Resolving Occlusion in Augmented Reality: A Contour Based Approach without 3D Reconstruction

Topics: augmented reality, resolving occlusions.

Abstract

*We present a new approach for resolving occlusions in augmented reality. The main interest is that it does not require 3D reconstruction of the considered scene. Our idea is to use a contour based approach and to label each contour point as being "*behind*" or "*in front of*", depending on whether it is in front of or behind the virtual object. This labeling step only requires that the contours can be tracked from frame to frame. A proximity graph is then built in order to group the contours that belong to the same occluding object. Finally, we use some kind of* active contours *to accurately recover the mask of the occluding object. Significant results prove the effectiveness of our approach.*

1 Introduction

Augmented reality systems aim at enhancing the user's vision with computer generated images [7, 6, 2]. Most of the time, such systems simply overlay computer generated images and only attempt to minimize object registration errors. However, such methods are effective only when there are no occlusions between real objects and computer generated objects.

If a real object occludes a virtual object V, we must compute the occluding mask m_V, that is the region of the image plane to which the visible part of V corresponds. The virtual object is then only displayed on this mask. Most of the time, occlusions are resolved interactively [3, 2] by allowing the user to delineate the occluding mask. Hence, solving automatically the occlusion problem for augmented reality is a challenge, especially when little is known about the world to be augmented.

Context:
We are only concerned with static scenes. Satisfying image composition requires a good temporal registration between the real scene and the virtual objects, as the camera (or the user) moves. This can basically be achieved using two approaches. The first solution is to use position sensors; but instrumenting the real world is not always possible. The other solution is to use modeled objects to perform registration.

As we do not want to modify or to instrument the environment with which we interact (especially because we often work with outdoor environments), we use model based registration. This implies that the 3D model of some features in the scene is known. In fact, this is not too restrictive: for numerous applications, the main structures of the scene are often known (ground, main objects...). The transformation between the camera and the scene frame can then be computed using these features.

Let us now get back to the occlusion problem itself.

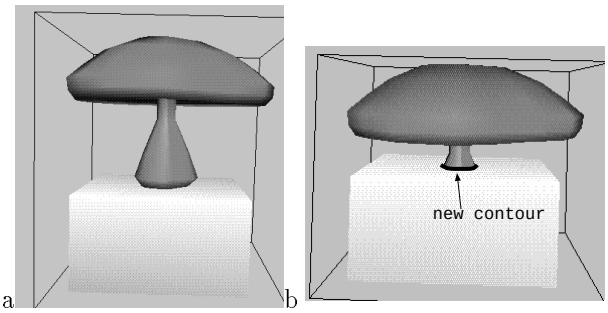
Related works
While several augmented reality systems are described in the literature, few of them address the occlusion problem. If the model of the 3D scene is known, as in [3], the problem can easily be solved. Otherwise, the problem is very difficult. Theoretically, resolving occlusion could be achieved by inferring a dense map from a stereo pair (or from two consecutive images) so as to compare the depth of the virtual and that of the real object. Some people [9] have tried to develop a real-time stereo algorithm to be used for resolving occlusions in augmented reality. But their results are visually unsatisfactory. In fact, despite new advances in 3D reconstruction, the depth map lacks accuracy and cannot be used as is.

Instead of performing 3D reconstruction, we advocate a contour-based approach that allows us to recover the boundaries of the mask. Before describing our approach, we focus on some specific problems that may arise when the user want to add a virtual object in a scene.

2 The link between the occlusion mask and the occluding contours

2.1 Adding a virtual object

Adding a virtual object in the scene requires that the user specifies the relationship of this object with the



a b

Figure 1: If the new object (the mushroom) is not consistently positioned (b), the occluding mask contains contours that belong neither to the image contour nor to the virtual object contours.

scene. The user can either give the geometric position of the object in the scene frame, or place visually the object in the image through an interface: the user can then rotate or translate the object in the scene frame until a satisfying position has been reached. In this case, the user does not have at his disposal the 3D map of the scene but only a 2D image (or a sequence), and he can only visually assess the result. Hence, he may sometimes make mistakes while adding an object in the scene. For instance, he may put an object lower than the level ground and unrealistic situations may arise. Consider for instance the problem of adding a virtual mushroom on a real cube (Fig.1). If the position of the mushroom is realistic (Fig.1.a) the occluding mask is only composed of contours present in the scene and of contours belonging to the virtual object. For an unrealistic position (see Fig.1.b, the mushroom should penetrate the cube), the occluding mask of the mushroom contains a new contour and cannot be recovered from the image and from the virtual object. These contours can only be computed if the model of the real scene is known. As mentioned previously, the bad quality of the depth map stemming from a stereo reconstruction does not generally allow the contours to be computed with an acceptable accuracy.

Our conclusion is as follows: if the position of the virtual object is realistic, the occluding mask can be computed using only image data (the edge map) and the occluding contours of the virtual objects. Otherwise, the mask contains some contours that can be recovered only if the 3D model of the scene is known.

Nevertheless, we think that most of the time, the user can consistently specify the position of the object he wants to add in the scene, possibly with small errors. Indeed, visual assessment generally enables collisions between virtual and real objects to be avoided.

We thus assume that the occluding mask is only composed with some parts of occluding contours stemming from objects standing in front of V and parts of the occluding contours of the virtual object.

2.2 Identifying the occluding contours?

Thus, the identification of the occluding contours can be of great help for the computation of the occluding mask. But, it is not an easy problem. Vaillant [8] was the first to propose an identification method: assuming that the surface of the object along the rim is locally a cylinder, he can compute the radius of the cylinder using three consecutive images. The classification between occluding and regular contours [1] is then performed by testing the radius (the radius for a regular edge is theoretically 0). Nevertheless we did not manage to successfully identify the occluding contours for our images with this kind of method. Another alternative for the identification makes use of purposive vision [5]. Unfortunately, this approach is untractable because the observer's motion cannot be controlled for most augmented reality applications.

Due to the inaccuracy of the reconstruction process and to the difficulty of the identification task of the occluding contours, we resorted to an alternative solution, which does not require 3D reconstruction. Using the edge map of the image, we attempt to identify all the contours (regular contours as well as occluding contours) which stand in front of the virtual object. By grouping the contours according to a proximity criterion, we produce a first rough approximation of the occluding mask. This first estimation can possibly be somehow erroneous, as all the contours of the occluding objects are not necessarily extracted by the edge detector. Conversely, some edge parts can be mis-identified and can be labeled as *in front of* although they are behind the virtual object. To cope with these problems, we resort to regularization. More precisely, strategies inspired from the *active contours paradigm* allow us to accurately define the occluding mask.

3 Overview

In this section, we describe the outlines of our algorithm for resolving occlusions using two successive images. Let V be the virtual object to be added in the real scene. We assume that both the motion between the two frames and the camera parameters are known. In our case, the motion is computed using object based registration.

The main steps of our algorithm are summarized below. For each step, we refer the reader to Fig. 5, which

[1] Unlike occluding contours that are viewpoint dependant, regular contours correspond to the same physical event in the 3D scene

illustrates the key points of the algorithm. In the example shown, we attempt to add a virtual rectangle parallel to the frontal plane of the calibration grid. The motion has been computed using the small circles on the calibration grid.

Step 1: Computation of the initial mask
We compute what we call the initial mask m_I, that is the region in the image to which the object V corresponds, assuming that the whole object is visible. We thus compute the occluding contours of the virtual object (Fig. 5.b).

Step 2: Tracking the contours
The contour chains are extracted in the region of the image corresponding to m_I (Fig.5.c) (in fact, we consider all the chains that cross the initial mask; we explain why in section 6). Then these chains are tracked (Fig.5.d) in the next image using a curved- based tracker which we have developed previously (for further details, see [1]) [2]. Finally, the matching of the contours points between the two images is performed by using the epipolar constraint: for each contour point m belonging to a given chain C, the corresponding point is computed as the intersection between the epipolar lines and the corresponding tracked chain C' (as usual, heuristic considerations or constraints are used if several intersections are available).

It must be noticed that a chain can be composed of contours belonging to two (or more) objects (see Fig. 5.c: the side of the book and the occluding contour of the pot are linked by the same chain). It does not matter as the next step allows us to discriminate between the parts of the chain through the labeling stage.

Step 3: Labeling the contour points with *behind* or *in front of*
This important step is detailed in section 4 and is based on the comparison for each image point m between (i) the optic flow of the real point in the scene that projects onto m and (ii) the optic flow of the 3D point belonging to the virtual object that projects onto m (Fig 5.e and f). The originality of this method is that it does not require 3D reconstruction of the scene.

Step 4: Computation of the occluding mask
This step is detailed in sections 5 and 6. The problem is to recover the occluding mask from the contours labeled *in front of*. Using the force field created by these contours, the active contours allow us to detect a first estimation of the mask. This estimation is then refined using the initial image (Fig. 5.g,h,i and j).

[2] Too small chains are not tracked, as the tracking process is not reliable in this case

4 Discriminating between "*behind*" and "*in front of*" points

4.1 The criterion

Fig 2 illustrates the basic geometry of the camera model; for each camera position we consider the coordinate system $\mathcal{R}_1$ (resp $\mathcal{R}_2$) where the z axis is the optical axis and the (x, y) axis are parallel to the image axis.

Let I_1 and I_2 be two consecutive frames. Let $m_1 \in I_1$ and $m_2 \in I_2$ be two corresponding points given by the tracking process. We note Z_{real} the z coordinate of the real point corresponding to (m_1, m_2). Now let M_{obj} be the 3D point belonging to the virtual object,whose projection in I_1 is m_1. This virtual point occludes the real one only if $Z_{obj} < Z_{real}$. We show in the following how to solve that without computing Z_{real}.

Since the calibration is known the projection m_{obj} of M_{obj} in I_2 can be computed. Fig. 2 shows intuitively that the points lying in front of or behind the object can be inferred from the relative position of m_2 and m_{obj} on the epipolar line.

More precisely, let us define f_{m_1} (Fig. 2):

$$f_{m_1} : Z \rightarrow proj_{I_2}(m1_x, m1_y, Z)$$

where $m_1 = (m1_x, m1_y)$ and $proj_{I_2}$ is the projection in image I_2.

We will show that f_{m_1} is an homography of Z whose coefficients only depend on m_1 and of the relative position of the camera between the two frames.

Let u_1, v_1, k_{u1}, k_{v1} (resp u_2, v_2, k_{u2}, k_{v2}) be the intrinsic parameters of the cameras used for the first two frames. The euclidian displacement mapping $\mathcal{R}_1$ onto $\mathcal{R}_2$ is called $[R, T] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$.

Hence, the coordinates in $\mathcal{R}_1$ of the 3D point M that projects onto m_1 are

$$M = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \frac{m1_x - u_1}{k_{u1}} Z \\ \frac{m1_y - v_1}{k_{v1}} Z \\ Z \end{pmatrix} \quad (1)$$

Hence

$$f_{m_1}(Z) = \begin{pmatrix} u_2 + k_{u_2} \frac{r_{11}X + r_{12}Y + r_{13}Z + T_x}{r_{31}X + r_{32}Y + r_{33}Z + T_z} \\ v_2 + k_{v_2} \frac{r_{21}X + r_{22}Y + r_{23}Z + T_y}{r_{31}X + r_{32}Y + r_{33}Z + T_z} \end{pmatrix} \quad (2)$$

Substituting (1) into (2) allows then $f_{m_1}(Z)$ to be expressed as an homographic function of Z whose coefficients depend on the calibration process and on the image point m_1.

Let us return to the occlusion problem. We have

$$m_2 = f_{m_1}(Z_{real}) \quad and \quad m_{obj} = f_{m_1}(Z_{obj})$$

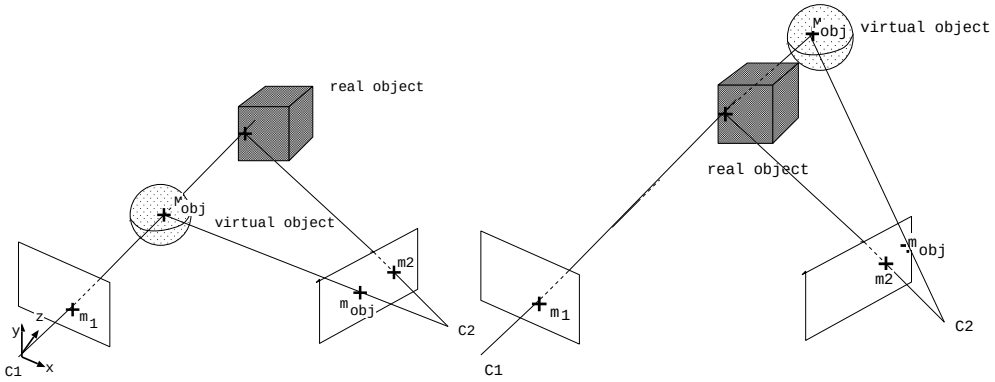


Figure 2: The relative positions of the real and the virtual objects.

Due to the monotony of the homography, it is therefore easy to compare Z_{real} and Z_{obj} : let a, b, c, d be the coefficients of f_m^x, that is $f_m^x(Z) = \frac{aZ+b}{cZ+d}$.
If $(ad - bc > 0)$, then

if $m_2 > \frac{a}{c}$ and $m_{obj} < \frac{a}{c}$, then $Z_{real} < Z_{obj}$

else if $m_2 < \frac{a}{c}$ and $m_{obj} > \frac{a}{c}$, then $Z_{real} > Z_{obj}$

else if $m_2 < m_{obj}$ then $Z_{real} < Z_{obj}$

else if $m_2 > m_{obj}$ then $Z_{real} > Z_{obj}$

Thus, the two homographies f_m^x and f_m^y can be used to decide whether the point is in front of or behind the virtual object. Each point m can therefore be labeled with *behind* or *in front of* only if the test on the two coordinates give the same result. Otherwise, the label is *doubtful*.

4.2 The occluding contours

The preceding criterion has been established for points belonging to regular contours. In fact, we also use it for points belonging to occluding contours. We explain why in Fig. 3: for an occluding contour,the corresponding points m_1 and m_2 belongs to the tangent lines τ_1 and τ_2. Hence m_2 is the projection of the intersection P of the two tangent lines. Then, using the above criterion with (m_1, m_2) leads us to compare the positions of P and V instead of comparing the position of V and P_1. If V does not lie between P and P_1, the result given by the criterion is correct (outside this interval P and P_1 are on the same side of V, i.e. both before or both in front of V, and the criterion succeeds).

Problems may obviously arise if the virtual object could be added between P and P_1. We claim that this is not practically possible for most cases. To prove that, let us compute PP_1. Let R be the radius of the real object. We have

$$PP_1 < \hat{P_1P_2} = R\alpha$$

where α is the angle between the tangents; it is usually small as the motion between two frames is small. We can approximate α with $\alpha \approx arctg(C_1C_2/d)$ where d is the distance between the camera and the object. For our practical case, the distance camera/object is around 2 meters, the distance $C_1C_2 < 5cm$ and the radius of the object in the scene are less than 8cm. Hence $PP_1 < 8 \times arctg(5/200) = 0.2cm = 2mm$. Thus the distance where the criterion fails is small. Moreover, because of visual assessment, the user can hardly attempt to add an object in so small an area.

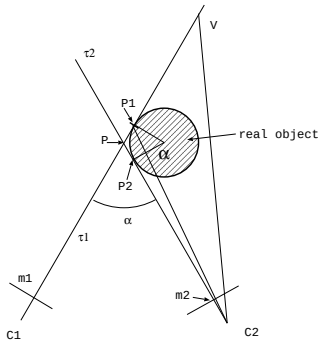


Figure 3: The case of occluding contours

5 Identification of the occluding objects

After the previous step, each contour point is labeled with *behind*, *in front of* or *doubtful*. We now consider the new connected chains $C_{i\,\{1\leq i\leq N\}}$ that are only made up of *in front of* points. Our aim is to group the chains that correspond to the same occluding object (Fig. 5.e).

Our algorithm is based on a proximity criterion. As usual, we define the distance of two curves C_i and C_j by $distance(C_i, C_j) = inf_{\{x\in C_i, y\in C_j\}} d(x, y)$. Given a threshold s, we consider that two chains such that $distance(C_i, C_j) < s$ belong to the same object (in practice s is equal to a few pixels).

We can therefore build a proximity graph $\mathcal{G}$: the nodes represent the chains $C_{i\,\{1\leq i\leq N\}}$; two nodes are connected only if the distance between the corresponding chains is less than s.

Then, detecting the occluding objects amounts to com-

pute the cliques in the proximity graph, that is the sets of curves $\mathcal{H} \subset \mathcal{G}$ such that: $\forall C \in \mathcal{H}, \exists C' \in \mathcal{H} | C$ *and* C' *are connected.*

6 Computing the occluding masks

Inferring the occluding mask from a set $\mathcal{H}$ of contours is an arduous task for several reasons:

- $\mathcal{H}$ may contain some chains that do not belong to the real occluding mask. This may for instance be due to a threshold s which has been chosen too large. Problems can also originate in tracking errors.
- We have to build what we call the *envelop* of a set of contours. But all the contours that composed the occluding mask are of course not necessarily detected in the initial contour map and are of course not tracked. Thus, we have to compute the curve that approximates the set $\mathcal{H}$ as well as possible (Fig 5)

To cope with these problems, we resort to the regularization theory and especially to the active contours models [4].The underlying idea is to add regularity constraints (smoothing constraints) on the solution that will produce a unique solution, and overcome the problems stemming from lacking or erroneous contours.

Active contours are curves minimizing an energy term:

$$\int \alpha |v'(s)|^2 + \beta |v''(s)|^2 ds - \int |\nabla I(v(s))| ds$$

Starting from an initializing curve, the snake will converge towards the nearest edge and produces the regular curve compatible at best with the image.

It is well known that the snakes naturally tend to shrink if the image gradient is null. We take advantage of this property in our application by initializing a snake with a closed curve outside the set $\mathcal{H}$ (In practice, we use the smallest rectangle that contains $\mathcal{H}$ Fig. 5.g). Then, we let the snake evolve under the influence of the field created by the contours in the following way: Let I_0 be the contour image:

$$\begin{aligned} I_0(x,y) &= 255 \; if \; (x,y) \; belongs \; to \; one \\ &\quad of \; the \; chains \in \mathcal{H} \\ &= 0 \; otherwise \end{aligned}$$

Then let $F = I_0 * Gauss(\sigma)$ be the convolution of the contour image with a gaussian kernel with a sufficiently large standard deviation ($\sigma = 2$ for instance), in order to create the influence field.

As the initializing curve contains $\mathcal{H}$, the snake will shrink itself until it reaches the contours of $\mathcal{H}$. This will produce the most regular curve resting on $\mathcal{H}$. In fact, the curve does not always contain all the chains belonging to $\mathcal{H}$. This is quite normal as bumps or spurious contours can be removed by the regularization process.

The advantage of such a method is twofold:

- spurious contours that have been erroneously grouped with the mask are not taken into account most of the time. In fact, they often appear as bumps on the mask boundary and are removed by the regularization process.
- this algorithm prevents us from using tricky geometric algorithms allowing some kind of envelop to be computed.

Remark: We now explain why we use all the points of the chains that cross the mask instead of using only the part of the chain that lies inside the mask region. This is due to the behavior of active contour models in some specific situations: consider the simple scene shown in Fig. 4 and suppose we want to add a rectangle behind the black triangle. The expected result is shown in Fig. 4.b. If we only use the part of the contours which belongs to the initial mask, then the position reached by the snake is the one shown in 4.d and the computed mask (4.e) is not really satisfactory. This behavior is quite normal as the gradient is null on the horizontal part of the mask (Fig. 4.c). Hence the snake is only submitted to the smoothness constraints and it tends to retract itself. This behavior is all the more perceptible as the angle is sharp. This explains why the lower part of the mask is badly detected whereas the upper part is satisfactory

This problem is mainly due to the fact that we only take into account the part of region surrounded by the contour that is contained into the mask of the virtual object. If we consider instead the whole contour (Fig. 4.f), we recover a greater part of the occluding object which allows us to overcome the problem stemming from contours broken by the mask. The final mask is then obtained by keeping the part of the mask that belongs to the initial mask of the virtual object. For instance, in the case of Fig. 4, we will recover the whole occluding object. In the case of Fig. 5, we recover the base of the clown, and this yields a good estimation of the mask.

7 Results

Our algorithm has been extensively explained on the example shown in Fig. 5. Another more complex example is shown in Fig. 6, where the virtual object is occluded by two objects of the scene. For the sake of simplicity, the virtual object is a rectangle that is near both the pot and the clown (the distance is around 1 cm, whereas the size of the scene is around 40 cm). Fig. 6.a shows the result of the classification criterion: the points labeled *in front of* are plotted in red, whereas the points labeled *behind* are plotted in green (a small number of points are labeled *doubtful* and are plotted in yellow). The results are visually correct except for the contours that correspond to letters written on the book: these contours are very jagged and are of course difficult to track with sufficient accuracy. This explains the small errors in the classification. Four objects are found after the grouping step (Fig. 6.b). The blue and the red ones are discarded because the area surrounded by these contours is too small. Using the snake strategies on

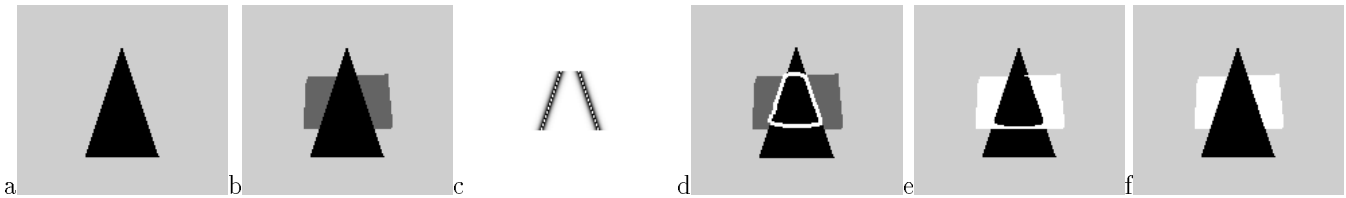


Figure 4: The whole contours are needed to detect the mask properly
(a) the original scene (b) the expected result (c) the gradient field created by the contours (d) the occluding mask reached by the snake (e) the result.(f) the result if we use the whole contour.

the yellow and green objects allows then the two occluding objects to be detected (Fig. 6.c). The final result is quite convincing.

Another observation can be made from this experiment: all the contour points do not appear on Fig.6.a (see for instance the parallel lines on the pot). This is due to the fact that the contour at these points is almost parallel to the epipolar line (which are approximately horizontal here) and the matching fails. In the neighborhood of such points, the matching is not very accurate; this explains why some parts of the parallel lines of the pot are labeled *behind* instead of in front of. Finally, a funny example merging a real scene and strip cartoon heros is shown in Fig.6.e.

8 Discussion

We discuss in this section the robustness of our method and the possible improvements for future works.

8.1 Robustness of the classification criterion

The classification criterion described in section 4 is one of the key points of our algorithm. The effectiveness of this step depends on:

- the quality of the matching:
  if the matching (m_1, m_2) is erroneous, the criterion will obviously fail. The quality of the matching depends on the tracking process and on the accuracy of the computed motion between the two frames. Hence, cluttered or textured environments may cause trouble, as the tracking process is less efficient in these situations.

- the distance between the virtual object V and the scene:
  For a given scene and a given image point m, let d be the 3D distance between V and the corresponding point in the scene. It is quite obvious that the greater d is, the easier the classification is.

- the camera motion:
  Our experiments show that the approach is more effective with large motions. Indeed large motions generally produce more important difference between m_2 and m_{obj}. Moreover, possible imprecisions on the location of m_1, m_2 then have less influence on the classification criterion.

8.2 Concluding remarks

We have presented an alternative to the 3D reconstruction for resolving occlusions in augmented reality. The main interest of our approach is that it only involves 2D computation to check the position of the virtual object against the scene. Resorting to regularization then allows us to compute the mask compatible at best with the classification stage. Moreover, our algorithm is quick, easy to implement and robust.

Of course, our approach depends on the quality of the contour map. Some lacking contours may have no influence on the final result as the snake process fills in possible holes in the contours. Conversely, one missing contour may have a great influence on the result. Consider for instance the case of the pot in Fig. 6. As all the interesting contours are not always detected by the edge detector, and as the epipolar matching fails, the pot is almost split into two parts. Fortunately, the upper contour of the pot ensures the connection between the two parts of the pot.

Thus, one missing contour may lead the algorithm to detect two occluding objects instead of one object and may lead us to trouble. Increasing the value of s is not a satisfactory solution because it will sometimes lead the process to merge possible close occluding objects. Hence, a possible improvement could consist in multi scale analysis, in order to take advantage of some weak contours that do not appear at high scale.

To conclude, even if our method would probably give less interesting results for very textured environments, it is well suited for images where the contour information is relevant.

References

[1] M.-O. Berger. How to Track Efficiently Piecewise Curved Contours with a View to Reconstructing 3D Objects. In *Proceedings of the 12th International Conference on Pattern Recognition, Jerusalem (Israel)*, volume 1, pages 32–36, 1994.

[2] M.-O. Berger, G. Simon, S. Petitjean, and B. Wrobel-Dautcourt. Mixing Synthesis and Video Images of Outdoor Environments: Application to the Bridges of Paris.

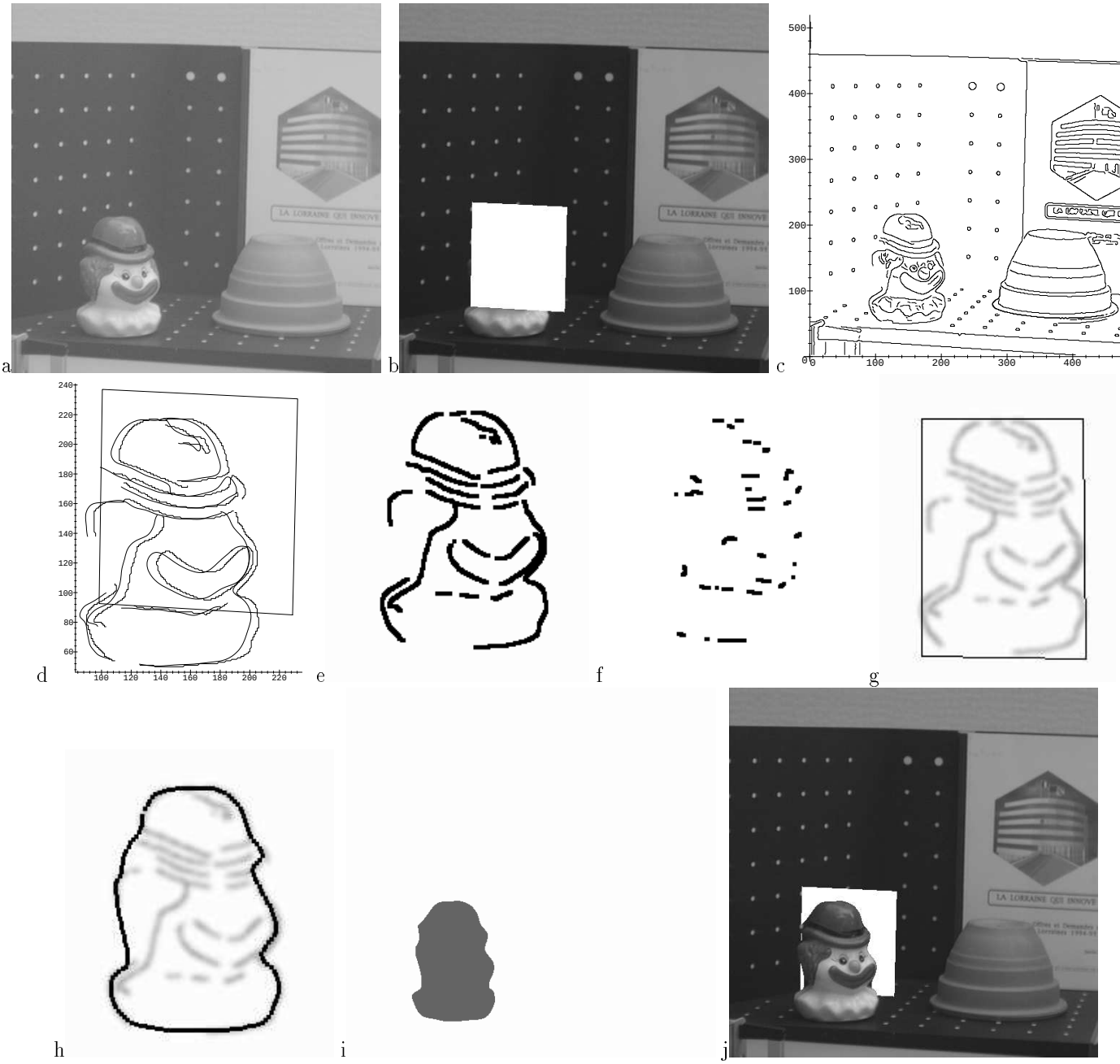


Figure 5: Adding a virtual rectangle in a real scene
(a) the original image (b) the mask of the virtual object to be added overlayed on the image (c) the edge map (d) result of the tracking process (the two corresponding curves are shown), the quadrilateral is the mask of the virtual object (e) contours that are labeled *in front of* (f) mis-classified points: contours that are labeled *behind* (g) the gradient field created by the contours labeled *in front of* and the initialing snake (h) the mask after the snake process (i) the mask of the occluding object (j) The result.

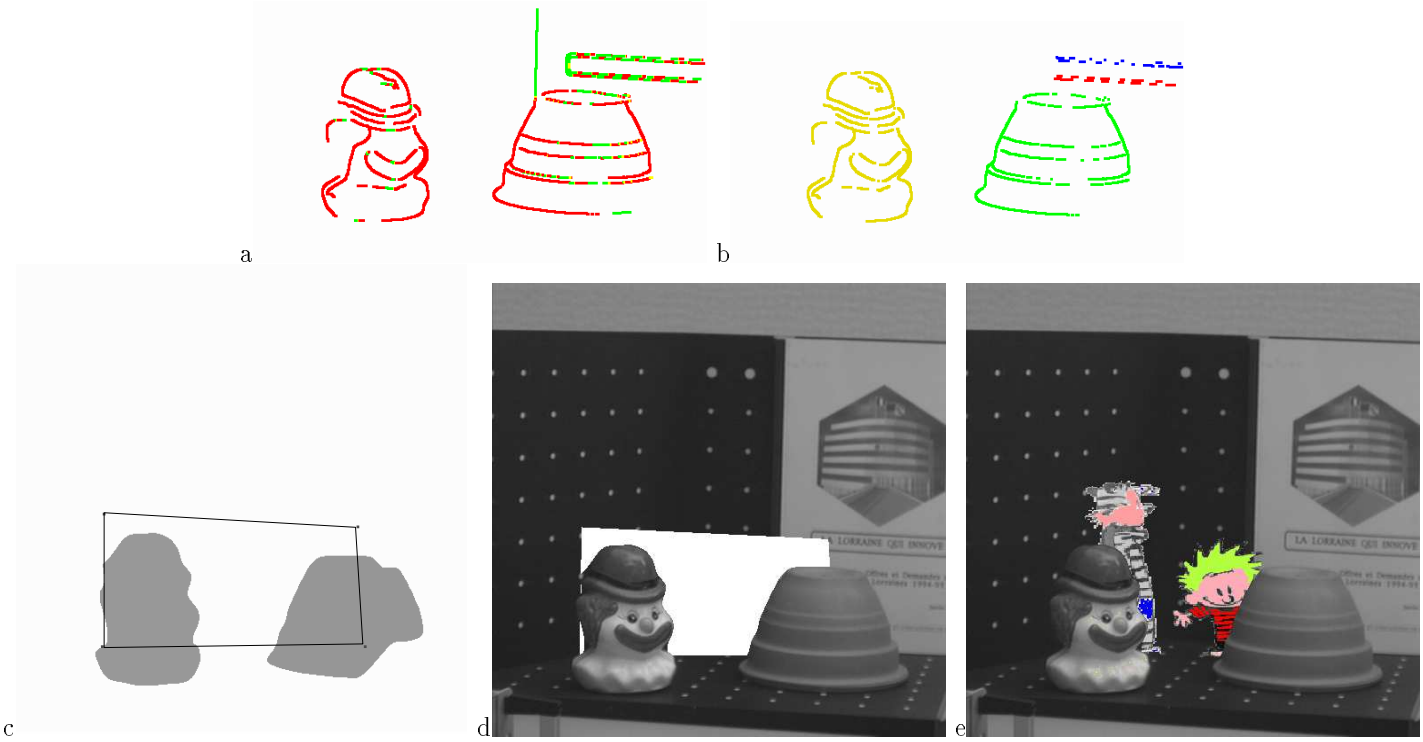


Figure 6: A virtual object occluded by two objects.
(a): the classification stage: the points in red are *in front of* the virtual objet, the points in green are *behind* the virtual object (b) four objects are obtained after the grouping stage (c) The mask of the occluding objects (d) The result of the composition (e) a funny composition

In *Proceedings of the 13th International Conference on Pattern Recognition, Vienna (Austria)*, volume 1, pages 90–94, August 1996.

[3] D. Breen, R. Whitaker, E. Rose, and M. Tuceryan. Interactive Occlusion and Automatic Object Placement for Augmented Reality. In *EUROGRAPHICS'96, Poitiers, France*, 1996.

[4] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1:321–331, 1988.

[5] K. Kutulakos and C. Dyer. Occluding Contour Detection Using Affine Invariants and Purposive Viewpoint Control . In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Seattle, Washington (USA)*, 1994.

[6] S. Ravela, B. Draper, J. Lim, and R. Weiss. Tracking Object Motion Across Aspect Changes for Augmented Reality . In *ARPA Image Undertanding Worshop, Palm Spring (USA)*, August 1996.

[7] M. Uenohara and T. Kanade. Vision based object registration for real time image overlay. *Journal of Computers in Biology and Medecine*, 1996.

[8] R. Vaillant and O. Faugeras. Using Extremal Boundaries for 3-D Object Modeling. *IEEE Transactions on PAMI*, 14(2):157–173, February 1992.

[9] M. Wloka and B. Anderson. Resolving Occlusions in Augmented Reality . In *Symposium on Interactive 3D Graphics Proceedings, (New York)*, pages 5–12, August 1995.