Cursor Stereo

Stephen Gomory, Richard Wallace

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
New York, New York 10012

Technical Report #0665

Abstract

Matching pairs of stereo images by computer is a computationally expensive and not completely reliable method for recovering depth information from a scene. However, applications exist in teleoperation and stereo cartography which require accurate matching of images for a small, arbitrarily shaped portion of the entire image, where there may be few matchable features. This work is a step toward creating dense depth maps for those applications. We propose to use a robust stereo-matching algorithm to get within a few pixels of the correct match and then refine our match to a fraction of a pixel using an adaptive-window algorithm.

As a first pass we use a multi-resolution image pyramid with a Moravec interest operator to get close to the right answer. We then apply an adaptive window algorithm due to Okutomi and Kanade. [20] [10] We present a framework for selecting subimages which unifies the one and two dimensional adaptive window algorithms they present, and test the algorithm on real images and artificial data sets. We report our observations about the workings of the adaptive algorithm.

1 Introduction

Creating a dense depth map from a pair of stereo images is useful in a variety of machine vision applications. It is often the case that correct depth is only needed in certain portions of the scene, not everywhere, so a method for accurately computing depth over a small region would be useful as it would reduce the computational cost of stereo matching.

In general it is not possible to match all regions of an image pair using passive stereo. However, there are large areas of many image pairs whose depth that can be reliably computed through passive stereo [4]. There are points which can be readily matched and others whose depth can only be estimated by modeling the surface and lighting conditions and interpolating. For a variety of reasons it is not possible to match a single pixel in one image with its counterpart in another image. Therefore some amount of context around the point to be matched during initial correlation, refinement, or interpolation, is needed. This context constitutes a window around the point to matched. One open question is what should be the optimal size and shape for the window? Although the accuracy of the lighting model, surface model, and other factors are important factors in determining the accuracy of the resulting depth map, window size and shape are also critical. For image areas with distinctive visual features, a relatively small amount of surrounding image is needed to find a match. For image areas with a small amount of visual interest a larger amount of surrounding context is needed. The accuracy of the computed depth will then depend on how similar the surrounding context is to the image area we are trying to match.

Kanade and Okutomi have proposed a method for computing optimal window size during the registration phase of stereo matching. [20] [10] That is, once an approximate match has been made, it can be refined, or registered, by looking at local image properties. They present a method for selecting the optimal rectangular windows for doing this on 2D images, and a method for selecting optimal windows for 1D signals which are comparable to a window on a single scanline from a 2D image. The amount of computing required to determine the optimal window size over an entire image is prohibitive. We unified these two window selection methods into one, which selects the optimal window constrained to lie within an arbitrary portion of a 2D image. One feature of this unified method is that it may be used when only a small part of the image is of interest. Although the method described below is not fast enough to be run interactively as currently implemented (even for a very small sub-image) there are clear steps that could be taken to greatly improve its speed.

Although the theory put forth by Kanade and Okutomi is appealing in that it provides a rational method for balancing the competing factors in window size selection, the algorithm had only been tested on a small number of real and artificial images. We tested the window selection algorithm on several more real and artificial data sets and found (to our suprise) that the adaptive window method did not always work better than the fixed window method. In the course of these investigations, we also discovered that the initial position of the window with respect the the feature to be matched was a significant factor in determining the “optimal” window size selected, and hence the accuracy of the resulting depth map. More generally, we found that the method by which the search space for the optimal window was pruned had an important effect on the outcome of the search. As a result of this, we adopted a window selection method which uses fixed windows and adaptive windows seeded at different starting positions. This search method proved more reliable than using adaptive or fixed windows exclusively.

2 Depth From Disparity

Stereo matching algorithms take two or more images of a scene as input and produce a depth map of the scene. The depth map may be sparse in which case the depth of only a few visible locations is computed, or it may be dense in which case a depth is computed for nearly every visible location. The basic procedure is to locate the projection of some scene portion onto each of the two images. Having found a corresponding pair of projections, we can use the difference in their locations in the two images (termed disparity or parallax), in conjunction with a camera model, to compute the depth of the point. Finding the pair of matching projections is called the correspondence problem and is considered the hard part of the problem.

The task of reconstructing scene depth from disparity is generally considered to be the easy part of the problem. However, to do this accurately, one must have a good camera model and know the relative position of one camera with respect to the other. Building accurate camera models is not trivial, but good methods do exist [30]. Since the certainty of the depth measurement decreases as depth increases, it is often useful to quantify the certainty of the computed depth, for which methods also exist [15] [1]. Extracting a depth value from the geometry of a given camera configuration is called triangulation, and the solution to the general case is presented in several places in the literature. [25]. The problem gets more difficult in situations where the relative camera configuration is not known ahead of time, but good solutions for this also exist.[31]

We will not focus on this aspect of the problem. Instead we will assume that we have perfect pinhole cameras, aligned along parallel axes, separated by a known "interpupilary distance," or "baseline" along the horizontal (x) axis. These assumptions simplify the notation and are not too restrictive because images obtained from a more complex system can always be made to conform to these assumptions by warping or reprojecting the original images. We will use this system model later to demonstrate the basic properties of the disparity function. Note, however, that with a pinhole model, a function in disparity is also a function in depth, but this is not necessarily the case with a more complex system, where lens distortions are considered.[28]

Given this camera configuration and camera model, we note that the correct match for a given pixel is always on the same scanline in the other image. That is, the shift is always in the horizontal, or "x" direction. The pinhole model allows us to use the perspective projection equations to compute the depth. Several coordinate frames are used in the literature and it is useful to be able to translate from one to the other. We refer to them as l_1, r_1, l_2, r_2, and s. These coordinate frames are illustrated in the accompanying diagram and their transformation matricies are given below.

$$ {}^{b}_{a}M \, {}^{a}v = {}^{b}v $$

$$ {}^{s}_{\ell 1}M = \begin{bmatrix} 1 & 0 & 0 & s/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & f \\ 0 & 0 & 0 & 1 \end{bmatrix} {}^{\ell 1}_{s}M = \begin{bmatrix} 1 & 0 & 0 & -s/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -f \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{s}_{r1}M = \begin{bmatrix} 1 & 0 & 0 & -s/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & f \\ 0 & 0 & 0 & 1 \end{bmatrix} {}^{r1}_{s}M = \begin{bmatrix} 1 & 0 & 0 & s/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -f \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

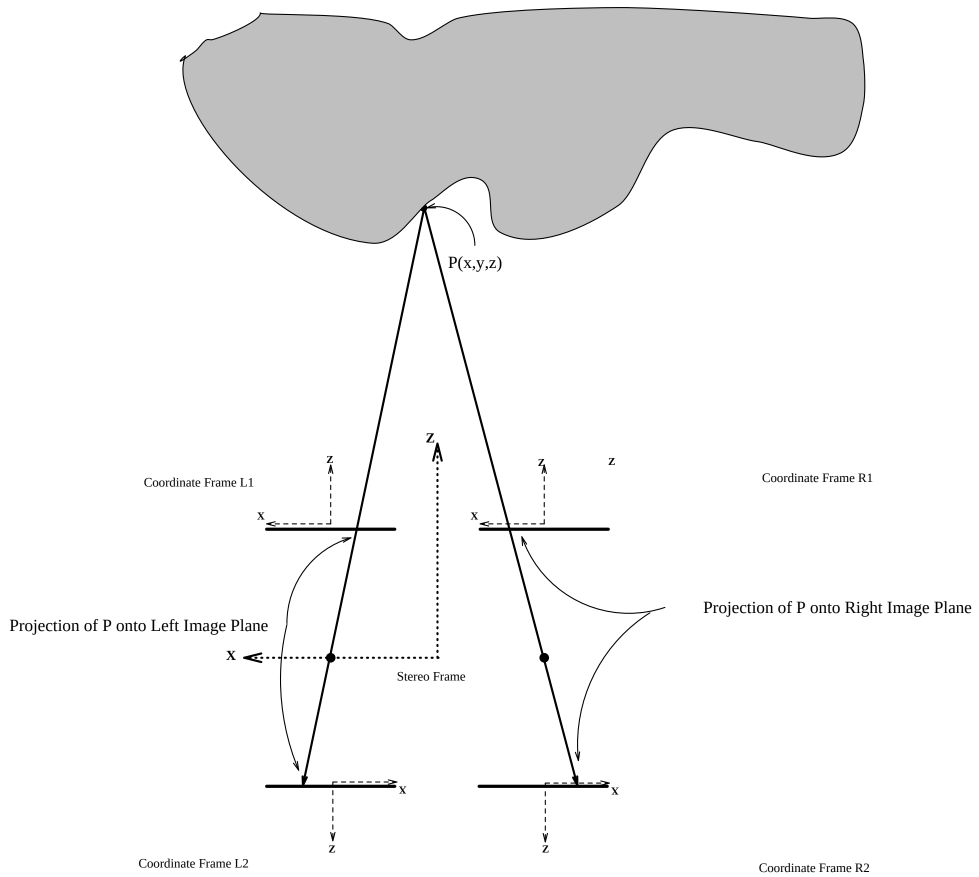


Figure 1: The five standard coordinate frames which appear in the literature.

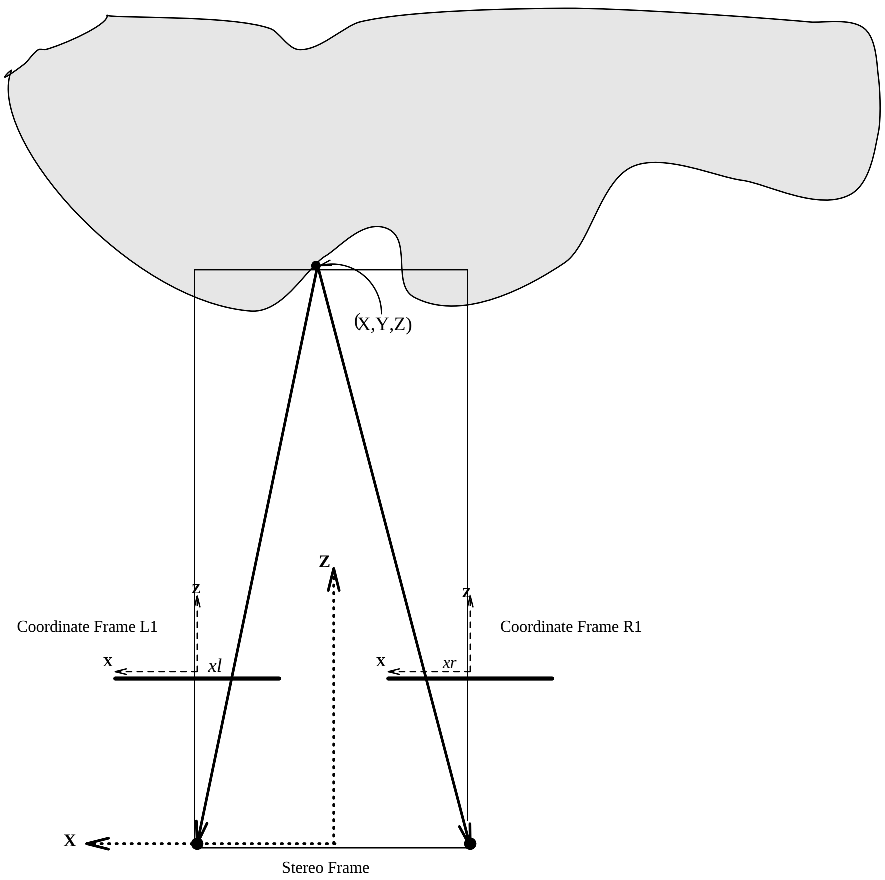


Figure 2: Depth can be computed from disparity by using similar triangles.

$$
{}^{r2}_{s}M = \begin{bmatrix} -1 & 0 & 0 & -s/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -f \\ 0 & 0 & 0 & 1 \end{bmatrix}
{}^{s}_{r2}M = \begin{bmatrix} -1 & 0 & 0 & -s/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -f \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{\ell 2}_{s}M = \begin{bmatrix} -1 & 0 & 0 & s/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -f \\ 0 & 0 & 0 & 1 \end{bmatrix}
{}^{s}_{\ell 2}M = \begin{bmatrix} -1 & 0 & 0 & s/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -f \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

let

$$
{}^{s}P = ({}^{s}X, {}^{s}Y, {}^{s}Z)
$$

$$ {}^{\ell 1}P = \left({}^{\ell 1}X, {}^{\ell 1}Y, {}^{\ell 1}Z\right) $$

$$ {}^{r1}P = \left({}^{r1}X, {}^{r1}Y, {}^{r1}Z\right) $$

and the projection of P onto the left and right imaging areas be

$$ \left({}^{\ell 1}x_l, {}^{\ell 1}y_l, 0\right) $$

and

$$ \left({}^{r1}x_r, {}^{r1}y_r, 0\right) $$

respectively. The focal length vector ${}^{s}\hat{f} = (0, 0, f)$ where $f < 0$. The stereo baseline or camera separation is denoted by s.

We then have

$$ {}^{\ell 1}X / {}^{s}Z = {}^{\ell 1}x / f $$

and

$$ {}^{r1}X / {}^{s}Z = {}^{r1}x / f $$

using

$$ {}^{\ell 1}_{s}M\, {}^{s}P = {}^{\ell 1}P $$

$$ {}^{r1}_{s}M\, {}^{s}P = {}^{r1}P $$

we obtain

$$ \frac{{}^{s}X - s/2}{{}^{s}Z} = {}^{\ell 1}x_l / f $$

$$ \frac{{}^{s}X + s/2}{{}^{s}Z} = {}^{r1}x_r / f $$

multiplying through by f, we get

$$ f * \frac{{}^{s}X - s/2}{{}^{s}Z} = {}^{\ell 1}x_l $$

$$ f * \frac{{}^{s}X + s/2}{{}^{s}Z} = {}^{r1}x_r $$

letting d denote disparity, we have

$$ d = {}^{r1}x_r - {}^{\ell 1}x_l $$

$$ d = sf / {}^{s}Z $$

so

$$^{s}Z = sf/d$$

If we apply this reasoning to the Y coordinate of the scene point, we find that $^{\ell 1}y_l = {}^{r1}y_r$ from which no depth information can be inferred.

Note that disparity, d is inversely proportional to distance implying that when disparity is small (depth is large) small errors in disparity may produce very large errors in depth. (Hence the need for sub-pixel accuracy in disparity measurements.) The error can be reduced by increasing the stereo base line, but as we will see later, this tends to make the correspondence part of the problem more difficult.

2.1 Photometrics and Correlation Measures

If images of real scenes taken from two different cameras were displaced versions of one another, the correspondence problem would be greatly simplified. However, this is not the case, so most matching algorithms assume that the two images appear similar but not identical. To find corresponding pairs of points, they then use a correlation measure that can identify similar (but not necessarily identical) features. The choice of correlation measure implies a model of the similarity of the two projections. Below we summarize the types of dissimilarites between images that are reported in the literature and then relate them to the common correlation methods.

2.1.1 Image Acquisition

In general no two cameras are identical, so given two identical inputs they will produce different output signals. If cameras A and B produce signals $A(x,y)$ and $B(x,y)$ we can approximate the difference between the two signals as $A(x,y) = gB(x,y) + h$, where g is the "gain" and h the "offset." We can solve for g and h experimentally by putting the cameras very close together and pointing them at a distant scene, thereby giving the cameras nearly identical inputs. If we compute the mean and variance of each signal we find

$$h = g\overline{B} - \overline{A}$$

$$g = \frac{\sigma_B^2}{\sigma_A^2}$$

where the mean, $\overline{A}$, and variance, σ_A^2, have their standard definitions:

$$\overline{A} = 1/ \mid A \mid \sum_{i,j \in A} A(i,j)$$

and

$$\sigma_A^2 = \sum_{i,j \in A} (A(i,j) - \overline{A})^2$$

So we can correct for these camera distortions by multiplying signal B by g and adding h to it, which normalizes the two signals.

Radial and tangential lens distortion will generally be different for the two cameras. Also the effective focal length of the two cameras may be different either due to the lens or focus. We hope to adequately account for these effects through the camera model, but the model is always something less than perfect. As well as presenting a new algorithm for computing the camera model, Tsai reviews the accuracy of the standard techniques. [30]

Noise in the electronics, such as "scanline jitter"[16], and voltage variation [18] can cause image differences. This type of difference is generally considered to be Gaussian. [32] [24]

Another source of noise is quantization error [23] in the imaging system. In the real world, light intensity is real valued, but in the imaging system intensity is quantized. In an 8 bit image, quantization will cause error in the range of 0 to 1/2 part in 512, or roughly 1/10 of one percent. In a 24 bit image, the quantization error (analog to digital conversion) is negligible. Another factor that can cause significant error is the digital to analog conversion. Cameras generally form and store images digitally. They then do a digital to analog conversion to transmit the image. The information loss at this point can be significant depending on the type of analog signal used. "RGB" cameras, which convert the digital image into three separate NTSC signals, which produce much better results than standard video cameras which convert information into a single NTSC signal. The standard NTSC signal does a fine job encoding the "black and white" signal for which it was originally intended, but the color information is not preserved well since only a small portion of the signal is reserved it. The result is that an edge detection done separately in red, green, and blue, from an NTSC output camera may position the edge in different places. An RGB camera can also produce this kind of shift, owing to the fact that the red, green, and blue sensors cannot be exactly coincident.

2.1.2 Camera Position Based Differences

Another important class of differences in the two signals results from the fact that the cameras are in two different locations. One of these differences is disparity, which is what we wish to recover. To solve the correlation problem we want to ignore all differences in the signals except for disparity.

To accurately model the light which reflects from a scene to the camera, we should model multiple light sources and inter-reflections, surface roughness and reflectance, attenuation effects, and other factors. To reduce the complexity, we make use of the simple Phong model in the following discussion.

I_t	Total amount of incident light
I_a	Intensity of incident ambient light
I_p	Intensity of a point light source
k_a	Ambient reflection constant
k_d	Diffuse reflection constant
k_s	Specular reflection constant
I_r	Intensity of the light reflected to the camera
N	Unit vector normal to the surface
L	Unit vector in the direction of the light source
R	Unit vector in the direction of maximal reflection
V_1	Unit vector in the direction of camera 1
V_2	Unit vector in the direction of camera 2
n	Constant determining the distribution of specular reflections

The light present in the scene is divided into ambient light and that due to a particular light source.

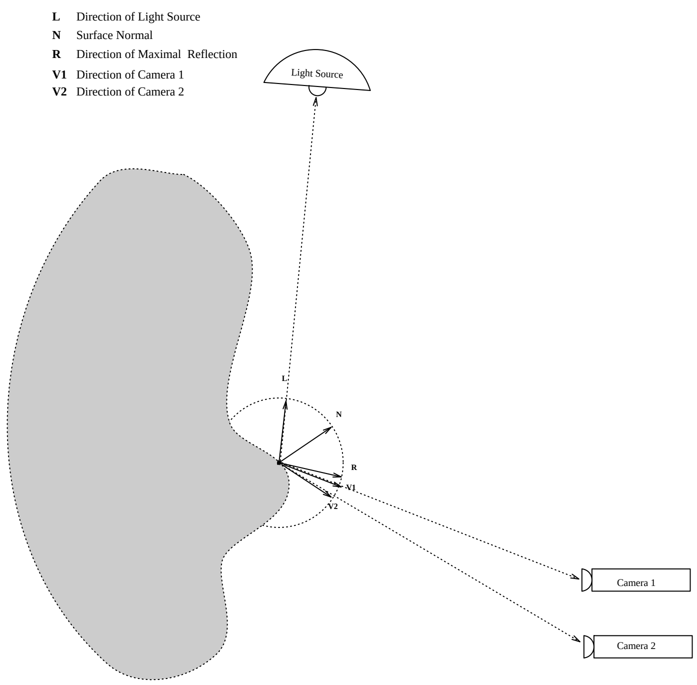
$$I_t = I_a + I_p$$

Dividing I_p into a diffuse component and a specular component, and letting k denote the reflectiveness of the surface. Following Foley et. al [6] we have:

$$I_r = I_a k_a + I_p k_d (< N, L > + k_s < R, V >^n)$$

In this model the ambient and diffuse light reaching the cameras are independent of camera position, but the specular light is not.

We are interested in how the same scene point appears from two different camera locations.



$$I_{r1} = I_a k_a + I_p k_d (< N, L > + k_s < R, V_1 >^n)$$

$$I_{r2} = I_a k_a + I_p k_d (< N, L > + k_s < R, V_2 >^n)$$

Letting $\Delta V = V_2 - V_1$ and $\Delta I_r = I_{r2} - I_{r1}$ noting that ΔV is the constant, interpupilary distance, we have:

$$\Delta I_r = k_s I_p (< R, V_1 > + < R, \Delta V >)^n - < R, V_1 >^n$$

In general the same point in the scene will appear different to each camera because $k_s < R, \Delta V > \neq 0$. ΔI is 0 for a pure lambertion surface, for which k_s is 0. This condition also occurs if it so happens that $R \perp \Delta V$. In these cases the cameras see the same brightness regardless of their position, which is desirable from the point of view of matching. If the point light source is infinitely far away, (as with sunlight) and the scene is planar, $< R, \Delta V >$ is a constant across the image. In this case corresponding points in I_{r1} and I_{r2} will differ by a simple constant, but in general the appearance of the scene in the two views will differ by a more complex term.

Consider a single nearby light source and two points in a scene X_1 and X_2. The difference between the recovered light intensity for the two points is:

$$\Delta I_r = I_a(k_a 2 - k_a 1) + I_p(k_{d_2} < N_2, L > - k_{d_1} < N_1, L > + k_{s_2} < R_2, V_2 >^n - k_{s_1} < R_1, V_1 >^n)$$

We can rewrite this expression as two terms:

$$\Delta I_r = I_a(k_{a2} - k_{a1}) + T$$

The first term depends purely on the properties of the two scene points and the second term T depends on the properties of the two scene points as well as other factors. Clearly this expression can be minimal for points other than $X_1 = X_2$. This creates the potential for mis-matching pairs of points during correlation. Surface roughness can also create mis-matches as the appearance of a textured scene is dependent on the orientation of the cameras and the light source.

2.1.3 Disparity Functions

Let us assume for the time being that the two images appear identical in each camera except that one image is a shifted version of the other.

Consider a signal B that is created by taking A and shifting it various amounts. We can write $B(x + d(x,y), y) = A(x,y)$ where $d(x,y)$ is the disparity, or "shift" of the signal at x, y. Note that $d'(x,y)$ denotes the partial derivative of d with respect to x. Unless stated otherwise $d(x,y)$ is a function of the left signal, A, so $d(x,y) \geq 0$ always holds. It is potentially useful to think of the disparity function in terms of signal B, in which case $d(x,y) \leq 0$ always holds.

When $d(x,y)$ is a constant function ($d'(x,y) = 0$), the second signal is simply a displaced version of the first. This signal would be generated from a pair of cameras viewing a plane that is square to the cameras, (ie. whose normal is parallel to the camera axes). 2.1.3

When $d'(x,y) < 0$ and $d''(x,y)$ is a constant, the second signal will be an expanded and displaced version of the first. When $d'(x,y) > 0$ and $d''(x,y)$ is a constant the second signal is a compressed and displaced version of the first. Since disparity is inversely proportional to scene depth, these disparity functions correspond to real scenes where the cameras view a plane whose

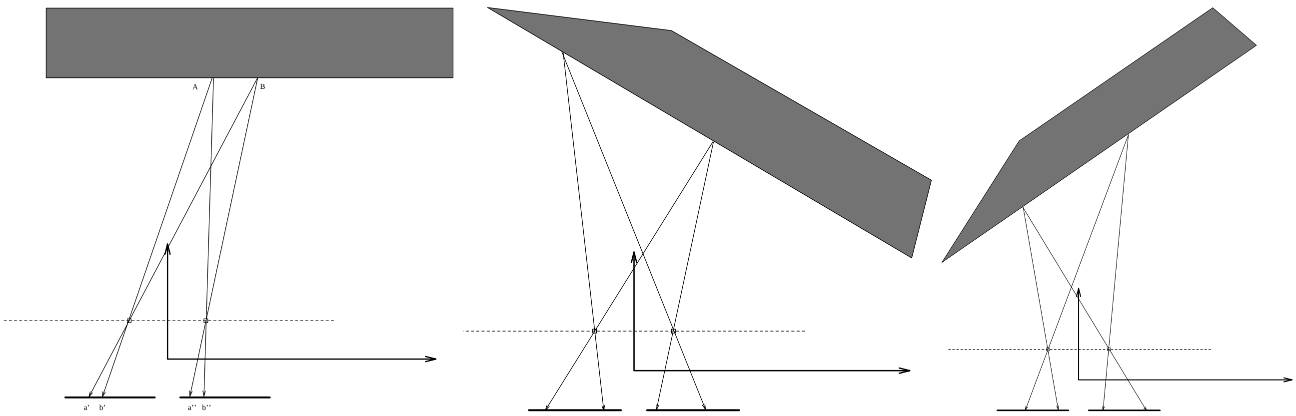


Figure 3: When viewing a plane which is at an angle to the camera axes, the projection of the scene area between A and B is a different size in the two images.

normal is not parallel to the camera axes. Note that the gradient of the disparity map in such cases will be different if measured with respect to A or B. Note also that projections of such objects will be different sizes on the two image planes; so there is no single window size which will just cover the projections in both image planes.

When $d''(x, y)$ is non-constant, it is possible for the second signal to be a "folded version" of the first. That is, more than one value of A is mapped to the same location in B.

let x_1, x_2, x_3, y index the image A let

$$x_1 < x_2 < x_3$$

Given some y, we have a "forward mapping" if

$$x_1 + d(x_1, y) < x_2 + d(x_2, y)$$

And we have a "backward mapping" if

$$x_2 + d(x_2, y) > x_3 + d(x_3, y)$$

Rearranging the terms in the above two expressions we have:

$$\frac{d(x_2, y) - d(x_1, y)}{x_2 - x_1} > -1$$

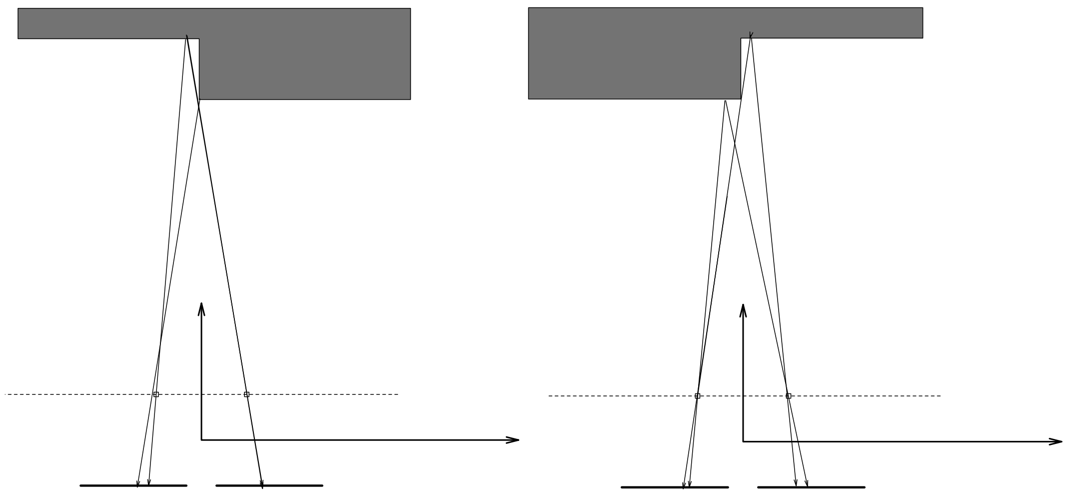


Figure 4: Right and left occlusions.

$$\frac{d(x_3, y) - d(x_2, y)}{x_3 - x_2} < -1$$

Taking the limit of these expressions we find that

$$d'(x, y) = -1$$

at a fold.

Occlusion is an unfriendly form of the disparity function. If the scene consists of one continuous surface, and a fold has occurred in creating B from A then there will be points in A which do not appear in B. For such surfaces, only the nearest scene point of the many mappings will be visible, so $d'(x, y) \geq -1$ will always hold. If it is not a continuous surface or is semi-transparent then backward mappings will exist. So image patches will appear in a different order in both images.

3 Stereo Matching Algorithms

In general, to find the corresponding point in another image, an exhaustive search must be performed. Due to the epipolar constraint, this search is limited to a search in one dimension. Area-based methods often use knowledge of the depth range of the scene, and multi-resolution images to further limit

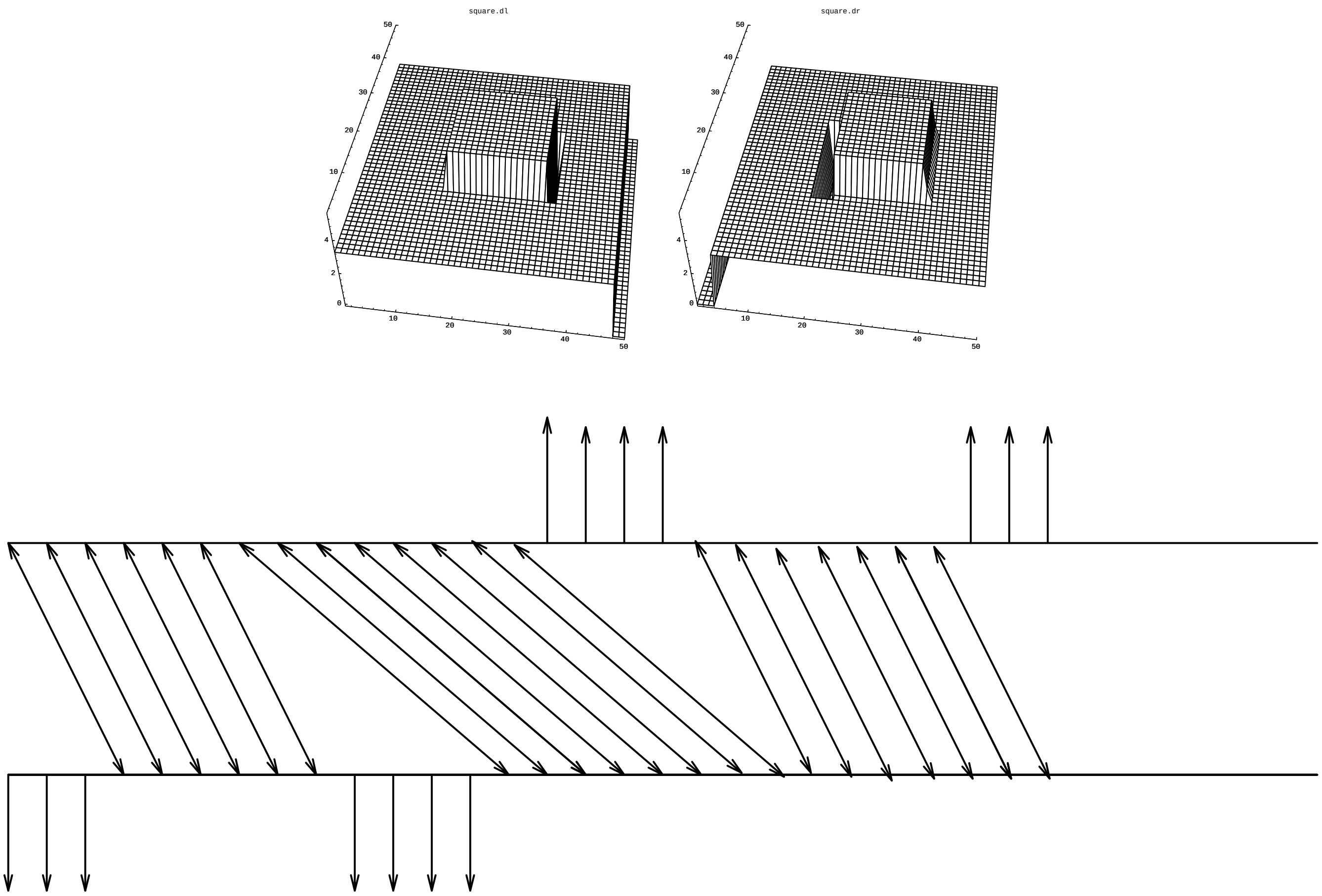


Figure 5: Left and right disparity maps for a square disparity pattern. Undefined disparities are set to 0. Below, the mapping for a single scanline. The arrows represent the mapping between the same scanline in the left and right image image. Occluded areas (single arrows) do not map from one image to the other. Double arrows match in both directions and so are part of an onto mapping.

the end points of the search space. Edge based methods often impose an ordering constraint to restrict their search. This information reduces the search space, which makes the algorithm run faster and also reduces the chance of finding a false match. If more than two cameras are used, the redundancy can be used to help overcome adverse conditions such as imaging noise, false matches, and occlusion. [21] [18] In general it is not enough to find the global minimum of a correlation measure to find the right match. We tested this using the sum of squared differences measure (SSD) [1] and found that even after thresholding out poor correlation scores, the resulting depth map was quite noisy. Some surface model is needed in order to remove false matches with high correlation scores.

Correlation measures assume that a given scene location looks similar to each camera. Each measure has its own definition of what similar is. We should expect these methods to work well as long as their assumptions are not violated by the scene. No correlation measure will do well in scenes which contain occlusions and specularities because these scenes violate the basic assumptions behind all correlation measures. However, it is sometimes possible to compensate for these effects in other parts of the matching algorithm. Nayar [19] has proposed algorithms which remove specularities by a complex reflectance model and a rotating polarizing filter. Once the specularities have been removed, stereo matching can proceed. But this requires extensive computation and special hardware. In the case of occlusion, matching is impossible, because the corresponding pair simply isn't there. Some stereo matching algorithms [4] search for an onto mapping between the images, rejecting any matches that do not work from left to right and right to left. This is helpful in eliminating unmatchable points from further consideration. Chung describes an edge based method which classifies different types of occlusions based on monocular cues, then uses this information during stereo matching.[22].

3.1 Correlation Measures

With or without occlusion and specularity, it is necessary to detect similarities between the left and right projections in order to solve the correspondence problem. Let us examine how the standard correlation measures attempt to

[1]This measure is defined in the next section.

extract disparity from this collection of image differences.

It is necessary to include some context when searching for the match for a particular pixel in order to avoid mismatches. This context is the window around the pixel. Very small windows, (1×1 being the smallest), tend to produce false matches. Very large windows (the entire image being the limiting case) lead to inexact matches, since the measure must be some kind of (potentially weighted) average of the depths associated with the pixels in the window. So window size has an important impact on how well the correlation works.

We will use $w(x, y)$ or simply w to refer to the set of pixels in a rectangular $m \times n$ window around pixel (x, y) So

$$c = \sum_{i,j \in w(x,y)} (A(i,j) - B(i+d,j))^2$$

is equivalent to

$$c = \sum_{i=-n/2}^{n/2} \sum_{j=-m/2}^{m/2} (A(x+i,y) - B(x+i+d,y))^2$$

In this case

$$0 \leq c$$

The above measure is the sum of squared differences (SSD) of the pixels in the two windows. Here, we attempt to find the match for some pixel $A(x, y)$ by moving the window over B, recording the disparity value d for which the correlation measure is minimal.

If the only difference between the two signals were the disparity shifts, this measure would probably be optimal. However, even assuming this to be true, it is important to get the size of the window right. If the scene is a plane whose normal is parallel to the camera axes, $d(x, y)$ constant, and the summation will evaluate to 0 at the correct disparity. If the window size is too small this may occur at several locations. For this disparity function, the larger the window size the better. If $d(x, y)$ is nonconstant, even if the center pixel of the window is matched correctly, its neighbors will be misaligned by an amount proportional to the magnitude of $d'(x, y)$, so the measure will not be zero. In general as we grow the window with a non-constant disparity function, the minimum value returned by the measure will stand out less sharply due to the averaging effect of the window, but the chance of mismatch will be reduced. (Interestingly, the human visual system cannot fuse several of depths at one time either, as can be verified by lining one's thumb up with a distant point; we see either two thumbs or two points.)

If we allow other image differences into our scene model, the correct match can be further obscured. For example an "offset" in part of the image may result in a poor correlation score at the correct disparity and causing a possible mismatch. We have seen this can occur when there is specular reflection on a planar part of the scene. The measure is also sensitive to outlyer noise, since all differences are squared. However, it is computationally inexpensive, and preserves the information about the basic differences between the two images, which could be useful at a later stage of processing. However, the poor performance of the measure in the face of common image differences has lead most researchers to use a different measure, called normalized correlation. This measure hides the actual differences by modeling and compensating for them.

Let $a(i,j)$ denote $A(i,j) - \overline{A}$

$$c = \frac{\sum\limits_{i,j \in w} a(i,j) b(i+d,y)}{\sigma_{A_w} \sigma_{B_w}}$$

where $-1 \leq c \leq 1$ and σ_{A_w} refers the the standard deviation of the signal within the window.

This measure normalizes the two signals as discussed earlier except only within the window. It then correlates them. If the values in one window are a linear combination of the values in the other, the correlation measure is 1. This will compensate for some of the camera position based differences due to lighting and surface properties. It also has the benefit of taking care of the need normalizing the signals from the two cameras since gain and offset are removed within each correlation window. However, this correlation method ignores actual contrast differences between the two windows, that is one linear combination is as good as the next which appears to be a drawback, as the gain and offset from real pairs of images tend to be fairly small. The measure is undefined when the denominator is 0 which can occur if there is no signal variation within the windows.

Pseudo-normalized correlation

$$c = \frac{2\sum\limits_{i,j \in w} a(i,j)b(i+d,j)}{\sum\limits_{i,j \in w} a^2(i,j)b^2(i+d,j)}$$

$$0 \leq c \leq 1$$

is computationally more efficient and pays more attention to the actual difference between intensities. [18] Other variants of normalized correlation have also been used. [12]

$$\frac{\sum\limits_{i,j \in w} A(i,j)B(i,j) - \overline{AB}}{\sigma_A \sigma_B}$$

Other correlation measures with potential application to stereo appear in the literature. Absolute value measures as well as several others are compared by Dvornychenko, [5] Robust window operators are discussed by Besl, and [2] least median squares are used by Jiang.[9] Smith presents a correlation method based on image irradiance which makes many fewer assumptions about image similarities but requires that the orientation of the scene surfaces be known a priori.[25]

3.2 Lack of information

The common consensus is that there is not enough information in stereo pairs to unambiguously match every pixel of a scene projection. (Consider matching a white patch with another white patch – it will match equally well in a number of locations under any intensity based correlation measure.) This problem occurs frequently in the matching of aerial photographs which contain large patches of farmland or forest. [17] Two approaches have been taken to deal with this problem. The most popular approach is to use a "feature operator" to select pixels in the image that have enough contrast to be matched unambiguously. This produces a sparse depth map which is then interpolated at a later stage. The other approach is to dynamically change the window size for each pixel so that the algorithm never attempts to match a patch with low intensity variation.

3.2.1 Feature Based Methods

The Moravec feature operator measures the intensity variation around a pixel in two directions. [18] and is designed to work well with a multi-resolution correlation scheme. Another interesting feature extraction method is auto-correlation which convolves a window with itself and fits a Gaussian to the result. The standard deviation of the Gaussian is then used to determine whether the area is appropriate for matching. A small standard deviation implies that there is enough interest for matching. Another feature operator is the difference of two Gaussians, [13] which is essentially an edge detector.

These techniques reduce the amount of computing time spent searching for correct matches as a much smaller subset of objects is matched. They also have the potential for being quite tolerant of differences in appearance of scene features as projected onto the two cameras. However, such techniques are still susceptible to the drifting occluding contour problem mentioned earlier, and to misplaced matches due to specularities. For smoothly curved objects, such as spheres, (continuous first derivative) the occluding contour appearing in each image will correspond to different scene locations. They tend to perform matching based on local criteria and then modify or eliminate matches according to the global surface model — often implemented through a voting scheme which determines the degree of local support for the match. Some algorithms include the global model into their algorithm [8] [27], and others use it to interpolate as a separate second step. [7] Some methods match features at a higher semantic level by correlating the values of the feature extractor instead of the intensities. For example, the algorithm can detect zero-crossings of the second derivative of the intensity function, and then match them based on the magnitude and orientation of the detected edge. [11] Such techniques often require that semantic features appear in the same order in each image, an assumption which breaks down when backward mappings exist. In any case, interpolation is needed, and interpolating a piecewise continuous function from sparse data points is not simple. Techniques for doing so are presented by Terzopolous and others. [14] [29] [3].

3.2.2 Adaptive Window Methods

Adaptive windows have the advantage of producing a dense depth map. They can be considered a form of local segmentation, as they divide the image into logical units to be considered separately. Here the logical unit is an image area with enough visual interest for a good match but not too much depth variation. Levine and then Yakimovsky present two similar approaches to getting a dense depth through adaptive windows. Both use the smallest symmetric window that contains enough signal variation to produce a reliable match using normalized correlation. Yakimovsky uses a sparse diamond shaped mask (which is equivalent to a subsampled rectangular window), and selects a window for which the autocorrelation function falls off sharply and which has the property that the signal to noise ratio in the window is greater than an empirically found constant (3/10).[33] The earlier work of Levine uses a square window. Both the algorithms select window size purely on the basis of intensity variance within the window, and then smooth the resulting noisy depth map.[12] Mori iteratively warps the right image while searching for a match, thereby effectively changing the amount of signal considered by a fixed size window.[17] Strat and Fischler [26] present an approach for a segmented images which gives a coherent framework for using warped or "virtual" images.

The adaptive window methods discussed above implicitly incorporate disparity variation into their choice of window size by selecting the *smallest* window which satisfies the signal variance constraint. We turn now to the adaptive window algorithm of Okutomi and Kanade which explicitly models disparity variation, thereby presenting a reasonable way of weighing the two competing factors in determining window size.

3.2.3 Okutomi Algorithm

Ideally we would like to make accurate assumptions about the nature of image similarity, and use windows at a single depth that include enough information for reliable matching. Okutomi uses Gaussian noise to model all dis-similarities between images which, given no prior information about the image dis-similarities, may be a reasonable approach. The algorithm balances the competing factors of image interest and uniform disparity on choosing a window size by comparing the intensity variation of the signal within the window with the disparity variation in the variation from the pixel of interest. Since the disparity values of pixels within the window are needed in order to compute the disparity variance, and this is what we are trying to compute, the method assumes that an initial disparity estimate already exists for every point in the image, and then uses an iterative improvement scheme to refine the disparity estimate.

The assumptions that we are near a correct match, and that the image intensity is smooth, allows us to directly compute the disparity without doing an exhaustive search through the image. In this sense, this algorithm is not a stereo matching algorithm but a registration technique.

The algorithm works as follows:

Given an initial window size and an initial disparity estimate for each pixel
Repeat until convergence or a certain number of iterations
    foreach pixel $I(i, j)$
        consider a set of candidate windows W
        foreach w in W
            compute the certainty of the disparity value
             that would be obtained using w
            record the w with the smallest uncertainty
    foreach pixel $I(i, j)$
        compute a new disparity estimate for $I(i, j)$ using
         w with the lowest uncertainty saved from above

$$\psi_1 = I_1(\xi, \eta) - I_2(\xi + d(0,0), \eta)$$

$$\psi_2 = I_2'(\xi + d(0,0), \eta)$$

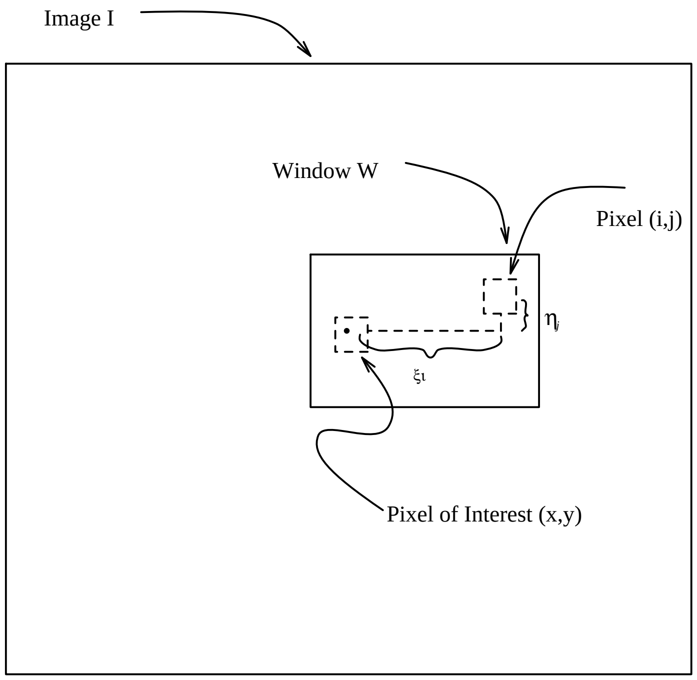
$d(0,0)$ refers to the current disparity estimate for the pixel of interest. So if (ξ, η) is a point in the window W in the left hand image, $(\xi + d(0,0), \eta)$ is the corresponding point in the right hand image.

Let $\hat{\alpha_f}$ define a measure of the variance of the first derivative of a window on the right hand.

$$\hat{\alpha_f} = 1/ \mid W \mid \sum_{i,j \in W} {\psi_2}^2$$

Let $\hat{\alpha_d}$ define a measure of the weighted variance of the disparities in a window on the left hand image.

$$\hat{\alpha_d} = 1/ \mid W \mid \sum_{i,j \in W} \frac{(d(\xi_i, \eta_j) - d(0,0))^2}{\sqrt{{\xi_i}^2 + {\eta_j}^2}}$$



The new disparity estimate is the sum of the disparity estimates from the last iteration and $\hat{\Delta}d$. $\hat{\Delta}d$ is computed as follows:

$$\hat{\Delta}d = \frac{\dfrac{\sum\limits_{i,j \in W} \psi_1(\xi_i, \eta_j)\psi_2(\xi_i, \eta_j)}{2\sigma_n^2 + \hat{\alpha_f}\hat{\alpha_d}\sqrt{{\xi_i}^2 + {\eta_j}^2}}}{\dfrac{\sum\limits_{i,j \in W} (\psi_2(\xi_i, \eta_j))^2}{2\sigma_n^2 + \hat{\alpha_f}\hat{\alpha_d}\sqrt{{\xi_i}^2 + {\eta_i}^2}}}$$

This is the correction to the old disparity estimate. Note that the contribution of each pixel is weighted by its distance from the pixel of interest. The weighting is arbitrary, the square of the distance, or 1/2 the distance are equally plausible weights. In the weighting is an implicit surface model.

The uncertainty of the disparity correction is denoted by $\sigma_{\Delta d}$

$$\sigma_{\Delta d}^2 = \frac{1 \sum\limits_{i,j \in W} (\psi_2(\xi_i, \eta_j))^2}{2\sigma_n^2 + \hat{\alpha}_f\hat{\alpha}_d\sqrt{{\xi_i}^2 + {\eta_i}^2}}$$

For a complete development of these equations, the reader is referred to the work of Okutomi and Kanade [20][10] and Ryan. [24] Here we examine how these equations would apply to a window consisting of a single pixel so that the reader can develop some intuition about how the disparity correction is computed.

Suppose we wish to improve our disparity estimate $d(x,y)$, for some pixel $I(x,y)$. We look at the difference:

$$I_1(x,y) - I_2(x + d(x,y), y) = n(x)$$

$n(x)$ is our error. By looking at the first derivative of I_2, and $n(x)$ we can see if our estimate was too far to the right or too far to the left. If $n(x)$ is 0 then there is no need for correction. If we denote the correct disparity as $d_r(x,y)$ then there is some Δd such that $d_r(x,y) = d(x,y) + \Delta d$, forcing $n(x)$ to be 0. We can compute Δd by using the Taylor expansion for I_2'

$$I_2(x + d(x,y) + \Delta d, y) \simeq I_2(x + d(x,y), y) + \Delta d I_2(x + d(x,y), y)$$

solving for Δd we get

$$\Delta d = \frac{I_2(x + d(x,y), y) - I_1(x,y)}{I_2(x + d(x,y), y)}$$

Just to make things concrete, if $I_1(x,y) > I_2(x + d(x,y), y)$ and $I_2'(x + d(x,y), y)$ is positive, then Δd should be positive. See Figure 3.2.3. This reasoning is applied to every pixel in the window, and their contribution to $\hat{\Delta} d$ is weighted in inverse proportion to the distance from the pixel of interest. When applied to an entire window, $n(x)$ will never be 0. Instead, the error $n(x)$ is modeled as Gaussian noise and Bayesian statistics are then used to find the window size that yields the $\hat{\Delta} d$ most likely to improve the disparity estimate. The basic reasoning is the same, and the formula for computing $\hat{\Delta} d$ bears a strong resemblance to the equation above.

The algorithm computes an appropriate window size for each pixel based on purely local information. Because it acts locally it can potentially be modified to run on subimages in time proportional the subimage area. The method does not deal with how to expand windows at image boundaries or in general what to do when disparity or intensity information is missing. However, it is clear that a small amount of systematic noise could badly

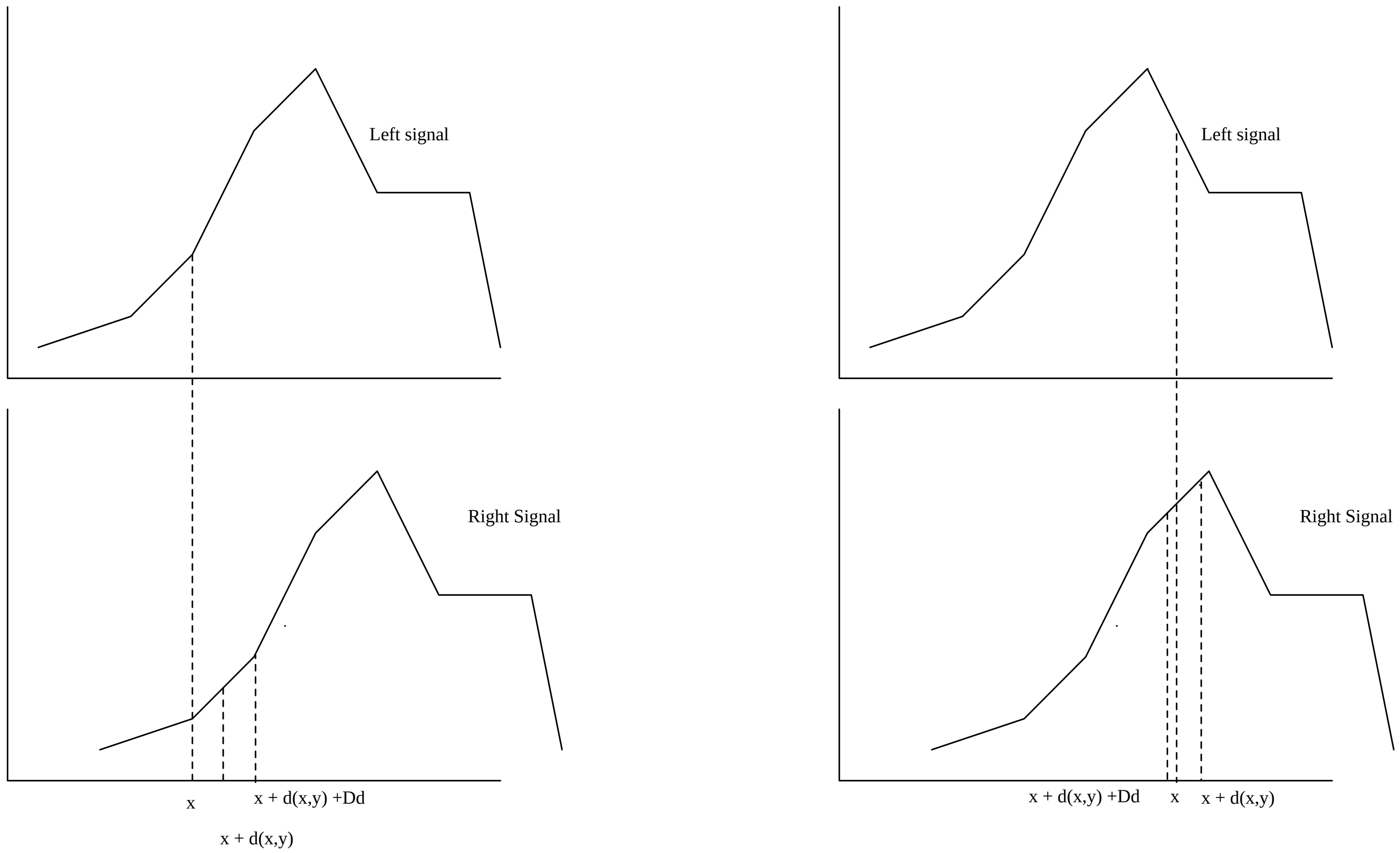


Figure 6: The example on the left shows a situation where the disparity correction Δd changes the disparity estimate in the right direction. The example on the right shows a case where Δd changes the disparity estimate for the worse.

throw off the computation of $\hat{\Delta} d$, skewing the final disparity estimate. For example an additive constant of just a few grey levels in part of the left or right image will change the computed disparity for any pixel using that part of the image in its window. Also, in order for Δd to correct the disparity estimate, the initial estimate must be close to the correct disparity. If half the wavelength of the first derivative of the signal is greater than the difference between the disparity estimate and the true disparity, Δd is unlikely to help converge to the correct solution.

4 Cursor Stereo

From our analysis of the algorithm, it seems that the adaptive window algorithm should be able to run on any shaped window, provided that the window is spacially continuous and that all the information needed to evaluate whether to expand the window to include a particular pixel is available. When evaluating a window around a pixel, we should keep in mind that there are really two windows involved, the first is the window on the left image which surrounds the pixel for which we are trying to compute a new disparity (pixel of interest); the other is a window on the right image, which is the same size and shape but is shifted to the right by the current disparity estimate for the pixel of interest. Every pixel in the left window must have a disparity estimate, and an intensity value. Every pixel in the corresponding window on the right image must have a first derivative of intensity, and an intensity. We term a window meeting these criteria a "complete window." Leaving aside for a moment the implementation, it seems clear that complete windows are what wes need because the same set of pixels should be used for computing $\hat{\alpha_f}$, $\hat{\alpha_d}$, $\sigma_{\Delta d}$, and Δd. Since we are only considering a portion of the image at a time, it will be important to adapt the algorithm to cope with pixels at the edge of a sub-image of interest. (If the sub-image of interest is the entire image, this corresponds to adapting the algorithm to work at the image edge.) Since the pixels at the edge of a sub-image will have incomplete information about the disparity of their neighbors, the algorithm cannot be applied directly. Beyond the image edge there is no intensity information; beyond the subimage edge there is no disparity information.

4.1 Selecting Candidate Windows

Given that some measure of appropriate window size exists, we need a reasonable method of selecting candidate windows to evaluate. In the 1D adaptive window algorithm, each scanline is treated separately. [20]

All possible windows centered on the pixel of interest that are within fixed size limits are considered. So if the range of sizes is from 3 to 39 pixels per window, windows of size 3, 5, 7,...39 would be tried. In general if $1 \times n$ is the largest allowable window, $n/2$ windows would be evaluated. In the 2D version of the algorithm, [10] rectangular (not necessarily square) windows are allowed. Also, windows are considered whose pixel of interest is not the center. If $n \times n$ is the largest allowable window, then trying every possible window in this situation results in $O(n^4)$ possible window sizes. Partly to prune this search space to a reasonable size, a greedy algorithm (path of steepest decent) is used to decide which windows to evaluate. This results in $O(n)$ window evaluations.

Consider the problem of selecting windows to evaluate within an arbitrary sub-image. If the sub-image is the entire image, we would like the candidate windows to be those considered by the two dimensional algorithm. If the sub-image consists of a single scanline, we want the windows to grow only within that scanline. For any sub-image which is a curve, we would like to use the 1D algorithm, where the candidate windows grow along the curve instead of the scanline. However, if the sub-image has no particular geometric form, and is simply a "blob", we must represent the sub-image explicitly by naming every pixel within the sub-image. We then need a method to find candidate windows inside such a sub-image. Once we are willing to name every pixel in the sub-image, we can grow rectangular windows as in the 2d algorithm but only consider pixels which are in both the rectangular window and the sub-image. Other possibilities would be a function that weights the pixels in a large constant sized window. A mask is a function tht is 0 or 1; a more complex weighting function could be used— where the weights could reflect the confidence that the information associated with the pixel is correct. If we think of the pixels within a window as members of a set, we can write:

$$W_{candidate} = W_{rectangular} \cap SubImage$$

If we take the intersection of the points considered by the 2D algorithm with the points which are in the subimage, we find that we get the 1d algorithm

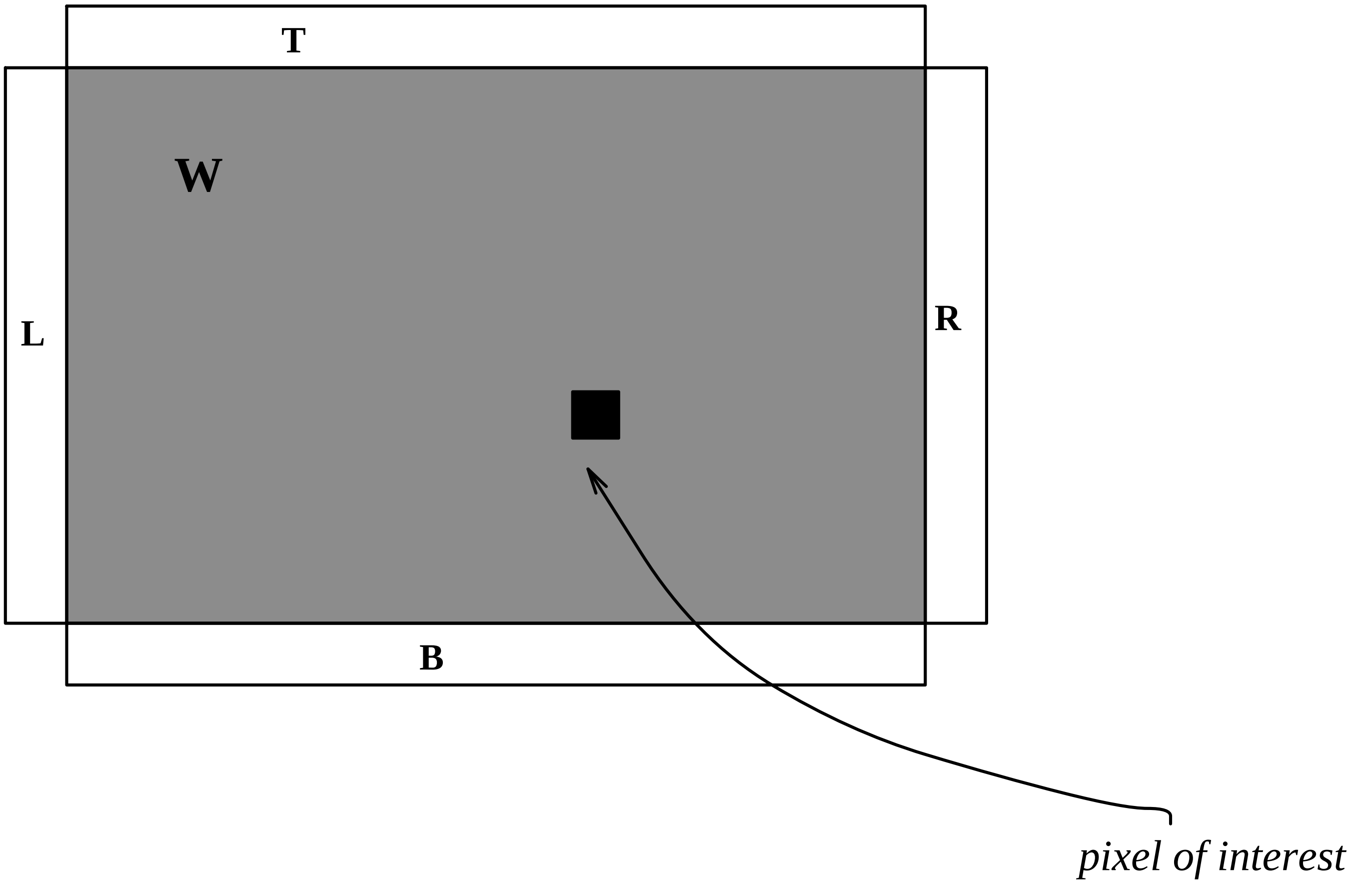


Figure 7: Expanding an off-center rectangular window.

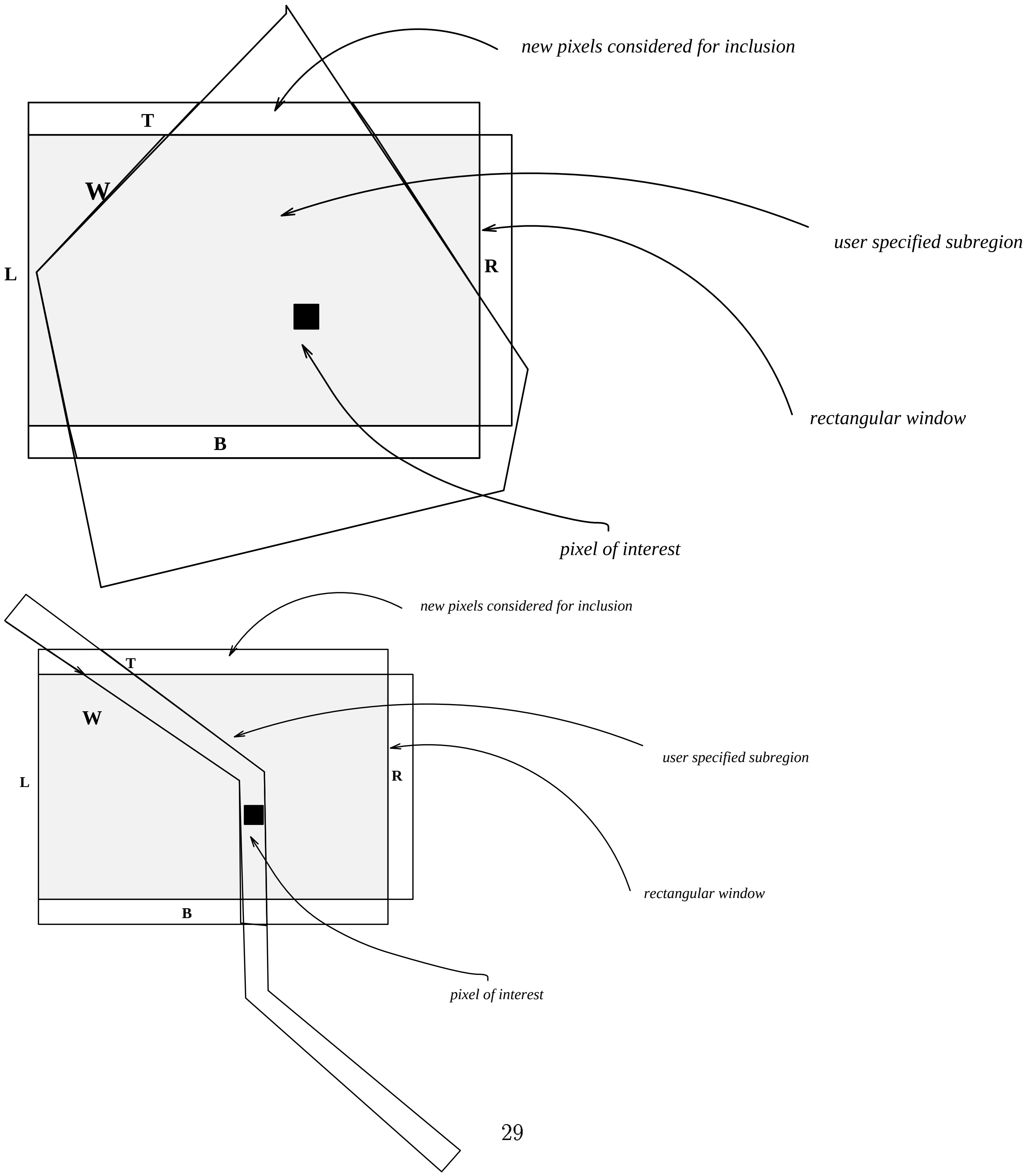


Figure 8: Intersection of a rectangular window and a sub-image mask

when the subimage contains a single scan line and we get windows which grow grow only along the curve, if the sub-image is a curve. Likewise, windows within a blob expand as in the 2d algorithm but conform to the edge of the subimage when it is encountered. If the sub-image is the entire image then we have the 2D algorithm with a reasonable way of handling the image edge.

If a bitmap is used to represent the subimage, the set intersection can be handled by a logical "and" of each pixel in the rectangular window. This computation should not increase the asymptotic complexity of the algorithm, so the running time for processing the subimage compared to using the algorithm on the whole image will be in the same proportion as the subimage to the image. Note that a little extra bookkeeping is involved to figure out if the window should be expanded in a direction that is completely masked out. That is, where $W_{rectangular_1} \cap SubImage = W_{rectangular_2} \cap SubImage$. Consider a subimage which is a diagonal line at 45 degrees. If we prohibit expansion of the rectangular window in directions which include no new pixels for the candidate window, we see that we cannot grow the candidate window at all. However, expanding an edge of the rectangular window when expanding in every direction cannot change the the candidate window is clearly a waste of time. So we need to keep a tally of the number of times a window expansion was attempted which resulted in no new pixels being added to the window.

We implement the window growing in the following way. We keep a bitmap which records which pixels in the left image are part of the selected sub-image. These are the only pixels having initial disparity estimates and are the only pixels for which new disparity estimates will be computed. In the right image, we also keep a bitmap. All but the leftmost pixels (which nothing from the left image could map to) and pixels at the image edge (having undefined first derivatives) were "masked in" making them eligible window members. We assumed that the depth range of the scene was known ahead of time, so the minimum and maximum disparities could be pre-computed. The leftmost columns of the right image are eliminated because they cannot be part of an onto mapping. The number of columns eliminated is the minimum disparity for the image pair. Similarly the same number of columns are eliminated from the right edge of the left image. So we add a further condition to membership in a complete window — all pixels should be part of the onto mapping between images. If occlusions have been detected at an earlier stage in processing, this information is recorded in the bit-maps for the left and right images and those pixels excluded from consideration when evaluating windows.

Since windows could be an arbitrary shape, we kept a linked list of which pixels were in a given window. For example, when attempting to expand a window down, we move the current bottom of the rectangular window down one row. We then take the intersection of the pixels in that new area with the sub-image mask. For each of these pixels, we then check the bitmap of the right image and check if the pixel shifted over by the current disparity estimate is part of the right image bitmap. If it is, then the pixel is added to the linked list of pixels belonging to the window, otherwise it is not. All computation of window statistics is then done by walking the list rather than by continually checking the bitmap.

4.2 Generating Artificial Images

Before giving the results of our experiments using the algorithm on arbitrarily-shaped subimages, it would be helpful to describe how we generated the artificial images we examine. Our artificial data attempts to model the transformation of a real-valued intensity and depth function for a single opaque surface into a pair of integer-valued discretely sampled images.

We start with a real-valued intensity pattern and real-valued depth map for the scene and assign intensity values from our real-valued intensity function to the left image. By looking at the underlying depth map we get a disparity value for each pixel. If the disparity value is such that it is part of a forward mapping, the disparity value is recorded in the left disparity map and its negative assigned to the corresponding location in the right disparity map. The intensity value of that pixel is assigned to its corresponding location in the right image. Since the disparity will in general be real-valued, the corresponding pixel in the right image will not be an integer. To minimize round-off error, we do this mapping from left to right at very high resolution. If the disparity value does not create a forward mapping, this pixel is occluded in the right image, so the disparity is marked as undefined, and nothing is assigned to the right disparity map or image.

At this point we have a complete high resolution left image, a left and right disparity map and a right image with holes in them where nothing has been assigned or the values are undefined. Holes larger than one pixel at final resolution are deemed to be part of an occluded area. Since the right camera sees something here that the left image doesn't, intensity values are assigned to these locations from an independent intensity pattern. The disparities associated with these locations are left marked undefined. We then apply a Gaussian filter to these images to produce the final low resolution images, thereby imitating the workings of a video camera which receives millions of light rays and averages them to get a single pixel value.

The frequency of the intensity pattern was very low in all cases.

5 Experiments On Artificial Data

Here we compare the sub-image algorithm to the full-image algorithm for a slowly changing disparity function. We obtained results comparable to the full-image algorithm using the sub-image masking technique. Since windows cannot always grow the full size they would attain if the entire image were available, we should expect somewhat different behavior from these restricted windows. For the same reason, the processing time per pixel should be somewhat less for the sub-image version. Running time was measured by wall time on a Sun4 on a heavily used system. We show the results of the algorithm running on images with signal to noise ratio of $400:1$ and $50:1$. [2] Table 9 shows the processing times and average error per pixel for the one dimensional and two dimensional subimages running on a pair of images with little noise added. The same algorithm is run in all cases, with different amounts of the image masked off. The initial average error per pixel was 1.357. The algorithm was run for six iterations, the first three using a fixed window and the last three using a variable window. In all cases enough iterations were allowed so that the algorithm converged.

SubImage	**time(secs)**	**points**	**seconds/point**	**RMS error**
entire image	3180.0	1444	2.20	.086
1D horizontal	21.0	36	.58	.092
1D vertical	25.6	36	.71	.071
triangular	57.8	54	1.16	.108

Figure 9:

[2]Signal to noise ratio was computed as σ_s/σ_n not $\log_k(\sigma_s/\sigma_n)$ as is sometimes done.

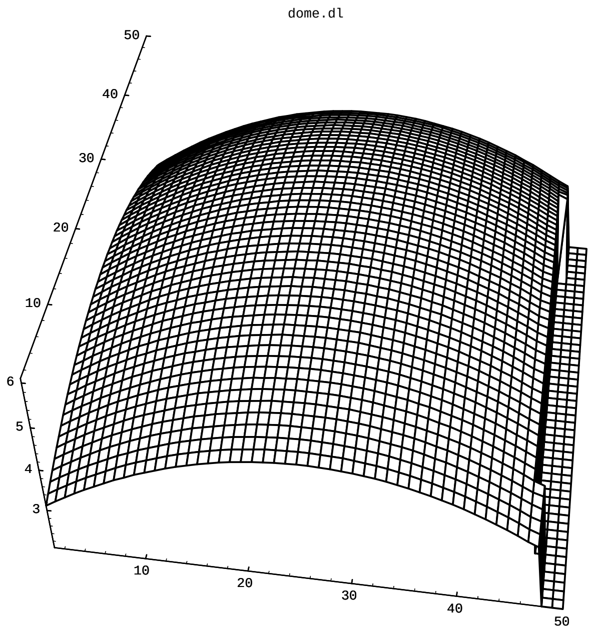


Figure 10: Correct disparity map and initial starting disparities.

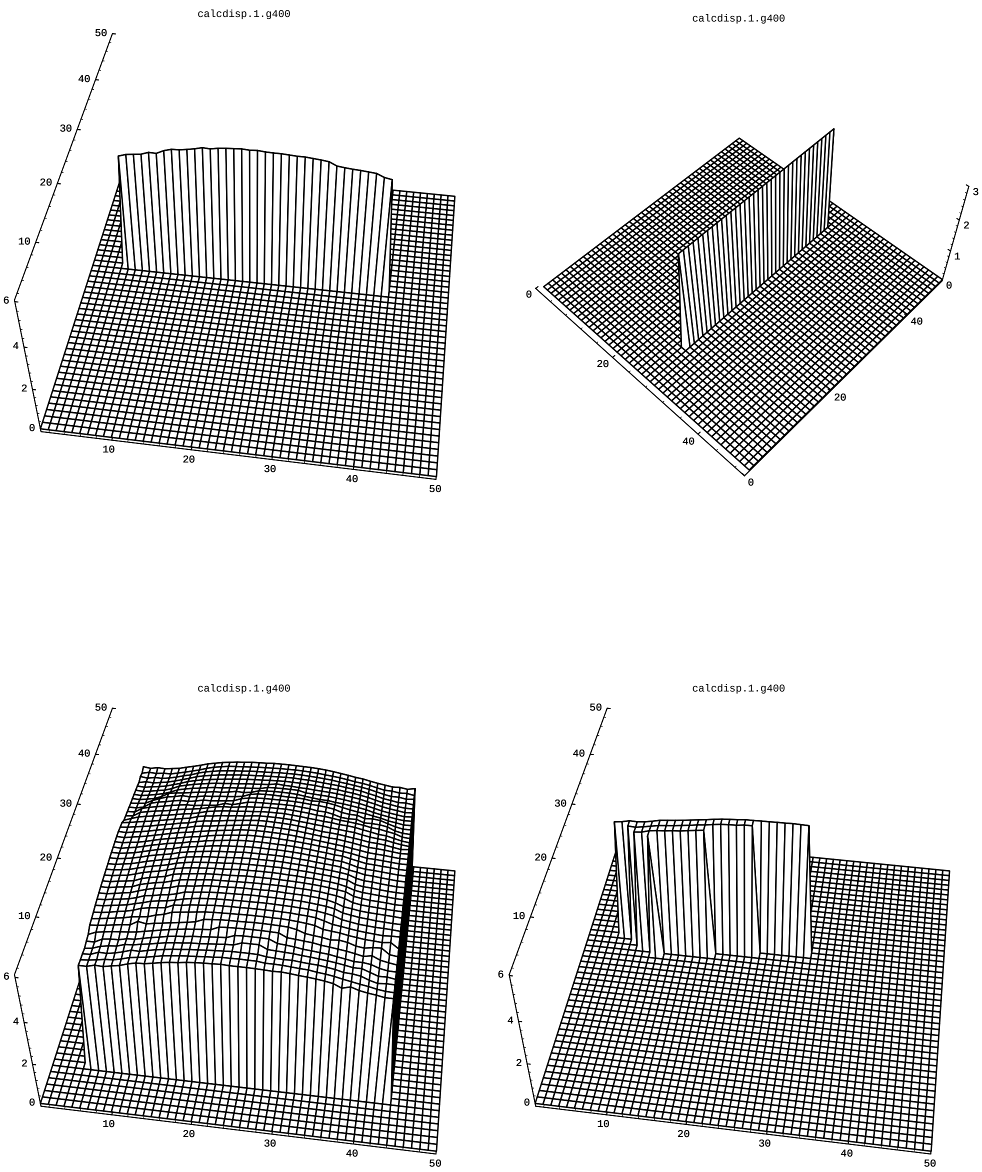


Figure 11: The results of running the sub-image algorithm on a single scan-line, on a single column, on the entire image, and on a triangular sub-image.

We then repeat the experiment at the higher noise level. Table 12 shows the processing times and average error for the same set of sub-images, again using six iterations. In the presense of a large amount of noise, the performance falls off.

SubImage	**time(secs)**	**points**	**seconds/point**	**RMS error**
entire image	1568.3	1440	1.09	.29
1D horizontal	23.5	36	.65	.25
1D vertical	25.4	36	.70	.35
triangular	72.9	54	1.35	.27

Figure 12:

We found that we could improve the results slightly on both data sets by limiting the maximum window size allowed to 20×20 pixels. The algorithm appears to select a window size which is too large, thereby excessively flattening the disparity map. By way of comparison, a fixed window of 3×3, 11×11, and 20×20 run on the horizontal scanline produced average error per pixel of .406, .254, and .265 respectively, for the $50 : 1$ signal to noise ratio data set. So the right sized fixed window in this case did comparably to the variable sized one, in about 1/10 the time. Although the variable window algorithm always outperformed the fixed window algorithm in the presense of sharp edges, this was not always the case for smoother maps. This was somewhat surprising since the variable size window should determine what window size is optimal and select it. One possible explanation for this was that the best window was never considered because it did not happen to lie on the path of steepest decent. Another possibility is that there was some unmodeled benefit to a symmetrical window.

We tried several different search strategies for finding the optimal window shape and size. The Kanade-Okutomi 1D algorithm uses an exhaustive search through symmetrical windows, and the 2D version uses off-center windows, found by steepest decent. We tried weighting the 2D windows to favor more symmetric windows, but found only a marginal improvement. We also tried a combination of exhaustive search through all possible small windows, combined with steepest decent for larger windows. Although we obtained better results on images having smooth disparity patterns, we found that this technique obscured the edges in sharp disparity patterns. For example,

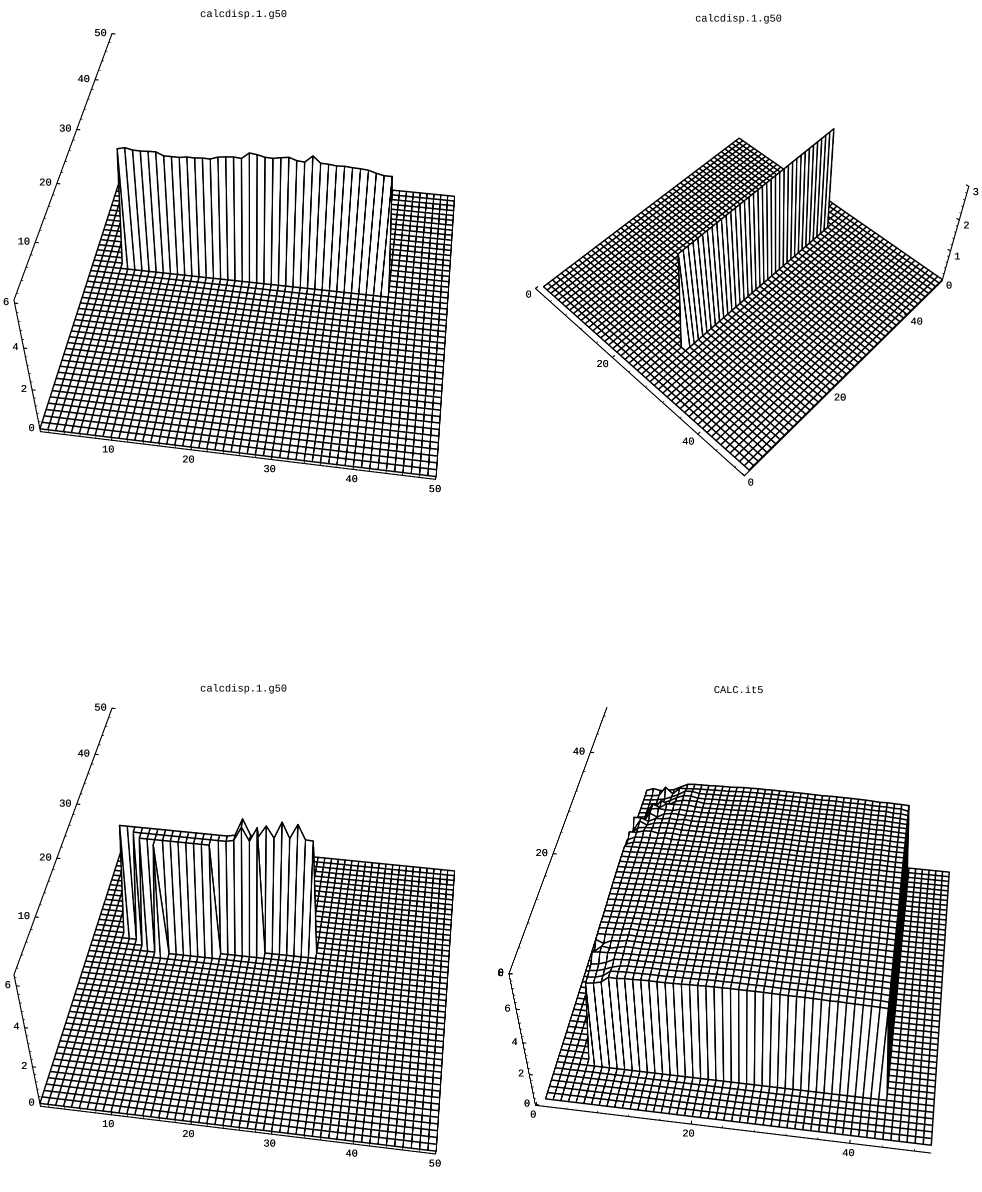


Figure 13: Window appears to grow too large. Note the single-column vertical sub-image has become completely flattened. The signal to noise ratio for this data set was 400 : 1.

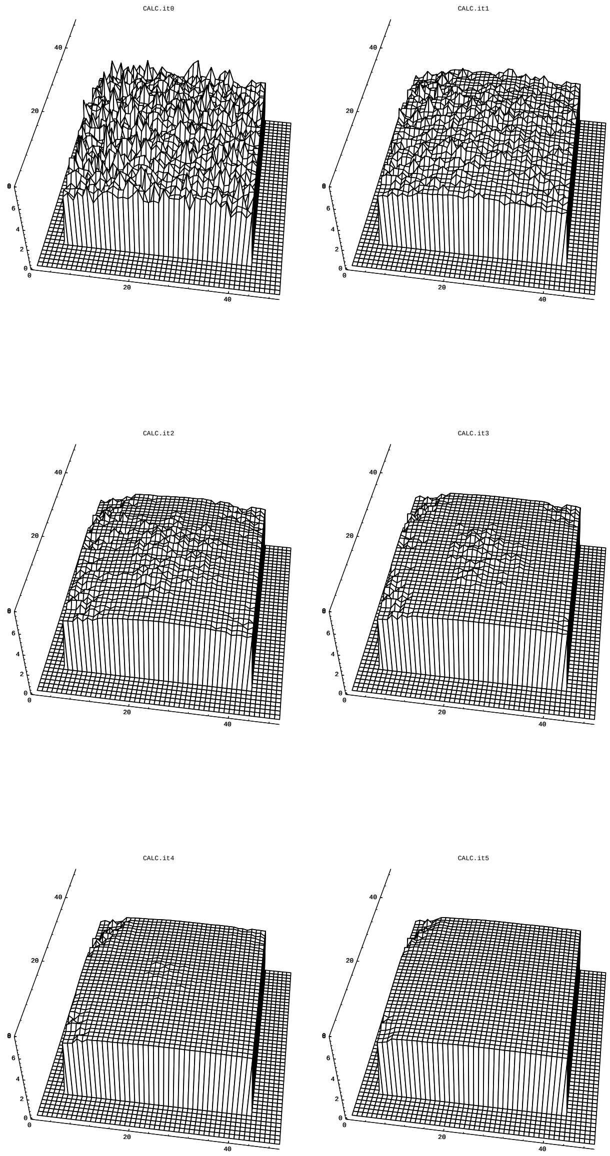


Figure 14: The algorithm in progress on a noisy image pair. The window appears to grow too large in the presense of large amounts of noise, flattening out the subtle change in disparity. Iterations 0, 1, 2, use a fixed window. Iterations 3, 4, 5 use a variable sized window.

starting from a disparity map which was perterbed an average of 1.5 pixels, this combination search strategy failed to detect a sharp disparity edge of height 2.0 pixels. For pixels near the disparity edge, larger symmetric windows which covered the edge and deemed better than the small windows which avoided the edge. The error difference was 5 times larger for edge pixels using this combination search strategy (.497 vs .094). It appears that the steepest decent search strategy is a necessary part of the edge-finding process. We ultimately found that the algorithm simply selected windows that were too big when the disparity pattern was smooth. This suggests that α_d might better be computed by weighting the disparity contributions of window members by something like $1/2\sqrt{\xi^2+\eta^2}$ rather than $\sqrt{\xi^2+\eta^2}$.

We found that we obtained comparable results using the sub-image masking technique that we obtained using the algorithm on the entire image. Since the running time for sub-images is so much less, this constituted a valuable debugging tool.

One of the things we were curious about was whether the starting window size and shape had much of an effect on the final window selected. We compared several versions of the algorithms on a data set containing a sharp horizontal edge. We used a horizontal edge here so as not to introduce the occlusions that necessarily occur with a sharp vertical edge. The algorithms are ordered by their sensitivity to disparity information. The algorithm least sensitive to the disparity data is the "fixed window" algorithm which makes no use of disparity information in determining the window size, always using the same sized window (in this case 11×11). "Variable window 1" pays attention to disparity information when growing the window to a larger size, but does not use disparity information in selecting the original starting window size. A small symetrical window (3×3) is always used. "Variable window 2" pays attention to disparity information when selecting an initial window and when growing the window to a larger size. This algorithm does an exhaustive search of all the 3×3 windows that contain the pixel of interest, then selects the one with the lowest weighted disparity variance, as a starting window. We also tried 1×3 and 3×1 for initial windows but the results were irratic, sometimes improving sometimes creating substantial error. The combination algorithm, uses the fixed window algorithm for its first pass, variable window 1 for its next 4 passes, and variable window 2 for its last two passes. It pays more and more attention to the disparity information as the algorithm progresses.

As can be seen in Figure 17, the fixed window algorithm blurs the edge and gets relatively poor results. Variable algorithm 1 does substantially better, it blurs the results only for the pixels right on the edge. Variable algorithm 2 does well on one data set but very poorly on the other. It has presumably been led astray by paying attention to disparity estimates which were incorrect, and does not converge at the correct answer. It is interesting that the starting root mean square error per pixel in the case where variable window 2 goes astray is actually lower than the starting error in the data set where it converges to the correct answer. What is critical is whether the error right at the disparity edge is misleading or not. The combination algorithm does the best, as it pays more attention to the disparity information as it becomes more reliable.

The arrangement of running one pass of the fixed window, 4 passes with variable algorithm 1 and 2 with variable algorithm 2 is arbitrary. It would be better to quantify the uncertainty of each disparity estimate and use this to determine how much attention to pay to disparity during window selection.

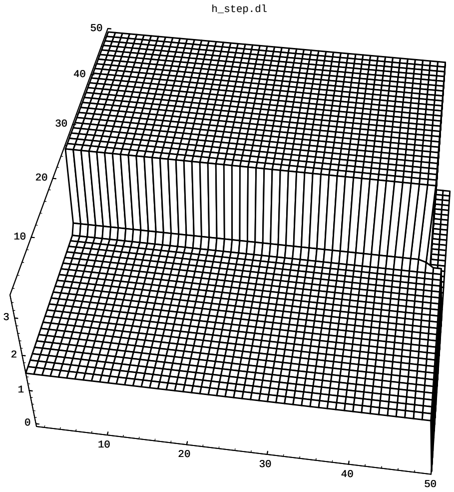
Table 15 compares the root mean square disparity error per pixel for various window selection algorithms on data with a horizontal disparity edge which fell between pixels. The signal to noise ratio was 400 : 1. Each algorithm used seven iterations. There were roughly 110 points in each subimage.

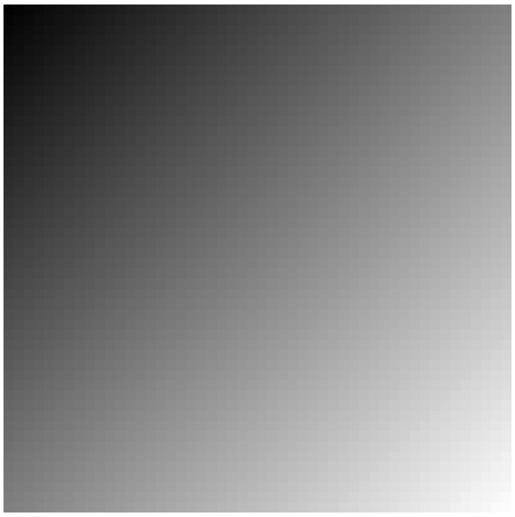
Algorithm	**subimage A**	**subimage B**	**SubImage C**	**SubImage D**
no processing	1.462	1.286	1.310	1.283
fixed window	.257	.262	.910	.736
variable win 1	.081	.087	.704	.269
variable win 2	.021	.219	.586	.380
combination	.021	.029	.437	.132

Figure 15:

In Table 19 we compare the average disparity error per pixel for the same set of variations on the algorithm, with different amounts of Gaussian noise added. In this case the disparity edge falls on pixels not between them. Each algorithm was run for seven iterations.

It was often the case that the algorithm would get near the right answer and then drift away from it. In almost all cases the algorithm did converge to a solution. Very occasionally it would ossilate back and forth.





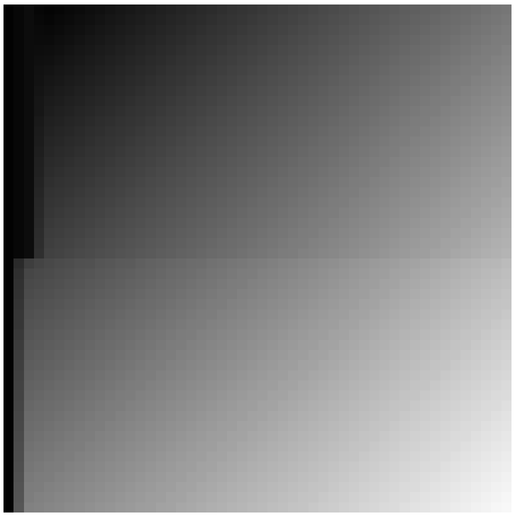


Figure 16: The left and right disparity maps, and the left and right images.

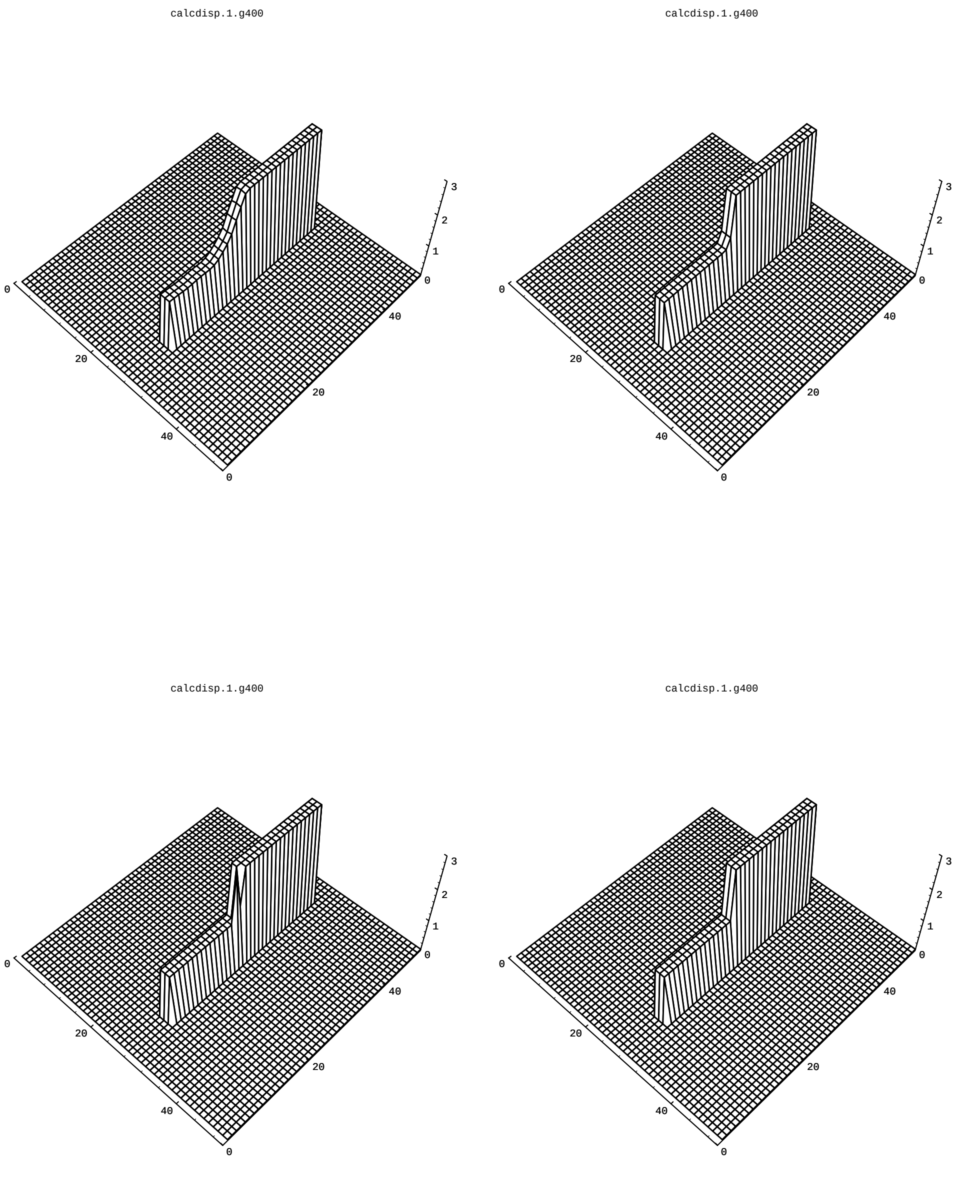


Figure 17: Results of the fixed, varible1, variable2, and combination window selection algorithms, on SubImage B.

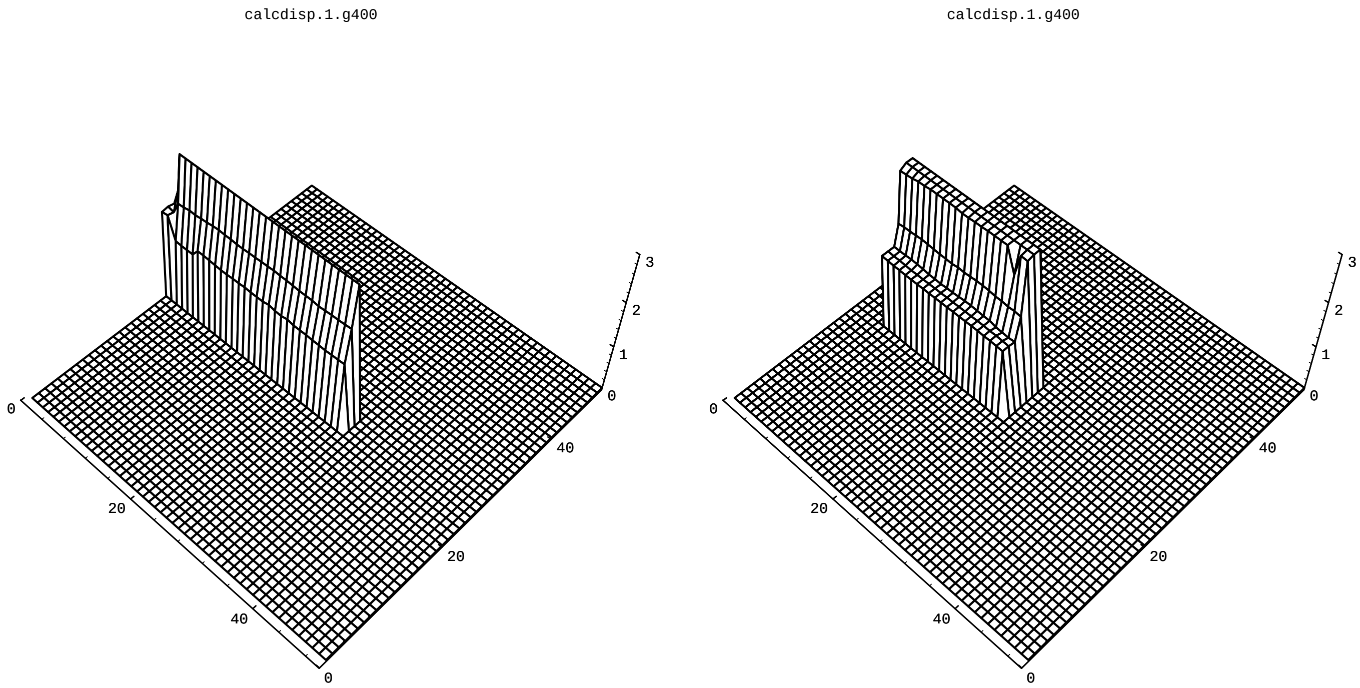


Figure 18: Recovered disparities for subimage C and D above using the combination algorithm.

Algorithm	**400:1**	**100:1**	**50:1**
no processing	1.306		
fixed window	.499	.563	.634
variable win 1	.252	.379	.561
variable win 2	.342	.424	.529
combination	.094	.336	.581

Figure 19:

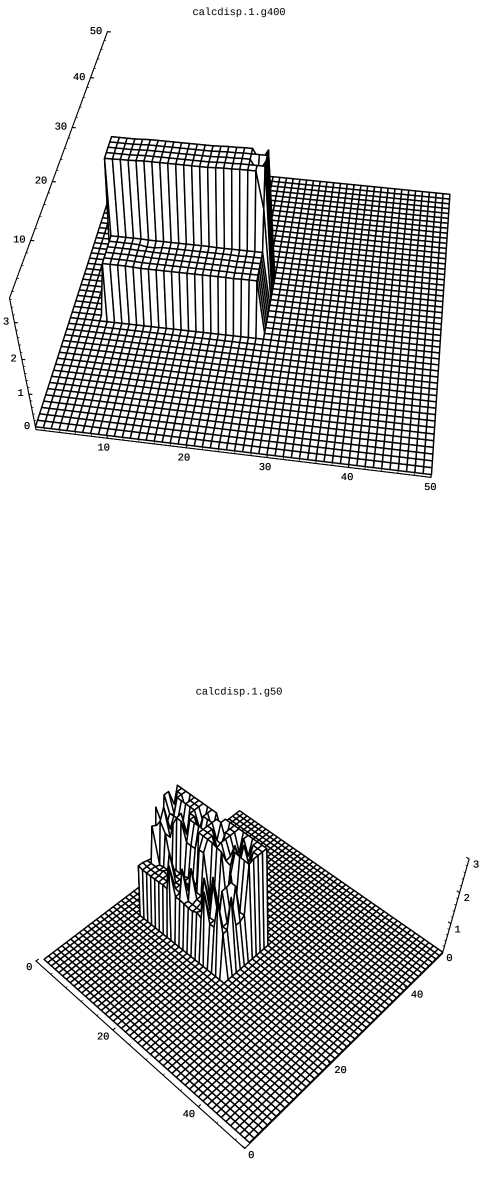


Figure 20: The combination algorithm at signal to noise ratio 400 : 1 and 100 : 1

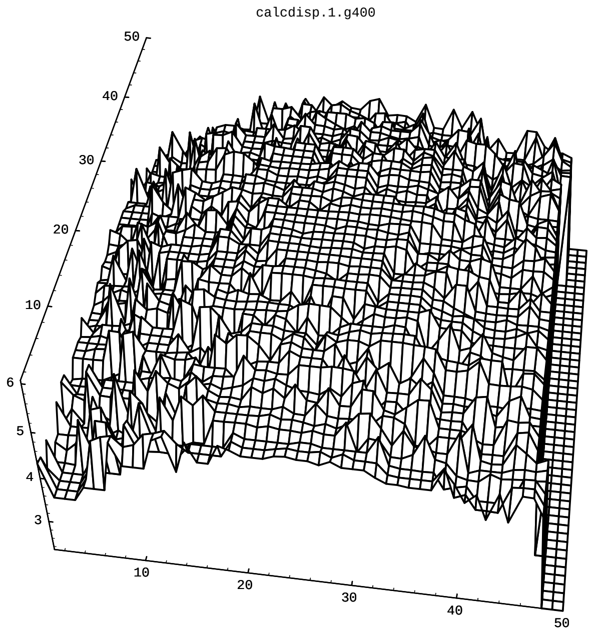


Figure 21: This is what the off-center algorithm does in the absence of sharp edges.

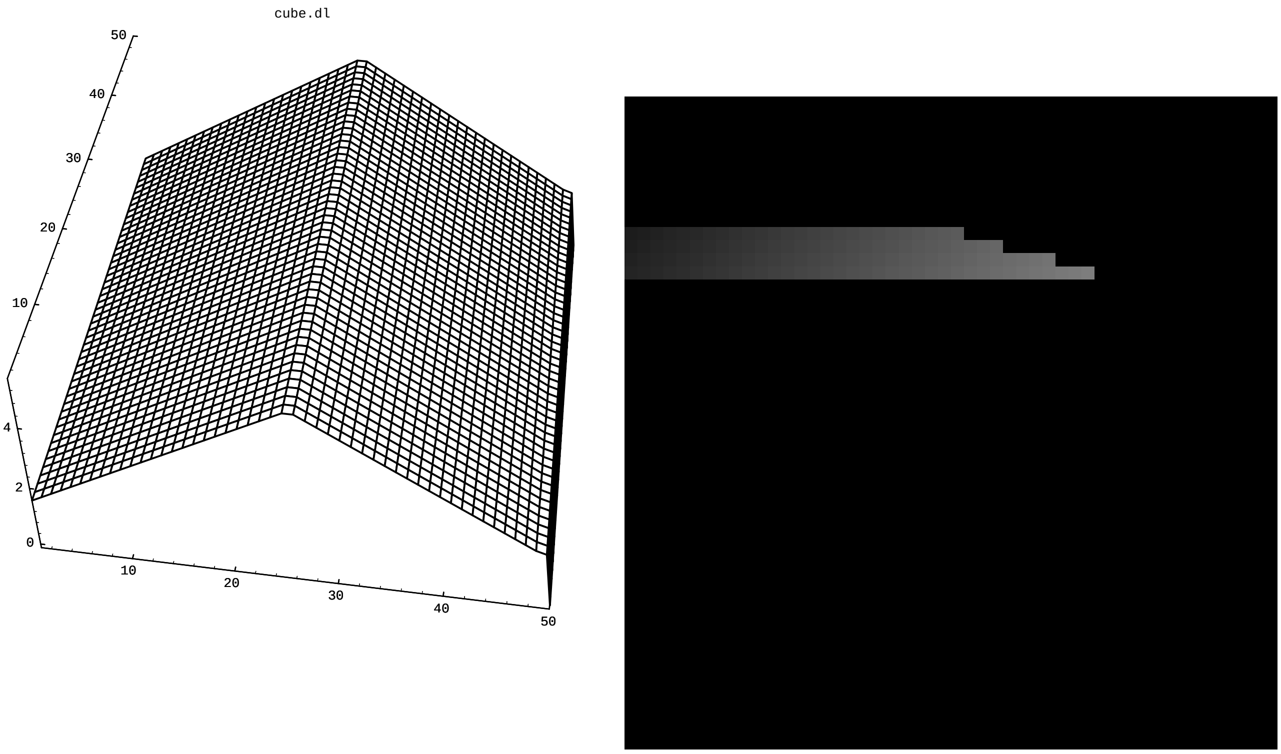
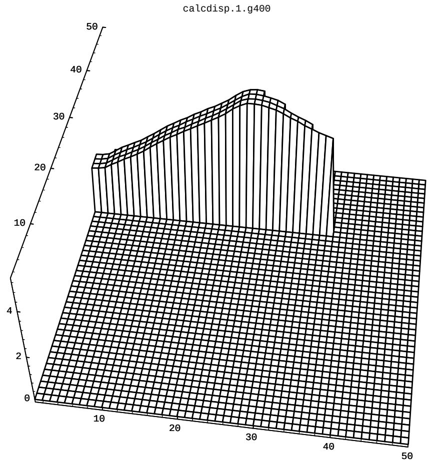
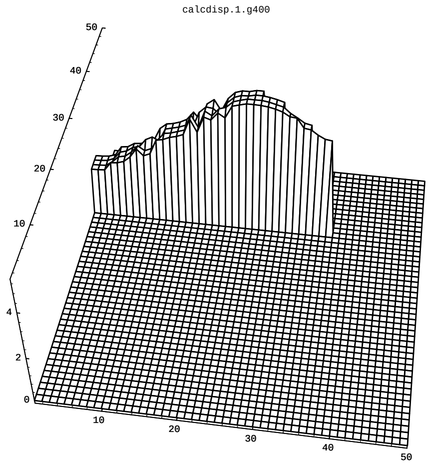


Figure 22: The entire disparity pattern and selected subimage.





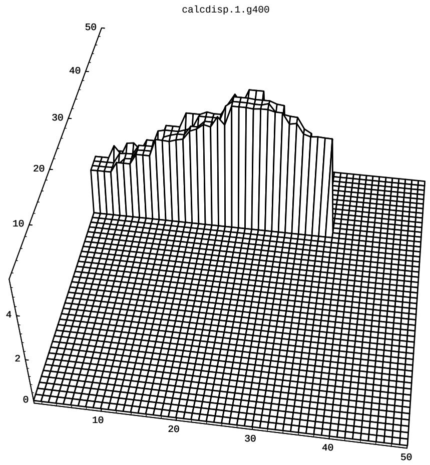


Figure 23: final disparities using the fixed window algorithm, and two versions of the variable window algorithm. The respective root mean square error per pixel for each algorithm was .089, 155, .187.

Here is an example where the fixed window algorithm works better than the variable window. The variable window algorithms using a steepest decent search strategy tend to find flat-spots where they don't exist. By selecting a window which has little depth variation, once a flat spot develops, it tends to remain, since pixels from outside the region will not be considered. So the same set of pixels will be considered for all the pixels on the plateau — which will give nearly the same disparity for each pixel, thereby maintaining the plateau. The fixed window on the other hand considers a different set of pixels when computing the disparity for each pixel, so doesn't develop this effect.

Here we show the behavior of the algorithm in the presense of occlusions. The sharp increase in disparity creates a right occlusion. There is a "hole" in the values mapped from the left image to the right. This hole corresponds to the dark strip in the right image. Since these values do not come from the left image we are free to put an unrelated intensity pattern there. For starting disparity values, for the pixels with undefined disparities, we used linear interpolation between neighboring pixels having defined disparities. Since this part of the right image is not part of the onto mapping, it should be left out of the complete window. We compare the results of leaving these pixels in and masking them out. The improvement is observable but surprisingly small.

——————————————————————————-

6 Experiments On Real Data

Finally, we tested the algorithm on real data. In order to avoid the extra variable of the camera model, we compute disparity rather depth. We also used a single camera on a horizontal track to make sure that we did not have rotational or vertical mis-allignment between the two views. The camera we used encodes color information onto an NTSC signal which was then digitized back into RGB on a RasterOps video capture card. From this we created an 8 bit image by summing the color compenents and dividing by 3. To get initial disparities we used the Moravec multi-resolution correlation technique [3], to get integer-valued disparities for a sparse set of nearby points which had enough visual interest for reliable matching. We then interpolated those

[3] implemented by Janez Funda, IBM T.J. Watson Research Center

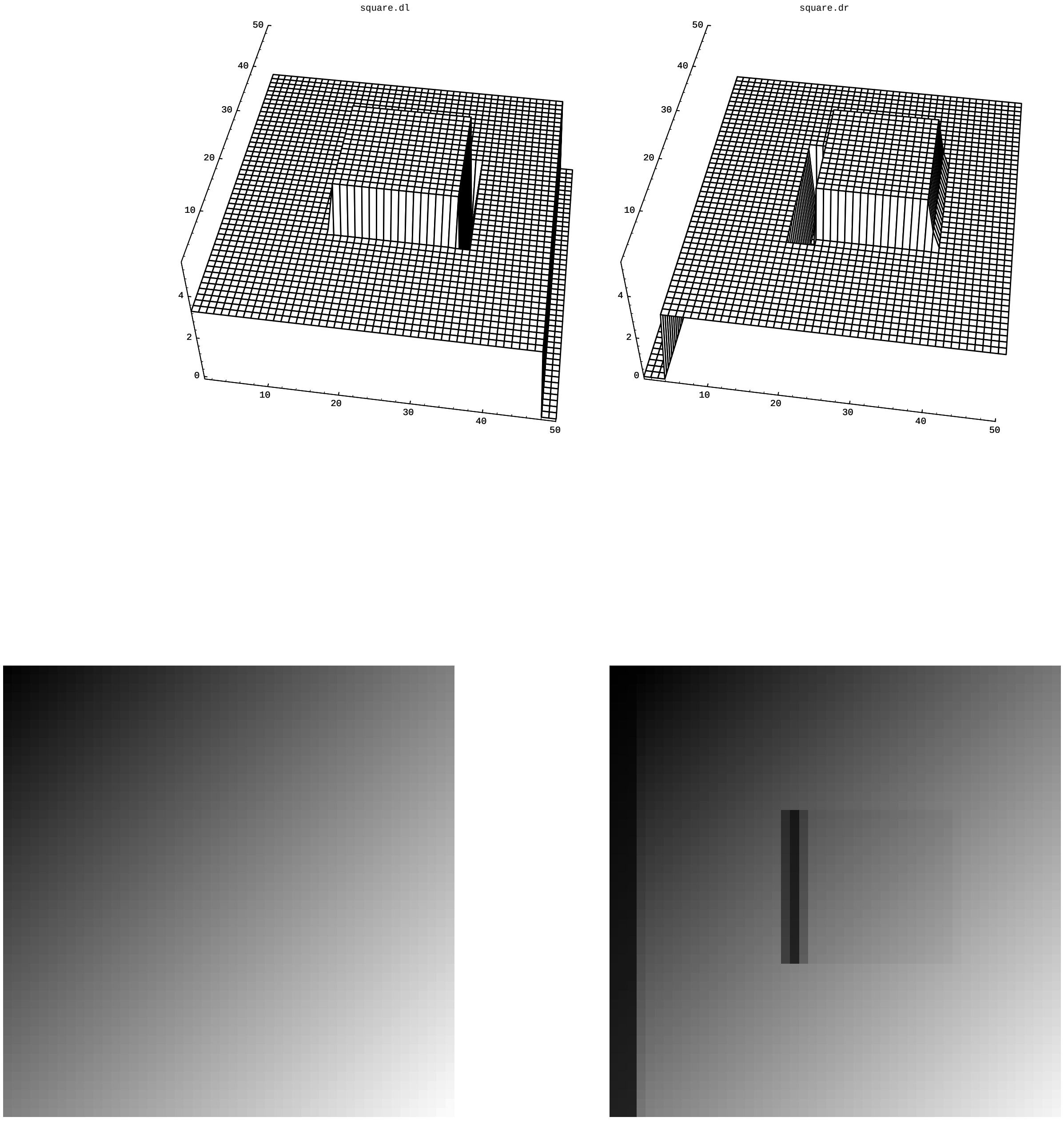


Figure 24: left and right disparity maps, with undefined areas masked out

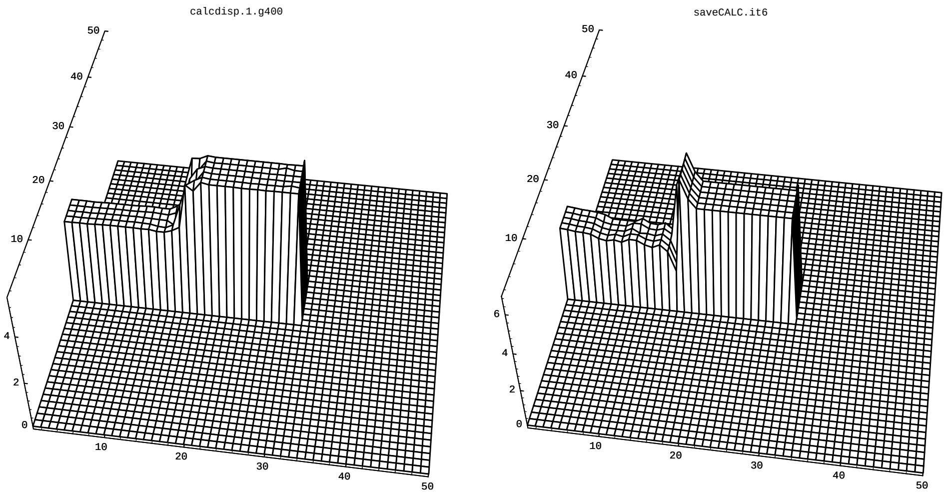


Figure 25: Disparity maps for the selected subimages. On the left, pixels not part of the onto mapping (having undefined disparities) are masked out; on the right, pixels with undefined disparities are left in.

points using the “soap-bubble” algorithm. The result was a dense starting disparity map that was within two to three pixels of the correct match. We chose 1.0 somewhat arbitrarily for σ_{noise}.

In order to establish that the algorithm worked at all on real images, we took a series of snap shots of a pattern on a blackboard. The stereo baselines used were 100mm, 105mm, 110mm, and 120mm respectively. The blackboard was 3.2 meters in front of the camera. We took two sets of images, both with friendly lighting conditions. The images are shown in Figure 26. We took the disparity values obtained from the Moravec algorithm to be a rough ground truth. Since the image plane was parallel to the blackboard, the resultant disparity map should be flat. We used a uniform displacement of 42 pixels for each trial, so that we started within a half a pixel of the correct answer for image number one, within two and a half pixels of the correct answer for image number two, etc.

The results summarized in Table 27 show several things. First of all, when starting within 0 to 4 pixels, the algorithm returned believable results. When the starting disparity was too far from correct (8 pixels), the algorithm could not register the two images, demonstrating that the correlation range for this algorithm is limited. Secondly, the results of the mean within this range were quite consistent for the two different types of lighting. Thirdly, the variance was quite high. Since the surface was flat and square to the camera, the variance should be 0. In general the variance for the variable window algorithm was higher than that of the fixed window algorithm. This is consistent with our finding on artificial data that the fixed window algorithm did as well or better in situations where there were no sharp disparity edges.

We next demonstrate the algorithm on a more complex data set. The “mask” which has a low frequency intensity derivative and started fairly near the right answer, did quite well. The disparity map does not contain big spikes and the form of the mask is recognizable improvement from the starting state. In the second experiment, there is higher frequency intensity information and we started farther from the correct answer. The algorithm fails to register the two images. (See Figures 28 and 30). ls -l

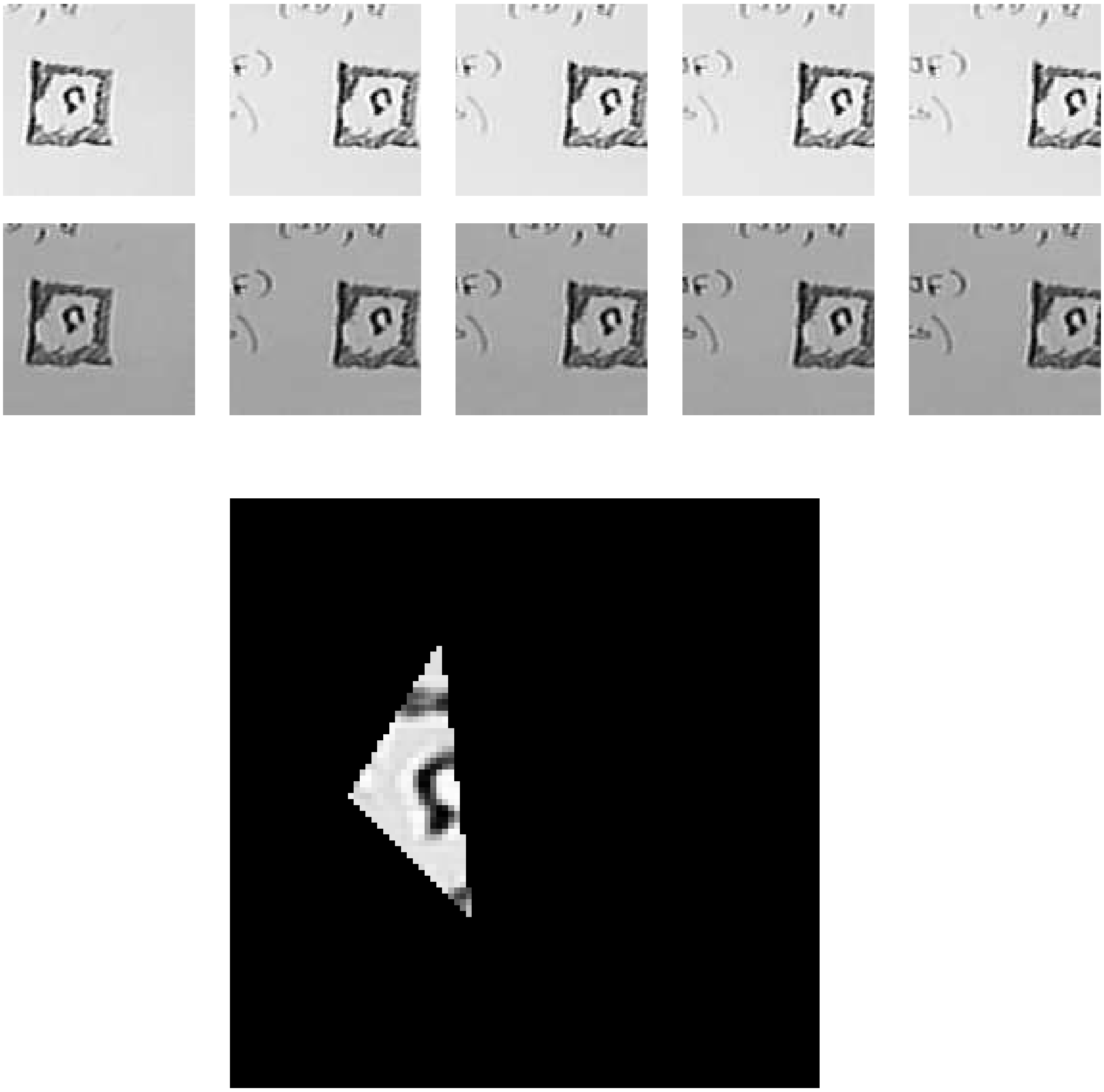


Figure 26: Above: series of images of the blackboard under one lighting condition. Middle: the same series under different lighting conditions. Below: the selected subimage.

Algorithm	Image 1		Image 2		Image 3		Image 4	
	mean	var	mean	var	mean	var	mean	var
Moravec	42	0	44	0	46	0	50	0
fixed	42.27	1.31	44.08	1.35	45.56	3.06	43.68	9.24
variable	42.31	2.20	43.98	2.97	45.64	3.48	44.48	14.64

Algorithm	Image 1		Image 2		Image 3		Image 4	
	mean	var	mean	var	mean	var	mean	var
Moravec	42	0	44	0	46	0	50	0
fixed	42.23	0.76	44.08	0.69	45.51	3.36	43.53	7.09
variable	42.19	1.92	44.11	2.67	45.48	5.22	44.41	13.39

Figure 27: The mean and variance of the selected sub-image under two different lighting conditions. Since the disparity map is flat, the variance would be zero if the algorithm worked perfectly.

7 Conclusion and Notes On Implementation

The goal of these experiments was to find a method for generating a dense disparity map for an a user-specified sub-image of arbitrary shape and size. We attempted to do this through a combination of two methods, the first being a multi-resolution search for matchable features, the second being an adaptive window for fine registration. We found that our criteria for deciding which pixels should be included in window evaluation during the registration phase, resulted in a scaled down version of the original adaptive window algorithm that was comparable in accuracy and speed to the original. We found that the algorithm was sensitive to noise, and that it would converge on the correct solution from only a small distance away. We also found that the algorithm handled sharp disparity edges well, but did not perform significantly better than the much simpler fixed window algorithm for smoothly changing disparities. We elaborate on these findings below.

We have seen that the performance of the algorithm is degraded by the addition of Gaussian noise. It seems very likely that non-Gaussian noise will further reduce the effectiveness of the algorithm. Since image differences due to surface texture, non-uniform lighting, and occlusion are systematic (non-Gaussian) this is an important area for further testing.

Although the algorithm will register two images over which are mis-

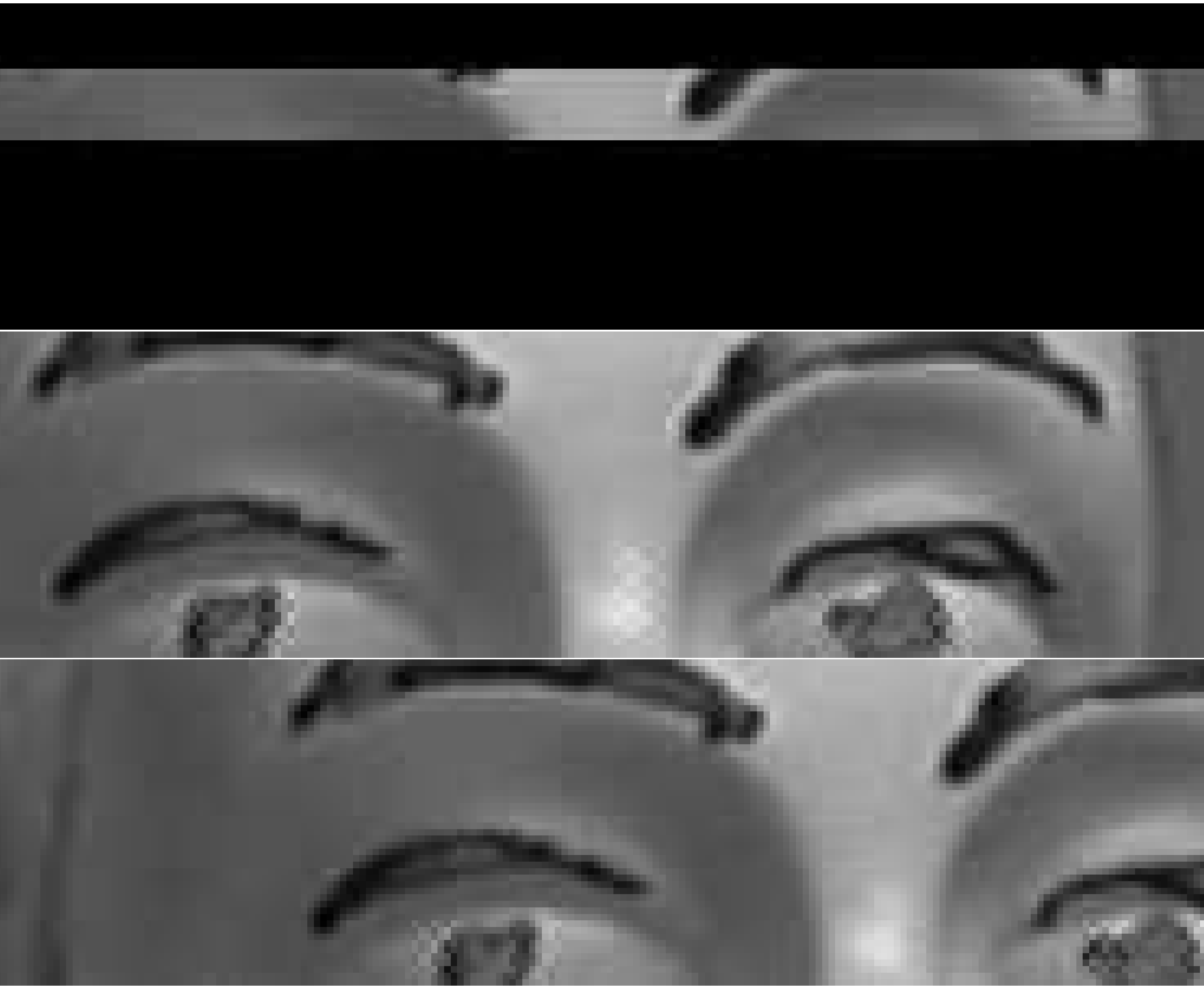
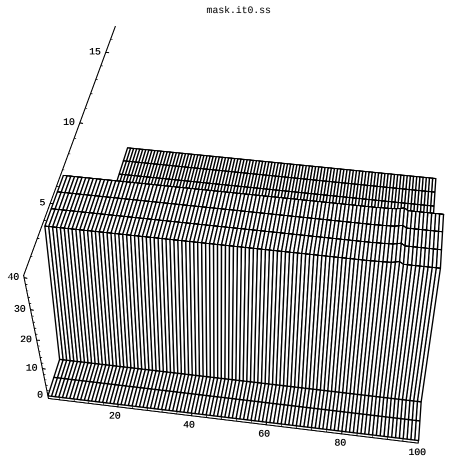


Figure 28: The selected left image, the entire left image, the entire right image.



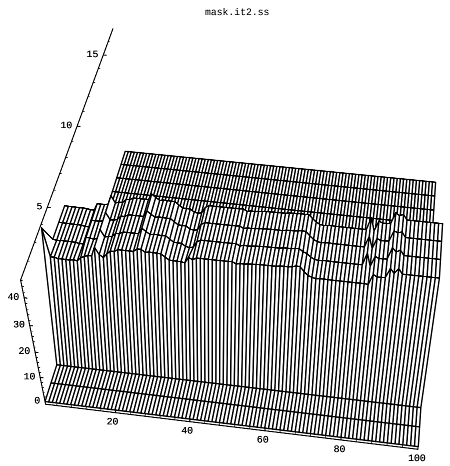
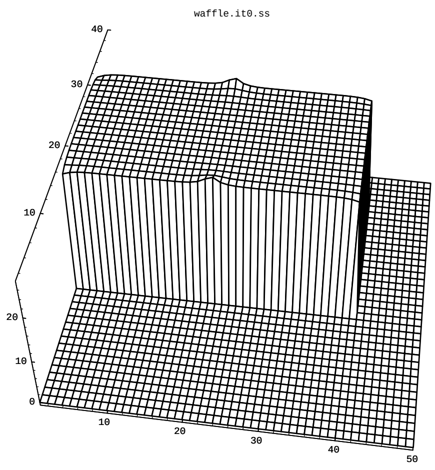


Figure 29: Top: disparities computed from Moravec algorithm, Bottom: final disparities

Figure 30: Selected subimage, and left and right complete images.



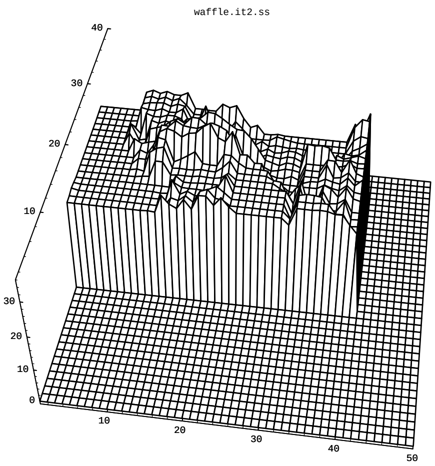


Figure 31: Average waffle size was 414 pixels per window. Running time was 9 hours and 5 minutes. 50x120 subimage.

aligned by only a small amount, the range could almost certainly be extended by using a multi-resolution scheme. By smoothing the images, the frequency could be reduced to a point where the disparity estimate is always within one half the wavelength of the highest frequency of the intensity signal. This hypothesis is supported by our results on test images which contained uniformly low frequencies. In the absence of sharp disparity edges, we tended to get nearly identical results if we started at the correct solution or displaced from it by an average of a pixel and a half, or even two pixels.

We noticed an increase in the inaccuracy of the algorithm when the disparity was changing with respect to x. In these situations it is not possible to fit a single window size over the same set of features in both images. Since intensity information is not random, the mis-aligned pixels will tend to pull the disparity estimate in one direction or the other. The effect is minimized by adaptive windows because they reduce the window size when faced with steep disparity functions, but the effect is not altogether eliminated. In these same situations, the $\frac{\delta d}{\delta x}$ is not the same in each image. Since disparity gradients are a factor in determining window size, it would make sense to use different size windows in the left and right image and scale them to some standard size before comparison. This would be comparable to warping the second image so that it has the same disparity function as the left image, thereby eliminating the error-causing misalignments.

The use of a rigid sub-image is a potential defect of our approach. The algorithm tacitly assumes that if the window is allowed to grow large enough, it will eventually contain enough visual information to make an unambiguous match. Windows around pixels at the edge of the image, may run out of image before they grow to a "good" size. Our approach to this problem was to find the best window obtainable under the circumstances, that is, to let the window grow until it reaches the image edge. In the case of sub-images, it may in fact be possible to do better than this. If the user has specified a small sub-image containing very little visual interest, it might be desirable to expand the sub-image under consideration. Of course, we would have to stop short of expanding the sub-image so much that every pixel could obtain its optimal window size, as this would degenerate into an analysis of the entire image. Under such a scheme, the user-specified sub-image could be treated as a suggestion; the algorithm could then determine an appropriate sub-image based on the user's suggestion as well as the local intensity and disparity information, and then work completely within that region.

Another aspect of our implementation which could stand improvement is our computation of $\hat{\alpha_d}$.

$$\hat{\alpha_d} = 1/ \mid W \mid \sum_{i,j \in W} \frac{(d(\xi_i, \eta_j) - d(0,0))^2}{\sqrt{{\xi_i}^2 + {\eta_j}^2}}$$

Since the algorithm processes a discretely sampled image, we have 0/0 when $(x, y) = (i, j)$. Although we could use L'hôpital's rule to evaluate this expression, this is tedious and not always resolve the problem. We tried several several constants for the denominator $.25, .5, 1$ as well as omitting this case in the summation. Although we found little difference between the constants we settled on .5. A better implementation [4] would be to interpolate the image so we had values for every 1/2 pixel. Then such summations could be evaluated on the 1/2 pixel thereby avoiding the degenerate case. Since interpolation must be performed to compute $\hat{\alpha_f}$ and $\hat{\Delta d}$ in any case, this seems like a better method.

One of the major drawbacks of the algorithm is its slowness. At roughly one second per pixel on a Sun4, the algorithm takes three days to run on a single NTSC size image. We made no serious attempt at optimization and some of the computation could be easily reduced. For example, with the exception of the smallest size window, we always know $\hat{\alpha_d}$ and $\hat{\alpha_f}$ $\sigma_{\Delta d}$ at the next smallest window size when we compute these values. If we could take advantage of this by computing them incrementally we could save a lot of work. This can be done for $\hat{\alpha_d}$ and $\hat{\alpha_f}$ as follows.

$$\alpha_{n+\delta W} = \frac{n\alpha_n + \mid \delta W \mid \alpha_{\delta W}}{\mid n + \delta W \mid}$$

where $\mid W \mid$ is the number of points in window W. However, we were not able to find a comparable expression for $\sigma_{\Delta d}$ whose derivative involves an expression requiring evaluation over the entire window. However it might well be worth approximating this expression to achieve the speedup.

Another approximation of the algorithm which could produce significant speedups would be possible if we took advantage of the fact that almost all of the benefit of the adaptive window comes where the disparity changes sharply. Since these locations usually appear as edges in the scene, it would

[4] due to Richard Wallace

make sense to begin with an edge-based stereo-matching algorithm rather than an intensity based algorithm, and triangulate the resulting sparse depth map. Since every face of the triangulated surface is a plane, the value $\hat{\alpha_d}$ will be nearly identical for all values on that face. If we approximate $\hat{\alpha_f}$ on the face by taking the average value over the face, a single, approximately optimal, window could be be used for each pixel on the face. The cost for computing the window size to be used for the face would be the same as the cost for computing the window size for a single pixel using the present algorithm. If we add that windows not be allowed to cross edges, we would have a much less expensive approximation of the adaptive window algorithm.

Although there are many aspects of the adaptive window algorithm which could be further tested, optimized, or improved, there is a real question whether this is the right approach to take. If we look at the algorithm as something that takes in a fairly accurate but sparse disparity map and returns a dense and more accurate one, we should compare it to other techniques for achieving the same end. It may well be that a sophisticated interpolation of the sparse depth map is more robust and more accurate or less computationally expensive way of solving the same problem.

References

[1] J. K. Aggarwal and J. Rodriguez. Quantization error in stereo imaging. *Computer Vision and Pattern Recognition*, June, 1988.

[2] P.J. Besl and L. T. Birch, J. B.and Watson. Robust window operators. In *Proceedings 2nd International Conference on Compututer Vision*, 1988. pp591-560.

[3] R. Bolle and B. Vemuri. On three-dimensional surface reconstruction methods. *IEEE Transactions Pattern Analyssi and Machine Intelligence*, PAMI(13):1, 1991.

[4] Robert C. Bolles, Harlyn H. Baker, and Marsha Jo Hannah. The jisct stereo evaluation. In *Proceedings: Image Understanding Workshop*, April 1993.

[5] V.N. Dvornychenko. Generalized correlation measures for use in signal and image processing. In *Conference on Pattern Recognition and Image Processing*, August 1981. pp.554-561.

[6] Foley, vanDam, Feiner, and Hughes. *Computer Graphics Principles and Practice*. Addison-Wesley, 1990.

[7] W. E. L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:1:17–34, 1985.

[8] N. C. Griswold and Yeh C. P. A new stereo vision model based upon the binocular fusion concept. *Computer Vision Graphics and Image Processing*, 41:153–171, 1988.

[9] Fan Jiang and B. Schunk. Recovering absolute depth and motion of multiple objects from intensity images. *SPIE, Stochastic and Neural Methods in Signal Processing Image Processing and Computer Vision*, 1991.

[10] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. In *IEEE International Conference on Robotics and Automation*, April 1991.

[11] Kim and Agarwal. Finding range from stereo images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1985.

[12] Martin D. Levine, Douglas A. O'Handley, and Gary M. Yagi. Computer determination of depth maps. *Computer Graphics and Image Processing*, 2(4):131–150, 1973.

[13] David Marr. *Vision*. W. H. Freeman and Company, 1982.

[14] J. L. Marroquin. Surface reconstruction preserving discontinuities. Technical report, M.I.T., publications@ai.mit.edu, August, 1984.

[15] Larry Matthies and Steven Shafer. Error modeling in stereo navigation. *IEEE Journal of Robotics and Automation*, pages 239–48, 1987.

[16] Larry Matthies, Richard Szeliski, and Tokeo Kanade. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3:209–236, 1989.

[17] Mori, Kidode, and Asada. An iterative prediction and correction method for automatic stereo comparison. *Computer Graphics and Image Processing*, 3:393–341, 1973.

[18] Hans P. Movavec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, Computer Science Department, Stanford University, September 1980.

[19] Shree Nayar. Surface reflection: Physical and geometrical perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 13, No. 7, July:611 to 634, 1991.

[20] K. Okutomi and M. Kanade. A locally adaptive window for signal matching. *International Journal of Computer Vision*, 7:143–162, 1992.

[21] T. Okutomi M., Kanade. A multiple baseline stereo. Technical report, Carnegie Mellon University, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, December 1990.

[22] R.Nevata R. Chung. Use of monocular groupings and occlusion analysis in a heirarchical stereo system. *Computer Vision and Pattern Recognition*, 1991.

[23] S. Reichenback and S. Park. Artificial scenes and simulated imaging. *SPIE, Stochastic and Neural Methods in Signal Processing Image Processing and Computer Vision*, 1991.

[24] T. Ryan, R. Grey, and B. Hunt. Prediction of correlation errors in stereo images. *Optical Engineering*, Volume 19(3), May/June:312–322, 1980.

[25] Grahame B. Smith. The relationship between image irradiance and surface orientation. In *IEEE CVPR 83*, 1983.

[26] Thomas M. Strat and Martin A. Fischler. One-eyed stereo: A general approach to modeling 3-d scene geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8 No. 6:730–741, 1986.

[27] R. Szeliski and G. Hinton. Solving random-dot stereograms using the heat equation. In *IEEE Computer Society Conference of Computer Vision and Pattern Recognition*, June 1985. pp.284-288.

[28] Liang Chen Terry Boult. Analysis of two new stereo algorithms. In *Computer Vision and Pattern Recognition*, June 1988. pp. 177-182.

[29] Demetri Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):413–424, 1986.

[30] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, Volume RA-3, No. 4, August:323 to 344, 1987.

[31] Vadiya. Stereopsis and image registration from extended edge features in the absence of camera pose information. *Computer Vision and Pattern Recognition*, June, 1991.

[32] Holly Wenaas. Computer generated correlated noise images for various stochastical distributions. *SPIE, Stochastic and Neural Methods in Signal Processing Image Processing and Computer Vision*, 1991.

[33] Yakimovsky Y. and Cunningham R. A system for extracting three-dimensional measurements from a stereo pair of tv cameras. *Computer Graphics and Image Processing*, 7:195–210, 1978.