Synthese von 3D Videos

Martin Lambers

Diplomarbeit
15. Dezember 2005

Thema gestellt von
Prof. Dr. Xiaoyi Jiang
Westfälische Wilhelms-Universität
Institut für Informatik

Contents

1. Introduction

Three-dimensional video systems promise a more natural viewing experience than existing two-dimensional systems such as television. Currently only specialized 3D solutions exist, for applications such as 3D modeling or computer games. These solutions often require specialized software and hardware (shutter glasses, 3D projectors). Recently [13], efforts were made to improve all steps needed for general 3D video systems: content creation, encoding, transmission, and display technologies. The progress in this area is largely due to the use of depth image based rendering (DIBR).

The human visual system normally gets two images from the eyes, representing two perspectives of a scene, with a small deviation. This stereo pair is needed to create a perception of depth[1]. All 3D display systems thus work by presenting stereo pairs to the viewer in one way or another.

Early schemes for 3D video systems worked with stereo pairs in all steps involved, from content creation to displaying. Newer approaches use depth image based rendering instead: the content consists of a conventional 2D video and a depth map for each video frame. A 2D video frame represents a "middle" view of a scene. The corresponding depth map holds information about the distance of an object in the scene from the camera, for each pixel. The two views that form the stereo pair can be computed from this middle view and its depth map after the distance between the two virtual eyes was chosen. This generation of stereo pairs needs to be done only in the displaying step.

Key advantages of this approach are:

- Different display systems can choose different virtual eye distances for optimal performance.
- The viewer can adapt the display system to his needs. Tests show that different viewers experience an optimal 3D effect for different distances of the virtual eyes [6].

[1] Many other aspects are important as well, most notably the user's experience and knowledge about objects in the scene.

- Backward compatibility: existing 2D video systems can simply discard the depth information and just display the 2D video stream.

The focus of this work is content creation: the goal is to compute a complete set of depth maps for a video scene from only a few given depth maps. This allows to

1. Ease the recording of 3D material. Instead of using expensive real-time depth sensors that are able to deliver $25-30$ depth maps per second, it is possible to use cheaper sensors that generate less depth maps, and fill in the missing maps automatically.

2. Enhance existing 2D video material with 3D effects by automatically completing depth information from a few manually created depth maps. Given the vast amount of existing 2D material, this is an important application.

Chapter 2 gives a complete overview of the current state of 3D video systems based on depth images, with an accent on the rendering of the stereo pairs. A technique to synthesize depth maps using only sparse depth information is presented in chapter 3. Experimental results are shown in chapter 4. Chapter 5 concludes this work. Details of the implementation are listed in chapter 6.

2. 3D video systems based on depth images

2.1. Overview

Technology for all components of a complete 3D video system based on depth images is available today [13]. The following components are required:

- 3D content generation. 3D content in this context is a conventional 2D video with a depth map for each video frame.
- Encoding. The video and depth data needs to be encoded efficiently for transmission.
- Transmission. The encoded data must be transmitted to the user, using existing digital video broadcasting techniques if possible.
- Rendering. This step is preferably integrated into the display device. A stereo pair must be generated for each 2D video frame and its corresponding depth map.
- Display. The display system presents the stereo pairs to the user.

2.1.1. Content creation

Content creation falls into one of two categories:

1. Recording 3D content with a 2D camera and a depth sensor.

   Active range cameras are used to record 2D video and depth information simultaneously. Such camera systems are still limited in their scope.

   The *Zcam* [9] from 3DV Systems[1] uses a low power infrared laser to send light pulses into the scene. Depth information is extracted by measuring the *time-of-flight* of the pulse.

[1] `http://www.3dvsystems.com/`

According to its producer, the Zcam achieves a video frame rate of 25 frames per second and a resolution of 720×576 pixels. This matches the european PAL TV standard. The per-pixel depth information has an accuracy of 0.5 cm. The manufacturer recommends to use the camera only for indoor scenes with ranges between 1 and 10 meters.

The *HDTV Axi-Vision Camera* prototype [11] replaces the pulsed laser of the Zcam with a LED that emits constantly increasing and decreasing light. This way, the camera is able to record video and per-pixel depth information at high resolutions without the need for high power laser sources. The time-of-flight of the LED light can be measured when intensity altering effects (such as different reflectance of objects in the scene) were eliminated by combining images taken at increasing and decreasing light.

The prototype has seen several improvements over the past years [10]. The system is currently able to record frames with 1280×720 pixels at a rate of 29.97 frames per second, or 853×480 pixels at a rate of 59.94 fps. This meets HDTV requirements. Per pixel depth information is recorded with a resolution between 1.7 cm (at 2 meter distance) and 4.9 cm (at 10 meter distance). The system is still limited to indoor use only [10].

2. Enhancing existing 2D video with depth information.

   Many attempts were made to automatically gain depth information from 2D images and videos:

   - Shape from shading: The goal of this family of methods is to derive a 3D scene description from one or more 2D images. See [22] for a comparison of different methods.
   - Structure from motion: Recovering 3D information from a group of 2D image frames. An overview is given in [12].
   - Depth from focus: Calculating distances to points in a scene by modeling the effect that the camera's focal parameters have on the image. An overview can be found in [5].
   - Depth from edges: This technique is used in [17] to generate pseudo depth maps as a surrogate for real depth data in certain situations.
   - Depth from two or more views of a scene.

   All these methods are limited to certain types of images that must fulfill strong requirements. No method is known to extract reliable depth information from arbitrary 2D images or video sequences.

However, these methods can help with the manual creation of initial depth maps for a 2D video scene. Starting with a few such depth maps, the rest can then be computed with the methods presented in chapter 3.

2.1.2. Encoding

Many encoding methods for 2D video are available and can be used to encode the 2D video part of the 3D content. The depth maps are usually stored as graylevel video frames with the same resolution as the 2D video. This allows to distinguish 256 depth values, from "far" (0, black) to "near" (255, white). In this form, the depth maps can be handled just as usual graylevel video content. Since most multimedia container formats allow multiple video (and audio) data sources to be multiplexed into one file or stream, the combination of the 2D and depth data is not a problem. A comparison of various video codecs and their applicability to combined 2D and depth data can be found in [6].

2.1.3. Transmission

Technologies that are able to transmit digital video and audio data exist today, for example DVB-T, digital satellite TV programs, or digital cable TV. These existing channels can be used to distribute 3D video content.

2.1.4. Rendering

In the rendering step, the 2D video frames and their depth maps are converted into stereo image pairs. This step is preferably integrated into the display device, to allow the device and its user to choose the optimal parameters. The details of this step are discussed in the next section.

2.1.5. Display

The display device must present stereo image pairs to the user. Each eye of the user must receive the appropriate image, so that the user experiences the best possible 3D effect. Multiple systems are in common use:

- Anaglyph glasses: Each stereo pair is combined into a single image, using special colors. The user wears colored glasses (red/blue, red/green or red/cyan) that filter the combined image so that each eye sees only one image of the original pair. This technique is inconvenient for the user, degrades the image quality (especially with respect to color perception), and tires the

eyes after a short while. Its advantage is that it can be used with any existing 2D display at very low costs.

- Shutter glasses: A common 2D monitor switches between the left and right image of a stereo pair with a high frequency. The user wears special glasses that are synchronized with this monitor frequency (often via a special synchronization cable). The glasses darken the left eye every time the monitor displays the image destined for the right eye, and vice versa. While this technique is an improvement in comparison to anaglyph glasses, mainly because the image quality is not degraded, the user is still required to wear glasses, and the flicker between the left and right view tires the eyes.

- Autostereoscopic monitors: These monitors are usually 2D monitors that use a special filter to control the area in which each pixel is visible. That way, one group of pixels is only visible in one set of areas, while the second group of pixels is only visible in the complementing set of areas. The user must position its head so that the left eye views the group of pixels that display the left part of the stereo pair, and the right eye views the pixels that display the right part of the stereo pair. Multiple methods to achieve this visibility control exist, and there are multiple common ways to divide the pixels of a screen into two halves. The importance of the head position reduces the usability of 3D monitors, but the viewing experience is clearly more comfortable than with special glasses. 3D monitors of this kind are available today, for example from DTI[2].

- Autostereoscopic monitors with head tracking: To overcome the limitations imposed by the fixed head position, newer monitors integrate head tracking techniques that are able to adapt the pixel visibility controls to the position of the user's head. One such monitor is available from the Fraunhofer Institute for Telecommunications[3]. A prototype of a monitor suitable for multiple viewers that can freely move inside a room-sized area is presented in [2].

2.2. Depth image based rendering

Depth image based rendering is the key part of 3D video systems as examined in this work. It means the creation of a stereo image pair from a 2D video frame and a corresponding depth map. Note that the creation of stereo pairs works by processing one frame at a time; earlier and later frames are not considered.

[2] http://www.dti3d.com/

[3] http://www.hhi.fraunhofer.de/german/im/products/Cebit/free2C/

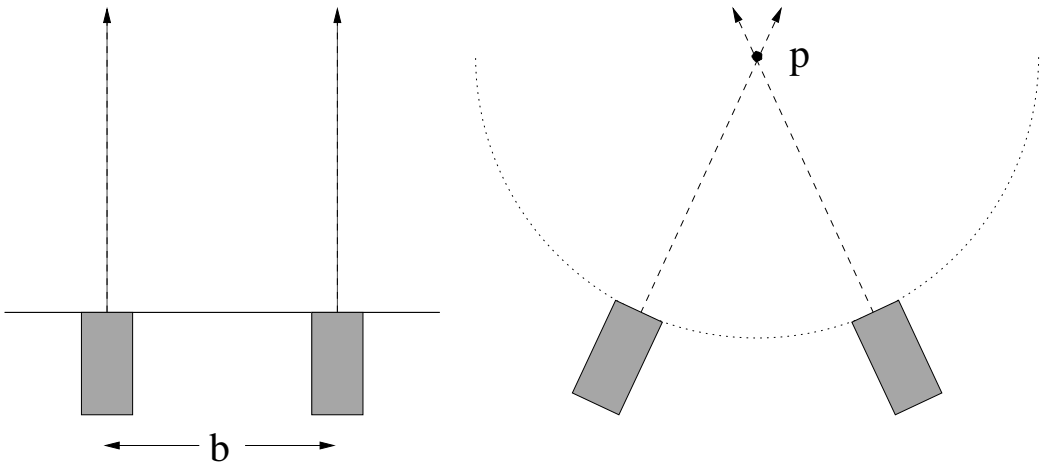

Figure 2.1: Typical dual-camera setups: the shift-sensor approach (left) and the toed-in approach (right).

DIBR is a special case of 3D image warping. The necessary computation formulas can thus be derived from the general 3D warping equation (see for example [7] section 3.1 and [14] section 11.2.5). In the following subsection, the formulas will be deduced geometrically instead, because this will clarify some important characteristics of the rendering step.

2.2.1. Stereoscopic view synthesis

A stereoscopic view consists of two identical cameras viewing the same scene. The camera positions and the two optical axes lie in one plane, the view plane. Two approaches exist in typical hardware dual-camera systems (see figure 2.1):

1. The *shift-sensor* approach. The cameras are arranged so that the optical axes are parallel, and the direction of their base line aligns with the direction of the scan lines. The horizontal distance of the cameras is b. This setup is also known as the standard stereo geometry.

2. The *toed-in* approach. The visible areas of the two cameras overlap, and the optical axes intersect in the point p.

The shift-sensor approach causes some problems for hardware dual-camera setups (the adjustment of parallel optical axes is difficult), but it leads to a simpler mathematical model. It is thus better suited for use in the synthesis of stereoscopic views.

For DIBR, it is assumed that the 2D image is recorded by a camera C_c that is placed between the two virtual cameras C_l (left view) and C_r (right view) as in

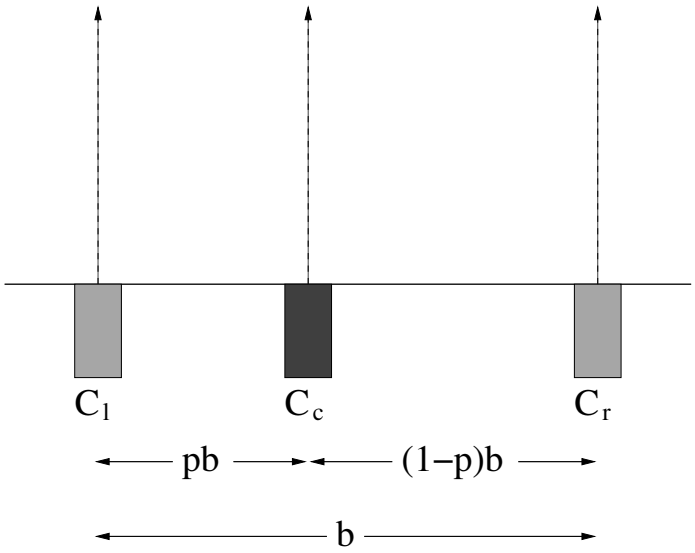

Figure 2.2: Virtual camera setup for depth image based rendering.

figure 2.2. The position p of C_c can vary between C_l ($p = 0$) and C_r ($p = 1$), but is usually the exact middle ($p = 0.5$).

To examine how to compute the views of the two virtual cameras from the given center view, it is sufficient to consider the geometry of two cameras C_1 and C_2, as illustrated in figure 2.3: in a first step, C_1 is identified with C_l, and C_2 with C_c. This allows to examine the computation of the view of C_l. In a second step, the same can be done for C_r by identifying C_1 with C_c and C_2 with C_r.

In figure 2.3, the focal length of the cameras C_1 and C_2 is f, and their distance b' equals pb in the first step and $(1-p)b$ in the second step. The point m has a distance d to the focal plane, and is projected onto the points m_1 and m_2 on the image planes I_1 and I_2. The shifts of the image points m_1 and m_2 to the center of the image planes are x_1 and x_2. Note that these are purely horizontal shifts. A 2D image can therefore be handled by processing the scan lines independently from each other.

The theorem on intersecting lines gives the relation

$$\frac{d}{f} = \frac{u_1}{x_1} = \frac{u_2}{x_2} \tag{2.1}$$

Using $b' = u_1 + u_2$, this equation can be written as

$$x_1 + x_2 = \frac{b' f}{d} \tag{2.2}$$

If the position x_2 and the depth d of a pixel in I_2 are known, then the corresponding position x_1 in I_1 of this pixel can be computed with equation 2.2 when

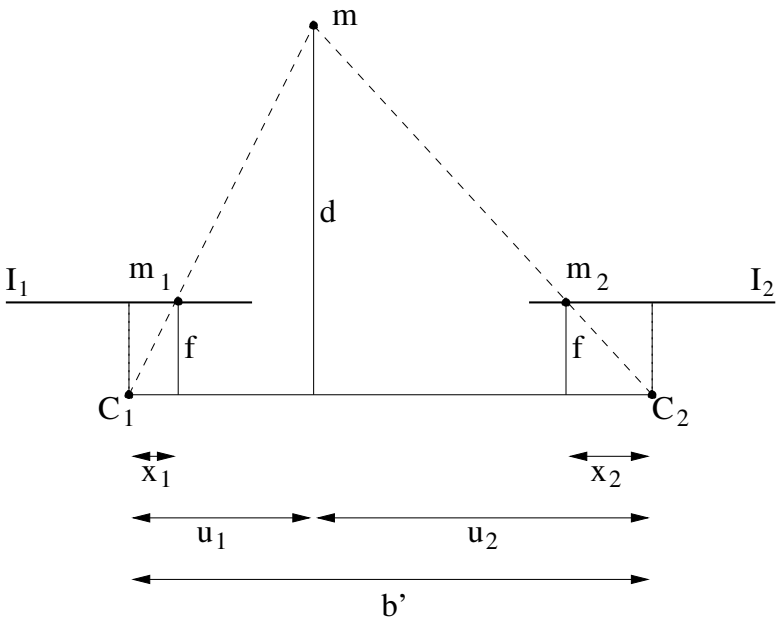

Figure 2.3: The standard stereo geometry with two cameras.

b' and f are known. This is the key principle of DIBR.

In the first step, with $C_1 = C_l$ and $C_2 = C_c$, equation 2.2 results to

$$x_l = x_c + p\frac{bf}{d} \tag{2.3}$$

Similarly,

$$x_r = x_c - (1-p)\frac{bf}{d} \tag{2.4}$$

can be derived in the second step, with $C_1 = C_c$ and $C_2 = C_r$.

These formulas allow to compute the left and right views I_l and I_r from a center view I_c with a given depth map: for each scan line of I_c, the necessary new horizontal positions x_l and x_r of each pixel can be computed from its position x_c and depth d.

Note that the shifts x_l and x_r are inversely proportional to the depth of the current pixel. For points in infinite distance, both shifts are zero.[4]

[4]The depth at which the shift is zero is called the *zero parallax setting* (ZPS) by some authors, and is set to 127 ("middle") instead of 0 ("far"), so that objects behind the ZPS plane are shifted in the opposite direction (see for example [21]).

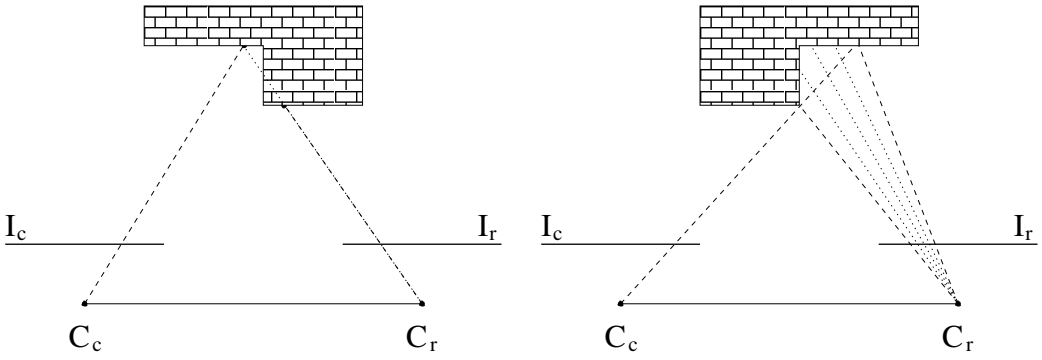

Figure 2.4: Occlusion (left) and disocclusion (right) when computing I_r from I_c.

2.2.2. Occlusions and disocclusions

Two problems can arise when computing the new view I_r from I_c:

1. Points that C_c can see but not C_r are occluded in I_r by other points.
2. Points that C_r can see but not C_c are unknown. These points are called *disocclusions*.

This is illustrated in figure 2.4. When computing I_l from I_c, the situation reverses: occlusions in I_r correspond to disocclusions in I_l, and vice versa.

Occlusions can be handled in the obvious way: the nearer point overwrites the farther point. Disocclusions are more problematic: the holes that they cause in the resulting image must be filled. Figure 2.5 shows an example of occlusion and disocclusion effects at a depth discontinuity.

Disocclusion hole are usually filled line-by-line, to make full use of the line-oriented nature of the DIBR algorithm.[5] This restriction leaves only the colors of the pixels to both side of a hole as candidates for the color to fill this hole with. Several simple techniques can be used:

- Use the color of the pixel that belongs to the nearer object.
- Use the color of the pixel that belongs to the farther object.
- Use the average of both colors.
- Use a linear color gradient.

Figure 2.6 illustrates the effects of the different hole filling techniques. Note

[5] It would be easy to extend the algorithm so that holes can be processed as 2D objects instead. This would allow more sophisticated filling methods such as image inpainting techniques [3]. But since the image rendering step is supposed to be done by the 3D display (and realized in hardware if possible), such techniques are considered too complicated to be applicable [6].

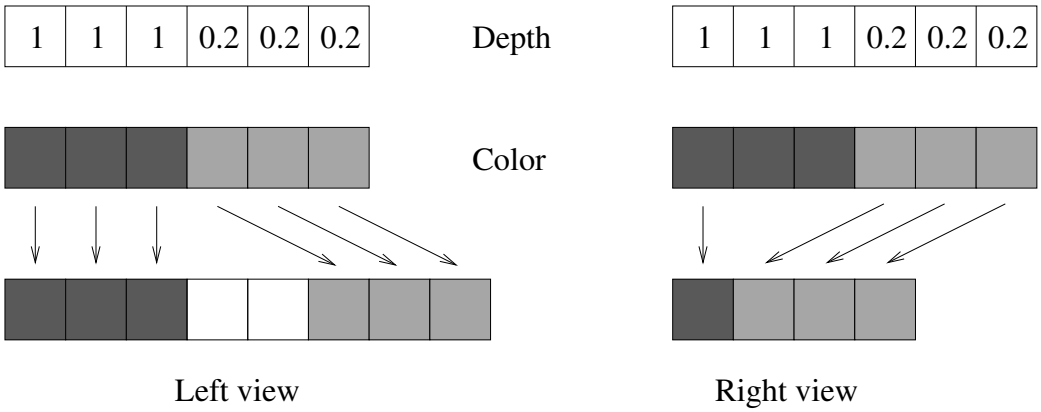

Figure 2.5: Occlusion (in the right view) and disocclusion (in the left view) at a depth discontinuity.

that the width of the holes was chosen to highlight the effects of each technique; disocclusion holes are usually smaller in real applications. The figure suggests that using the color of the far object is the best choice, but this is only true if depth discontinuities are perfectly aligned with object boundaries. This is usually not the case even with depth data from range sensors, and techniques to reduce disocclusion holes intentionally further reduce this alignment; see the next subsection.

Algorithm 2.1 describes the complete DIBR process.

2.2.3. Reducing disocclusions

Because of the restriction to very basic hole filling techniques, it is important to avoid wide disocclusion holes as much as possible, to reduce their negative effects on image quality. This gives a strong reason to choose $p = 0.5$ as the position of the camera C_c: the disocclusion holes are then equally divided between left and right view. This gives better results than having one view with no disocclusion holes at all and one view with wide holes, and outweighs the advantage of the positions $p = 0$ and $p = 1$, where only one virtual view has to be computed because the other is identical to I_c.

To further reduce the width of the holes, it is necessary to understand the cause for wide holes: equation 2.2 shows that far objects produce a small shift in the left and right view while near objects cause a big shift. A sudden change in the depth map from a far object to a near object increases the shift in the left view abruptly, thus causing a large hole. Similarly, a sudden change from a near object to a far object causes a large hole in the right view.

This leads to the idea to avoid sudden changes in the depth map by smoothing them. This can be done with a Gauss filter. Figure 2.7 shows the left views re-

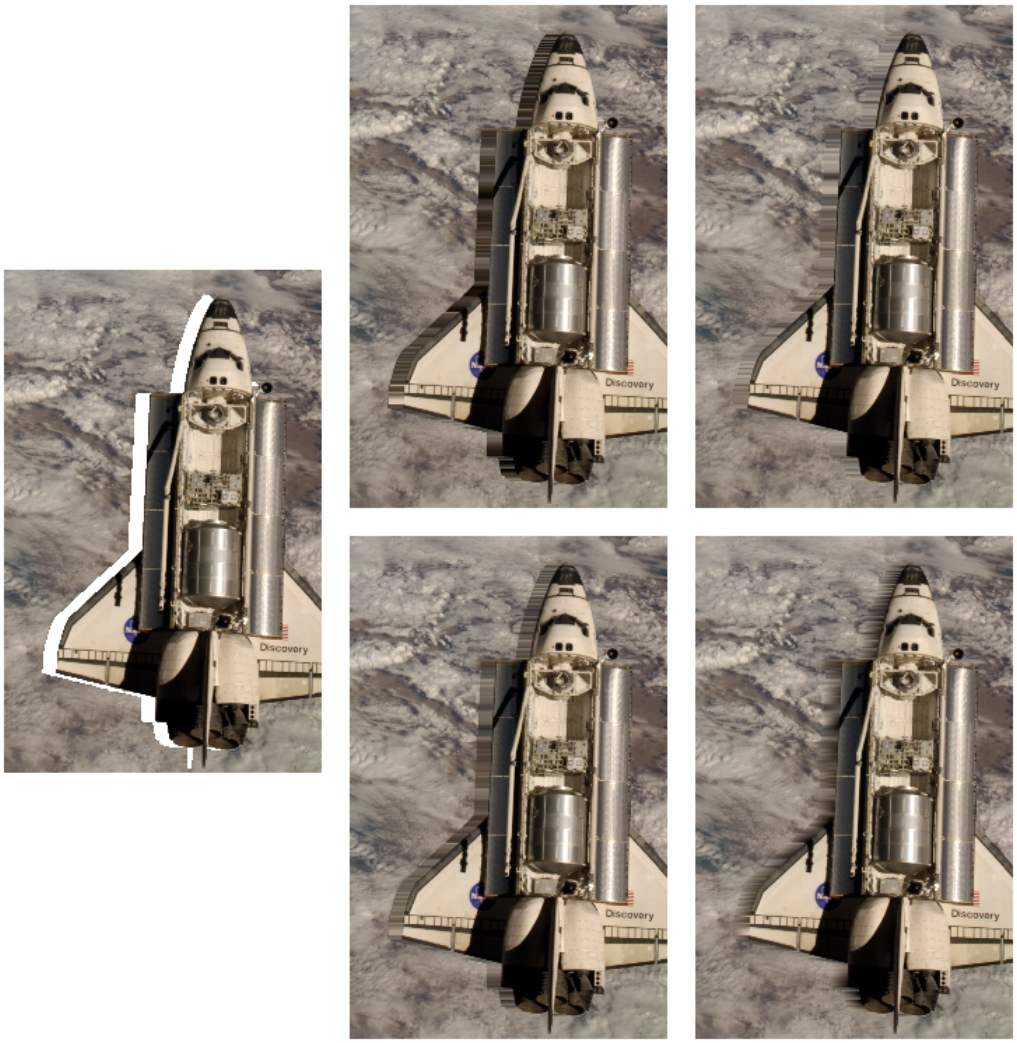

Figure 2.6: Different techniques to fill the disocclusion holes shown in the left-most image: color of the near object (top left), color of the far object (top right), average of both colors (bottom left), and linear gradient between both colors (bottom right).

Input: Image F of size $N \times M$
Depth map D of size $N \times M$
Output: Left view L and right view R

For $y = 0, \ldots, N-1$ {
  For $x = 0, \ldots, M-1$ {
    $L_d[x] = -1$; $R_d[x] = -1$
  }
  For $x_c = 0, \ldots, M-1$
    $d = 1 - ((D[x_c, y] + 1)/256)$
    $x_l = x_c + p\frac{bf}{d}$
    $x_r = x_c - (1-p)\frac{bf}{d}$
    If $L_d[x_l] < D[x_l, y]$ {
      $L[x_l, y] = F[x_c, y]$
    }
    If $L_d[x_r] < D[x_r, y]$ {
      $R[x_r, y] = F[x_c, y]$
    }
  }
  Fill the holes in L (where $L_d = -1$)
  Fill the holes in R (where $R_d = -1$)
}

Algorithm 2.1: Depth image based rendering.

sulting from depth image based rendering steps with different depth data. The disocclusion holes appear white because they were not filled. The holes are clearly much wider for the unsmoothed depth map. The finer holes caused by the smoothed depth map can be expected to have a lesser negative effect on image quality. At the same time, they are not aligned with the boundaries of the foreground objects anymore, so that the 3D accuracy of the view decreases. The impacts of these two effects were examined in several tests [16, 18]. The authors concluded that smoothing the depth maps can improve the image quality of resulting 3D video material while preserving the quality of perceived depth.

However, it was shown in [21] and, more detailed, in [20], that strong smoothing can lead to geometric distortions: depending on the depth in neighboring regions, vertically straight objects can become curved. See for example the table leg in the top row of figure 2.8: it has the same depth along its vertical length. In the smoothed depth map in the middle row, this is no longer the case, since the table leg lies next to near objects at its upper and lower ends but not in the middle. This causes the depth values in its middle to be larger than at its upper and lower ends.

Asymmetrical smoothing with emphasis on the vertical direction is proposed in [20] to reduce this problem, but it does not solve it entirely: while the table leg in the bottom row of figure 2.8 is clearly straighter than with symmetric smoothing, there are still geometric distortions in horizontal direction, for example at the lower end of the water dispenser in the background.

Note that figure 2.8 was created with parameters that result in very strong disparities, to emphasize the effects of smoothing. This is the reason for the strong rendering artifacts near to the heads and elsewhere in the image. In real applications, these artifacts are usually less visible.

In summary, it is advisable to choose the smoothing parameters so that disocclusion holes are reduced and the geometry of the scene is not damaged.

Figure 2.7: Reducing the size of disocclusion holes by smoothing the depth map.

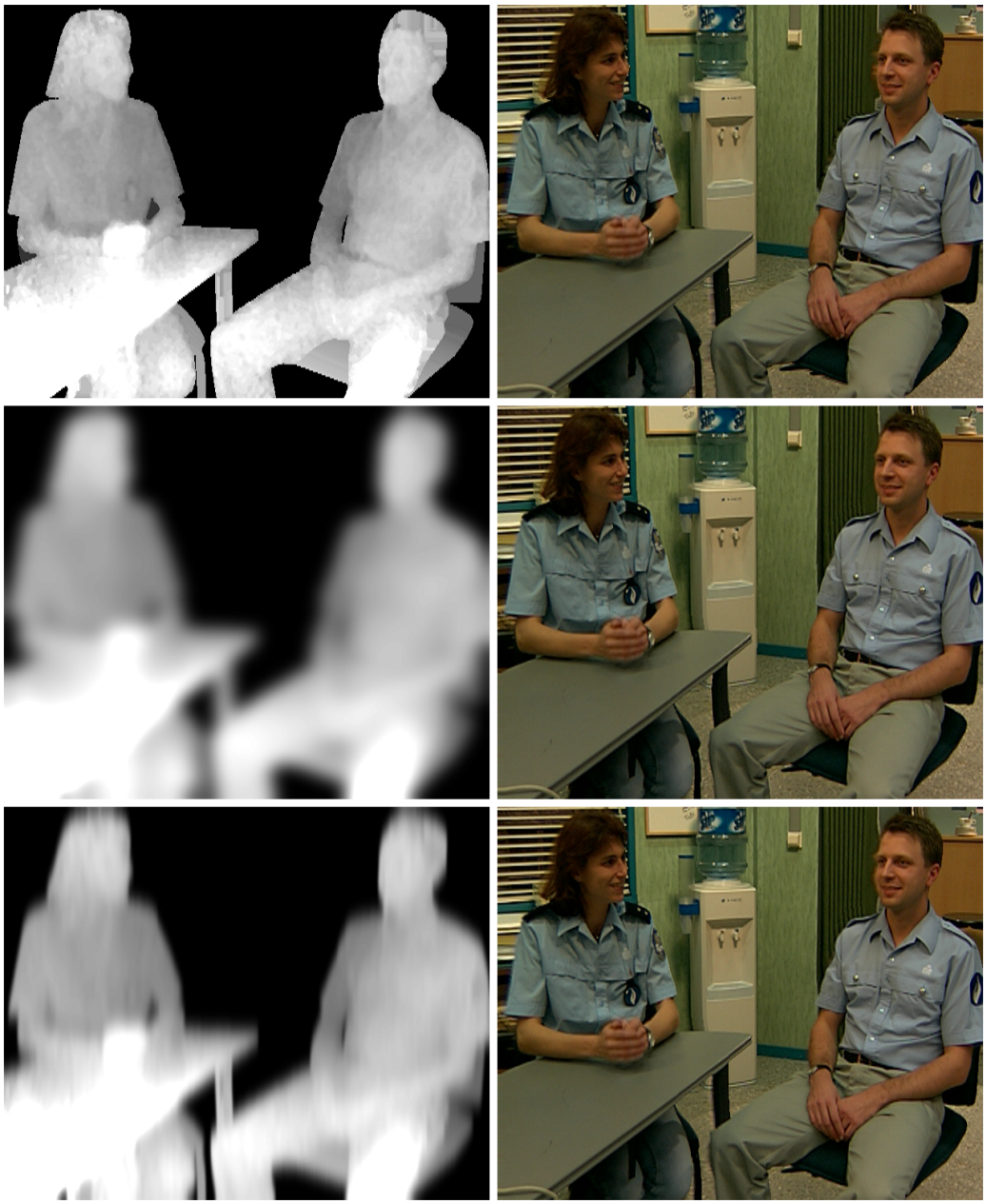

Figure 2.8: Geometric distortion caused by smoothing the depth map. Top row: original depth data. Middle row: symmetrically smoothed depth data. Bottom row: asymmetrically smoothed depth data.

3. Synthesizing depth maps

As outlined before, there are situations in which depth information is only available for a small number of video frames. For the purpose of this chapter, it is assumed that a 2D video scene is given by $n \geq 3$ frames $F_0, \ldots, F_{n-1}$, along with depth information D_0 and D_{n-1} for the first and last frame.

The goal is to expand the depth information from D_0 and D_{n-1} so that a complete set of depth maps $D_0, \ldots, D_{n-1}$ is available for a subsequent depth image based rendering step.

3.1. Depth tracking

The basic idea is to track each point in the scene as it moves to different pixel positions from frame to frame. For a point P at position $p_i \in F_i, 0 < i < n-1$, the corresponding positions $p_0 \in F_0$ and $p_{n-1} \in F_{n-1}$ are then known, and therefore also the associated depths d_0 and d_{n-1}. The depth d_i required for the unknown depth map D_i can then be computed by a linear interpolation of d_0 and d_{n-1}.

To be able to compute the position p_{i+1} in F_{i+1} of a point P with a given position p_i in F_i, it is necessary to know the *forward motion* vector $f_i(pi)$:

$$p_{i+1} = p_i + f_i(pi) \tag{3.1}$$

Similarly, the position p_{i-1} in F_{i-1} of the point can be computed when the *backward motion* vector $b_i(pi)$ is known:

$$p_{i-1} = p_i + b_i(pi) \tag{3.2}$$

To handle all points of the scene, the forward and backward motion vector fields $f_0, \ldots, f_{n-2}$ and $b_0, \ldots, b_{n-2}$ that contain motion vectors for each pixel position must be computed. It is not enough to know the motion vectors of only one direction, because the per-pixel motion between two frames is not bijective. The complete procedure is illustrated in figure 3.1.

If a point P with a given position p_i in F_i cannot be tracked to both F_0 and F_{n-1}, a linear interpolation of the depth d_i is not possible because of lack of information. If it can be tracked to one of both end frames, then the depth d_i is set

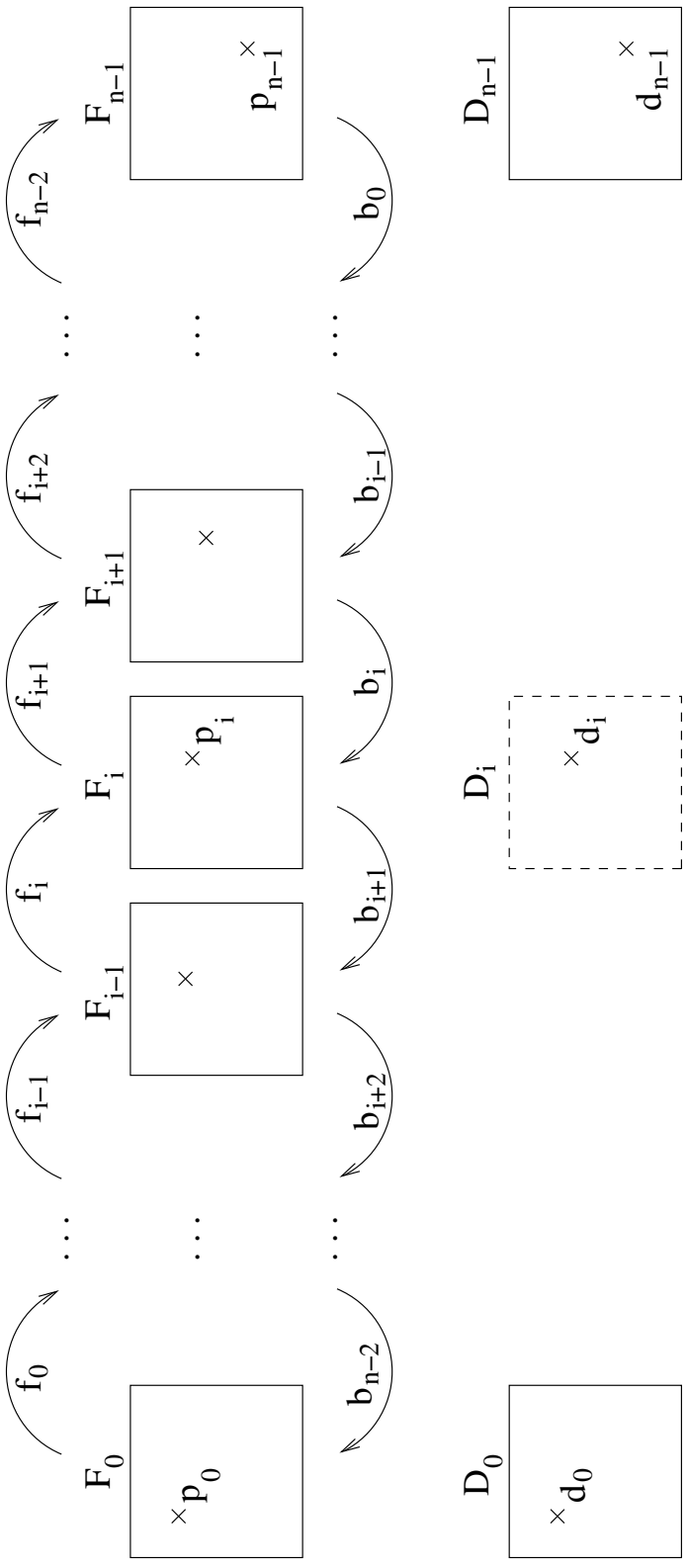

Figure 3.1: Depth tracking. By tracking the position p_i of the point P across the frames $F_0,\ldots,F_{n-1}$, it is possible to interpolate the depth d_i that it has in frame F_i from the depths d_0 and d_{n-1}.

to the depth at this end frame, thus assuming that it remains constant. If it can neither be tracked to F_0 nor to F_{n-1}, then the depth d_i is arbitrarily set to "far".

To avoid too frequent occurrences of such situations, it is necessary that the observed scene does not change too much, so that roughly the same objects are visible in F_0 and F_{n-1}, albeit at different positions. If this requirement is not fulfilled, the scene may have to be divided into sub-scenes.

The complete depth tracking method is described in algorithm 3.1.

If all forward and backward motion fields $f_i, b_i, 0 \leq i \leq n-2$ contain only zeros, then the depth tracking method is reduced to a simple linear interpolation between the depth maps D_0 and D_{n-1}. When the depth maps are stored as graylevel frames, this is equivalent to crossfading.

The key part of depth tracking is the per-pixel motion estimation: the more precise it is, the better the depth approximation. Two general approaches to compute motion vector fields are known: optical flow and block matching.

3.2. Optical flow

The per-pixel motion vector fields are in fact dense optical flow fields. Optical flow methods assume that changes in the lightness of pixels correspond to motion of objects in the scene. All methods presented in this section are based on the *image brightness constancy equation* (IBCE), which states that the brightness $f(x,y,t)$ of a point at position (x,y) at the time t remains the same when the point moves by the (small) vector (dx,dy) in the (short) time dt:

$$f(x,y,t) = f(x+dx,y+dy,t+dt) \tag{3.3}$$

A first order Taylor series expansion gives

$$\begin{aligned} f(x+dx,y+dy,t+dt) &= f(x,y,t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt + O(\partial^2) \\ &\equiv f(x,y,t) \end{aligned} \tag{3.4}$$

and therefore

$$\frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} = -\frac{\partial f}{\partial t} \tag{3.5}$$

or

$$f_x \cdot u + f_y \cdot v = -f_t \tag{3.6}$$

where f_x, f_y, f_t are discretizations of the partial derivatives, and (u,v) is the motion vector.

Equation 3.6 is not sufficient to compute the optical flow. Different optical flow methods use different additional constraints.

Input: n video frames $F_0,\dots,F_{n-1}$ of size $N\times M$
2 depth maps D_0, D_{n-1} of size $N\times M$
$n-2$ forward motion vector fields $f_0,\dots,f_{n-2}$ of size $N\times M$
$n-2$ backward motion vector fields $b_0,\dots,b_{n-2}$ of size $N\times M$
Output: $n-2$ depth maps $D_1,\dots,D_{n-2}$

```
For i = 1,...,n-2 {
    For y = 0,...,N-1 {
        For x = 0,...,M-1 {
            p_0 = (x,y)
            For j = n-i-1,...,n-2 {
                p_0 = p_0 + b_j(p_0)
                Check if p_0 is valid
            }
            p_{n-1} = (x,y)
            For j = i,...,n-2 {
                p_{n-1} = p_{n-1} + f_j(p_{n-1})
                Check if p_{n-1} is valid
            }
            If both p_0 and p_{n-1} are valid {
                D_i(x,y) = linear interpolation of D_0(p_0) and D_{n-1}(p_{n-1})
            } else if p_0 is valid {
                D_i(x,y) = D_0(p_0)
            } else if p_{n-1} is valid {
                D_i(x,y) = D_{n-1}(p_{n-1})
            } else {
                D_i(x,y) = 0
            }
        }
    }
}
```

Algorithm 3.1: Depth Tracking.

All optical flow methods based on the IBCE have to compute discretizations of the image derivatives in spatial and temporal dimensions. Therefore, the input images are usually smoothed with a gaussian filter to reduce the sensitivity to noise and improve the results. Recently, it became common to apply three-dimensional, spatiotemporal Gauss smoothing to video streams instead of simpler per-frame 2D Gauss smoothing. This way, the temporal dimension is not disadvantaged in comparison to the spatial dimensions, and the computation results improve accordingly [4]. In the 3D case, it is even more important than in the 2D case to utilize the separability of the Gauss filter to reduce computation costs.

3.2.1. Horn/Schunck

The Horn/Schunck algorithm [8] is a well known global optical flow approach that minimizes a global energy E_{HS}:

$$E_{\mathrm{HS}} = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} (f_x(x,y) + f_y(x,y) + f_t(x,y))^2 + \lambda \cdot \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} s(x,y) \tag{3.7}$$

The first term stems from the IBCE. The second term is a smoothness constraint derived from a discretization of the term $(|\nabla u(x,y)|^2 + |\nabla v(x,y)|^2)$:

$$\begin{aligned} s(x,y) = \frac{1}{4} \cdot [&(u(x+1,y) - u(x,y))^2 + (u(x,y+1) - u(x,y))^2 \\ &+ (v(x+1,y) - v(x,y))^2 + (v(x,y+1) - v(x,y))^2] \end{aligned} \tag{3.8}$$

Larger values of λ lead to smoother flow fields.

This minimization problem depends on all motion vectors (u,v). It can be solved with an iterative method [8]. The number s of iteration steps is a parameter.

The Horn/Schunck method is known to be susceptible to noise [1, 4]. See figure 3.2 for an example result. An advantage of this method is the *filling-in effect* [4]: in areas in which the local derivatives are near to zero, the second energy term fills in some neighborhood information, so that a dense flow field is computed.

3.2.2. Lucas/Kanade

The Lucas/Kanade algorithm [15] is a well known local approach. The additional constraint of this method is that the optical flow is constant in a small neighborhood of size $(2k+1) \times (2k+1)$ around (x,y):

$$\forall_{r,c:-k \le r,c \le k} \quad \begin{pmatrix} u \\ v \end{pmatrix}(x,y) = \begin{pmatrix} u \\ v \end{pmatrix}(x+c, y+r) \tag{3.9}$$

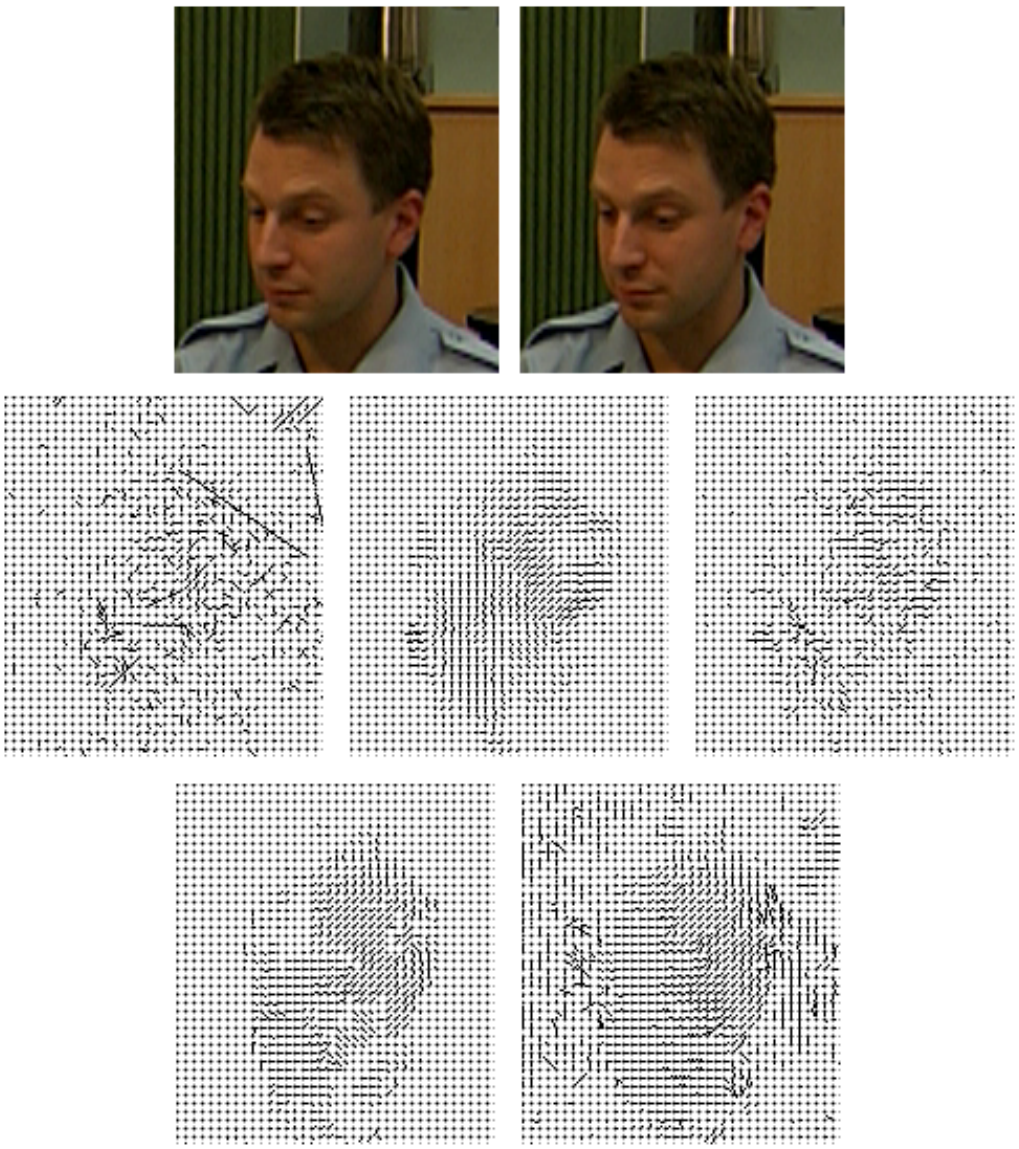

Figure 3.2: Example results of motion estimation techniques. Top row: the original frames. Middle row: Horn/Schunck, Lucas/Kanade, and the combined local/global method. Bottom row: SAD and ASW block matching.

From the IBCE equation 3.6 and this additional constraint, the following set of equations can be derived for the optical flow in each point (x,y):

$$\forall_{r,c:-k\leq r,c\leq k} \quad f_x(x+c,y+r)\cdot u(x,y) + f_y(x+c,y+r)\cdot v(x,y) = -f_t(x+c,y+r) \tag{3.10}$$

This system of equations can then be solved using the least squares method:

$$\begin{pmatrix} u \\ v \end{pmatrix}(x,y) = A^{-1}(x,y)\cdot -c(x,y) \tag{3.11}$$

with

$$A(x,y) := \begin{pmatrix} \sum_{r,c} f_x^2(x+c,y+r) & \sum_{r,c} f_x(x+c,y+r) f_y(x+c,y+r) \\ \sum_{r,c} f_x(x+c,y+r) f_y(x+c,y+r) & \sum_{r,c} f_y^2(x+c,y+r) \end{pmatrix} \tag{3.12}$$

and

$$c(x,y) := \begin{pmatrix} \sum_{r,c} f_x(x+c,y+r) f_t(x+c,y+r) \\ \sum_{r,c} f_y(x+c,y+r) f_t(x+c,y+r) \end{pmatrix} \tag{3.13}$$

The Lucas/Kanade method has the advantage to be quite robust to noise, and to generate smooth flow fields [1]. See figure 3.2 for an example result. A disadvantage is the lack of the filling-in effect of the Horn/Schunck method.

3.2.3. Combined local/global method

This approach tries to unify the Horn/Schunck and Lucas/Kanade methods to combine their strengths [4]. To that end, a new energy function is defined that contains both global and local terms.

First, the least square solver 3.11 for the Lucas/Kanade system of equations is reformulated as an energy function that must be minimized:

$$E_{\mathrm{LK}} = \sum_{y=0}^{N-1}\sum_{x=0}^{M-1} A(x,y)\cdot\begin{pmatrix} u \\ v \end{pmatrix}(x,y) + c(x,y) \tag{3.14}$$

Then a new energy function E_{CLG} is defined using E_{LK} (which includes the IBCE) and the smoothness constraint of E_{HS}:

$$E_{\mathrm{CLG}} = \sum_{y=0}^{N-1}\sum_{x=0}^{M-1}\left[\left[A(x,y)\cdot\begin{pmatrix} u \\ v \end{pmatrix}(x,y) + c(x,y)\right] + \lambda\cdot s(x,y)\right] \tag{3.15}$$

An iterative method to solve the resulting minimization problem is given in [4].

Figure 3.2 shows that the results indeed combine qualities of Horn/Schunck and Lucas/Kanade: the sensitivity to noise is not very strong, and the results are relatively smooth.

3.3. Block matching

Block matching techniques are commonly used in feature tracking applications and in stereo correspondence search: to find a match for the pixel at position p in frame F_0 at some position q in frame F_1, a block of size $(2k+1)\times(2k+1)$ around p is examined, and the best match for this neighborhood is searched in F_1.

If $\hat{q}$ is the position of the match candidate currently under consideration, then its matching costs are defined as

$$C(p,\hat{q}) = \sum_{r=-k}^{k}\sum_{c=-k}^{k} c(p+(r,c),\hat{q}+(r,c)) \tag{3.16}$$

The candidate position $\hat{q}$ with the lowest matching costs wins. The cost function c differs between various block matching variants.

In feature tracking, usually only one feature point is tracked [14] across several video frames, and in stereo correspondence search, the area to search a match in can be limited to one line, the epipolar line [14]. In the more general case of per-pixel motion estimation, a maximum distance d that p and q are allowed to have must be introduced, and the whole area defined by d must be searched for every pixel position p. Therefore, if the size of the video frames is $N\times M$, the computation costs for one motion vector field are $O(NMk^2d^2)$ if no special precautions are taken to reduce them.

The computation costs are not the only problem that stems from the wide search area: the uncertainty grows, too, since more candidates have to be considered. This may increase the error rate.

Block matching approaches search an ideal match for every single pixel. Depending on the block matching variant, its parameters, and the image material, this can lead to motion vector fields with a very high precision, but it can also lead to inconsistent (non-smooth) fields.

3.3.1. Sum of absolute differences

This approach, known from both stereo analysis and feature tracking, uses the absolute difference of pixel values as the cost function c in equation 3.16:

$$c_{\mathrm{SAD}}(p,q) = |F_0(p) - F_1(q)| \tag{3.17}$$

This pixel value difference does not need to be restricted to brightness values.

The YUV color space is widely used in video processing applications. The Y component represents the brightness of a point, and the U and V components

define its hue and saturation[1]. Thus, the following term can be used:

$$\begin{aligned} c_{\mathrm{SAD}}(p,q) = {} & L \cdot |Y(F_0(p)) - Y(F_1(q))| \\ & + (1-L) \cdot \frac{1}{2} \cdot (|U(F_0(p)) - U(F_1(q))| + |V(F_0(p)) - V(F_1(q))|) \end{aligned} \tag{3.18}$$

Each of the components Y, U, V is expected to be in $[0,1]$ in this equation. L is the luminance weight: it determines how much influence luminance differences should have in comparison to color differences.

To introduce the necessary cost reduction and simultaneously reduce the uncertainty of the method, the SAD block matching variant for depth tracking uses the following matching cost function, which is an extension of equation 3.16:

$$\begin{aligned} C_{\mathrm{SAD}}(p,\hat{q}) = {} & D \cdot \frac{\mathrm{distance}(p,\hat{q})}{k} \\ & + (1-D) \cdot \sum_{r=-k}^{k} \sum_{c=-k}^{k} c_{\mathrm{SAD}}(p+(r,c), \hat{q}+(r,c)) \end{aligned} \tag{3.19}$$

The additional term is a distance penalty: larger motion vectors cause higher matching costs. The parameter D determines the balance between the distance penalty and the sum of absolute pixel value differences. It should be small (≤ 0.1), because large distance penalties corrupt the results.

The introduction of a distance penalty for the candidate position $\hat{q}$ allows to shortcut the computation of C_{SAD}: if motion vectors $\hat{q} - p$ are tested in the order of ascending length (see figure 3.3), then the full costs C_{SAD} do not have to be computed if the distance penalty alone is already greater than the total costs of previously tested vectors. This is frequently the case in background areas of scenes: since no object moves, a very good match with C_{SAD} near zero is already found at a motion vector near $(0,0)$. Motion vectors tested at a later time then exceed these very low costs already because of their length, so the sum of differences does not need to be computed. The distance penalty also reduces the uncertainty, for example in areas with periodic textures, where good matches are found at multiple positions.

The SAD block matching method gives good results in practice. Again, an example result is shown in figure 3.2.

3.3.2. Adaptive Support-Weights

Not all pixels inside a block are equally well suited to determine the best match: some may belong to foreground objects and some to the background. At such

[1] Though not in the direct way as for example the HSL color space does.

	−2	−1	0	+1	+2
−2	22	18	12	19	23
−1	14	6	4	7	15
0	10	2	1	3	11
+1	16	8	5	9	17
+2	24	20	13	21	25

Figure 3.3: Order in which motion vectors are tested in the SAD block matching variant for depth tracking. This example shows the order of the 25 vectors that result from a maximum allowed distance $d = 2$.

depth discontinuities, the matching block should be selected for each pixel adaptively, to respect the higher importance of foreground pixels. This can be done in one of two ways: choosing one of several block templates of different size and shape, or assigning weights to each position in a fixed (large) block.

The adaptive support-weight method [19] uses the more general second approach. It is inspired by observations on the human visual system: the importance of a pixel in a block is computed based on its color difference and distance to the pixel at position p for which a match is currently searched.

To resemble the perception of the human visual system, the color difference is computed in the CIELab color space, and the euclidean distance d_{euc} is used. Then, the weight of a pixel pair inside the same block is defined as

$$w(p,q) = \exp\left(-\left(\frac{d_{\text{CIELab}}(F_0(p),F_1(q))}{\gamma_c} + \frac{d_{\text{euc}}(p,q)}{\gamma_d}\right)\right) \tag{3.20}$$

The parameters γ_c and γ_d determine the balance between color difference and distance.

The weights are then used to compute the matching costs C_{ASW}:

$$\begin{aligned} C_{\text{ASW}}(p,\hat{q}) = &\left(\sum_{r=-k}^{k}\sum_{c=-k}^{k} w(p,p+(r,c))\cdot w(\hat{q},\hat{q}+(r,c))\right)^{-1} \\ &\cdot \sum_{r=-k}^{k}\sum_{c=-k}^{k} w(p,p+(r,c))\cdot w(\hat{q},\hat{q}+(r,c))\cdot c_{\text{raw}}(p+(r,c),\hat{q}+(r,c)) \end{aligned} \tag{3.21}$$

The raw matching costs c_{raw} are the same sum of YUV pixel value differences that the SAD method uses.

The ASW block matching method is very successful in finding correspondences in stereo image pairs[2]. For per-pixel motion estimation, the increased uncertainty and computation costs have to be dealt with. An additional distance penalty, like the one used for the SAD method, cannot be introduced here: it would interfere with the concept of perceptual similarity of pixels. Instead, a pyramid method is used: initial estimations of motion are computed in a low resolution. These estimations are then used to initialize the motion estimation of the next step that works with a higher resolution, thereby reducing the search radius. A drawback is that errors from the first, low-resolution step might propagate.

The pyramid reduces both computation costs and uncertainty, but the effects of the latter cannot be eliminated completely: when applied to stereo image pairs, the pyramid based version does not match the original version that searches correspondences on a single line only.

An example result is shown in figure 3.2.

3.4. Consistency check

A postprocessing step similar to the one presented here is often used in stereo correspondence search [14], but it is also applicable to motion vector fields for depth tracking, regardless of the motion estimation method they were created with.

To catch unreliable motion vectors, the motion estimation is done in both directions: from F_0 to F_1, leading to the vector field f, and from F_1 to F_0, leading to the vector field b. The reliability of a motion vector v in f will be high if the corresponding motion vector in b points back to the position of v or near to it. See figure 3.4. A threshold t determines the maximum allowed difference for vectors to be considered reliable. If the difference is greater, the vector v in f is marked as unreliable. In a second step, all unreliable vectors in f are replaced by interpolating neighboring reliable vectors. This is done in a way that ensures that vectors with a high number of reliable neighbors are replaced first, to avoid propagating errors as much as possible. The result is an improved vector field f^*. See the example in figure 3.5.

By swapping the roles of f and b, the same can be done with b, leading to the improved field b^*. Since the motion estimation for depth tracking has to be done in both directions anyway, the consistency check imposes little additional computation costs.

[2] See for example the ranking on `http://bj.middlebury.edu/~schar/stereo/newEval/php/results.php`

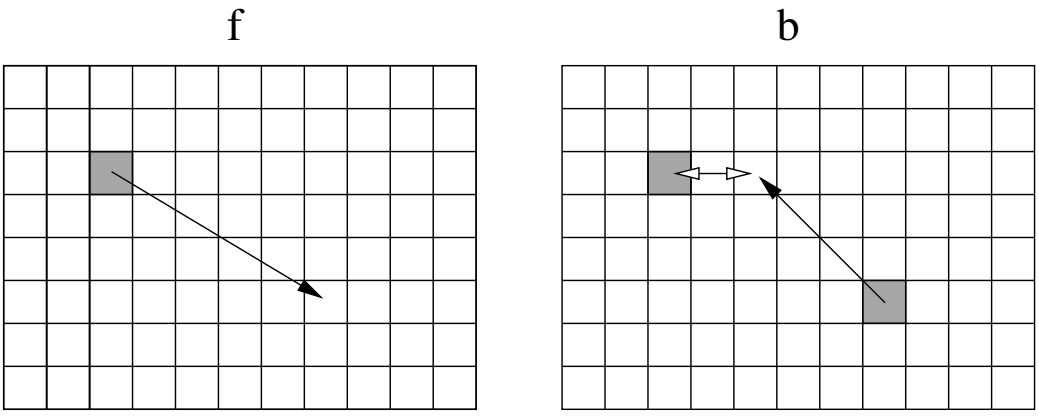

Figure 3.4: Consistency check for per-pixel motion estimation methods.

Figure 3.5: Consistency check applied to motion estimation vector fields from the combined local/global method. Bottom left: no consistency check. Bottom right: consistency check with tolerance $t = 1$.

4. Experimental results

4.1. Test material

Three example video scenes were used:

1. *Interview:* This scene shows a man and a woman sitting at a table and having a chat. It is roughly ten seconds long (251 frames) and has a resolution of 720×576 pixels, which corresponds to the european PAL TV standard.

2. *Orbi:* In this scene, a camera moves around a table with various sorts of toys. Its length is five seconds (126 frames), and its resolution is 650×540.

3. *Nasa:* This scene shows an astronaut working on a space shuttle, with the earth in the background. It is eighteen seconds long (451 frames). The resolution is 528×360.

The scenes *Interview* and *Orbi* have known depth maps for each video frame, recorded with a real-time depth sensor. See figure 4.1. They were kindly provided by Fraunhofer Institute for Telecommunications, Berlin. The real depth data can be used as Ground Truth when evaluating computed depth data.

The scene *Nasa* is a part of a NASA mission video[1]; see figure 4.2. It is a conventional 2D video without depth data. Three depth maps were created manually: one for the first frame, one for the middle frame, and one for the last frame. This was done by assigning one of three depth steps to the objects in the scene: "very near", "near", and "middle". In the first depth map, a gradient between "very near" and "near" was used for one part of the shuttle that points away from the viewer. See the first depth map in figure 4.2. Since this part of the shuttle moves out of the visible area, no gradient was necessary for the second and third depth map. Each depth map additionally contains a sphere in the "far" range representing the earth. Since these "far" distances are represented by dark shades of gray, the sphere may not be visible in figure 4.2.

[1] `http://spaceflight.nasa.gov/gallery/video/shuttle/sts-114/qtime/114_fdh05_clip3.mov`

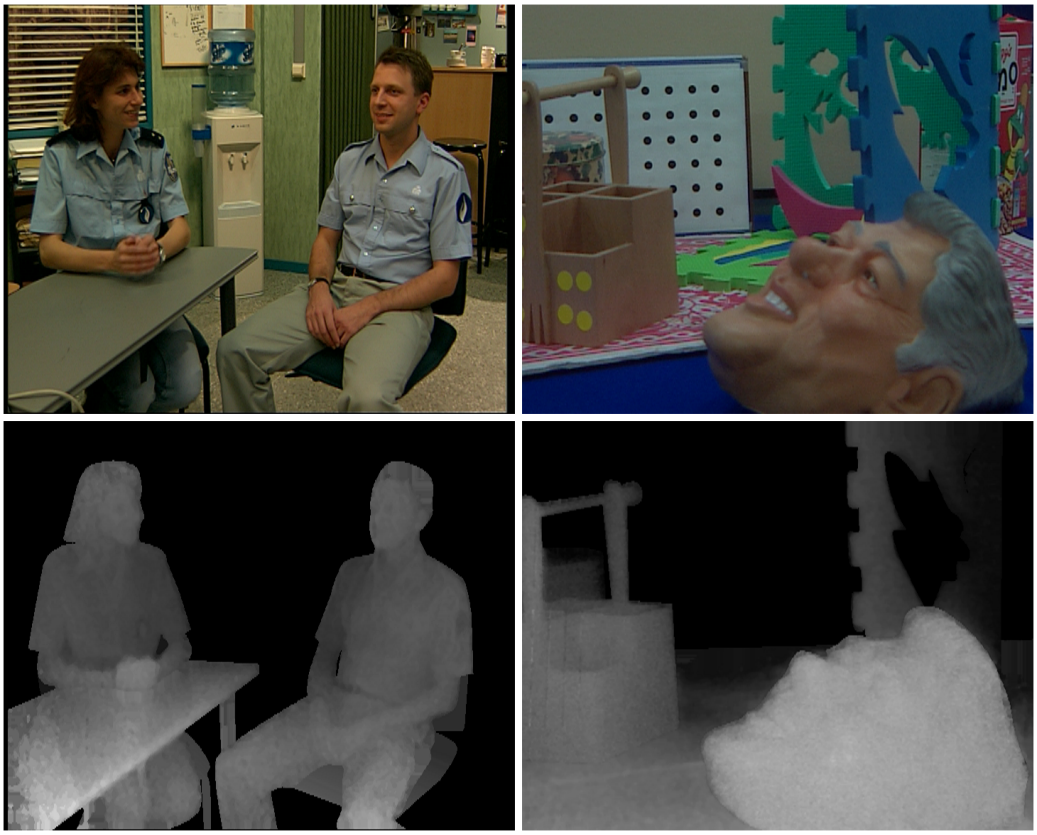

Figure 4.1: Video scenes *Interview* and *Orbi* with real depth data.

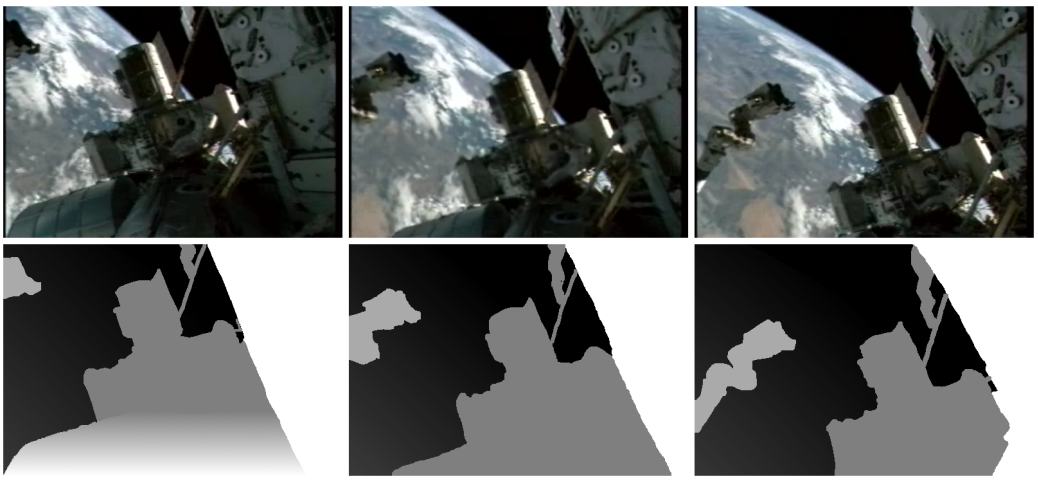

Figure 4.2: Video scene *Nasa* with artificial depth data for the first, middle, and last video frame.

4.2. Depth tracking

The three optical flow methods and the two block matching methods presented in chapter 3 were tested with various parameter sets to find the best per-pixel motion estimation method for depth tracking.

To this end, the frames $F_0, \ldots, F_{250}$ (ten seconds) of the *Interview* scene were extracted, along with real depth data $D_0, D_{25}, \ldots, D_{250}$ for every 25th frame. The missing depth maps were computed using depth tracking with the motion estimation method under examination.

Figure 4.3 shows the depth map D_{12} for *Interview* produced with the five motion estimation methods. The effects of the individual advantages and disadvantages of each method are visible:

- The Horn/Schunck method tends to create speckled depth maps. This is a result of its sensitivity to noise.
- The Lucas/Kanade depth map is much smoother, but the motion of objects does not seem to be followed correctly. For example, around the head of the man, the effects of linear interpolation outweigh the effects of motion estimation; the result looks similar to simple crossfading.
- The depth map of the combined local/global method combines the smoothness of the Lucas/Kanade depth map with the higher motion estimation quality of the Horn/Schunck depth map.
- The SAD block matching method seems to deliver the best motion estimation, resulting in the most accurate depth map. Around the head of the man, for example, only small effects of linear interpolation are visible; for the most part, its motion was followed correctly.
- The ASW block matching depth map shows a better motion estimation than the depth maps from optical flow methods, but it also shows regions with high errors and/or speckles.

The differences between Ground Truth and the computed depth maps can be used as an error measurement:

$$E_{\mathrm{DT}} := \frac{\sum_{i=0}^{n-1} \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} |D_i(x,y) - D_{\mathrm{GT}(x,y)}|}{nNM} \tag{4.1}$$

This average depth error value is in $[0, 255]$ (the smaller, the better). Figure 4.4 shows the results for the complete series of depth maps from which the examples in figure 4.3 where taken. The depth maps with the lowest errors according to this

Figure 4.3: *Interview* depth maps D_{12} from depth tracking with different motion estimation methods. Top row: Ground Truth and Horn/Schunck. Middle row: Lucas/Kanade and combined local/global. Bottom row: SAD and ASW block matching.

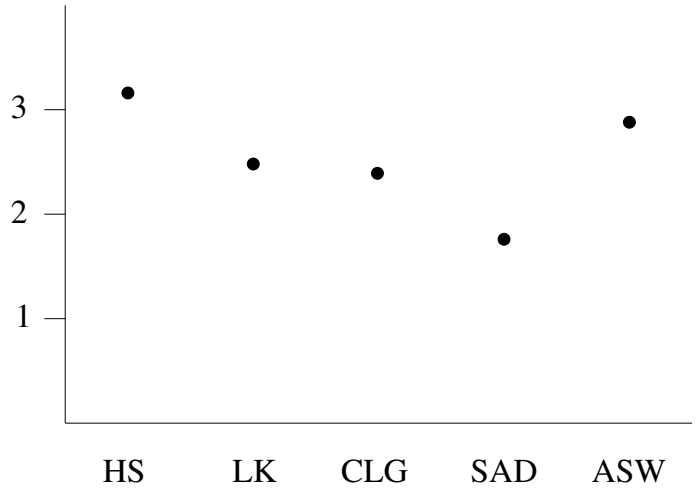

Figure 4.4: Error values for all *Interview* depth maps created with the motion estimation methods Horn/Schunck, Lucas/Kanade, Combined Local/Global, SAD block matching, and ASW block matching.

measurement are the ones from the SAD block matching method. This confirms the first impressions from figure 4.3.

The scene *Interview* poses some problems for motion estimation algorithms in general:

- Fuzzy object boundaries:
  The scene suffers from inappropriate illumination, which leads to fuzzy object boundaries; see for example the surrounding of the head in figure 4.5.

- Blurred motion:
  Moving objects are blurred in the video. This is the same effect that results from too long exposure times in photography. See figure 4.6.

- Fast motion:
  Some objects, like the shaking hands, move so fast that the shift in position between two consecutive frames is too high for current motion estimation algorithms. See figure 4.7.

No known optical flow or block matching method handles these types of problems well. This leads to the weaknesses observed in the depth maps in figure 4.3.

The *Orbi* video was apparently recorded under conditions similar to those that the *Interview* scene was subject to. Therefore, the motion estimation methods have similar problems, and the computed depth maps show the same weaknesses.

The *Nasa* scene suffers from artifacts introduced by heavy video compression, but it is nevertheless considerably better suited for motion estimation. This is

Figure 4.5: Fuzzy object boundaries in the *Interview* scene.

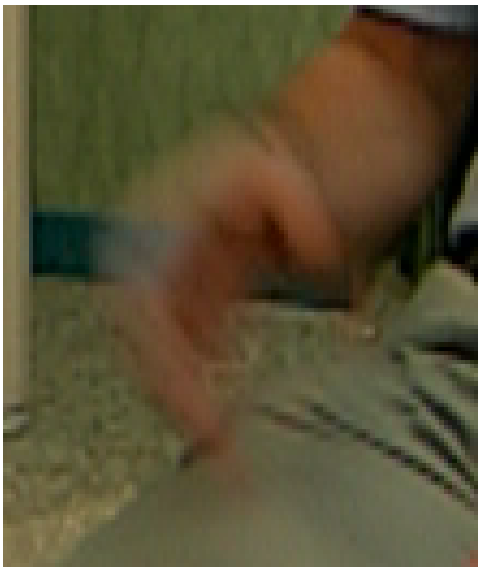

Figure 4.6: Blurred motion in the *Interview* scene.

Figure 4.7: Fast motion in the *Interview* scene: the hands move by roughly 40 pixels between these two consecutive frames.

mostly because the motion of objects is slow enough to only cause shifts of a few pixels between consecutive frames. Since only three hand-drawn depth maps are available, the distance of two known depth maps is 225 frames (nine seconds). Figure 4.8 shows the depth maps D_{112} that result from the different motion estimation methods when the depth maps $D_1, \ldots, D_{224}$ are computed. Note that the visible area in the *Nasa* scene changes because of camera panning. This leads to points that cannot be tracked to both F_0 and F_{225} because they are only visible in one of them. In addition to motion estimation errors, these points suffer from incomplete depth information. The results are as follows:

- The Horn/Schunck and Lucas/Kanade methods clearly fail to follow the object motion across all 226 frames.
- The combined local/global method performs much better, but the effects of linear interpolation are visible: obviously, the motion estimation was not correct for parts of the scene.
- The block matching methods manage to follow object boundaries quite precisely across the relatively long distance of 226 frames. The ASW variant suffers from some noise, but the SAD variant gives good results.

Since no real depth data is available, computed depth maps cannot be compared to Ground Truth, and no error measurement is available.

4.3. Quality rating

Five 3D video variants of each of the scenes *Interview*, *Orbi*, and *Nasa* were computed from different depth data, and prepared for display on the 2018XLQ

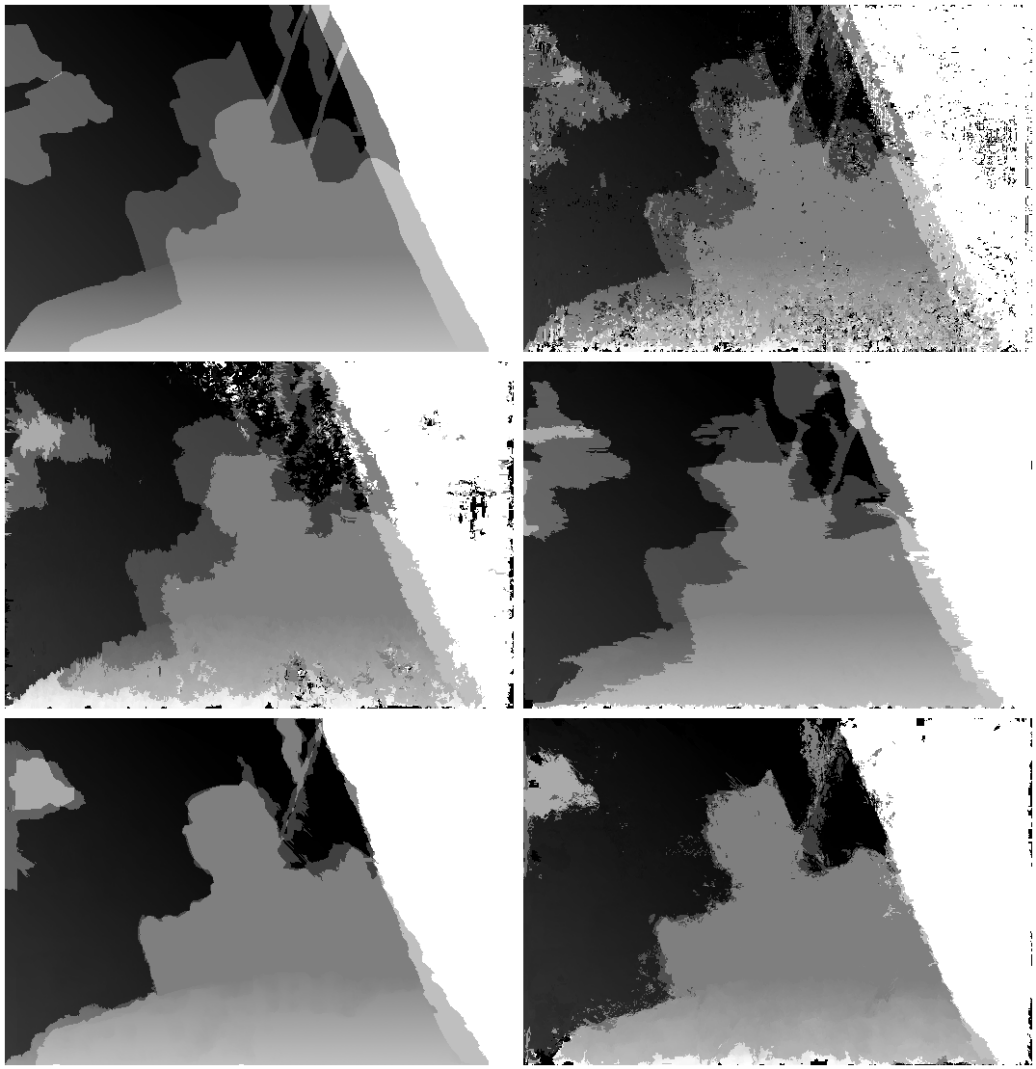

Figure 4.8: *Nasa* depth maps D_{112} from depth tracking with different motion estimation methods. Top row: Crossfading and Horn/Schunck. Middle row: Lucas/Kanade and combined local/global. Bottom row: SAD and ASW block matching.

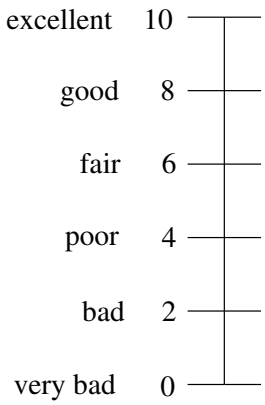

Figure 4.9: Rating scale.

19" 3D monitor from DTI[2]. For each scene, the order of the five variants was randomized.

A group of 10 test viewers was asked to rate both the image quality and the quality of the 3D effect of each variant, on a scale from 0 ("very bad" or "nonexistent") to 10 ("excellent"), as shown in figure 4.9. A note handed out to each test viewer clarified that image quality means the absence of noise and distortion in the image, and 3D effect quality means the impression of real depth. The viewers did not have any experience with 3D videos, and they did not know anything about the nature of the videos and their variants.

The following five variants were tested:

1. *2D:* The original 2D video. Presenting this variant allows to measure the impact that depth image based rendering has on image quality.

2. *Ground Truth:* A 3D video based on real depth maps. This variant is expected to show the best results in terms of 3D effect quality. For the *Nasa* scene, a surrogate for this variant was used due to the lack of real depth data.

3. *Depth Tracking:* A 3D video based on depth data that was computed using the depth tracking method. For *Interview* and *Orbi*, every 25th depth map from Ground Truth was used as initial depth data. For *Nasa*, the three artificial depth maps were used.

4. *Crossfade:* A 3D video computed from the same initial depth data used for the *Depth Tracking* variant. The missing depth maps were created by

[2] `http://www.dti3d.com/`

crossfading the available depth maps, which is equivalent to depth tracking when all motion vectors are $(0,0)$.

5. *Depth from edges:* A 3D video computed from depth data acquired with the "depth from edges" method. This method [17] creates pseudo-depth maps from the edge information in the scene: each 2D frame is processed with the Canny edge detector (Gauss smoothing, Sobel edge detection, non-maximum suppression, Hysterese binarization). The resulting binary edge maps are then interpreted as depth maps and smoothed with a Gauss filter. The smoothing is stronger in vertical than in horizontal direction. This asymmetry is expected to improve the quality of the resulting 3D video [17].

   In [17], this method was applied to single images only. When applying it to videos, an additional problem arises: small changes between consecutive frames may cause the binary edge lines to move by one or two pixels. This results in flicker in the depth maps, and therefore also in the 3D video. To reduce this effect, the asymmetric spatial Gauss smoothing was replaced by an asymmetric spatiotemporal Gauss smoothing. The strength of the smoothing in temporal direction was chosen so that it reduces the flicker but does not blur moving objects too much.

For the *Interview* and *Orbi* scenes, the motion estimation method used for depth tracking was the SAD block matching method, with parameters $L = 0.5$, $D = 0.1$, and $k = 2$. d was 50 for *Interview* and 12 for *Orbi*. An additional consistency check with tolerance $t = 2$ was applied.

For the *Nasa* scene, depth tracking was done with the SAD block matching method with parameters $D = 0.01$, $L = 0.5$, $k = 8$, and $d = 5$. Since no real depth data was available, the *Ground Truth* variant could not be tested. To compensate, the *Depth from edges* variant was presented two times, so that the *Nasa* scene was no exception to the test viewers.

The 3D videos were then produced using the following steps:

- Smoothing of the depth maps with a Gauss kernel of size 7×7.
- Depth image based rendering with parameters $p = 0$, $f = 1$. b was chosen for each scene separately: $b = 20$ for *Interview*, $b = 30$ for *Orbi*, and $b = 15$ for *Nasa*. These different values for b were simply chosen so that the result "looked good"; no automatic way to determine the best value for b is known. Disocclusion holes were filled with the "average color" technique.

The rating results are shown in figure 4.10.

Comparing the rating of the *2D* variant with the results of other variants shows that the image quality always suffers a little from depth image based rendering.

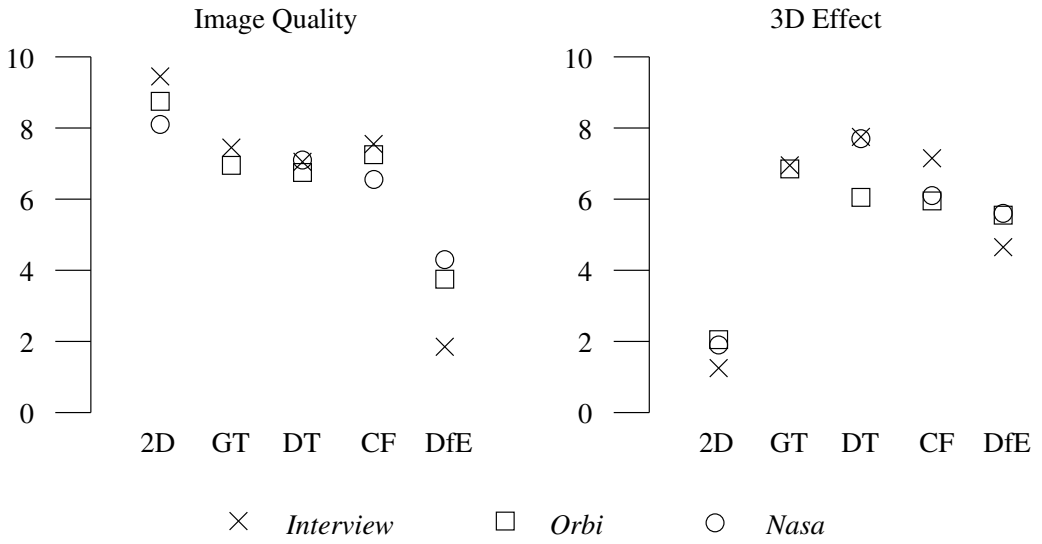

Figure 4.10: Rating of image quality and 3D effect for the variants *2D*, *Ground Truth*, *Depth tracking*, *Crossfading*, *Depth from edges* of the video scenes *Interview*, *Orbi*, and *Nasa*.

This effect may be reducible by choosing better parameters for the rendering step, but this was not subject of the test.

Most viewers noticed the absence of any 3D effect in the *2D* variant, but others still rated it with up to 5 points ("fair"). Also, it is interesting to note that the two identical *Depth from edges* variants of *Nasa* were rated with differences of $1-2$ points by almost all viewers. This, and the low number of test viewers, suggests that the end results should be interpreted with care: they are subject to strong fluctuations.

The low image quality rating for the *Depth from edges* variant is probably caused by the effect of flickering binary edges as explained above. If a way could be found to reduce this flicker more efficiently, then the *Depth from edges* results would probably be nearer to that of the other 3D variants, and would overall be rated in the "fair" range, although the underlying depth data is the most unnatural of all. This is consistent with the single-image results from [17].

For *Interview* and *Orbi*, the results of both *Depth Tracking* and *Crossfading* are roughly the same as those of *Ground Truth* – a better result cannot be expected. Although the *Depth tracking* variant uses depth maps that are closer to the original depth data than those of the *Crossfading* variant, this does not seem to lead to a difference in perception results.

For *Nasa*, the results of *Depth Tracking* are the best. The *Crossfading* depth data of *Nasa* contains greater relative errors than that of *Interview* and *Nasa*, because the distance between two available depth maps is greater (225 frames in-

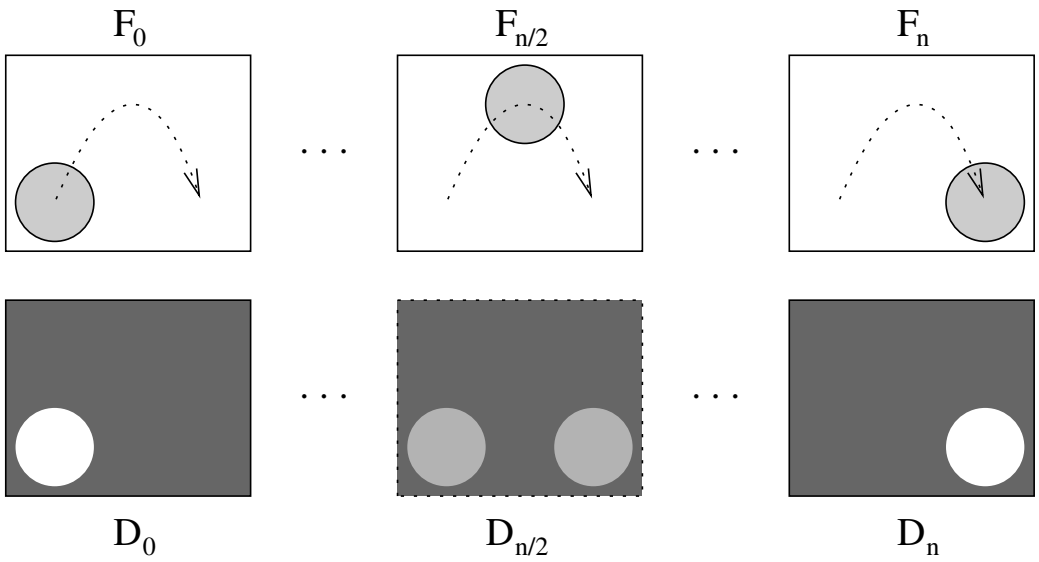

Figure 4.11: Expected errors of the crossfading method when estimating the depth map $D_{n/2}$ when n is big and the object movement is non-linear.

stead of 25). In addition, the object movements in *Interview* and *Orbi* are either

- small (the heads in *Interview*), so that the crossfading errors are small in general,
- fast (the shaking hands in *Interview*), so that viewers do not notice errors, or
- linear (the dominant movement in *Orbi*), so that crossfading does not make too much errors.

With growing distance between known depth maps, and with non-linear moving objects in the scene, the difference between depth tracking and crossfading grows, because crossfading inescapably produces errors in such situations. A hint of this can be seen in the *Nasa* results. See figure 4.11 for a sketchy example of a video scene for which crossfading would fail to produce acceptable depth maps.

5. Conclusion

Depth Tracking is a method suitable to expand depth information from sparse depth maps for a whole video scene. The quality of the results mainly depends on the per-pixel motion estimation method. This determines the applicability of the method:

- If non real-time range sensors can be expected to deliver at least one depth map per second, then the remaining holes can be filled using depth tracking even if the 2D video material is problematic for current motion estimation methods. This is a conclusion from the *Interview* scene example: the 2D video material is clearly not optimal for motion estimation, but the results were nevertheless rated positively by test viewers.

- For 2D video scenes that fulfill the requirements of current motion estimation methods to some extent, it is possible to enhance the 2D video with a 3D effect with relatively little manual work: for the *Nasa* scene with a length of 18 seconds, only three simplistic hand-drawn depth maps were necessary for good results, even though the visible area of the scene changes due to camera panning.

  While current Depth-from-X techniques are not able to generate suitable depth maps automatically, they can still speed up the process of manual depth map creation by giving a good start point.

  Parts of the scene that cause difficulties for the motion estimator can be detected with the consistency check from section 3.4: a warning can be issued if the number of unreliable motion vectors is above a certain threshold. The user can then provide additional depth maps for these specific parts of the video, to reduce the errors of depth tracking. Parts of the video for which the motion estimation is more reliable need less initial depth maps, because depth tracking will make less errors.

In the context of depth tracking, the following qualities are desirable for improved motion estimation methods:

- An increased per-pixel precision and reliability.

- The ability to handle object movement that spans more than a few pixels between two frames.
- Lower computation costs, especially for block matching approaches.[1]

To achieve these goals, it may be necessary to take information from all video frames into account and estimate the motion of objects across the whole scene, instead of processing only two frames at a time.

A motion estimation method with above properties would be able to reduce the number of required depth maps significantly while still allowing better depth tracking results. This would expand the applicability of depth tracking, especially for enhancing existing 2D video material with 3D effects.

[1] One approach to reduce the computation costs that does not depend on the motion estimation method would be to reduce the resolution of the depth maps: the tests presented in chapter 4 as well as in [16] and [17] indicate that the human visual system does not need depth information with pixel precision to perceive a 3D effect.

6. Implementation

6.1. Source and documentation

All algorithms were implemented in C as defined by the ISO/IEC 9899:1999 standard, also known as C99.

The image manipulation algorithms are implemented in the CVL library. A command line front end for this library, cvtool, allows easy access to the functionality provided by CVL. The source code uses the conventions of GNU[1] software: usually, the steps

```
$ ./configure
$ make
$ make install
```

are necessary to compile and install both the CVL library and cvtool. The GNU Scientific Library[2] provides numerical algorithms used by CVL; it is required to compile the source code. A complete API documentation for CVL is generated automatically and will be installed together with the rest of the software.

The CD-ROM contains source code, documentation, and binaries for Solaris (sparc64), GNU/Linux (x86), and Windows. The binaries are statically linked to avoid a dependency on the external GSL library.

6.2. Using cvtool

Cvtool is designed to be used as a filter: it reads images from standard input and writes its results to standard output. This allows easy programming of complex tasks with shell scripts or scripting languages such as Perl or Python. Both the NetPBM[3] image formats (pbm, pgm, ppm) and the YUV4MPEG2 video format used by the mjpegtools[4] are supported.

[1] http://www.gnu.org/
[2] http://www.gnu.org/software/gsl/
[3] http://netpbm.sourceforge.net/
[4] http://mjpeg.sourceforge.net/

Cvtool expects a command name as its first argument. All further arguments are options to that command. For example,

```
$ cvtool blur --method gauss -x 3 -y 3
        < input.y4m > output.y4m
```

would blur each frame in the video `input.y4m` with a Gauss filter of size 3×3 and write the output to the video `output.y4m`.

A list of available commands can be printed with

```
$ cvtool help
```

Detailed information about a command `cmd` and its options is given by

```
$ cvtool help cmd
```

Commands that are of particular interest with respect to this work are

- `blur`: Spatial and spatiotemporal blurring
- `edge`: Edge detection for depth from edges
- `opticalflow`: Various per-pixel motion estimation methods
- `trackdepth`: Depth tracking
- `dibr`: Depth image based rendering
- `stereoview`: Preparing stereoscopic images or videos for different kinds of displays.

From a source video file `video.y4m`, containing 100 frames, and the two depth maps `d00.pgm` and `d99.pgm` for the first and last frame, a stereoscopic 3D video can be produced and prepared for display on a DTI 3D monitor with the following commands:

```
$ cvtool opticalflow -m bm-sad -k 2 -M 50 -D 0.1 -L 0.5
        < video.y4m > flow-f.txt
$ cvtool reverse < video.y4m > oediv.y4m
$ cvtool opticalflow -m bm-sad -k 2 -M 50 -D 0.1 -L 0.5
        < oediv.y4m > flow-b.txt
$ cvtool trackdepth -n 100 -f flow-f.txt -F flow-b.txt
        -d d00.pgm -D d99.pgm > depth.pgm
$ cvtool blur -m gauss -x 3 -y 3
        < depth.pgm > depth-gauss.pgm
$ cvtool dibr -d depth-gauss.pgm -t 20
        < video.y4m > stereovideo.y4m
$ cvtool stereoview 3d-display -f tb
        < stereovideo.y4m > stereovideo-dti.y4m
```

Bibliography

[1] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of Optical Flow Techniques. *International Journal of Computer Vision*, 12(1):43–77, February 1994.

[2] R. Bates, P. Surman, I. Sexton, M. Craven, K. C. Yow, and W. K. Lee. Building an Autostereoscopic Multiple-Viewer Television Display. In *Proceedings of Asian Symposium on Information Display (ASID) '04*, pages 400–404, Nanjing, Jiangsu, China, February 2004.

[3] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester. Image Inpainting. In *Proceedings of SIGGRAPH 2000*, pages 417–424, New Orleans, LA, USA, July 2000.

[4] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[5] J. Ens and P. Lawrence. An Investigation of Methods for Determining Depth from Focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):97–108, February 1993.

[6] C. Fehn. Depth-Image-Based Rendering (DIBR), Compression and Transmission for a New Approach on 3D-TV. In *Proceedings of SPIE Stereoscopic Displays and Virtual Reality Systems XI*, pages 93–104, San Jose, CA, USA, January 2004.

[7] C. Fehn, K. Hopf, and B. Quante. Key Technologies for an Advanced 3D-TV System. In *Proceedings of SPIE Three-Dimensional TV, Video and Display III*, pages 66–80, Philadelphia, PA, USA, October 2004.

[8] B. Horn and B. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, 1981.

[9] G. J. Iddan and G. Yahav. 3D Imaging in the Studio and Elsewhere .... In *Proceedings of SPIE Videometrics and Optical Methods for 3D Shape Measurements '01*, pages 48–55, San Jose, CA, USA, January 2001.

[10] M. Kawakita, K. Iizuka, H. Nakamura, I. Mizuno, T. Kurita, T. Aida, Y. Yamanouchi, H. Mitsumine, T. Fukaya, H. Kikuchi, and F. Sato. High-definition Real-time Depth-mapping TV Camera: HDTV Axi-Vision Camera. In *Opt. Express*, volume 12, pages 2781–2794, June 2004. `http://www.opticsexpress.org/abstract.cfm?URI=OPEX-12-12-2781`.

[11] M. Kawakita, T. Kurita, H. Kikuchi, and S. Inoue. HDTV Aix-Vision Camera. In *Proceedings of International Broadcast Conference '02*, pages 397–404, Amsterdam, The Netherlands, September 2002.

[12] J. Oliensis. A Critique of Structure-from-Motion Algorithms. *Computer Vision and Image Understanding: CVIU*, 80(2):172–214, 2000.

[13] A. Redert, M. Op de Beeck, C. Fehn, W. A. IJsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, I. Sexton, and P. Surman. ATTEST – Advanced Three-Dimensional Television System Technologies. In *Proceedings of 3D Data Processing Visualization and Transmission 2002*, pages 313–319, Padova, Italy, June 2002.

[14] O. Schreer. *Stereoanalyse und Bildsynthese*. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, March 2005.

[15] B. D. Lucas T. and Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, BC, Canada, April 1981.

[16] W. J. Tam, G. Alain, L. Zhang, T. Martin, and R. Renaud. Smoothing Depth Maps for Improved Stereoscopic Image Quality. In *Proceedings of Three-Dimensional TV, Video and Display III (ITCOM'04)*, volume 5599, pages 162–172, Philadelphia, PA, USA, October 2004.

[17] W. J. Tam, A. Soung Lee, J. Ferreira, S. Tariq, and F. Speranza. Stereoscopic Image Rendering Based on Depth Maps Created from Blur and Edge Information. In *Proceedings of the SPIE: Stereoscopic Displays and Virtual Reality Systems XII*, volume 5664, pages 104–115, San Jose, CA, USA, January 2005.

[18] W. J. Tam and L. Zhang. Non-uniform Smoothing of Depth Maps Before Image Based Rendering. In *Proceedings of Three-Dimensional TV, Video*

and Display III (ITCOM'04), volume 5599, pages 173–183, Philadelphia, PA, USA, October 2004.

[19] K.-J. Yoon and I.-S. Kweon. Locally Adaptive Support-Weight Approach for Visual Correspondence Search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 924–931, San Diego, CA, USA, June 2005.

[20] L. Zhang and W. J. Tam. Stereoscopic Image Generation Based on Depth Images for 3D TV. *IEEE Transactions on Broadcasting*, 51(2):191–199, June 2005.

[21] L. Zhang, W. J. Tam, and D. Wang. Stereoscopic Image Generation Based on Depth Images. In *Proceedings of IEEE Conference on Image Processing*, pages 2993–2996, Rochester, NY, USA, October 2004.

[22] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape from Shading: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, August 1999.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Münster, 15. Dezember 2005

Martin Lambers