A Fuzzy Logic Algorithm for Dense Image Point Matching

Christian B. U. Perwass
Institut für Informatik
CAU Kiel,
24105 Kiel, Germany
christian@perwass.com

Gerald Sommer
Institut für Informatik
CAU Kiel,
24105 Kiel, Germany
gs@ks.informatik.uni-kiel.de

March 13, 2001

Abstract

We present a conceptually simple algorithm for dense image point matching between two colour images. The algorithm is based on the assumption that the topology only changes slightly between the two images. Following this assumption we use an iterative fuzzy inference process to find the likeliest image point matches. Advantages of this algorithm are that it is fundamentally parallel, it does not need any exact geometric information about the cameras and it can also give image point matches within homogeneous areas of the two images.

1 Introduction

Image point matching is important for many aspects of Computer Vision. Most notably for stereo vision and optical flow [1]. There are a number of strategies to reduce the complexity of the problem in the case of stereo vision. One is to simplify the task by extracting features in the two images, which are then matched [2, 3]. An obvious drawback of this approach is, that only certain areas of the images are matched. Of course, depending on the application, this may be sufficient. However, feature based 3D reconstruction from a stereo image pair only gives us a sparse model of the 3D world.

There are quite a number of algorithms which produce dense disparity maps, given the geometry of the camera system [4, 5, 6, 7]. Most of these algorithm make their decisions locally without taking into account global constraints on the image point matches. Marr and Poggio [8, 9] were one of the first to apply global constraints to dense disparity map algorithms. Their assumptions about stereo were uniqueness and continuity of a disparity map. We apply similar constraints directly to the images. An interesting paper in this context is [10], where a scale-based connectedness on images is used for image segmentation.

In [11] Barnard and Thompson present an algorithm which also assumes a continuous disparity map. They use this constraint to find the disparity of feature points in two images, exploiting the fact that nearby feature points should have similar disparity values. The probability of the correct disparity values is then estimated through an iterative relaxation labeling technique. The basic ideas in [11] are quite similar to our fundamental matching strategy. However, their algorithm only applies to feature points and the actual implementation of the constraints is quite different to ours.

Another approach to dense matching is to assume that there is an affine transformation between the two images [12]. Finding the appropriate affine transformation is then a least squares problem. A drawback of this approach is that the parameter search space might be quite big. Furthermore, the approximation of what is really a projective transformation by an affine transformation, may fail for close range objects.

In this paper we present an algorithm which attempts to match two colour images pixel by pixel. To achieve this we make some *qualitative* assumptions about the geometry of the camera system that took the images and the 3D objects we look at. We do *not* need to know the exact epipolar geometry of the camera setup, though.

2 Theory

The general task of matching two images taken from quite different view points of the same 3D object, is quite hard. Here we try to simplify this task by making certain assumptions about the camera setup and the 3D objects we look at.

1. The two cameras are separated more or less horizontally by a small distance.

2. The vertical axes of the two cameras are approximately aligned and the optical axes point to the same area in the 3D scene.

3. There are no depth discontinuities in the 3D scene.

First of all note that these assumptions are only of a *qualitative* nature. That is, we do not need to know the exact geometry of the camera setup. Assumption 3 is quite strict, especially since there are typically quite a number of depth discontinuities in 3D scenes. The effect of this assumption will be discussed later on.

Assumptions 1 and 2 basically model the geometry of the human visual system. They ensure that the changes in the two generated images are not too big, i.e. in the order of a few pixels. Taking the three assumptions together gives us yet another bit of information. Let us denote the two given images by A and B, respectively. A pixel at position (x, y) in image A is denoted by $A_{x,y}$ and similarly for image B by $B_{x,y}$. Let $(A_{i,j}, B_{r,s})$ be a pixel match between images A and B. Then it follows from the three assumptions that pixel $A_{i+1,j}$ should have a match *near* $B_{r+1,s}$ in image B, and similarly for the other pixels adjacent to $A_{i,j}$. That is, we basically assume that the topology of the two images is similar.

In order to use this information we have to be able to evaluate the similarity between two pixels. When using gray images this is simply the difference between the gray values of the pixels. However, gray images contain much less information than colour images, which is why we chose to use the latter. Evaluating the similarity of two colour pixels is not as straight forward as for gray pixels, though. The method we used is described later. For now let us assume that we have a function $\psi(A_{i,j}, B_{r,s})$ which gives the similarity between pixels $A_{i,j}$ and $B_{r,s}$. ψ returns a value in the range $[0, 1] \subset \mathcal{R}$, whereby 1 denotes equality and 0 inequality. Calculating ψ for each pixel in image A with all pixels in image B gives a set of similarity distributions. We define $\Psi^{i,j}_{r,s} \equiv \psi(A_{i,j}, B_{r,s})$, which can be regarded as a four dimensional, discrete fuzzy set. For each (i, j) the matrix $\Psi^{i,j}_{\bullet,\bullet}$ is thus a 2 dimensional, discrete fuzzy set. Looking at the similarity distributions in this way, we can start thinking about applying fuzzy logic operators to these sets.

Before we start describing the algorithm, we will give a very short introduction to fuzzy sets and fuzzy operators. Fuzzy sets can be used to describe inexact or "fuzzy" information. With standard sets we can either say that an object is an element of this set or not. That is, for any subset $A \subset X$ of some set X there exists a function $f_A : X \to \{0, 1\}$, which says for any element $x \in X$ whether $x \in A$ or not. If $f_A(x) = 1$ then $x \in A$ and if $f_A(x) = 0$ then $x \notin A$. The idea behind fuzzy sets is to define a *degree of membership* by a value between 0 and 1. In order to model a fuzzy set A on X, a membership function $\mu_A : X \to [0, 1]$ is introduced which returns the degree of membership of any $x \in X$ to A. In our algorithm we can interpret each $\Psi^{i,j}$ as a fuzzy set, where the degree of membership of a pixel $B_{r,s}$ to this set is given by $\Psi^{i,j}_{r,s}$, the discrete membership function.

Fuzzy sets A, B can also be combined $A \cup B$ and intersected $A \cap B$. This is done through their membership functions μ_A, μ_B, via two functions $s, t : [0, 1] \times [0, 1] \to [0, 1]$, the so called s-norm and t-norm. That is, $\mu_{A \cup B}(x) := s(\mu_A(x), \mu_B(x))$ and $\mu_{A \cap B} := t(\mu_A(x), \mu_B(x))$. The actual form of s and t depends on the application. The functions s, t may for example be defined as $\max, \min$, respectively. In this way, s represents the logical OR operator and t the logical AND operator. An appropriate choice of s and t is in fact quite important for our algorithm.

We will express the topological similarity between images A and B in the following way. As before, let $(A_{i,j}, B_{r,s})$ be an image point match. Then we say that pixel $A_{i+1,j}$ has probability ρ_0 to match with pixel $B_{r,s}$, probability ρ_1 to match with $B_{r+1,s}$, probability ρ_2 to match with $B_{r+1,s+1}$ and probability ρ_3 to match with $B_{r+1,s-1}$.

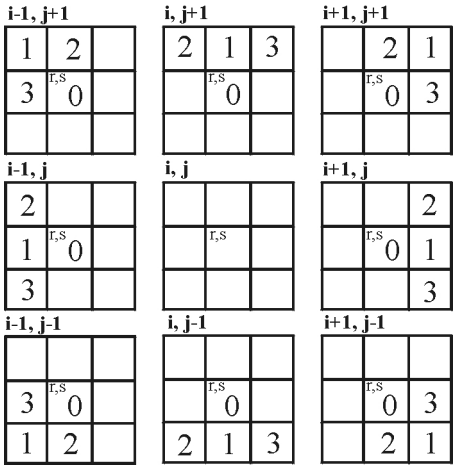

Figure 1: Schematic representation of similarity distribution patch $\Psi^{i,j}_{r,s}$ surrounded by neighbouring patches. Numbers $0, 1, 2, 3$ index pixels on image B that are candidates for matching corresponding pixel on image A.

Figure 1 shows the similarity distributions $\Psi^{i,j}, \Psi^{i+1,j}, \ldots$, for pixel $B_{r,s}$ and its adjacent pixels. The indices $0, 1, 2, 3$ indicate the pixels on image B that are candidates for matching the corresponding pixel on image A. For example, pixel $A_{i+1,j}$ has the similarity distribution $\Psi^{i+1,j}$ shown at the center right of figure 1. Assuming that the topologies of images A and B are similar, pixel $A_{i+1,j}$ could match only with the pixels indexed $0, 1, 2, 3$ of image B, as shown in the figure. The indices $0, 1, 2, 3$ also refer to the index of the probabilities $\{\rho_i\}$ for each match. Furthermore, we will write $\Psi^{i,j}_k$ to denote the pixel with index k in $\Psi^{i,j}$. The pixels indexed with 1 give the main topological direction. By this we mean that if images A and B were identical, the pixels along the main topological direction would give the correct match.

Now we can formulate the condition that a pixel match has to satisfy, if it is a correct match. Note that this is a fuzzy inference process, since we are working with fuzzy

sets. The AND ($\wedge$) and OR ($\vee$) operators we will use in the following therefore refer to fuzzy logic operators. In principle they may be implemented with any t-norm and s-norm, respectively.

We evaluate the confidence that a point pair $(A_{i,j}, B_{r,s})$ is indeed an image point match, using the following expression.

$$\Big[\bigwedge_{(u,v)} \Big(\bigvee_{k} (\rho_k \Psi_k^{i+u,j+v}) \Big) \Big] \ \wedge \ \Psi_{r,s}^{i,j}, \qquad (1)$$

where $k \in \{0,1,2,3\}$ and $(u,v) \in \mathcal{S}$, with

$$\mathcal{S} := \big\{ \ (1,0),(1,1),(0,1),(-1,1), \\ (-1,0),(-1,-1),(0,-1),(1,-1) \big\}.$$

The inner OR operation of equation (1) goes over all indexed pixels for the particular $\Psi^{i+u,j+v}$ similarity distribution. The outer AND operation counts over all similarity distributions surrounding the central distribution. The set $\mathcal{S}$ simply defines the appropriate offset vectors.

In words equation (1) says that the point pair $(A_{i,j}, B_{r,s})$ is an image point match if the pixels themselves are similar AND pixel $A_{i+1,j}$ is similar to $B_{r,s}$ OR $B_{r+1,s}$ OR $B_{r+1,s+1}$ OR $B_{r+1,s-1}$, AND pixel $A_{i+1,j+1}$ is similar to $B_{r,s}$ OR etc.

The confidence value returned by equation (1) lies in the range $[0,1]$. We could use this value directly or as the input to some decision function ϕ. To stay as general as possible, we will use the latter. Applying equation (1) for each pair of image points produces a new set of confidence values.

$$\Phi_{r,s}^{i,j} := \phi\left(\Big[\bigwedge_{(u,v)} \Big(\bigvee_{k} (\rho_k \Psi_k^{i+u,j+v}) \Big) \Big] \ \wedge \ \Psi_{r,s}^{i,j} \right). \qquad (2)$$

$\Phi_{r,s}^{i,j}$ gives a confidence measure that pixel $A_{i,j}$ corresponds to pixel $B_{r,s}$. Qualitatively, this is the same as $\Psi_{r,s}^{i,j}$, since higher similarity between pixel colours also means higher likelihood that the pixels correspond to each other. Hence, we could use equation (2) recursively by replacing Ψ with Φ.

Evaluating Φ enhances the confidence measure for those pixel matches that have a high confidence themselves and whose neighbouring pixel matches also have high confidences. The confidence measure of matches where this is not the case is reduced. The effect of iterating equation (2) once is that also the match confidence information from two pixels distance to every pixel is taken into account. For example, after the first evaluation of Φ, the confidence measure of $\Phi_{r,s}^{i+1,j}$ is based on $\Psi_{r,s}^{i+2,j}$ and other elements of Ψ. After iterating equation (2) once the confidence measure of $\Phi_{r,s}^{i,j}$ is based on $\Phi_{r,s}^{i+1,j}$, which in turn is based on $\Psi_{r,s}^{i+2,j}$. If we keep on iterating, larger and larger structures have an influence on the matching of every pixel. Therefore, it should also be possible to match over homogeneous areas in images, as long these areas are bounded by borders and we iterate long enough so that the whole structure is taken into account. Note that the transformation between two images is found here by allowing the topology to change slightly on a pixel scale.

3 Implementation

Although the algorithm presented in the last section is quite simple in principle, there are a number aspects we need to give some more thought to, in order to implement it. We need to give the exact form of the functions ψ and ϕ and choose appropriate t- and s-norms for the AND and OR operations in equation (2). Furthermore, Ψ and Φ can become quite large and evaluating them is computationally expensive. However, note that iterating equation (2) is emulating a feedback process. In principle this algorithm could be implemented as a large neural network which performs one iteration in a single step. Nevertheless, we implemented the algorithm on a standard computer, which imposed a number of constraints.

First of all we demanded in the last section that Ψ and Φ are evaluated for each pixel in image A over all pixel in image B. This is clearly not feasible for large images. In order to reduce the computational load we define a "source patch" in image A and a "target patch size" for image B. Then Φ and Ψ are evaluated for each pixel in the source patch of image A over a target patch in image B. However, now we have the problem of where to place the target patch in image B. This means we need to have an initial image point match $(A_{i,j}, B_{r,s})$. Then the target patch for pixel $A_{i,j}$ is centered about $B_{r,s}$. In fact, $(A_{i,j}, B_{r,s})$ does not need to be an exact match. We only have to demand that the correct match for $A_{i,j}$ lies within the target patch centered on $B_{r,s}$. Still, we might run into problems if the correct match for $A_{i,j}$ lies right at the border of the target patch. Note that by "correct match" we mean the best match available. Due to noise on the images and the transformations present, there will typically not be an *exact* match.

Finding the first approximate match is quite simple in the case of optical flow. We just choose the centers of images A and B, since it is the same camera that took the two images within a short time. We could also select a number of points from which the algorithm is started simultaneously. Of course, there may be situations when this strategy fails.

The situation is more difficult for a stereo camera pair. Here we would first need a method to adjust the cameras until their optical axes cross near a point in the 3D scene. Then we could again use the image centers of images A and B as the approximate first match. Note that the concept of attention will play an important role in such a method. If we had a sufficiently fast implementation of the algorithm

presented here, we could use it to adjust the geometry of the stereo camera system, until a proper match is found, using the image centers as an approximate initial match.

For now we will assume that an approximate initial image point match is given by $(A_{i,j}, B_{r,s})$, say. Then we can find $\Psi^{i,j}$ for the target patch centered on $B_{r,s}$. However, if we center the target patch for pixel $A_{i+1,j}$ about $B_{r,s}$, as well, its correct match may not lie within this target patch. Therefore, we center the target patch for some $A_{i+u,j+v}$ on $B_{r+u,j+v}$, i.e. on the pixel along the main topological direction. This means, of course, that the correct match for some $A_{i+u,j+v}$ may lie outside the corresponding target patch, if the topology has deviated sufficiently from the main topological direction. In effect, this constrains the size of the source patch.

In the following we will denote the n^{th} iteration of $\Phi^{i,j}_{r,s}$ by ${}^{n}\Phi^{i,j}_{r,s}$, whereby we define ${}^{0}\Phi^{i,j}_{r,s} \equiv \Psi^{i,j}_{r,s}$. We can therefore write equation (2) as

$$ {}^{(n+1)}\Phi^{i,j}_{r,s} = \phi\left(\left[\bigwedge_{(u,v)}\left(\bigvee_{k}(\rho_k\, {}^{n}\Phi^{i+u,j+v}_{k})\right)\right] \wedge {}^{n}\Phi^{i,j}_{r,s}\right), \tag{3} $$

with $n \geq 0$. At each iteration ${}^{n}\Phi^{i,j}_{r,s}$ is evaluated for each pixel in the source patch over all pixel in the corresponding target patch. Performing the inner OR operation is then simply a component wise OR operation on the matrices ${}^{n}\Phi^{i+u,j+v}_{k}$, with an appropriate offset. For example, the matrix ${}^{n}\Phi^{i+1,j}_{1}$ has no offset, since it was evaluated along the main topological direction and the target patches are centered on the pixel along this direction. The matrix ${}^{n}\Phi^{i+1,j}_{0}$ has to be offset by one pixel to the right before ORing it with ${}^{n}\Phi^{i+1,j}_{1}$ (see figure 1). In this way the inner OR operation of equation (3) is performed for all pixel in the target patch in one step. Because the target patches are of finite dimensions, offsetting a matrix introduces an additional row or column. We set this extra row or column to zero before performing the OR operation. This means that the pixels right at the border of a target patch will not take into account as much information as the inner pixels. Note that before ORing the matrices we also multiply them component wise with the probabilities $\{\rho_k\}$.

We found that the inner OR operation is represented well by the $\max$ function. For the AND operation over the elements of $\mathcal{S}$ we investigated a number of fuzzy operators. Two different operators produced good results: the λ-operator and the mean operation. The λ-operator is a mixture between the algebraic product and the algebraic sum. It is defined as

$$ f_\lambda(a,b) := \lambda\,[ab] + (1-\lambda)\,[a+b-ab]\,, \quad \lambda \in [0,1]. \tag{4} $$

If $\lambda = 0$, $f_\lambda(a,b)$ gives the algebraic sum of a and b, which compares to an OR operation. If $\lambda = 1$, $f_\lambda(a,b)$ gives the algebraic product of a and b, which corresponds to an AND operation. That is, the λ-operator can be something in between AND and OR. We found that for $\lambda = 1$ the iteration converged quickly to a good match, because spurious matches are eliminated quickly. However, sometimes correct matches are also eliminated prematurely, which makes the iteration unstable. This is because the algebraic product is a "strict" AND operator: it is enough if one of the elements is zero in order to make the whole result zero. Thus, if the confidence for a single pixel match goes to zero it will propagate through the whole image and make all confidences zero. This is clearly not what we intended. Smaller values for λ reduce this effect but also increase the number of iterations needed to obtain good matches.

A better way to express the AND operation is through the mean operation. That is, AND becomes an operation "between" OR and AND. Here we take the mean component-wise over all matrices. In this way, a single, or even a few, zero components do not make the result zero. Using the mean operation stabilizes the iteration process, although more iterations are needed to obtain a good match. However, this should also be somewhat desired, because we want larger structures to have an influence on the matching process. Nevertheless, for the outer AND operation in equation (3) we use the algebraic product. This gives a good compromise between speed and stability.

Another important choice is the form of the function ϕ. So far we have not found a completely satisfactory ϕ, although the following implementation works quite well. Let ${}^{n}\bar{\Phi}^{i,j}_{r,s}$ be defined by ${}^{n}\Phi^{i,j}_{r,s} = \phi({}^{n}\bar{\Phi}^{i,j}_{r,s})$. The ϕ we use scales each matrix ${}^{n}\bar{\Phi}^{i,j}_{\bullet,\bullet}$ separately, such that the largest component of the matrix becomes 1. This scaling is necessary because we are using a single byte for each confidence value due to memory constraints. Furthermore, the confidence values tend to become smaller and smaller with each iteration. With this renormalization applied through ϕ, we make sure that each matrix ${}^{n}\bar{\Phi}^{i,j}_{\bullet,\bullet}$ has at least one confident match. This is also the disadvantage of using this ϕ, because in real 3D scenes we do have depth discontinuities and thus occlusion, which in turn means that not every pixel in image A needs to have a match in image B. If occlusion occurs, the algorithm will give the next best match for the pixel. Future work will investigate how to incorporate occlusion into this algorithm in a better way.

There is one part of the algorithm left which we have not discussed, yet. This is the form of the function ψ. Recall that ψ should give the similarity of two pixel. This may be based on any kind of information, not just their colours. We restricted ψ to give the similarity of two pixels only based on their colours, though. Of course, this poses the question which colour space to use and how to obtain a similarity measure [13]. Since this is not essential to our algorithm, we will use the simplest way to represent the colour of a pixel, namely as a 3D-vector in an RGB-colour space. The difference between two colours represented by colour vec-

tors $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ may then be calculated as $||\boldsymbol{c}_1 - \boldsymbol{c}_2||$. However, this means that black and white are more different from each other than are black and red, say. Phenomenologically, white should be regarded as as different from black, as red is from black, though. For our purposes it would be desirable that all pairs of colours black, white, red, green and blue, are regarded as being equally dissimilar. This can be achieved in the following way.

Let $\boldsymbol{\Delta} = \boldsymbol{c}_1 - \boldsymbol{c}_2$, and denote the components of $\boldsymbol{\Delta}$ by $(\delta_r, \delta_g, \delta_b)$ with $\delta_r, \delta_g, \delta_b \in [0, 1]$. Then we find the similarity of colours $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ as follows.

$$\psi(\boldsymbol{c}_1, \boldsymbol{c}_2) = 1 - \frac{||\boldsymbol{\Delta}||}{||\boldsymbol{\Delta}/\max(\delta_r, \delta_g, \delta_b)||}, \tag{5}$$

if $\boldsymbol{c}_1 \neq \boldsymbol{c}_2$. For $\boldsymbol{c}_1 = \boldsymbol{c}_2$ we define $\psi(\boldsymbol{c}_1, \boldsymbol{c}_2) = 1$.

To summarize, equation (3) takes the following form in our implementation.

$${}^{(n+1)}\bar{\Phi}^{i,j}_{r,s} = \left[\tfrac{1}{8} \sum_{(u,v)} \sup_k \left(\rho_k {}^n\Phi^{i+u,j+v}_k\right)\right] * {}^n\Phi^{i,j}_{r,s}, \tag{6}$$

and

$${}^n\Phi^{i,j}_{r,s} = \frac{{}^n\bar{\Phi}^{i,j}_{r,s}}{\sup_{r,s}\left({}^n\bar{\Phi}^{i,j}_{r,s}\right)} \tag{7}$$

4 Experiments

Figure 2: Initial image of mug.

We present here three experiments with real images to show some aspects of the algorithm. Figures 2 and 3 show two images of a coffee mug taken from slightly different positions. These images of the mug were taken with a digital camera with a resolution of 1024×768 pixels. This resolution was reduced to 200×150 pixels in a post-processing step. For the following test we used the center of the flower at the top left as the starting point for the algorithm. The source patch was 41×41 pixels and the target patch 7×7 pixels. Note that the source and target patches do not need to be square. However, the target patch should have odd dimensions.

Figure 3: Image of mug from a slightly different perspective.

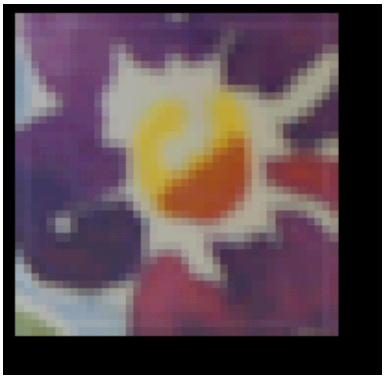

Figure 4: Reconstructed image of translation test.

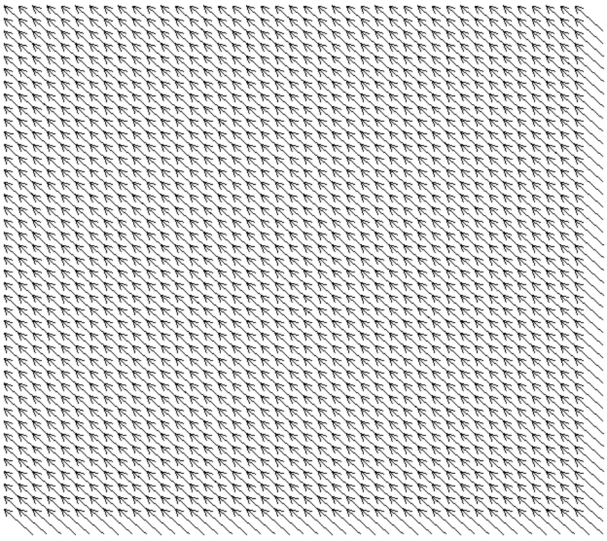

Figure 5: Flow field of translation test.

The $\{\rho_k\}$ from equation (3) were set to the following values: $\rho_1 = 1$ and $\rho_0 = \rho_2 = \rho_3 = 0.9$. That is, the main topological direction was slightly preferred. The result images are shown after 20 iterations which took about 3

minutes and 20 seconds. The computer used was a Pentium II with 233 MHz running Windows Me.

Figure 6: Image of mug rotated clockwise by 10 degrees.

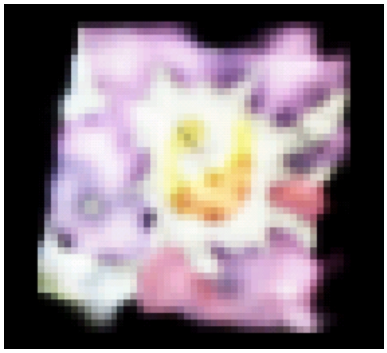

Figure 7: Reconstructed image of rotation test after 7 iterations.

The implementation of the algorithm was not optimized for speed but for adaptability. In any event, real time capability is most likely to be achieved by a hardware implementation of the network structure that lies at the root of the algorithm.

First we present a simple test which shows the behaviour of the algorithm with ideal data. In this experiment we used the image in figure 2 as both, source and target image. However, the initial image point match used was off by two pixels to the right and two pixels down. The reconstructed image after 20 iterations is presented in figure 4 and the corresponding flow field in figure 5. Since the reconstruction and flow field are drawn relative to the target pixel of the initial image point match, we should expect the reconstruction to be translated two pixels to the left and two pixels up, with a corresponding flow field. This is exactly what we find. Note that the correct flow is also found for homogeneous areas in the image.

Figure 8: Reconstructed image of rotation test after 20 iterations.

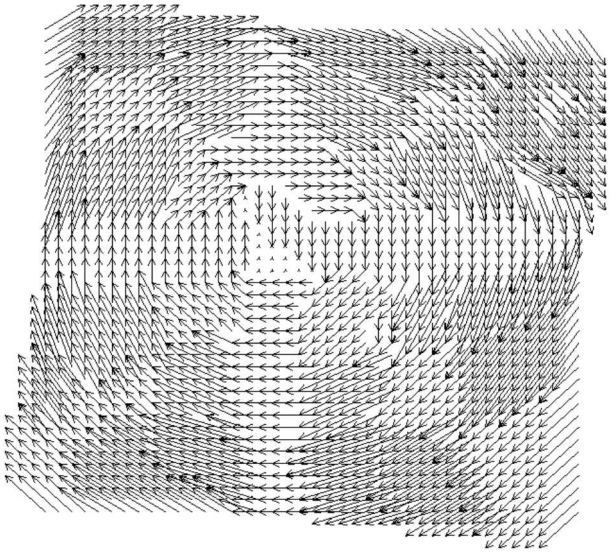

Figure 9: Flow field of rotation test.

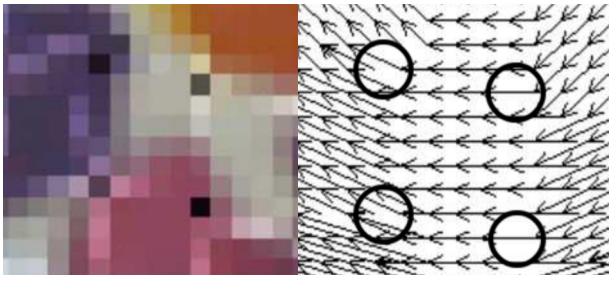

Figure 10: Comparison of reconstructed image with corresponding flow field of rotation test.

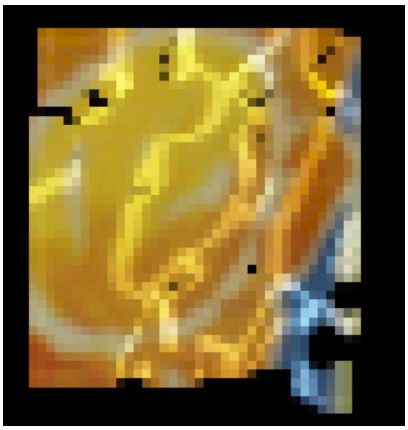

Figure 11: Reconstructed image of third test.

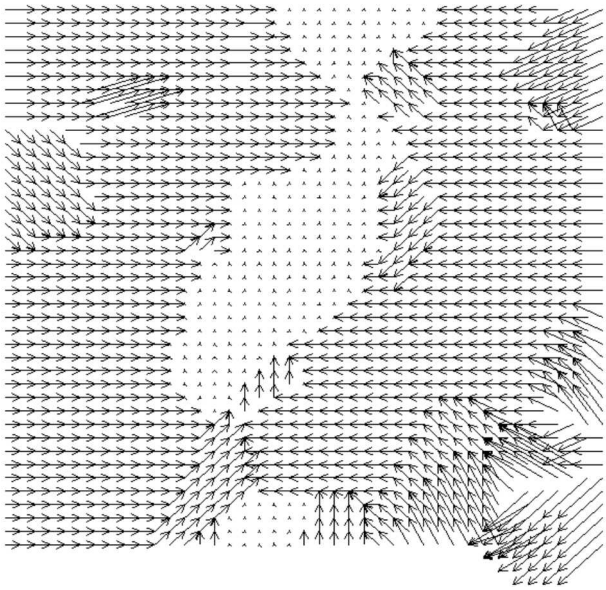

Figure 12: Flow field of third test.

The reconstruction is done in the following way. Each target patch is multiplied with the colour of its corresponding source pixel. All target patches are then added component-wise with the correct offsets using the algebraic sum (see equation (4)). Therefore, if a number of different target patches predict that their corresponding source pixel lies at the same target pixel, the colours of the different source pixels will be added with the algebraic sum. The effect of this is that such a pixel will appear more white in the reconstruction. If you have a good reproduction of this paper and you take a close look at the pixels one pixel in from the border of the image in figure 4, you will see that these pixels are somewhat lighter. This is a border effect of the algorithm. The pixels right at the border of the source patch only obtain matching information from the inside of the source patch. This makes their localization in the direction towards the inside of the source patch somewhat more uncertain. Note that the arrows in the flow field point to the pixel with the highest confidence in the target patch.

The next test of the algorithm is a bit harder. We take the image from figure 2 as the source image, and a version of it rotated clockwise by 10 degrees, as the target image. This is shown in figure 6.

This test is harder because the rotation of the initial image will have created pixels whose colour is an interpolation between adjacent pixels. That is, pixels in the source image will not necessarily have an exact colour match in the second image. The image reconstructions after 7 and 20 iterations are shown in figures 7 and 8, respectively. The corresponding flow field after 20 iterations is shown in figure 9.

The results of this test show that the algorithm had some more problems in finding the correct matches, especially towards the border of the source patch. Nevertheless, most pixels are still matches consistently, even in areas of nearly homogeneous colour. Figure 7 shows very nicely how the algorithm works. The confidence distributions in the target patches converge more quickly to a single pixel match at colour edges, whereas within homogeneous areas they are more spread out. However, after some more iterations, the border information ”propagates” into these homogeneous areas and produced sharp pixel matches.

Note that we do not restrain the algorithm to give unique pixel matches. The algorithm is free to match two or more pixels from the source image to the same pixel in the target image. It may also match a source image pixel to many pixels in the target image. Although this freedom is initially desired, it can create problems, especially in the presence of occlusion.

The black spots in the reconstructed image show pixels which are not matched with any source pixel. Comparing the areas where this happens with the corresponding areas in the flow field shows that these black spots can appear where the change of flow closer to the center of the image happens faster than further on the outside. Figure 10 shows a comparison of black spots in the reconstructed image with the corresponding holes in the flow field. This effect is basically due to the discrete nature of the algorithm.

The last test of the algorithm we present here uses figure 2 as source image and figure 3 as target image. This time we have two images taken at different times. This means that there will be different noise on the images. Furthermore, due to the changed perspective, the colours will appear slightly different. For this last test we will use the sun at the top right of the mug. The starting match was inside the sun. If you look closely, you will see that figure 3 was taken from a position slightly left of the position figure 2 was taken from. This means that the sun in figure 3 is slightly compressed compared to figure 2. The reconstructed image after 20 iter-

ations is shown in figure 11 and the corresponding flow field in figure 12.

The bright lines in the reconstructed image point to compression, since more than one pixel is reconstructed at the same target image position. This can also be seen in the flow field. Recall that the flow field is drawn relative to the target pixel in the initial image point match. Therefore, in the center of the flow field there is no change of pixel positions. However, the flow to the right points left and vice versa, which shows that there is a compression of the sun along the horizontal axis. This is exactly what we should expect. Still, the reconstruction is not perfect. Spurious matches occur especially towards the borders.

5 Conclusions

In this paper we have presented a conceptually simple algorithm for dense image point matching between two colour images. The algorithm relies on the fact that the topology of the two images has only changed slightly. It then employs some fuzzy combinatorics to find the likeliest image point matches. So far, a disadvantage of the algorithm is that it cannot handle occlusion particularly well. Furthermore, spurious matches cannot always be easily distinguished from proper matches, because the algorithm always returns the likeliest match, independent of an absolute confidence value. These are problems that will be addressed in future research.

A big advantage of this algorithm is that it is fundamentally a parallel algorithm. In fact, our implementation had to emulate this parallelism. The structure of the algorithm also readily lends itself to a hierarchical computation scheme, in which the matching proceeds from low to high resolution versions of the images.

The initial similarity measure between pixels does not need to be constrained purely to colour information. Any kind of information reflecting a property of a pixel can be used. This may, for example, be the gradient at a pixel, or even structural information from the structure tensor.

Another positive feature of the algorithm is that it gives image point matches in homogeneous areas, away from colour edges. This works because a certain topology is assumed, which can only be changed slightly on a pixel basis. Borders in an image therefore act like ”anchors” that constrain the change of topology between the two images. In effect, the algorithm matches borders and interpolates between them. However, this is done *implicitly*, following a simple iterative procedure. The advantage here is, that not too many heuristical elements enter the algorithm, like deciding what should count as an edge and what should not.

Although there are still a number of problems that need to be addressed, we believe that our algorithm shows some promise to become a good estimator for stereo matching and optical flow.

References

[1] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, “Performance of optical flow techniques,” *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77, 1994.

[2] H. H. Baker and T. O. Binford, “Depth from edge and intensity based stereo,” in *Proc. Seventh Int. Joint Conf. Artificial Intelligence*, pp. 631–636, 1981.

[3] S. A. Lloyd, E. R. Haddow, and J. F. Boyce, “A parallel binocular stereo algortihm utilizing dynamic programming and relaxation labelling,” *Computer Vision, Graphics, and Image Processing*, vol. 39, pp. 202–225, 1987.

[4] R. Szeliski and P. Golland, “Stereo matching with transparency and matting,” *Int. Journal of Computer Vision*, vol. 32, no. 1, pp. 45–61, 1999.

[5] M. Okutomi and T. Kanade, “A multiple-baseline stereo,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 4, pp. 353–363, 1993.

[6] C. L. Zitnick and T. Kanade, “A cooperative algorithm for stereo matching and occlusion detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 7, pp. 675–684, 2000.

[7] G. le Besnerais and H. Oriot, “Disparity estimation for high resolution stereoscopic reconstruction using the gnc approach,” in *Proc. IEEE Int. Conf. on Image Processing*, vol. 2, pp. 594–597, 1998.

[8] D. Marr and T. Poggio, “Cooperative computation of stereo disparity,” *Science*, vol. 194, pp. 209–236, 1976.

[9] D. Marr and T. Poggio, “A computational theory of human stereo vision,” *Proc. Royal Soc. London B*, vol. 204, pp. 301–328, 1979.

[10] P. K. Saha, J. K. Udupa, and D. Odhner, “Scale-based fuzzy connected image segmentation: Theory, algorithms and validation,” *Computer Vision and Image Understanding*, vol. 77, pp. 145–174, 2000.

[11] S. T. Barnard and W. B. Thompson, “Disparity analysis of images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 4, pp. 333–340, 1980.

[12] G. Gimel’farb and J. Zhong, “Matching multiple views by the least square correlation,” *To be publ. in LNCS series*, 2000. Presented at the 10th International Workshop ”Theoretical Foundations of Computer Vision”, Schloss Dagstuhl, Germany.

[13] S. J. Sangwine and R. E. N. Horne, eds., *The Colour Image Processing Handbook*. Chapman & Hall, 1998.