The Stereo Problem

Gunnar Farnebäck

Computer Vision Laboratory
Linköping University, Sweden

February 2001
Revised December 2002

1 Introduction

Consider the pair of images in figure 1. At a first glance they may look identical but closer inspection shows small variations. In fact this is a stereo pair, a pair of images obtained from slightly different views, just like our left and right eyes see almost but not exactly the same things. As is discussed in the next section, and should be intuitively clear, the differences between the two images give information about the distance to the objects in the scene. To determine these distances, also called depths, from a pair of images is called the stereo problem.

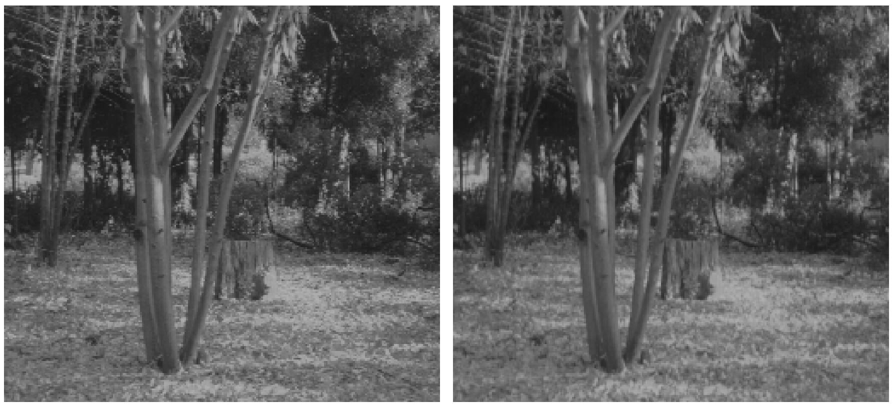

Figure 1: Stereo pair.

As is almost always the case in this area, the human vision excels at the task of depth estimation. Stereo is not the only cue to do this, however. Other important information comes from the ability to vary the focus and the relative angle between the directions the eyes are looking. Even from just one of the two images in figure 1, a human can tell which objects are close and which are further away. This is primarily due to contextual knowledge. We can segment the image into consistent objects and we know that an occluding object ought to be in front of an occluded one. We can also identify a part of the image as ground and expect it to be essentially planar and horizontal. Local variations in texture and shading also give depth information, but this should probably count as estimation of object shape, rather than of depth as such.

That a human can perceive depth from only stereo information is well illustrated by random dot stereograms. These are constructed in such a way that the left image is entirely random and the right image is created from the left one by shifting parts of it in a way which is consistent with the desired stereo effect. By glaring at the two images from a certain distance, the viewer can fuse them into a single image which exhibits a depth variation.

In the remaining sections of this text we explore how depth can be estimated from stereo, in a computer vision setting. We start in section 2 with an investigation of the geometry of stereo vision, using the model of two parallel pinhole cameras. In section 3 we discuss common approaches to disparity estimation and some of the difficulties these have to deal with. Sections 4 and 5 introduce two generally useful signal processing techniques, polynomial expansion and normalized averaging, in preparation for section 6, where we derive a simple disparity estimation algorithm in detail. More information about the methods presented in sections 4–6 can be found in [1].

2 The Geometry of Stereo Vision

2.1 The Pinhole Camera

In order to study the stereo problem we need to explore its geometric properties and have a model for the cameras used to obtain the images. Beginning with the latter we assume we have a camera with a thin lens and use the simplest approximation suitable to model perspective projection; the pinhole camera. This is illustrated in figure 2. The camera coordinate system (x, y, z) is chosen so that the *image plane* coincides with the x, y-plane and with the center of the image plane at the origin. Thus the *optical axis* axis coincides with the z axis and we have the *optical center* at coordinate $(0, 0, f)$. f is known as the *focal length* and is a parameter of the camera. The image is formed by light rays, either being emitted or reflected from objects in the scene, being projected through the optical center down onto the image plane as shown in figure 2. A point at world coordinate (X, Y, Z) is projected onto camera coordinate $(x, y, 0)$, where we for simplicity assume both coordinate systems coinciding. The relation between the coordinates can easily be found by the use of similar triangles, as shown in figure 3, giving

$$\frac{x}{f} = -\frac{X}{Z - f}, \tag{1}$$

$$\frac{y}{f} = -\frac{Y}{Z - f}. \tag{2}$$

Notice the minus signs. These are caused by the fact that the pinhole camera produces an image which is "upside down" or to be more exact rotated 180° compared to how we see the scene. A useful trick to avoid this, which we will make use of henceforth, is to place the image plane in front of the optical center instead of behind. Of course the physical motivation for the pinhole camera model breaks after this modification, but mathematically it only leads to a simplifying sign shift and in practice it means that we no longer have to rotate the obtained image by 180° to recognize the scene.

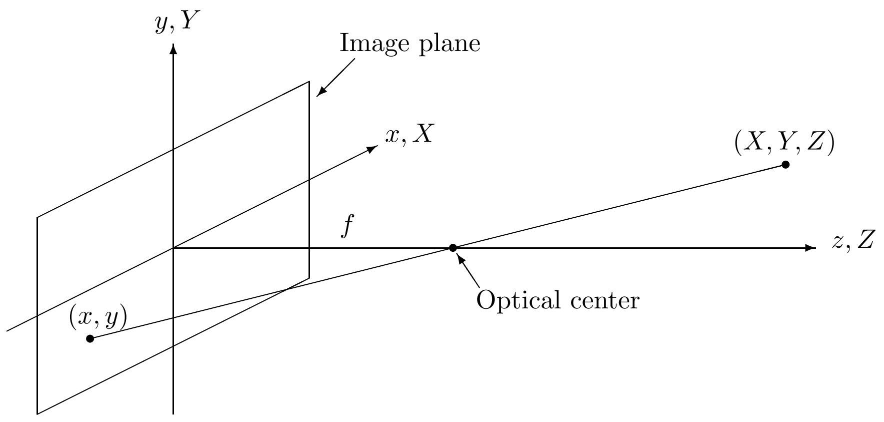

Figure 2: Pinhole camera.

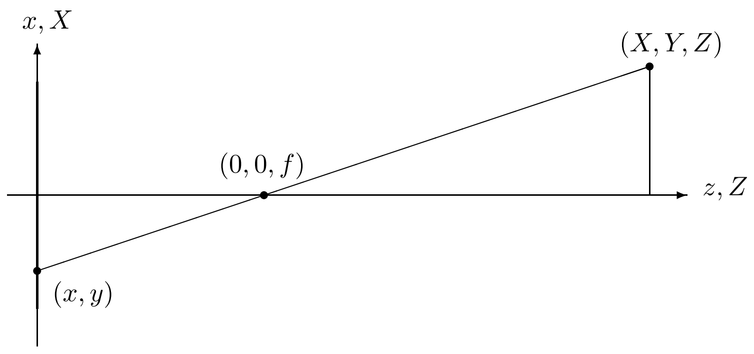

Figure 3: Projection of figure 2 onto the x, z-plane. The two triangles are similar.

2.2 Elementary Stereo Geometry

To obtain an elementary stereo geometry we set up two identical pinhole cameras in the canonical configuration shown in figure 4. This configuration is characterized by the *baseline*, the line connecting the two optical centers, being aligned with the x axes of the cameras and both optical axes being parallel. The world coordinate system does no longer coincide with either camera coordinate system, instead we have the origin of the left camera at world coordinates $(-h, 0, f)$ and the origin of the right camera at $(h, 0, f)$, where $2h$ is the distance between the optical centers of the two cameras. Again we can use similar triangles to obtain the two relations[1]

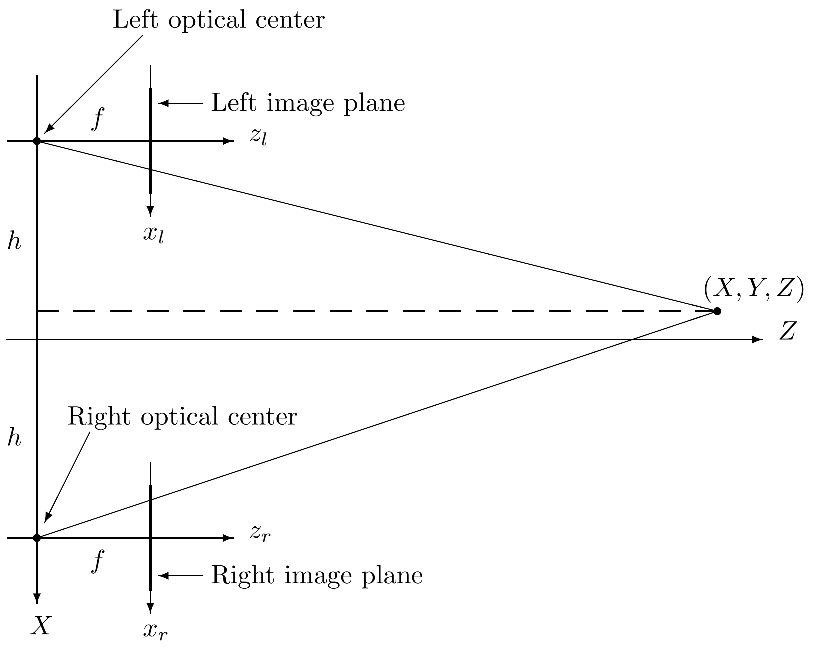

Figure 4: Canonical stereo configuration.

$$\frac{h+X}{Z} = \frac{x_l}{f}, \tag{3}$$

$$\frac{h-X}{Z} = \frac{-x_r}{f}. \tag{4}$$

Summing these gives us

$$\frac{2h}{Z} = \frac{x_l - x_r}{f}. \tag{5}$$

We now introduce the *disparity* d, defined as

$$d = x_l - x_r \tag{6}$$

and get the following two relations between depth and disparity,

$$Z = \frac{2hf}{d}, \tag{7}$$

$$d = \frac{2hf}{Z}. \tag{8}$$

[1]Notice that the geometry guarantees that the y coordinates in the left and right images are identical.

An important conclusion from these equations is that with this camera configuration, depth is simply inversely proportional to disparity. Therefore we will concentrate on the problem of estimating disparity in the following sections, which can be done directly from the left and right images.

Another important observation from equation (8) is that if we know a priori the minimum and maximum distances to objects in the scene, we can also a priori compute upper and lower bounds for the disparity,

$$d_{\min} = \frac{2hf}{Z_{\max}}, \tag{9}$$

$$d_{\max} = \frac{2hf}{Z_{\min}}. \tag{10}$$

Even if we do not know $Z_{\max}$, we still have the relation $d \geq 0$.

3 Estimating Disparity

3.1 Common Approaches

Disparity estimation is in principle the problem of finding corresponding points in the two images of a stereo pair, where we are limited to translations along the x axis. To do this automatically is a classic computer vision problem and a large number of algorithms have been published, each with their own strengths and weaknesses. The algorithms can broadly be divided into three groups:

- Correlation-based
- Feature-based
- Local structure-based

Algorithms in the first group essentially try to find corresponding pixels in the two images directly by comparing intensity values. A typical example is block matching. Obviously matching intensity values of single pixels would be too ambiguous so the matching is done for entire neighborhoods or blocks, typically about 5×5 or 7×7 pixels large. The blocks from one image are then compared with blocks in the other image at positions which have been translated different numbers of pixels. The similarity of two blocks can be measured e.g. by standard correlation or some least squares measure. The translation giving the most similar blocks of course determines the disparity at the point. Block matching is relatively simple but tends to be slow.

The algorithms in the second group take a more high-level approach. The first step is to identify points or sets of points with characteristic features, e.g. edges, lines, or corners. After this has been done for both images, it is still necessary to find how the feature points correspond. The difference compared to doing this directly on intensity values is that now there are much fewer correspondencies to find and there may also be additional information to help the matching, such as the length or orientation of an edge. Algorithms in this group tend to be complex but are often more robust than correlation-based methods and allow higher precision.

The last group consists of algorithms which avoid the search for correspondencies altogether. Instead they analyze the signal locally at the same point in both images and try to determine how much it has moved. A typical example from this group is phase-based disparity estimation. The basic idea is simple. Consider two sinusoid signals which are identical except for a small shift. If we can estimate the local phase of both signals at some point and the frequency of the signal, we can determine the disparity from this information. Algorithms in this group are usually fast and allow very high precision. A weakness is that they can break down if the assumptions about the local structure do not hold. For example phase-based algorithms assume locally narrow-banded signals. Another weakness is that the accuracy tends to be much lower for large disparities than for small ones.

3.2 General Difficulties

The stereo correspondence problem is inherently ambiguous and can not be solved in general. The most extreme case is a scene only containing a large, flat, non-textured object, giving two images of uniform brightness. There is no information at all available to determine the disparity so any value is as plausible as another.

Another, usually less drastic, problem is self-occlusion. This happens when a point on an object in the scene can be seen from one camera but is occluded from the other camera. As simple example is a perfect sphere, where it is clear that the two cameras will see slightly different sections of its surface. For the sphere this is usually not a big problem in practice since the self-occlusion tends to occur in a very small area, quite possibly only a fraction of a pixel in the image. The extreme case of self occlusion is a scene containing a very thin flat object, placed between the cameras in such a way that they see two different surfaces. Then there does not need to be any relation at all between the two images.

A third problem is a kind of aliasing caused by repetitive patterns. In general we cannot distinguish between disparity candidates differing by a multiple of the period of the pattern. Many times this problem can be resolved by a multi-scale approach. The idea is that altough a pattern may have a repetitive fine structure, there may be less ambiguity at a coarser scale. In fact it is generally a good idea to start with analysis at a coarse scale and refine the disparity estimates at finer scales, regardless which kind of algorithm is being used. Usually this helps to improve either robustness or speed of an algorithm, and sometimes both.

4 Polynomial Expansion

In section 6 we will derive a complete disparity estimation algorithm, which in the terminology of the previous section is local structure-based. As the first step of the signal analysis we do a local polynomial expansion in a neighborhood of each pixel in the image. To simplify the presentation we assume that the image is discretized with values at all integer coordinates and to begin with we consider the expansion in a neighborhood of the origin.

One rationale for doing a polynomial expansion is the Maclaurin theorem. This theorem states, e.g., that any sufficiently well behaved function of two variables in a neighborhood of the origin can be approximated by a second degree polynomial,

$$\begin{aligned} f(x,y) = {} & f(0,0) + \frac{\partial f}{\partial x}(0,0)x + \frac{\partial f}{\partial y}(0,0)y \\ & + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(0,0)x^2 + \frac{1}{2}\frac{\partial^2 f}{\partial y^2}(0,0)y^2 + \frac{\partial^2 f}{\partial x \partial y}(0,0)xy + O((x^2+y^2)^{\frac{3}{2}}). \end{aligned} \tag{11}$$

This is, however, not immediately useful in the context of discretized images. The first problem is that the rest term only is guaranteed to be very small in some sufficiently small neighborhood of the origin. Since such a neighborhood may cover only a single pixel, we would not have much use for the expansion. The second problem is that we strictly cannot even compute the partial derivatives at the origin, needed to obtain the coefficients in the polynomial expansion. The difficulty is that the mathematical definition of derivatives involves a limit, e.g.

$$\frac{\partial f}{\partial x}(0,0) = \lim_{x \to 0} \frac{f(x,0) - f(0,0)}{x}, \tag{12}$$

which requires knowledge of function values arbitrarily close to the origin, which we simply do not have.[2]

We can avoid both these complications by choosing a different method for determining the coefficients in the polynomial expansion. More specifically we use quadratic polynomials,

$$f(x,y) \sim p(x,y) = r_1 + r_2 x + r_3 y + r_4 x^2 + r_5 y^2 + r_6 xy, \tag{13}$$

and determine the coefficients $r_1, \ldots, r_6$ by a least squares minimization

$$\arg \min_{r_1,\ldots,r_6} \sum_{x,y} (w(x,y)(f(x,y) - p(x,y)))^2, \tag{14}$$

where $w(x,y)$ gives different weights to the pixels around the origin. We will limit ourselves to use Gaussian weights truncated to an $N \times N$ window, i.e.

$$w(x,y) = \begin{cases} e^{-\frac{x^2+y^2}{2\sigma^2}}, & |x| \leq \frac{N-1}{2},\ |y| \leq \frac{N-1}{2}, \\ 0, & \text{otherwise}, \end{cases} \tag{15}$$

where $N \geq 3$ is odd and decides the size of the neighborhood for which we do the approximation. The standard deviation of the Gaussian, σ, should not be too large compared to N, i.e. the truncated values outside the $N \times N$ window should be small.

In order to solve this least squares problem, we first transform it into a linear algebra setting. This is done by considering the $N \times N$ values in our neighborhood as an N^2-dimensional vector $\mathbf{f}$, e.g. by stacking the columns onto each other. The monomials in the expansion (13) are also considered as vectors $\mathbf{b}_1, \ldots, \mathbf{b}_6$. We illustrate this for $N = 3$, i.e. a 3×3 neighborhood, which gives

[2] It should be pointed out that image derivatives still can be approximated in various ways and that these approximations often are useful.

a vector space of dimension 9.

$$\mathbf{f} = \begin{pmatrix} f(-1,-1) \\ f(-1,\ \ 0) \\ f(-1,\ \ 1) \\ f(\ \ 0,-1) \\ f(\ \ 0,\ \ 0) \\ f(\ \ 0,\ \ 1) \\ f(\ \ 1,-1) \\ f(\ \ 1,\ \ 0) \\ f(\ \ 1,\ \ 1) \end{pmatrix}, \mathbf{b}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{b}_2 = \begin{pmatrix} -1 \\ -1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{b}_3 = \begin{pmatrix} -1 \\ 0 \\ 1 \\ -1 \\ 0 \\ 1 \\ -1 \\ 0 \\ 1 \end{pmatrix},$$
$$\mathbf{b}_4 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{b}_5 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \mathbf{b}_6 = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \\ 1 \end{pmatrix}. \tag{16}$$

In order to realize the pointwise multiplication with the weights in equation (14) we put these into the diagonal of a matrix $\mathbf{W}$, which in the 3×3 case would look like

$$\mathbf{W} = \begin{pmatrix} w(-1,-1) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w(-1,0) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w(-1,1) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w(0,-1) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w(0,0) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & w(0,1) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & w(1,-1) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & w(1,0) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w(1,1) \end{pmatrix}. \tag{17}$$

Finally we collect the expansion coefficients $r_1, \dots, r_6$ into a vector $\mathbf{r}$ and stack the monomials $\mathbf{b}_1, \dots, \mathbf{b}_6$ next to each other in a basis matrix $\mathbf{B}$,

$$\mathbf{r} = \begin{pmatrix} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 \end{pmatrix}^T, \quad \mathbf{B} = \begin{pmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_6 \\ | & | & & | \end{pmatrix}. \tag{18}$$

This allows us to express equation (14) as

$$\arg\min_{\mathbf{r}} \|\mathbf{W}(\mathbf{Br} - \mathbf{f})\|^2. \tag{19}$$

It is now useful to remind ourselves of the fact that the least squares problem

$$\arg\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|^2 \tag{20}$$

can be solved by the normal equations [3]

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}. \tag{21}$$

With $\mathbf{A} = \mathbf{WB}$ and $\mathbf{b} = \mathbf{Wf}$, this gives the solution of equation (19) as [4]

$$\mathbf{r} = (\mathbf{B}^T\mathbf{W}^2\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}^2\mathbf{f}. \tag{22}$$

We now have an effective algorithm for computing a polynomial approximation of an arbitrary function in a neighborhood of the origin. We can repeat this in the neighborhood of any pixel by introducing a local coordinate system through translation of the global coordinate system so that the origin is placed at the point of interest.

To summarize, all this gives us a kind of transform, which maps the gray-value image $f(x, y)$ onto six values per point $\mathbf{r}(x, y)$, so that for each neighborhood around (x_0, y_0) we have an approximation

$$\begin{aligned} f(x,y) \sim\ & r_1(x_0,y_0) + r_2(x_0,y_0)\,(x-x_0) + r_3(x_0,y_0)\,(y-y_0) \\ & + r_4(x_0,y_0)\,(x-x_0)^2 + r_5(x_0,y_0)\,(y-y_0)^2 \\ & + r_6(x_0,y_0)\,(x-x_0)(y-y_0), \end{aligned} \tag{23}$$

which is optimal in a weighted least squares sense. It may seem like this would be a computationally very demanding operation, and with a naive implementation of the algorithm above it would. With further investigation, however, it turns out that the computations all boil down to a set of convolutions, which can be very efficiently implemented with a hierarchical scheme of separable convolutions. This is outside the scope of this presentation, but can be found in [1]. An illustration of the polynomial expansion of a neighborhood from the left stereo image in figure 1 is given in figure 5.

5 Normalized Averaging

Normalized averaging is a technique to process uncertain signals. The concept of uncertainty means that we sometimes do not know the signal value at a point, or that we do not fully trust it. One example may be a sensor which has a few defect elements, and we know where they are. Another example is pixels off the edge of an image. Many times the signal is arbitrarily set to zero at these points, but that is not a good solution if we want to process the signal further. What we should do is to keep signal values f and their certainties c separate; this is called the signal/certainty principle. Where we have full knowledge of the signal, the certainty is set to one. Where we have no knowledge at all, the certainty is set to zero and the value may be arbitrary. Intermediate certainty values can also be used.

Like the polynomial expansion in the previous section, normalized averaging is computed pointwise from the values in some neighborhood of each point.

[3] If the columns of $\mathbf{A}$ are linearly dependent, $\mathbf{A}^T\mathbf{A}$ will be singular and in fact there is not a unique solution $\mathbf{x}$ to the minimization problem. We ignore this complication since it will not happen in this application.

[4] Whenever something is written on the form $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, it should be interpreted as $\mathbf{x}$ being the solution to $\mathbf{Ax} = \mathbf{b}$. In practice this is almost never computed by means of an explicit inverse.

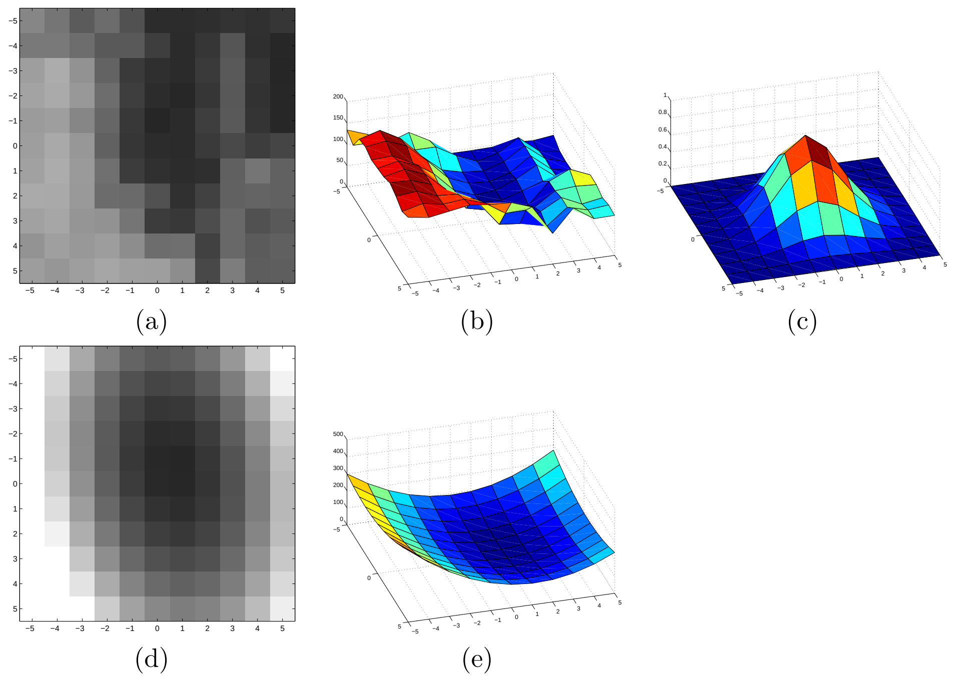

Figure 5: An image neighborhood, shown in gray-scale in (a) and as a surface in (b), is approximated by a quadratic polynomial, using the Gaussian in (c) as weight function. The resulting coefficients are $r_1 = 42.9$, $r_2 = -10.6$, $r_3 = 4.6$, $r_4 = 7.8$, $r_5 = 2.8$, and $r_6 = -1.5$. The corresponding polynomial is shown as a gray-scale image in (d) and as a surface in (e).

Here too this neighborhood is defined by a weighting function a, usually called applicability, which determines the relative importance of points at varying distances from the center.

Remember that the weighted average of a set of values $x_1, \dots, x_n$ with associated weights $w_1, \dots, w_n$ is defined as

$$x_{\text{avg}} = \frac{\sum_k x_k w_k}{\sum_k w_k}. \tag{24}$$

In normalized averaging we have two distinct weights, the certainty c which follows the signal and the applicability a which follows the neighborhood. These are simply multiplied and in the one-dimensional case we compute normalized averaging at a point x as

$$f_{\text{avg}}(x) = \frac{\sum_k f(x-k)c(x-k)a(k)}{\sum_k c(x-k)a(k)}. \tag{25}$$

Over the whole signal we recognize this as a quotient of two convolutions, thus

$$f_{\text{avg}} = \frac{(fc) * a}{c * a}, \tag{26}$$

which we use to define normalized averaging for signals of any dimensionality. Further discussion on normalized averaging and a good example of how it can be used on image data can be found in chapter 9.6 of [2] and in [1].

6 A Disparity Estimation Algorithm

6.1 Key Observation

We have now made all necessary preparations for deriving a disparity estimation algorithm. Assume for a moment that we have a right image containing an exact quadratic polynomial

$$f_r(\mathbf{x}) = p(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c. \tag{27}$$

This is the same model as equation (13) with

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}, \ \mathbf{A} = \begin{pmatrix} r_4 & \frac{r_6}{2} \\ \frac{r_6}{2} & r_5 \end{pmatrix}, \ \mathbf{b} = \begin{pmatrix} r_2 \\ r_3 \end{pmatrix}, \ c = r_1. \tag{28}$$

Construct the left image from the right image[5] by a global translation $\mathbf{d}$ and expand the new polynomial

$$\begin{aligned} f_l(\mathbf{x}) = p(\mathbf{x}-\mathbf{d}) &= (\mathbf{x}-\mathbf{d})^T\mathbf{A}(\mathbf{x}-\mathbf{d}) + \mathbf{b}^T(\mathbf{x}-\mathbf{d}) + c \\ &= \mathbf{x}^T\mathbf{A}\mathbf{x} + (\mathbf{b}-2\mathbf{A}\mathbf{d})^T\mathbf{x} + c + \mathbf{d}^T\mathbf{A}\mathbf{d} - \mathbf{b}^T\mathbf{d} \\ &= \mathbf{x}^T\mathbf{A}\mathbf{x} + \tilde{\mathbf{b}}^T\mathbf{x} + \tilde{c}, \end{aligned} \tag{29}$$

[5]It may seem backwards to construct the left image fram the right image rather than the other way round. The reason for this is to become consistent with the disparity definition in equation (6), where x_r is subtracted from x_l in order to get positive disparity values.

where the new coefficients $\tilde{\mathbf{b}}$ and $\tilde{c}$ are given by

$$\tilde{\mathbf{b}} = \mathbf{b} - 2\mathbf{A}\mathbf{d}, \tag{30}$$

$$\tilde{c} = c + \mathbf{d}^T\mathbf{A}\mathbf{d} - \mathbf{b}^T\mathbf{d}. \tag{31}$$

The key observation is that by equation (30) we can formally solve for the translation $\mathbf{d}$ as

$$\mathbf{d} = -\frac{1}{2}\mathbf{A}^{-1}(\tilde{\mathbf{b}} - \mathbf{b}). \tag{32}$$

6.2 Practical Disparity Estimation

Obviously the assumptions about the entire image being a single polynomial and the left and right images being related by a global translation are unrealistic. The remaining work is to transform the observation into a practically useful algorithm, which works for real images.

Rather than assuming that the entire image can be described by one polynomial, we do this locally in each neighborhood. This is exactly why we did the polynomial expansion in section 4. It is still optimistic to hope that the typical neighborhood will fit a polynomial perfectly of course, but this is not required. The real assumption is that the polynomial approximation captures relevant information about the neighborhood, and that this information is sufficient for our purposes, i.e. to estimate a disparity. This is nothing we can be certain of a priori. As any signal model it has to be validated by experiments. Naturally we also assume a space variant disparity, trying to obtain one disparity estimate for each pixel.

With this is mind, we start by doing a polynomial expansion of both the left and the right images, giving us $\mathbf{A}_l(x, y)$, $\mathbf{b}_l(x, y)$, and $c_l(x, y)$ for the left image and $\mathbf{A}_r(x, y)$, $\mathbf{b}_r(x, y)$, and $c_r(x, y)$ for the right image. $\mathbf{A}$, $\mathbf{b}$, and c of course relate to the expansion coefficients $\mathbf{r}$ as in equation (28). In the ideal case given by equation (29) we see that $\mathbf{A}_l(x, y)$ and $\mathbf{A}_r(x, y)$ should be equal, but in practice this is nothing we can rely on. As a practical solution we use the arithmetic mean,

$$\mathbf{A}(x, y) = \frac{\mathbf{A}_l(x, y) + \mathbf{A}_r(x, y)}{2}. \tag{33}$$

There are less difficulties with $\mathbf{b}$ and we can directly use $\mathbf{b}_r$ as $\mathbf{b}$ and $\mathbf{b}_l$ as $\tilde{\mathbf{b}}$. To simplify later steps of the algorithm we introduce

$$\mathbf{\Delta b}(x, y) = -\frac{1}{2}(\mathbf{b}_l(x, y) - \mathbf{b}_r(x, y)). \tag{34}$$

This turns equations (30) and (32) into

$$\mathbf{A}(x, y)\mathbf{d}(x, y) = \mathbf{\Delta b}(x, y), \tag{35}$$

$$\mathbf{d}(x, y) = \mathbf{A}(x, y)^{-1}\mathbf{\Delta b}(x, y). \tag{36}$$

In principle we should now be able to obtain a disparity estimate at (x, y) from equation (36). Notice that this gives a 2D displacement $\mathbf{d}$, which may have a non-zero y component. Since the camera geometry guarantees that we have a displacement only in the x direction, the disparity, we can use the relative size of the y component as a confidence measure.

Unfortunately, and not very surprising, these estimates turn out to be too noisy and uncertain to really be useful. There is also the problem that $\mathbf{A}(x,y)$ may be singular or close to singular. The result of doing this operation on the stereo pair in figure 1 is shown in figure 6. To get further we need to make an assumption about the disparity field, namely that it is only slowly varying. This assumption means that we can reduce the noise in the disparity estimates by averaging. The drawback is that step discontinuities, which may very well occur in images, typically at occlusion boundaries, will be smoothed out. This is a common type of tradeoff in computer vision algorithms. If it is important to correctly detect step discontinuities, some other approach must be used.

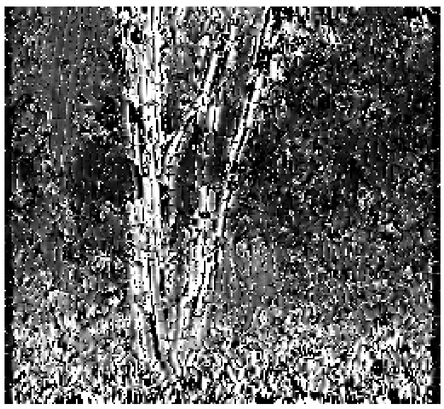

Figure 6: Disparity estimates from equation (36). Values outside the interval $[0,6]$ have been truncated. Disparity 0 is shown as black and 6 as white.

6.3 Improved Disparity Estimation

The naive way to implement the averaging would be to simply compute a local average of the pointwise disparity estimates obtained through equation (36), e.g. by filtering them with some lowpass filter. As we can see in figure 7, this gives a very unsatisfactory result. The main reason for this is that we have occasional disparity estimates which are way off, possibly several thousand pixels large or more. Such values are called outliers and in general need special consideration. As noted in equations (9) and (10) we know a priori an interval within which the disparity is guaranteed to fall and thus any disparity value outside this interval can be classified as an outlier.

This points to a better solution than the plain averaging done above, namely normalized averaging. For applicability we once more use the Gaussian from equation (15). Certainty is computed from three different sources. The first one is a confidence measure obtained from the relative size of the displacement component in the x direction compared to the one in the y direction,

$$c_1(x,y) = \frac{d_x(x,y)^2}{d_x(x,y)^2 + d_y(x,y)^2}. \tag{37}$$

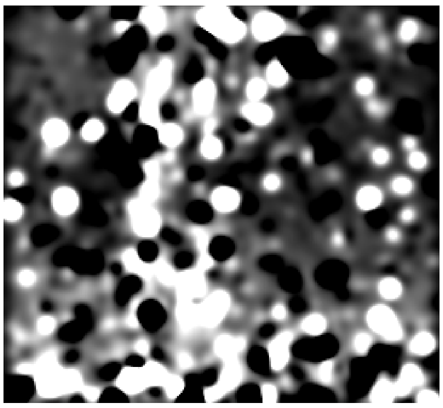

Figure 7: Local averages of the disparities shown in figure 6.

The second source for certainty is that we do not trust the outlier values at all, i.e.

$$c_2(x, y) = \begin{cases} 1, & d_{\min} \leq d_x(x, y) \leq d_{\max}, \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

The last certainty source is related to the computation of the polynomial expansion. The fast algorithm for this, mentioned at the end of section 4, ignores the signal/certainty principle and effectively assumes that the image values are zero outside the border. Thus the disparity estimates along the edges are not to be trusted. Since the Gaussian weights are limited to an $N \times N$ window, the values within half this distance from the edge may be corrupted,

$$c_3(x, y) = \begin{cases} 0, & (x, y) \text{ within } \frac{N-1}{2} \text{ pixels from the edge,} \\ 1, & \text{otherwise.} \end{cases} \tag{39}$$

The total certainty is computed as the product of these three,

$$c(x, y) = c_1(x, y)c_2(x, y)c_3(x, y). \tag{40}$$

The first certainty component c_1 and the total certainty c are shown in figure 8.

The signal used in the normalized averaging is of course the disparity estimates, i.e. the x component of the displacement computed in equation (36). The result can be seen in figure 9 and is acceptable. It is certainly not perfect, but adequate considering that the primary design goal of the algorithm was simplicity.

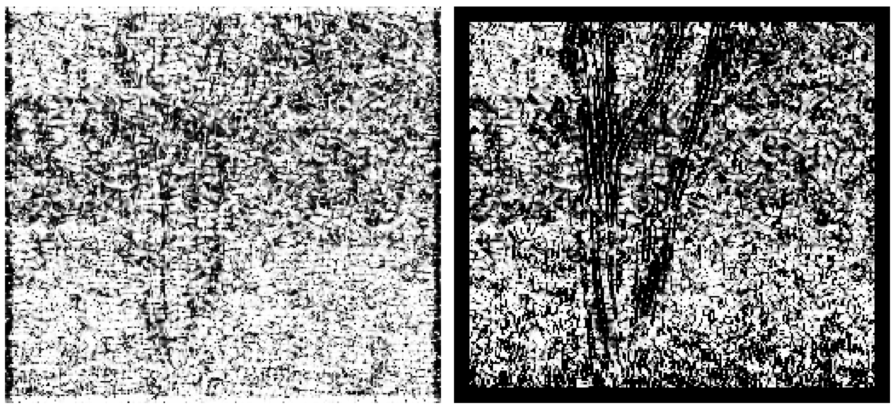

Figure 8: Certainty fields c_1 and c. Black corresponds to certainty 0 and white to certainty 1.

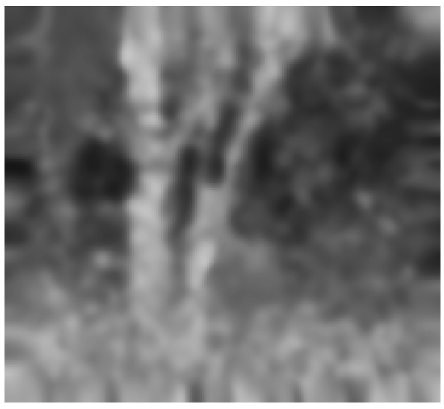

Figure 9: Disparity estimates after normalized averaging.

6.4 Algorithm Summary

The final algorithm is summarized below:

1. Compute polynomial expansions $\mathbf{A}_l$, $\mathbf{b}_l$, c_l and $\mathbf{A}_r$, $\mathbf{b}_r$, c_r for the left and right images respectively, using a Gaussian weight with standard deviation σ_1 and size $N_1 \times N_1$.
2. Compute $\mathbf{A}$ and $\mathbf{\Delta b}$ according to equations (33) and (34).
3. Compute displacement vectors $\mathbf{d}$ by solving 2×2 equation systems according to equation (36).
4. Set up a Gaussian applicability a with standard deviation σ_2 and size $N_2 \times N_2$.
5. Compute certainty values according to equations (37) – (40).
6. Apply normalized averaging to the x component of the displacement computed in step 3, using a as applicability and c as certainty. This gives the final disparity estimates.
7. Transform the disparities to depth values according to equation (8).

References

[1] Gunnar Farnebäck. *Polynomial Expansion for Orientation and Motion Estimation.* PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, 2002. Dissertation No 790, ISBN 91-7373-475-6.

[2] G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision.* Kluwer Academic Publishers, 1995. ISBN 0-7923-9530-1.