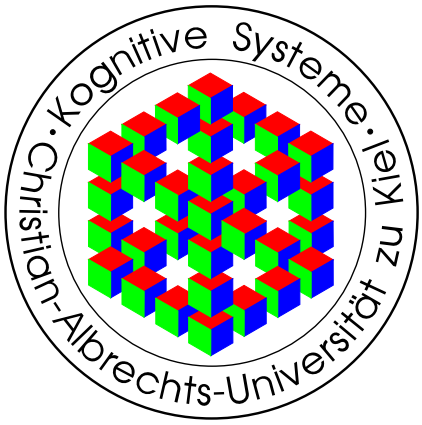
Dense Image Point Matching with Explicit Occlusion Detection for Stereo Disparity and Optic Flow



Diploma Thesis

Cognitive Systems Group
Institute of Computer Science and Applied Mathematics
Christian–Albrechts–University of Kiel

Proposed by
Kevin Köser
kk@ks.informatik.uni-kiel.de

April 2003

Abstract

This diploma thesis is an extension of the matching model proposed in *Dense Image Point Matching through Propagation of Local Constraints* (see [1]), where C. Perwass and G. Sommer develop an iterative algorithm, which computes the most probable correspondence for each pixel between two similar images. Initialized with pixel (e.g. color) similarity, the algorithm accumulates global information by iterated application of a local constraint, that demands similar displacements of neighboring pixels.

The probability model is now refined further, so that it takes occlusions into account and integrates their influence on match probability into the assumed distributions. Additionally, the explicit detection of half-occluded pixels evolves from this model by exploiting bidirectional correspondence information. Since half-occluded pixels usually occur at depth discontinuities, which are often smoothed by matching algorithms, the new derivation uses occlusion probability to sharpen flow fields at depth discontinuities.

The general model is then adapted to two main applications for matching, which are computation of stereo disparity and optic flow, where advantage is taken of the expected structure of the image data. For these scenarios, an algorithm is developed, which does no longer depend on accurate a priori information about where correspondences should be searched and how large the search area for a particular image pair should be.

Contents

Chapter 1

Introduction

1.1 Matching in Computer Vision

By taking advantage of the two slightly different perspectives of his eyes or by moving (even with only one eye open), a human is able to obtain a three-dimensional impression of the environment, to recognize particular structures as belonging together or to track objects moving through space. The three-dimensional perception and the estimation of depth becomes possible, because the appearance of objects close to the spectator is very sensitive to movement or small perspective differences (as of the eyes), while size and position of objects far away look nearly unchanged.

Realizing that, in particular computer vision research fields one tries to obtain spatial information by taking advantage of the displacement of certain objects regarding some corresponding images. However, in order to compute or measure these changes between similar images of the same scene, corresponding points have to be searched in the images, which are 2d-projections of the same entity in 3d-space. Classifying points of different images as referring to each other in this way is called the *Correspondence Problem*. Most matching algorithms try to find these correspondences by using the raw or somehow preprocessed image signal, but usually without deep semantic knowledge about the scene. Having gathered information about many correspondences in the images, high level applications can obtain spatial data about the scene and make their respective conclusions, e.g. about positions or movements of objects or cameras. This information can then be used for navigation, grasping and tracking purposes or for other problems depending on visually supervised interaction with the environment.

Matching algorithms can roughly be divided into two groups regarding their strategy: There are feature matching methods on the one hand (as

e.g. [7] or [8]), which search correspondences only for some characteristic points, e.g. at contour lines or corners. The properties of these characteristic points are assumed to be invariant in some way and therefore these points (as the projections of so called "features") are expected to be matchable very robustly. Hence, these algorithms usually need a preprocessing step, which extracts these features in both images. For areas between these characteristic points, correspondence information is either not used or interpolated.

On the other hand there are dense pixel match algorithms which try to get the most probable correspondence for every pixel in the image, e.g. by making use of statistical correlations (like in [13]) or evaluation of analytic models of the image data (as in [10]), even for regions where there is only sparse structural information. Using the statistical correlation approach, authors usually assume some defined region around the pixel to be matched (a so called window) to be similar in the other image. That is, the window is compared to all windows in the target region using correlation functions from signal theory. Then the window with the highest similarity is assumed to be the correct match. The main problem here is to find an appropriate window size, where the compared area is large enough to be significant but small enough to be invariant. Usually additional assumptions (e.g. on the scale of the image) are imposed to constrain the matching even further. Some authors propose that the correspondence problem can be compared in that case to some minimization processes in nature, e.g. to the problem of finding the minimum energy for some crystal structure or molecule (see [25]).

A slightly different approach is chosen in bayesian information diffusion algorithms (as in [20]), where the global solution is not searched explicitly as a whole. Instead, one expects to approximate a good solution by propagating information in parallel from single pixels to neighborhoods and finally to the whole image. Here, a large group of authors has been influenced by the work of Marr and Poggio (see [11]), who have related results from neurophysiology of the human visual system to the nature of the stereo correspondence problem in computer vision. This has resulted in an iterative and highly parallel algorithm for stereo matching, based only on some local constraints, which propagate through interaction between neighboring pixels at iteration steps. These constraints are *uniqueness* and *continuity*. The first states - roughly speaking - that each pixel is a projection of some physical object and thus can only have one displacement into the other image, while the matching pixel must be assigned an inverse displacement, since it is a projection of the same physical object. The second assumption is based on the observation that surfaces of objects usually change only smoothly and thus demands a continuous displacement field almost everywhere, that is, neighboring pixels are expected to have similar displacements.

Regardless of the strategy a matching algorithm uses, there are some main problems in matching which will briefly be explained now.

- *Invariance*: The crucial question in matching is: Which properties are invariant under which circumstances? An objective general criterion is hard to find and seems to be application specific. Shape changes with perspective, surfaces look different from another angle, color and intensity change with illumination, similar problems occur for most properties or features.

- *Noise and Errors*: Due to environmental influence in the scene, camera inaccuracy and other measurement errors, the image data can be disturbed. There may be no exact color equality for corresponding points and correspondences may lie at subpixel positions. Algorithms should be robust enough to handle this noise on the images in order to prevent ambiguous or wrong matching results.

- *Aperture Problem*: Many matching algorithms use local operators or local invariance assumptions to match small areas or pixels between the images. Due to the algorithms' "local view", different match candidates may be equally likely. As an example, some image regions may locally have an intrinsically one-dimensional structure (or no at all), as for instance an edge with no corner nearby. Local knowledge may be insufficient to match these areas, because there is no information in the direction of the edge. Thus many matches can become equally likely. The true one can only be found by enlarging the area under inspection and therefore using more and more global knowledge. Apart from often dramatically increasing computational complexity for large region matching, most models assume only small structures to be invariant (local invariance), which conflicts with a matching of larger regions. Additionally, large regions are more likely to contain depth discontinuities, where similarity between the images cannot be assumed.

- *Depth Discontinuities*: Most regions in natural images are continuous. That means, depth changes only smoothly in those regions. Therefore it may be a good assumption that neighboring pixels should have similar displacements, which is quite essential for most matching algorithms. Only at the region borders (e.g. boundaries of objects) there are cuts in the depth maps. Since there is no semantic understanding of the image, the positions of object boundaries are not known and thus the depth discontinuities violate the smoothness assumption locally. Additionally, at depth discontinuities objects in the foreground often occlude objects in the background, which can disturb similarity even more.

Figure 1.1: Problems in Matching: Occlusion and Depth Discontinuities (Rear Wheel), Noise and Invariance (Door Handle), Aperture Problem (Lower Window Edge)

- *Occlusion and Borders*: In presence of depth discontinuities there are often pixels in one image which do not have a correspondence in the other image. Usually these pixels belong to an object in the background, which is (partly) hidden by an object in the foreground. Such pixels are called *half-occluded* or simply *occluded*, since pixels that cannot be seen in any image are disregarded anyway. A problem very similar to occlusion occurs at the image borders, where correspondences sometimes lie outside the image and therefore pixels cannot be matched. There is simply no information available beyond the borders.

Some of the problems can be seen in figure (1.1), which shows a small section of a slightly altered artificial scene from the street sequence[1]. The left image shows the scene, the middle image is an enlarged part of it with depth discontinuities and the right image shows the same section (referring to pixel coordinates) of a subsequent image. Some noise has been added, which is usually unavoidable in natural images. Occluded areas are the rear wheel of the car and parts of the background. Due to the noise, RGB colors are not absolutely invariant over the images and due to the low resolution the door handle appears quite noticably different. The aperture problem can be seen at the lower part of the car door, where local information is very sparse due to low contrast, making several matches equally likely. At the lower edge of the window, there is certainty about the vertical position of possible matches, but it is nothing known about the correct horizontal position, unless one takes into account larger neighborhoods, which include horizontal structure.

[1]The street sequence is an artificial image sequence with ground truth from the University of Otago, New Zealand. For more information, please refer to [6].

1.2 Optic Flow and Stereo Disparity

Using knowledge about how the images have been created can reduce complexity and make the matching process more robust and accurate. Most algorithms are designed for special geometries, where two main groups can be distinguished, though there are some other applications and some general algorithms that try not to use additional constraints about image creation. Algorithms taking advantage of different perspectives of cameras, which simultaneously take images of a scene, are called stereo algorithms, while those using a time series of images of the same camera are often called optic flow algorithms.

The expression optic flow is an idea borrowed from hydrodynamics, where flow models based on the movement of very small fluid volumes have been applied succesfully for a long time in current analyzation. Analogously, in image processing pixels can be treated as smallest picture elements which flow (or move) across the image plane, as is pointed out in the well-known paper of Horn and Schunck [10]. The displacement field of corresponding pixels between images of a time series is then called the *optic flow*. Taking advantage of the knowledge that the images to be matched are subsequent frames of such a series, additional constraints on the data can be imposed, simplifying and stabilizing the general matching algorithm. A common assumption is that the camera grabs images very frequently, while changes in the scene take place only slowly. Consequently, movements (and optic flow) between two subsequent images are minor and velocity can be assumed to be constant over some images.

In stereo matching one tries to relate pixels from an image of camera A with pixels of an image of some other camera B to get three-dimensional information of the scene, which both cameras show. A common geometry is that the optical axes of the cameras used are (nearly) parallel, the image planes lie in the same plane in 3d-space, and the image lines are parallel to the baseline of both cameras, such that the displacements of pixels between left and right image are only horizontal. In stereo matching applications this displacement is called *disparity*. If there are objects in the scene quite near to the cameras (compared to the distance of the cameras), large disparity variations between these objects and the background or other objects may occur. A good stereo algorithm has to cope with these variations, which may be easier to handle, if one matches over a scale pyramid, starting with images of a heavily reduced size, e.g. as Barnard shows in [25].

Many approaches to the stereo matching problem simplify the environment and use fixed reference points, which can then be found easily in all images, remove the background, use objects with different colors and so on.

The exact knowledge of camera geometry reduces the search space to one dimension, since a point in image A has to lie on a line in image B and the only remaining degree of freedom is the depth of the point in space. Using more information about the scene under inspection can lead to more stable and exact results. However, integrating additional knowledge deeply into an algorithm means that it can only be applied succesfully to such scenes, which means a loss of generality. Hence, the model of this thesis is always based on unknown camera parameters, though an adaptation for rectified stereo images is quite simple, as has already been shown in [1].

1.3 Previous Matching Model

The model derived by Perwass et al. (see [1]) uses a statistical approach to match every pixel of image A to a pixel (or subpixel position) of image B, it is therefore a dense point matching strategy. The correspondences are not calculated explicitly but by application of an iterative algorithm to some region of candidate pixels, in which each pixel is assigned a match probability. Using multiple iteration steps, the algorithm finally converges to the best matching position. The images to be matched are assumed to fulfill certain properties, which are stated informally here and will be discussed more in detail in section (2.1):

1. A and B have to be similar, i.e. only small movements in the scene and small perspective differences are allowed.

2. There must be a (continuous) measure of similarity between pixels, and corresponding pixels should be somewhat invariant regarding that measure.

3. Pixels of A and B have to implement an ordering constraint, that is, the left neighbor of a pixel x_A in A should have a correspondence which is also located left of x_A's correspondence in B. In other words, neighboring pixels should have similar displacements.

Equations and transformation rules used for matching are based on probability distributions, which reflect these assumptions: For each pixel there is a test patch in the other image, which contains all match candidates. To compute the best candidates (the most probable matches), probability distributions are calculated across these test patches. They are initialized with the pixel similarity of the candidate under inspection and updated with match probability information from neighboring pixels at each iteration.

Property one states that the aim of the algorithm is to calculate a correspondence for each pixel without semantic information about the image. It is not intended to reconstruct a scene with information from a completely different point of view. Property two is necessary, since similarities are used as correspondence probabilities which are weighted and averaged. These values propagate throughout the image in this way, which is not possible with pure binary data. Note that property three is similar to the idea leading to the smoothness constraint of the Horn and Schunck (see [10]) theory and the continuity constraint of Marr and Poggio (see [11]). Since it is responsible for the diffusion of information through the image, it is very substantial for the algorithm. It has to be stated that for regions belonging to the same object it is quite a good assumption and that it is inevitable for the convergence to a single match position. However, since it punishes large variations of neighboring displacements, it is quite counter-productive at object boundaries, where depth discontinuities - and thus discontinuities of the true displacement field - are very likely. Unfortunately, there is no information, if there are such boundaries or where they are, and so the algorithm tends to smooth depth discontinuities under some conditions.

Furthermore, the model expects every pixel to have a match and calculates the most probable correspondence for each pixel relying on this assumption. But the presence of occluded pixels might influence basic probability distributions used in the model. Additionally, during information diffusion a pixel depends on the patch distributions of neighboring pixels, regardless of the confidence of their match values, introducing errors at uncertain or occluded pixels.

1.4 Thesis Contents and Structure

The main goal of this thesis is to extend the model with respect to occlusion, i.e. consequences on the probability distributions used are inspected, if the assumption that every pixel has a match is dropped. Additionally, a way to explicitly find these occluded pixels is developed and using this information the handling of depth discontinuities should be improved. Finally, two main applications for matching, optic flow and stereo disparity computation are integrated into the model and the new algorithms are tested on artificial and natural images.

For some test images Perwass et al. showed in [1] that the algorithm can handle occluded pixels and that the introduced errors are locally constrained. However, the model implicitly forbids occluded pixels by assuming some particular probability distributions, as will be shown in chapter two.

Furthermore, 'faulty' match information of occluded pixels (having no correspondence in the other image) is treated the same way as the true information of pixels having a match. It is therefore desirable to detect occluded pixels explicitly, to reduce their influence on 'good' pixels and to integrate the results into the probability model. This should reduce the introduced errors further, leading to a better matching result and providing additional occlusion information, which is gained by exploiting bidirectional match probabilities. Chapter two shows the derivation of the new model in detail after a brief repetition of the base model as it is proposed in [1].

The third chapter explains some application specific refinements and how they can be integrated into the algorithm. The first application is stereo matching, where varying displacements between the images are expected. To handle these and to catch all occluded pixels and depth discontinuities, Gauss pyramids are constructed for each image, i.e. the image width and height are repeatedly reduced by a factor of two. The matching process is then started by executing the general algorithm at the smallest pyramid layer, where the images are expected to look more similar and therefore to be more easily matchable. Occlusion and match information is then used to initialize the next greater layer, until the actual images are matched at the original size. For optic flow, objects are assumed to have a constant velocity for a short time period and thus pixels are expected to have a constant velocity concerning some images. This constraint is integrated into the similarity function, where not only the pixels of the current two frames are compared, but also their extrapolated positions in older and more recent images.

Chapter four proposes a hardware implementation as a recurrent neural network, since the algorithm is fundamentally parallel. However, for qualitative analysis it is implemented in software first and implementation details, parameters and error measures are shown in this chapter, too, as well as a complexity analysis of the algorithm. Chapter five is dedicated to evaluating the matching quality and occlusion detection of the general algorithm with respect to artificial and natural images. The developed integrations for optic flow and stereo geometries regarding the patch positioning and improvement of the matching results are also tested here using some image sequences and stereo scenes. Finally, the last chapter presents a conclusion of this work and shows some future extension possibilities of the model.

Chapter 2

Matching Two Images

2.1 The Previous Model

2.1.1 An Overview

How does the original algorithm from [1] work in principle? At first, match probabilities from A to B are computed by comparing pixel values for a pixel x_A in image A and each pixel of a so called test patch $\mathcal{T}$ (see figure (2.1)) in image B. In this example, the RGB values of a pixel and its true correspondence in the other image are assumed to be similar.

The above mentioned test patch is the target area, where the match of the relevant pixel in A is expected to be located. Every pixel x_A has its own test patch, which is centered on position $x_A + d$ in B, where d is the average expected displacement for all pixels. The similarities are then normalized to sum to unity and regarded as correspondence probabilities for pixel x_A.

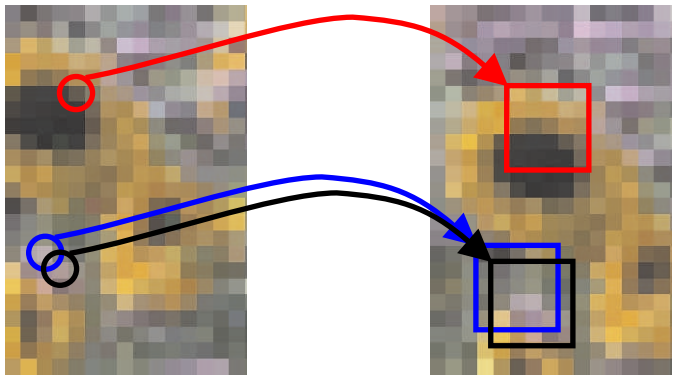


Figure 2.1: Matching from image A (left) to image B (right): Test patches

Each pixel x_B in that test patch now has a probability of being the correct match for x_A. This can be seen in figure (2.2), where white means high and black low probability. Now this is done also for all the other pixels of A and

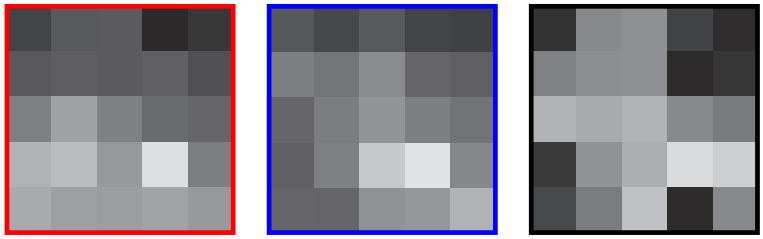


Figure 2.2: Probability distributions in patches

their test patches. Finally, there is a probability distribution (the values of the test patch) for each pixel in image A. Computing the expectation value across the patch would yield the most probable match position by now, but the information used will generally not be sufficient yet, because there may be several pixels having the same or similar colors.

A neighborhood constraint (also called ordering constraint) states that neighboring matches have similar displacements. Hence, for neighboring pixels patch probabilities are compared and those having no appropriate support in the neighboring patches are reduced, while those displacement candidates, which are also likely for the neighbors, are raised regarding their probability. This is applied to all eight neighbors of a pixel simultaneously, which keeps the displacement map smooth.

For instance, the origins of the blue and the black patch of figure (2.2) are (diagonal) neighbors and the test patches have the same relative displacement. Thus the relevant match is expected to be at nearly the same position in the patches (the true match is in both patches one pixel down and one right from the center). The neighborhood constraint will now reduce all probable (bright) pixels in the black patch, if there is a low probability (indicated by a darker square) at and around the same position in the blue patch and vice versa. Again, this is applied to all pixels in A, respective to their test patches. The probability of the test patches now contains color information of the pixel and structural information of the direct neighbors of the pixel.

The same procedure is then done from image B to image A and the resulting direction dependent probabilities are averaged with those of the first direction. The neighborhood constraint and the bidirectional averaging are now applied repeatedly, collecting more and more global information, which is called the diffusion of local constraints. At each iteration the neighborhood

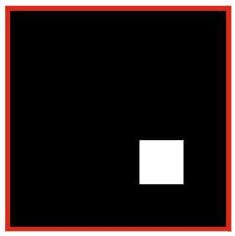


Figure 2.3: Ideal patch distribution after multiple iterations

and the structural information that influences the patch under inspection is expanded, because the neighborhood constraint always uses the updated data of the neighbors, which has been influenced by the neighbors' neighbors at the previous iteration and so on. In this way local information is distributed over the whole image, if enough iterations are executed.

The subsequent bidirectional merging stabilizes the matching, since both directions should yield similar probabilities, if the assumptions hold. The probability distribution in the test patches sharpens, since more and more match candidates are suppressed by more or less neighboring pixels. Finally, the expectation value (within the test patch) yields the match position for each pixel.

Perwass et al. have shown in [1] that the algorithm is expected to converge to a single match position and have used some test images to validate the algorithm. The aperture problem is overcome due to information diffusion from neighboring pixels and therefore local information is transformed into global information over time. No strict window of relevant information has to be pre-set or adapted to the data, thus the algorithm works on all scales. This is a real improvement compared to other, say window-based approaches, because their window is often too small or too large. Due to the selection of similarity and neighborhood probability distributions, the algorithm is pretty noise resistant and shows good matching results on artificial and real images, as has also been shown in [1].

2.1.2 Main Assumptions, Distributions and Equations

Given two images A and B, the most likely set of correspondences between the pixels of these images is searched. Of course, in real images there are not always sharp pixel correspondences and the true match of a pixel x_A in image A may lie between some pixels in image B and vice versa. Additionally, the data may be noisy and distorted due to illumation and perspective changes, so the idea is to assign a probability to correspondence pairs between images A and B.

This is implemented through random variable pairs (X_A, X_B), where X_A can take on all pixel positions in A and X_B can take on all pixel positions in B. The event $(X_A = x_A, X_B = x_B)$ means that position x_A in image A corresponds to x_B in image B. If a true match is located between pixel positions (at a *subpixel* position), its probability is expected to be distributed among the neighboring pixels. This should be a good assumption if brightness (or the used pixel value) does not change too quickly from one pixel to another. Hence, calculating match probabilities for candidate pixels x_B in the region of correspondence for some pixel x_A, these probabilitites should always sum to unity, even if the true correspondence is a subpixel position.

Before setting up the model, some symbols have to be defined: The images under inspection may be multi-modal (e.g. color images), so that their pixels are M-dimensional vectors and $\mathcal{M} := \{1..M\}$ is the set of modes. For an RGB image M is three and for a gray image M is one. The ν^{th} mode of a pixel p (its projection onto the ν^{th} axis of the color space) is accessed by p^ν. The set of pixel positions in an image of size $(Img_x \times Img_y)$ is called $\mathcal{I}$, where

$$\mathcal{I} := \{(x, y) : x \in \{1..Img_x\}, y \in \{1..Img_y\}\}$$

The subset of the power set (indicated by a $\mathcal{P}$), which does contain only sets with one element, will be needed for the definition of the probability space later on and is defined as $\mathcal{I}_P$.

$$\mathcal{I}_P := \{s \in \mathcal{P}(\mathcal{I}) : |s| = 1\}$$

The set of values S a pixel can take on is an even partitioning of the interval $[0; 1]$, that is, the pixel value range (for an 8 bit channel typically 0..255) has to be scaled to fit into this interval:

$$\mathcal{S} := \{(r/(S-1)) : r \in \{0, ..., S-1\}, S \in \mathbb{N}_{>1}\}$$

$$\mathcal{S}_P := \{s \in \mathcal{P}(\mathcal{S}) : |s| = 1\}$$

Now the images have to be modelled: A_i^ν and B_i^ν (where $i \in \mathcal{I}$, $\nu \in \mathcal{M}$,) are defined to be random variables in the probability space $(\mathcal{S}, \mathcal{S}_P, P)$, where the probability space and random variable definitions are used according to [27]. Informally speaking, there is a random variable for each pixel position and each mode, which can take on only one value at a given time. The images will always be given in this model and no further assumptions about which images are likely and which not are needed at this point.

The correspondence problem is formalized in a similar way. X_A and X_B are defined to be random variables in probability space $(\mathcal{I}, \mathcal{I}_P, P)$, i.e. they can each take on all possible pixel positions, but only one at a given time.

Additionally, a constraint is imposed, that demands only outcomes x_A of X_A and x_B of X_B to be inspected, where (x_A, x_B) is a correspondence. For (x_A, x_B) to be a correspondence, it is necessary that some point of the three-dimensional scene (of an object's surface) is projected at position x_A into image A and at x_B into image B.

Perwass et al. assume, that every pixel in A (B respectively) has such a correspondence in B (A). The set of the eight nearest neighbors (also called the 8-neighborhood) of the origin $(0, 0)$ is called $\mathcal{N}$, where

$$\mathcal{N} := \{(u, v) : u, v \in \{-1, 0, 1\}, (u, v) \neq (0, 0)\}$$

A test patch (a region where the match is searched) is defined as

$$\mathcal{T} := \{(x, y) \in \mathbb{Z}^2 : -T_x \leq x \leq T_x, -T_y \leq y \leq T_y\} \quad T_x, T_y \in \mathbb{N},$$

where $T_W = 2 \cdot T_x + 1$ and $T_H = 2 \cdot T_y + 1$ are width and height of the test patch. Thus, for symmetry reasons test patch dimensions are always odd.

The expected mean displacement from image A to B is called $d = (d_x, d_y)$. Perwass et al. center a test patch for a pixel x_A in A at $(x_A + d)$ in B (and consequently at $(x_B - d)$ in A for a pixel x_B in B). Hence, a test patch for some particular pixel $x_A = (x, y)$ is positioned in the other image in the following way:

$$\mathcal{T}_{(x,y)} := \{(x + u \pm d_x, y + v \pm d_y) \in \mathbb{Z}^2 : -T_x \leq u \leq T_x, -T_y \leq v \leq T_y\}$$

The distinction $\pm d$ is necessary due to direction dependence of d, the upper signs represent the case A to B, while the lower concatenations are used if images are matched from B to A.

The Gauss function is called $g()$, here it is defined up to a scalar factor which will be defined by the context, in which it is used:

$$g(x, y, \sigma) := \rho \exp\left(-\frac{(\|x - y\|_2)^2}{2\sigma^2}\right) \tag{2.1}$$

The s-function provides a measure of similarity between two pixels, based on their multi-modal values:

$$s(a, b) := \prod_{\nu \in \mathcal{M}} g(a^\nu, b^\nu, \sqrt{2}\sigma^\nu) \tag{2.2}$$

This is an implementation of the maximum likelihood principle. Measured pixel values are assumed to be noisy (normally distributed) samples of a true value. The s-function returns the maximum probability that both samples

stem from the same value. The standard deviation σ in the Gauss function defines the shape of the Gauss and thus the critical pixel value difference, from which on the pixels' likelihood strongly decreases. This similarity is applied to each mode and the results are multiplied to obtain the overall similarity of the two pixels.

In the same way the h-function defined in equation (2.3) describes the similarity between two displacements. It is maximized if two pixels x_A and y_A have similar displacements, i.e. $(x_A - x_B) = (y_A - y_B)$, and yields lower values the more these displacements differ.

$$h(x_A, x_B, y_A, y_B) := g(y_B - x_B, y_A - x_A, \sigma_h) \tag{2.3}$$

Now it will be explained how these functions are used to calculate the desired probabilities and where they are applied: As pointed out before, for each pixel $x_A = (x, y)$ in the image, a test patch $\mathcal{T}_{x,y}$ is positioned in the other image. All pixels $x_B \in \mathcal{T}_{x,y}$ from this patch are match candidates, and a probability distribution function (pdf) for the correct match position is computed across the patch. Without information about neighbors or about the images, all match candidates within such a patch are equally likely. Formally, given some position x_A in A, the probability of x_B to be the correspondence of x_A is:

$$P(X_B = x_B \mid X_A = x_A) = \mathcal{U}_{\mathcal{T}_{x_A}}(x_B),$$

where the uniform distribution $\mathcal{U}$ is defined on a set $\mathcal{T}$ in the following way:

$$\mathcal{U}_{\mathcal{T}}(x) = \begin{cases} x \in \mathcal{T}: & 1/|\mathcal{T}| \\ x \notin \mathcal{T}: & 0 \end{cases}$$

All random variables are always printed in capital letters, while their outcomes are printed in small letters. Some particular outcome of one of these variables is usually named the same way as the variable, except for the capital letter. Consequently, $P(X_B = x_B \mid X_A = x_A)$ can be abbreviated and may also be written as $P(X_B \mid X_A)$. However, to avoid confusion, the outcomes will be stated explicitly whenever possible.

Given the images, the probability of some pair (x_A, x_B) of pixels being a match depends on their similarity. On the other hand, given that two pixels represent a correspondence, the probability of their pixel values is given by the similarity function s of equation (2.2). If they stem from the same point in 3d-space, strongly differing values are improbable. Note that $A|_{x_A}$ is the (multi-modal) pixel value at position x_A in image A:

$$P(A|_{x_A} = a, B|_{x_B} = b \mid X_A = x_A, X_B = x_B) \simeq s(a, b) \tag{2.4}$$

Since the probabilities must sum to unity, the above similarity is only defined up to a scalar factor here. After initialization the patch is normalized and now contains a basic distribution of match probability, based only on pixel value similarity.

The second substantial assumption is that neighboring pixels will have similar displacements, that is, if (x_A, x_B) is a true match and y_A is a left neighbor of x_A, y_B is expected to be the left neighbor of x_B, while other positions of y_B have smaller probabilities. Proceeding formally in the model, neighboring match probability is defined for $(x_A - y_A) \in \mathcal{N}$, without notion of the images:

$$P(Y_B = y_B \mid X_A = x_A, X_B = x_B, Y_A = y_A) \simeq h(x_A, x_B, y_A, y_B) \tag{2.5}$$

Merging the similarity and the neighborhood constraint, Perwass et al. derive a *pixel-match pdf*, where $\simeq$ again means equality up to a scalar factor:

$$\hat{P}(X_B = x_B \mid A, B, X_A = x_A) \simeq \tag{2.6}$$

$$s(A|_{x_A}, B|_{x_B}) \frac{1}{|\mathcal{N}|} \sum_{y_A : (y_A - x_A) \in \mathcal{N}} \max_{y_B \in \mathcal{T}_{y_A}} (s(A|_{y_A}, B|_{y_B}) h(x_A, x_B, y_A, y_B))$$

Given some particular x_A, for each candidate pixel x_B its similarity is evaluated. The values (regarding s- and h-functions) of the eight nearest neighbors y_A of x_A are averaged, where for each neighbor exactly that correspondence y_B is evaluated that best satisfies the aforementioned constraints. This way a new patch distribution evolves, which is then again normalized to unity. For each pixel in image A the relevant patch is evaluated in this way.

Until now matches have always been calculated from image A to image B. Of course, there is no preference on the match direction when having two images. Hence, the same procedure is applied to the opposite direction, which yields redundant (and maybe conflicting) information, since for every correspondence candidate x_B in a patch of pixel x_A there is a probability for the opposite direction, too. The only difference is, that now x_A is a match candidate in the patch of x_B. If x_A and x_B are a true correspondence, the probabilities should be similar, motivating a direction independent probability:

$$P(X_A, X_B \mid A, B) \simeq \sqrt{P(X_B \mid X_A, A, B) P(X_A \mid X_B, A, B)} \tag{2.7}$$

By this geometric averaging false candidates are suppressed, which have no support by the inverse direction, while true matches are nearly untouched.

After normalizing, there is a new probability distribution for each patch. The new probability can then be viewed as a new measure of similarity between the pixels, taking into account more information than just the different

modes as the s-function does. Consequently, more and more global and stable information is gained using equation (2.7) instead of the s-functions in equation (2.6). In order to form an iterative algorithm from these equations, first a function f() is defined, which contains the correspondence probabilities of some round. It is initialized with pixel similarity:

$$f^0(x_A, x_B) := s(A|_{x_A}, B|_{x_B}),$$

where the set of relevant values of f^t is called $\mathcal{F}^t$:

$$\mathcal{F}^t := \{f^t(x, y) : x \in \mathcal{I}, (y - x - d) \in \mathcal{T}\}$$

The resulting pdf now depends on the iteration step t and uses information of the last iterations. It can thus be interpreted as an inhomogeneous Markov Chain as pointed out in [1]:

$$P(X_B = x_B \mid \mathcal{F}^t, X_A = x_A) \simeq \tag{2.8}$$

$$f^t(x_A, x_B)\frac{1}{|\mathcal{N}|}\sum_{y_A:(y_A - x_A)\in\mathcal{N}} \max_{y_B \in \mathcal{T}_{y_A}} \left(f^t(y_A, y_B)h(x_A, x_B, y_A, y_B)\right)$$

For $t = 0$ it is equal to the first definition of the pdf in equation (2.6). The iteration rule is completed by application of the bidirectional merging:

$$f^{t+1}(x_A, x_B) \simeq \tag{2.9}$$

$$\sqrt{P(X_B = x_B \mid X_A = x_A, \mathcal{F}^t)P(X_A = x_A \mid X_B = x_B, \mathcal{F}^t)}$$

Note that this is only a brief summary of the model and some of the equations are slightly different to [1] to avoid unneccessary complexity, e.g. some factors have been left away, which vanish in normalization anyway. For the exact model and its derivation, please refer to [1].

2.2 Improving the Assumptions

To derive the pdf of $P(X_B \mid A, B, X_A)$ in equation (2.6) and the following iterative algorithm, some assumptions have been stated explicitly or by using some probability distributions for the random variables, among which the distribution of $P(X_A \mid A, B)$ evolves to be the most important for this thesis. The implications of defining a match pair as a pair of random variables on the outcome of one of these variables alone is examined in this section, as well as the effect of wrong patch positioning on $P(X_A \mid A, B, X_B)$. Additionally, new assumptions are derived for the case that there are occluded pixels.

2.2.1 Modelling Occlusion

First the assumption has to be formalized, that there may be occluded pixels: Which probability distributions are influenced and which are not? Basically, the assumption states that there may be some pixel x_A in image A which has no match in image B, because something else occludes it. The first thing one has to notice is that in this case $P(X_B \mid A, B, X_A = x_A)$ is undefined, because conditional probability is only defined for events that have a probability greater than zero. To clarify the reader's intuition on the probabilities and events used in this model, analogous probabilities in a simpler dice experiment are inspected.

Suppose there is a red die R and a blue die B, both with six surfaces (identified by natural numbers 1..6). On each surface s a natural number is printed on, where the function R(s) (B(s) respectively) returns the number printed on surface s of the red (blue) die. Let X_R (X_B respectively) be the random variable for casting die R (B), where possible events are the surfaces and an outcome of a dice experiment is as usual the surface pointing up. Since both dice are fair, the probability of getting a particular surface x_R with die R (and B respectively) is $P(X_R = x_R \mid R) = \frac{1}{6}$.

Now both dice are cast and it can be seen that they show the same number, i.e. $R(x_R) = B(x_B)$. What is the probability that R scored x_R given that B scored x_B? Clearly,

$$P(X_R = x_R \mid R, B, X_B = x_B) = \delta_{x_R, x_B}$$

where δ represents the Kronecker-delta. Different surfaces are impossible, since the dice cannot show the same value in that case. Note that the left hand side of the above equation is undefined if the event $(X_B = x_B)$ is impossible (e.g. x_B=8) and thus the probability $P(X_B = x_B)$ is zero. Conditional probability is only defined for events with strictly positive probability.

From now on, only dice results are inspected, where both dice scored the same number, other outcomes are disregarded. Furthermore, suppose that die B is manipulated. Instead of the one, there is another six printed on surface 1, i.e. B(1)=6 and B(6)=6. Clearly, casting this die, the value one cannot be achieved. Without knowing what die B scored in particular, we want to know how probable some surface x_R of R is. Counting over all possible results yields

$$P(X_R = x_R \mid R, B) = \begin{cases} x_R = 1 & : \quad 0 \\ 1 < x_R < 6 & : \quad \frac{1}{6} \\ x_R = 6 & : \quad \frac{2}{6} \end{cases}$$

Since the assumption "both dice scored the same" stated $R(x_R) = B(x_B)$, there is no correspondence for surface one on R, where a one is printed on, while there are two possible valid B-surfaces for $x_R = 6$. Hence, without knowing what die B scored, the outcomes of R are not equally distributed, i.e. they are not statistically independent of the dice's nature.

Changing the point of view (and knowledge), the reverse direction is inspected now: After executing a very large number of experiments with the dice R and B (having many more than six surfaces), where again only outcomes are counted where both dice scored the same, some surface x_1 on R can be observed more often than some other surface x_2 on R. Again, the exact result for B is unknown.

$$P(X_R = x_1 \mid R, B) > P(X_R = x_2 \mid R, B)$$

What does this mean for these surfaces x_1 and x_2? There must be more surfaces on die B with number $R(x_1)$ than with $R(x_2)$!

Returning to the matching model, there are neither such exact observations nor are there sharp probabilities, but the intuition is comparable: How likely is it that a pixel x_A at some position has a correspondence? How likely is it, that some random variable X_A takes the value x_A, if not every pixel actually has a match?

This strongly depends on the images A and B, since the probability for X_A to take on a value x_A is only greater than zero if there is a corresponding x_B, which can be taken on by X_B, so that (x_A, x_B) is a correspondence pair. Remember that a correspondence pair is a pair of pixels (z_A, z_B), where z_A is in image A and z_B is in image B, so that there is a point z in 3d-space (belonging to some object), which is projected into image A at position z_A and into image B at position z_B.

If such a pair does not exist for some x_A, e.g. because the correspondence is beyond a border or occluded, the probability is zero, i.e. under the correspondence pair assumption, the random variable will never take on that value. Hence, the event $(X_A = x_A)$ is no more independent of the chosen images A and B and thus $P(X_A = x_A, A, B)$ is no more separable into two independent probabilities $P(X_A = x_A)P(A, B)$. Furthermore, $P(X_A|A, B)$ is not equal for all x_A.

Quite the reverse, the value $P(X_A = x_A \mid A, B)$ can give us a hint about the probability that x_A is half-occluded as we have seen in the dice example. If for some reason, e.g. by a sophisticated computation, it is known that $P(X_A = x_A \mid A, B)$ is zero or extraordinarily small for some x_A, it is very likely that x_A has no correspondence in B. Of course, all considerations also apply to the B to A matching direction.

2.2.2 Wrong Patch Positioning

By now, the probabilities for a test patch are normalized such that their sum yields unity. This is based on the assumption that the match is in the test patch and not anywhere else in the image. This is not necessarily true, since it is possible that the patch is not well-positioned. If the assumption of correct patch positioning is formalized explicitly, the probabilities of the patches are *a posteriori probabilities*, given that the true match (x_{true}) of a pixel x_A is actually in the patch:

$$\sum_{x_B \in \mathcal{T}_{x_A}} \hat{P}(X_B = x_B \mid A, B, X_A = x_A, x_{true} \in \mathcal{T}_{x_A}) = 1$$

This may be re-converted into an *a priori probability* (regarding the positioning). Using the definition of conditional probability yields:

$$\sum_{x_B \in \mathcal{T}_{x_A}} \hat{P}(X_B = x_B \mid A, B, X_A = x_A) = P(x_{true} \in \mathcal{T}_{x_A} \mid A, B, X_A = x_A) \tag{2.10}$$

If there is information about how well the patch has been positioned, it is a good idea to initialize the patch normalization with an appropriate value. However, almost all patches are usually well-positioned and no robust way of estimating this probability has been found yet, so that patches are always assumed to contain the correct match if there is one in the other image.

2.2.3 Integrating the Borders

Image borders are an intricate problem in image processing, especially in matching. Most algorithms are designed to work on infinitely large images and do not take into account, that the image size is limited. Mostly this problem is put aside as 'border effects' or even left to the implementation as a parameter optimization (or guessing) problem.

Clearly, there is less information at the borders and therefore small images with large border areas (with respect to the whole image area) are problematic. To some degree operations cannot be executed at the borders, but it is desirable that borders are not treated explicitly but as part of the model, at least as far as possible. The given model provides an acceptable solution of this problem.

From a physical point of view borders can be regarded as infinitely expanded (e.g. black) bars very close to the camera (e.g. an aperture) which

hide the rest of the scene. If a pixel gets out of sight from one to another image, it is actually occluded by the black bars in front of the camera. Hence, if the algorithm is occlusion tolerant, no separate handling of the borders is necessary. The image can be regarded as arbitrarily (or even infinitely) large, but we do only calculate the region of interest, which is the image in the original sense.

2.2.4 Other invariant Properties for Similarity

At this point it is possible to unite the world of feature and dense pixel based matching algorithms. The standard pixel similarity is based on RGB values. As pointed out before, the model is also applicable to gray level images or any other images with some invariant pixel properties, for which a continuous similarity measure can be defined.

Promising features may be corners, edges or those provided by the structure multivector defined in [21]. Standing alone, accurate feature information (e.g. corners) is usually very sparse, but using information diffusion based on dense RGB information should give a robust base for invariant and characteristic features. Usually feature extraction algorithms mark some pixels, where the feature has been detected, while the other pixels are defined to lack this feature. Thus, in such a feature image only the probabilities one (present) and zero (absent) occur. To achieve subpixel accuracy in matching and to weight the RGB information more heavily, it is a good idea to smooth the feature image, which corresponds to the idea of a continuous similarity measure for the feature. However, the selection and adaptation of appropriate features is beyond the scope of this thesis and will not be discussed here.

2.3 Adapting the Model

For the original derivation of the match probability distribution, two assumptions have been made in [1], that turn out to be no more correct, if occluded pixels are allowed in the images. Before re-deriving the pdf under the new assumptions, the two main differences are briefly explained:

The original information propagation throughout the image (using the pixel-match pdf of equation (2.8) respective (2.6)) relies on the assumption that the neighboring patch probabilities are equally reliable and therefore the expectation value can be calculated by non-weighted averaging the neighboring patches. It is assumed that every neighbor has a match.

Instead, in the new model, half-occluded pixels may occur, which must not be used for expectation value calculation from a probability theoretic

point of view. Neighboring pdfs should only be used if the neighbor actually has a match in the other image. Hence, in the derivation of the pixel-match pdf neighbors are not assumed to have a correspondence and the pixel-match pdf has to be calculated in a different way.

Furthermore, the distributions $P(X_A \mid A, B)$ and $P(X_B \mid A, B)$ have been assumed to be constant in [1]. As pointed out in section (2.2.1), these distributions depend heavily on the images A and B and are usually not even constant for some fixed pair of images A and B. These distributions are unknown and contain information about which pixels are occluded. Hence, they must not be left to some neglectable scalar factor but have to be used adequately.

2.3.1 Re-Deriving the Pixel Match PDF

Unless stated explicitly and with exception of the previous section, all probability distributions and symbols stay the same as in the former model. Of course, allowing occlusion still means that most of the pixels must have a match to compute some meaningful displacement field. The derivation of the pixel-match pdf is done over large parts in analogy to the derivation done in [1].

Again, let (X_A, X_B) and (Y_A, Y_B) be the random variables of two neighboring pixel correspondences, i.e. for some $X_A = x_A$ only outcomes y_A of Y_A are inspected, where $(x_A - y_A) \in \mathcal{N}$. This constraint is not stated in every probability of this section as an a priori assumption, though it would be correct to do so. However, for the sake of readability it is not explicitly printed and left to the reader to keep in mind.

Now starting to derive a new pixel-match pdf, Bayes' formula (see [27]) states that

$$P(X_A, Y_A, X_B, Y_B \mid A, B) = \frac{P(A, B \mid X_A, Y_A, X_B, Y_B) P(X_A, Y_A, X_B, Y_B)}{P(A, B)}$$

The goal of this derivation is to find a probability distribution for match candidates x_B (as outcomes of X_B), given some pixel x_A (as outcomes of X_A). The pdf should only assume X_A and the images A and B to be given and no other neighboring correspondences.

Remember that small letters stand for the outcomes of the corresponding capital random variables (in particular x_A, x_B, y_A and y_B). To keep these equations readable, the outcomes are not printed explicitly in this derivation. Thus solving for $P(X_B = x_B, Y_B = y_B, Y_A = y_A \mid A, B, X_A = x_A)$ after utilizing the definition of conditional probability (according to [27]) yields:

$$P(X_B, Y_B, Y_A \mid A, B, X_A) =$$

$$P(A, B \mid X_B, Y_B, X_A, Y_A) P(X_B, Y_B \mid X_A, Y_A) \frac{P(X_A, Y_A) P(A, B)}{P(A, B) P(A, B, X_A)}$$

Removing all constant factors and constraining the images to the relevant pixels, the pdf can be expressed as:

$$P(X_B, Y_B, Y_A \mid A, B, X_A) \simeq$$

$$\frac{P(A|_{x_A}, B|_{x_B} \mid X_B, X_A) P(A|_{y_A}, B|_{y_B} \mid Y_A, Y_B) P(X_B, Y_B \mid X_A, Y_A)}{P(X_A \mid A, B)}$$

This is - except for the denominator - identical to the former model's derivation. Substituting the s- and h-functions as before (see equation 2.6) yields:

$$P(X_B = x_B, Y_B = y_B, Y_A = y_A \mid A, B, X_A = x_A) \simeq \tag{2.11}$$

$$\frac{s(A|_{x_A}, B|_{x_B}) s(A|_{y_A}, B|_{y_B}) h(x_B, y_B, x_A, y_A)}{P(X_A = x_A \mid A, B)}$$

Note that in the probability at the left hand side the correspondence Y_A is no more a priori assumed to be given - in contrast to [1], because the fact that y_A actually has a correspondence is also no longer assumed to be given. The equation describes a joint probability for candidate x_B regarding the neighbor y_A, which can become zero, if there is no correspondence y_B for y_A. In the former definition, the whole pixel-match pdf of equation (2.6) is mathematically undefined, if a neighbor is occluded, since one of the neighboring probabilities used for expectation value calculation[1] is undefined in that case. In the new model, the uncertainty introduced by dropping the precondition of full correspondence for neighboring pixels will be exploited later on. However, all four random variables still occur in the new distribution, though only a statement about the match probability of x_A and x_B is desired.

Analogous to the former derivation, at first this probability distribution is made independent of some particular match candidate y_B of the neighbor. Therefore the best matching y_B (regarding equation (2.11)) is assumed to be the match of y_A and its match information is used to evaluate the neighborhood and similarity constraint for x_A.

$$\hat{P}(X_B, Y_A \mid A, B, X_A) := \rho \max_{y_i} \hat{P}(X_B, Y_B = y_i, Y_A \mid A, B, X_A)$$

This is the joint probability that x_B is the correct match for x_A (given that x_A has a match) and that the neighboring pixel y_A has a similar correspondence in its test patch.

[1] Please refer to [1] for the detailed derivation of the former model.

However, the match probability for x_B should be stated independently of a particular neighbor y_A of x_A. Using the law of total probability the previous joint probability can be summed over all y_A to become independent of y_A. Two conditions must hold for this to be correct: When summing over events of the random variable Y_A, the sample space has to be covered completely and there must not be an overlap of any of these events. Since y_A and x_A are neighbors, the sum must run over all eight neighbors of x_A to be complete. These events are disjoint as can be easily seen.

$$\hat{P}(X_B \mid A, B, X_A) := \sum_{y_A:\ (y_A - x_A) \in \mathcal{N}} \hat{P}(X_B, Y_A = y_A \mid A, B, X_A)$$

This is in contrast to the previous definition, where Y_A is assumed being given and the expectation value is calculated over the neighbors. Here, another way is chosen, where all joint probabilities are summed without such an assumption.

Now equation (2.11) is used in this pdf, which finally yields a new match probability distribution, that is quite similar to the previous one:

$$\hat{P}(X_B = x_B \mid A, B, X_A = x_A) \simeq$$

$$\frac{s(A|_{x_A}, B|_{x_B})}{P(X_A = x_A \mid A, B)} \cdot \sum_{y_A:(y_A - x_A) \in \mathcal{N}} \max_{y_B} \left(s(A|_{y_A}, B|_{y_B}) h(x_A, x_B, y_A, y_B)\right) \tag{2.12}$$

Note that the division by $P(X_A \mid A, B)$ is independent of the sum and is thus only a normalization factor. It will be compensated later, since an inverse factor is introduced by bidirectional merging. Furthermore, since this probability is equal for all candidate pixels in the patch, it vanishes in normalization anyway. Note that there is no such division by a similar probability for y_A.

When comparing the old (equation (2.6)) and the new pdf (equation (2.12)), they differ only in two factors: $(1/|\mathcal{N}|)$ and $(1/P(X_A \mid A, B))$. In each model one factor seems to be missing, while the other is present. Remember that during both derivations, some factors (which are assumed to be constant) have been left away and that the probabilities have only been defined up to a scalar factor. Hence, the missing factors are mainly hidden in the assumptions what is constant.

2.3.2 Direction Merging

Since $P(X_A = x_A \mid A, B)$ is no more constant, the bidirectional merging has also to be changed. By definition of conditional probabilty it is known

that making the patch probabilities direction independent introduces an additional factor:

$$P(X_A, X_B \mid A, B) = P(X_A \mid A, B)P(X_B \mid A, B, X_A) \tag{2.13}$$

$$P(X_A, X_B \mid A, B) = P(X_B \mid A, B)P(X_A \mid A, B, X_B) \tag{2.14}$$

Comparing these equations to those from [1] it can be easily seen, that again the only missing factor is the correspondence probability, which is assumed to be constant in the former model. Multiplying equations (2.13) and (2.14) and calculating the square root yields:

$$P(X_A, X_B \mid A, B) =$$

$$\sqrt{P(X_B \mid A, B)P(X_A \mid A, B, X_B)P(X_A \mid A, B)P(X_B \mid A, B, X_A)} \tag{2.15}$$

This provides a new stabilized measure of similarity between the two pixels, that can be used in the next iteration.

However, now one has to distinguish between the direction dependent match probability, that is located in a patch (see equation (2.12)), and the overall (direction-independent) probability that two pixels correspond to each other (left hand sides of equations (2.13) and (2.14)). Apparently, they differ by a factor of $P(X_A \mid A, B)$ respective $P(X_B \mid A, B)$.

This is quite important and states the main difference of the present model compared to the former one, where the distinction between $P(X_A, X_B \mid A, B)$ and $P(X_B \mid A, B, X_A)$ vanishes due to the assumed constancy of the missing factor.

2.3.3 Occlusion Detection

In the previous sections a new pixel-match pdf and a new rule for bidirectional merging have been derived. Both depend on the probability distributions $P(X_A \mid A, B)$ and $P(X_B \mid A, B)$, which have been assumed to be constant in the former model. In section (2.2.1) it has been shown that this assumption is not valid in presence of occlusion. Consequently, this pdf is no longer given and must therefore be computed in some way.

Before it is defined formally, the principle used for occlusion detection is explained: Suppose two images are matched using the former algorithm. If everything works as expected, the test patches converge to a single position. Now suppose that there are some occluded pixels, e.g. a pixel x_O in A, which has no correspondence in image B. It is very likely that also the test patch of x_O converges to a single match position, say x_B in B, which represents the most likely match for x_O. However, if x_B has a true match x_A in A, x_B's

test patch will most likely converge to this position and vice versa. That is, x_O's position (in the test patch of x_B) should have a very low probability. One might say that x_O chose x_B as a correspondence partner but does not get support from the inverse direction.

Now, the observation that occluded pixels do not get support from the inverse direction is exploited to detect them. Since the left hand sides of equations (2.13) and (2.14) are identical, the right hand sides have to be equal, too:

$$P(X_A \mid A, B)P(X_B \mid A, B, X_A) = P(X_B \mid A, B)P(X_A \mid A, B, X_B) \quad (2.16)$$

This equation can be evaluated for every single match candidate in the test patch and may be a measure of occlusion probability in some way. But it is quite probable that there will be noisy results at the borders and information will diverge over the patch, i.e. some candidates will fulfill the equality, for some the left hand side will be smaller and for some the right. The problem, which match candidate is the relevant one for occlusion detection is hardly decidable, if the patch has not yet converged completely.

Instead of inspecting single correspondences, it is more promising to evaluate all possible matches in the other image and thus to get more robust information. As pointed out before, the above stated equality holds for any pixel pair (x_A, x_B). Particularly, given some pixel x_A, it is not wrong for pairs (x_A, x_B) where x_B can be every match candidate of x_A. A set of $(T_W \cdot T_H)$ equations of the type given in equation (2.16) may be set up. Summing up the right hand sides and the left hand sides of equations for the whole test patch of x_A yields:

$$\sum_{x_B \in \mathcal{T}_{x_A}} P(X_A = x_A \mid A, B, X_B = x_B)P(X_B = x_B \mid A, B) =$$
$$\sum_{x_B \in \mathcal{T}_{x_A}} P(X_B = x_B \mid A, B, X_A = x_A)P(X_A = x_A \mid A, B)$$

Note that the probability $P(X_A \mid A, B)$ at the right hand side is independent of the sum and can be moved in front of it. Hence, only the patch probabilities are left inside the sum. Remember that summing the probabilities over the test patch yields the normalization, which is usually unity. Since the test patch is normalized to a strictly positive value at any rate, the above equation may be divided by this sum.

$$P(X_A = x_A \mid A, B) = \frac{\sum_{x_B \in \mathcal{T}_{x_A}} P(X_A = x_A \mid A, B, X_B = x_B)P(X_B \mid A, B)}{\sum_{x_B \in \mathcal{T}_{x_A}} P(X_B = x_B \mid A, B, X_A = x_A)} \quad (2.17)$$

Since there is no preference on matching from A to B or from B to A, the same can be applied to the B to A direction analogously:

$$P(X_B = x_B \mid A, B) = \frac{\sum_{x_A \in \mathcal{T}_{x_B}} P(X_B = x_B \mid A, B, X_A = x_A) P(X_A \mid A, B)}{\sum_{x_A \in \mathcal{T}_{x_B}} P(X_A = x_A \mid A, B, X_B = x_B)} \tag{2.18}$$

It can be seen quite easily, that the probability $P(X_A \mid A, B)$ does indeed depend on the support of the inverse direction. Unfortunately, it does also depend on the correspondence probability of that direction. Since both probabilities depend on one another, they cannot be calculated explicitly before the start of the algorithm. Instead, each correspondence probability has to be initialized with some value and is updated iteratively utilizing the above equations, which is referred to as the *collection of support* in the other image. A significant correspondence probability for every pixel in the images may then finally evolve this way. Informally, smaller probabilities mean greater occlusion likelihood, greater ones mean higher two-to-one match probability, with too much support from the inverse direction.

Which probability exactly means that the pixel is occluded and which means that it is certainly not? The answer to that question is image specific and does also depend on some other values, like the test patch size, the iteration number and so on. However, an approach is made to roughly estimate the critical size for occluded pixels.

Since $P(X_A = x_A \mid A, B)$ is a probability (of the event x_A), summing it over all possible (disjoint) events must yield unity:

$$\sum_{x_A \in \mathcal{I}} P(X_A = x_A \mid A, B) = 1$$

Now, remember that there are $|\mathcal{I}|$ pixels in image A and thus $|\mathcal{I}|$ outcomes for the random variable X_A. If every pixel has a match (assumption of complete correspondence (CC)), there is no preference for X_A to take on some value and thus $P(X_A \mid A, B)$ may be assumed to be a uniform distribution:

$$P(X_A = x_A \mid A, B, CC) = \mathcal{U}_{\mathcal{I}}(x_A)$$

If a single pixel x_O in A is occluded, the true distribution for X_A yields slightly higher values for all pixels having correspondences and zero for $(X_A = x_O)$. The more pixels are occluded, the more the correspondence value for pixels with correspondences increases. That is, in theory occluded pixels have zero probability while the others always have a probability of at least $1/|\mathcal{I}|$.

However, there will be noise on the distributions and there will always be some spurious support for and from occluded pixels, so that these sharp distinctions cannot be made. Hence, a pixel x_O should only be classified as occluded, if

$$P(X_A = x_O \mid A, B) \ll \frac{1}{|\mathcal{I}|}$$

For formal reasons the case $P(X_A = x_A \mid A, B) = 0$ has to be inspected, since the probability occurs at some denominators and the event is used as an a priori assumption. In case it was known that x_A has no match at all in B due to occlusion, $P(X_B \mid A, B, X_A = x_A)$ would be undefined, since $P(X_A = x_A \mid A, B) = 0$. But in this model there is no such knowledge, and probabilities are calculated from similarities. The assumed distributions will never reduce a probability to zero, since the Gauss function used for similarity will never yield zero. This is no inaccuracy due to calculation, it is simply the uncertainty given by the model.

Hence, every pixel in the test patch always has a (possibly very small) match probability greater than zero. Thus, the correspondence probability $P(X_A = x_A \mid A, B)$ is also always strictly positive. However, it should be clear that if a pixel is detected to be occluded, i.e. it has a very low correspondence probability, the patch distribution $P(X_B = x_B \mid A, B, X_A = x_A)$ should be treated as undefined.

Note that intrinsically the correspondence probability is some kind of measure for the support from the inverse direction. Additional to detecting occluded pixels it also yields a low value in case of low similarity (and great uncertainty) for true matches. Hence, this probability can also be a hint for the confidence in a pixel's match value, indicating how well the pixel and the neighborhood are found in the other image.

2.3.4 Consequences on the Test Patch Size

Though the algorithm itself does not depend heavily on the test patch size, the occlusion detection may do. The test patch probabilities are relative to the other probabilities in the patch due to the normalization. It is very likely that for large patches the neighborhoods differ somewhat, because depth discontinuities can touch the patch, maybe only in areas, which are not relevant for the actual match, but these pixels influence the patch and thus the support collection. Ideally, support should only be evaluated for relevant pixels and thus the patch size should be as small as possible.

On the other hand, perspective and illumination changes affect very small patches more heavily: If the match is not located at the patch center but at the border, incomplete neighborhoods may be used. Some pixel giving

true support may lie out of the patch and is therefore not evaluated. Such problems are better compensated in larger patches, since the probability of positioning them completely wrong is smaller.

Hence, the occlusion detection may be trapped in a similar dilemma of trading between small and large patches as the window-correlation approaches for matching (compare [13]).

2.4 The Algorithm

2.4.1 Propagation of Local Constraints

As it is in the former model, the neighborhood constraint and the pixel-match pdf refer only to direct neighbors of a pixel. For many images, more global information is needed for stable matching results, since the aperture problem is very relevant for these small neighborhoods. This section binds the derived probability distributions and equations into an iterative algorithm, where local information propagates throughout the images step by step.

Analogous to the former model, the function f^t is defined to contain the similarities from the t^{th} iteration and is initialized with pixel similarity:

$$f^0(x_A, x_B) := s(A|_{x_A}, B|_{x_B})$$

Note that in the iteration equations of the old and the new model, the $f^0()$-function is used for the probability $P(X_A, X_B \mid A, B)$, while originally $P(A, B \mid X_A, X_B)$ (see equation (2.4)) has been used at that location in the pixel-match pdfs. The above mentioned probabilities may be used synonymously here, since they are equal up to a scalar factor:

$$\begin{aligned} P(X_A, X_B \mid A, B) &= P(A, B \mid X_A, X_B)\frac{P(X_A, X_B)}{P(A, B)} \\ &\simeq P(A, B \mid X_A, X_B) \end{aligned}$$

Proceeding in the algorithm, there are functions c_A (for image A) and c_B (for image B) to represent the correspondence probabilities for the pixels at each iteration. They are initialized with their expectation value here:

$$c_A^0(x_A) := 1/|\mathcal{I}|$$

$$c_B^0(x_B) := 1/|\mathcal{I}|$$

Again, $\mathcal{F}^t$ contains the information available at iteration t.

$$\mathcal{F}^t := \{f^t(x, y) : x, y \in \mathcal{I}\} \cup \{c_A^t(x) : x \in \mathcal{I}\} \cup \{c_B^t(x) : x \in \mathcal{I}\}$$

The resulting pdf can then be written as:

$$\hat{P}(X_B = x_B \mid \mathcal{F}^t, X_A = x_A) \simeq \tag{2.19}$$

$$\frac{f^t(x_A, x_B)}{c_A^t(x_A)} \sum_{y_A:(y_A - x_A)\in\mathcal{N}} \max_{y_B \in \mathcal{T}_{y_A}} \left(f^t(y_A, y_B) h(x_A, x_B, y_A, y_B) \right)$$

Apparently, the above equation uses a correspondence probability of the previous iteration. Compared to the former algorithm, there is an additional step necessary to compute these probabilities.

$$c_A^{t+1}(x_A) = \hat{P}(X_A = x_A \mid \mathcal{F}^t) \simeq \frac{\sum_{x_B \in \mathcal{T}_{x_A}} \hat{P}(X_A = x_A \mid \mathcal{F}^t, X_B = x_B) c_B^t(x_B)}{\sum_{x_B \in \mathcal{T}_{x_A}} \hat{P}(X_B = x_B \mid \mathcal{F}^t, X_A = x_A)} \tag{2.20}$$

Finally, the iteration rules are completed by application of the bidirectional merging:

$$f^{t+1}(x_A, x_B) = \hat{P}(X_A, X_B \mid \mathcal{F}^t) \simeq \tag{2.21}$$

$$\sqrt{\hat{P}(X_A = x_A \mid \mathcal{F}^t, X_B = x_B) c_B^t(x_B) \hat{P}(X_B = x_B \mid \mathcal{F}^t, X_A = x_A) c_A^t(x_A)}$$

2.4.2 Comparison to the Former Model

Now that both models are complete, the differences of the old and the new one regarding the direction dependency of the match probabilities can be inspected. This shows the key difference between the two approaches. For clarity reasons, the $f()$-functions are replaced by their probability-theoretic meaning, which is $P(X_A, X_B \mid \mathcal{F})$. Since the bidirectional merging is not important for these considerations, it can be disregarded here and one may imagine that the following iteration rule is used in both models instead of equation (2.21) respective equation (2.9):

$$P(X_A, X_B \mid \mathcal{F}^{t+1}) = P(X_B \mid \mathcal{F}^t, X_A) P(X_A \mid \mathcal{F}^t)$$

That is, there is no bidirectional stabilization and the values of the last iteration are directly reused. Clearly, $P(X_A \mid \mathcal{F}^t)$ is only a constant factor in the old model and is thus normalized away. However, for comparison purposes it is kept here.

In appendix A it is shown that in the old derivation of the pixel-match pdf, the constant factors $(1/P(X_A \mid A, B))$ and $(1/P(Y_A \mid A, B))$ have been left away. To make the old model comparable to the new one, suppose for the moment, that these factors are present in the old pixel-match pdf. Since they are constant there, they make no difference in a "$\simeq$" relation.

$$P_{old}(X_B \mid \mathcal{F}^{t+1}, X_A) \simeq \tag{2.22}$$

$$\frac{P(X_A, X_B \mid \mathcal{F}^t)}{P(X_A \mid \mathcal{F}^t)} \frac{1}{|\mathcal{N}|} \sum_{y_A} \max_{y_B \in \mathcal{T}_{y_A}} \left(\frac{P(Y_A, Y_B \mid \mathcal{F}^t)}{P(Y_A \mid \mathcal{F}^t)} P(X_B, Y_B \mid X_A, Y_A) \right)$$

$$= P(X_B \mid \mathcal{F}^t, X_A) \frac{1}{|\mathcal{N}|} \sum_{y_A} \max_{y_B \in \mathcal{T}_{y_A}} \left(P(Y_B \mid \mathcal{F}^t, Y_A) P(X_B, Y_B \mid X_A, Y_A) \right)$$

It is obvious that in this equation, there is no direction independent match probability of any two pixels. That is, to compute the direction dependent match probability $P(X_B \mid \mathcal{F}, X_A)$ of the next iteration, actually only direction dependent probabilities of that direction are used. This agrees with the intuition of unidirectional matching, where only those probabilities are available.

Now, inspecting the new model, there is only a factor $(1/P(X_A \mid A, B))$, which is expressed by $(1/c_A^t(x_A))$ in equation (2.19), but there is no factor $(1/P(Y_A \mid A, B))$ in the sum of the pixel-match pdf. Suppose the same interpretation is used and the direction dependent match probabilities of the next iteration are calculated using only probabilities of the same direction. Those referring to x_A may be handled as before, since the factor is present for x_A, but those referring to y_A have to be multiplied by y_A's correspondence probability in order to compensate for the missing factor.

$$P_{new}(X_B \mid \mathcal{F}^{t+1}, X_A) \simeq$$

$$P(X_B \mid \mathcal{F}^t, X_A) \sum_{y_A} P(Y_A \mid \mathcal{F}^t) \max_{y_B \in \mathcal{T}_{y_A}} \left(P(Y_B \mid \mathcal{F}^t, Y_A) P(X_B, Y_B \mid X_A, Y_A) \right)$$

Since the factor $P(Y_A \mid \mathcal{F}^t)$ is independent of the maximization regarding y_B, it may be moved in front of this maximization. Hence, it can be interpreted as a weighting factor when summing up the neighbors. Note that this is no implementation detail and no artificial factor, but that the pixel-match pdf is actually influenced by the correspondence probabilities of the neighbors.

This is a very important feature of the new algorithm, since pixels with reduced correspondence probability have less influence on their neighbors. As it has been pointed out before, occluded pixels and such with a noisy correspondence are exactly those with reduced support and therefore reduced correspondence probability. Conflicting match information from those pixels does not (or not so heavily) propagate further, which promises an optimization of the matching result. Nevertheless, since the correspondence probability will never reach zero, these weights are relative. Even if all neighbors have

a low correspondence probability, each pixel still depends on its neighbors: Occluded pixels (or those supposed to be occluded) still have to choose the candidate, that best satisfies the assumed constraints. The same applies to pixels in noisy areas with weak support.

Furthermore, occluded pixels often form contour lines at depth discontinuities. Depth discontinuities in the scene mean also discontinuities in the displacement fields, which are often smoothed due to faulty information propagation across these borders. Since the influence of occluded pixels to both directions is reduced, it may be possible to sharpen the displacement maps with this approach because the diffusion across the occlusion lines is reduced. Pixels to both sides of occlusion lines are only influenced by the regions they belong to. Hence, a good occlusion detection can also help high level applications to segment the image.

2.4.3 Schematic Overview

Now all equations and update rules for the algorithm have been set up. By applying the equations and iterating several times, probabilities will be updated with respect to neighboring probability patches and therefore information will diffuse through the whole image. Note that the equations have only been stated explicitly for the A to B direction. Of course they have to be applied to the B to A direction (analogously), too. The following is a schematic overview of the steps to be executed:

1. Position Test Patches
2. Set up Test Patch Normalization (Probability of Correct Positioning)
3. Estimate Correspondence Probabilities (Inverse Occlusion Measure)
4. Initialize Test Patches with Pixel Similarity
5. Apply Neighborhood Constraint (Pixel-Match-PDF from (2.19))
6. Update Correspondence Probabilities (Support from (2.20))
7. Stabilize Directions (Bidirectional Merging from (2.21))
8. Continue at Step 5

Compared to the previous algorithm, step (3) and step (6) are new. Note that test patches are normalized to the value set in step (2) after applying steps (4), (5) and also after step (7) and that correspondence probabilities also have to be normalized after step (6).

2.4.4 Embedding in the Old Model

The new algorithm has been derived from the same equations as in the former model, but uses a slightly different way and also a few other assumptions. However, one may ask, whether there is an image pair respective some probability distribution, such that the new algorithm embeds into the old one. It is obvious, that for this purpose the images have to be chosen in a way, that the pdfs $P(X_A \mid A, B)$ and $P(X_B \mid A, B)$ are uniform distributions, since these probabilities are assumed to be constant in the old model.

Suppose some particular images A and B are given, such that the above mentioned pdfs are detected to be uniform distributions at each iteration. As pointed out before, for some pixel position x, the correspondence probability has to have the value $1/|\mathcal{I}|$ in that case:

$$c_A^t(x) = c_B^t(x) =$$

$$P(X_A = x \mid A, B) = P(X_B = x \mid A, B) = \frac{1}{|\mathcal{I}|} \;\; \forall\, x \in \mathcal{I}, t \in \mathbb{N}$$

Note that this assumption implies that there is full correspondence between the images (no occluded pixels allowed).

It will now be shown that the algorithms behave in the same way, i.e. they differ only by a scalar factor which vanishes during the normalization. There are two steps to be inspected:

- Neighborhood Evaluation
- Bidirectional Merging

Having a closer look at the first one, it is obvious that both pdfs only differ by a few factors: The old one (according to equation (2.8)) contains the constant factor $1/|\mathcal{N}|$, while the new one (see equation (2.19)) contains the factor $1/P(X_A \mid A, B)$, which is $1/|\mathcal{I}|$ by assumption and thus also constant. Hence, the patches are identical after normalization. The bidirectional merging also differs only by constant factors $1/P(X_A \mid A, B)$ and $1/P(X_B \mid A, B)$ under the square root. That is, before normalization all values are scaled by $1/|\mathcal{I}|$ in the new model (see equation(2.21)) compared to the old one (as in equation (2.9)). Again, this is compensated for by normalization, so that the algorithms actually behave the same.

For this scenario, the convergence proof of the algorithm is absolutely analogous to the one given in [1], which is cited in appendix B for completeness.

Chapter 3

Stereo and Optic Flow Integration

In the previous chapter an algorithm has been derived, which can match two similar images even in presence of occlusion. However, there are still some parameters of the test patches, which may be critical for the matching results: For instance, it is clear that the smaller the patches are, the more exact the positioning must be, since the true match has to be in the patch and a completely wrong positioned patch prevents the algorithm from finding the correct match. On the other hand, when using large patches, many matches are likely at the first iteration steps and diffusion will take quite a while. Large patches will additionally give spurious support for occluded pixels, so that the occlusion detection needs many iterations, before acceptable results are obtained, which are often very noisy in that case. Furthermore, the larger the patches are, the more resources will be needed to execute the algorithm.

Unfortunately, in general there is no information about where to position the test patches in advance and what size they should have. Hence, an approximation has to be used, where a maximum displacement between the images is assumed. The test patch size and position is then adapted to this displacement, i.e. all pixels have test patches of that size and are usually positioned with respect to some kind of estimated mean displacement. This will usually be a waste of resources, since patches will (for safety reasons) be much greater than they needed to be.

Secondly, in areas with intrinsically weak structure multiple match candidates may be equally likely at the start of the algorithm. In presence of depth discontinuities they may be matched incorrectly due to conflicting information from border regions. Furthermore, some spurious match candidates may become more likely than the true ones in presence of noise or perspective distortions.

Hence, there are some details that may be optimized if there is more information about how the images have been created and thus about what kind of data is expected. This chapter shows how additional assumptions for image sequences or for stereo images can be imposed and exploited, which optimize the test patch positioning and make the algorithm more robust in areas without strong structure or for noisy images. Since the algorithm structure has to be kept in principle, these constraints have to be integrated into the probability distributions (e.g. for neighborhood or similarity), into initialization values or into the patch positioning.

3.1 Stereo Matching

3.1.1 Properties of Stereo Images

In stereo imaging, two images of the same scene are taken simultaneously, but from slightly different positions, usually by two cameras A and B. In our scenario, images may not differ too much and therefore cameras should point into a similar direction and have to be quite close to each other (compared to their distance to the scene). The distance of the cameras is measured along the connection line between the two optical centers of the cameras, which is also called the *baseline* of the stereo system.

A common geometry of the stereo system is that the optical axes of the cameras used are parallel, the image planes lie in the same plane in 3d-space, and the image lines are parallel to the baseline of both cameras, such that the displacement of pixels between left and right image is only horizontal. In stereo matching applications this displacement is called *disparity*. Images created by a stereo system not fulfilling the above properties, but being transformed in a way that they could have been taken by such a system are called *rectified*. Since these image pairs cannot be distinguished from those pairs really created by such a stereo system, subsequently all image pairs fulfilling these properties will be called rectified.

For rectified stereo images, true correspondences differ only in their horizontal position and test patch heights of three or even one pixel are sufficient for a good match. The minimum test patch width depends on the distance of the nearest objects to the cameras, since the nearer the object is, the greater the disparity will be. This can be seen in figure (3.1), where object O_1 is closer to the cameras than object O_2. Hence, O_2 is projected nearly to the same position in both images, while O_1's position changes remarkably. In large image pairs great disparities may occur occasionally (for foreground objects), while most pixels may have smaller ones. Disparity can vary strongly over

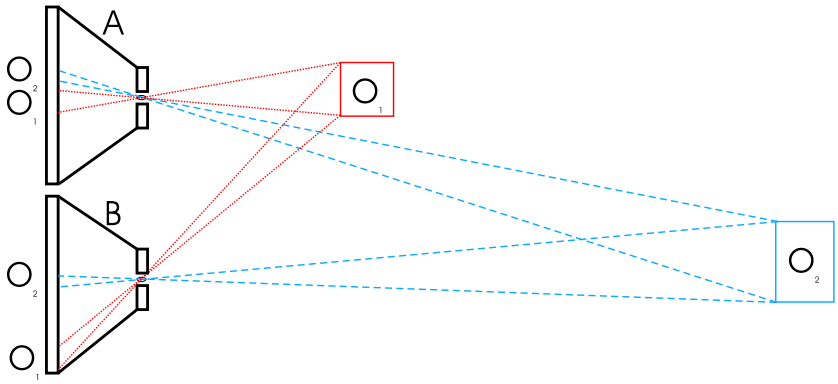


Figure 3.1: Disparity of objects with different distances to the cameras

the images and from scene to scene, so that adaptive test patch positioning and sizing may be promising.

Note that rectified image pairs have only been used to strengthen the reader's understanding of the problems in stereo matching. In this model the assumptions about the stereo system are not that strict. Corresponding pixels may differ in their vertical position, too, and optical axes do not have to be parallel. If the strict assumptions hold yet, the test patch height may be reduced to one, as has been shown in [1], but having such stereo images is not a precondition for the proposed algorithm.

Another problem of stereo images is that due to different perspectives occluded pixels do not only occur separately or along some border lines. Instead, whole areas may be occluded if perspectives differ strongly. There may also be such occluded areas in the other direction, and it is possible that these areas give random support to each other. The detection of these areas is also problematic in the proposed algorithm, since pixels in the interior of these areas usually get conflicting information from their neighbors. It is not clear, whether their test patches will quickly converge to a single position or whether they will be uncertain (nearly equally distributed) for many iterations. The bidirectional stabilization may introduce this uncertainty into the other direction. Hence, large patches of that direction might still give some support for these pixels and thus their detection may depend on their color or some noise. A more robust detection of these areas is desirable.

3.1.2 Applying the Model to Stereo Images

To avoid the drawbacks of very large patches, the patch positioning is altered now: Until now, patches have always been centered relative to a mean dis-

placement, which depends on the whole image. This displacement and the test patch size are a kind of a priori information needed by the algorithm. Depending on this information is the main problem for the given algorithm, since it has to be adapted to the image pairs. For a general algorithm it would be preferable, if it could be applied to a greater class of images using the same parameters. In the new model the displacement is roughly estimated by the algorithm for every pixel and the patch is positioned according to this guess, which additionally makes the use of much smaller patches possible. Furthermore, the correspondence probability is estimated, too, so that it can be used from the first iteration on, even for pixels, which lie in the interior of large occluded areas.

How can the positioning be estimated? As pointed out before, large disparities result from objects close to the camera and disparity variations within the image result from different distances of objects in the scene to the camera. If the cameras are moved far away from the scene, all objects will approximately have the same distance to the cameras. The distances between the objects become smaller compared to their distances to the cameras. It is assumed that no other objects come into sight and that only the original scene is inspected. Consequently, this scene is projected to a smaller region and uses fewer pixels in the new images A' and B'. If these images are matched using the standard algorithm, smaller patches can be used than for the original images. The results of this correspondence search (and occlusion detection) can then be used to initialize the match algorithm for the original images. Furthermore, occluded areas melt down to single pixels or lines in A' and B' and can be detected more robustly.

Of course it is not possible in general to move the cameras arbitrarily far away from the scene. This imagination just helps to understand the idea used for position estimation. Instead, for the two images A and B Gauss pyramids are constructed, i.e. the image size is repeatedly reduced, and the resulting images are used to initialize test patch position and occlusion information for the correspondence search between A and B.

3.1.3 Gauss Pyramid Construction

When creating a Gauss pyramid for some image, the image width and height are recursively reduced by a factor of two. For convenience reasons, it is assumed that dimensions of images A and B to be matched are always powers of two (though calculation is also possible for odd image sizes) and that these images are squares of a minimum size:

$$Img_x = Img_y = 2^n, \text{ where } n \in \mathbb{N}_{>4}$$

The original images are defined to be pyramid layer zero. Since width and height are reduced when travelling up one pyramid layer, the image size at layer l is:

$$Img_x^l = Img_y^l = 2^{n-l}, \text{ where } l \in \mathbb{N},\ 0 \leq l \leq n-4$$

When centering a test patch to some pixel and the patch does not fit into the image, the pixel is defined to belong to the border area. Note that for meaningful matching results the border areas of an image should be small compared to the whole image size. Therefore the number of layers is limited by the size of the smallest reasonable layer, which is assumed to be sixteen pixels here. For large images it may be a good idea to set this minimum size to 32 or an even larger value to avoid uncertainties at the borders.

In order to reduce the image size and match at a smaller scale, care must be taken of the sampling theorem (see [35]). It states that the sampling rate must be higher than two times the highest image frequency, otherwise the image signal cannot be reconstructed correctly from the sample values. The critical frequency defined by dividing the sampling frequency by two is called the *Nyquist Frequency* f_N (according to [34]). If the signal contains spectral components above the Nyquist Frequency, the sampling will introduce errors into the image, known as aliasing in signal theory or the Moiré effect in image processing. These artificial structures would disturb the matching and have to be avoided.

Consequently, before down-sampling the image it has to be low pass filtered, i.e. all frequencies equal to or greater than half of the new sampling frequency f_s have to be suppressed. On the other hand, lower frequencies must stay untouched to keep as much structure as possible for matching. Intuitively, high frequencies are responsible for sharp edges or corners, while spectral components at low frequencies only form smooth transitions. It is therefore assumed that the most significant structures for accurate matching are created by higher frequencies, while the others can only be used for an approximate positioning. Hence, a steep filter is needed that supresses frequencies higher or equal to f_s (and thus irregular structures introduced by Moiré effects) but keeps all others untouched (to save significant structures).

Unfortunately, the uncertainty relation sets up a minimum product of variances of Fourier pairs in frequency and spatial domain, which is known as the time-bandwith-product in signal theory. That is, infinitely steep filters in the frequency domain have an infinite impulse (respective point) response and therefore filters with finite impulse/point responses (FIR filters) cannot be ideal low pass filters. Clearly, large filter masks will take longer to compute or need more resources than shorter ones.

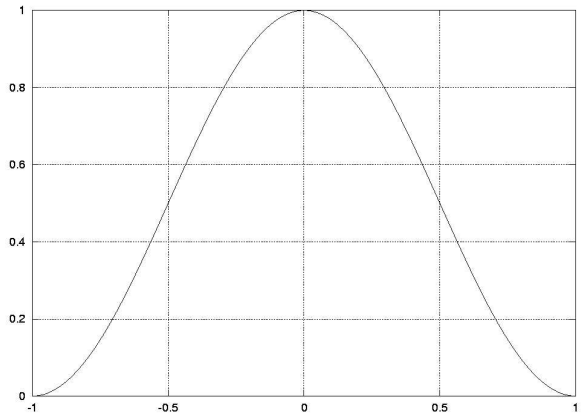


Figure 3.2: Transfer function of $\mathcal{B}^2$ (1d), x-axis scaled to Nyquist frequency

A compromise with respect to the accuracy of the low pass filtering has to be made. The filter has to be as localized as possible in the spatial domain and also as steep as possible in the frequency domain. The Gauss function (which also corresponds to a Gauss function in the other domain) is optimal with respect to the uncertainty relation, since it provides a minimum product of spatial and frequency variance. In [35] Jähne states that transfer functions of binomial filters approximate such Gauss functions and that they can be implemented efficiently. However, the problem in using Gauss functions as low pass filters is that they do not have a border, they are infinitely expanded and smooth absolutely everywhere. Thus there is no critical frequency and the only characteristic points are the turning points, which are defined by the standard deviation of the Gauss.

$$\mathcal{B}^2 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Two-dimensional second order binomial filter $\mathcal{B}^2$

For a second order binomial filter $\mathcal{B}^2$ this turning point is located exactly at frequency $0.5 f_N$, which suits our needs quite well. This spectral component is reduced to 50%, while components of higher frequencies are reduced much more, since the transfer function decreases strongly from that point on. Note that components near the Nyquist frequency are suppressed almost completely and that the filter has the property that smaller structures (with

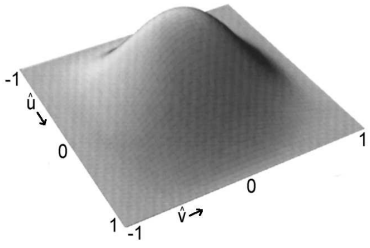


Figure 3.3: Transfer function of $\mathcal{B}^2$, $\hat{u} = 1$ and $\hat{v} = 1$ are Nyquist frequencies

Figure 3.4: Images with recursively reduced size (achieved by low-pass filtering using $\mathcal{B}^2$ and subsequent down-sampling by a factor of two for both directions)

higher frequencies) are weakened much more than larger ones as pointed out in [35]. For a pure low pass filter, this may still be too much of the high frequencies, but remember that frequencies only a little below $0.5 f_N$ are expected to be very important for matching. Higher order binomial filters with filter masks of size 5 or 7 would suppress the disturbing higher frequency components even more but would also heavily reduce structural information important for matching. Using $\mathcal{B}^2$, there should be almost no noise in the images (apart from the introduced aliasing), since noise usually affects high frequencies. Small similarity disturbances due to Moiré effects may therefore be acceptable. After all, matching over scale is just used to produce an initial guess of the final matching process, which will be applied to the unchanged original images. It is assumed that errors at the higher pyramid layers will not be so bad that the patches are positioned completely wrong and small errors can be compensated.

After low pass filtering the image using $\mathcal{B}^2$, it can be down-sampled to get the first pyramid layer (l=1). This image can then be treated the same

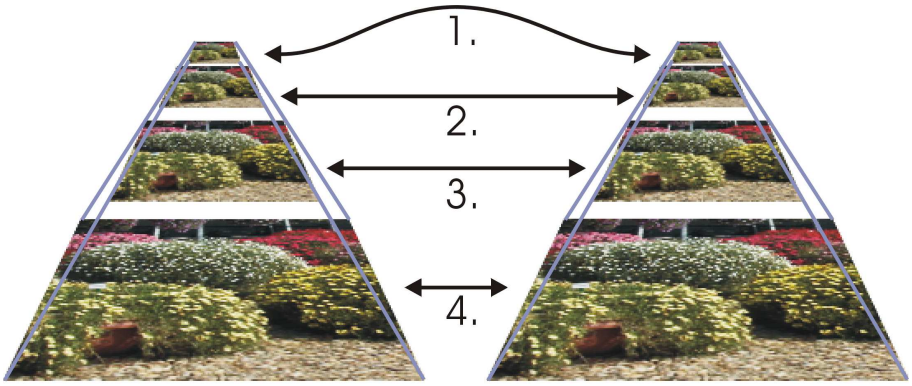


Figure 3.5: Gauss pyramids for image A (left) and B (right): First the smallest layers are matched, while layer zero (containing the original images) is matched at the last step.

way to generate the next one. As pointed out before, the matching can only be applied to images of a certain minimum size. Hence, there is no need to calculate smaller scales.

3.1.4 Scale Matching

After the pyramid is calculated up to a minimum size for each image, the corresponding images of the pyramids are matched, beginning with the smallest layer. The resulting information is used to initialize the matching of the next larger images and so on. Finally, the images of the original size are matched using the information calculated by the second largest match.

First the smallest images of both perspectives are matched. The test patch size may be chosen very small, since a test patch width of n at layer m corresponds to an exponentially increased width of size $n2^m$ at the original layer zero. However, it has to be chosen in a way that all possible disparities are covered. A displacement of zero ($d = (0; 0)$) is used and the occlusion information is initialized constantly as stated in the previous chapter.

The resulting match information is sparse, since it is only calculated for the pixels of the smallest images (a fourth of the number of the next larger image's pixels). To proceed to the next layer, it has to be interpolated for the additional pixels in the next larger images. The test patches for the next match are then centered to the (interpolated) expectation values and the occlusion information is also initialized using the interpolated data of the smaller scale. The test patch size used at the smallest layer applies to

all scales, so that the area where the match may lie is relatively constrained further and further, when travelling down the pyramid. Now, the matching is executed for this layer and the results are used to initialize the next one. This process continues until the original images are matched.

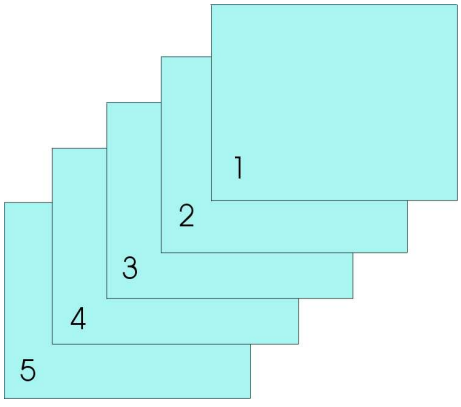
This should not only solve the problem of appropriate test patch size but can also help in matching regions with weak structure, where the former algorithm would have needed many iterations for test patch convergence. Note that occluded areas in the original images will shrink to pixels at small scales, which are detectable more robustly.

Suppose the match information is always interpolated for all pixels of the original image. The resulting process regarding this information can then be viewed as a coarse-to-fine matching approach, starting with rough information of low spectral components and taking into account more and more higher frequencies, when travelling down the pyramid. At the original scale, the match is influenced by the highest spectral components, which are responsible for the final accuracy but which are also most susceptible to noise.

3.2 Optic Flow

3.2.1 Image Sequences

Optic flow means displacement of corresponding pixels from images of a time series taken by a single camera. That is, the camera may change its position, objects in the scene may move and the environment may change (e.g. the illumination). However, usually images are taken very frequently compared to changes in the scene, i.e. there are only small changes between two subsequent images of a time series.



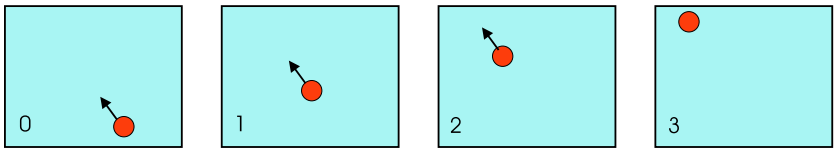


Figure 3.6: Constant velocity assumption for image sequences

In contrast to the observations made for stereo images, displacements of corresponding pixels will usually be very small. The main reason for this is that the camera is expected to be quite far away from the objects in the scene compared to the different positions of the camera (or the objects) at two subsequent frames. Hence, the movements do not cause perspective distortions between the images as heavily as in stereo imaging. However, to some degree these distortions will still appear and occlusion is also possible.

Additionally, an assumption about the movements in the scene is imposed, which should be valid for most natural scenes with rigid objects. The velocity of a 3d-object will only change smoothly over time (due to inertia) and so will its projection into the camera plane. Hence, all points projected by this object will have smooth velocity over time.

3.2.2 Similarity for Optic Flow

If the projection of objects into the camera plane changes only smoothly over time and images are taken very frequently compared to changes in the scene, the movement of a pixel can be assumed to be constant for a few images. That is, if a point in space is projected to pixel x_1 in image 1 and $x_1 + v$ in image 2, it is assumed to be projected to $x_1 + 2v$ in image 3 and $x_1 + 3v$ in image 4. This does not necessarily mean that velocity is constant for a certain pixel *position*, since one position can belong to various objects in different images. Instead, it is assumed to be adherent to the object (and thus to the pixel), which will possibly change its position from image to image. For some specific applications it may be more promising to assume velocity being adherent to the position (as for static scenes with many rotating objects), or to make completely different assumptions. However, in this model this is not assumed and objects are expected to move slowly through space and behave as pointed out before.

The constant velocity assumption is now implemented into the pixel similarity function: For subsequent images I_1 and I_2 of a time series, pixels are expected to have constant velocity. The positions of the corresponding pixels in previous and future images can then be extrapolated linearly, though this

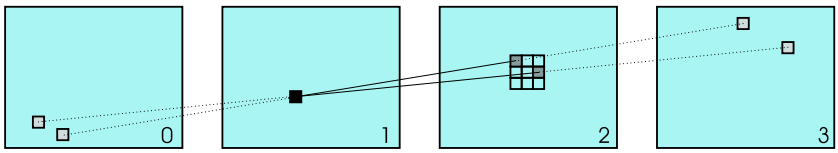


Figure 3.7: Constant velocity assumption for pixels

is a very rough approximation, since there is no subpixel accuracy. On the other hand, this assumption can also be utilized to find the optimal match candidate.

Let x_1 be a pixel position in image I_1, which is the second frame of the image sequence (I_0,I_1,I_2,I_3). If pixel x_1's correspondence in image I_2 is searched, by now the patch probabilities are initialized with the similarities of these two pixels only. That is, for match candidate x_2 in I_2, the s-function is used to evaluate the similarity of pixels x_1 and x_2. Now, additionally the values of the extrapolated positions in images I_0 and I_3 are used. If x_1 and x_2 actually are projections of the same point in space, it is assumed that this point will project to the extrapolated positions x_0 and x_3 in images I_0 and I_3. Hence, $x_3 := x_2 + (x_2 - x_1)$ in I_3 is inspected as well as position $x_0 := x_1 - (x_2 - x_1)$ in I_0. Note that not all possible combinations of similarities are evaluated but only those of pixels of subsequent frames, that is x_0 to x_1, x_1 to x_2 (as before) and x_2 to x_3.

Since the constant velocity assumption is very rough and provides no subpixel accuracy, the extrapolated pixel positions are not compared as strictly as the actual pixels. The correspondence x_0 in I_0 and x_1 in I_1 is assumed to be noisier than the actual correspondence x_1 in I_1 and x_2 in I_2, thus the Gauss used for similarity is not that strongly peaked but has a greater variance. The same applies to the similarity of x_2 in I_2 and x_3 in I_3. Nevertheless, for a good match, it is expected that all four pixels have about the same value.

The similarity function used for flow is defined based on the normal similarity function $s()$ and a function $\hat{s}()$, that is identical to $s()$ except for the fact that $\sigma_{\hat{s}} := 2\sigma_s$. This demands a high similarity for the two images under inspection and allows greater differences for the extrapolated pixel positions.

$$s_{Flow}(I_2|_{x_2}, I_3|_{x_3}) = \hat{s}(I_1|_{x_1}, I_2|_{x_2}) s(I_2|_{x_2}, I_3|_{x_3}) \hat{s}(I_3|_{x_3}, I_4|_{x_4})$$

Apparently, not only pixel colors of two images are compared but also the extrapolated position of that pixel in the previous image and in the subsequent image are expected to have a similar color (or some invariant property).

3.2.3 Positioning of Test Patches

To avoid large patches and to improve occlusion detection, the test patch positioning in optic flow computation is done using a scale approach, too. However, since only small displacements are expected, it should be sufficient to use only the next higher scale of a pyramid to initialize the patch positioning. For larger homogeneous areas, it may be better to use the full pyramid, since these areas shrink to pixel size at some smaller level. Nevertheless, an explicit homogenity measure (for local scale) in the similarity function seems to be more promising to solve this problem. It is therefore put aside and only one higher pyramid layer is used for test patch positioning and correspondence probability estimation.

3.2.4 Correspondence Probability Initialization

Though the correspondence probability is initialized with $1/|\mathcal{I}|$ at the upper most pyramid layers as pointed out before, some other start values may be reasonable for optic flow applications and also for stereo scenes. A good initialization is desirable, since the algorithm uses correspondence probability for matching from the first iteration on and an almost correct value can improve the matching in early phases of the algorithm.

A first idea is to expect occlusions at edges, since depth discontinuities are located at object borders which can often be found at edges in the image data. It has been tested to apply an edge detection operator to the image and to declare regions containing edges as being more likely for occlusion. Unfortunately these regions are also the regions containing structure and thus those from which the information propagates to the more homogeneous image parts. If this diffusion is disturbed due to reduced influence on the neighbors, convergence becomes very slow. Results become bad because pixels are matched due to noise and therefore this initialization is not used. The idea of exploiting the similarity of the most likely pixel in the patch for correspondence probability estimation does not work properly either. Consequently, the initialization value $1/|\mathcal{I}|$ is used as proposed in the model.

Good values are usually available after some iterations, i.e. occluded pixels can be roughly distinguished from the others. Initializing the correspondence probability with the constant expectation value and executing some iterations of the algorithm should yield very exact values. The test patches may then be reset, while the correspondence probabilities are kept and used as a start value for the second algorithm execution. This (quite exact) method of initialization is called correspondence probability pre-computation. However, it is only used for comparison purposes, since it requires additional iterations.

Chapter 4

Implementation

4.1 Complexity Analysis

In this section the complexity of the general algorithm is briefly examined. In computer vision, applications are usually time-critical, for instance if they have to act in a natural environment and must achieve real-time capabilities. Of course, memory and other resource requirements of algorithms have to be satisfied, too, but they can usually be assumed to be available and may be neglected here at first. Thus, the focus here is set on time complexity.

Traditionally, time complexity of an algorithm is measured by counting all operations regarding some important variables (e.g. image size, iterations and so on). If the complexity relation regarding these variables is a polynomial, only the highest order term is used and constant factors are left away. Using such a rough complexity measure, one speaks of the order $\mathcal{O}()$ of some algorithm. Two implementations are compared now: The software implementation with standard resource requirements and the specialized hardware implementation with maximum parallelity.

The software implementation of the algorithm on some standard CPU has to serialize all computations, such that all operations have to be run step by step. The algorithm is started with the initialization of the test patches with pixel similarity. For each pixel, the similarities have to be calculated for all candidates in the test patches. This takes time of the order $\mathcal{O}(Img_x \cdot Img_y \cdot T_W \cdot T_H)$, where T_W and T_H are width and height of the test patches.

The next step is the neighborhood step of equation (2.8), which is responsible for the diffusion process: For each pixel in each direction at each iteration, every probability of its test patch has to be calculated. Each probability again is a weighted sum over all eight neighbors' values. In order to

get such a neighbor's value, all pixels of its test patch are weighted by the ordering pdf (using the h-function as described in equation (2.5)) and the maximum is seached. The weighting depends on the original pixel and thus cannot be reused for other patches. Note that there may be an optimization, if, for instance, a constant patch positioning is used, but this is not assumed here.

Hence, complexity depends quadratically on test patch height and width and linearly on image height and width: $\mathcal{O}(Img_x \cdot Img_y \cdot T_W^2 \cdot T_H^2)$. The Gauss functions may be precalculated and accessed via lookup tables, since they are only applied to some small discrete set of values.

Regarding the whole algorithm, the most expensive (and thus the relevant) operation here is the diffusion process while the initialization may be neglected. Support collection for occlusion detection and bidirectional merging can also be ignored, since their complexity is similar to the initialization complexity. Nevertheless, the diffusion complexity is multiplied by the number of iterations it.

$$C_{Software} \approx \mathcal{O}(it \cdot Img_x \cdot Img_y \cdot T_W^2 \cdot T_H^2)$$

However, the diffusion process is so computationally expensive, that a software implementation of the algorithm is far away from real-time and can only be used for qualitative analysis.

Nevertheless, the proposed algorithm is highly suitable for a hardware implementation where independent operations may be executed in parallel to optimize execution time. Certainly, this requires very special hardware, e.g. FPGA (field programmable gate array) chips, where such parallel implementations are possible (see [32]). If sufficient resources are assumed to be available and the algorithm is implemented 'as parallel as possible', only those operations take effect on the time complexity, that depend on some previous result and thus cannot be executed simultaneously with the previous operation.

It is clear that no calculations depend on other results in the initialization, thus it takes only constant time if every candidate in every patch is initialized in parallel. Constant time means, that the execution time does not depend on any of the parameters but on a fixed number of operations. At each iteration, the steps neighborhood, correspondence and direction merging have to be executed sequentially, because they depend on the results of the previous step. However, within one step, all pixels and their test patches may be computed simultaneously. From the software implementation it is known that by far most of the operations are used for the neighborhood step, which can be considered at the heart of the algorithm. Since it is fundamentally parallel,

all operations can be executed simultaneously in a hardware implementation which yields a constant time complexity. Note that the constant factor may be quite high, since for every pixel multiplications and additions have to be executed.

Formally, a hardware implementation will thus need time in the order of:

$$C_{Hardware} \approx \mathcal{O}(it \cdot (\log(T_W \cdot T_H) + \log(Img_x \cdot Img_y))$$

This is not due to the neighborhood function, but because the normalizations and the support collection for occlusion detection always sum up across the test patch and the correspondence probability normalization sums up across the whole image. Using a common implementation with cascaded adders, the addition of n numbers needs $\log_2(n)$ layers, which are the relevant steps here. Note that in a very specialized low-level implementation even these additions may theoretically be done in one step, since the addition of n numbers with m bit each can be implemented by a very large lookup table (2^{nm} entries with multiple bits each) representing a binary function with mn input values. The complexity would then depend only on the iterations, but arguing this way, one may also state that the whole matching could be pre-computed for every image pair in one large lookup table. Clearly, such hardware and that much pre-computation time is not and will not be available, and therefore the complexity is estimated only for reasonable resource assumptions.

Even for the first (more realistic) hardware implemetation suggestion, all pixels and all patches have to be calculated in parallel. In particular such a hardware must be capable of accepting images of the given size and provide test patch resources for all those pixels. These requirements are beyond the capabilities of FPGA boards of today, but may be available not too far in the future.

4.2 Hardware Implementation

4.2.1 Recurrent Neural Network Model

The model can also be interpreted as a recurrent neural network, which inspires a parallel implementation, too. Remember that there is a test patch for each pixel, which is set to a size of 3x3 in this example. One test patch is shown by the square at the upper left corner of figure (4.1). Each of the nine match candidates in the patch is represented by a circle. These circles are now called neurons, they represent simple computation units with a large number of interconnection lines. For each neuron, the pixel the patch belongs to is printed in the small square in the neuron's right part. Note that neurons

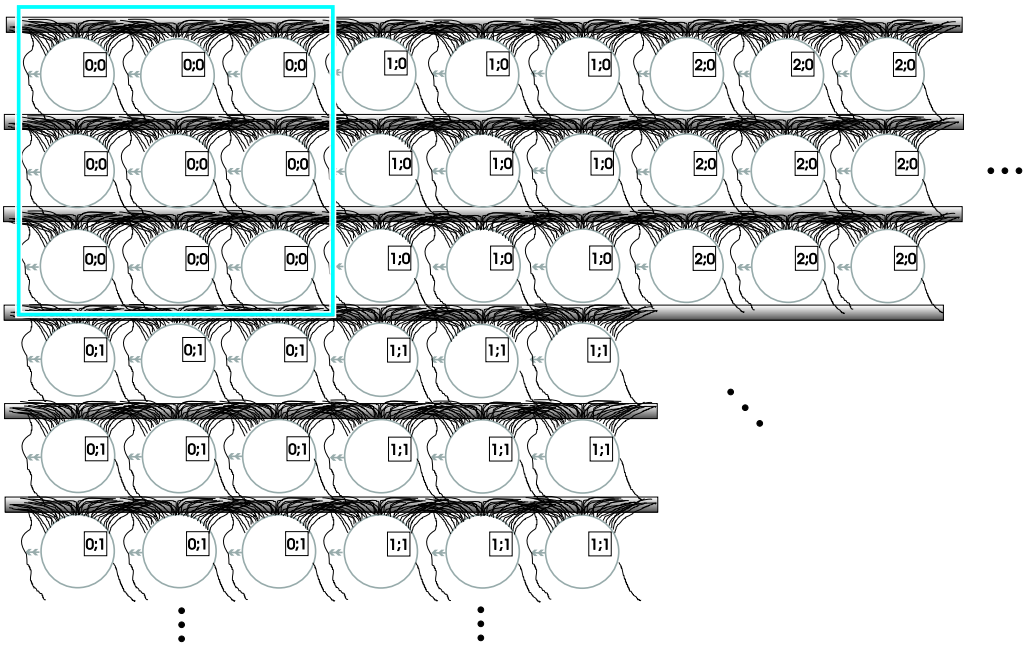


Figure 4.1: Model interpretation as a recurrent neural network : The blue square represents a 3x3 test patch.

of the same test patch do therefore always contain the same number at this position (compare blue square in figure (4.1)).

Each neuron gets its input signals from weighted outputs of other neurons. The input lines are always shown on top of the neuron with one exception: The connection to the neuron needed for bidirectional merging is placed at the right of each neuron to demonstrate its special meaning. The output signal of each neuron is located at the left side. These outputs are used again as inputs for some other neurons so that loops are constructed, which leads to the name recurrent neural network. Which neurons exactly are connected to which others is displayed in the next figure.

Note that the neurons are only indicated for the A to B direction, but there is also the same number of patches and match candidates for the inverse direction. In this model, there is massive interaction between the neurons, which is indicated by the dense structure of the connections. To understand how the neurons are connected and how the model fits into the algorithm of the former chapters, one neuron is shown in figure (4.2) in more detail.

Remember the main parts of the pixel-match pdf (equation (2.12)) of chapter two. For a pixel x_A, the match candidate x_B is inspected. This is indicated by the small x_A in the right part of the neuron and the light gray

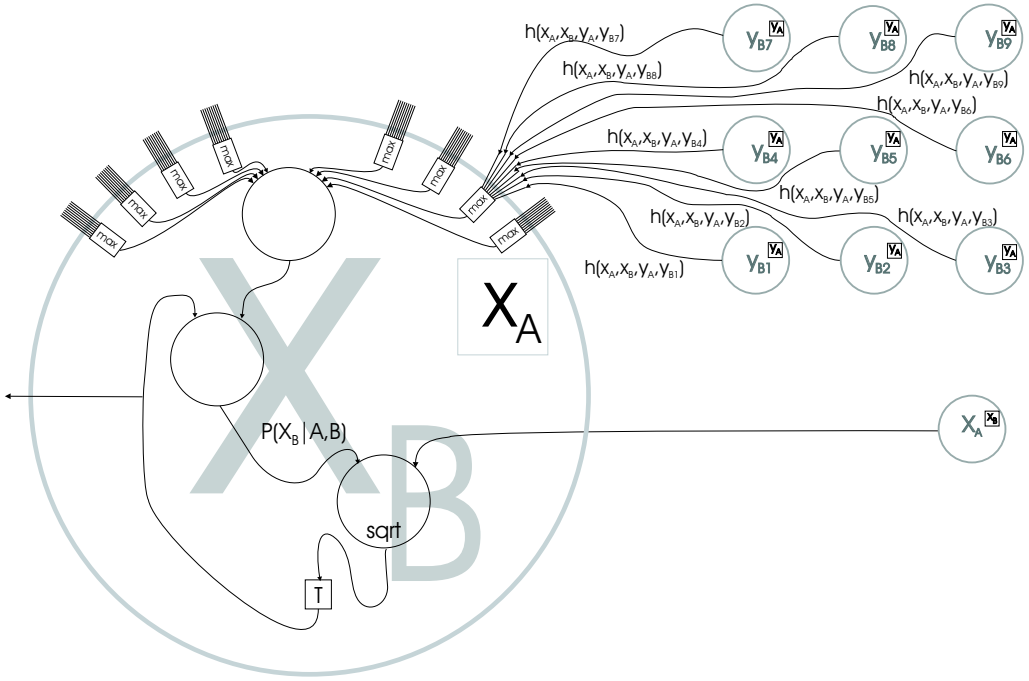


Figure 4.2: Connections and internal neuron structure

x_B in the background. First, the similarity of x_A in A and x_B in B has to be evaluated as an initialization step, which is not displayed here. However, it can be imagined that the square with the T initially contains this similarity value. Now for all eight neighbors of x_A it is checked, how well they support a correspondence (x_A, x_B), i.e. whether for some neighbor y_A there is a likely y_B with a similar displacement. Mathematically this is done by multiplying the similarity (from the T) by a sum of probabilities over all eight neighbors, which is indicated by the summation sign in the upper part of the neuron.

Now the above mentioned sum is inspected, in particular the single addends. These are the values of the best matching y_B for a given x_A,x_B and y_A. To find this best matching candidate, the similarities of the whole test patch of y_A are weighted by the h-function and the maximum value is taken. Note that these weights are fixed when the test patch is positioned. Clearly, this maximum search is not only applied to one neighbor y_A of x_A but to all. However, these connections and neurons look the same and are not displayed here to avoid too much complication.

The resulting product of sum of neighbor values and similarity is the direction dependent patch probability, as defined in the pixel-match pdf of

equation (2.19). The next step is the bidirectional merging of equation (2.21). The probability has to be made direction independant for this and is therefore weighted by the correspondence probability. Note that this probability has to be updated dynamically (as defined in equation (2.20), which is not displayed here. Additionally, the normalizations have been left away, too, for the sake of readability. The direction independant probability is then multiplied by the one calculated for the opposite direction. The input signal to the right is used to symbolize this probability.

Both probabilities are then multiplied and the square root is computed, as proposed in equation (2.21). Note that instead of geometric averaging an arithmetic averaging is also possible, since it can be derived the same way. This would avoid the computation of the square root and replace it by a weighted sum, which is more suitable for a neural computation, since it can be done by very simple operations. However, deriving an optimal neural network with specialized operations is beyond the scope of this thesis.[1] The goal here is to show that the interpretation as such a network is possible in general.

Finally, the averaged value is used for the next iteration. Note that the idea of iterations requires some kind of clock to distinguish the different time intervals. To cope with the recurrence, the model is assumed to work in synchronous mode here. That is, there are time ticks and only at such a clock tick the T-box checks the input signal, saves the input and updates its output one moment later. The next clock tick is not given before the system has stabilized. Hence, within one iteration, the network has no loops.

4.2.2 FPGAs

As pointed out in the complexity analysis, the algorithm should be implemented in hardware to be fast enough and as much computations as possible have to be executed in parallel. Such an implementation requires a very special hardware, e.g. FPGA chips.

These FPGAs can be regarded as large programmable hardware chips, which provide a huge number of basic circuits, which may be connected to each other and which can compute simple functions (see also [32]). Hence, they offer a way for a parallel implementation of the algorithm. Ideally, a suitable FPGA would provide resources for every match candidate in every test patch. The neural network of the previous section may then be a good way to implement the algorithm.

[1] For more information about neural networks and further references please refer to [29]. Relations to the human visual system and neurobiology can be found in [30].

For the time being, it has to be stated that the algorithm cannot be implemented fully in parallel and thus its capabilities cannot be used efficiently. Rabsch (see [22]) has implemented the former model using some FPGA chip and shows how an implementation is possible in principle, though he has to make some approximations. Missing resources can be traded against time, but real-time capability can only be achieved, when it is running (almost) fully parallel. However, since hardware resources and clock speeds are still increasing very quickly, a completely parallel implementation may become possible in the future.

4.3 Software Implementation

For qualitative analysis purposes, the algorithm is implemented in software for a standard CPU (e.g. an x86-compatible or similar sequential processor). The actual implementation is done using the Perception Action Components Library[2] (PACLib), which among other things supplies basic functionality such as image display and camera access. Since a software implementation will always be far away from real-time capabilities, no approach has been made to get a time-optimal solution. Instead, a more adaptable experimental model has been used. This has to be kept in mind (and the proposed hardware implementation) when execution times are compared to other matching models.

4.3.1 Parameters

For all experiments generally the same parameters have been used as in [1]. That is, the standard deviations of the h- and s-function are set to:

$$\sigma_h = 1$$

$$\sigma_s = 0.16$$

Matching is always executed bidirectionally, test patches have the same size for all pixels and at each scale. That is, on a high pyramid layer, such a patch covers a much greater area (with respect to the original image) than on the lowest layer. Usually 20 iterations have been used and unless stated otherwise the patches have converged after this number of iterations.

The default test patch size chosen is 7x5, which showed good preliminary matching results for various scenes. Nevertheless, for rectified stereo images

[2]For detailed information about this software library and related programs please visit the PACLib homepage at http://www.ks.informatik.uni-kiel.de.

the test patch height can be set to unity, in order to make use of the additional information. In stereo scale matching a width of seven pixels covers all reasonable horizontal displacements, since it is also applied to the smallest pyramid images, where small displacements correspond to large ones at the original size. For optic flow it is also large enough, if the assumptions of small movements hold. Choosing the width larger than the height expresses the assumption that the expected maximum horizontal displacement is larger than the maximum vertical one. Note that for certain specialized geometries other test patch sizes may be more promising.

4.3.2 Patch Positioning

Unlike in the former model, there is no global expected mean displacement, which takes effect on the test patch positioning as it is pointed out in the stereo and optic flow adaptions. Test patches may be placed individually, i.e. patches for neighboring pixels do not have to overlap at any rate. However, in general this will still be the case, since there is no a priori information where to put the patches. If the scale approach is not used and nothing else is stated, a constant displacement of $d = (0; 0)$ is used.

4.4 Error Metrics

The following error metrics are used, which have been chosen in accordance to [9] and which make it possible to relate the results to the work of others: The local error vector $\mathbf{e}_L(x)$ at the position x is defined to be the difference between the ground truth vector and the calculated displacement vector. The *local error* $e_L(x)$ is the length of the local error vector at position x according to L_2-Norm. The *mean error* e_m for an image is then defined to be the sum of all local errors at positions of so called good pixels, divided by their number.

$$e_m = \frac{1}{|\mathcal{G}|} \sum_{x \in \mathcal{G}} e_L(x)$$

The set $\mathcal{G}$ of good pixels contains only those pixels at least half a test patch size away from the border who have a correspondence in the other image. Therefore a low mean error means that all relevant pixels in the image have been matched well. Since many small errors are more acceptable than a few big ones, it is desirable to have a metric on the error distribution. The standard deviation σ_e can give a hint about that.

$$\sigma_e = \sqrt{\frac{1}{|\mathcal{G}|} \sum_{x \in \mathcal{G}} (e_m - e_L(x))^2}$$

The aperture problem is a very crucial problem in matching. To evaluate how well an algorithm can cope with the aperture problem, a local and a *mean aperture error* is defined according to [9]. The first image to be matched is converted to gray levels by arithmetic averaging of RGB channels, after that the brightness gradient is calculated at each pixel position: A perpendicular vector $\hat{\mathbf{p}}(x)$ of length unity is then defined and the local error vector is projected onto this new vector.

$$e_{la} := \mathbf{e}_L^T(x) \cdot \hat{\mathbf{p}}(x)$$

The result is called the local aperture error and represents the amount of error that points into the most homogenous (and uncertain) direction. Again, averaging the absolute values of local aperture errors across all good pixels yields the mean aperture error.

$$e_{ma} := \frac{1}{|\mathcal{G}|} \sum_{x \in \mathcal{G}} e_{la}(x)$$

If any of the above symbols refers to a whole sequence of images and is therefore averaged across these images, this is indicated by a bar on top of the error (e.g. $\bar{e}$).

Chapter 5

Experiments

5.1 Applying the Model

To check how the model works in practice, some artificial stereo images are evaluated as a first proof of concept. An RGB image of size 32x32 pixels containing random noise (equally distributed over the pixel range, pixelwise independant) is used as background (symbolized yellow in figure (5.1)) and an image of size 16x8 pixels (symbolized blue) also containing such noise is used as a foreground object. To generate image A, the small image is inserted into the large one at some position near the center, simulating a rectangular object, which hides parts of the background. In image B, the foreground object is moved by two pixels to the right and one up, so that it hides a slightly different area of the background.

The true displacement can be seen in figure (5.3). The brightness represents the absolute value of displacement, i.e. black pixels have the same

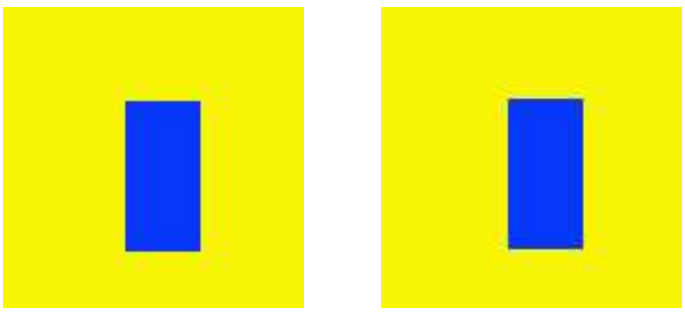


Figure 5.1: Generation of artificial images: Blue area (foreground object) starting at (13;10) hiding parts of yellow area (background object) is moved from image A (left) to B (right) by two pixels left and one pixel up.

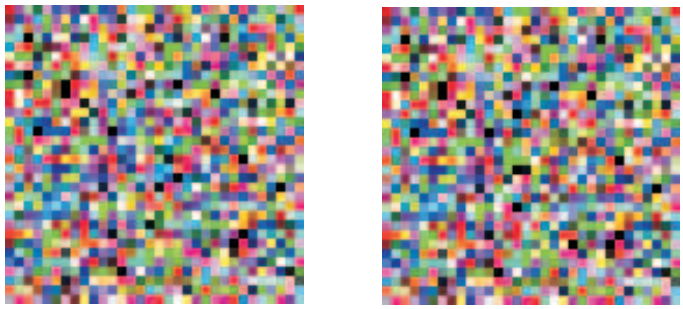


Figure 5.2: Generated stereo images

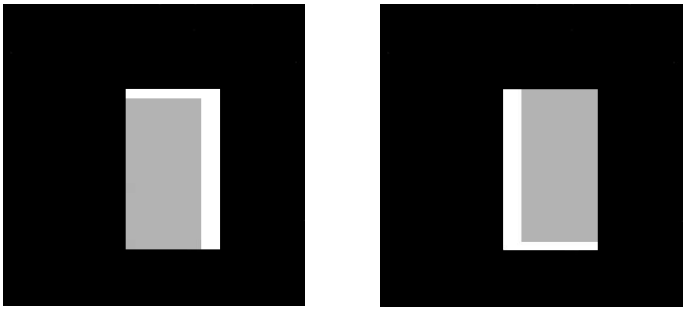


Figure 5.3: Ground truth images: Displacement for A to B direction (left) and B to A direction (right). Pure white pixels are (half-)occluded.

position in both images, while gray pixels are displaced. In this artificial experiment all pixels of the foreground object have a constant displacement (of a whole number of pixels), while those of the background have no displacement. The occlusion information is integrated here into the displacement images in a way that pure white areas indicate half-occluded pixels, which have no correspondence in the other image. Since the image generation process is known, one may say that they belong to the background and that their disparity would be zero, if they were not occluded in the other image. However, having only the two images and no further information, the only possible statement here is, that they have no correspondence. Thus they are not evaluated in error analyis. For evaluation purposes the A to B direction displacement images are shown, but all observations are also valid for the B to A direction. Using the old model (see figure (5.4)) most pixels are matched correctly (bidirectional mode, test patch size 5x3, 20 iterations), but there are some problems at the borders of the foreground object. Background pixels left to and below of the foreground object are matched badly, although they have well-defined correspondences in the other image. As pointed out, occluded pixels (right of and above the foreground object) have no corre-

spondences, so their match values are neglected here. The displacement field has been smoothed at least at the left and lower object boundary: Pixels of the background have been influenced strongly by foreground object pixels and are matched as if they were moving with this object. This occurs due to the neighborhood constraint, that demands similar displacements for neighboring pixels. Since there is a very high degree of information and pixel color differs noticably for wrong match candidates, pixels should be matchable quite uniquely. Nevertheless, there is always a tradeoff between similarity and smoothness. Neighboring pixels belonging to a different object are trusted exactly as much as really neighboring pixels are, whose origin in 3d-space is also neighboring to the origin of the point under inspection. The absolute value of matching error is high in these border regions, as can be seen in the error image in figure (5.4).

Now the improvements of the new algorithm are tested. It is run without stereo or optic flow constraints (also bidirectional mode, test patch size 5x3, 20 iterations) on the same images. At first the occlusion detection capabilities are inspected. The images in figure (5.5) show the probabilities that some pixel in A has a correspondence in B (left image) and vice versa (right image). All occluded pixels have been detected, i.e. their match probabilities are significantly lower than those of pixels not occluded. Some pixels have slightly less support than others, especially at the borders, but qualitatively all occlusions are detected and there are no false positives.

The implications on the matching result can be seen in figure (5.6). There are sharp displacement field discontinuities at the object's lower and left borders. The other borders are not relevant since the pixels right and on top of the foreground object in A are occluded in B. These results apply to the inverse direction analogously. At first sight it is somewhat surprising that the matching results have improved for the A to B direction in regions where there are actually no occluded pixels, whose influence could have been reduced. This is achieved by bidirectional merging. Occlusions from the other direction discouple the flow field and optimize the matching accuracy of this direction, too.

Adding some noise to image A and image B independently does not disturb the matching results and occlusion detection as can be seen in table (5.1). The number of pixels to match is in the size of 1000 and pixels not lying at boundaries have been matched absolutely correct. Thus viewing the mean matching error in thousandths can give approximately the number of pixels being matched really wrong.

Note that the images used have two artificial properties, which are not always valid for natural images: First, there is very strong structure present, which simplifies the matching and the occlusion detection. Usually natural

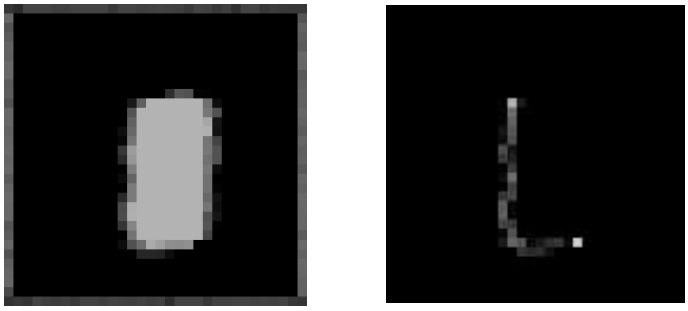


Figure 5.4: A to B results for old model: Disparity (left), error (right). Pixels above and to the right of the foreground object are occluded and therefore disregarded in the error analysis.

Figure 5.5: Correspondence probability of new model: A to B (left) and B to A (right)

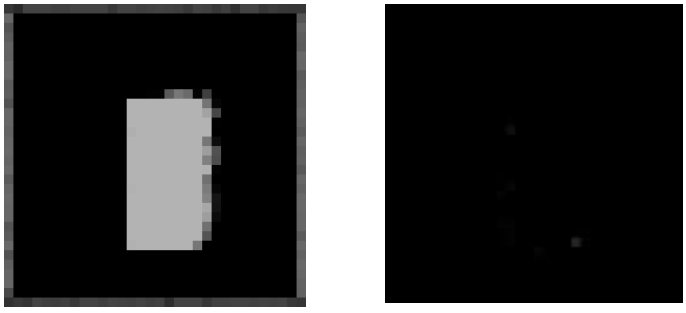


Figure 5.6: A to B results for new model: Disparity (left), error (right). Pixels above and to the right of the foreground rectangle are occluded and therefore disregarded in the error analysis.

Noise Level	Old e_M / $\frac{1}{1000}$	New e_M / $\frac{1}{1000}$
0 %	34.1	1.6
3 %	34.3	1.7
5 %	34.3	1.7
10 %	36.3	1.6

Table 5.1: Mean matching error in presence of Gaussian noise (in percent of the dynamic range of the pixel values) added independently to both images

images are smoother and contain regions with low contrast. Secondly, there are no subpixel correspondences, i.e. no probabilities are distributed across some neighboring pixels and there is a one-to-one correspondence. However, in that scenario, occlusion detection works and improves matching results at image borders, even in presence of noise.

The next example (see figure (5.7)) is more realistic, but still a semi-artificial scene, i.e. it contains natural and artificial elements. This experiment is made with various test patch sizes to check the dependence of the occlusion detection on this size and the implications for matching. Image A shows a small area of a cup with flowers printed on. The smallest flower (left of the image center) is artificially inserted, i.e. its pixels hide a region of the cup, which is not regtangular in this experiment. The flower is then moved to the right to construct image B. Note that this experiment has full ground

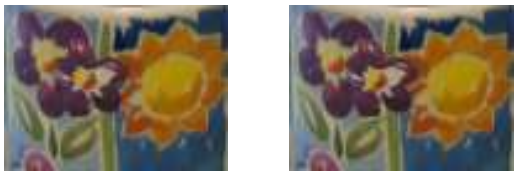


Figure 5.7: Artificially altered scene: Small flower from image A (left) moved to the right to create image B (right)

truth, too. Every pixel is displaced by a whole number of pixels, so that there are no subpixel correspondences. Again, some pixels of the background are half-occluded in the images.

The occlusion detection still extracts the occluded pixels, but with different accuracy, which depends on the test patch size, as shown in figure (5.8). The correspondence probability evolves quite slowly if the test patches are large, because there is more spurious support in large patches. Consequently, many iterations are needed to detect occluded pixels. Since information also

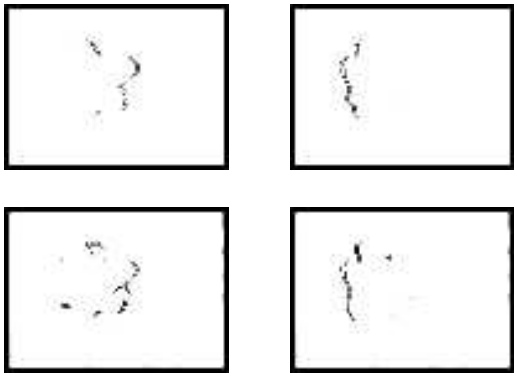


Figure 5.8: Correspondence probabilities for test patch size 5x1 (upper images) and size 7x5 (lower images): A to B direction (left), B to A direction (right)

Figure 5.9: Error distribution using old model (left image), error enhancement in flower region using new model (right image)

propagates before the occluded pixels are detected properly, matching results for large patches are the same in the new and in the old model. However, using $d = (1; 0)$ allows a patch size of 3x1, in which occluded pixels can be detected very well. The mean error (after 20 iterations) shrinks from $26,5 \cdot 10^{-3}$ in the old model to $22,4 \cdot 10^{-3}$ in the new model. Again, it is remarkable that the errors occur only at the border of the small flower, where a smoothing of the displacement field can be observed in the old model, that is present all around the flower. These errors are reduced in the new model. The difference of these errors in the region of the flower's border between the old and the new model are displayed in figure (5.9). However, note that there are still errors present using the new model, since completely correct matching based only on two images is usually not possible.

To understand this, imagine a fried egg centered on a white plate as some image A and the same egg moved by some small distance to the right (but still on the plate) as image B. If the white parts of the egg have the same color as the plate and one does not know the contour of the egg, matching

these images is quite difficult. One can only state that the egg yolk has moved to the right, but since there is no information, which white pixels belong to the egg and which belong to the plate, it is completely uncertain, how these pixels have behaved. Maybe for the pixels at the plate border it is more likely to belong to the plate and thus to have no displacement, while for those near the center it is more likely to belong to the egg and thus to have the same displacement as the yellow egg pixels. However, these ambiguities cannot be resolved without exact knowledge about the egg's contour and in no way by using a general dense pixel matching algorithm based on pixel similarity. The problem can only be solved with semantic information, e.g. the notion that there is an egg on the plate, how eggs usually look like and how big they are. Note that in most natural scenes such situations occur and that this has to be kept in mind when evaluating a matching algorithm.

The next image pair under inspection is an aerial stereo pair (see figure (5.10)) of the Pentagon provided by CMU/VASC[1]. The algorithm has been applied using d=(-6;0), a test patch size of 13x1. The disparity and conrrespondence probability images have been taken after 12 iterations, where the algorithm had converged. As pointed out before, white pixels (in the middle images of figure (5.10)) mean great correspondence probability while black pixels mean low support and thus indicate occluded pixels or pixels which violate the neighborhood or similarity constraints. The correspondence probability images are also referred to as confidence images from now on, since the brightness of these images reflects the over-all likelihood for a correspondence and in some way also the quality of the matching result.

In the images of figure (5.10) the Pentagon may be regarded as a foreground object, which is nearer to the camera than the areas around it. Note that in the right image, half-occluded pixels are usually those of the ground, smaller objects and walls. These pixels can be seen right of the foreground object in the right image and vice versa in the left image. That is, the right correspondence probability image is black (low correspondence probability) for pixels at right borders of foreground objects, while the left image shows half-occluded pixels at left foreground object borders.

Note that the main occlusion lines are detected, but that these lines are not absolutely accurate and sharp. Hence, the matching results are also not sharpened as much as it may have been expected. For completeness reasons the matching results are displayed in the disparity image of figure (5.10), although this example is intended to demonstrate the occlusion detection

[1]The stereo image pair can be obtained from the Vision and Autonomous Systems Center's Image Database of Carnegie Mellon University, Pittsburgh, USA at http://vasc.ri.cmu.edu/idb/html/stereo/pentagon

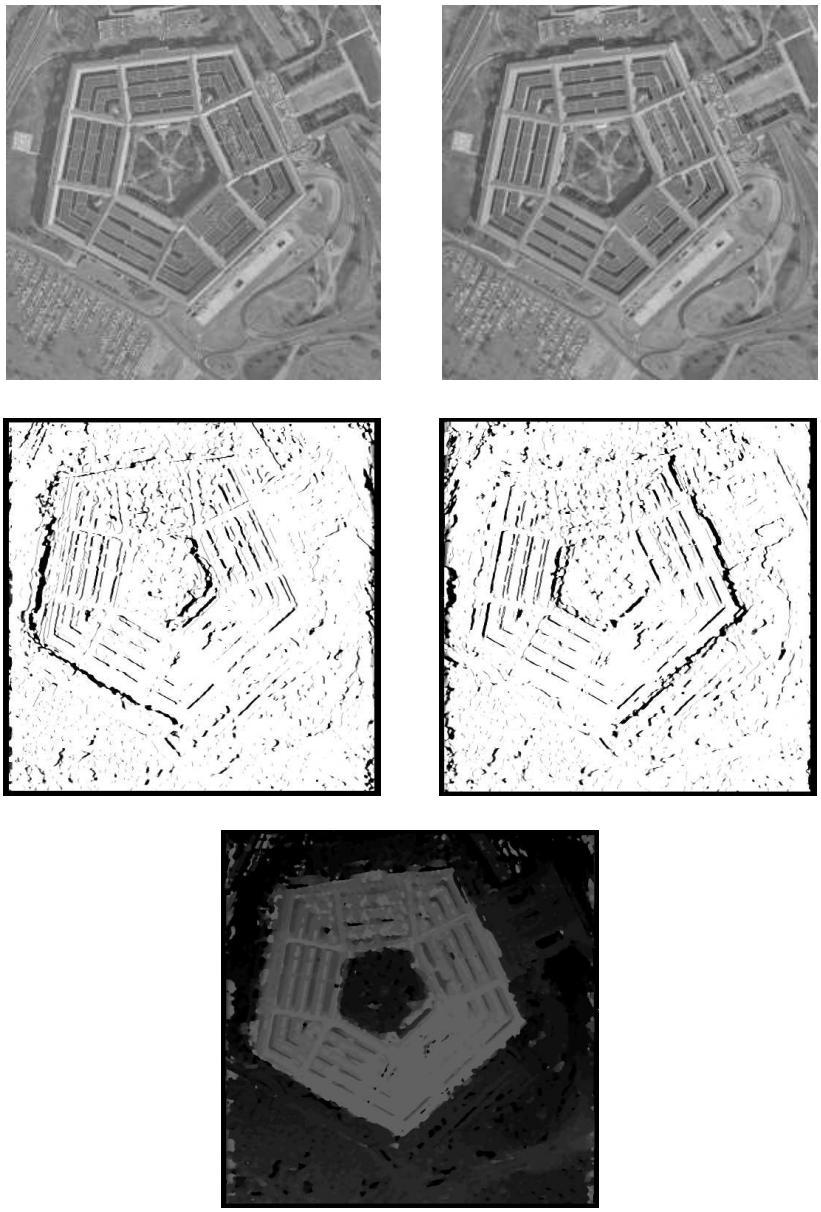


Figure 5.10: Aerial images of the Pentagon (upper images), confidence using new model (middle images) and disparity (lower image)

in real images. Since there is no real ground truth available for this image pair, it is not exactly known, where the occlusions are and how they look like. If a bird flies over the building or some tree is moving in the wind, occlusion changes. However, the detected occlusions refering to the building look pretty realistic.

5.2 Stereo Image Pairs

In the previous experiments, no assumptions about how the images have been created are integrated into the algorithm. A good matching result depends on a good selection of the test patch size and an appropriate mean displacement d and thus on a priori knowledge from outside. The experiments described in this and in the next section now use the stereo and optic flow constraints and do not need such a d or a special test patch size. However, for rectified stereo images the test patch height is set to one.

First, a stereo scene with some real objects partly occluding the background and other objects is used. Although this scene does not contain many large homogeneous areas, there are some effects that disturb the matching, e.g. reflection and transparency at the bottle and the phone as well as strong noise including pixel errors.

Since there is no ground truth available for this scene, the results are evaluated only roughly by comparing the matching results qualitatively. First, the former algorithm is applied to the images with a test patch size of 13x1 and $d = (-5; 0)$, which should be optimal parameters for that model. After 15 iterations the results shown in figure (5.12) evolve, which shows also the results of the new model. The new algorithm uses scale matching with a constant test patch size of 7x1 and no special d, i.e. $d = 0$, which are the default parameters for scale matching on rectified images. The minimum image width has been set to 40, while the original images are of size 320x240 pixels.

Although there is no ground truth available, it is clear that both algorithms produced some errors and that the object boundaries are not as sharp as they should be. However, due to the scale matching the surfaces of the bottle and the background are much smoother using the new model, while the object borders are not over-smoothed. A drawback of the scale approach can be seen at the upper phone boundary in the foreground: It is more a curve than a sharp line, which is an error introduced by problems at higher pyramid layers. Due to the homogeneous color of the phone, occlusions are not detected properly and expectation values are falsified, which leads to a wrong patch positioning.

Figure 5.11: Stereo scene using real objects

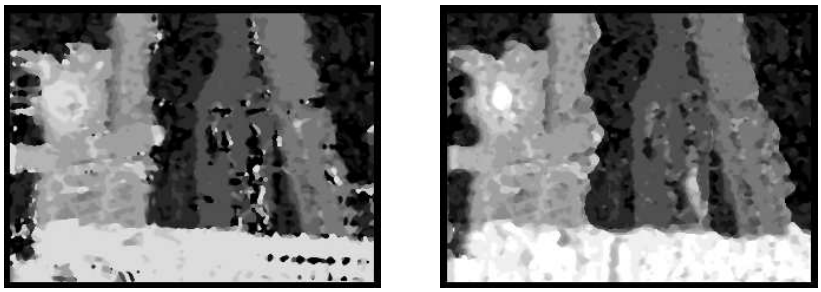


Figure 5.12: Disparity for old (left) and new model (right)

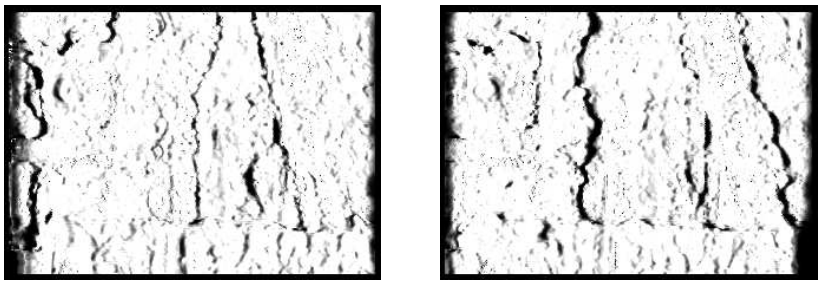


Figure 5.13: Correspondence probabilities

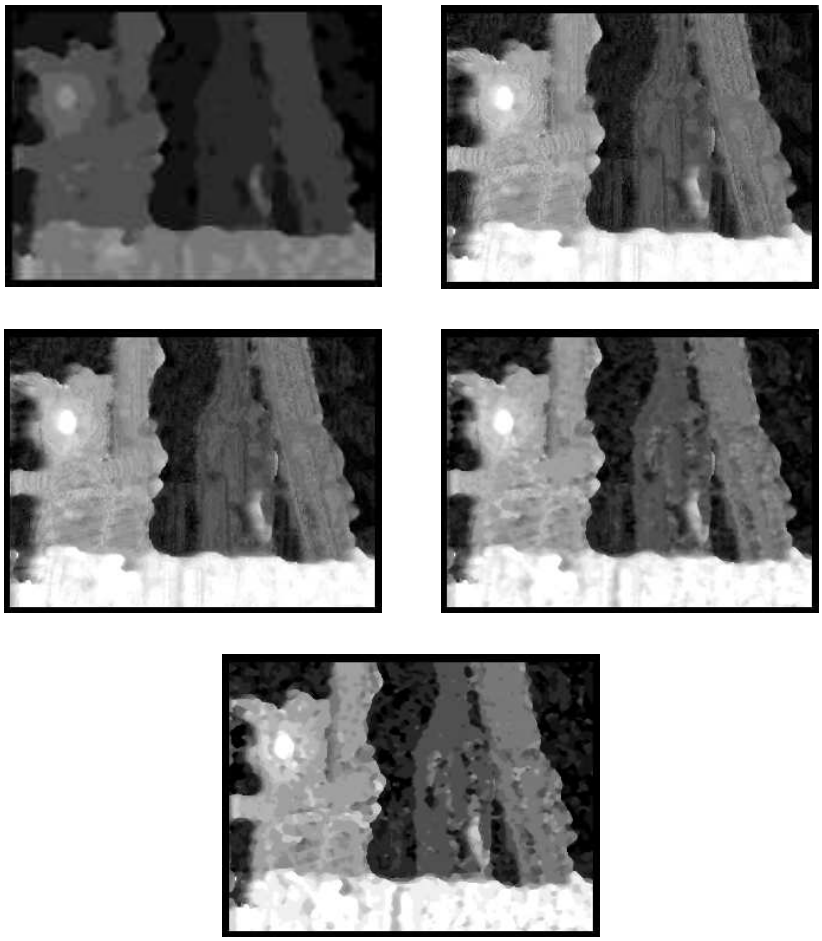


Figure 5.14: Disparity before and after first iteration (upper images), after two and five iterations (middle images) and after twelve iterations (lower image). Disparity before first iteration is interpolated from smaller scale, where absolute displacements are smaller (and thus darker) than at original size.

Apart from that these images again contain strong structure and subjectively, most half-occluded pixels have been found. At least the main occlusion lines at contours can be seen, although the test patches are smaller than the maximum disparity. This shows that they have been positioned very well by the scale approach. The main occlusion lines can be found at the left respective right foreground object borders. Note that there are also many false positives, i.e. pixels with matches, that have a low correspondence probability. The reasons for this are mainly noise, subpixel correspondences and wrong matching results. However, single false positives do not disturb the matching process, since the only effect of low correspondence probability is that the pixel's matching weight is reduced. They depend on their neighbors as before, which should still have enough other neighbors themselves, that provide information.

Another important point is that the matching results get worse at the last scale, since the images are not smoothed and contain heavy noise. This effect is documented in figure (5.14). At the second largest scale, the matching result is quite good, as can be seen in the (interpolated) disparity image before the first iteration. In the following iterations, the matching process gets trapped in optima created by noise.

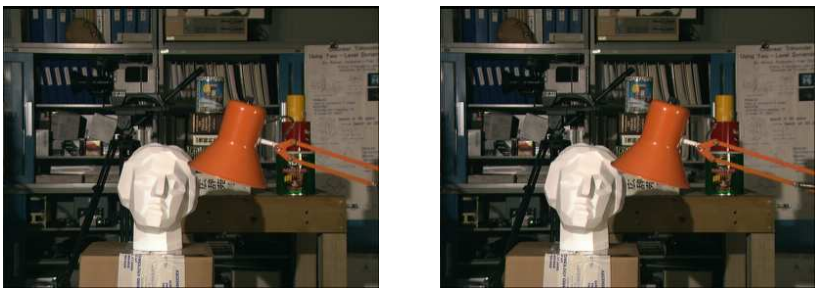


Figure 5.15: Tsukuba stereo pair

The next example is the well-known Tsukuba scene[2], which again contains some depth discontinuities. The images are matched over scale using the stereo constraints. As pointed out in the previous chapter, a more accurate method of initializing the correspondence probabilities than using the higher pyramid layer may be their *pre-computation* (see section (3.2.4)), which takes additional iterations and therefore slows down the algorithm. In this example the occlusion information calculated using the smaller pyramid images is

[2]The Tsukuba image pair is a real (rectified) stereo image pair. It is provided by Dr. Y. Ohta and Dr. Y. Nakamura from the University of Tsukuba, Japan

now compared to the pre-computation of correspondence probability. Both methods yield similar *matching* results, but the confidence images differ as can be seen in figure (5.16). Using the scale approach they contain broader regions of occlusion (not always absolutely well-placed), which is simply a result of the interpolation. Comparing the pre-computation results to the (estimated) ground truth occlusion, almost all occlusion lines are detected but they are slightly too thin. Consequently, only if extremely fine and exact occlusion information is desired, the drawback of additional iteration steps may be acceptable. An optimization of the matching result, however, is hardly noticable in general, so that the chosen approach (using occlusion information of the higher scale) is sufficient for matching.

Furthermore, the correspondence probability images show that the algorithm can extract occluded pixels even in real images with (partly) weak structure beneath object borders. Though these probability images are quite noisy, the main occlusions can be seen well. Note again, that the left confidence images only show half-occluded pixels at left foreground object borders. It is also interesting to see that these images are completely black at the left image border. This is no error or inaccuracy due to border problems. Quite the reverse, these pixels are also half-occluded, since they have no correspondence in the other image: The true match lies beyond the image border and therefore it can provide no support for them. Hence, they are detected to be occluded as if they were hidden behind an aperture.

The disparity image of (5.18) shows a good matching result compared to the ground truth image. The scale approach has positioned the test patches well and the depth discontinuities are represented in the disparity image. However, this image also shows a problem of the algorithm: In the lower left and upper right corners there are brighter areas, which are matched completely wrong: Their test patches must have been positioned at a completely wrong location. This can happen in large homogeneous regions, which have low contrast even in the smallest images of the pyramid. The expectation values of the patches do not reflect the true match values. That is, patches of the larger pyramid images are positioned at wrong positions. Matching results in these regions have no chance to be correct then. This behaviour can be observed in the corners of the image but also for some smaller regions near the center.

5.3 Optic Flow

The evaluation of optic flow is quite complicated, since for natural image sequences there is usually no ground truth available. On the other hand, for

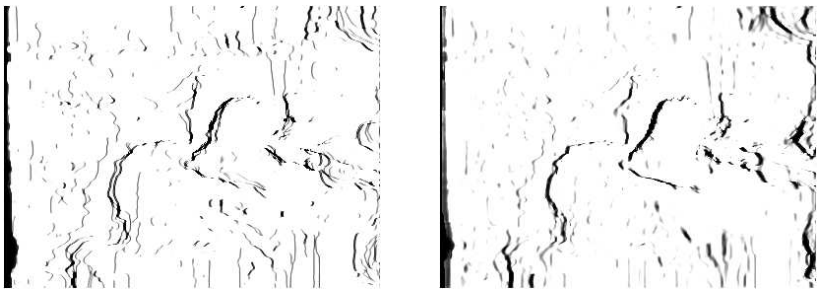


Figure 5.16: A to B confidence initialization of Tsukuba pair: Pre-computation using 8 iterations (left), interpolated values of smaller match (right)

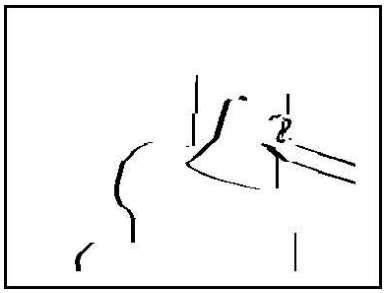


Figure 5.17: A to B (estimated) ground truth occlusion of Tsukuba pair

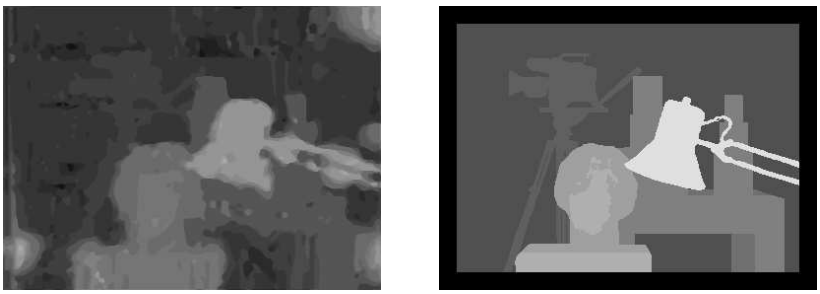


Figure 5.18: A to B disparity of Tsukuba stereo pair: Computed results (left), estimated ground truth (right)

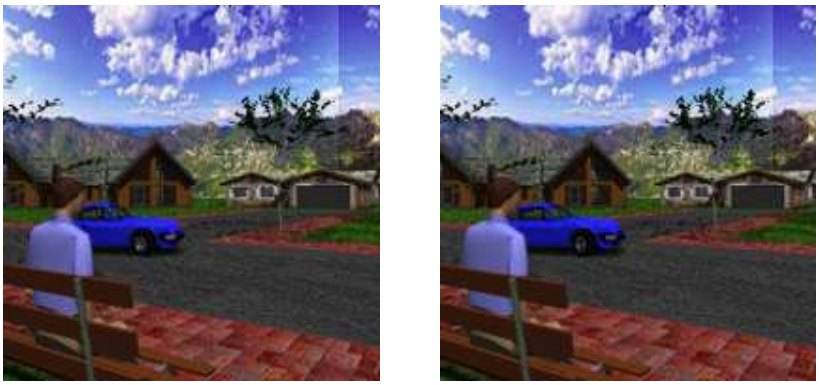


Figure 5.19: Frames 5 and 6 from the Otago street sequence

synthetically rendered scenes and patterns the question arises, how relevant they are for real applications. A quite promising approach for evaluation of optic flow algorithms is used in [6], where a ray-tracer is used to render semi-artificial scenes that look more realistic than line patterns or random dot images, but for which full ground truth is provided.

The next experiment uses 20 images of such a scene, which is called the street sequence[3]. Figure (5.19) shows some frames (of size 200x200 pixels) of this sequence. A human is sitting in the foreground while a car passes by on the street. During the scene the camera pans slowly to the right.

Subsequent frames of this sequence are matched using the constant pixel velocity assumption for four images. A test patch size of 7x5 pixels is used and the positioning is done using the scale approach. This test patch size is the default for images, which are not rectified but which are expected to contain (slightly) larger horizontal movements than vertical ones.

The absolute value of the computed optic flow is displayed in figure (5.20). It can be seen in the ground truth image, that only the car is moving in the two frames under inspection. The computed optic flow reflects this, but there are several disturbances due to the person in the foreground.

The mean error is calculated taking into account all images of the sequence and can be seen in table (5.2). In [9] McCane et al. compare several optic flow algorithms applied to this scene. The error range from [9] can also be seen in the table. Evidently, the algorithm performs quite well compared to other optic flow algorithms used for this sequence regarding the mean error e_m. An improvement from the old one to the new one can be measured,

[3]This sequence is an artificial image sequence with full ground truth provided by the University of Otago, New Zealand. For more information please refer to [6].

Figure 5.20: Computed (left) and true (right) absolute optic flow

	Old Model	New Model	Value Range from [9]
$\bar{e}_m$	0.27	0.21	0.16 .. 0.45
$\bar{\sigma}_e$	0.72	0.47	-
$\bar{e}_{ma}$	0.17	0.13	0.11 .. 0.48

Table 5.2: Results for the street scene

too, while the error variance decreases significantly. An interpretation of this is, that the error distribution is still comparable to the former model but the local errors are all reduced quite uniformly. Note that this scene fulfills the constant velocity assumption to a certain extent, though displacements are not always whole pixels.

The value of e_{ma} is quite low, which can be understood if the local error vector is viewed as a sum of two vectors: One in direction of the brightness gradient and a perpendicular vector. Using simple geometric considerations, it turns out that e_{ma} is not significantly greater than the absolute value of error in direction of the gradient. In [9] McCane et al. state that this value can give a hint about how well the algorithm can cope with the aperture problem. Nevertheless, this value refers only to the first order neighborhood of some pixel and does not depend on the absolute value of the gradient. However, regarding that measure the algorithm can handle the aperture problem quite well, since the error vectors do not always point into the (locally) most uncertain direction.

For the next experiment, the well-known flowergarden sequence[4] has been

[4]The flowergarden sequence can be obtained from Brown University at Providence (RI), USA: http://www.cs.brown.edu/people/black/Sequences

Figure 5.21: Four subsequent frames of the flowergarden sequence

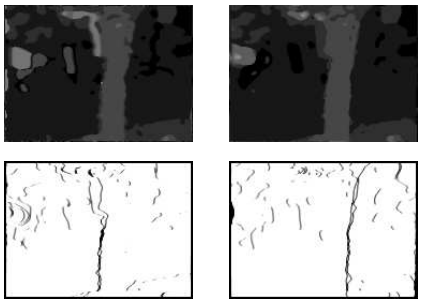


Figure 5.22: Matching results for flowergarden sequence using old model (upper left) and new model (upper right) including computed confidence using new model (lower images)

used. This is a typical rural scene, which looks quite real. Nevertheless, it is an artificial image sequence, where the displacements are roughly known. Unfortunately, no exact ground truth is available here, such that only four frames of the sequence are displayed here to show the optic flow resuls. Additionally, the results of the former algorithm are displayed. Whenever the old and the new algorithm are compared, one has to keep in mind that test patch size and positioning are manually optimized (as far as possible) for the old one, while the new one does not have such parameters, or at least they are not that important there as has been pointed out before.

The actual match is done between the two middle images, while extrapolated pixel positions of the outer images are additionally compared according to the constant pixel velocity assumption. The old algorithm is applied with a test patch size of 9x1 and $d = (-1; 0)$, while the new one uses test patches of size 7x1 at both scales.

Figure (5.22) shows that the tree in the foreground is matched similarly in both algorithms and the results are quite acceptable. Some disturbances in the optic flow occur near the left border due to large homogeneous cloud regions. However, there are less such errors when using the new model, since the scale initialization can compensate for some of them. Again, the confidence images show the main occlusion lines at the borders of the foreground

tree. Some of the small trees in the left half of the images are mixed with their blue background, that is, the sky seems to be moving with the trees. This is again not an error but occurs due to the lack of semantic knowledge.

Chapter 6

Conclusion

The model proposed in [1] has been extended with respect to occlusion. The assumed probability distributions now take into account that not all pixels must have a match in the other image. It is quite pleasant, that the new model is still similar to the old one and that the structure is preserved in principle. All assumptions used are explicit (due to the chosen probability distributions), the model is strictly based on these equations and is thus well-founded.

Additionally, a robust way of detecting occluded pixels has been derived from the model and integrated into the algorithm. For particular images without occlusion, the new model embeds into the old one, if a correct computation of correspondence probabilities is assumed. The resulting algorithm is still highly parallel and suitable for a neural network implementaion on special hardware chips (e.g. FPGAs). However, here it has been implemented for qualitative analysis as a software solution, which is quite slow but shows promising results for a future hardware implementation.

Two main applications for matching, stereo disparity and optic flow computation, have been analyzed with respect to an improved integration into the model. For optic flow, a constant pixel velocity model has been assumed and integrated into the initial pixel similarity function to stabilize the matching over several frames. The resulting matching accuracy is quite comparable to other flow algorithms tested in [9]. The optic flow integration may be enhanced even more, if the patch positioning for subsequent frames takes into account previous matching results.

In stereo disparity computation displacements vary strongly in presence of depth discontinuities. Thus stereo images are now matched using a Gauss pyramid, where match and occlusion information of a higher layer is utilized to initialize base layers and travels down the pyramid this way. Especially in scale matching, homogeneous regions are matched much better and with

faster convergence than in the former algorithm. For both integrations, improvements can be seen for the matching process compared to the old model.

However, the main difference is that the test patch positioning is handled by the algorithm and that the test patch size does not depend on the images, i.e. there are less parameters, which have to be set and upon which the matching results depend. The pyramid matching still lacks a scale parameter in the image data (as well as the former algorithm): Homogeneous regions at the highest level (with smallest image size) may introduce errors in the expectation value for the match and can push the patches of the higher scales to completely wrong positions. The same applies to fine structures in the foreground, which cannot be seen at smaller scales: If the test patches are positioned wrong, there is no way to match this pixel correctly. Since the algorithm generally has problems in matching images with large homogeneous regions, the most promising extension may be the development of some invariant measure for pixels regarding the homogeneity of their neighborhood.

Using artificial images, it has been shown that the occlusion detection does work in heavily structured regions and helps to improve the matching results at depth discontinuities, even if the pixels with low correspondence probability lie only in the other image. They influence the matching process of the given direction through bidirectional merging. However, the detection depends on the test patch size, the iteration count and a good test patch positioning. Furthermore, the correspondence probability indicating occlusions is often very noisy and also contains false positives in real images, since support is collected from the whole test patch. Constraining the support evaluation to the relevant pixels (e.g. due to reduction of the test patch size or choosing only promising candidates) may be an improvement of the occlusion detection. Nevertheless, some occlusions are not detectable without semantic knowledge of the images, e.g. if borders of foreground objects look similar to the background or objects move over a homogeneous region. To some degree these effects occur at most object borders, making an accurate detection complicated. Keeping this in mind, the occlusion detection may be at the limit of what is possible, if no semantic information is available.

Appendix A

Constant Factors in the Former Pixel-Match PDF

This section briefly shows, why the original pixel-match pdf may be interpreted to contain direction dependent probabilities instead of joint (direction independent) ones. Note that the pdf is only defined up to a scalar factor in [1] and that the factors left away are constant in the old model. Hence, these consideration do not influence the old model in principle. They are only intended to show the handling of certain factors. The derivation of the former model starts by application of Bayes' formula:

$$P(X_A, Y_A, X_B, Y_B \mid A, B) = \frac{P(A, B \mid X_A, Y_A, X_B, Y_B)P(X_A, Y_A, X_B, Y_B)}{P(A, B)}$$

The probability of the left hand side is separated according to definition of conditional probability:

$$P(X_B, Y_B \mid A, B, X_A, Y_A)P(X_A, Y_A \mid A, B) =$$

$$\frac{P(A, B \mid X_A, Y_A, X_B, Y_B)P(X_B, Y_B \mid X_A, Y_A)P(X_A, Y_A)}{P(A, B)}$$

Now Perwass et al. state that X_A and Y_A are statistically independent of the images and split the joint probabilities. Though this assumption is still valid for that model, the probabilities $P(X_A \mid A, B)$ and $P(Y_A \mid A, B)$ are kept for comparison reasons. However, given $X_A = x_A$, Y_A may be assumed uniformly distributed among the $|\mathcal{N}|$ neighbors, so that the following equation evolves:

$$P(X_B, Y_B \mid A, B, X_A, Y_A) =$$

$$\frac{P(A, B \mid X_A, Y_A, X_B, Y_B)P(X_B, Y_B \mid X_A, Y_A)P(Y_A \mid X_A)P(X_A)}{P(A, B)P(X_A \mid A, B)P(Y_A \mid A, B)}$$

Leaving out some constant factors and constraining the images to the relevant pixels yields:

$$P(X_B, Y_B \mid A, B, X_A, Y_A) \simeq$$

$$\frac{P(A|_{x_A}, B|_{x_B} \mid X_A, X_B) P(A|_{y_A}, B|_{y_B} \mid Y_A, Y_B) P(X_B, Y_B \mid X_A, Y_A)}{P(X_A \mid A, B) P(Y_A \mid A, B)}$$

Comparing this to the equations used by Perwass et al., it can be seen clearly, that the denominator factors have also been left away due to the constancy assumption. If they had not, the factors would still appear in the former pixel-match pdf:

$$P(X_B = x_B \mid A, B, X_A = x_A) \simeq$$

$$\frac{P(X_A, X_B \mid A, B)}{P(X_A \mid A, B)} \frac{1}{|\mathcal{N}|} \sum_{y_A} \max_{y_B \in \mathcal{T}_{y_A}} \left(\frac{P(Y_A, Y_B \mid A, B)}{P(Y_A \mid A, B)} P(X_B, Y_B \mid X_A, Y_A) \right)$$

$$= P(X_B \mid A, B, X_A) \frac{1}{|\mathcal{N}|} \sum_{y_A} \max_{y_B \in \mathcal{T}_{y_A}} \left(P(Y_B \mid A, B, Y_A) P(X_B, Y_B \mid X_A, Y_A) \right)$$

Appendix B

Original Proof of Convergence

This chapter is a copy of the original proof of convergence given by Perwass et al. in [1]. Note that equation numbers have been adapted to the corresponding equations in this thesis:

"In this section we will show that if the image data does indeed satisfy the assumed ordering constraint, then iterating equation (2.9) does indeed extract the correct image point matches. In order to give this proof we need to have a model of the data we expect to encounter.

We expect features that can be matched between two images not to be point like, i.e. not to be constrained to a single pixel. Instead there will be a peaked distribution of pixel similarities centered on the correct match. Note that the correct match position may lie between pixels. Nevertheless, we will make the approximation that we can identify a single pixel with the correct match position. This introduces an error of at most half a pixel.

We assume that a test patch can be approximated by a linear combination of Gaussians, whereby one Gaussian represents the correct match and the rest represent spurious matches. That is,

$$f^0(\mathbf{x},\mathbf{y}) = \rho_{\mathbf{x}} \sum_{r=0}^{R_{\mathbf{x}}} \tau_{\mathbf{x}}^r \, g(\mathbf{y},\, \mathbf{z}_{\mathbf{x}}^r,\, \sigma_{\mathbf{x}}^r), \tag{B.1}$$

where $\rho_{\mathbf{x}}$ is a normalization factor. As before $\mathbf{x} \in \mathcal{I}$ is a pixel position in image A and $\mathbf{y}$ is a position in the corresponding test area in image B ($\mathbf{y}-\mathbf{x}-\mathbf{d} \in \mathcal{T}$). Each candidate match is defined through a triplet $(\tau_{\mathbf{x}}^r,\, \mathbf{z}_{\mathbf{x}}^r,\, \sigma_{\mathbf{x}}^r)$, which gives its peak amplitude, its mean position and its standard deviation. Let $r = 0$ be the index of the correct match. That is, $(\mathbf{x}, \mathbf{z}_{\mathbf{x}}^0)$ is a correct pixel match. Then $R_{\mathbf{x}}$ gives the number of spurious matches in test patch $\mathcal{F}_{\mathbf{x}}^0$ with

$$\mathcal{F}_{\mathbf{x}}^0 := \{f^0(\mathbf{x},\mathbf{y}) \,:\, (\mathbf{y}-\mathbf{x}-\mathbf{d}) \in \mathcal{T}\}.$$

Let $(\mathbf{x}_A, \mathbf{y}_A)$ be two neighboring pixel in image A, i.e. $(\mathbf{y}_A - \mathbf{x}_A) \in \mathcal{N}$. Also let $\mathbf{z}^0_{\mathbf{x}_A}$ and $\mathbf{z}^0_{\mathbf{y}_A}$ denote the correct pixel matches for $\mathbf{x}_A$ and $\mathbf{y}_A$, respectively. Then our assumption is that the a priori combined pdf of the correct matches $\mathbf{z}^0_{\mathbf{x}_A}$ and $\mathbf{z}^0_{\mathbf{y}_A}$ is given by

$$P(\mathbf{Y}_B = \mathbf{z}^0_{\mathbf{y}_A},\, \mathbf{X}_B = \mathbf{z}^0_{\mathbf{x}_A} \,|\, \mathbf{Y}_A = \mathbf{y}_A, \mathbf{X}_A = \mathbf{x}_A) = \frac{1}{|\mathcal{T}|}\, h(\mathbf{x}_A, \mathbf{z}^0_{\mathbf{x}_A}, \mathbf{y}_A, \mathbf{z}^0_{\mathbf{y}_A}). \quad \text{(B.2)}$$

For a given correct match $(\mathbf{x}_A, \mathbf{z}^0_{\mathbf{x}_A})$ the pdf of $\mathbf{z}^0_{\mathbf{y}_A}$ is therefore given by

$$P(\mathbf{Y}_B = \mathbf{z}^0_{\mathbf{y}_A} \,|\, \mathbf{Y}_A = \mathbf{y}_A, \mathbf{X}_A = \mathbf{x}_A,\, \mathbf{X}_B = \mathbf{z}^0_{\mathbf{x}_A}) = h(\mathbf{x}_A, \mathbf{z}^0_{\mathbf{x}_A}, \mathbf{y}_A, \mathbf{z}^0_{\mathbf{y}_A}). \quad \text{(B.3)}$$

However, any $\mathbf{z}^p_{\mathbf{x}_A}$ and $\mathbf{z}^q_{\mathbf{y}_A}$ *alone* has a uniform pdf.

$$P(\mathbf{X}_B = \mathbf{z}^p_{\mathbf{x}_A} \,|\, \mathbf{X}_A = \mathbf{x}_A) = \frac{1}{|\mathcal{T}|}\,; \quad P(\mathbf{Y}_B = \mathbf{z}^q_{\mathbf{y}_A} \,|\, \mathbf{Y}_A = \mathbf{y}_A) = \frac{1}{|\mathcal{T}|}.$$

Furthermore, given an incorrect match $(\mathbf{x}_A, \mathbf{x}_B)$, the pdf for $\mathbf{z}^0_{\mathbf{y}_A}$ is a uniform distribution. Also, given a correct match $(\mathbf{x}_A, \mathbf{z}^0_{\mathbf{x}_A})$, the pdf for an incorrect match $\mathbf{z}^q_{\mathbf{y}_A}$, $q > 0$ is a uniform distribution. That is, if $(p, q) \neq (0, 0)$ then

$$P(\mathbf{Y}_B = \mathbf{z}^p_{\mathbf{y}_A} \,|\, \mathbf{Y}_A = \mathbf{y}_A, \mathbf{X}_A = \mathbf{x}_A,\, \mathbf{X}_B = \mathbf{z}^q_{\mathbf{x}_A}) = \frac{1}{|\mathcal{T}|}. \quad \text{(B.4)}$$

In order to show that the algorithm converges we have to show that the expectation value of $\hat{P}(\mathbf{X}_B \,|\, \mathcal{F}^t,\, \mathbf{X}_A)$ as given in equation (2.8) is maximized for a correct match $(\mathbf{x}_A, \mathbf{z}^0_{\mathbf{x}_A})$. Note that we can write equation (2.8) as follows.

$$\begin{aligned} &\hat{P}(\mathbf{X}_B = \mathbf{x}_B \,|\, \mathcal{F}^t,\, \mathbf{X}_A = \mathbf{x}_A) \\ &\quad \simeq \frac{1}{|\mathcal{N}|} \sum_{\{\mathbf{y}_A : (\mathbf{y}_A - \mathbf{x}_A) \in \mathcal{N}\}} \max_{\mathbf{y}_B}\; f^t(\mathbf{x}_A, \mathbf{x}_B)\, f^t(\mathbf{y}_A, \mathbf{y}_B)\, \hat{h}(\mathbf{x}_A, \mathbf{x}_B, \mathbf{y}_A, \mathbf{y}_B), \end{aligned} \quad \text{(B.5)}$$

We will now consider the expectation value E_s for a single pair of correspondences $(\mathbf{x}_A, \mathbf{x}_B)$ and $(\mathbf{y}_A, \mathbf{y}_B)$, where $\mathbf{x}_A$, $\mathbf{x}_B$ and $\mathbf{y}_A$ are fixed.

$$E_s := E\big[f^0(\mathbf{x}_A,\, \mathbf{x}_B)\; f^0(\mathbf{y}_A,\, \mathbf{y}_B)\; \hat{h}(\mathbf{x}_A,\, \mathbf{x}_B,\, \mathbf{y}_A,\, \mathbf{y}_B)\big].$$

First we substitute for the f^0 functions using equation (B.1).

$$\begin{aligned} E_s \;=\; & E\Bigg[\rho_{\mathbf{x}_A}\rho_{\mathbf{y}_A} \left(\sum_{p=0}^{R_{\mathbf{x}_A}} \tau^p_{\mathbf{x}_A}\, g(\mathbf{x}_B, \mathbf{z}^p_{\mathbf{x}_A}, \sigma^p_{\mathbf{x}_A})\right) \left(\sum_{q=0}^{R_{\mathbf{y}_A}} \tau^q_{\mathbf{y}_A}\, g(\mathbf{y}_B, \mathbf{z}^q_{\mathbf{y}_A}, \sigma^q_{\mathbf{y}_A})\right)\Bigg] \\ & \times \hat{h}(\mathbf{x}_A,\, \mathbf{x}_B,\, \mathbf{y}_A,\, \mathbf{y}_B). \end{aligned}$$

Now we write the expectation value as the sum over the function values multiplied with the corresponding probability that this function value occurs.

$$
\begin{aligned}
E_s = \rho_{\mathbf{x}_A}\rho_{\mathbf{y}_A}\,\hat{h}(\mathbf{x}_A,\,\mathbf{x}_B,\,\mathbf{y}_A,\,\mathbf{y}_B)\sum_{\mathbf{w}_{\mathbf{y}_A}} \\
\Bigg[\quad \tau^0_{\mathbf{x}_A}\,\tau^0_{\mathbf{y}_A}\,g(\mathbf{x}_B,\mathbf{z}^0_{\mathbf{x}_A},\sigma^0_{\mathbf{x}_A})\;g(\mathbf{y}_B,\mathbf{w}_{\mathbf{y}_A},\sigma^0_{\mathbf{y}_A})\;h(\mathbf{x}_A,\,\mathbf{z}^0_{\mathbf{x}_A},\,\mathbf{y}_A,\,\mathbf{w}_{\mathbf{y}_A}) \\
+\quad \frac{1}{|\mathcal{T}|}\sum_{p=1}^{R_{\mathbf{x}_A}}\tau^p_{\mathbf{x}_A}\,\tau^0_{\mathbf{y}_A}\,g(\mathbf{x}_B,\mathbf{z}^p_{\mathbf{x}_A},\sigma^p_{\mathbf{x}_A})\,g(\mathbf{y}_B,\mathbf{w}_{\mathbf{y}_A},\sigma^0_{\mathbf{y}_A}) \\
+\quad \frac{1}{|\mathcal{T}|}\sum_{q=1}^{R_{\mathbf{y}_A}}\tau^0_{\mathbf{x}_A}\,\tau^q_{\mathbf{y}_A}\,g(\mathbf{x}_B,\mathbf{z}^0_{\mathbf{x}_A},\sigma^0_{\mathbf{x}_A})\,g(\mathbf{y}_B,\mathbf{w}_{\mathbf{y}_A},\sigma^q_{\mathbf{y}_A}) \\
+\quad \frac{1}{|\mathcal{T}|}\sum_{p=1}^{R_{\mathbf{x}_A}}\sum_{q=1}^{R_{\mathbf{y}_A}}\tau^p_{\mathbf{x}_A}\,\tau^q_{\mathbf{y}_A}\,g(\mathbf{x}_B,\mathbf{z}^p_{\mathbf{x}_A},\sigma^p_{\mathbf{x}_A})\;g(\mathbf{y}_B,\mathbf{w}_{\mathbf{y}_A},\sigma^q_{\mathbf{y}_A})\Bigg].
\end{aligned}
$$

The Gaussians we use are normalized such that

$$
\sum_{\mathbf{w}_{\mathbf{y}_A}} g(\mathbf{y}_B,\mathbf{w}_{\mathbf{y}_A},\sigma^q_{\mathbf{y}_A}) = 1.
$$

Therefore it follows that

$$
\begin{aligned}
E_s = \rho_{\mathbf{x}_A}\rho_{\mathbf{y}_A}\,\hat{h}(\mathbf{x}_A,\,\mathbf{x}_B,\,\mathbf{y}_A,\,\mathbf{y}_B) \\
\times\Bigg[\quad \underbrace{\tau^0_{\mathbf{x}_A}\,\tau^0_{\mathbf{y}_A}\sum_{\mathbf{w}_{\mathbf{y}_A}}\Big(g(\mathbf{x}_B,\mathbf{z}^0_{\mathbf{x}_A},\sigma^0_{\mathbf{x}_A})\;g(\mathbf{y}_B,\mathbf{w}_{\mathbf{y}_A},\sigma^0_{\mathbf{y}_A})\times h(\mathbf{x}_A,\mathbf{z}^0_{\mathbf{x}_A},\mathbf{y}_A,\mathbf{w}_{\mathbf{y}_A})\Big)}_{\text{Term 1}} \\
+\quad \underbrace{K_{\mathbf{x}_A,\,\mathbf{y}_A}}_{\text{Term 2}}\Bigg]
\end{aligned}
\tag{B.6}
$$

The constant $K_{\mathbf{x}_A,\mathbf{y}_A}$ only depends on $\mathbf{x}_A$ and $\mathbf{y}_A$. It is a sum over the amplitudes of spurious match candidates. This value is large if there are many spurious match candidates present. This is usually the case in areas of low contrast of an image. In areas of high contrast of an image $K_{\mathbf{x}_A,\mathbf{y}_A}$ has typically a low value.

It can be seen quite easily that term 1 is maximal whenever $h(\mathbf{x}_A,\mathbf{x}_B,\mathbf{y}_A,\mathbf{y}_B)$ is maximal, that is if the pixel pairs $(\mathbf{x}_A,\mathbf{x}_B)$ and $(\mathbf{y}_A,\mathbf{y}_B)$ satisfy the a priori distribution for true pixel matches. In this case also the factor $\hat{h}(\mathbf{x}_A,\mathbf{x}_B,\mathbf{y}_A,\mathbf{y}_B)$ of equation (B.6) is maximal.

The expectation value E_c of the probability that a pixel pair $(\mathbf{x}_A, \mathbf{x}_B)$ is a correct match (cf. equation (2.8)) is now given by

$$\begin{aligned} E_c(\mathbf{x}_A, \mathbf{x}_B) &= E\big[\hat{P}(\mathbf{X}_B = \mathbf{x}_B \,|\, \mathcal{F}^0,\, \mathbf{X}_A = \mathbf{x}_A)\big] \\ &= \frac{1}{|\mathcal{N}|} \sum_{\mathbf{y}_A} \max_{\mathbf{y}_B} E_s(\mathbf{x}_A, \mathbf{x}_B, \mathbf{y}_A, \mathbf{y}_B). \end{aligned} \tag{B.7}$$

We have seen in equation (B.6) that $E_s(\mathbf{x}_A, \mathbf{x}_B, \mathbf{y}_A, \mathbf{y}_B)$ is maximal if $(\mathbf{x}_A, \mathbf{x}_B)$ is a correct match and if $\mathbf{y}_B$ is chosen such that $h(\mathbf{x}_A, \mathbf{x}_B, \mathbf{y}_A, \mathbf{y}_B)$ is maximized. Therefore E_c is maximal if $(\mathbf{x}_A, \mathbf{x}_B)$ is a correct match. For all other, incorrect pairs E_c has a lower value which depends on $K_{\mathbf{x}_A, \mathbf{y}_A}$ from equation (B.6). That is, within a homogeneous region of an image, the expectation value for a correct match is only slightly higher than those for incorrect matches. In an area of high contrast, on the other hand, the maximum of E_c is strongly peaked.

The extreme case of this behavior can be seen in figure [...][1]. Within the (perfectly) homogeneous part of the initial images, the test patches are uniformly white and applying the algorithm to those areas further away from the image center, does not change them, since all pixel matches are equally likely. Towards the center, however, the most likely pixel matches are strongly peaked.

Figure [...][1] also demonstrates that a single evaluation of equation (2.8) is not sufficient to find the correct matches throughout the image. A number of iterations are necessary to distribute constraints on the pixel matches through the image.

Since E_c is peaked for correct matches, an iterative application of equation (2.8) will amplify these peaks while attenuating spurious matches. Nevertheless, due to noise in the images, spurious matches may satisfy the a priori distribution of correct matches better than the correct matches. However, this does not change the fact that the algorithm will converge to a single match position."

[1] See [1] to view these figures.

Bibliography

[1] C.B.U. Perwass and G. Sommer. Dense Image Point Matching through Propagation of Local Constraints. Christian-Albrechts-University Kiel, Technical Report Nr. 0205, 2002

[2] C.B.U. Perwass and G. Sommer. An iterative Bayesian Technique for Dense Image Point Matching. In Proceedings of Dynamic Perception, pp.283-288, 2002.

[3] C.B.U. Perwass and G. Sommer. A fuzzy logic algorithm for dense image point matching. In Proceedings of Vision Interface, pp.39-47, 2001.

[4] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6(6):pp.721-741, 1984.

[5] J.L. Marroquin, F.A. Velasco, M. Rivera, and M. Nakamura. Gauss-Markov measure field models for low-level vision. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(4):pp.337-348, 2001.

[6] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills, Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms. In Proceedings of Ninth British Machine Vision Conference, pp.195-204, Southhampton, United Kingdom, 1998.

[7] W. Hoff, N.Ahuja. Surfaces from Stereo: Integrating Feature Matching, Disparity Estimation, and Contour Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(2):pp.121-136, 1989.

[8] G. Medioni, R. Nevatia, Matching images using linear features, IEEE Transactions on Pattern Analysis and Machine Intelligence 6(6), 675-685, 1984

[9] Brendan McCane, Ben Galvin and Kevin Novins. On the Evaluation of Optical Flow Algorithms, Fifth International Conference on Control, Automation, Robotics and Vision, pp.1563-1567, Singapore, 1998.

[10] B.K.P. Horn and B.G. Schunck. Determining optical flow, Artificial Intelligence, Volume 17, pp.185-204, 1981.

[11] D. Marr and T. Poggio. Cooperative computation of stereo disparity. Science, 194: pp.209-236, 1976.

[12] Stephen T. Barnard and William B. Thompson. Disparity analysis of images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2(4): pp.333-340, 1980.

[13] Takeo Kanade and Masatoshi Okutomi. A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment, IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(9): pp.920-932, 1994.

[14] C.L. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(7): pp.675-684, 2000.

[15] Peter N. Belhumeur. A Bayesian approach to binocular stereopsis. International Journal of Computer Vision, 19: pp.237-262, 1996.

[16] N. Vasconcelos and A. Lippman. Empirical bayesian motion segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(2): pp.217-221, 2001.

[17] L. Zhou, D.B. Goldgof, and K. Palaniappan. Tracking nonrigid motion and structure from 2d satellite cloud images without correspondences. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(11): pp.1330-1336, 2001.

[18] P.H.S. Torr, R. Szeliski, and P. Anandan. An integrated bayesian approach to layer extraction from image sequences. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(3): pp.297-303, 2001.

[19] H.W. Haussecker and D.J. Fleet. Computing optical flow with physical models of brightness variation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(6): pp.661-673, 2001.

[20] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. International Journal of Computer Vision, 28(2): pp.155-174, 1998.

[21] M. Felsberg. Low-Level Image Processing with the Structure Multivector. Christian-Albrechts-University Kiel, Technical Report Nr. 0203, 2002.

[22] T. Rabsch. Implementierung eines Dichte-Korrespondenzfindungs-Algorithmus auf einem FPGA. Diploma Thesis, Christian-Albrechts-University Kiel, 2003. To be published.

[23] X. Jiang, H. Bunke. Dreidimensionales Computersehen, Springer Verlag, Berlin, 1997.

[24] P. Haberäcker. Praxis der Digitalen Bildverarbeitung und Mustererkennung, Hanser Verlag, München, 1995.

[25] S. T. Barnard. Stochastic Stereo Matching over Scale. International Journal of Computer Vision, 3, pp.17-32, 1989.

[26] E. P. Simoncelli. Course-to-fine Estimation of Visual Motion. IEEE Signal Processing Society, Eighth Workshop on Image and Multidimensional Signal Processing, Cannes, September 1993.

[27] J. Haigh. Probability Models, Springer Verlag, London, 2002.

[28] H. Bauer. Wahrscheinlichkeitstheorie, de Gruyter, Berlin, Fourth Edition, 1991.

[29] C.M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, Oxford, 1996.

[30] P.S. Churchland and T.J. Sejnowski. Grundlagen zur Neuroinformatik und Neurobiologie. Vieweg, Braunschweig, 1997.

[31] I.N. Bronstein and K.A. Semendjajew. Taschenbuch der Mathematik, Verlag Harri Deutsch, Frankfurt am Main, 13th Edition, 1973.

[32] M. Wannemacher, Das FPGA-Kochbuch, International Thomson Publishing, Bonn, 1998.

[33] B. Stroustrup. Die C++ Programmiersprache. Addison-Wesley, München, Third Edition, 1997.

[34] K.D. Kammeyer. Nachrichtenübertragung. B.G. Teubner, Stuttgart, Second Edition, 1996.

[35] B. Jähne, Digitale Bildverabeitung, Springer Verlag, Heidelberg, Fourth Edition, 1997.