Dense Disparity Maps in Real-Time with an Application to Augmented Reality

Jochen Schmidt, Heinrich Niemann
Lehrstuhl für Mustererkennung
Universität Erlangen-Nürnberg
Martensstr. 3, 91058 Erlangen, Germany
{jschmidt, niemann}@informatik.uni-erlangen.de,
http://www5.informatik.uni-erlangen.de

Sebastian Vogt
Siemens Corporate Research, Inc.
Imaging & Visualization Department
755 College Road East,
Princeton, NJ 08540, USA
sebastian.vogt@scr.siemens.com

Abstract

This work presents a technique for computing dense disparity maps from a binocular stereo camera system. The methods are applied in an Augmented Reality setting for combining real and virtual worlds with proper occlusions. The proposed stereo correspondence technique is based on area matching and facilitates an efficient strategy by using the concept of a three-dimensional similarity accumulator, whereby occlusions are detected and object boundaries are extracted correctly. The main contribution of this paper is the way we fill the accumulator using absolute differences of images and computing a mean filter on these difference images. This is where the main advantages of the accumulator approach can be exploited, since all entries can be computed in parallel and thus extremely efficient. Additionally, we perform an asymmetric correction step and a post-processing of the disparity maps that maintains object edges.

1. Introduction

In this paper we introduce a method for computing dense disparity maps for a stereo image pair. The application we have in mind is Augmented Reality, where computer generated virtual objects are to be rendered into a real scene with proper occlusion handling. A vast amount of literature is available on the geometry and calibration of binocular stereo camera systems [5, 8], hence we will not go into detail at this point. Besides the classical approaches for establishing correspondences between two images, which are feature- and area-based matching, other techniques such as phase- and energy-based ones have been developed. An overview can be found in [1]. We propose the use of block-matching by exploiting the advantages of a similarity accumulator resulting in a very efficient computation scheme for consistent dense disparity maps. 3-D accumulator concepts have been developed previously, e. g. see [15] and [18]. Augmented Reality applications using disparity maps can be found e. g. in [12, 16]. Other approaches do not compute disparity maps, but use contour-based methods for resolving occlusions, as in [3]. In [12] a recursive approach for computing disparity maps for video-conferencing scenarios is presented. The topic of obtaining disparity for telepresence applications by combining optical flow techniques and block-matching is addressed in [16].

A disparity computation algorithm that is to be used in an Augmented Reality application has to meet the following requirements:
Real-Time: Efficiency of the stereo algorithm is a determining criterion for the latency and thus the real-time capability of the whole Augmented Reality system.
Left and Right Depth Maps: For a stereoscopic Augmented Reality system it is necessary to have depth maps for both camera images. It is important to get consistent maps in both images, i. e. occlusions in the left image must fit to occlusions in the right image and vice versa.
Dense Disparity Maps: The disparity maps should be dense, i. e. they should contain a disparity for each pixel, otherwise occlusion of virtual objects by real objects will be insufficient.
Sharp Edges: Edges of real objects should be extracted very well since exactly at these locations virtual and real objects meet each other, and smooth transitions would lessen the immersion into the augmented scene.
Detection of Occlusions: In the case of occluding objects in the real scene no corresponding points can be detected for some areas of the two images. The correspondence algorithm should be able to detect these locations in the images so that gaps in the disparity maps can be filled in a post-processing step.

In the following section we will present a technique for computing dense disparity maps meeting the requirements above. Results are presented in Sect. 3 together with some examples of real scenes augmented by virtual objects.

2. Disparity Calculation

In the following we will describe how to compute disparity maps from two rectified images. The term *rectification* denotes a transformation of a given stereo image pair such

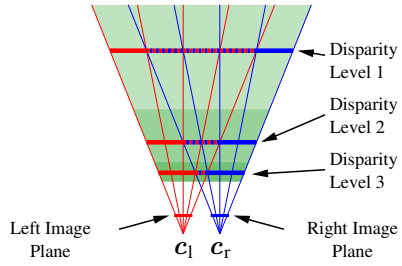

Figure 1. Disparity levels of a (rectified) stereo camera system.

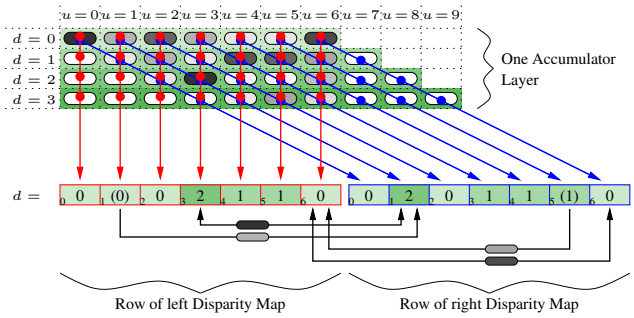

Figure 2. Schematic representation of an accumulator layer and strategy for determining the respective row in the left (D_l) and right (D_r) disparity map. The gray-value of an accumulator cell reflects the similarity between left and right image of a certain pixel when the disparity is given. Dark means strong, light means weak similarity. See text for a more detailed description.

that corresponding epipolar lines are colinear and parallel with one of the image axes, usually the horizontal one. This rectified image pair, also called normalized stereo image pair, can be considered as images taken by a stereo camera system that can be derived from the original one by rotating the cameras around the optical center. The main advantage of using rectified images is that corresponding points can be found on the same scanline in both images which makes searching much easier and faster since no resampling of the images along the epipolar lines has to be done over and over again. In [7] an efficient rectification algorithm is presented, which is used here.

For correspondence computation we introduce the concept of a *similarity accumulator*. Although other authors [15, 18] also use three-dimensional accumulators with disparity as the third dimension, our concept is new in the sense that it can be seen as an abstract intermediate step between the filling of the accumulator cells using a certain similarity measure and the actual matching for obtaining the two disparity maps. Additionally, we perform an asymmetric correction step and a post-processing of the disparity maps that maintains object edges.

In the following sections we will describe the similarity accumulator and how to compute dense and consistent disparity maps from the filled accumulator cells.

2.1. Similarity Accumulator

For finding corresponding points in a stereo image pair we use a block-matching method based on computing the sum of absolute differences (SAD) of two blocks:

$$\varepsilon_{\mathrm{SAD}}(u,v,d) = \sum_{\mu=-w}^{w}\sum_{\nu=-w}^{w} \left|I_{\mathrm{l}}(u+\mu,v+\nu) - I_{\mathrm{r}}(u-d+\mu,v+\nu)\right| , \tag{1}$$

where I_{l} and I_{r} denote the left and right image, d is the disparity, w the window size and (u,v) are the coordinates of the center pixel of the block, where $\varepsilon_{\mathrm{SAD}}$ is computed.

Figure 1 shows the origin of the block displacements between left and right image. For simplicity only four viewing rays (image columns) are shown from top. The main point is the overlapping of the fields of view of the two cameras. A planar object lying in the plane denoted by disparity level d is displaced in the right image exactly d pixel columns with respect to the left image. Disparity level 0 is at infinity, level 4 means that the object is only visible for one camera.

It is possible to build a three-dimensional similarity accumulator containing the two dimensions of the image planes plus disparity as the third dimension. The intensities $I_{\mathrm{l}}(u,v)$ and $I_{\mathrm{r}}(u-d,v)$ are used for computing the similarity measure in accumulator cell $a(u,v,d)$. The size of the accumulator is defined by the image size $u_{\mathrm{m}} \times v_{\mathrm{m}}$ and the disparity range $[d_{\mathrm{min}}, d_{\mathrm{max}}]$ in consideration: $(u_{\mathrm{m}} + d_{\mathrm{max}}) \times v_{\mathrm{m}} \times (d_{\mathrm{max}} - d_{\mathrm{min}} + 1)$.

The actual computation of accumulator entries will be described in Sect. 2.4. First, we will show how to get disparity maps from an already filled accumulator.

2.2. Consistent Left and Right Disparity Maps

The similarity accumulator allows us to tell which disparity is the most likely one for each pixel, if all accumulator cells were computed.

Figure 2 shows a schematic representation of the procedure described in the following. One can see a layer v of the accumulator a that allows us to compute the corresponding row v in the left and right disparity map. The entries of the accumulator are painted in different shades of gray; the darker the gray-value the more likely is the assignment of the corresponding disparity value. In order to get the optimal disparities of an arbitrary row v for the left disparity map D_{l}, a *vertical search* (with respect to the scheme) is done in the accumulator:

$$\forall u \in [0,\, u_{\mathrm{m}}-1] : D_{\mathrm{l}}(u,v) = \underset{d\in[d_{\mathrm{min}},d_{\mathrm{max}}]}{\operatorname{argmin}} \; a(u,v,d) \; . \tag{2}$$

It is assumed that smaller accumulator entries give a higher similarity, which is true when using $\varepsilon_{\mathrm{SAD}}$ (equation (1)). The row v of the right disparity map D_{r} is computed by a *diagonal search* in layer v of the accumulator:

$$\forall u \in [0,\, u_{\mathrm{m}}-1] : D_{\mathrm{r}}(u,v) = \underset{d\in[d_{\min},d_{\max}]}{\operatorname{argmin}} \; a(u+d,v,d)\ . \tag{3}$$

To get the complete maps (2) and (3) are evaluated for all rows $v \in [0,\, v_{\mathrm{m}}-1]$, i. e. all accumulator layers.

One problem is that up to now we do not know anything about the confidence of a disparity value computed by (2) and (3). This is especially important if there are occluding objects in the images, since no left/right correspondences are available in that case. A simple solution would be to introduce a threshold: If all accumulator entries in a certain diagonal or vertical direction are below the threshold, an occlusion and thus an undefined disparity is likely. In Fig. 2 this is the case for $D_{\mathrm{l}}(1,v)$ and $D_{\mathrm{r}}(5,v)$. But it is still open how to choose this threshold.

Another problem is the *consistency* of the pair of disparity maps $D_{\mathrm{l}}, D_{\mathrm{r}}$. If the maps are to be used for a stereoscopic augmentation of a real scene, it is important that *each* scene point has matching disparities in the left *and* right map, otherwise we would get different occlusions of virtual and real objects in the two images. We will now show how to solve these problems.

In the majority of cases homogeneous image regions have the effect that the accumulator has many very good similarity entries in diagonal and vertical direction, respectively. If a homogeneous region (e. g. a white wall) is partially occluded by a foreground object (e. g. a floor lamp) in the right image, the non-occluded homogeneous parts in the right image lead to wrong correspondences in the left image which are actually correspondence-less. The problem here is that those matches have a good similarity value and thus cannot be filtered out by the threshold method mentioned above. Therefore we will now give a technique for verifying disparities in the accumulator.

In early papers on computing dense disparity maps from stereo images [13, 14], Marr and Poggio make two assumptions about a stereo system, *uniqueness* and *continuity* of the disparity map. That means, each pixel in the two images can have exactly one disparity and thus exactly one depth in the scene. If we look at the accumulator cells $(u=1,v,d=0)$ and $(u=3,v,d=2)$ in Fig. 2, that give the best values for the vertical search for $D_{\mathrm{l}}(u=1,v)$ and $D_{\mathrm{l}}(u=3,v)$, we can see that the uniqueness assumption is violated, since both cells lie on a diagonal and the diagonal search for $D_{\mathrm{r}}(u=1,v)$ has to decide for the better one of both values. Thus two pixels in the left image are matched to the same pixel in the right image. In Fig. 2 this is emphasized by the arrows below the rows of the computed disparity maps.

Hence for each pixel (u,v) the computed left disparity map is tested for satisfying the following constraint:

$$|D_{\mathrm{r}}(u - D_{\mathrm{l}}(u,v),\ v)\ -\ D_{\mathrm{l}}(u,v)| \ \le\ d_{\mathrm{tol}}\ . \tag{4}$$

If the constraint is violated, the entry in the disparity map $D_{\mathrm{l}}(u,v)$ is set to *undefined*. This additional verification step enforces the uniqueness if $d_{\mathrm{tol}} = 0$. In some cases it is desirable to permit small deviations, i. e. to permit a slightly different disparity for the corresponding pixel. For stereo images with a wide disparity range $[d_{\min}, d_{\max}]$ a tolerance value of $d_{\mathrm{tol}} = 1,\ldots,5$ can make sense if changes of one disparity level are not significant.

The right disparity map is verified by the following condition:

$$|D_{\mathrm{l}}(u + D_{\mathrm{r}}(u,v),\ v)\ -\ D_{\mathrm{r}}(u,v)| \ \le\ d_{\mathrm{tol}}\ . \tag{5}$$

Thus we gain (with $d_{\mathrm{tol}} = 0$) consistent left and right disparity maps. Both maps still contain undefined entries which have to be filled. In Fig. 2 undefined disparities are parenthesized.

2.3. Filling the Gaps

Most methods (e. g. [12]) for computing dense disparity maps fill gaps in the maps by simple linear interpolation in the direction of the epipolar lines. The drawback is that neighboring lines are treated independently of each other and thus often violate the postulated continuity criterion in the disparity map in vertical direction. Hence it is important to use all available disparity information near the gap to be filled. We found the following technique to give very satisfying results.

First, a Median filter with a small mask size (3×3 or 5×5) is applied to the map in order to fill small gaps, especially those that came from missing correspondences caused by perspective distortions. Additionally, the Median filter suppresses noise while maintaining edges in the disparity map. A generously dimensioned morphological closing operator (i. e. dilation followed by erosion) is used for filling larger gaps. This brings the desired effect that foreground objects consist of contiguous disparity regions. Examples for the dimension of the closing operator can be found in Sect. 3.

At the end large undefined regions are filled along the scanlines (which are the epipolar lines since we rectified the images) using the smaller of the two disparities at the left and right end of the gap. The reason for this is that undefined regions are mainly caused by occlusions where objects far away (having small disparities) are occluded by near objects (having large disparities).

In the following section we will answer the still open question of how to compute the entries of the accumulator.

2.4. Computing Similarity Accumulator Entries

A cell of the three-dimensional accumulator $a(u,v,d)$ rates the similarity of the pixel $I_{\mathrm{l}}(u,v)$ and $I_{\mathrm{r}}(u-d,v)$ while considering at least the local neighborhood. A simple block matching would be sufficient, but we will see that the construction of the accumulator allows us to be more efficient.

With a window size of $(2w+1)\times(2w+1)$ the computation of $\varepsilon_{\mathrm{SAD}}(u,v,d)$ for a given triple (u,v,d) needs

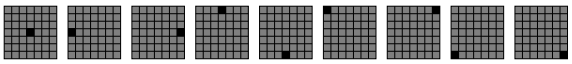

Figure 3. The 9 windows used for block matching for a window size of 7 × 7, 8 of them asymmetric.

$(2w+1)^2$ subtractions and calculating the absolute values, and $(2w)^2$ additions. With an image size of $u_\mathrm{m} \times v_\mathrm{m}$ and a disparity range of $[d_\mathrm{min}, d_\mathrm{max}]$ this results in $u_\mathrm{m} v_\mathrm{m} d_\mathrm{m} (2w+1)^2$ absolute differences and $u_\mathrm{m} v_\mathrm{m} d_\mathrm{m} (2w)^2$ additions, if $d_\mathrm{m} = (d_\mathrm{max} - d_\mathrm{min} + 1)$ and if we omit the image borders.

The similarity accumulator allows us to use a much more efficient method for computing $\varepsilon_\mathrm{SAD}(u, v, d)$ as an entry $a(u, v, d)$. We start by computing the absolute differences of the two images at a certain disparity d. Thus we get d_m difference images; the absolute differences are stored in the accumulator. This way only $u_\mathrm{m} v_\mathrm{m} d_\mathrm{m}$ absolute differences have to be computed.

The result could be used as a similarity measure for corresponding pixels in the left and right image: Large values mean low, small values high similarity. These difference images alone are equivalent to a block matching with window size 1×1 (i. e. $w = 0$) and thus are very noisy accumulator entries. This can be avoided by applying an additional mean filter on the difference images with window size $(2w+1) \times (2w+1)$. Since we will work with integers in practice, one should use a modified mean filter that only sums the values without dividing by their number. Such a modified mean filter is separable and thus can be implemented to work in linear time complexity with respect to the image size, because for each pixel only one addition and one subtraction is necessary, giving a total of $2 u_\mathrm{m} v_\mathrm{m} d_\mathrm{m}$ operations for computing the sum of absolute differences of the d_m layers of the similarity accumulator.

2.5. Extensions

What we did not consider yet is the kind of window that defines the neighborhood for block matching. Up to now we used a symmetric window; however, this leads to bad SAD values when the window covers edges of objects and thus different disparities. Therefore it is useful to apply a concept introduced in [6], where 9 different windows are used, 8 of them asymmetric. Figure 3 shows the 9 windows for a window size of 7×7 (i. e. $w = 3$). The window giving the best SAD value determines the disparity of the current pixel.

The method described is very efficient and can easily be integrated in the concept of the accumulator: First the disparity maps are computed as described above; for computing the cell entries the symmetric mean filter is used. After constructing the maps by vertical and diagonal search, for each pixel the disparity and the SAD value are compared with the 8 neighboring SAD values, where neighborhood is defined by the black pixels in the windows shown in Fig. 3, which are combined into one mask by taking the current pixel as the center of each window. The window size has to be the same as the one of the symmetric mean filter. If one of the 8 pixels in the map has a better SAD value, that one is used instead of the original one. This makes sense since the current pixel is apparently one belonging to the edge of an object, because this pixel was used at the border of the window for computing the symmetric mean of the chosen neighbor.

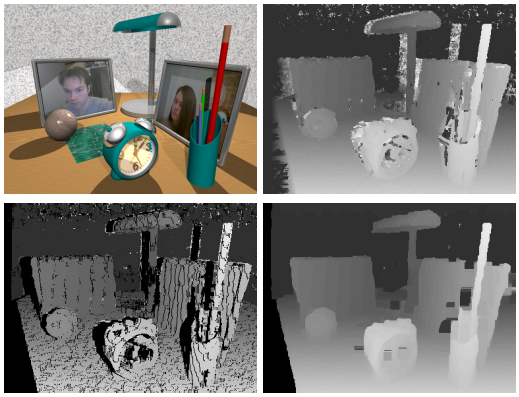

Figure 5. Original image (top left), disparity map before (top right) and after (bottom left) synchronization, and final disparity map (bottom right) of `desk` image pair. Shown is the left image only.

Figure 4 shows the complete process of computing consistent disparity maps. An additional speed-up can be gained by using Gaussian pyramids in the following way: Instead of computing the disparity maps at the finest resolution, we compute a pyramid and the maps at the coarsest resolution. Subsequently these maps can be scaled to the original resolution by a Gaussian filter. This results in disparity maps having less quality, but the speed up is enormous: When using a pyramid with just two levels, the speed up is a factor of 8, when using three levels the factor is 64. This results from the size of the accumulator, which is halved in each dimension at each additional pyramid level.

3. Experiments

For implementation we used the Intel Image Processing Library IPL [10] and Intel OpenCV Library [11] because of their optimized routines. Figure 5 shows a sequence of processing steps, starting from an original image pair created by Povray. Large disparities are encoded as light gray-values, small disparities as dark gray-values; black pixels are undefined. In Fig. 5 top right the disparity map before synchronization of both maps is shown. It can be seen that this map was computed very reliable at locations where correspondences could be actually established. However, regions that were occluded in one of the images result in obviously wrong disparities. Synchronization with a tolerance of $d_\mathrm{tol} = 0$ results in the map shown at the bottom left.

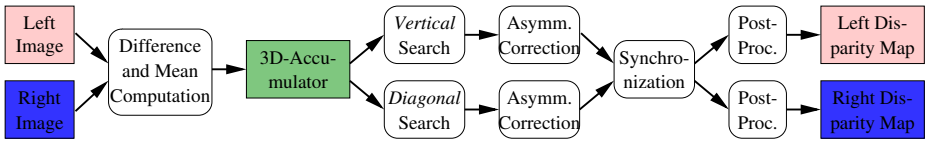

Figure 4. Data-flow for computing consistent disparity maps for the left and right image.

Parameter	desk	head	tree	roff1	roff2
Image Size	384×288	384×288	256×233	720×576	720×576
Pyramid Levels	1	1	1	1	2
Disparity Range	12–59	0–15	0–7	28–75	14–37
No. Disparity Layers	48	16	8	48	24
Size of Correlation Window	5×5	3×3	3×3	7×7	3×3
Tolerance Value d_{tol}	0	1	0	0	0
Size of Median	5×5	5×5	5×5	5×5	3×3
No. of Closing Operations	5	3	1	5	2

Table 1. Parameter values for disparity computation.

The wrong disparities are now undefined, resulting in many small gaps. As shown in the final map in Fig. 5 bottom right, those gaps were filled after applying a Median filter of size 5×5 in a post-processing step. In addition to the Median we used a closing operator, consisting of a dilation which was applied 5 times followed by 5 erosions. Mask size was 3×3 in both cases. The parameters used for computing the disparity maps can be found in Table 1, computation times in Table 2. In addition to the artificial Povray image pair

Disparity Comp.	desk	head	tree	roff1	roff2
RGB → Gray	2	2	1	7	9
Accumulator Comp.	91	32	17	342	44
Vert.+Diag. Search	98	41	11	396	51
Asymmetric Corr.	24	26	15	87	23
Synchronization	1	2	1	9	2
Post-Processing	11	10	6	42	9
Pyramid Comp.	—	—	—	—	17
Total	**227**	**113**	**51**	**883**	**155**

Table 2. Disparity computation times in msec on a Linux-PC with Intel Pentium 4, 1.9 GHz.

desk we give the parameters and computation times for the stereo reference image pairs head [17] and tree [4]. Disparity maps for head and tree are shown in Fig. 6. The roffice image pair [2] was taken by a hand-held camera (not stereo) at our institute. Calibration information was obtained by a structure from motion approach [9]. Figure 7 (top) shows the rectified image pair. The columns in Table 1 and 2 denoted by roff1 and roff2 refer both to the image pair roffice. The difference is that for roff2 we applied a two level pyramid algorithm as described in the previous section. Computed disparity maps for the parameters roff1 are shown in Fig. 7 (middle). The application of the Gaussian pyramid gives a slightly coarser resolution of the maps, but results in a speed up factor of 5–6.

Since our goal is to apply the computed disparity maps in Augmented Reality, we transformed them into depth buffer entries that can be used directly by OpenGL-Hardware. The result of augmenting the roffice scene with four objects using the disparity maps from Fig. 7 (middle) is shown in

Figure 6. Left image and disparity map of head (top) and tree (bottom), parallel stereo configuration.

Fig. 7 (bottom). The occlusion of virtual objects by elements of the real scene is fairly good, especially if you look at the edges of the objects.

4. Conclusion

We proposed a method for computing dense disparity maps for two rectified images obtained by a stereo camera. The edges of objects are maintained and we are able to compute the maps for smaller image sizes almost in real-time. Disparities are obtained using a method based on the concept of a similarity accumulator. The basic principle is an area-based matching using the sum of absolute differences which can be accelerated considerably by a three-dimensional accumulator, because the differences between the same pixels have to be computed only once. Since we use rectified images only horizontal disparities have to be considered. To get the disparities we compute $(d_{\text{max}} - d_{\text{min}} + 1)$ difference images obtained by displacing

Figure 7. `roffice` **image pair, taken by a hand-held camera. Top: rectified images. Middle: Disparity maps for parameters** `roff1`**. Bottom: Augmented scene**

the two images horizontally by $d = d_{\min}, \ldots, d_{\max}$ pixels followed by pixel wise computation of absolute differences. These difference images are mean-filtered using a small mask; the result is used as the entries of the cells of the three-dimensional similarity accumulator.

A special method for accessing the accumulator cells allows us to compute two disparity maps, one for each stereo image. An asymmetric correction step is performed on the maps in order to get an update from a symmetric to an asymmetric block-matching that maintains object edges in the disparity maps.

Since occlusions lead to correspondence-less areas in the images, the two maps are synchronized, i. e. one pixel in the left image may only correspond to one pixel in the right image and vice versa. This enables us to localize wrong correspondences and to extract object edges exactly, since usually occlusions are caused at these edges which often lead to wrong disparities. The resulting maps are post-processed using a Median filter and a morphological closing operator.

The quality of the maps was evaluated using rendered and real stereo images. Computation time mainly depends on the resolution and the disparity range and was about 100–200 milliseconds for image pairs having a resolution of 384×288.

The method can be accelerated by using a resolution pyramid, since a bisection of the image size in both dimensions reduces the volume of the similarity accumulator and thus computation time by $1/8$.

References

[1] L. Alvarez, R. Deriche, J. Sanchez, and J. Weickert. Dense Disparity Map Estimation Respecting Image Discontinuities: A PDE and Scalespace Based Approach. Technical Report RR-3874, INRIA, January 2000.

[2] Augmented Reality Page, Chair for Pattern Recognition, University of Erlangen-Nuremberg. `http://www5.informatik.uni-erlangen.de/~ar`.

[3] M. Berger. Resolving Occlusions in Augmented Reality: A Contour-based Approach without 3D Reconstruction. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 91–96, Puerto Rico, 1997.

[4] Carnegie Mellon University – VASC: The Vision and Autonomous Systems Center. http://www.vasc.ri.cmu.edu.

[5] O. Faugeras and Q.-T.-Luong. *The Geometry of Multiple Images, The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications*. MIT Press, Cambridge, Massachusetts, 2001.

[6] A. Fusiello, V. Roberto, and E. Trucco. Experiments with a new Area-Based Stereo Algorithm. In *Proc. of the 8th Int. Conf. on Image Analysis and Processing*, pages 669–676, Firenze, Italy, September 1997.

[7] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.

[8] R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, Cambridge, 2000.

[9] B. Heigl. *Plenoptic Scene Modelling from Uncalibrated Image Sequences*. Dissertation, Lehrstuhl für Mustererkennung, Universität Erlangen-Nürnberg, 2002. to appear.

[10] Intel IPL. http://support.intel.com/support/performance tools/libraries/ipl/index.htm.

[11] Intel OpenCV. http://www.intel.com/research/mrl/research/opencv/.

[12] P. Kauff, N. Brandenburg, M. Karl, and O. Schreer. Fast Hybrid Block- and PixelRecursive Disparity Analysis for Real-Time Applications in Immersive Tele-Conference Scenarios. In *Proc. of 9th International Conf. in Central Europe on Computer Graphics, Visualization, and Computer Vision*, pages 198–205, February 2001.

[13] D. Marr and T. Poggio. Cooperative Computation of Stereo Disparity. *Science*, 194:209–236, 1976.

[14] D. Marr and T. Poggio. A Computational Theory of Human Stereo Vision. In *Proc. of Royal Society London*, volume B 204, pages 301–328, 1979.

[15] K. Mühlmann, D. Maier, J. Hesser, and R. Männer. Calculating Dense Disparity Maps from Color Stereo Images, an Efficient Implementation. *Int. Journal of Computer Vision*, 47(1-3):79–88, April-June 2002.

[16] J. Mulligan and K. Daniilidis. View-independent scene acquisition for tele-presence. In *Proc. of the IEEE and ACM Int. Symp. on Augmented Reality*, pages 105–110, Munich, Germany, October 2000. IEEE Computer Society.

[17] University of Tsukuba – Computer Vision and Image Media Lab. http://image-gw.esys.tsukuba.ac.jp.

[18] C. L. Zitnick and T. Kanade. A Cooperative Algorithm for Stereo Matching and Occlusion Detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(7):675–684, July 2000.