensional example in Section A.2 of this chapter, the 2-dimensional frequency domain correlation function is formed. Taking a 2-dimensional Inverse Fast Fourier transform using MATLAB's `ifft2()` function results in the spatial correlation function. Solving the stereo correspondence problem between the left and right images of

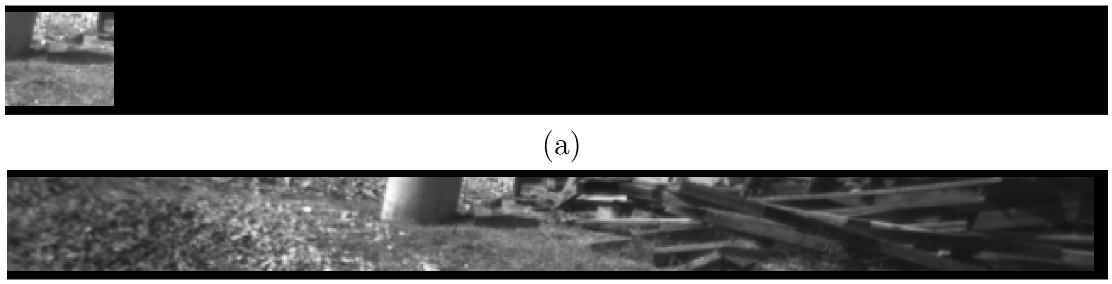


Fig. 11. 2-Dimensional implementation, object of interest and search space images: (a) Padded object of interest image; (b) Padded search space image.

the stereo pair is thus reduced to finding the delta function peak in spatial correlation function. A row plot of the spatial correlation function containing the delta function peak for the object of interest and search space images in Figs. 11 is shown in Fig. 12. The delta function peak representing the solution to the stereo correspondence problem between the object of interest image and the search space image is clearly visible. Using the equation of disparity that is given in Section B.3 of Chapter I to locate the central point of the object of interest image in left image of the stereo pair, and then drawing a box of the same size, yields the two corresponding regions in the left and right images of the stereo pair, shown in Fig 13.

Thus it is verified that the algorithm can successfully solve the correspondence problem between the left and right image and locate the same point in each for a typical stereo image pair. The next step is to see how reliably it can solve for stereo correspondences, which is to say, if it solves for the correspondence match for every pixel in the overlapping region of the stereo pair, will it generate an accurate disparity map of the scene. By running the matching code in a double `for` loop as a simple, brute force application, its performance over the entire image may be evaluated. Because this MATLAB code is implemented as a proof of concept for the ability of

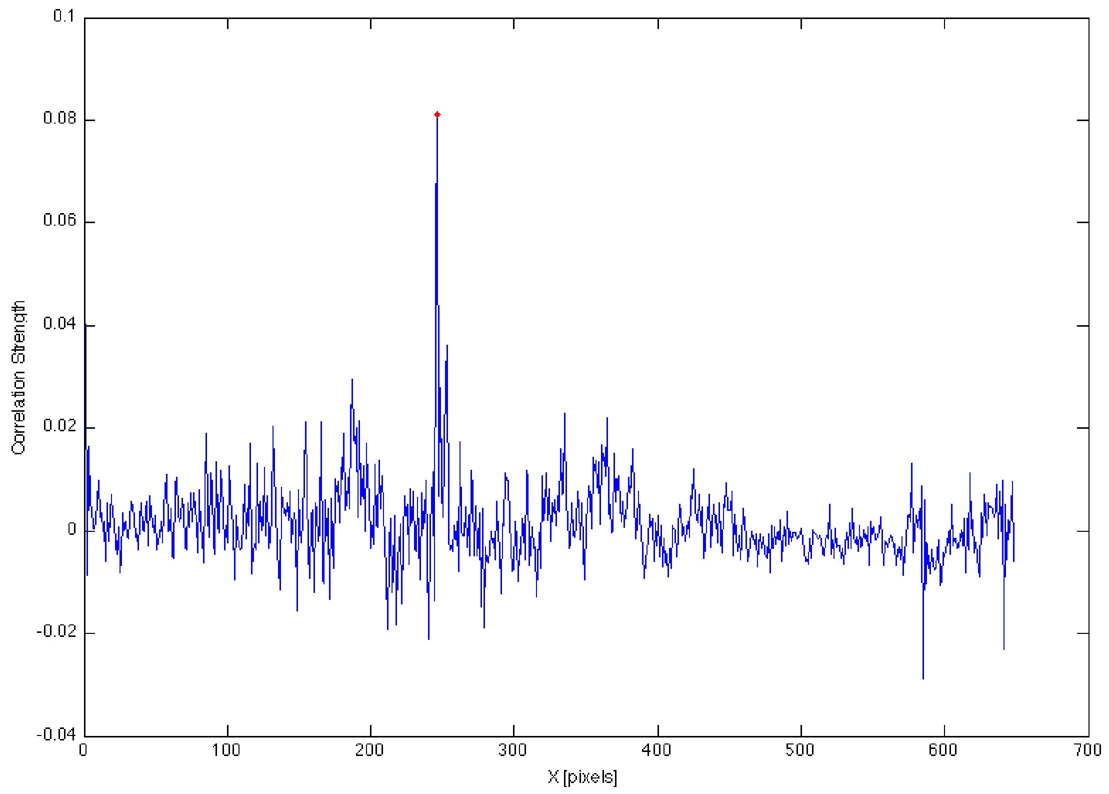


Fig. 12. 2-Dimensional implementation, row plot from typical spatial correlation function.

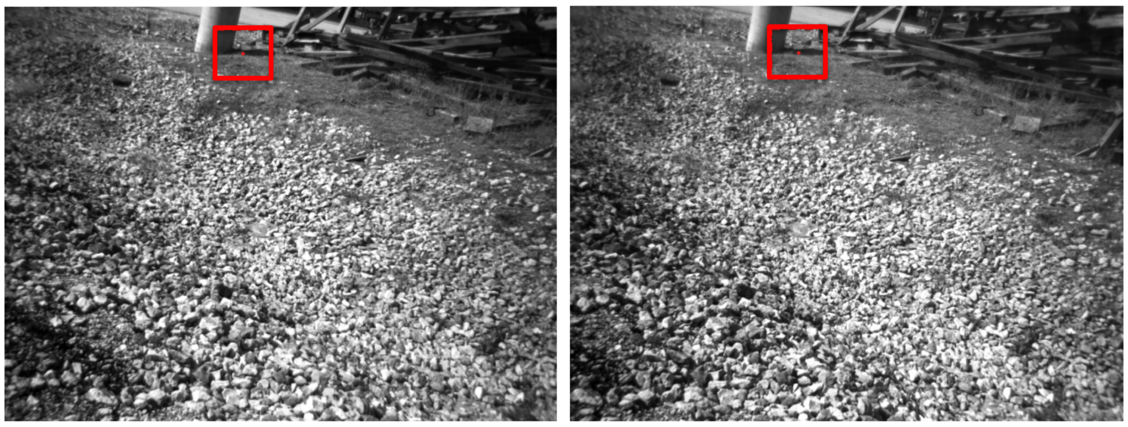


Fig. 13. 2-Dimensional implementation, matched regions in left and right image.

the algorithm to find correspondence matches between single points of left and right images, it is not optimized for speed or efficiency, and this test for every pixel in the overlapping area takes much longer to run than an algorithm designed to run across entire images would.

Run time aside, the results are promising. Fig 14 shows the resulting right image disparity map for the stereo image pair in Fig. 10. In this image, lighter shades correspond to larger disparity, and thus closer points to the camera in object space. Darker shades represent lower disparity, and black represents zero disparity regions, where no correspondence could be solved. The reason for the black border around the outside of the image is that a large object of interest image (a 64 pixel by 64 pixel mask) is used in these initial tests to ensure correspondence matching can be achieved, and thus the pixels within half the size of object of interest mask from the edge of the image cannot be solved. This border is larger on the right side of the image because this is a right disparity image (referenced to the right image in the stereo pair), and so the far right pixels are not visible in the left camera, and thus no correspondence is possible in this area.

By visual inspection, this disparity map appears to be a decent representation of the scene, however, it is corrupted by salt and pepper noise (individual pixels whose values are much higher or lower than those of their neighbors). This salt and pepper noise is due to the fact that the disparity value for each pixel is solved independently, and thus there is no smoothing. Many dense stereo algorithms have constraints that force continuity between pixels in disparity space. No such constraints are present in the phase based dense stereo algorithm. The fact that the noise is fairly evenly distributed, and is not more prevalent than it is, is actually a testament to the self-consistency of the algorithm. Since each pixel is individually solved, a mostly smooth disparity map indicates that the solution to the correspondence problem is at least

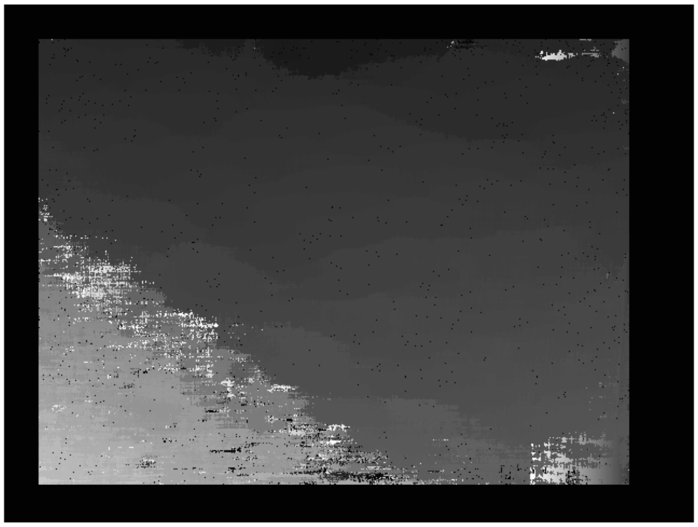


Fig. 14. 2-Dimensional implementation, resulting raw disparity map.

similar and repeatable for neighboring pixels. Most of the salt and pepper noise for isolated pixels can be statistically eliminated by convolving the disparity map with a non-linear median filter. Fig. 15 shows the disparity image of Fig. 14 which has been median filtered. More rigorous tests of the actual accuracy of the resulting disparity maps from the algorithm are presented in Chapters III and V.

C. 1-Dimensional Implementation in C

The algorithm discussed in Section II.B is effective for solving the correspondence problem of matching between left and right images, but because it is implemented in MATLAB, and utilizes 2-dimensional FFTs, it runs much too slowly for any practical application. This section describes a 1-dimensional version implemented in C which is much faster. This algorithm was developed in 64-bit Debug mode in Microsoft Visual Studio 2008 on a computer with Microsoft Windows Vista, and utilizes many pre-programmed functions from the open source software package OpenCV 2.1.

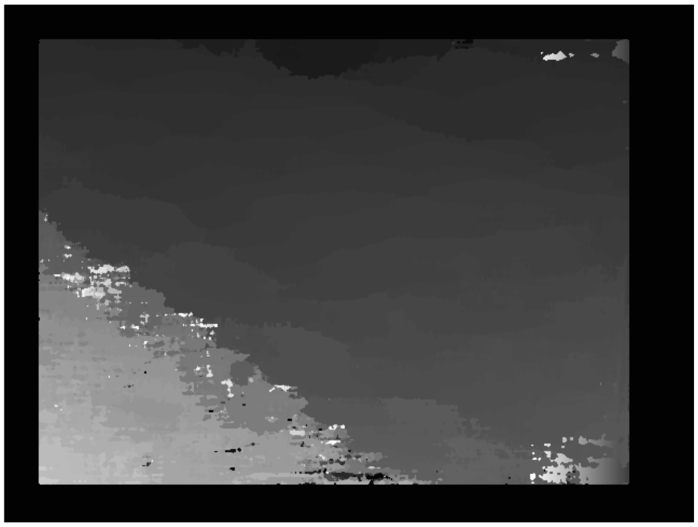


Fig. 15. 2-Dimensional implementation, resulting median filtered disparity map.

OpenCV is a library of optimized functions for computer vision, designed to allow a user to quickly build and debug computer vision related programs. It is a platform invariant library containing C structures and functions, C++ classes, and Python bindings. The algorithm in question utilizes OpenCV data structures, image manipulation and filtering functions, and FFT functions. The library has excellent documentation, both online in wiki format [30], and in book format [19].

This 1-dimensional implementation works along entire image columns at a time, solving the correspondence problem for each row individually. Because 1D FFTs are used, the images need only be padded in the horizontal direction. With the assumption that the input images are rectified, the correct solution to the correspondence problem between left and right image is restricted to a one dimensional search along epipolar lines bounded by the minimum and maximum possible disparity. For each column in the right image, a vertical strip of the image is taken, centered around that particular column, which forms the object of interest array. From the left image, a vertical strip is also selected, its location being chosen by the column number from the right image plus an initial guess of the disparity (the minimum disparity is used), and its length being the strip length from the right image (a tuning parameter) plus the maximum disparity (so that the same strip will always be visible in both images). These image strip boundaries are summarized by Eq. 2.8.

$$\begin{aligned} X^{min}_{ObjOfInterest} &= ColNum - StripLength/2 \\ X^{max}_{ObjOfInterest} &= ColNum + StripLength/2 - 1 \\ X^{min}_{SearchSpace} &= ColNum + InitGuess - StripLength/2 \\ X^{max}_{SearchSpace} &= ColNum + InitGuess + StripLength/2 + MaxDis - 1 \end{aligned} \tag{2.8}$$

In these equations, the term $InitGuess$ is an estimate of the disparity value for the pixel of interest. This guess may come either from a feature based stereo algorithm which has been run on the image pair prior to the dense stereo, or in the absence of that, the value of $MinDis$ may be used.

To eliminate the edge effects caused by the periodic nature of the Fourier Transform, each row is multiplied by a Hann Function of length $X^{max} - X^{min}$. This multiplication is carried out as an arraywise multiplication in the spatial domain, before any FFTs are taken. The purpose of this window function is to allow the image signal in each row to smoothly transition between the high values in the area of the image strip to the zero values in the padding strip without a sharp discontinuity. If the region being matched is a bland or otherwise difficult part of the image to match, this discontinuity (which shows up in the Frequency Domain as a very low frequency feature) will dominate the image signal and cause false matches. A Hann Window function of length M is given by Eq. 2.9,

$$h(x) = c + (c-1)\cos\frac{2\pi x}{M-1} \tag{2.9}$$

where $0 \leq x \leq (M-1)$ [31].

An example object of interest array and a search space array are shown in Fig. 16. These arrays are typical of what would be encountered solving the correspondence problem for a column of pixels in the stereo pair of images from the ditch scene of Fig. 10. Each row in these arrays is an individual 1D correspondence problem, the solution of which is the disparity value for one single pixel in the right image. The value of the disparity is found by the same method detailed in the 1 dimensional example of Section A.1 of this chapter.

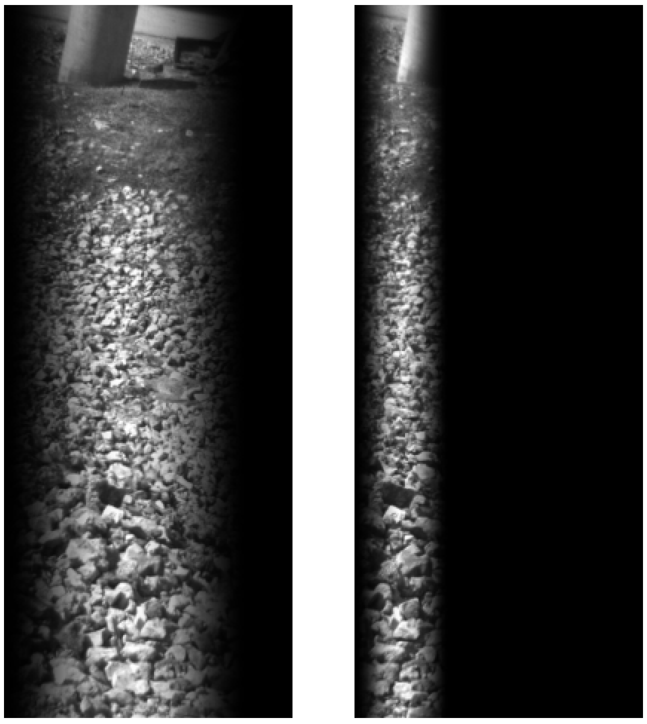


Fig. 16. 1-Dimensional implementation, padded search space and object of interest arrays: (a) Padded search space array; (b) Padded object of interest array.

An example of the typical spectral response for a column of pixels in a well textured scene like the stereo image pair in Fig. 10 is shown in Fig. 17. Each row of the 2-dimesional spatial correlation response array is the equivalent to the 1-dimensional spatial correlation function shown in Fig. 5 for the simple sine wave example. For example, the row indicated by the horizontal white stripe is plotted. A clear peak is seen, the offset of this peak from the left edge of the array denoting the disparity value of the pixel of interest. Each row is the solution for a different pixel, and each column has a similar 2-dimensional array.

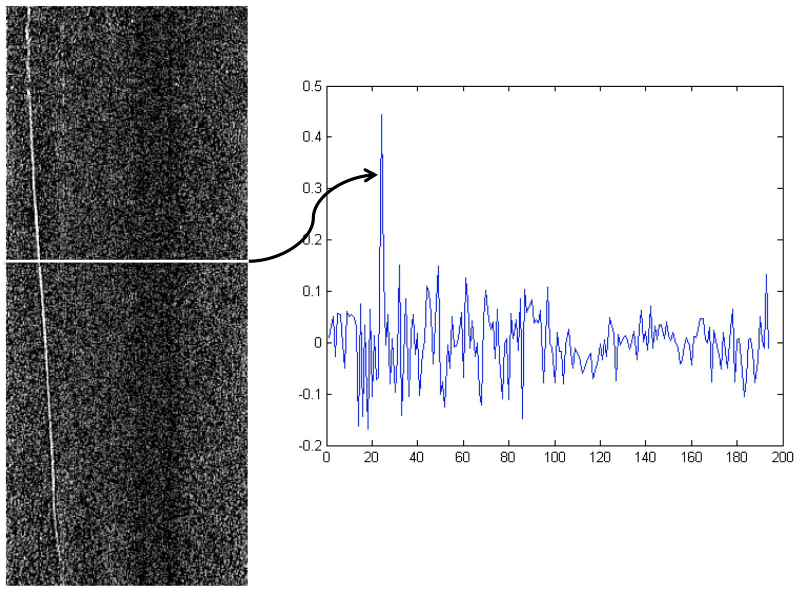


Fig. 17. 1-Dimensional implementation, typical spatial correlation function response for a single column.

By solving for the disparity of each pixel in the overlapping region of the stereo pair, and by placing the value of the disparity in a separate array, its location in the array determined by the location of the pixel of interest in the original image, a disparity map is formed. Because there are two (or more) cameras required to capture a stereo pair of images, different disparity maps may be generated for the same image pair which are referenced to the different cameras. Fig. 18 shows a resultant disparity map of the ditch scene from the stereo image pair of Fig. 10. This particular disparity map is referenced to the right stereo image, meaning the right image was used to generate the search space arrays, and the left image was used to generate the object of interest arrays. It is easy to generate a left referenced disparity map simply by switching the input images (much more on this in Ch. III). In this image, lighter shades correspond to larger disparity, and thus closer points to the camera in object space. Darker shades represent lower disparity, and black represents zero disparity regions, where no correspondence could be solved.

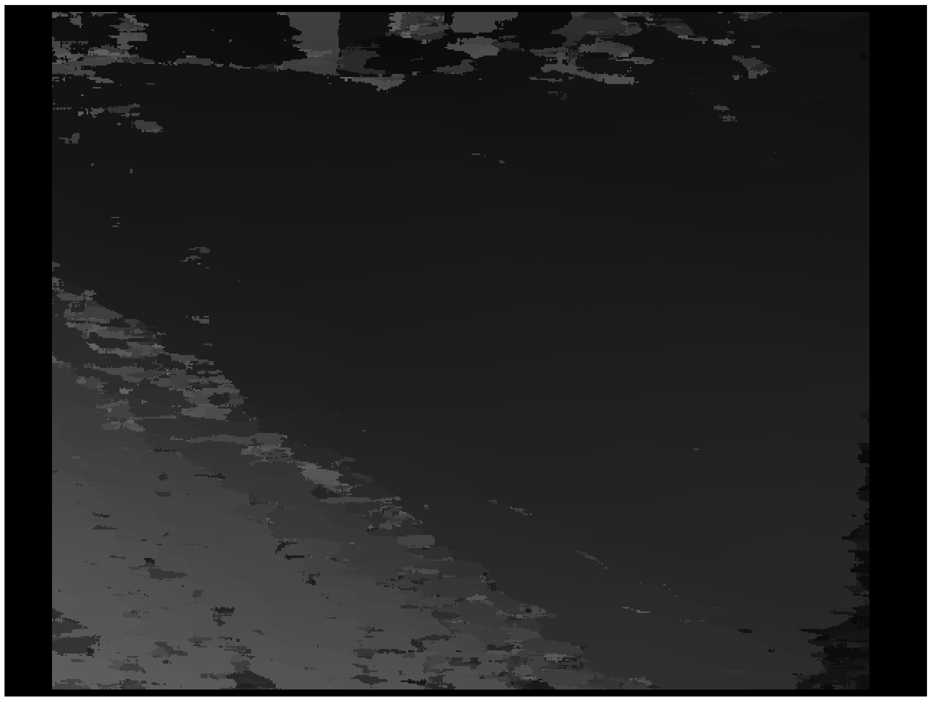


Fig. 18. 1-Dimensional implementation, typical right referenced disparity map.

This resulting disparity map shown in Fig. 18 resembles the one which is produced by the two dimensional MATLAB algorithm of Section B, shown in Fig. 15. There appear to be more fine details and closer spaced disparity steps in this image, and that is because the 2D disparity map was generated using a huge spatial mask (64 pixels by 64 pixels), so any image features of size less than that are blurred. This is not as prevalent for the 1D implementation (at least in the vertical direction), because one dimensional image strips are used. There is still blurring in the horizontal direction, the extent of which is determined by the size of image strip used. This concept is explored more in Chapter V.

Although the disparity map shown in Fig. 18 could be classified as “better” than the resultant disparity map of the two dimensional implementation, it still is not perfect. There clearly remain incorrect solutions to the correspondence problem for some pixels (for a variety of reasons), i.e., the pixels near the top of the images which contain the lamp post and the wood pile. There are also pixels near the edge of the disparity map where a good correspondence could not be solved because the pixels are close to the edge of the image and not enough information was available. These incorrectly matched correspondences must be removed in order to have confidence in the accuracy of the reconstructed geometry of the scene. In Chapter III, the quality of the correspondence match for each pixel is quantified, and routines are developed for the removal of pixels which have been incorrectly matched. Also, several improvements to the algorithm are developed in this thesis, which drastically decrease the number of incorrectly matched pixels.

D. 1-Dimensional Implementation in MATLAB

As a verification step during the development of the one dimensional phase based dense stereo algorithm in C and CUDA, the same algorithm was developed simultaneously in MATLAB. This served the purpose of allowing rapid development of improvements, and also as an independent verification of results. All three implementations return the same results, as is explored in Chapter IV.

CHAPTER III

ALGORITHMIC IMPROVEMENTS, QUALITY METRICS, AND 3D RECONSTRUCTIONS[1]

This chapter presents algorithmic improvements to enhance the performance and accuracy of the phase based dense stereo algorithm, as well as quality control methods that remove incorrect correspondence matches from the resulting disparity maps. Also discussed is an implementation of the reprojection from disparity space to object space.

A. Algorithmic Improvements

1. Averaging Spatial Correlation Functions

Fig. 17 shows a typical spatial correlation array. This array is the solution to the correspondence problem for an entire column of image pixels, each row a separate solution for a separate pixel within the column. A clear delta function peak is visible for most rows (seen as a bright white stripe through the array). In some cases the correspondence match may not be so clear, and the correct peak may be lost within the noise. This scenario where the signal to noise ratio becomes close to one is visible near the bottom of the spatial correlation array where the white stripe becomes less clearly defined. Shibahara et al. [24] call the ratio of the height of the correct peak to the background noise (visible as the other spiky noise in Fig. 17) the Peak-to-Noise Ratio (PNR), and their proposed solution to improve the PNR in bad regions is to average together the one dimensional correlation functions from several rows. The reasoning behind this is that the noise will be randomly distributed whereas the

[1]Portions of this chapter are in preparation for publication [29].

correct correlation peak will be in the same location (or nearly the same location) in consecutive rows. An obvious assumption that is being imposed here is that the disparity does not vary too quickly across rows, so it is essential to only average a small number of rows together or significant error can potentially be introduced.

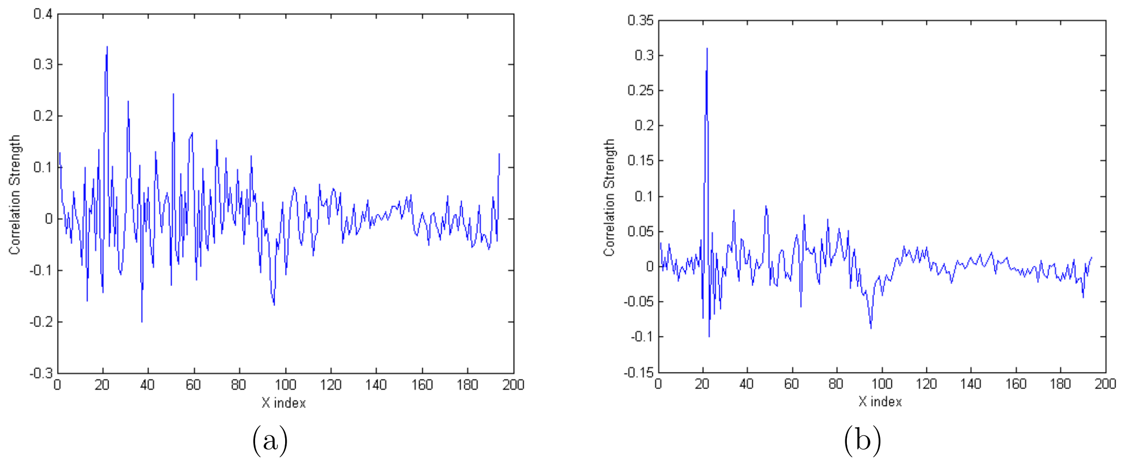


Fig. 19. Averaging spatial correlation functions, effect on one dimensional spatial correlation function: (a) Un-averaged typical spatial correlation function; (b) Result of averaging 9 spatial correlation functions of consecutive rows.

Fig. 19 shows the improvement by averaging together several spatial correlation functions from adjacent rows above and below the row of interest. In Fig. 19(a) the PNR of a single one dimensional spatial correlation function from a noisy section of the image exhibiting poor matching is about 1.5, however by averaging the correlation functions from 9 consecutive rows, the PNR is increased to about 3 as shown in Fig. 19(b).

2. Sub-Pixel Disparity Calculation

In Section A.1 of Ch. II the one dimensional spatial correlation function is defined as the result of the Inverse FFT of the frequency domain correlation function. The maximum delta function spike located in the spatial correlation function marks the disparity shift of the pixel of interest. Shibahara et al. [24] state that the peak in the spatial correlation function may be modeled as

$$r(n) \simeq \frac{\alpha}{N} \frac{\sin\{\pi(n+\delta)\}}{\sin\{(\frac{\pi}{N}(n+\delta)\}} \tag{3.1}$$

where $n = -M, ..., M$ is the index number of the element of the function, $N = 2M+1$ is the length of the Fourier Transform, and α is a scaling parameter. By using this approximation to the true peak shape, a curve may be fit to the spatial correlation function using α and δ as fitting parameters. Thus the disparity is estimated as a constant quantity instead of as an integer quantity (recall that in the description in Ch. II, simply the index number of the maximum value of the correlation function is taken to be the value of the disparity). Estimating the disparity as a continuous quantity allows for smoother surfaces and eliminates the jaggedness in disparity maps and 3D reconstructions caused by forcing disparity to assume an integer value. The idea of a non-integer disparity algorithm is nothing new. Tian and Huhns [32] provide an excellent survey (written early in the history of digital dense stereo calculation history) of available sub-pixel disparity estimation techniques.

The peak shape function in Eq. 3.1 is an approximation to a sinc function, which itself is an approximation to a delta function. A one dimensional Gaussian may more easily and efficiently be fit to the peak shape, which is a simple linear least squares estimation problem. The problem may be summarized as taking a small strip of length n from the one dimensional spatial correlation function containing the peak and fitting an n point Gaussian of the form

$$\tilde{m}_i = a \exp\left(\frac{-(x_i - \mu)^2}{2\sigma^2}\right) \tag{3.2}$$

where a is the amplitude, x_i is the index location of the ith point, μ is the mean, and

σ^2 is the variance. To perform the fit, the variable ξ_i is defined as

$$\xi_i = \ln \tilde{m}_i = \ln a - \frac{(x_i - \mu)^2}{2\gamma} \tag{3.3}$$

where $\gamma = \sigma^2$. This expression may be simplified to

$$x_i^2 = 2\gamma \ln a - \mu^2 + 2\mu x_i - 2\xi_i \gamma, \tag{3.4}$$

which when the substitutions $c = 2\gamma \ln a - \mu^2$ and $x_i^2 = \tilde{y}_i$ are introduced, becomes

$$\tilde{y}_i = c + 2\mu x_i - 2\gamma \xi_i. \tag{3.5}$$

Writing this equation as a matrix yields

$$\begin{aligned} \tilde{\boldsymbol{y}} &= \begin{bmatrix} 1 & 2x_1 & -2\xi_1 \\ \vdots & \vdots & \vdots \\ 1 & 2x_n & -2\xi_n \end{bmatrix} \begin{bmatrix} c \\ \mu \\ \gamma \end{bmatrix} \\ \tilde{\boldsymbol{y}} &= H\hat{\boldsymbol{x}}. \end{aligned} \tag{3.6}$$

Therefore the parameters may be estimated using the standard equation of linear least squares [33, 34]

$$\hat{\boldsymbol{x}} = (H^T H)^{-1} H^T \tilde{\boldsymbol{y}}. \tag{3.7}$$

The estimated parameter μ is the mean of the Gaussian function, which corresponds to the estimated peak location. An important distinction to make is that this function must be fit to an un-averaged correlation function. The averaging of correlation functions (as described in the previous section) will change the shape of the functions. Therefore, the averaged functions can be used to located the correct delta function spike in the spatial correlation function, but the Gaussian function should be fit to an un-averaged signal.

The reason for estimating disparity as a constant quantity is because disparity is inversely related to depth (Z) normal to the focal plane of the camera. If disparity is estimated as an integer value there will be discrete steps in the Z direction in the three dimensional reconstructions of the scene at which all the pixels will lie. Obviously reality does not contain discrete steps (at least not at the scale in question here), so in order to get lifelike reconstructions of scenes, disparity must be estimated as a continuous quantity.

Fig. 20 shows constructed disparity maps for a well textured plane that is almost fronto-parallel to the stereo camera system. Because the plane is nearly fronto-parallel, the discrete disparity map only contains six levels that are easily distinguished in this false color image. The disparity map of the same scene generated by treating disparity as a continuous quantity shows much smoother variation. The variation is still not perfect, (meaning the disparity map is not a perfect gradient) but it is a much better approximation to the surface than the stair stepped discrete version. The results here are shown in disparity space, in Section C three dimensional reconstructions generated with disparity as a constant quantity are shown.

3. Geometric Foreshortening Correction

Because of the baseline distance between the cameras in a stereo imaging system, objects imaged by the pair will appear slightly differently from one image to the next. For example, Fig. 21 shows a stereo pair of images (taken with a horizontal baseline between the cameras of 24cm) of a textured plane, in this case a Mayan blanket, which is rotated horizontally with respect to the focal plane of the cameras. By looking at the apparent horizontal size of the black circles near the bottom of the blanket the foreshortening effect is evident, for example, the second black circle from the left, which contains a tall bird creature. Because the left and the right camera see the circle object from different off-normal angles, there is forshortening

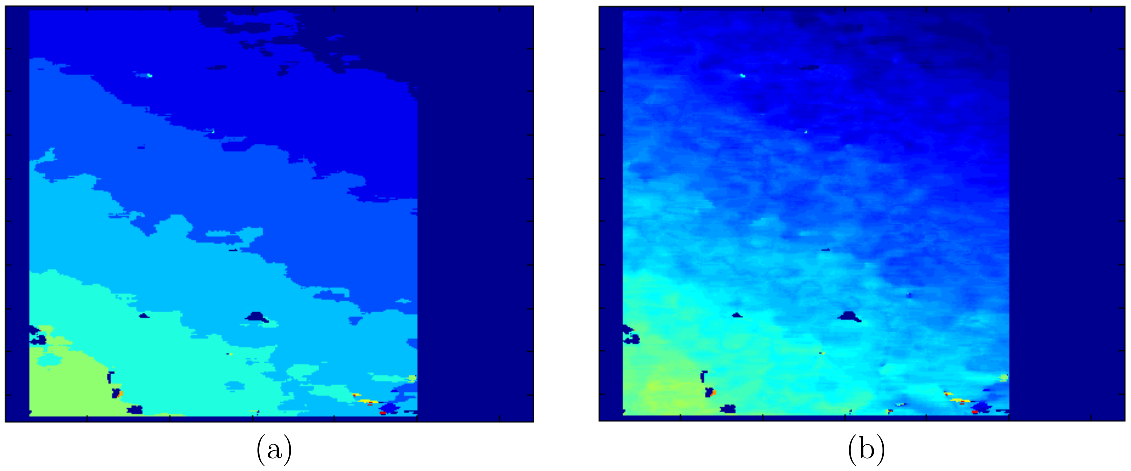


Fig. 20. Disparity map of nearly fronto-parallel plane, discrete and continuous disparity versions: (a) Disparity map of nearly fronto-parallel plane, discrete disparity version; (b) Disparity map of nearly fronto-parallel plane, continuous disparity version.

between the two camera views, and thus the circle appears horizontally compressed or expanded depending which image is taken as the reference image. This foreshortening effect represents a stretching or compression factor of the local texture, which in frequency space corresponds to a stretching or compression of the spatial frequency components. In order to most accurately solve for disparity, the foreshortening effect must be modeled and accounted for. Maimone and Schafer [35] present a method for modeling the foreshortening effect and derive a scaling factor between left and right images in a stereo pair to correct for the effect. Similarly, El-Etriby, Al-Hamadi, and Michaelis [36] provide the same scaling factor. Jones and Malik [37] use the foreshortening spatial frequency effects as a cue for their dense stereo.

Fig. 22 is a geometric cartoon of the Mayan blanket scene from Fig. 21 as seen by an observer looking from above. This geometry figure, and the ensuing developments to model the effects of foreshortening follow the arguments of Maimone and Shafer

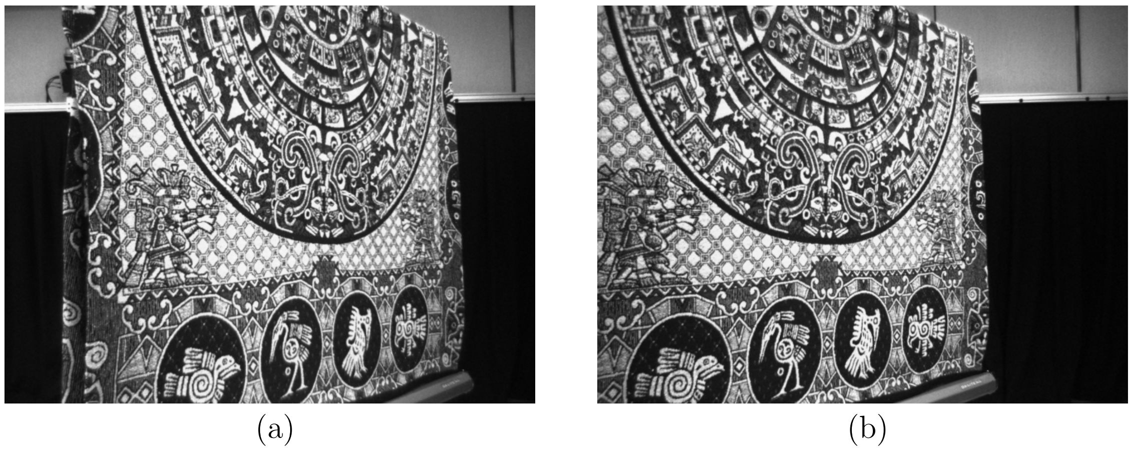


Fig. 21. Stereo image pair, exemplifying foreshortening effects: (a) Left image; (b) Right image.

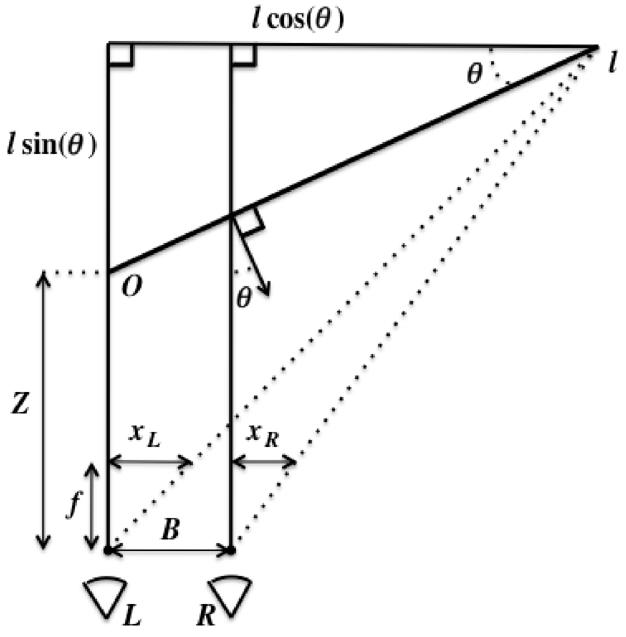


Fig. 22. Foreshortening effect geometry, stereo camera and off-normal rotated plane.

[35]. In Fig. 22 the plane of the Mayan blanket is denoted by the line which connects the point O (the point which lies on the plane that is directly in front of the left camera), and the point l. The plane is tilted in the horizontal direction by the angle θ from the focal planes of the left and right cameras L and R. The cameras both have focal length f and are separated by baseline B. x_L and x_R are the points of intersection with the left and right camera focal planes of the ray originating from the edge of the blanket at point l.

The goal is to model the ratio of the projection into the left and right image spaces (x_L and x_R) of the distance l (in object space). This sampling ratio is given by horizontal projection into the left camera image space divided by the horizontal projection into the right camera image space, which in equation form is

$$SamplingRatio = \frac{\frac{\delta l}{\delta x_L}}{\frac{\delta l}{\delta x_R}} = \frac{\delta x_R}{\delta x_L}. \tag{3.8}$$

This equation is a function of the left and right focal plane image projection size, which are both a function of the disparity. Since the disparity is the quantity of interest, a geometric description is necessary in order to continue.

By making use of the principle of similar triangles it is apparent that the following two relations are true:

$$\begin{aligned} \frac{x_L}{f} &= \frac{l\cos\theta}{Z + l\sin\theta} \\ \frac{x_R}{f} &= \frac{l\cos\theta - B}{Z + l\sin\theta}. \end{aligned} \tag{3.9}$$

Solving both of these for l yields

$$\begin{aligned} l &= \frac{Zx_L}{f\cos\theta - x_L\sin\theta} \\ l &= \frac{Zx_R + Bf}{f\cos\theta - x_R\sin\theta}. \end{aligned} \tag{3.10}$$

By setting the two expressions in Eq. 3.10 equal to each other and solving for x_R, the expression

$$x_R = x_L\left(1 + \frac{B\tan\theta}{Z}\right) - \frac{Bf}{Z} \tag{3.11}$$

is revealed. This expression is useful to convert the sampling ratio of Eq. 3.8 into a form containing only geometric quantities, which is accomplished by substitution of the equation of x_R in geometric terms (Eq. 3.11) into the equation of the sampling ratio (Eq. 3.8). Doing so yields

$$\begin{aligned}\frac{\delta x_R}{\delta x_L} &= \frac{\delta\left(x_L\left(1+\frac{B\tan\theta}{Z}\right)-\frac{Bf}{Z}\right)}{\delta x_L} \\ &= 1+\frac{B\tan\theta}{Z}.\end{aligned} \tag{3.12}$$

It is necessary to eliminate the distance between the camera and the object (Z) from Eq. 3.12 because that is not a known quantity when solving for the disparity in image space. This may be accomplished by invoking the definition of disparity as described in Chapter I:

$$\begin{aligned}disparity &= x_L - x_R \\ &= -x_L\frac{B}{Z}\tan\theta + \frac{Bf}{Z}.\end{aligned} \tag{3.13}$$

Solving Eq. 3.13 for the quantity $\frac{B}{Z}$ yields

$$\frac{B}{Z} = \frac{disparity}{f - x_L\tan\theta} \tag{3.14}$$

which, when substituted into Eq. 3.12 for the geometric form of the sampling ratio gives the final version of the sampling ratio between left and right cameras

$$SamplingRatio = 1 + \frac{disparity\tan\theta}{f - x_L\tan\theta}. \tag{3.15}$$

This sampling ratio is a function of the horizontal component of the off normal angle of the surface θ, the position horizontally in the image x_L, the focal length f, and the disparity. At first glance this seems like an unrealistic set of *a priori* information to have while calculating correspondence matches since disparity is the quantity of interest, but later in this section it is described why this is not the case.

To reiterate, because the left and right cameras in a horizontal baseline stereo imaging system observe a non fronto-parallel object from slightly different angles due to the baseline separation between them, the same scene will appear differently as observed by the two cameras. The sampling ratio of Eq. 3.15 is the ratio of the apparent size of an object in 3-dimensional image space, as it appears projected into the 2-dimensional image space of the left and right camera. Equivalently, it is the factor of stretching or subsampling that must be done to make a small strip of one of the images have the same spatial scale as the other. It is important to note that, because one of the arrays is stretched or compressed relative to the other, it is essential to apply a correction factor to the disparity that is calculated using the modified array, serving to normalize the estimated disparity value by the correction factor. Thus, knowing the sampling ratio between the two images, the effect of geometric foreshortening may be corrected for in the dense stereo algorithms described in Ch. II and Ch. IV.

In order to portray the importance of this correction factor, its effect is demonstrated on the well textured (planar) Mayan blanket scene of Fig. 21. Uncorrected for geometric foreshortening, the native texture of the blanket is compressed drastically due to the 45 degree horizontal off normality of the plane of the blanket with respect to the focal planes of the cameras. This angle causes the phase based dense stereo algorithm (without the correction factor) to have difficulty performing correspondence matching even on the well textured plane, as illustrated by the left and right uncorrected disparity maps in Fig. 23. These disparity maps are admittedly not very good, examination reveals that the plane of the blanket is almost entirely lost within the noise caused by incorrect matches. For this planar laboratory test case,

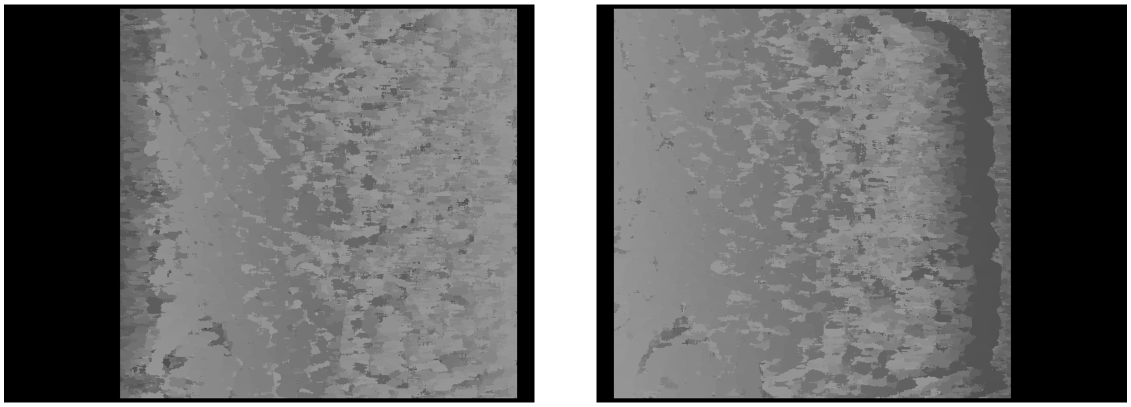


Fig. 23. Left and right referenced disparity maps from Mayan blanket scene, uncorrected for geometric foreshortening effect: (a) Left referenced uncorrected disparity map; (b) Right referenced uncorrected disparity map.

the off-normal inclination of the Mayan blanket plane is known to be 45 degrees. In order to make use of this information, a first cut dense stereo correspondence match is executed across the image pair to generate uncorrected disparity maps as shown in Fig. 23. Using these uncorrected disparity maps a rough idea of the disparity is found, which is subsequently used in the corrective stretching or condensing formula to match the image strip from one image with the other. The correction factor causes the spatial frequency of the native texture to be corrected for the sampling error caused by the different points of view and the phase based dense stereo algorithm can achieve better correspondence matching. Fig. 24 shows left and right disparity maps for the planar Mayan blanket scene which have had the geometric foreshortening correction applied. These corrected versions, when compared with the uncorrected disparity maps of Fig. 23 show the immense improvement to off-normal surfaces with the geometric foreshortening factor applied. The plane is clearly visible as the light to dark gradient of the disparity maps from left to right. There are much fewer spikes caused by incorrect disparity matches (with some exceptions in places where the texture shows little variation or has a periodic repeating pattern), and those that do exist may easily be removed by the error checking mechanisms described in Section B.

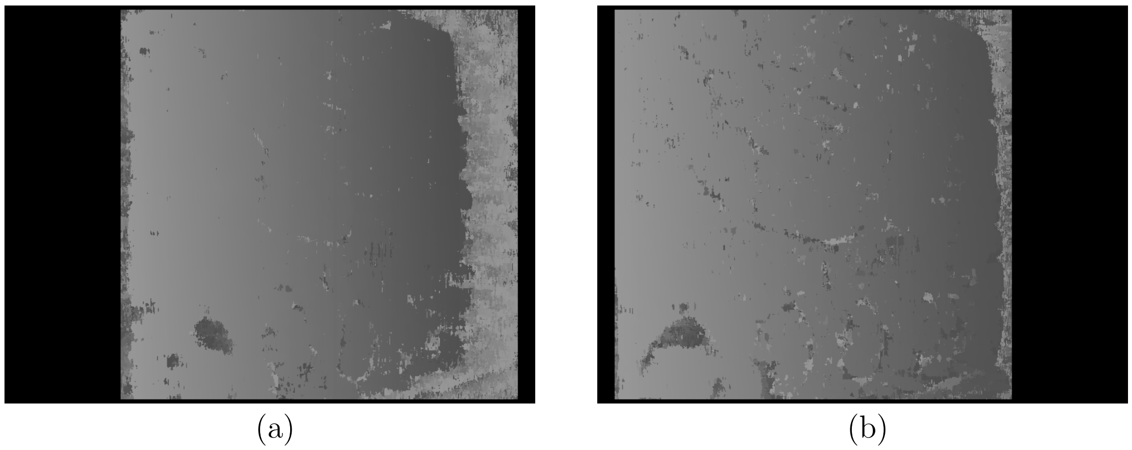
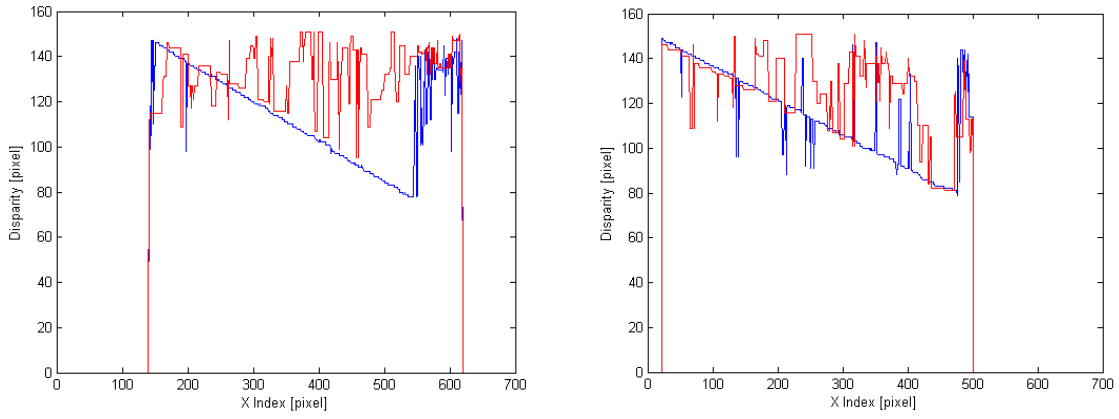


Fig. 24. Left and right referenced disparity maps from Mayan blanket scene, corrected for geometric foreshortening effect: (a) Left referenced foreshortening corrected disparity map; (b) Right referenced foreshortening corrected disparity map.

Fig. 25 shows a representative row plot from the disparity maps of Fig. 23 and Fig. 24. In this figure, the noisy red curve is the uncorrected disparity row plot. The extent to which the noise dominates the signal is apparent. The blue curve is the row plot from the corrected disparity map. It is almost entirely free of noise, and follows the slope of the plane. The spiky parts at the ends are from the parts of the image which do not contain the plane (i.e. the black curtains on the right side of the images in Fig. 21).

The benefit derived from the geometric foreshortening correction is apparent by



(a) (b)

Fig. 25. Typical row plot from left and right referenced disparity map of Mayan blanket scene, uncorrected versus foreshortening effect corrected: (a) Left referenced disparity map row plot, typical uncorrected and foreshortening effect corrected; (b) Right referenced disparity map row plot, typical uncorrected and foreshortening effect corrected.

this example, however, it could be argued that this example is not valid because the geometry was known beforehand in order to generate the foreshortening correction factor. The reason that this argument does not have a lot of traction is because the typical application of dense stereo, and thus this entire phase based dense stereo algorithm, is as a subroutine in some kind of higher level vision based mapping or guidance system. There are many precedents for using sparse stereo to feed dense stereo, for instance Gallup et al. [38] use sparse stereo features (calculated during the motion estimation phase of the SLAM problem) to drive their plane-sweep dense stereo. Similarly, feature based stereo may be used to gain a rough idea of the geometry of the scene, and then the phase based dense stereo algorithm with the foreshortening correction factor applied may be used to fill in the details. This procedure is used in a rudimentary form in the analysis of Chapter V by using SURF features to calculate the off-normal rotation angle of a planar scene.

B. Quality Control Metrics

In the dense stereo correspondence matching process there are most certainly going to be some percentage of pixels which are incorrectly matched between images, and are thus assigned an incorrect disparity value. These pixels could be in areas which are occluded in one of the images of the stereo pair, and thus there is no correct solution to the correspondence problem. Incorrect matches could also be caused by poor surface texture in the images, be it bland surfaces, periodic textures, or non-Lambertian surfaces causing reflections. Two methods of detecting and eliminating incorrect matches are presented in this section (eliminating them means the value of these areas are set to zero in the resulting disparity maps, thus indicating that no correspondence matching was achieved).

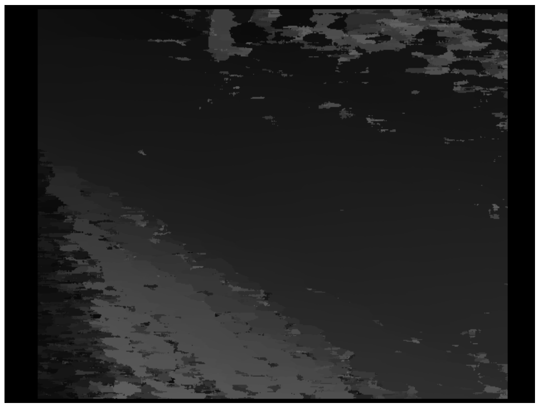
1. Left/Right Disparity Cross Checking

During development, the phase based dense stereo algorithm was coded and debugged using the left image as the search space and strips of the right image as the object of interest due to a simple mnemonic ("left=large"). This convention is not necessary, and in fact the algorithm works equally well in reverse, using the right image as the search space and the left image to build the object of interest array. This fact is what makes the post-processing cross check routine so simple and powerful. By solving the correspondence problem for the disparity value of a pixel using one convention, and then solving for the same pixel's value using the other convention, two independent measurements of the disparity value are achieved. If the disparity value proposed for the pixel by the two independent processes is the same, then it is much more likely that it is the correct value and should be trusted. This verification method was proposed early in the history of digital stereo correspondence solving (see [39] or

[18]), and has become somewhat of an industry standard method.

There is one important caveat to note on the subject of left and right disparity cross checking. Performing dense stereo from right to left (i.e. using the right image as the object of interest and the left as the search space) will generate a right-referenced disparity map. Switching the input images will generate a left referenced disparity map. In order to compare the two, they must be mapped into the same reference space. By examining the equation of disparity ($disparity = x_L - x_R$), the transformation between right and left referenced disparity maps becomes apparent ($x_L = x_R + disparity$, and $x_R = x_L - disparity$). That is to say, if the pixel of interest comes from a right referenced disparity map, and it is desired to compare it to a left referenced disparity map, comparing the pixel in the same location in each would be in error. However, by looping over the pixels in the disparity map and transforming them by the equation of disparity, a pseudo disparity image in the other reference space may be generated. This pseudo disparity map may be compared with the computed disparity map in the first reference space by performing an array-wise subtraction to find per pixel error. Using this technique, pixels in the disparity map which have obviously inconsistent values may be discarded.

The power of this error checking metric is illustrated for disparity maps from the ditch scene of Fig. 10. Fig. 26 shows raw left and right disparity maps generated by the one dimensional phase based dense stereo algorithm which are un-verified, and thus contain some obviously inconsistent correspondence matches. For example, in the background of the stereo image pair of Fig. 10 there is a pile of rubble and a lamp post, both of which have many occluded areas and high frequency discontinuities. In the corresponding locations in the disparity maps there are noisy areas which contain many mismatches. Also, in the bottom left of the left disparity map, and the bottom


(a)

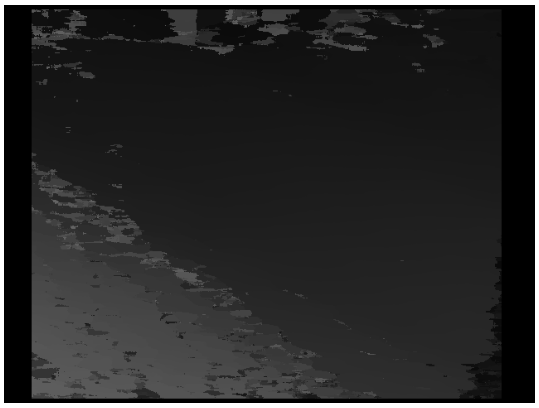

(b)

Fig. 26. Left and right referenced raw disparity maps from ditch scene: (a) Left referenced raw disparity map; (b) Right referenced raw disparity map.

right of the right, there are some pixels which did not get a strong correspondence match because the area is of high disparity and thus is very close to the edge of the other image. A third example of correspondence problems in this scene is near the left side of the disparity maps, on the downslope of the ditch, there are occluded areas which the algorithm had trouble matching, and which show up as discontinuous regions in the otherwise smooth disparity images.

Fig. 27 shows the pseudo left and right disparity maps generated by transforming the raw disparity maps of Fig. 26 by the equation of disparity to transform them to the other reference space. On the pseudo left disparity map, in the bottom left (in the same place as in the actual left disparity map) there is no disparity information, indicating that the right disparity map does not contain correspondence information for this region.

Fig. 28 shows the result of removing from the raw disparity maps the pixels for which the disparity value in the calculated disparity map does not agree with the value in the pseudo projected disparity map. The noisy erroneous values from the rubble pile and lamp post at the top are removed, as is the lower left erroneous

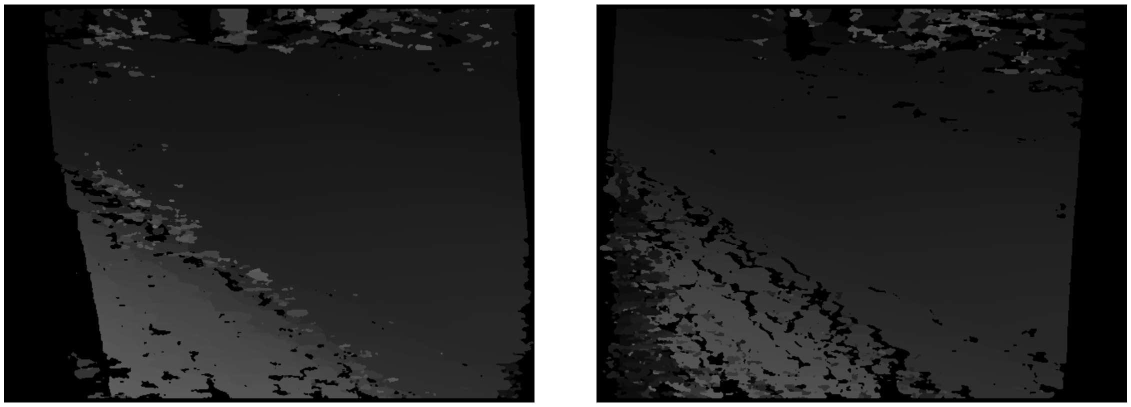


Fig. 27. Left and right referenced psuedo disparity maps, generated by transforming right and left raw disparity maps: (a) Left referenced pseudo disparity map; (b) Right referenced pseudo disparity map.

region in the left disparity map. The downslope of the ditch and its occlusions have been thinned out, and the remaining values have an overall gradient that follows the terrain, as opposed to the noisy spikes that were there in the raw disparity maps.

Thus by cross checking two independent measurements from right to left dense stereo and left to right dense stereo, correspondence matching errors caused by a variety of issues may be removed.

2. Correlation Peak Strength

During the process of correspondence matching with the phase based dense stereo algorithm, the location of the delta function peak in the spatial correlation function must determined. The process may be as simple as locating the maximum value of the function at the discrete index points and calling that the peak as described in Ch. II, or a curve may be fit the data using the analytical equation of the peak shape as described earlier in this chapter. Regardless of the method used, at the time that the location of the peak is found and recorded, it is just as easy to also

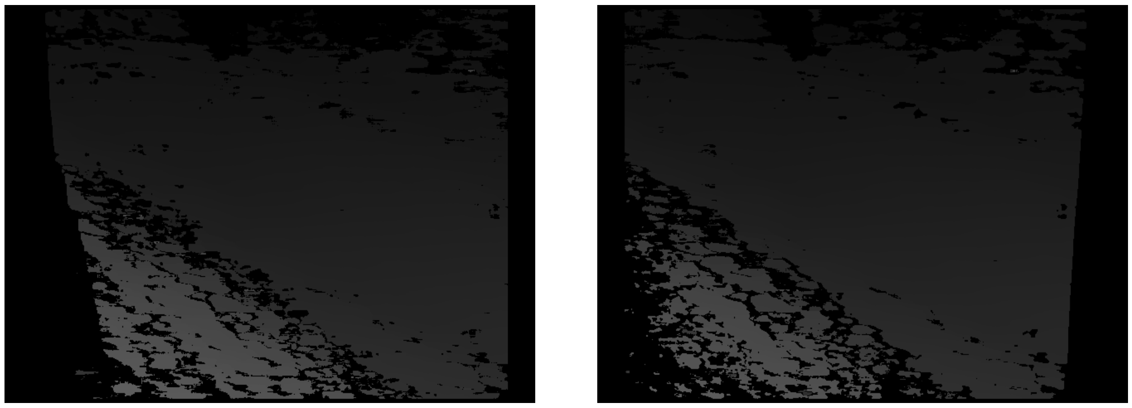

(a) (b)

Fig. 28. Left and right referenced verified disparity maps: (a) Left referenced verified disparity map; (b) Right referenced verified disparity map.

record the height of the correlation function peak. This peak height is a measure of the correspondence match certainty. By setting a minimum peak height that correspondence matches must meet to be considered valid, spurious matches may be discarded. This is essentially thresholding the correspondence matches and retaining only those which have a certainty greater than the threshold.

Mandating that correspondence matches have a given confidence before accepting them as true is not a new idea, nor is it unique to phase based dense stereo. See for instance Rzeszotarski et al. [40] in which a confidence metric is applied to the Sum of Absolute Difference (SAD) method of dense stereo and is used to threshold which correspondence matches to include in disparity maps.

This minimum peak height metric is useful, but it should be noted that it is not a golden rule, and requires (scene-dependent) tuning before it will properly discard erroneous matches. Even with tuning it is not guaranteed to properly eliminate bad correspondence matches while preserving correct matches. No guarantee can be made because a correct match may have a low peak strength (and thus a low certainty), while an incorrect match may have a very high peak strength. Consider pixels in a bland surface texture region which are just barely able to be to correctly matched, and thus have a low certainty despite being the correct solution. Compare these to pixels in a region of very strong surface texture which is also periodic. The repetitive texture would cause multiple strong correlation spikes, and in the presence of noise the wrong one could be chosen as a match, but still have a very high certainty. In this case the thresholding might eliminate the correct match and preserve the incorrect match. Obviously this is a single scenario, and is more the exception than the rule, but it is meant to illustrate that the peak strength metric (like any process) has its weaknesses. Pointing out the weaknesses of the minimum peak height metric is not intended to imply that it is worthless. In fact, quite contrarily, it is very useful either alone, or in conjunction with the left/right cross checking method described in the previous section.

The correlation peak height metric may be used in two different operation modes, either in real-time with an absolute global threshold, or as a post-processing step with a relative threshold. The absolute thresholding method checks correspondence matches as they occur, ensuring that they are greater than a pre-determined global threshold and rejecting them if they are not. The relative thresholding method waits until the entire stereo image pair has had correspondence matches solved for, and then filters out matches with a correlation peak height lower than a relative threshold (empirically, this value is usually about $0.25 * MaxPeakHeight$).

Fig. 29(b) is a right referenced correlation peak height map for the ditch scene, of which the right stereo image is shown in Fig. 29(a). Representing the information in this form is not particularly useful for analysis, it is mostly included for the sake of examination. This peak height map is a double precision image where the value for each pixel is the height of the correlation function peak that was chosen as the correct solution, the range is theoretically between zero and one (but in practice the highest correlation spikes are around 0.7).

Fig. 30(a) shows the raw right referenced disparity map for the ditch scene. There are many areas which have obviously incorrect correspondence matches that need to be removed in order to have confidence in any 3D reconstruction. Fig. 30(b) is the same right referenced disparity map for which any pixel with a correlation peak height of less than $0.25 * MaxPeakHeight$ is set to zero. The peak strength metric does a decent job of removing incorrect correspondence matches. The occluded regions on the downslope of the ditch do not have the noisy spikes that are present in the raw disparity map, and the spikiness of the rubble pile in the background is for the most part cleaned up.

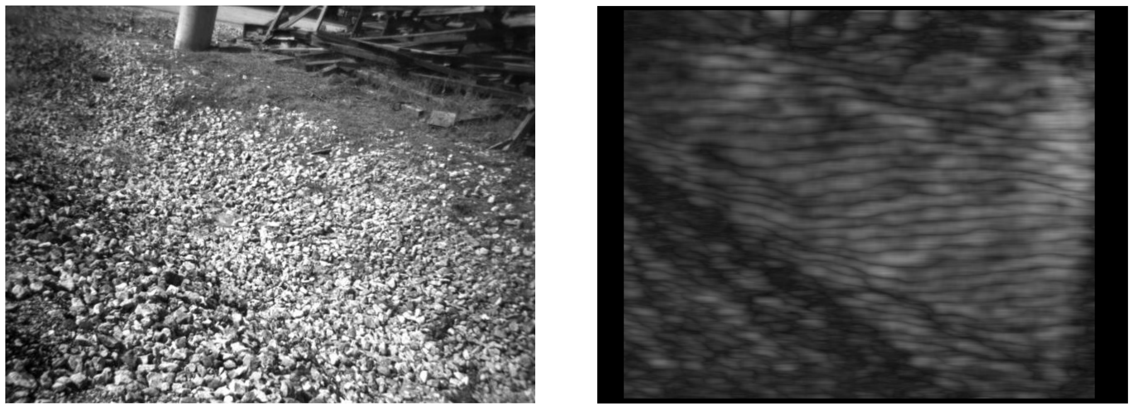

(a) (b)

Fig. 29. Ditch scene, right image of stereo pair and right referenced correlation peak height map: (a) Ditch scene, right image of stereo image pair; (b) Right referenced correlation peak height map.

It is interesting to note that, by examination of Fig. 29(b), the high values of correlation (appearing as light gray in the figure) are located where one would expect, namely fronto-parallel surfaces with strong surface texture at varied spatial

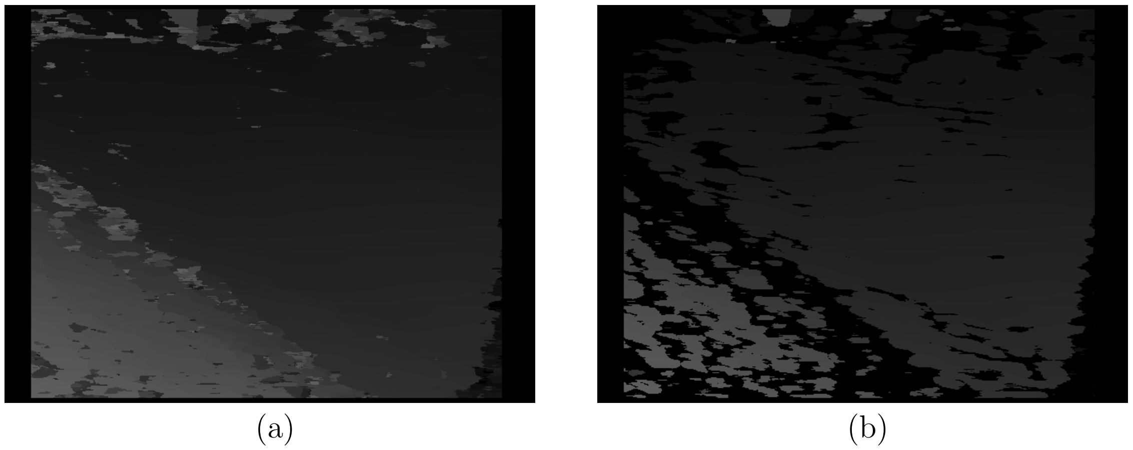


Fig. 30. Ditch scene, right referenced raw disparity map and right referenced filtered disparity map: (a) Right referenced raw disparity map; (b) Right referenced filtered disparity map.

frequencies. Weak correlation peaks are also located where one would expect, which are occluded regions, and areas with sharp discontinuities or bland surface texture.

3. Comparison of Quality Control Methods

By visual inspection both methods detailed in this section do a decent job of eliminating pixels from the disparity map which are blatantly incorrect solutions to the correspondence problem. Both methods have their pros and cons.

The pros of the left/right cross checking method are that every pixel that it leaves in the disparity map has been verified by comparing two independent measurements. Additionally, it can remove pixels which have a strong correlation value but are still incorrect, such as pixels taken from a section of periodic texture or non-Lambertian surface reflection. The downside of the method is that in order to use it, every correspondence match must be performed twice, once in a right to left stereo mode and then again in a left to right mode. This is computationally expensive, nearly doubling the computation time per stereo image pair (not exactly doubling since many aspects of the dense stereo algorithm are initialized in the same way regardless of operation mode, and thus do not need to be re-initialized for both operation modes).

The correlation peak height metric is much less computationally expensive than the left/right checking method because hardly any additional processing beyond the base phase based dense stereo is required to implement it. The downside is that it requires some scene dependent tuning in order to function properly. An appropriate threshold must be chosen such that the blatantly erroneous pixels are removed but not too many correct ones are. It is better to declare a slightly higher threshold (err on the side of caution), which will cause a higher percentage of incorrect matches to be removed, but also more correct matches will be removed.

C. 3D Reconstructions from Dense Stereo

The goal of computer vision is to allow computers and robots to sense and interact with their surroundings, thus enabling them to have a higher degree of autonomy and artificial intelligence. Stereo cameras are one type of sensor that can passively create three dimensional reconstructions of the scene being imaged without disturbing the scene or announcing their presence. A camera captures an image by saving the appearance of the projection of three dimensional object space on its two dimensional image plane. By observing the scene with two cameras that are slightly displaced from each other, it is possible to geometrically reproject the two dimensional image space coordinates, together with the disparity shift between the two images, (x, y, d) back into three dimensional object space (X, Y, Z) using Eqs. 1.3 in Ch. I.

OpenCV contains an excellent function for reprojecting from disparity space to object space which does not use the ideal equations, but instead uses a reprojection matrix Q containing camera parameters (which must be known from calibration beforehand) [19]. In this implementation the OpenCV function is used to build an algorithm that takes disparity maps as input and generates three dimensional reconstructions as an output. After reprojection to object space, the scenes are written to VRML world files (.wrl) so that the results may be inspected by humans [41]. For this application, no meshing or surface fitting is attempted, the points are simply displayed in a VRML points node, and given a color from the original stereo image pair which was used to solve the stereo correspondence problem.

Below is shown an example three dimensional reconstruction for the ditch scene of Fig. 10. This reconstruction is from the C implementation of the phase based dense stereo algorithm, using sub-pixel disparity estimation as described in Section A.2 of this chapter, and has had spurious results removed with the peak strength metric as described in Section B.2.

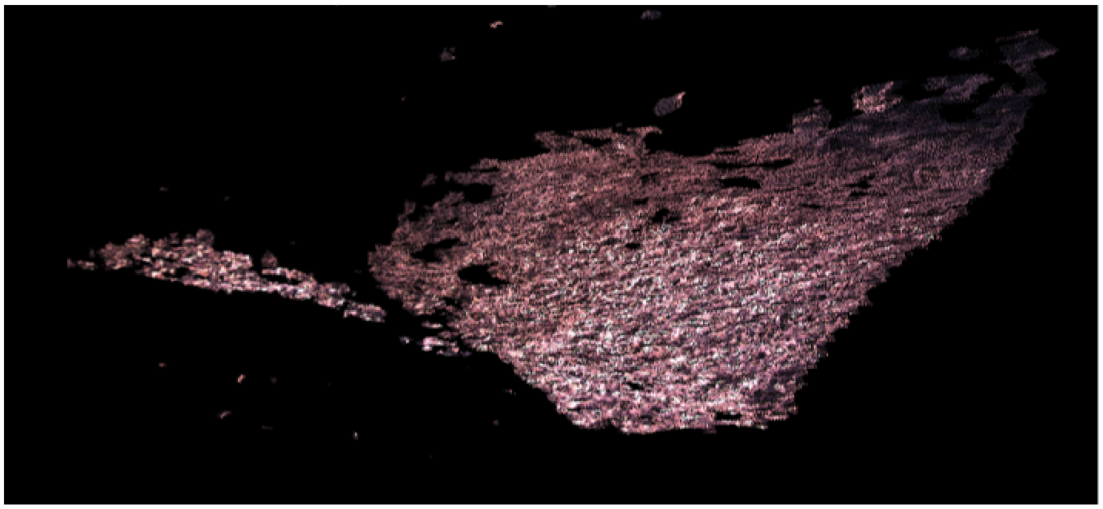


Fig. 31. Three dimensional reconstruction of ditch scene from phase based dense stereo algorithm.

Examining Fig. 31 the utility of a three dimensional reconstruction becomes clear. By looking at this image it is much easier to comprehend the geometry of the scene than by looking at the original stereo image pair in Fig. 10, or than by studying disparity space representations such as Fig. 26. The ideal way of viewing this reconstruction is by looking at the VRML file on a computer in a VRML viewer program. This way the user may rotate the scene and view it from all angles, even zoom in on details for a closer look, which is much more enlightening and intuitive than a static picture on a page.

As a practical use, this local model from one stereo image pair is saved as a VRML file which is then merged into a global model consisting of models from adjacent image pairs displayed together. In order to merge the models coherently, a tracking algorithm must be employed to estimate the camera motion with respect to the scene. Knowing the camera motion, an inverse motion may be applied to determine the orientation of one local model with respect to the rest. This is main problem statement of the SLAM problem, and the main reason for performing dense stereo.

CHAPTER IV

CUDA IMPLEMENTATION[1]

The process of implementing and optimizing an algorithm in CUDA is a non-trivial undertaking. A developer must intuitively understand the hardware layout of the Graphics Processing Unit (GPU) in order to successfully tailor their implementation to the device's strengths. Additionally, the mindset while developing an application for parallel processing is different than the traditional approach to coding an algorithm. The Land Air and Space Robotics (LASR) Laboratory at Texas A&M University has a collaboration with the research group of Dr. Jan-Michael Frahm at the University of North Carolina Chapel Hill. Under the watchful eye of Dr. Frahm and his post-doc Dr. Pierre Georgel, and graduate student Tim Johnson, strategies for CUDA development and optimization were instilled which allowed for the optimization of this particular CUDA algorithm. This chapter briefly covers some of the basic concepts of developing in CUDA, discusses some recent advances in dense stereo due to parallel computing, and describes the phase based dense stereo algorithm implemented in CUDA.

A. CUDA Background

CUDA (Compute Unified Device Architecture) is a computational architecture developed by NVIDIA to bring the parallel computational power of a modern day supercomputer to the average desktop computer. Designed to run on NVIDIA Graphics Processing Units (GPUs), CUDA is a set of C extensions and a runtime library that allows a programmer to control the launch and execution of user defined functions

[1]Portions of this chapter are in preparation for publication [29].

on the GPU (called kernels). A program developed in CUDA has standard C code running on the CPU (the host), and CUDA code running on the GPU (the device) simultaneously. Overall program execution tasks such as memory transfers to and from the CUDA device and kernel launches are controlled by the host. The CUDA API is documented excellently by NVIDIA CUDA reference manuals [42, 43]. Some basic discussion of CUDA fundamentals are developed in this section, but the interested reader should refer to the above sources.

1. Program Flow

CUDA functions (kernels) that execute on the device must be designed by the programmer in a non-traditional way. These functions act n times in parallel, where n is a configuration parameter that is defined at the time the kernel is launched. Thus, instead of designing intricate loops and repetitions to cause the program to act across the entire range of the input data, the program may be configured to act in parallel across the range of the data, for example, by having one thread acting upon each member of the input data set simultaneously. This is best illustrated with an example. Listing 1 shows a simple CUDA kernel definition, as well as the mechanism for calling said kernel.

Listing IV.1 Basic CUDA kernel which squares the elements of a vector.

```
//define kernel
__global__ void squareVectorElements( float* inVec, float*
   outVec ) {
        int j = threadIdx.x;
        outVec[j] = inVec[j] * inVec[j];
}

int main( ) {
        //blah blah blah main program
        squareVectorElements<<<1, n>>>( vec1, vec2 );
        //finish main program
}
```

There are two parts to the code shown in Listing. 1. The first part, specified by the declaration `__global__` is the kernel definition. A kernel is defined similarly to a standard function definition in C, except that one must be mindful that the code executed by the kernel will be happening simultaneously for many threads. CUDA has a built in variable called `threadIdx`, which is a three dimensional indexing system for keeping track of the threads executing in a kernel. The `threadIdx` is a unique addressing system used to access data elements and perform operations on multi-dimensional data. In this case, `threadIdx.x` ranges from 0 to $n-1$, and `threadIdx.y` and `threadIdx.z` are 0. Thus, the kernel above launchs n threads, each of which simultaneously grab one element of the vector $inVec$, square it, and save the result to the corresponding element of $outVec$.

The fact that this particular kernel is launched with n threads is no coincidence. The terms within the $<<<>>>$ brackets (in this case $<<< 1, n >>>$) tell CUDA how many blocks and threads to launch for the task of carrying out the instructions given by the kernel definition. A block (short for thread block) is a group of threads executing the same task, working from the same shared memory, on the same processor core. Thread blocks have their own unique three dimensional addressing system `blockIdx`. Thus, in this example, one block of n threads is launched for the vector element squaring kernel. All blocks launched simultaneously for a certain task have the same shape and dimension. By using this identification scheme, it is easy to specify which part of the data a block, and then within a block, a thread, is to operate upon. All the threads work separately on their own part of the data and then reconvene after the computations have been completed. CUDA provides functions for manually managing the timing and synchronization of threads and blocks such as `__synchthreads()`, which forces all the threads in a block to wait until every other one has finished before anything new can begin. Really, the miracle of working with

CUDA is that such a powerful architecture is simplified to the point that a user that is comfortable working in C can program powerful parallel algorithms with a little bit of practice.

2. Memory Structure and Kernel Optimization

In order for a CUDA kernel to be able to operate on a data structure, that structure must be contained in device memory. Device memory is physically contained on the video card, and is entirely separate from host memory which is owned by the host computer. Therefore data must be transferred from host to device before it may be manipulated by any CUDA kernel, and must subsequently be transferred back to the host before it may be manipulated by any host code. Within the class of device memory there are several different types, each optimized for its own purpose and access pattern.

The reason for the importance of the division of labor between threads and blocks as described in Section 1 is because of the scope of the different types of device memory. Threads within the same block may communicate with each other and cooperate through shared memory, whereas threads in different blocks may not. Global memory is a large repository that any thread may access at any time, but it is much slower to access because it is not located physically on the processor chip like shared memory is. Texture memory is also located off chip, but it is a cached memory bank, and thus may be accessed much more quickly than global memory. At a smaller scale, individual threads each have their own private store of memory (called local memory and registers) which are used during the lifetime of the thread.

It is important for a CUDA developer to understand the types of memory and the most efficient access patterns therein. Global memory prefers to be read or written in certain sized chunks (called words). Attempting to access the memory bank in a non-multiple of the word size, or in random scattered batches will greatly reduce the efficiency of the memory access because CUDA will treat the single access as multiple steps that must be executed in serial. The process of accessing the memory in the correct word size, and in adjacent locations is called coalescing the memory accesses. Shared memory has a common issue in that it is divided into smaller chunks called banks which are designed to be accessed in parallel. If a single bank is asked to read or write twice at the same time the situation is called a bank conflict and the access will happen in serial, thus effectively scaling the access time by the number of accesses that are called for. If the access pattern the programmer wishes to use is not well suited to global or shared memory, texture memory may be a good option. Texture memory is designed to be accessed at locations that are close together in two dimensions, (a square chunk of an image for example). Memory access conflicts may be avoided by understanding the structure of the different types of memory and designing code that effectively works around the access types. Good algorithmic performance requires that good memory practices be observed, the difference between efficient and inefficient memory use may be as drastic as an order of magnitude when translated to algorithmic run time. Techniques for optimizing a CUDA algorithm using proper memory types and access patterns, block shape, and function calls are documented by NVIDA in their "Best Practices" manual [44].

The process of optimizing a CUDA application requires a deeper understanding than the scope of this discussion. The learning curve is somewhat steep, but with patience and practice, optimization is a challenge that is possible to overcome [45]. There are many fine books, websites/blogs, and training courses available for learning and refining CUDA skills [46, 47, 48].

3. CUFFT Library

NVIDIA has designed a package of Fast Fourier Transform functions, implemented in CUDA, which are highly optimized for transforms of any size (i.e., not restricted to powers of two) and dimensionality up to three, and this CUFFT package is included in a CUDA installation. This API is designed according to the model of FFTW [49], which means that prior to taking any transforms, the user develops an FFT "plan" that configures the transform optimally from several types of available transformation algorithms. Using this same plan, an arbitrary number of transformations may be taken with little overhead. The CUFFT plans automatically configure the GPU resources and thread launch parameters for an optimal FFT or IFFT with little interference necessary by the user [50].

B. Existing Parallel Dense Stereo Algorithms

Because dense stereo is generally the most computationally expensive step of the SLAM problem, the incentive to reduce computation time by parallelizing the process is high. Also, because dense stereo involves many repetitive calculations, it is inherently massively parallel. Yang, Pollefeys, and Li's early parallel Sum of Absolute Differences (SAD) algorithm is able to achieve over 289 million disparity evaluations per second [51]. Recently, much attention has been devoted to parallelizing the process of dense stereo, including many approaches implemented in CUDA. Zhu, Butenuth, and d'Angelo compare GPU implementations of SAD and Semi-Global Matching (SGM) algorithms in CUDA with similar CPU algorithms and find comparable quality and substantially higher speed on GPU [52]. Many other parallel approaches have been implemented that promise dense stereo in real time (for instance [53, 54, 55]). A notable implementation in CUDA is documented by Stam [56], and source code and instruction is given.

C. 1-Dimensional Implementation in CUDA

The CUDA implementation of the phase based dense stereo algorithm is quite similar to the C implementation described in Section C of Chapter II. However, instead of utilizing convenient OpenCV functions as the building blocks of the algorithm, CUDA kernels had to be designed from scratch and optimized for efficiency (the only exception being the CUFFT library as described above). The result is that the core modules of the phase based dense stereo algorithm (the same core modules as described in Ch. II) are implemented in CUDA, and function efficiently enough to generate disparity map information at a rate of nearly ten frames per second for a 640 by 480 image. All the algorithmic improvements (as described in Ch. III) could be added to the CUDA implementation to improve the quality of results by the same factor as they improved the results of the C algorithm, however adding these features will increase the computational cost of the CUDA algorithm.

Fig. 32 is a flow chart of the CUDA phase based dense stereo algorithm, intended to portray a top level view of the algorithmic flow. All the processing occurs on the GPU. The stereo image pair is transfered to the device memory at the beginning of the program, and the answer (the disparity map) is transfered back to host memory when the GPU has finished its computations.

In Fig. 16 the search space and object of interest arrays are shown as utilized in the serial implementation for the disparity calculation of a single column of image pixels (these are taken from strips of the images that are then padded with zeros). In these images, each row may be treated as a separate correspondence problem and solved independently. Thus the disparity is calculated for all the image pixels in the

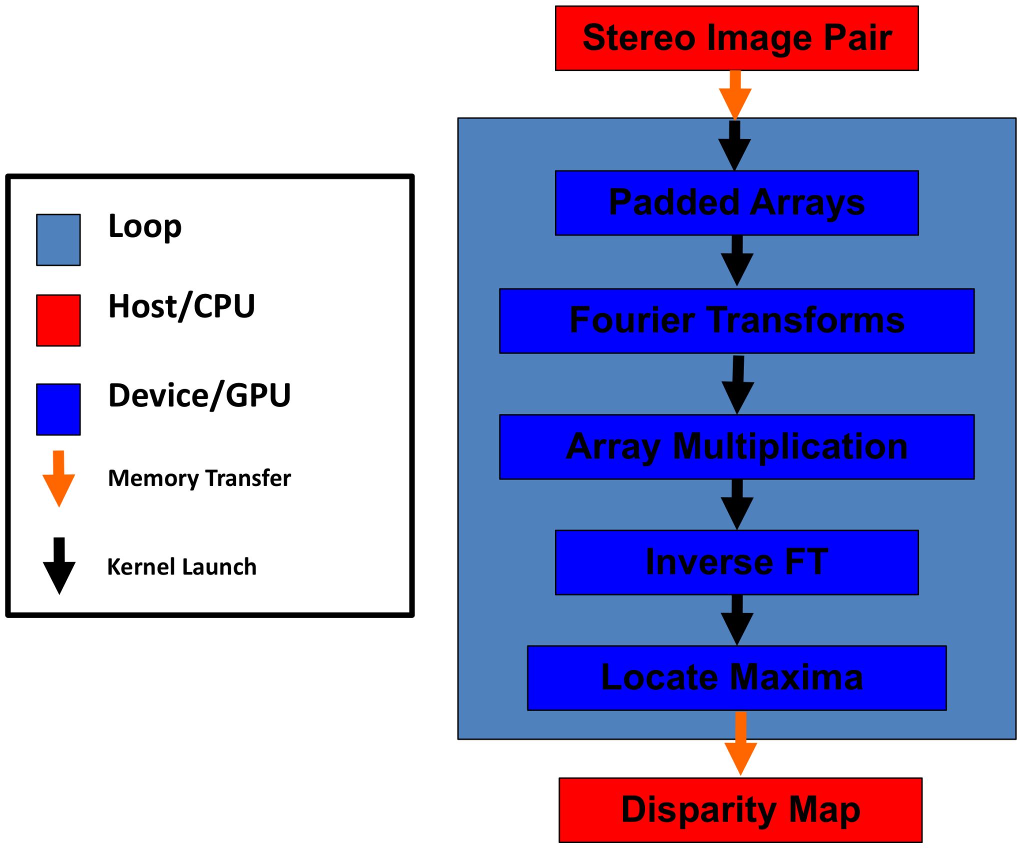


Fig. 32. Flow chart of CUDA implementation of phase based dense stereo algorithm.

column using this array, and then new arrays are formed for the next column and all the rows therein are solved independently, and so on for all the columns in the overlapping region of the stereo images. For the case of the CPU implementation, row-wise FFTs are taken of the image pairs and then elementwise multiplications are performed to find the frequency domain correlation function. Row-wise inverse FFTs are then taken of the frequency domain correlation function, to form the spatial correlation function, for which the delta function peak in each row is located. Each step is performed by a function call, whose launch carries a small amount of overhead time. Since all of the operations involved are quite repetitive, this leads to highly inefficient use of processor resources. In the CUDA version, instead of working on one column worth of pixels at a time and solving the stereo correspondence problem for each row, a large stack of these search space and object of interest arrays may be formed and solved simultaneously.

A large number (100 in the current implementation) of image arrays (like those in Fig. 16) are built and stacked for parallel processing. Fourier transforms are taken across the rows in parallel. Elementwise multiplications are performed on the arrays simultaneously, and inverse FFTs are taken across rows in parallel. The amount of memory taken up by the huge arrays of stacked image columns, and the arrays to hold the complex number results of Fourier transforms take up a large amount of memory, and thus the images must be processed in “chunks” at a time. The chunks are controlled by a main loop (indicated by the outer box in Fig. 32). The necessity for this loop is regrettable in an otherwise completely parallel algorithm as it does linearly scale the overhead time for internal memory transfers in preparation for FFTs and IFFTs by the number of repetitions of the loop, but for the current program structure it is unavoidable. The silver lining is that the “chunking” process is all done onboard the GPU to minimize host to device transfers. 100 columns worth of the image translates to roughly 20% of the overlapping region for a 640 by 480 image of an average scene, so usually only five repititions of the main loop are necessary (that is the number used in all the discussion here).

Each of the subroutine steps within the main CUDA loop in Fig. 32 corresponds to a kernel (or two in the case of "Padded Arrays"). Each CUDA kernel has a kernel definition which specifies what each thread is to do during the execution of that particular kernel. Each kernel also has a C "wrapper" function which is compiled and run on the host computer. Thus, even though all the actual computational work is done on the GPU, the kernel launches are configured and initialized on the host computer. The wrapper functions are limited to extremely trivial operations which take hardly any time to execute, such as pointer calculations and definition of thread and block configurations. The subsections below describe the kernels that accomplish each of the tasks in the flow chart of Fig. 32.

1. Initialization

Before any calculations are carried out (and before the main loop begins), memory is allocated for all the arrays that will be used for the entire process. FFT and IFFT plans are also initialized before the main loop begins, as these can be used repeatedly, and only need to be created once. Next, the left and right stereo images are transfered to the device and bound to texture memory. Texture memory allows for faster access to the data stored therein than regular global memory since it is a cached memory bank. Once the images are stored and ready, the main loop begins. As mentioned above, the size of the arrays formed for the calculations involved are the restricting factor in how many disparities may be calculated at once.

2. Padded Arrays

Giant stacks of 1D padded search space and object of interest arrays are formed by grabbing strips of images centered around certain columns and placing them within larger arrays filled with zeros to form the padded boundary regions. The locations and lengths of the strips are governed by the same equations as those of the C implementation described in Chapter II (Eqs. 2.8). The kernels `cudaPadBigKernel()` and `cudaPadLilKernel()` access the left and right image data from texture memory and construct the stacks. The process of accessing the images in texture memory requires specialized function calls called texture fetches. The overall size of the arrays formed by this texture fetching process is critical because the CUFFT package uses different FFT algorithms depending on the number of sample points within the FFT. The fastest and most efficient transform of relevant length is of length 256 (2^8) because the package is most efficient for lengths which decompose under prime factorization to a single, small number. Thus, all arrays used are of size $(100m, 256)$, where m is the vertical size of the input images.

3. Fourier Transforms

Fast Fourier Transforms are carried out in the row-wise direction of the arrays generated by the kernels described above. These transforms are governed by FFT plan structures created by the `cufftPlan1d()` function during the initialization phase. The transforms are of the type real-to-complex, and perform the transformations in parallel for the $100m$ rows.

4. Array Multiplication

The kernel `modulateAndNormalizeKernel()` performs the elementwise multiplication defined by the first term in Eq. 2.6. All of the complex numbers in this project are stored in a C structure which allows easy access to the real and imaginary part, thus complex number multiplication is easy, even when complex conjugates are involved. Simultaneous with the complex multiplication, a normalization factor is applied because the FFT algorithms in the CUFFT package do not perform a normalization.

5. Inverse FFT

Using the IFFT plan defined before the main loop began, $100m$ inverse transforms are performed in parallel across the rows of the frequency domain correlation functions generated by the kernel in the last section. The IFFTs are of the type complex-to-real, and the result in each row is the spatial correlation function for that particular pixel.

6. Locate Maxima

Algorithmically, the maxima location kernel is the most difficult to implement. The input to the function is an array of size $(100m, 256)$, where each row is an individual 1D spatial correlation function. Also input is a guess as to the minimum and maximum disparity (i.e. a region within the 1D functions in which to search for the maximum). The most efficient way to to calculate a maximum in parallel is by a process called reduction. In this implementation, one block is launched per row of the spatial correlation function array, each of these blocks containing one thread per element of the array. At the launch, the 1D correlation function for that particular row is loaded into shared memory for the block so that all the threads within the block may access it. The process of reduction consists of threads comparing their own value to another value which is located at their own location in memory plus exactly half the size of the array. Thus, for a size 8 array, the first thread compares its value with the fifth thread, the second with the sixth, and so on. In these comparison, the higher value is kept while the lower is discarded. The higher value is stored in shared memory at the location of the thread doing the comparison, and the original index location of the “winning” value is stored in a separate indices array created to keep track of the location of the larger values. This process is repeated over and over, and at each step the number of comparisons is reduced by half. Thus, for a 256 element input array, a total of eight comparison steps are needed to determine the maximum. For more information on the process of parallel reduction refer to NVIDIA’s guide on the subject [57].

The kernel `cudaFindMaxKernel()` launches $100m$ blocks simultaneously, each of them performing a reduction on one row of the array of spatial correlation functions. The returned value for each row is the location of the delta function peak, which marks the disparity value of the pixel whose correspondence problem is being solved by that row. These disparities are assembled into an image, which becomes the disparity map for the scene. Each iteration of the main loop populates another 100 image columns worth of pixels, each column containing m separate correspondence problems, where m is the height of the input image.

7. Termination

After all of the iterations of the main loop have concluded, the disparity image is fully assembled and is then transfered from device memory to the host. The transfer signifies the end of the dense stereo problem for the stereo image pair. At this point, the memory can be freed and the CUDA context can be terminated, or the arrays can be cleared and a new stereo image pair can be transfered to the device and the process can start over.

CHAPTER V

COMPARISON OF RESULTS[1]

In this chapter the quality of the disparity maps and 3D object space reconstructions generated by the phase based dense stereo algorithm are examined. Because the core components of the phase based dense stereo algorithm are implemented in CUDA, but the algorithmic improvements and quality control checks are not, the CUDA implementation is first compared to the core C and MATLAB implementation. This comparison ensures that the parallel CUDA algorithm returns the same results as the serial versions. Thus the accuracy of the serial implementations represents the overall achievable accuracy of the parallel version. From there, the full serial algorithm is compared to industry standard lab truth disparity maps. The results of two open source dense stereo algorithms from the OpenCV library are compared against the same truth maps to determine the relative quality of the phase based method. Finally, 3D object space reconstructions from the serial phase based dense stereo algorithm are compared against reconstructions from a standard Sum of Absolute Differences algorithm for a plane at various off-normal rotation angles. This experiment illustrates the effectiveness of the phase based method on highly geometrically foreshortened surfaces.

A. CUDA Implementation Results

In Fig. 33 a disparity map generated by the CUDA algorithm for a well textured and nearly fronto-parallel planar scene is compared to the disparity map for the same scene generated by the one dimensional MATLAB implementation described in

[1]Portions of this chapter are in preparation for publication [29].

Section D of chapter II. These disparity maps are generated by the core components of each algorithm, meaning none of the algorithmic improvements or error checking from Ch. III are applied to either.

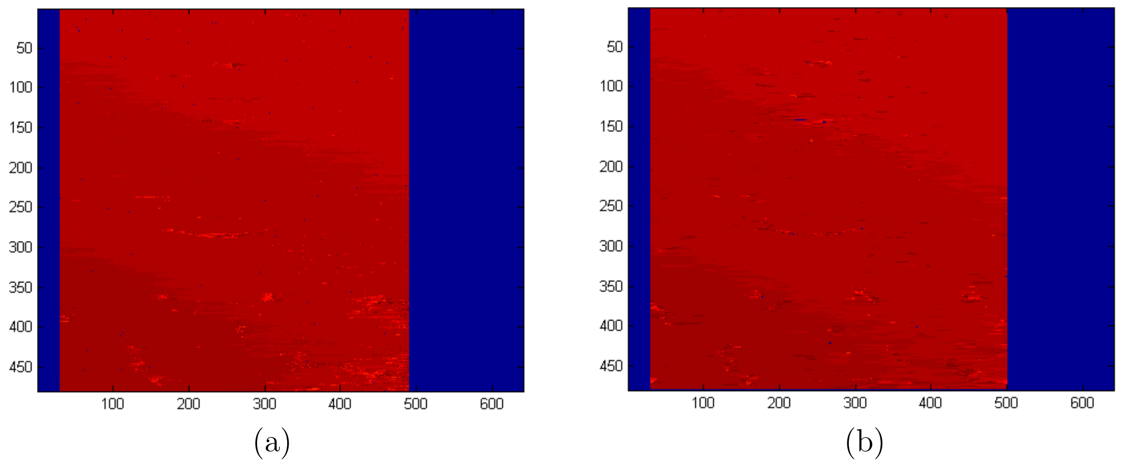


Fig. 33. Resulting disparity map, CUDA versus MATLAB: (a) Disparity map of nearly fronto-parallel plane generated by CUDA algorithm; (b) Disparity map of nearly fronto-parallel plane generated by MATLAB algorithm.

By examination it is hard to determine whether the disparity maps from the two different algorithms are actually the same. To shed more light, Fig. 34 shows a column plot of the same column from both of these disparity maps. Because the scene contains a nearly fronto-parallel plane, there are only five distinct disparity steps. The CUDA algorithm is plotted in red, and the MATLAB result in blue. The two algorithms produce exactly the same disparity value in all locations except in the transition between discrete disparity steps, although these transition regions occur in the same locations. The reason that the disparity steps are somewhat noisy in Fig. 34 (they jump back and forth in the transition regions) is because there is no averaging of rows of correlation functions to improve the signal to noise in these core implementations, nor is there sub-pixel estimation to cause the plane to appear continuous and smooth. Additionally, there are spikes of incorrect correspondence matches present for both algorithms because the error checking methods have not been used to remove bad correspondence matches.

Thus, the results of the core CUDA implementation and the core MATLAB implementation are consistent. The fact that the uncorrected versions of both algorithms produce the same results implies that implementation of the algorithmic improvements and quality control checks in CUDA will yield equally good disparity maps as those produced by the serial algorithm in MATLAB or C.

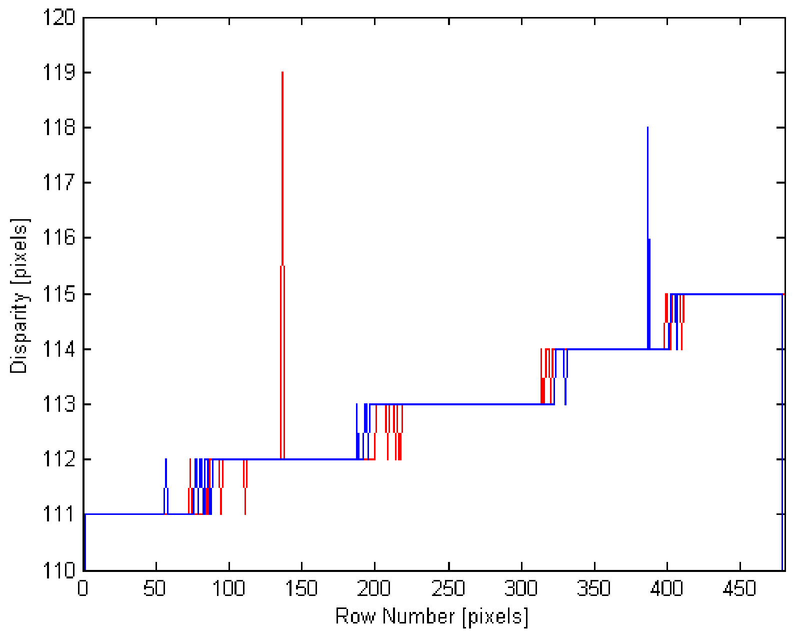


Fig. 34. Column plot comparison of CUDA disparity map and MATLAB disparity map for nearly fronto-parallel planar scene.

To better illustrate the result of the CUDA implementation for the Mayan blanket scene, a three dimensional reconstruction from the disparity map shown in Fig. 33(a) is shown in Fig. 35. The plane is rotated in this image (it is a three dimensional collection of points being displayed as VRML file) to better show the fact that it is planar. There are some noisy points caused by incorrect correspondence matches in areas of bland or repetitive texture, and the disparity values are discrete, but otherwise the reconstruction is planar.

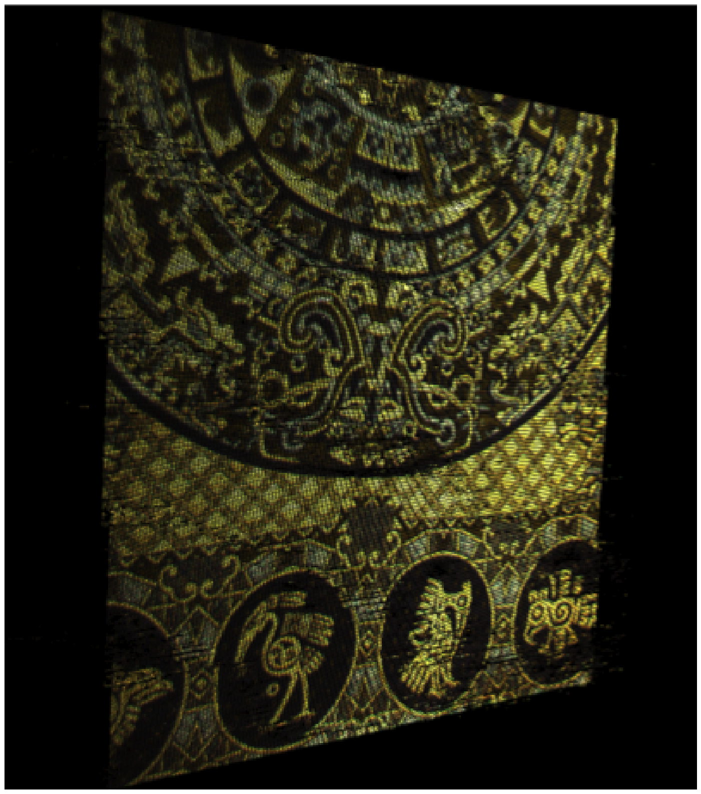
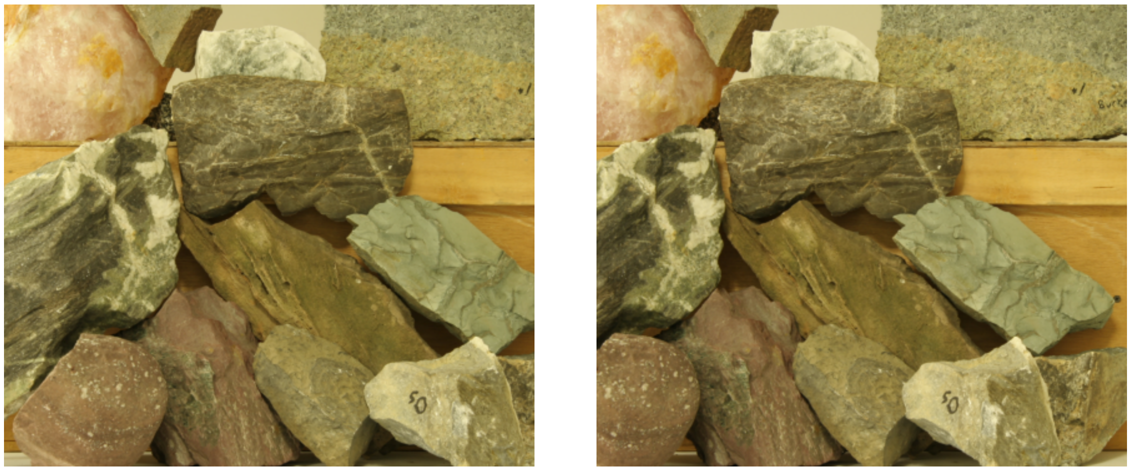


Fig. 35. Three dimensional reconstruction generated by CUDA phase based dense stereo algorithm, from nearly fronto-parallel Mayan blanket plane scene.

B. Comparison to Ground Truth Disparity Maps

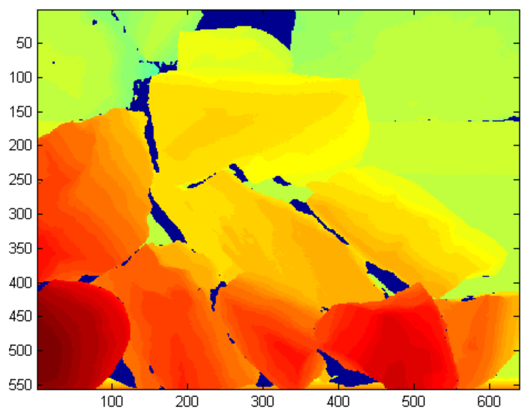
Scharstein and Pal [58], and Hirschmuller and Scharstein [59] provide stereo image pairs and ground truth disparity maps for verification of dense stereo algorithms, such as the image pair in Fig. 36. This image pair is designed to have many areas of occlusions and extreme depth variations, the vicinity of which are difficult for any dense stereo algorithm to provide accurate correspondence matches. The right referenced ground truth disparity map for the rocks scene is shown in Fig. 37(a), and will be used to test the accuracy of the disparity maps generated by the phase based dense stereo algorithm.



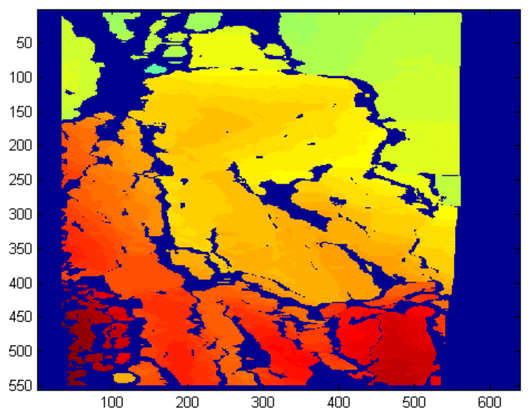
(a) (b)

Fig. 36. Left and right stereo image pair, rocks scene: (a) Left stereo image of rocks scene; (b) Right stereo image of rocks scene.

Fig. 37 is a comparison of the right-referenced ground truth disparity map for the rocks scene to the output of the serial phase based dense stereo algorithm (C implementation results shown in Fig. 37(b)). In the truth map, the occluded regions for which no disparity may be calculated have been removed. Within the phase based algorithm, the threshold for the correlation strength metric was set high to eliminate as many incorrect matches as possible, and left/right cross check verification was performed.



(a)



(b)

Fig. 37. Right referenced disparity maps from rocks scene, ground truth and phase based dense stereo algorithm output: (a) Ground truth disparity map; (b) Output of serial phase based dense stereo.

The absolute value of the difference between the phase based algorithm output disparity maps and the truth map forms an error map for the scene, as shown in Fig. 38. For better visibility the error map is converted to a binary image where the black regions designate correct correspondence matches by the algorithm and the white regions are places where the disparity did not match the truth map to within one pixel. For areas where the algorithm did not report a disparity (i.e. the error checking methods discarded the correspondence match due to a lack of confidence in the estimate), the error map is set to zero. Comparing this error map to the right stereo image, it can be seen that the main regions of error occur along depth discontinuities. This is because a finite strip length of the image must be compared

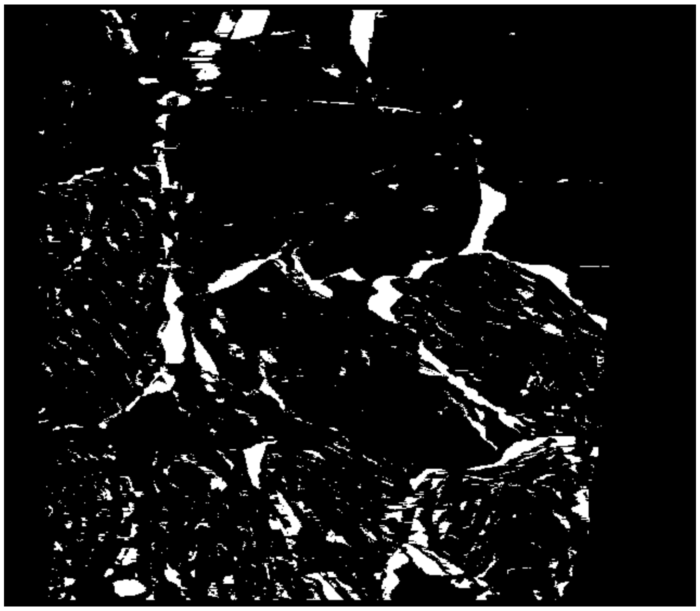


Fig. 38. Disparity error map for rocks scene compared to ground truth.

between the left and right image to find correspondence matches. When the strip contains information from both sides of a discontinuity the correspondence match is bound to be error prone. This is a general problem with dense stereo algorithms, but the effect is somewhat more prevalent in the phase based method because relatively long (usually about 40 pixel) one dimensional strips are used as the object of interest mask.

As a point of further comparison, the error maps from two open source dense stereo algorithms from the OpenCV package for the same scene are shown in Fig. 39. OpenCV contains a block matching (SAD) algorithm, and a graph cut algorithm [30]. The block matching algorithm performed correspondence matches using a 5 pixel by 5 pixel mask. The graph cut method achieves better results when it is initialized using a rough disparity map, such as the result of a sparse stereo, a trait it has in common with the phase based dense stereo. Because the phase based dense stereo in this case was not initialized with a coarse disparity map, it is not a “fair” comparison to initialize the graph cut method with one either. Thus the graph cut result shown here is from an un-initialized implementation.

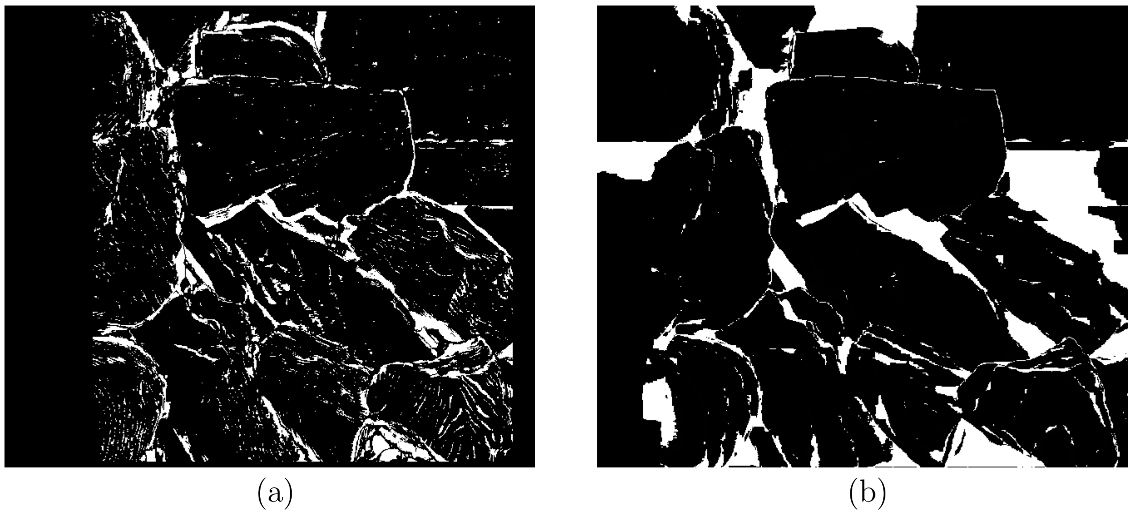


Fig. 39. OpenCV open source block matching and graph cut dense stereo algorithms error maps for Middlebury rocks scene.: (a) Error map for OpenCV block matching algorithm; (b) Error map for OpenCV graph cut algorithm.

The error regions for all three methods are similar. All three fail in the vicinity of sharp depth discontinuities or occlusions, and all three do well on well textured, non-occluded regions. Thus, for this scene, the results of the phase based dense stereo algorithm are competitive with the results from two open source, well optimized dense stereo methods.

C. Comparison to Planar Lab Truth

To demonstrate the effectiveness of the phase based dense stereo algorithm on off-normal (with respect to the focal plane of the cameras) surfaces, the algorithm is compared to a standard Sum of Absolute Differences (SAD) algorithm working on a planar surface at different off-normal angles. The planar surface is a vertical wall of the laboratory upon which a texture that is friendly to dense stereo matching is projected with a digital projector. The surface is known to be planar (to within building codes), so any variation from planarity in the results is solely due to the dense stereo algorithm. The stereo imaging system is moved in a semi-circle to image the scene from different angles ranging from approximately -35 degrees to +35 degrees of off-normality. At the extreme angles, the effect of the geometric foreshortening effect is drastic, as is illustrated by the results of the experiment.

To determine the laboratory truth, the Speeded Up Robust Features (SURF) algorithm in OpenCV is used to locate feature points in each stereo image pair [15, 19]. The matching feature points between each left and right stereo pair are reprojected into object space (X, Y, Z) and a plane is fit. By using a very high threshold for feature extraction and matching, the sparse (feature based) stereo provides a very accurate representation of the plane. Additionally, the plane that is fit may be used to calculate the θ off-normal angle needed in the geometric foreshortening factor (as given by Eq. 3.15). This is an example of a sparse stereo algorithm being used to drive a dense stereo algorithm.

Fig. 40 is a graph of the statistics of the plane fit to the SURF points. The red (lower) line is the mean absolute value of the residual error of the plane fit (in meters), and the blue (top) line is the residual error variance. A very high threshold was used in feature extraction and, depending on the angle, anywhere between 10 to 25 points were used in the plane fitting. The number varies because the ability of SURF points to successfully be matched decreases as the off-normal angle increases. The bi-peaked shape of the curves is hypothesized to be caused by the fact that the texture is being projected by a digital projector (a point light source), which causes the non-Lambertian reflection properties of the surface to affect the accuracy of the feature matches, and thus the accuracy of the three dimensional reconstruction. The asymmetry of the curve is most likely caused by the large windows in the room which provide uneven surface glare. Fig. 41 is an example of a typical plane fit to the reprojected feature points.

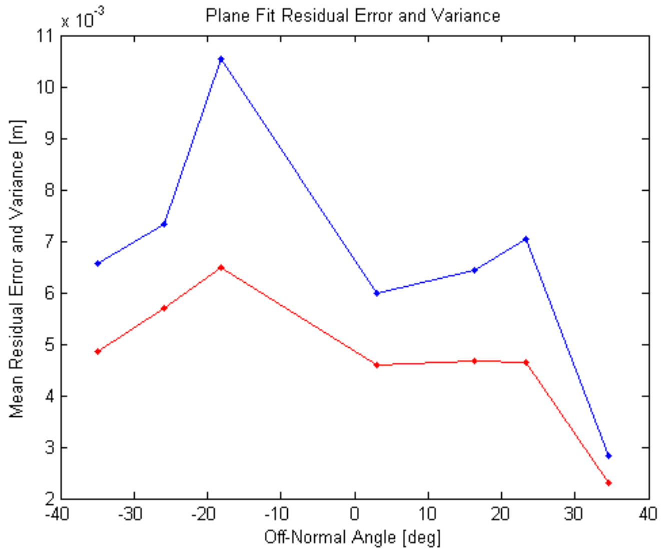


Fig. 40. Statistics for plane fit to SURF points as laboratory truth.

The C implementation of the phase based dense stereo algorithm is used to generate dense stereo reconstructions of the planar scenes. The angle θ for the geometric foreshortening factor is found from the plane fit to the SURF points. This result is compared to a simple Sum of Absolute Differences (SAD) algorithm [60]. The SAD algorithm does not have a left/right cross checking quality control function to remove incorrect matches, but a simple threshold is placed on the SAD cost function to eliminate as many spurious correspondence matches as possible. Because of the

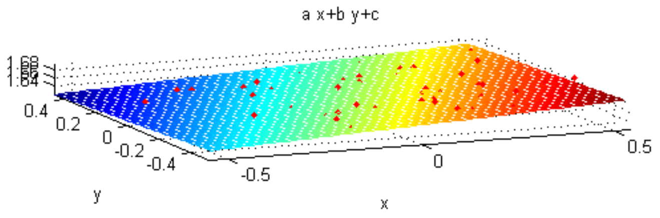


Fig. 41. Typical plane fit to SURF points as laboratory truth.

primitive nature of the error checking mechanisms, the SAD error is always higher than the phase based method's error, even for a fronto-parallel plane. Disparity is estimated as a discrete quantity for both algorithms because the SAD algorithm (this implementation of it) does not have continuous disparity estimation ability. The SAD algorithm uses a constant mask size of 11 pixels by 11 pixels, while the phased based algorithm uses a constant strip length of 40 pixels.

Fig. 42 shows the mean of the absolute value of the residual errors for each dense stereo reconstruction of the planar scenes by the SAD and the phase based algorithm, and Fig. 43 shows the variance of the residual error. Looking at residual error plot it is clear that the geometric foreshortening effect is a major factor as the plane becomes more rotated away from fronto-parallel. The residual error of the phase based algorithm with the foreshortening correction factor applied remains relatively flat, always remaining between 10 and 18 mm, whereas the error for the SAD algorithm grows rapidly as the plane becomes more rotated and thus more foreshortened. Looking at the variance it is clear that the SAD algorithm is returning noisy results with large deviations from planarity at higher off-normal angles, whereas the phase based method is returning clean, smooth results (the error checking mechanisms are discarding the incorrect noisy points). It is interesting to note that the residual error for the phase based algorithm has a bi-peaked shape, similar to to the SURF points. Again, this is most likely caused by the fact that the texture is being projected onto the surface from a point source, and thus has a reflection intensity which varies with angle.

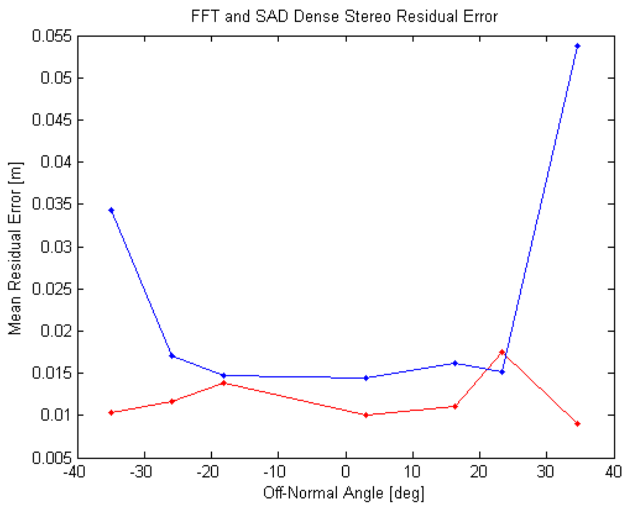


Fig. 42. Mean absolute value of residual errors for phase based dense stereo algorithm and SAD algorithm for planar scenes.

As an interesting side note, credit must be given to the OpenCV Block Matching algorithm. It is able to return extremely planar (and extremely clean) reconstructions of the scene, even at off-normal angles up to 50 degrees. This algorithm has many refinement and smoothing sub-routines, and seems extremely well suited for planar scenes such as this one. It is able to maintain residual errors as low as the SURF errors across the range of the rotations in this experiment (and beyond!). For example, Fig. 44 shows the resulting reprojection for the well textured plane rotated off-normal by 50 degrees.

Thus it is shown that the phase based dense stereo algorithm provides accurate

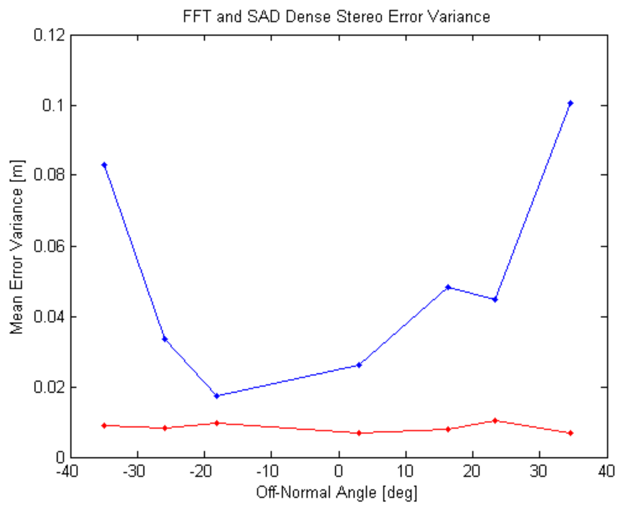


Fig. 43. Variance of residual errors for phase based dense stereo algorithm and SAD algorithm for planar scenes.

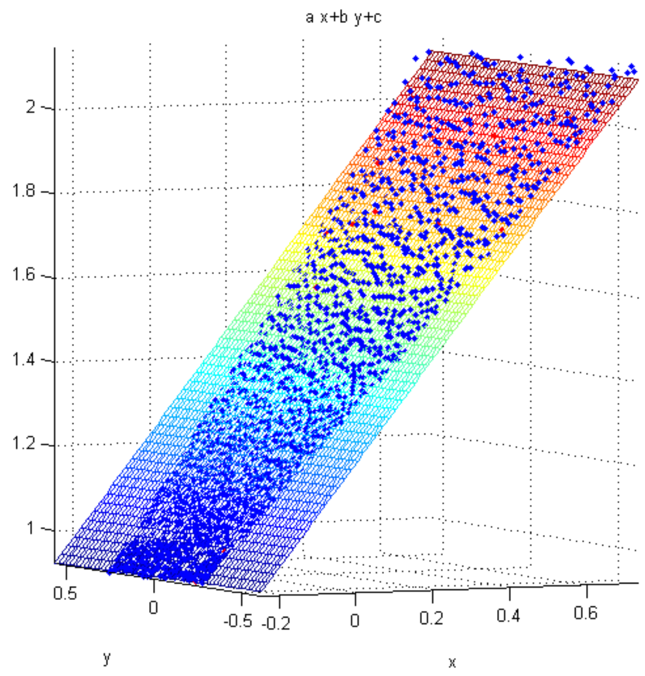


Fig. 44. OpenCV block matching algorithm 3D reconstruction of off-normal planar scene at extreme angle.

three dimensional reconstructions of non fronto-parallel scenes. The residual error and the error variance of the reprojection of the scene by the phase based dense stereo is far lower than the error and error variance by a Sum of Absolute Differences algorithm that does not correct for geometric foreshortening.

CHAPTER VI

CONCLUSION

This thesis presents a novel phase based dense stereo algorithm which is capable of creating high accuracy three dimensional geometric reconstructions of scenes using imagery from a stereo camera system. The algorithm is implemented in MATLAB and C, and the core components are implemented in the NVIDIA CUDA parallel computing environment, where they are capable of solving the dense stereo correspondence problem at a rate of 10Hz (for a 640 by 480 image pair). A series of algorithmic improvements have been developed in MATLAB and C which greatly improve the quality of the stereo disparity map for general scenes above standard phase correlation. These improvements, when migrated into the CUDA environment will provide equally drastic improvement over the core components. Error checking metrics are also developed which eliminate incorrect correspondence matches.

The C implementation of the phase based dense stereo algorithm has been compared to two open source dense stereo algorithms included in the OpenCV library by using ground truth disparity maps, and has been shown to be competitive in terms of quality. A comparison of the three dimensional object space reconstruction of the scene by the serial phase based dense stereo algorithm is shown to be superior to a comparable reconstruction by a SAD algorithm that does not correct for geometric foreshortening. The optimized parallel CUDA core version has been proved to return the same resulting disparity maps as the serial C implementation core components, but it does so at a frame rate of 10Hz. Thus, it is reasonable to expect that with the full complement of improvements and error checks implemented in CUDA, dense stereo correspondences could be solved across image pairs to create high quality three dimensional reconstructions at many frames per second.

Future work on this project naturally entails migrating the quality improvements and error checks to the CUDA environment. Accomplishing this would create a powerful and self-verifying dense stereo tool, capable of creating high quality three dimensional object space models at a rate of many frames per second. This result may be used in a variety of aerospace applications, ranging from spacecraft guidance for proximity operations to planetary rover navigation and mapping.

REFERENCES

[1] D. J. Kessler and B. G. Cour-Palais, "Collision frequency of artificial satellites: The creation of a debris belt," *J. Geophys. Res.*, vol. 83, no. A6, pp. 2637–2646, 1978.

[2] P. Eichler and A. Bade, "Strategy for the economical removal of numerous larger debris objects from earth orbits," *Acta Astronaut.*, vol. 29, no. 1, pp. 29 – 36, 1993.

[3] K. Ramohalli, "Economical in situ processing for orbital debris removal," *Acta Astronaut.*, vol. 26, no. 1, pp. 55 – 60, 1992.

[4] IAU Minor Planet Center, List of potentially hazardous asteroids, (2011, Feb.), [Online]. Available: http://minorplanetcenter.net/iau/lists/Dangerous.html.

[5] J. D. Giorgini, L. A. M. Benner, S. J. Ostro, M. C. Nolan, and M. W. Busch, "Predicting the earth encounters of (99942) apophis," *Icarus*, vol. 193, no. 1, pp. 1 – 19, 2008.

[6] V. V. Ivashkin and V. V. Smirnov, "An analysis of some methods of asteroid hazard mitigation for the earth," *Planet. and Space Sci.*, vol. 43, no. 6, pp. 821 – 825, 1995.

[7] J. N. Spitale, "Asteroid hazard mitigation using the yarkovsky effect," *Science*, vol. 296, no. 5565, pp. 77, 2002.

[8] S.B. Goldberg, M.W. Maimone, and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," in *Proc. IEEE Aerospace Conf.*, Apr. 2002, pp. 5–2025 – 5–2036.

[9] T. Huntsberger, H. Aghazarian, Y. Cheng, E.T. Baumgartner, E. Tunstel, C. Leger, A. Trebi-Ollennu, and P.S. Schenker, "Rover autonomy for long range navigation and science data acquisition on planetary surfaces," in *Proc. IEEE International Conf. on Robotics and Automation*, May 2002, vol. 3, pp. 3161 –3168.

[10] B.K. Quek, J. Ibanez-Guzman, and K.W. Lim, "Feature-based perception for autonomous unmanned navigation," in *Proc. IEEE Annual Conf. of International Electronics Society (IECON)*, Nov. 2005, pp. 1791–1796.

[11] S. Se, P. Jasiobedzki, and R. Wildes, "Stereo-vision-based 3d modeling of space structures," in *Proc. Society Photo-Optical Instrumentation Engineers Conf. (SPIE)*, May 2007, vol. 6555, pp. 65550E–1 – 65550E–13.

[12] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Conf. Alvey Vision*, 1996, pp. 147–151.

[13] W. Eric and L. Grimson, "Computational experiments with a feature based stereo algorithm," Tech. Rep. A. I. Memo 762, MIT Artificial Intelligence Laboratory, 1984.

[14] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE International Conf. on Computer Vision (ICCV)*, Sep. 1999, vol. 2, pp. 1150–1158.

[15] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comp. Vis. and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008.

[16] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *Int. J. of Comp. Vis.*, vol. 47, pp. 7–42, 2002.

[17] Middlebury stereo vision page, (2011, Feb.), [Online]. Available: vision.middlebury.edu/stereo/.

[18] P. Fua, “A parallel stereo algorithm that produces dense depth maps and preserves image features,” *Machine Vis. and App.*, vol. 6, pp. 35–49, 1993.

[19] G Bradski and A Kaehler, *Learning OpenCV*, Sebastopol, CA: O’Reilly, 2008.

[20] R. N. Bracewell, *The Fourier Transform and Its Applications*, Singapore: McGraw Hill, 2000.

[21] J. J. Weng, “Image matching using the windowed fourier phase,” *Int. J. of Comp. Vis.*, vol. 11, pp. 211–236, 1993.

[22] A. Alba and E. Arce-Santana, “Phase-correlation guided search for realtime stereo vision,” in *Proc. International Workshop on Combinatorial Image Analysis (IWCIA)*, 2009, vol. 5852 of *Lecture Notes in Computer Science*, pp. 212–223.

[23] M. A. Muquit, T. Shibahara, and T. Aoki, “A high-accuracy passive 3d measurement system using phase-based image matching,” *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E89-A, no. 3, pp. 686–697, 2006.

[24] T. Shibahara, T. Aoki, H. Nakajima, and K. Kobayashi, “A sub-pixel stereo correspondence technique based on 1d phase-only correlation,” in *Proc. IEEE International Conf. on Image Processing (ICIP)*, Sep. 2007, vol. 5, pp. 221–224.

[25] D. J. Fleet, A. D. Jepson, and M. R. M. Jenkin, “Phase-based disparity measurement,” *CVGIP: Image Understanding*, vol. 53, no. 2, pp. 198 – 210, 1991.

[26] Y. S. Kim, J. J. Lee, and Y. H. Ha, "Stereo matching algorithm based on modified wavelet decomposition process," *Pattern Recog.*, vol. 30, no. 6, pp. 929 – 952, 1997.

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

[28] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision*, New York, NY: Springer-Verlag, 2004.

[29] B. Macomber, M. Majji, and J. L. Junkins, "A phase based dense stereo algorithm implemented in cuda," *Machine Vis. and App.*, 2011.

[30] The OpenCV wiki, (2011, Feb.), [Online]. Available: http://opencv.willowgarage.com/wiki/.

[31] R. Gonzalez and R. Woods, *Digital Image Processing*, Upper Saddle River, NJ: Pearson Prentice Hall, 3rd edition, 2008.

[32] Q Tian and M. N. Huhns, "Algorithms for subpixel registration," *Comp. Vis., Graphics, and Im. Processing*, vol. 35, no. 2, pp. 220 – 233, 1986.

[33] J. L Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic Systems*, Boca Raton, FL: Chapman & Hall/ CRC, 2004.

[34] M. Majji, "Least squares methods for radial basis function approximation," Tech. Rep., LASR-110221, Texas A&M University, Land Air and Space Robotics Laboratory, Feb. 2011.

[35] M.W. Maimone and S.A. Shafer, "Modeling foreshortening in stereo vision using local spatial frequency," in *Proc. IEEE/RSJ International Conf. on Intelligent Robots and Systems*, Aug. 1995, vol. 1, pp. 519–524.

[36] S. El-Etriby, A.K. Al-Hamadi, and B. Michaelis, "Dense stereo correspondence with slanted surface using phase-based algorithm," in *Proc. IEEE International Symposium on Industrial Electronics*, June 2007, pp. 1807–1813.

[37] D. Jones and J. Malik, "Determining three-dimensional shape from orientation and spatial frequency disparities," in *Computer Vision — ECCV'92*, vol. 588 of *Lecture Notes in Computer Science*, pp. 661–669. Berlin, Germany: Springer, 1992.

[38] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys, "Real-time plane-sweeping stereo with multiple sweeping directions," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, June 2007, pp. 1–8.

[39] M. J. Hannah, "A system for digital stereo image matching," *Photogram. Engin. and Remote Sensing*, vol. 55, no. 12, pp. 1765–1770, 1989.

[40] D. Rzeszotarski, P. Skulimowski, and P. Strumiłł o, "A method for verification of dense disparity maps computed from the matching algorithm implemented in the stereovision system," in *Proc. Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments*, 2007, vol. 6937, pp. 69372R–1 – 69372R–10.

[41] C. Marrin and B. Campbell, *Teach yourself VRML 2 in 21 days*, Indianapolis, IN: Sams.net, 1997.

[42] *NVIDIA CUDA C Programming Guide, Version 3.2*, NVIDIA Corporation, Santa Clara, CA, Nov. 2010.

[43] *NVIDIA CUDA Reference Manual, Version 3.2 Beta*, NVIDIA Corporation, Santa Clara, CA, Aug. 2010.

[44] *NVIDIA CUDA C Best Practices Guide, Version 3.2,* NVIDIA Corporation, Santa Clara, CA, Aug. 2010.

[45] J.-M. Frahm, P. Georgel, T. Johnson, and B. Macomber, "Optimization of a frequency domain dense stereo algorithm in cuda," Tech. Rep., LASR-100831, Texas A&M University, Land Air and Space Robotics Laboratory, Aug. 2010.

[46] D. B. Kirk and W. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, Burlington, MA: Morgan Kaufmann, 2010.

[47] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Upper Saddle River, NJ: Addison-Wesley, 2010.

[48] Dr. Dobbs - CUDA supercomputing for the masses, (2011, Feb.), [Online]. Available: http://www.drdobbs.com/high-performance-computing/207200659.

[49] FFTW - Fastest fourier transform in the west, (2011, Feb.), [Online]. Available: http://www.fftw.org.

[50] *CUDA CUFFT Library*, NVIDIA Corporation, Santa Clara, CA, Aug. 2010.

[51] R. Yang, M. Pollefeys, and S. Li, "Improved real-time stereo on commodity graphics hardware," in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPRW)*, June 2004, vol. 2, pp. 36–41.

[52] K. Zhu, M. Butenuth, and P d'Angelo, "Comparison of dense stereo using cuda," in *Proc. ECCV Workshop 'Computer Vision on GPUs'*, Sep. 2010.

[53] W. Yu, T. Chen, F. Franchetti, and J.C. Hoe, "High performance stereo vision designed for massively data parallel platforms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 11, pp. 1509 –1519, Nov. 2010.

[54] W. Yu, T. Chen, and J.C. Hoe, "Real time stereo vision using exponential step cost aggregation on gpu," in *IEEE International Conf. on Image Processing (ICIP)*, Nov. 2009, pp. 4281–4284.

[55] S. Rogmans, J. Lu, P. Bekaert, and G. Lafruit, "Real-time stereo-based view synthesis algorithms: A unified framework and evaluation on commodity gpus," *Sig. Processing: Im. Comm.*, vol. 24, no. 1-2, pp. 49–64, 2009.

[56] J. Stam, *Stereo Imaging with CUDA*, NVIDIA Corporation, Santa Clara, CA, Jan 2008.

[57] M. Harris, *Optimizing Parallel Reduction in CUDA*, NVIDIA Corporation, Santa Clara, CA, 2008.

[58] D. Scharstein and C. Pal, "Learning conditional random fields for stereo," in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2007.

[59] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2007.

[60] M. Majji and B. Macomber, "3d image reconstruction using structured light projection and the sum of absolute differences method," Tech. Rep., LASR-091014, Texas A&M University, Land Air and Space Robotics Laboratory, Sep. 2010.

VITA

Brent David Macomber earned Bachelor of Arts degrees in physics and astrophysics from the University of California at Berkeley in December 2008. He earned his Masters degree in May 2011 at Texas A&M University under the supervision of Dr. John Junkins, and will be continuing at TAMU to earn his PhD. His research interests are in the areas of novel vision based sensors and algorithms, advanced parallel computing, and astrodynamics.

Brent may be contacted at Texas A&M University Department of Aerospace Engineering, H. R. Bright Building, Rm. 701, Ross Street - TAMU 3141, College Station, TX 77843-3141.