7. International Immersive Projection Technologies Workshop
9. Eurographics Workshop on Virtual Environments (2003)
J. Deisinger, A. Kunz (Editors)

Detecting Dynamic Occlusion in front of Static Backgrounds for AR Scenes

Jan Fischer,[1] Holger Regenbrecht[2] and Gregory Baratoff[2]

[1] WSI/GRIS, University of Tübingen, fischer@gris.uni-tuebingen.de
[2] Virtual Reality Competence Center, DaimlerChrysler Ulm, {holger.regenbrecht,gregory.baratoff}@daimlerchrysler.com

Abstract

Correctly finding and handling occlusion between virtual and real objects in an Augmented Reality scene is essential for achieving visual realism. Here, we present an approach for detecting occlusion of virtual parts of the scene by natural occluders. Our algorithm is based on a graphical model of static backgrounds in the natural surroundings, which has to be acquired beforehand. The design of the approach aims at providing real-time performance and an easy integration into existing AR systems. No assumptions about the shape or color of occluding objects are required. The algorithm has been tested with several graphical models.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Information Interfaces and Presentation]: Artificial, augmented, and virtual realities

1. Introduction

In Augmented Reality, virtual objects are combined with a real environment. Video see-through AR systems permanently acquire camera images of the real surroundings in order to perform this mixing process [1]. After determining the current position and orientation of the user within the scene, corresponding projections of the virtual graphical objects are then rendered on top of the camera image.

Whereas this method of mixing real and virtual elements is easy to implement and fast enough for real-time systems, it also has major drawbacks. Because all pixels of the graphical virtual elements are drawn over the camera image, they completely occlude the real environment. In many cases this does not represent the actual situation in the AR scene. It is possible that virtual objects should be behind real objects. A common example for this situation is user interaction within the AR scene. The user's hands and pointing devices are hidden by virtual objects although they normally are closer to the camera.

Several types of occlusion in AR scenes can be distinguished. On the one hand, occlusion can be characterized by what kind of object is hidden by what kind of occluder. A good overview of this has been given by Klinker [14]. In our research, we were only interested in virtual objects being occluded by real ones. This can again be split into two cases. In the first case, the exact geometry and location of the real objects are known. We call this static occlusion.

By contrast, in dynamic occlusion nothing is known about the shape, size or position of an occluding natural object. Thus it is necessary to detect potential occluders for graphical objects in the camera image. A simple approach for finding dynamic occluders in an AR scene would be to assume that occluding objects have a certain color. Our algorithm does not rely on any such assumption about the occluders. They may be of any geometry or texture.

In order to be able to find pixels belonging to dynamic occluders in the input camera image, our approach requires a graphical model for certain parts of the scene. This graphical model consists of a set of textured polygons, in front of which occlusion can be detected. These polygons have to correspond to surfaces in the real world. By comparing the camera image with the projection of the graphical model, dynamic occluders are found. Several steps are necessary to perform this operation. They are described in Section 4.

Any object occluding one of the given static backgrounds is assumed to be in front of the graphical elements of the AR scene. The occluder can then be drawn over the virtual objects for a more realistic display of the AR scene. Note that this method can not deliver any real depth information for the scene. This would be necessary for correctly handling more complex spatial relationships between the vari-

ous kinds of objects. The limitations that are imposed on our approach by this are discussed in more detail in Section 7.

2. Related work

Detecting occlusion for Augmented and Mixed Reality has been an area of active research for several years. Breen et al. have suggested a model-based method for handling static occlusion in AR [3]. This method has also been extended by combining a previously acquired geometric model with positional data from a tracking system for determining how virtual objects are hidden by the user's body [6].
Several approaches for stereo camera AR systems have been described. Some solve the occlusion problem in general using depth information delivered by stereo matching [11, 22]. The method developed by Gordon et al. [7] can correctly render interaction devices into the scene.
The AR system of Malik et al. [16] detects the occlusion of black and white marker regions by the user's hand. This is achieved using a histogram-based thresholding process refined by a flood-fill algorithm.
Some research has also been done into occlusion in non-real-time Augmented Reality. Correctly handling occlusion while adding virtual objects to stored video sequences was examined by Berger and Lepetit [2, 15].

3. Acquisition of the graphical model

Our method requires a graphical model of surfaces in the user's environment. Each surface is described by a number of textured coplanar polygons. We call such a surface *dynamic occlusion background* (*DOB*). Before the algorithm for detecting dynamic occlusion can be used in a given AR scene, the dynamic occlusion backgrounds for the relevant parts of the scene have to be acquired.
In order to generate the DOB models, we have implemented

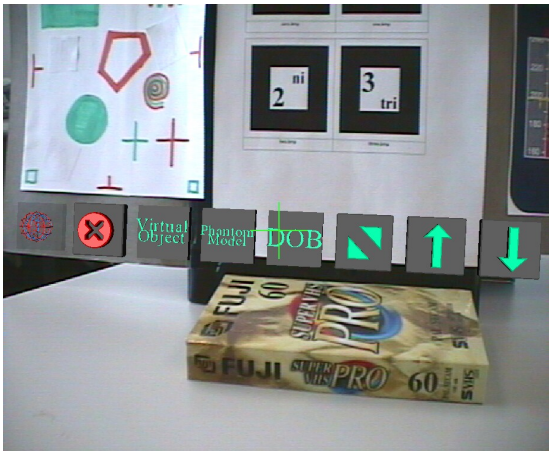

Figure 1: *Part of AugmentEd's menu system.*

an interactive tool for editing AR scenes. With this software, *AugmentEd*, it is possible to define a complete AR scene consisting of virtual objects, phantom models for static occlusion [3] and dynamic occlusion backgrounds (see Figure 1). *AugmentEd* uses Open Inventor [21] as I/O file format.
For defining a dynamic occlusion background from scratch, we designed a simple process spanning all steps from taking a digital image of the surface in the scene to correctly positioning the model in world space. Using this method, we have created a number of simple occlusion backgrounds, mostly cockpit instruments used in automotive and aircraft design (see example in Figure 2).

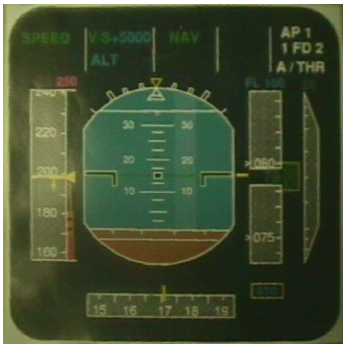

Figure 2: *Example DOB (aircraft instrument).*

4. Overview of the algorithm

We propose an algorithm for detecting dynamic occlusion in front of static backgrounds. This algorithm is based on the comparison of the graphical model with the current camera image. Where the camera image is different from the expected appearance suggested by the graphical model, the background is assumed to be occluded. If the camera image is (nearly) identical with the expectation, the background can be seen without occlusions. In order to make such a comparison possible, a number of pre-processing steps must be performed.
The fundamental idea of our approach is to render the graphical image for each DOB into an offscreen buffer. The camera position and orientation used for this rendering step are taken from our AR system's marker tracking [12]. Due to inaccuracies in this computed camera pose, the generated internal DOB image then has to be adapted. The reason for this is the fact that the final comparison is done on a per-pixel basis. Thus it is necessary that position and shape of the internal image exactly match the DOB in the camera image. A simple overview of the algorithm is the following:

```
a) Render DOB to offscreen buffer
b) Adapt internal DOB image to match
   camera image
c) Compare adapted internal DOB and cam-
   era image
```

The result of the final step is an occlusion mask. This is a bitmap in which a bit is set for every pixel where occlusion has been detected. This entire process has to be repeated for each dynamic occlusion background. The resulting final occlusion mask can be obtained by combining all individual occlusion masks using the binary OR as shown in Equation 1.

$$occMask = \bigvee_{i=1}^{N} partialOccMask_i \quad (1)$$

Here, $partialOccMask_i$ is the occlusion mask computed for the i-th dynamic occlusion background. The total number of DOBs in the scene is N.

In order to adapt the internal DOB to the camera image, a template matching method similar to the one used in the markerless tracking system demonstrated by Kato et al. [13] is applied. This method again consists of several steps, which determine how the internal image has to be transformed so that it matches the position of the background surface in the camera image. First salient points in the internal DOB image are detected. Then positions in the camera image corresponding to these salient features are searched. The result of this matching process is an array of displacement vectors describing the transformation of the internal DOB to camera image space. This 2D transformation is then calculated using standard numerical techniques. After that, the computed transformation is applied to the DOB image. In more detail, the steps of our algorithm are the following:

1. Render DOB to offscreen buffer
2. Detect salient points in the DOB image
3. Find correspondences for salient points in camera image
4. Filter correspondences according to their confidence
5. Compute 2D transformation
6. Transform internal DOB image
7. Compare adapted internal DOB and camera image
8. Perform final filtering of occlusion mask

The individual steps of our algorithm as listed above are discussed more comprehensively in Section 5.

5. Implementation details

For our research we developed an implementation of the algorithm for detecting dynamic occlusion. This implementation is based on fundamental AR system components developed at the VRCC. It uses a marker tracking similar to ARToolKit [12]. The basic AR system has been described more comprehensively by Regenbrecht et al. [18].

5.1. Offscreen rendering of the DOB image

Each DOB is rendered into an offscreen buffer. The description of the dynamic occlusion background is taken from the graphical model (see Section 3). For the offscreen rendering process, the Mesa graphics library is used [17]. (The obvious solution of using standard OpenGL and reading back the framebuffer was rejected because buffer readback is extremely expensive on most hardware.) Each textured polygon is drawn using 3D position and orientiation information from the marker tracking.

5.2. Finding salient points on the DOB

In the offscreen image buffer, salient points on the DOB are searched. This is done using the structure tensor [20], which is a basic image processing operation for finding two-dimensional structures (like corners). We assume that such points in the image are well suitable for finding correspondences in the camera image.

The structure tensor is defined as the following matrix:

$$S = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \quad (2)$$

In Equation 2, I_x and I_y are the partial derivatives of the image. The summations are made over a rectangular area in the image, which is centered at the pixel that is currently looked at. After this matrix has been computed, its eigenvalues are calculated. If the smaller of the two resulting eigenvalues is greater than a threshold value, there is a sufficiently interesting two-dimensional structure at this pixel [10, 20]. Only this smaller eigenvalue is important for further considerations.

The candidates for salient points delivered by the structure tensor are filtered because they often are too numerous. A specialized function of the Intel OpenCV library is used for this [10]. First of all, all points with a non-maximal eigenvalue

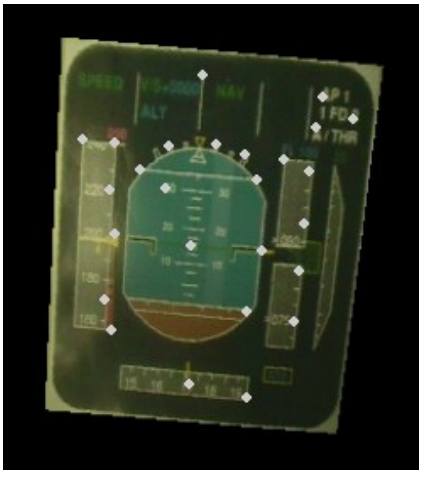

Figure 3: *Salient points detected on a DOB.*

in 3x3 neighborhoods are ignored. After that, the maximum of the eigenvalues of all remaining points is calculated. Only points which have an eigenvalue big enough relative to this maximum are further considered. Finally, a minimum distance between the salient points is ensured.
Subsequently another filtering step is necessary which is specific to our algorithm. The reason for this is that along the edges of the DOB polygons there can be many two-dimensional structures which in fact are aliasing artifacts of the renderer. Therefore we remove all salient points on the DOB borders. An example for salient points on a DOB is shown in Figure 3.

5.3. Establishing point correspondences

For each detected salient point p_i on the DOB, a correspondence is searched in the camera image. This is done using template matching. The template used is the quadratic area with side length *templateLength* centered at the salient point. A correspondence for this template is looked for in a quadratic area with side length *searchLength* in the camera image. This search area is centered at the same coordinates that the salient point has in the internal DOB image. This requires the DOB image in the offscreen buffer to have the same resolution as the camera image.
For template matching we use the normalized correlation coefficient, which is a similarity measure based on the cross correlation [10, 19]. The correlation coefficient is computed for every pixel in the search area. It has a value of 1 for maximum similarity (i.e. identity) and a value of -1 for minimum similarity. It is computed as follows [10]:

$$\tilde{R}(x,y) = \frac{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1} \tilde{T}(x',y')\tilde{I}(x+x',y+y')}{\sqrt{\sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1} \tilde{T}(x',y')^2 \sum_{y'=0}^{h-1}\sum_{x'=0}^{w-1} \tilde{I}(x+x',y+y')^2}} \tag{3}$$

In Equation 3, $\tilde{T}(x',y')$ is the difference in intensity between a template pixel and the average brightness $\bar{T}$ of the entire template, i.e. $\tilde{T}(x',y') = T(x',y') - \bar{T}$. Correspondingly, $\tilde{I}(x+x',y+y')$ is the difference between a pixel in the image and the average brightness $\bar{I}$ of the search area. The numerator of the fraction is simply a cross correlation over differences from the average brightness. Because the differences are used, template/image pairs with different overall brightness have comparable correlation coefficients. The denominator normalizes the result of the cross correlation, thus always delivering a $\tilde{R}(x,y) \in [-1;1]$.
In our algorithm, both w and h always have a value of *templateLength*. Note that Equation 3 is only defined for single-channel images. Therefore both the template image and the camera image are converted to grayscale before calculating the correlation coefficient. $\tilde{R}(x,y)$ is computed for the entire search area. The pixel with the maximum coefficient is defined as the correspondence q_i for the currently considered salient point p_i. The lines connecting each p_i to its corresponding q_i can be drawn as displacement vectors as shown in Figure 4.

Figure 4: *Point correspondences shown as displacement vectors.*

5.4. Computing the 2D transformation

The detected point correspondences are then used for calculating a 2D transformation. This transformation will later be applied to the internal DOB image. The input data for computing the transformation are the pairs of salient points p_i and their correspondences q_i. A matrix describing the transformation from DOB image space to camera image space is to be determined. Equation 4 expresses this relationship.

$$q_i = A \cdot p_i, \quad i = 1,\ldots,m \tag{4}$$

A homogeneous transformation is required since there can be a translational component when comparing the DOB and camera images. Thus the p_i and q_i points have to be extended by a homogeneous coordinate. Transformation matrix A is a 3x3 matrix. This is shown in Equation 5.

$$\left(\frac{q_i}{1}\right) = A \cdot \left(\frac{p_i}{1}\right), \quad i = 1,\ldots,m, \quad A \in \Re^{3x3} \tag{5}$$

There is one equation for each pair of salient point and correspondence. The combination of these is an over-determined equation system. This equation system is to be solved for matrix A. In order to do this using standard numerical methods, equation system (5) has to be reshaped in terms of the vector of unknowns $\hat{a} = (a_1, a_2, \ldots, a_9)^T$, the entries of the transformation matrix A:

$$\hat{M} \cdot \hat{a} = 0 \tag{6}$$

In Equation 6, $\hat{M}$ is a matrix containing all data of the salient points and their correspondences. A detailed derivation for

$\hat{M}$ can be found in [4], as well as a more general explanation in [8]. Here, it is defined as shown in Equation 7.

$$\hat{M} = \begin{pmatrix} x_{p_1} & y_{p_1} & 1 & 0 & 0 & 0 & -x_{p_1}\cdot x_{q_1} & -y_{p_1}\cdot x_{q_1} & -x_{q_1} \\ 0 & 0 & 0 & x_{p_1} & y_{p_1} & 1 & -x_{p_1}\cdot y_{q_1} & -y_{p_1}\cdot y_{q_1} & -y_{q_1} \\ x_{p_2} & y_{p_2} & 1 & 0 & 0 & 0 & -x_{p_2}\cdot x_{q_2} & -y_{p_2}\cdot x_{q_2} & -x_{q_2} \\ 0 & 0 & 0 & x_{p_2} & y_{p_2} & 1 & -x_{p_2}\cdot y_{q_2} & -y_{p_2}\cdot y_{q_2} & -y_{q_2} \\ \vdots & & & & \vdots & & & & \vdots \end{pmatrix} \quad (7)$$

In Equation 7, the $x_{p_i}, x_{q_i}, y_{p_i}$ and y_{q_i} are the respective coordinates of the salient points and their correspondences. The equation system is then solved using a numerical method based on the singular value decomposition [20]. The vector $\hat{a}$ is given by the eigenvector corresponding to the smallest eigenvalue of $\hat{M}$, which is the solution with the smallest residual error.

Since the values in $\hat{M}$ can become very large depending on the image resolution used, there can be numerical problems when calculating the singular value decomposition. Thus beforehand all 2D coordinates appearing in the equation system are normalized to be in the interval $[-1;1]$. This is explained in more detail in [8] and [4].

Another important pre-processing of the pairs of salient points and correspondences is done even before the construction of $\hat{M}$. When searching correspondences for the salient points, it is possible that no satisfying result is found. There is a number of possible reasons for this. Since the search area is of fixed size, marker tracking inaccuracies which are too big prevent the algorithm from finding a correspondence. Also if the relevant part of the camera image is hidden by other objects, no corresponding point can be detected. In order to distinguish valid correspondences from incorrect ones, a confidence measure c_i for each point pair is defined, which is simply the correlation coefficent calculated for the correspondence point. Only point pairs with a given minimum confidence $confThresh$ are taken into account for the equation system (see Equation 8) .

$$c_i = \tilde{R}(q_i), \quad i = 1,\ldots,k$$

$$\{(p_i,q_i)\}_{i=1}^{m} = \{(p_i,q_i)\,|\,c_i \geq confThresh\}_{i=1}^{k} \quad (8)$$

This usually reduces the number of point pairs from k to a smaller m. At least five point pairs have to remain after this selection. Otherwise, we cannot compute a solution for the equation system.

Note that we only compute a 2D transformation for the DOB image, although the underlying marker tracking inaccuracies can be three-dimensional. Nonetheless, a homogeneous tranformation in 2D image space is sufficient for correcting the DOB image. The reason for this is the fact that both in the graphical model and in reality all DOB polygons lie on a plane [4].

5.5. Applying the transformation

The computed transformation is then applied to the internal DOB image. This is done on a per-pixel basis using a specialized function of the Intel IPL image processing library [9]. Every pixel in the original internal DOB image is copied to its new position in a new image buffer according to transformation matrix A. Averaging or interpolation of pixel values are performed when necessary.

Note that not all vectors $\hat{a}$ that can be returned by the singular value decomposition are valid transformations in this context. The IPL function only works with projective transformations. Thus matrix A is tested for its validity beforehand. Degenerated matrices are computed if wrong or contradictory point correspondences are found in the preceding steps of the algorithm.

5.6. Performing the image comparison

After the DOB image has been transformed to eliminate the effects of marker tracking inaccuracy, the actual image comparison can take place. Position and orientation of the internal DOB image should now match the real DOB in the camera image. For the entire area of the DOB a pixelwise comparison criterion is evaluated. The criterion determines whether a pixel in the internal DOB is identical with the real DOB pixel. Depending on this, either 0 or 1 is stored in the occlusion mask for this pixel. We tested a number of different pixel comparison criteria in our research. They are described in Section 5.7.

After the entire occlusion mask has been computed as described, a final filtering step is executed. This is done so that "outlier" pixels which are not part of a coherent occluding object are removed from the mask. Therefore morphological operators [19] are applied to the occlusion mask. A sequence of Opening and Closing over 3x3 neighbourhoods is repeated several times.

5.7. Pixel identity criteria

We have evaluated a number of different criteria for pixel comparison. Among them there are rather simple and straightforward ones like a comparison of absolute intensity values. The complete list of criteria can be found in [4]. Here, we will elaborate only on two more complicated ones.

The first criterion is the comparison of the difference from average intensity in a window area. For computing this criterion, at first both images are converted to grayscale. Then for every pixel in both images a square window area surrounding it is defined, the size of which is fixed. The difference between the average intensity in this window and the pixel's intensity is computed. This is done both for the DOB and the camera image. The two differences are then compared, and their difference is thresholded for the criterion. The entire

process is shown in Equation 9.

$$\bar{I}_{dob}(x,y) = \frac{\sum_{y'=1}^{h}\sum_{x'=1}^{w} I_{dob}(x+x'-w/2, y+y'-h/2)}{w \cdot h}$$

$$\Delta I_{dob}(x,y) = I_{dob}(x,y) - \bar{I}_{dob}(x,y)$$

$\bar{I}_{cam}$ and ΔI_{cam} are calculated correspondingly

$$occ(x,y) = \begin{cases} 1, & |\Delta I_{dob}(x,y) - \Delta I_{cam}(x,y)| > threshold \\ 0 \end{cases} \quad (9)$$

In Equation 9, I_{dob} and I_{cam} are the intensity of pixels in the DOB and camera images. This criterion is very expensive because the local average intensity must be determined for every pixel. This generates an overall complexity of $O(n^2m^2)$ (with n being an approximation for the local window's side length and m an approximation for the image side length). But it is possible to arrange the summations for the calculation of the local averages so that a complexity of $O(m^2)$ is achieved [4].

The original motivation for this criterion was to eliminate the influence of local brightness variations during image comparison. Such variations can result for example from shadows cast upon the DOB. Another common cause are differences in lighting between the DOB texture in the graphical model and the camera image. Several tests have proven this criterion as unable to produce satisfying occlusion masks. The reason for this is a strong depedence on the size of the averaging window. A large averaging window prevents the criterion from reacting to local changes in lighting. When using a small averaging window the criterion actually computes the spatial derivation of the image function and cannot determine differences in homogeneous areas.

A sufficiently good occlusion mask is calculated by the "Adaptive HSV" criterion. This criterion also makes use of the pixel color information. In a first step both the DOB and the camera image are converted to HSV images. HSV images contain the relevant color information in their H (Hue) and S (Saturation) channels [5]. The problem is that color information delivered by the camera is almost random for very dark pixels. Therefore, we decided to adaptively balance the influence of color and intensity differences based on the pixel brightness. For brighter pixels, color differences play a bigger role.

$$\Delta H(x,y) = |H_{dob}(x,y) - H_{cam}(x,y)|$$

ΔS and ΔV are calculated correspondingly

$$\alpha(x,y) = \beta \cdot min(V_{dob}(x,y), V_{cam}(x,y))$$

$$o(x,y) = \alpha(x,y) \cdot \frac{\Delta H(x,y) + \Delta S(x,y)}{2} + (1-\alpha(x,y)) \cdot \Delta V(x,y)$$

$$occ(x,y) = \begin{cases} 1, & o(x,y) > threshold \\ 0 \end{cases} \quad (10)$$

In Equation 10, $\alpha(x,y)$ is the function determining the balancing factor. Color difference is calculated as the average difference in the H- and S-channels. Variable β is a user-defined parameter for setting the maximum influence that color differences can have.

We have found the "Adaptive HSV" criterion to deliver fairly good results in most cases, and it is the best of six criteria which we tested for the occlusion detection algorithm. Still, in some scenes finding DOB and camera image identity is not completely reliable.

Figures 5 and 6 show an example of an occlusion mask generated by the "Adaptive HSV" criterion.

Figure 5: *Example camera image. A DOB (the bright rectangle) is partly occluded by an object (a black briefcase).*

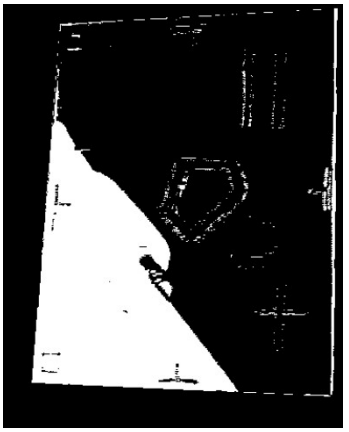

Figure 6: *Occlusion mask computed by the algorithm for the camera image in figure 5. White pixels in the bitmap indicate detected occlusion.*

6. Evaluation of the algorithm

We have devised several methods for evaluating the performance of the algorithm. One important measure of course is the runtime of the algorithm for a single frame. We have found our approach to take between 800 and 1500 milliseconds on the computer that it was developed and tested on. The runtime analysis for an example of a typical camera image is given in Table 1. Note that the detection of correspondences accounts for more than half of the execution time.

The quality of the results produced by the algorithm is another important information. There are two aspects of algorithm quality that can be examined. The first aspect is the correctness of the computed 2D transformation. The better it maps the internal DOB image to its actual position in the camera image, the more reliable is the image comparison. In order to make such an evaluation possible, a software tool for manual identification of DOBs in camera images was developed. Using this manual DOB definition and the DOB appearance suggested by the 2D transformation, a similarity measure can be computed. This similarity measure is in the interval $similarity \in [0;1]$.

Offscreen rendering of DOB	20 msecs
Salient points detection	70 msecs
Searching point correspondences	560 msecs
2D Warping	40 msecs
Findung occlusion (identity criterion)	300 msecs
Filtering occlusion mask	110 msecs
Total time	**1100 msecs**

Table 1: *Typical execution times for algorithm steps*

Using the DOB similarity measure, we performed a number of experiments. In these experiments, we examined the influence of the confidence threshold (see Section 5.4) on the quality of the 2D transformation. In each experiment, the DOB similarities were calculated for a given series of still camera images. The confidence threshold was changed for each test run. Statistical data on the resulting DOB similarities is shown in Figure 7.

All datasets were tested with confidence thresholds of 0.1, 0.3, 0.5, 0.7 and 0.9. Of these, 0.7 obviously is the best value. If a smaller threshold is chosen, it is possible that incorrect point correspondences are added to the equation system. This decreases the quality of the computed 2D transformation. If the threshold is too high, too few correspondences are taken into account, even if they are valid. Then the quality of the transformation can also become poor, or it can even become impossible to compute it at all due to lack of point correspondences.

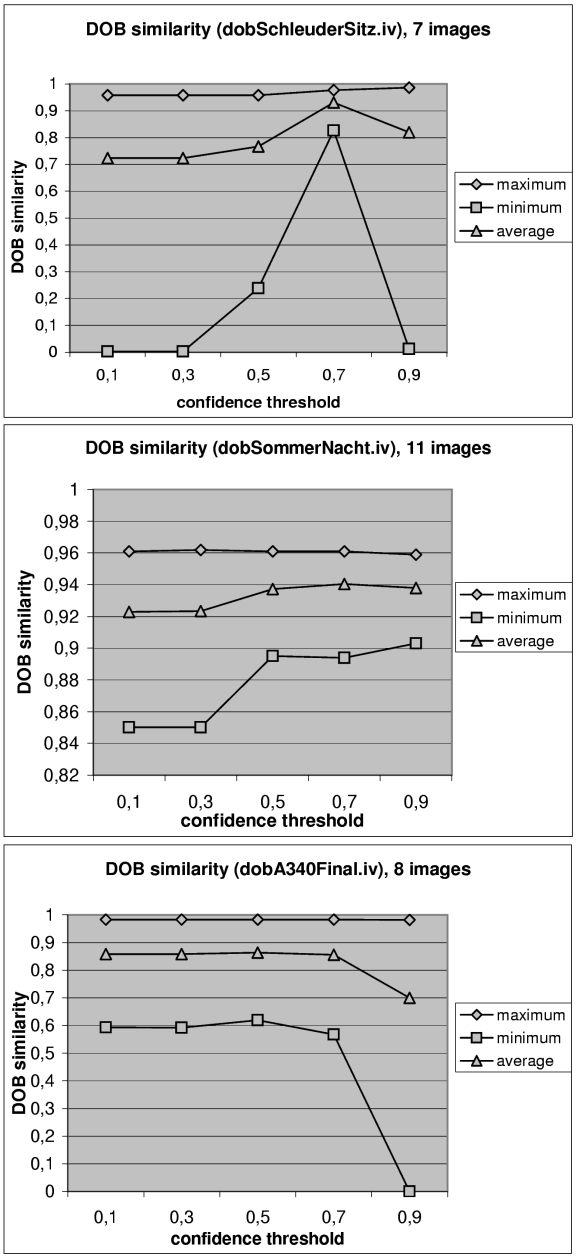

Figure 7: *DOB similarity depending on confidence threshold for three image sequences.*

Another useful quality measure for the algorithm is the similarity between the computed occlusion mask and the actual occluders. This requires manual identification of the occluders in the camera image. We have not yet experimented with that kind of measure.

7. Limitations and drawbacks of the approach

There are some limitations to the approach and our implementation. First of all, it's important to understand that our algorithm is not able to deliver actual depth information for the scene. Whenever a real occluder is detected in front of any DOB, it is assumed to also be in front of all virtual objects. In reality, this is not always the case. A natural object in the scene can be between two virtual objects. Thus, our occlusion detection approach is not suitable for all types of applications. It is primarily designed for interaction with the AR scene using hands or pointing devices, which can mostly be assumed to be closer to the user than virtual objects. Another limitation is that occlusion is only detected in front of DOBs. This can make occluding objects appear clipped at a DOB edge.

The algorithm performance described in Section 6 appears to be not good enough for real-time applications. But given the fact that state-of-the-art computers are several times as fast as the one used for development and testing, and after improvements discussed in Section 8, a considerable speed-up should be possible.

Problems can also arise from the way that the point correspondences are detected. Because of the fixed-size search window, very large marker tracking inaccuracies prevent correspondences from being found at all. How to make the algorithm more stable in this respect is also described in the following section. Finally, the pixel comparison criteria that we have examined do not always deliver flawless occlusion masks. Alternatives are described in Section 8.

8. Possible improvements and future research

Many possibilities for improving the algorithm can be conceived. The detection of salient points can be done in a separate pre-processing step, which stores the points in a separate file. In each frame, the stored points then only have to be projected according to the marker tracking pose. This speeds up the algorithm and allows for more sophisticated detection heuristics.

For the detection of correspondence points there are several alternatives to using a fixed-size search window. The window size could be adaptively increased to account for large marker tracking inaccuracies. Alternatively, detected correspondence points can be iteratively refined on a Gaussian pyramid or using different sub-sampling distances. Salient features could also be detected in the camera image as candidates for the template matching. These methods can increase both the stability and the speed of the correspondence detection, which currently is the most expensive algorithm step as mentioned in Section 6.

The selection of point correspondences can be improved by using the RANSAC method [8] instead of confidence thresholding. A true 3D pose estimation [20] could be performed based on the correspondences instead of computing a 2D transformation.

Finally, the pixelwise image comparison could be perfected by combining several simple comparison criteria. Another promising criterion seems to be the color quantisation approach, in which all colors on the DOB are mapped to a discrete color lookup table. This might speed up the pixel comparison and make it more reliable. Moreover a method for finding coherent occluder areas could be employed. This can be done using a flood-fill algorithm or an image segmentation method like region growing.

Apart from improving the algorithm for detection of dynamic occlusion, another direction of research is also conceivable. The steps of the approach that estimate the transformation between internal DOB and camera image could be used for a simple markerless tracking system. Pre-defined surfaces in the user's environment would then take the place of artificial markers. A combination of these tasks could provide both markerless tracking and occlusion handling based on the techniques described in this paper.

Acknowledgements

We would like to thank Alexander Neubeck for suggestions on some of the math presented here. Valuable support while writing this paper was provided by Dirk Bartz and Ángel del Río, who did the proof reading and commented on its structure.

References

1. R. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997.
2. M.-O. Berger. Resolving occlusion in augmented reality: a contour based approach without 3d reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 91–96, 1997.
3. David E. Breen, Ross T. Whitaker, Eric Rose, and Mihran Tuceryan. Interactive occlusion and automatic object placement for augmented reality. *Computer Graphics Forum*, 15(3):11–22, 1996.
4. J. Fischer. Interaktive Spezifikation von Domänen und Detektion partieller dynamischer Verdeckungen in Augmented-Reality Umgebungen. Diplomarbeit, Universität Ulm, 2002.
5. J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics - Principles and Practice*. Addison-Wesley, second edition, 1997.
6. Anton Fuhrmann, Gerd Hesina, François Faure, and Michael Gervautz. Occlusion in collaborative augmented environments. *Computers and Graphics*, 23(6):809–819, 1999.
7. G. Gordon, M. Billinghurst, M. Bell, J. Woodfill,

B. Kowalik, A. Erendi, and J. Tilander. The use of dense stereo range data in augmented reality. In *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, September 2002.

8. R. Hartley and A. Zissermann. *Multiple View Geometry in computer vision*. Cambridge University Press, 2000.
9. Intel Corporation. *Intel Image Processing Library Reference Manual*, 2000.
10. Intel Corporation. *Open Source Computer Vision Library Reference Manual*, 2001.
11. M. Kanbara, T. Okuma, H. Takemura, and N. Yokoya. A stereoscopic video see-through augmented reality system based on real-time vision-based registration. In *IEEE Virtual Reality 2000 International Conference(VR 2000)*, pages 255–262, March 2000.
12. H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of IEEE and ACM International Workshop on Augmented Reality*, pages 85–94, October 1999.
13. H. Kato, K. Tachibana, M. Billinghurst, and M. Grafe. Real-time tracking system based on matching templates generated from image texture (demo). In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, September 2002.
14. G. Klinker. Praktikum augmented reality - occlusion handling between real and virtual objects. http://wwwbruegge.in.tum.de/teaching/ss01/AR-Praktikum01/presentation/occlusions.pdf, 2001.
15. V. Lepetit and M.-O. Berger. A semi-automatic method for resolving occlusion in augmented reality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2000.
16. S. Malik, C. McDonald, and G. Roth. Hand tracking for interactive pattern-based augmented reality. In *Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, September 2002.
17. B. Paul. Mesa 4.0.4 (readme file). http://mesa3d.sourceforge.net/docs/mesa.html, 2002.
18. H. Regenbrecht, M. Wagner, and G. Baratoff. MagicMeeting - a collaborative tangible augmented reality system. *Virtual Reality - Systems, Development and Applications*, 6(3), 2002.
19. M. Sonka, V. Hlavac, and R. Boyle. *Image processing, Analysis and Machine Vision*. Chapman & Hall, 1993.
20. E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice-Hall, 1998.
21. J. Wernecke. *The Inventor Mentor (Release 2)*. Addison-Wesley, 1994.
22. M. Wloka and B. Anderson. Resolving occlusion in augmented reality. In *Symposium on Interactive 3D Graphics*, pages 5–12, 1995.