TOWARD 3D FREE FORM OBJECT TRACKING USING SKELETON

Djamel Merad
LIRIS laboratory
5 avenue Pierre Mendes-France, 69676 Bron

Jean-Yves Didier
LSC Laboratory
40 , Rue du Pelvoux, 91020 Evry Cedex

Keywords: Tracking, 3D free form object, Skeletonization, Graph matching.

Abstract: In this paper we describe an original method for the 3D free form object tracking in monocular vision. The main contribution of this article is the use of the skeleton of an object in order to recognize, locate and track this object in real time. Indeed, the use of this kind of representation made it possible to avoid difficulties related to the absence of prominent elements in free form objects (which makes the matching process easier). The skeleton is a lower dimension representation of the object, it is homotopic and it has a graph structure. This allowed us to use powerful tools of the graph theory in order to perform matching between scene objects and models (recognition step). Thereafter, we used skeleton extremities as interest points for the tracking.

1 INTRODUCTION

Object tracking is an important field of artificial vision. It is a difficult problem involved in tasks as motion evaluation or automatic visual control. The possible application are from medical imagery to security, including the road traffic analysis, etc.

The most part of algorithms allowing 3D pose estimation from a 2D images sequence are based on the same principle. The first step consists in recognizing the type of target in order to choose the best model from a database (initialization). The second step, performed during the treatment, consists in target tracking.

An initial pose of the objet is known either by an initialization process before tracking, or by a prediction process using the results obtained from the previous image. This initial pose allows to project the target's model onto the image's plane. Starting from this projection, it is possible to know if the pose is good or not. A comparison measure between the 2D image and the projected model allows a pose adjustment (fitting).

The general principle of tracking is common to all the methods. Nevertheless they significantly vary according to their matching method or prediction techniques. Among the different existing tracking methods, we will cite the one of Lowe (Lowe, 1992) (Lowe, 1987) , of Gennery (Gennery, 1982) (Gennery, 1992) and of Harris (Harris and Stennet, 1990) which we can qualify as generic methods. These three tracking methods give very good performances and use algorithms of different complexities.

Some other methods were developed since. Some model based approaches use the automatic visual control to track the salient edges of a model (Drummond and Cipolla, 2002) while some other methods use primitives as points, lines, or ellipses (Marchand et al., 1999) (Comport et al., 2003). However, the presented methods were been developed for polyhedral models having straight edges, which facilitate their detection. That is the reason why these methods can not be applied on free form object tracking.

To track free form objects, some methods use a priori and a posteriori information. These methods either will index several images (shots) of the target object depending on some points of interest and stock them in a database, or will build a model starting from this camera shot. The objets are tracked using ”patches” together with the indexed images for a homographical matching (Vacchetti et al., 2004). Another way to manage free form object tracking is to use a poor sparse metric environment model in order to perform real time model tracking using the camera (Skrypnyk and Lowe, 2004). Unfortunately, these systems need a learning step on several shots of the real object, they can not use the 3D model only.

The most of the effective tracking methods were

Merad D. and Didier J. (2006).
TOWARD 3D FREE FORM OBJECT TRACKING USING SKELETON.
In *Proceedings of the Third International Conference on Informatics in Control, Automation and Robotics*, pages 74-81
DOI: 10.5220/0001207400740081

developed for polyhedral objects. It would be enough to extract a polyhedral primitive of our free form object and to adapt the existing tracking algorithms. We succeeded by using skeletons for the free form object representation. Indeed, a skeleton is an object representation in an inferior dimension. It has the advantage to preserve the most of the topological and geometrical information of the silhouette. The skeletonization or the object representation by a set of segments, allows us to bypass the problem related to free form objects.

In image processing and computer vision in general, there are many applications for 2D and 3D objects skeletons (encoding, compression etc.). Further to our interesting results obtained with this formalism in free form object recognition and localization , we explored its application to tracking this kind of objects.

More precisely, we adapted the Lowe method (Lowe, 1992) to track a 2D skeleton using a 3D skeleton, therefore to track a 3D object in a 2D shot. The points of interest we used are the two skeleton extremities. The initialization step corresponds to the target pose computing, without any a priori knowledge of the scene. The initialization system is described in section (2) . Once the initial position of our object known, it is possible to start tracking.

The 3D skeleton is projected on the image plane, starting from its initial position, by using previously calculated rotation and translation matrices. The algorithm uses the fact that there is a small displacement between two consecutive images in a video. Therefore the positions of 2D skeleton points in two images will be very close. A comparison measure between the 2D skeleton of the image and the projected 3D skeleton of the model is used for a pose adjustment. This step is presented in section 3 and the obtained results are presented in section 4.

2 INITIALISATION

In this section we present the identification and the localization of 3D free form objects, starting from a 2D image of scene, where partial occlusions may appear.

We analyzed the architecture of 3D free form more popular recognition systems and we noticed the difficult problems to which they are confronted with.

Some of these difficulties are: to build the model for complex objects, primitive extraction allowing a correct matching and the lack of salient elements allowing a robust pose estimation. We also can cite complexity and occlusion problems which have not a satisfactory solution.

The approach we developed (figure 1), is based on using a 3D skeleton of an a priori known model and

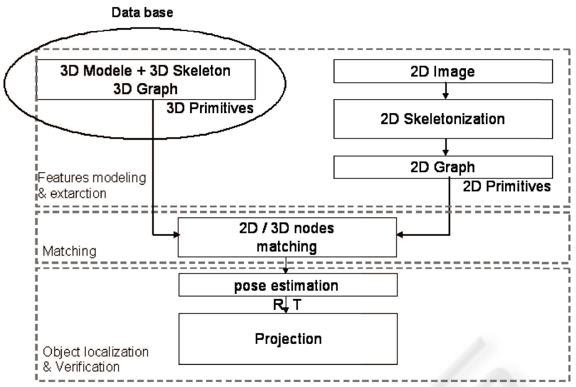


Figure 1: 3D free form object recognition by using skeletons.

the on-line extraction of the 2D skeleton of an image of the object in front of the camera. The utilization of this formalism allowed us to solve the problems presented before: pose estimation and robustness to occlusions and auto-occlusions. Our method is essentially composed of three phases.

- Primitives modelization and extraction: search for an appropriate representation and primitive detection in the image/model.
- Matching: search of the appropriate model for corresponding primitives
- Localization and verification: matching validation and pose estimation.

```
1.  RecoginitionSystem (image I, {3DGraph $G_2$}):
2.      S ⟵ Skeletonization (I);
3.      $G_1$ ⟵ SkeletonToGraphTransform(S);
4.      hypothesis ⟵ Indexation($G_1$, {$G_2$});
5.      for each $h$ from {hypothesis} do
6.          $h'$ ⟵ HypothesisChecking($G_1$, $h$);
7.          if $h' \neq null$ then
8.              OverlayImageWithModel;
9.              exit;
10.         else
11.             $h$ ⟵ NextHypothesis;
12.     display(“No corresponding model for this image”);
```

Figure 2: 2D/3D identification using skeletons.

2.1 Modelization and Primitives Extraction

The skeletonization represent an important step of our method. Since this concept appeared as form descriptor, many skeletonization algorithms have been proposed in the literature. The different skeletonization techniques can be classified in two categories.

The discrete methods, as thinning, ”grassfire” potential fields, the distances maps (Tek and Kimia, 1998) (Nilsson and Danielsson, 1997) (Malandain and Fernandez-Vidal, 1998) (Kegl and Krzyzak, 2002) and the continuous methods, based on Voronoi diagrams (Attali and Montanvert, 1997) (Fabbri et al., 2002).

The discrete domain algorithms are very popular, often used due to their simplicity. Unfortunately the obtained skeletons are often sensitive to image rotation and the connectivity is not always preserved and often needs a post-processing. In the continuous domain, the skeleton is a sub-graph of a Voronoi diagram. The points of the object shapes represent a sampling of the continuous surface. The Voronoi diagram is built starting from this sampling in order to extract the skeleton. This method has several advantages. Only the shape points are needed, which allows to considerably reduce the number of points to be processed. The obtained skeleton is connected and topologically equivalent, because the shape representation of the objects implicitly preserve these aspects. Do not using a discrete coordinates grid, allows us to use the Euclidean distance, therefore the skeleton is not sensitive to image rotation. Another benefit is the skeleton has a graph structure (without post-processing), easy to parse.

Unfortunately there are also some drawbacks for this kind of methods. The main problem is the choice of a sub-graph that represents the nearest approximation of the skeleton. Another problem of the continuous domain comes from the construction of the Voronoi diagram. The diagrams presented in this article are always limited to non-aligned and no cocircular seeds. For real images we have a great number of such cases. This forces us to move the seeds, leading to a bad localization of the skeleton. Furthermore, the skeletons are very sensitive to noise induced by the sampling. Therefore an extra-processing has to be performed to prune the unwanted edges.

We used a hybrid method to compensate for these shortcomings. In this method we combined the two techniques: the one based on distance map and the one based on Voronoi diagram, in order to benefit by their strong points. This skeletonization algorithm has been tested on different images. The experimental results showed that it guarantee the complete connectivity of the skeleton, the homotopy, the geometric invariance and also a good localization. The figures 3 and 4 show the results of this technique on real images.

In order to obtain an efficient matching, it is essential to have a close representation of the model and of the image. In our approach, the skeletonization results in two graphs named 3D and 2D. The 3D graph is obtained from the 3D homotopic skeleton which is itself computed from the 3D model. This represents an off-line operation.

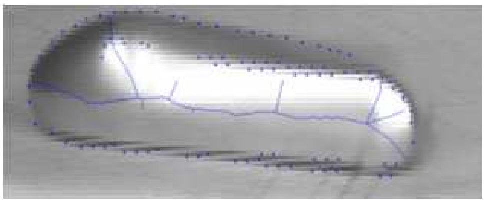


Figure 3: Hull object and 2D skeleton.

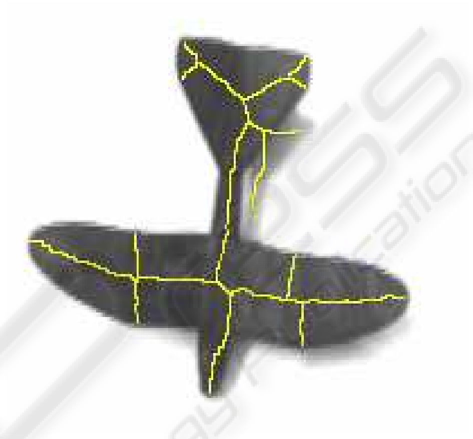


Figure 4: Plane object and 2D skeleton.

An on-line skeletonization of 2D scene image (second line of figure 2) allows to build the 2D graph (line 3 of figure 2). Our method code these graphs in such a way that each skeleton graph store topological and geometrical information of the initial form. Indeed, each node of the graph represents a skeleton element that is a 2D segment for the 2D graph and a 3D segment for the 3D graph. Each edge in the graph represents a link between the different elements of the skeleton (figure 5).

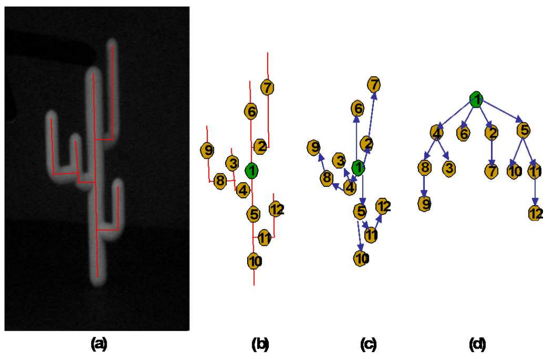


Figure 5: Graph construction from the skeleton.

2.2 Matching

The recognition consists in establishing an isomorphism between the 3D graph from the database and the 2D graph obtained from the image. The problem is to establish a good quality measure for graph matching. In our application, this task is difficult because the measure has to evaluate the similarity degree between similar structures of two sub-graphs, in order to manage the occlusion, shadow problems. Therefore, it has to allow, in the recognition step, to quickly choose the best model from the database (line 4 in figure 2) and also to perform a one to one node matching, in order to locate and validate this hypothesis (lines 5 to 12 in figure 2).

After studying several graph matching algorithms, we implemented an isomorphism-based method using the topological signatures of each node. This approach was used by Shokoufandeh (Shokoufandeh and Dickinson, 1999), Siddiqi (Siddiqi et al., 1999) and Macrini (Macrini et al., 2002) to match two shock graphs. We introduced several modifications on this method in order to obtain a robust matching algorithm at the final nodes level and also to carry out the nodes ”polygamy” i.e. a node can be matched at the same time with several nodes.

In order to topologically describe a tree we used graphs eigenspaces. We remember that each graph can be represented as a adjacency matrix of $\{0,1\}$. The values 1 give adjacent nodes of the graph (and 0 on the diagonal). The eigenvalues (EV) of the adjacency matrix of the graph store some significant structural properties of the graph (tree).

Explicitly, let T be a tree of maximum degree $\Delta(T)$ and $T_1, T_2 \ldots, T_s$ the sub-trees of its root. For each sub-tree T_i, the degree of its root is $\delta(T_i)$. To calculate this signature, we compute the eigenvalues of each adjacency matrix of each sub-tree T_i. Let S_i be the sum of $\delta(T_i)$ eigen values of T_i. Then the ordered elements S_i, become the components of the vector χ of dimension $\Delta(T)$ called topological signature and assigned to the tree's root. If the elements number S_i is lower than $\Delta(T)$, then the vector is filled with 0. This method is recursively repeated in order to assign a vector to each node of the tree.

As we already mentioned, a trees isomorphism can not exist between the image skeleton and the model skeleton because of the occlusions or/and the noise. The solution consists in finding a maximal cardinality and minimal weight matching in a bipartite graph covering the nodes of the two skeletons. The bipartite graph is a graph where each edge is pondered by the topological distance.

We remember that our algorithm allows in a first time to index the object's skeleton in a database - which corresponds to the recognition step. In a second time, it allow a one-by-one matching of the nodes needed in the localization and verification step.

2.3 Projection and Verification

In the precedent step, due to the noise, there are several matching hypothesis. To verify a hypothesis validity we will estimate the rigid transformation between the recognized (presumed) model and the camera. The model projection on the image allows to accept or reject this hypothesis. Each matched couple corresponds to a segment couple, one of the 2D skeleton and the other of the 3D curvilinear skeleton. We project the primitives of the 3D object model on the image plane. If the projected primitives do not coincide with the image primitives, according to a pre-established threshold, then the hypothesis is rejected and we treat the next hypothesis and so forth. If the points of the 3D skeleton projected on the image do not coincide with the points of the 2D skeleton, then the object is not recognized.

This verification module is based on the measurement of this error (1) which is the distance in the image (in pixels) between, each time, a 2D skeleton point and a 3D skeleton point projected on the image.

$$error = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left((\hat{u}_i - u_i)^2 + (\hat{v}_i - v_i)^2\right)} \quad (1)$$

Where : N is the number of matching points,
(u_i, v_i) the coordinates of a point image,
$(\hat{u}_i, \hat{v}_i)$ estimation of the object's point after projection on the image plane.

The 3D pose estimation consists in finding the rigid transformation (R, T) minimising some calculated error (as the sum of error squares) of the one of two collinearity equations (in the image space or in the object space). The two methods generally used to solve this problem are the Gauss-Newton and the Levenberg-Marquardt methods (Lowe, 1991).

We used the method of Lu and al(Lu et al., 2000) named Orthogonal Iteration (OI) algorithm. Contrary to classical methods, used to solve the optimisation problems on the whole, the OI algorithm cleverly exploits the specifical structure of the 3D pose estimation problem.

To estimate the object's pose, this algorithm uses an appropriated error function defined in the object's space. The error function is rewrited in order to accept an iteration based on the classical solution of the 3D pose estimation problem, called absolute orientation problem.

This algorithm gives exact results and converges quickly enough, therefore it is very interesting for real-time applications.

The figures 6 and 7 present some initialization results. On each figure, from left to right, there are the real object and the 3D model recognized by our algorithm and superposed on the real image. There are 10 models in the database and the tests showed that our methods can correctly recognize and localize several 3D objects even in presence of occlusions or auto-occlusions and shadows (due to the strength of our subgraph matching algorithm).

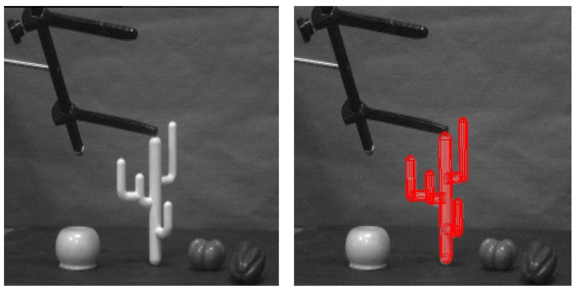


Figure 6: Cactus object recognition among several objects.

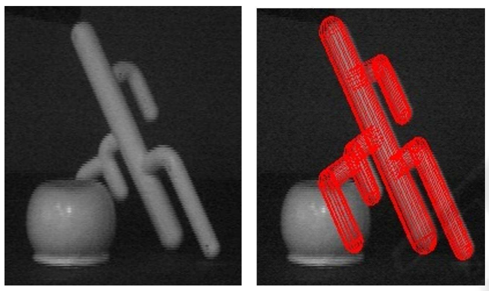


Figure 7: Cactus object recognition with occlusion.

3 TRACKING

The initialization step coresponds to target pose computing, without a priori knowledge of the scene. Once the initial position of our object known, the tracking process can start.

```
1.  Tracking(ImageSequence{$I_i$}):
2.    {R, T} ⟵ InitialValues ;
3.    while tracking not finished do
4.      $i \longleftarrow i+1$;
5.      {$E_1$}⟵ 3DSkeletonEndpointsProjection;
6.      {$E_2$}⟵ 2DSkeletonComputation( $I_i$);
7.      for each Endpoint $e_1$ of $E_1$ and $e_2$ of $E_2$ do
8.        d ⟵ DistanceComputation($e_1$,$e_2$);
9.      {(2D, 3D)} ⟵ CoupleChoice(d_min);
10.     {R,T} ⟵ NewPoseComputation ({(2D, 3D)});
```

Figure 8: Tracking algorithm using skeletons.

The original position we used is the one calculated on the first image during the initialization step (line 2 of figure 8). The 3D skeleton is then projected on the image plane, starting from this position, by using the rotation and translation matrices calculated before (line 5 of figure 8). The algorithm is based on the fact that there are small movements between two video frames. The positions of 2D skeleton points will be very close from one frame to another.

We call p_{ij} the positions of the extremities j of the 2D skeleton in the frame i, numbered from 1 to n_i (their number can vary with the frame). We call q_k the positions of the projections on the image plane of the 3D skeleton extremities, built using rotation and translation matrices R and T respectively.

We suppose the initialization was correct, therefore at the time of the projection of the 3D skeleton, the points q_k which were used to compute the position, were near their matching points p_{ij}. In the next frame (number $(i+1)$), the points $p_{(i+1)j}$ are different from points p_{ij}. Anyway, the deviation of the object being small, there is also a small variation for the 2D skeleton position. This way the points q_k which, by construction, were very close to some points p_{ij}, will be very close to some points $p_{(i+1)j}$. The movement of our object being very small, the old and the new 2D skeleton are very close one of the other, therefore theirs extremities are also very close.

By measuring the pixel differences separating the points $p_{(i+1)j}$ and q_k, and sorting in ascending order, we can form a number of pairs $(p_{(i+1)j}, q_k)$. Whereas during initialization matching between the extremities of the 2D and 3D skeleton required heavy computing, due to the fact that the position was unknown, it is almost immediate here by considering the points 3d at the origin of q_k (lines 6 to 9 in figure 8). The orthogonal iteration algorithm we used for initialization is applied on the pairs (2DPoint , 3DPoint). The 3D points being obtained by replacing q_k by the original points of the 3D skeleton. This provides us a new value for R and T, which could be used in new computations on the next frame (line 10 of figure 8).

We improved the Lowe method by integrating the step of the graph matching in the tracking loop. For example, in the case of an abrupt and irregular movement or of an important occlusion, the model loses the object. While applying, our algorithm of graph matching we succeed to quickly re-localize this object. We remind that in this step we already know which 3D graph corresponds to the 2D graph, in that case, it is sufficient to make a one-to-one nodes matching of two graphs. This step is very fast because the skeletons graphs are small. This procedure is not called during all the tracking process but only when the projection error is greater then threshold. The algorithm of the figure 8 becomes as the algorithm of the figure 9.

```
1.  Tracking(ImageSequence$\{I_i\}$):
2.      {R, T} ⟵ InitialValues ;
3.      while tracking not finished do
4.          $i \longleftarrow i + 1$;
5.          $\{E_1\}$⟵ 3DSkeletonEndpointsProjection;
6.          $\{E_2\}$⟵ 2DSkeletonComputation( $I_i$);
7.          for each Endpoint $e_1$ of $E_1$ and $e_2$ of $E_2$ do
8.              d ⟵ DistanceComputation($e_1$,$e_2$);
9.          D ⟵ $\sum(d)$;
10.         if D > Threshold then
11.             {(2D, 3D)}⟵1To1Matching(2DS,3DS);
12.         else
13.             {(2D, 3D)} ⟵ CoupleChoice(d_min);
14.         {R,T}⟵NewPoseComputation({(2D,3D)});
```

Figure 9: New tracking algorithm.

4 RESULTS OF SKELETON TRACKING

Tracking was implemented in C++ in order to realise real-time computing. There is a file in specific format allowing the passage from the initialization step, realized in Matlab, to tracking. This file contains the rotation and translation matrices corresponding to the calculated position, as well as the model of the 3D skeleton. The tests were realized on syntetic images in order to validate our hypotesis. Using syntetic images allows to compare and estimate projection errors more precisely, because we know the exact pose used to generate our video. Moreover, that let us to generate various types of movement in various configurations. Let us specify that although the used object is a virtual one, its tracking in a sequence of frames is made realistic (difficult) by simulating shades, reflections, noises, self-occlusions. The videos were built with POV-Ray software to generate the images and Virtual Dub to realize the video editing. To read the video stream we used Intel's OpenCV library. Its main advantage is that it integrates some image processing functions.

We did several tests and following we will present only a part. The figure 10 shows the tracking result on the cactus object, in a 68 first frames sequence. This figure represents, from left to right and from up to down, the frames 1, 8, 16, 24, 32, 40, 48, 54, 67 respectively. We present here the most complicated case. We simulated a complex movement with a strong rotation around axis Z (optical axis of the camera). The task becomes difficult because of the disappearance of a part of the object. The tracking is carried out at the rate of 19 Hz by a wireframe model (in magenta). The points of interests representating the extremities of the skeletons are red for the 3D points, and green for the 2D points. The graph in the figure 11(a) represents the reconstruction errors (in pixels) for this example. On the whole, the tracking is carried out correctly, it happens sometimes that the model loose the object and recapture it in the next sequence - image 2 of the figure 10 (frame 8) (due to the integration of the graph matching procedure). We can also observe this on the graph of the figure 11(a).

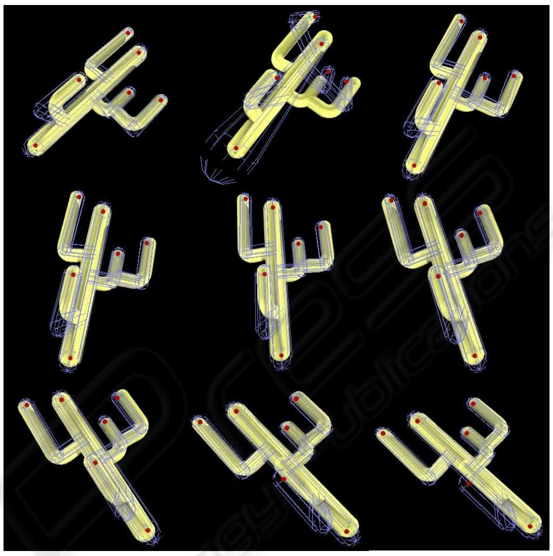


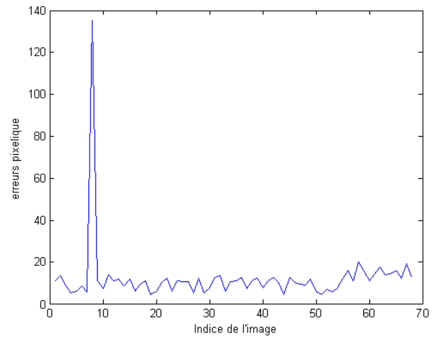
Figure 10: Object tracking sequence.

Another example is presented in the figure 11(b) which represents the reconstruction errors of the first 100 frames of a video (on a total of 2280 frames) in which we simulated a translation following x and y axes and a slight rotation following z. The figures 12, 13 and 14 represent the frames 27, 48 and 78 respectively. We also notice that tracking is correct.

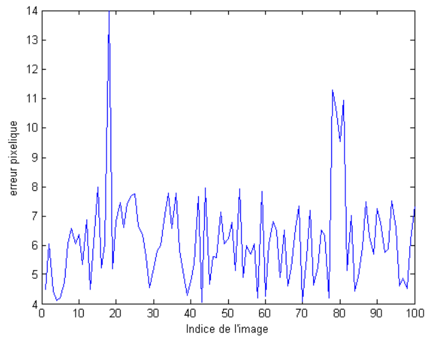
However, in some videos, the algorithm is not able to efficiently track the object (figure 15). This is essentialy owed to the absence of a sufficient number of matchings occuring in the case of occlusions or self-occlusions. The 3D pose parameters are not correctly computed. We remember that the used ortogonal iterations algorithm needs at least four good non-coplanar matchings. However, after few frames our algorithm recaptures the object and tracks it efficiently due to the integration of the graph matching procedure.

5 CONCLUSION

In this article we presented a model based method for 3D free form object tracking. The results of adapting our skeleton recognition method to object tracking were encouraging and justify our approach. Nevertheless we are conscient that some improvements



(a) During tracking in -R-



(b) During tracking in -RT-

Figure 11: Reconstruction error in pixels.

are needed for efficient tracking. In our future works we will focus on a 2D robust and fast skeletonization method. We also project to compensate the weakness of the Lowe method by adding a prediction step in our algorithm.

REFERENCES

Attali, D. and Montanvert, A. (1997). Computing & simplifying 2d & 3d continuous skeletons. In *Computing Vision and Image Understanding, Vol 3, N 67*, pages 261–273.

Comport, A. I., Marchand, E., and Chaumette, F. (2003). A real-time tracker for markerless augmented reality. In *Proceedings of the The 2nd IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 36–45.

Drummond, T. and Cipolla, R. (2002). Real-time visual tracking of complex structures. In *IEEE Transactions on Pattern Analysis and Machine Intelligence vol. 24, N. 7*, pages 932–946.

Fabbri, R., Estozi, L., and da F. Costa, L. (2002). On voronoi diagrams and medial axes. In *Journal of Mathematical Imaging and Vision, Vol 1, N 17*, pages 27–40.

Gennery, D. B. (1982). Tracking known three-dimensionnal objects. In *Conf. American Association of Artificial Intelligence*, pages 13–17.

Gennery, D. B. (1992). Visual tracking of known three-dimensionnal objects. In *International Journal of Computer Vision vol. 8*, pages 243–270.

Harris, C. and Stennet, C. (1990). Rapid, a video rate object tracker. In *British Machine Vision Conference*, pages 73–77.

Kegl, B. and Krzyzak, A. (2002). Piecewise linear skeletonization using principal curves. In *IEEE Transactions on Pattern Analysis and machine Intelligence,vol. 1, N 24*, pages 59–74.

Lowe, D. G. (1987). Three-dimensional object recognition from single two-dimensional images. In *Artificial Intelligence, vol. 31, no. 3*, pages 355–395.

Lowe, D. G. (1991). Fitting parametrized three-dimensional models to images. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 13, N 5*, pages 441–450.

Lowe, D. G. (1992). Robust model-based motion tracking through the integration of search and estimation. In *International Journal of Computer Vision vol. 8, no. 2*, pages 113–122.

Lu, C., Hager, G. D., and Mjolsness, E. (2000). Fast and globally convergent pose estimation from video. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 22, N 6*, pages 610–622.

Macrini, D., Shokoufandeh, A., Dickinson, S., Siddiqi, K., and Zucker, S. (2002). View-based 3-d object recognition using shock graphs. In *Proc. 16 Th Internatonal Conference on Pattern Recognition*, Quebec.

Malandain, G. and Fernandez-Vidal, S. (1998). Euclidean skeletons. In *Image and Vision Computing, vol. 16*, pages 317–327.

Marchand, E., Bouthemy, P., Chaumette, F., and Moreau, V. (1999). Robust real-time visual tracking using a 2d-3d model-based approach. In *International Conference on Computer Vision vol. 1*, pages 262–268, Greece.

Nilsson, F. and Danielsson, P. (1997). Finding the minimal set of maximum disks for binary objects. In *Graphical Models and Image Processing, N 59, vol. 1*, pages 55–60.

Shokoufandeh, A. and Dickinson, S. (1999). Applications of bipartite matching to problems in object recognition. In *Pro., ICCV Workshop on Graph Algorithms and Computer Vision*.

Siddiqi, K., Shokoufandeh, A., Dickinson, S. J., and Zucker, S. (1999). Shock graphs and shape matching. In *International Journal of Computer Vision, Vol 35, N 1*, pages 13–32.

Skrypnyk, I. and Lowe, D. (2004). Scene modelling, recognition and tracking with invariant image features. In *Proceedings of the The 3rd IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 110–117, Arlington,VA.

Tek, H. and Kimia, B. (1998). Curve evolution, wave propagation and mathematical morphology. In *fourth International Symposium on mathematical Morphology*.

Vacchetti, L., Lepetit, V., and Fua, P. (2004). Stable real-time 3d tracking using online and offline information. In *IEEE Transaction on Pattern Analysis and Machine, vol. 26, N 10*, pages 1385–1391.

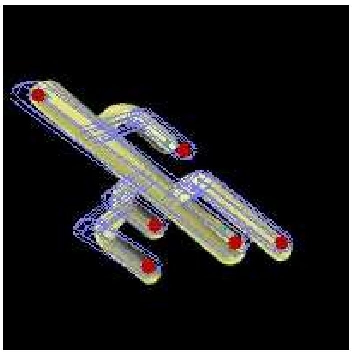


Figure 12: Frame 27.

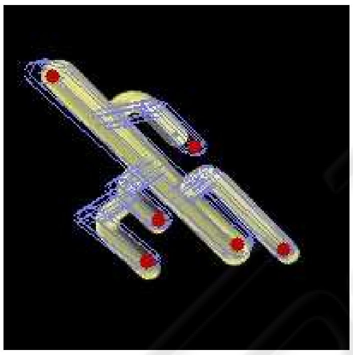


Figure 13: Frame 48.

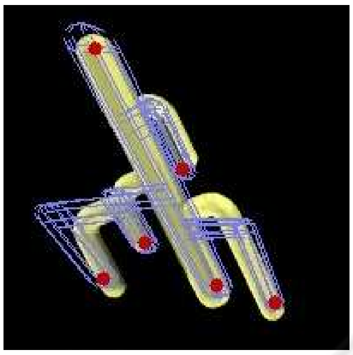


Figure 14: Frame 78.

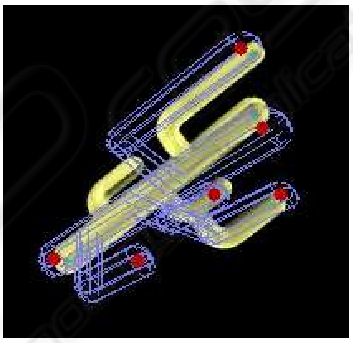


Figure 15: Incorrect tracking.