A two-stage robust statistical method for temporal registration from features of various type

Gilles Simon, Marie-Odile Berger

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A two-stage robust statistical method for temporal registration from features of various type

Gilles Simon, Marie-Odile Berger

N ° 3235

Août 1997

THÈME 3

ISSN 0249-6399

INRIA
LORRAINE

A two-stage robust statistical method for temporal registration from features of various type

Gilles Simon, Marie-Odile Berger

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet ISA

Rapport de recherche n° 3235 — Août 1997 — 33 pages

Abstract: A model registration system capable of tracking an object, the model of which is known, in an image sequence is presented. It integrates tracking, pose determination and updating of the visible features. The heart of our system is the pose computation method, which handles various features (points, lines and free-form curves) in a very robust way. It consists in using robust estimators in a two-stage process: a robust residual is computed for each feature and a robust function of these residuals is minimized. Our method is therefore able to give a correct estimate of the pose even when tracking errors occur. This method is also used to update the set of model features along the sequence, by finding the 2D contours that are tracked when new 3D features become visible. The reliability of the system has been shown on a 300-image sequence of a bridge shot at dusk time, for an augmented reality project.

Key-words: Pose computation, robust estimation, temporal registration, augmented reality, tracking.

(Résumé : tsvp)

This report is an extended version of the paper published in ICCV'98.

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY (France)
Téléphone : (33) 83 59 30 30 – Télécopie : (33) 83 27 83 19
Antenne de Metz, technopôle de Metz 2000, 4 rue Marconi, 55070 METZ
Téléphone : (33) 87 20 35 00 – Télécopie : (33) 87 76 39 77

Une méthode statistique robuste à deux niveaux pour le recalage temporel à partir de primitives de type différent

Résumé : Nous présentons un système de recalage temporel 3D/ 2D, capable de suivre dans une séquence d'images un objet 3D dont le modèle est connu. Il comporte le suivi des primitives, le calcul du point de vue et la mise à jour des primitives visibles. Le coeur de notre système est la méthode de calcul du point de vue, qui prend en compte de façon très robuste des primitives de type différent (points, droites et courbes 3D non nécessairement définies paramétriquement). Cette méthode consiste à utiliser des estimateurs robustes dans un processus à deux niveaux : un résidu robuste est calculé pour chaque primitive et une fonction robuste de ces résidus est minimisée. Notre méthode est donc capable de donner une estimation correcte du point de vue même lorsque des erreurs de suivi se produisent. Elle est aussi utilisée pour mettre à jour l'ensemble des primitives prises en compte sur toute la séquence, en déterminant les contours 2D qui sont suivis lorsque de nouvelles primitives 3D apparaissent. Nous montrons la fiabilité de notre système dans le cadre d'un projet de réalité augmentée.

Mots-clé : Calcul du point de vue, estimation robuste, recalage temporel, réalité augmentée, suivi.

1 Introduction

The problem

Our aim is to build a model registration system capable of tracking an object, the model of which is known, in an image sequence. Such systems are of great interest for augmented reality applications, in particular when virtual objects must be overlaid on an image sequence or when an object in the scene must be replaced with another one [Berger 96, Ravela 96, Uenohara 96].

The registration system is initialized with a user specified set of model/image correspondences and known camera parameters. These correspondences are used to estimate an initial pose. Once initialized, the system must be able to automatically track features in the images and to compute the pose from their correspondences with the 3D model. Since the world around us is not piecewise planar, significant features that can be extracted in an image are often curved contours, especially if we consider outdoor environments. Thus, the features we considered for pose computation are points, lines and curves.

It is generally assumed that correspondences are maintained during tracking. Unfortunately, tracking errors will sometimes result in a model feature being matched to an erroneous image feature. Even a single such outlier can have a large effect on the resulting pose. For point features, robust approaches allow the point to be categorized as outlier or not. When curved features are considered, the problem is not so simple, as some parts of the 2D curves can perfectly match the 3D model whereas other parts can be erroneously matched (see Figure 5.b for example). To make real progress, it is then necessary to devise a robust algorithm capable of extracting the parts of the features that matched the 3D model.

Besides the pose computation problem, other problems must be solved for maintaining registration over time: as the camera undergoes motion with respect to the object, new features may appear while old ones disappear. Hence, the set of model features that are tracked in the sequence must be dynamically updated. This means that we must determine the new 3D features that become visible in the sequence as well as the corresponding 2D features that will be tracked in the subsequent frames.

Related work

Computing the pose from 2D/3D correspondences often consists in the following steps. First, matching hypotheses are generated. Then, the pose is computed using these correspondences. In the tracking context, a rough estimate of the pose is available (the pose computed for the previous frame). Hence, the correspondent can be searched for in a limited neighborhood of the projected feature.

Since a single outlier can have a large effect on the resulting pose, special care is often taken in the matching process, to reduce possible false matches. For instance, Lowe [Lowe 92] uses a probabilistic criterion to guide the search for correct correspondences, before using a weighted least squares including *a priori* constraints for stabilization. Other methods [Gennery 92, Koller 92] use a velocity model and a Kalman filter to better predict the position of the image feature. Unfortunately the use of a velocity model imposes regularity constraints on the camera motion; this can be inappropriate for augmented reality applications for which the scene is often shot by a moving observer.

We therefore advocate a less constraining approach. Instead of attempting to refine the matching process, we prefer to use a robust statistical method to compute the pose from the matching induced by the tracking process. As a result, we obtain a robust estimation of the viewpoint as well as the parts of the tracked image features that actually correspond to the model. Moreover, this method allows us to update easily the set of tracked features by adding the features that become visible and by removing the ones that disappear.

Our approach has some common points with [Ravela 96]. They also use robust methods for pose computation after the tracking stage. But they only consider point features, whereas we consider points, lines or curved features. Another fundamental difference is in the way of updating the set of tracked features. In [Ravela 96], an aspect graph is used to encode sets of model points that are visible from each aspect. But these points are extracted manually and the aspect tables are constructed off line. Moreover, their tracking strategy based on correlation cannot be extended to complex features.

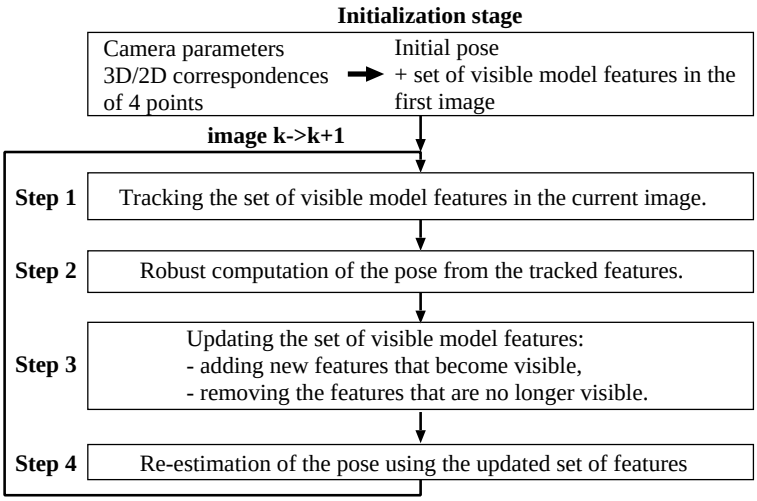


Figure 1: The temporal registration system.

2 System overview

Our temporal registration system is initialized with known camera parameters and a user specified set of 3D features that give rise to easily detectable 2D features. At the beginning of the process, the pose is computed from four 2D/3D corresponding points pointed out by the user, and the 2D features that will be tracked are automatically determined (Figure 3). Once initialized, the system follows a four step loop (Figure 1):

- **Step 1: Tracking the features**
  The set of features is tracked in the current image using a curve-based tracker that we have previously developed [Berger 94]: a rough estimate of the motion field is first computed on the contour to be tracked, by modeling the motion field with a 2D rigid displacement. This estimation is calculated using an iterative method involving only normal optical flows. A prediction of the contour location in the next image can then be computed. By using the active contours from this prediction, the corresponding contour can be detected. We have shown that our method is reliable, fast and that it is able to account for sufficiently large motions. Nevertheless, the tracking may sometimes fail if the snake is attracted by

some local edge maxima: this often results in a curve that is only partially well localized (see feature 4 on Figure 5.b). If the prediction step failed (because of the instability of the flow computation), the position reached is generally completely erroneous (see feature 5 on Figure 5.b). Hence, among the set of tracked curves, a small number may be mis-detected or completely erroneous.

- **Step 2: Computing the pose from various features ($R-ESTIM$)** This step is the heart of our system. It allows the pose to be computed from various features (points, lines, curves) despite possible tracking errors. We use some kind of *Iterative Closest Point* (ICP) algorithm to achieve this task (see section 4).

  Our system stands out from previous works by the use of a robust estimation in a two stage process :

  - **the local stage**: For each feature, we compute the residual, that is the distance between the 2D feature and the projected model feature. Instead of computing the usual average residual (average of the distance between each 2D feature point and the projected model feature), we compute a **robust residual** using M-estimators. As a result, the 2D points that do not correctly match the model feature are not accounted for, as they give rise to too large a distance. Actually, if the percentage of erroneous points is not too large (experimentally a maximum of 20 to 30%), the robust residual can be viewed as the distance between the 2D feature and the projected model for the points that are correctly matched. Otherwise, false matches completely spoil the process and the residual is large.
  - **the global stage** takes into account all the features. The use of a robust estimation (actually, the Tukey estimator) on the above residuals allows to discard erroneous features, *i.e.* features that contain too many false points (we call them *feature outliers*, whereas the other features are called *good features*). It also allows us to discard the portions of the good features that do not match the model feature.

The interest of our algorithm is two-fold: first it allows us to detect the parts of the features that match the 3D model; second, the pose is computed in a robust manner.

The choice of the robust estimators that are used for the local and the global stage is discussed in section 4.

- **Step 3: Updating the visible model features**:
  When the camera undergoes large motion (for instance panoramic motion), it becomes essential to update the set $\mathcal{S}$ of matched model features; otherwise, the number of features that are visible decreases (even becomes null) and the pose cannot be computed with sufficient reliability.

  Instead of searching for all the model features which appear in the image, we only want to detect the ones that give rise to easily detectable 2D features. That is the reason why we rest on the edge map to update the set of features; for each model feature which does not belong to $\mathcal{S}$, we consider all the contours that are sufficiently close to the projection of the model feature. For each contour c, we compute the pose corresponding to $\mathcal{S} \cup c$ using the pose algorithm of step 2. As a result of our algorithm, we can see if the contour c has been discarded from the computation. If not, this means that c is likely to be the projection of the model feature and is added to $\mathcal{S}$. We then provide a simple way to check whether a model feature becomes visible.

  In order to reduce the computational cost, the update process can be only performed every ten images.

- **Step 4: Re-estimation of the pose**:
  Finally, the pose is reestimated by taking into account the new set of model features.

The rest of the paper is organized as follows. In section 3, we review the basic notions of robust estimation. Our pose algorithm is detailed in section 4. Section 5 is dedicated to the update of the set of tracked features. Finally, results on an augmented reality application are shown in section 6.

3 Robust methods

This section gives a brief description of two popular robust methods, used in many computer vision problems [Haralick 89, Kumar 94, Zhang 95]: the *M-estimators* and the *least-median-of-squares* (LMS) techniques. We present here the pose computation problem from the matching of n points, which will be extended in section 4 to the case of n features of various type.

The pose estimation problem

The problem consists in finding the rotation $\mathbf{R}$ and the translation $\mathbf{t}$ which map the world coordinate system to the camera coordinate system. Therefore, if the intrinsic parameters of the camera are known, we have to determine 6 parameters (three for $\mathbf{R}$ and three for $\mathbf{t}$), denoted by vector $\mathbf{p} = [p_1, ..., p_6]^T$. In this section, we attempt to estimate $\mathbf{p}$ from the 2D/3D matching of n feature points, and from an initial estimate of $\mathbf{p}$.

Computation of the residual errors

Let M_i be a 3D point of the model, and m'_i its 2D correspondent detected in the image. If M_i in world coordinates gets mapped to point M_i^c in camera coordinates, we have

$$M_i^c = \mathbf{R}M_i + \mathbf{t}. \tag{1}$$

The 3D point M_i^c projects to an image pixel m_i by the equations

$$\begin{bmatrix} s.m_{i_x} \\ s.m_{i_y} \\ s \end{bmatrix} = \begin{bmatrix} k_x f & 0 & x_0 \\ 0 & k_y f & y_0 \\ 0 & 0 & 1 \end{bmatrix} M_i^c, \tag{2}$$

denoted $m_i = Proj(M_i^c)$, where x_0 and y_0 are the pixel coordinates of the projection of the optical center, $1/k_x$ and $1/k_y$ the horizontal and vertical lengths of a pixel, and f is the focal length. x_0, y_0, $k_x f$ and $k_y f$ can be determined by a calibration process [Faugeras 86].

Using equations (1) and (2), we can express the Euclidean distance between m_i and m'_i as a function of $\mathbf{p}$. Let r_i be this distance:

$$r_i = ||m'_i - Proj(\mathbf{R}M_i + \mathbf{t})||. \tag{3}$$

Robust estimators

The problem consists in finding the vector $\mathbf{p}$ which minimizes the reprojection error. A well-known estimator that could be used to perform this is the *least squares* (LS) estimator, given by

$$\min_{\mathbf{p}} \sum_{i=1}^{n} r_i^2. \tag{4}$$

Unfortunately this estimator is not robust, because the larger the residual is, the larger is its influence on the final estimate.

To tackle this issue, statisticians have suggested many different robust estimators. Among them, the two most popular are the M-estimators and the LMS method, which are not identical with respect to the tolerance to data errors and the accuracy of the estimate. Two measures are hence commonly used to judge the performance of an estimator :

- the *breakdown point*, which is the smallest fraction of outliers present in the input data which may cause the output estimate to be arbitrarily wrong,
- the *relative efficiency*, which is defined in Kim *et al.* [Kim 89] as the ratio between the lowest achievable variance for the estimated parameters and the actual variance provided by the given method (so that the best possible value is 1). In other words, this measure reveals the accuracy of an estimator.

M-Estimators. *M-estimation* techniques, developed by Huber [Huber 81], minimize the sum of a function of the residuals:

$$\min_{\mathbf{p}} \sum_{i=1}^{n} \rho(r_i), \tag{5}$$

where ρ is a continuous, symmetric function with minimum value at zero. Differentiating (5) with respect to the 6 parameters of $\mathbf{p}$ and setting the results equal to 0, the following equations are obtained:

$$\sum_{i=1}^{n} \psi(r_i) \frac{\partial r_i}{\partial p_j} = 0, \text{ for } j = 1, ..., 6, \tag{6}$$

where the derivative $\psi(x) = \mathrm{d}\rho(x)/\mathrm{d}x$ is called the *influence function* (it occurs as a weighting function in equations (6)).

Table 1 lists a few commonly used influence functions and their graphic representation (in the following, we will denote the M-estimator associated with Cauchy's function by M-Cauchy, and likewise for the other functions). For LS, the influence of a datum on the estimate increases linearly with its residual, which confirms the non-robustness of this estimator. By contrast, when using robust estimators, the influence of a datum is constant (M-Huber), decreases (M-Cauchy), or even is null (M-Tukey) when its residual is larger than a threshold c (c is taken equal to kS, where k is a constant, and S a scale factor which corresponds to the standard deviation $\hat{\sigma}$ of the residuals - see below).

These functions have high efficiencies (typically more than 0.9), but have breakdown points of $1/(p+1)$ or lower [Rousseeuw 87], where p is the number of unknowns ($p = 6$ for the pose estimation problem). They are hence well recommended when the presence of outliers in the data is kept below approximately 20%.

LMS / LTS. Another robust technique suggested by Rousseeuw and Leroy [Rousseeuw 87], called the *least median of squares* (LMS), consists of minimizing the median of the squared residuals:

$$\min_{\mathbf{p}} \mathrm{med}_i r_i^2. \tag{7}$$

Minimizing the median ignores the errors of the largest ranked half of the data elements. Thus, this method is able to handle data sets which contain less than 50% outliers (it has therefore a breakdown point of 0.5). However, the LMS has a low efficiency (see [Rousseeuw 87] for more details).

To repair this, [Rousseeuw 87] proposes the *least trimmed squares* (LTS) estimator, given by

$$\min_{\mathbf{p}} \sum_{i=1}^{h} r_{o(i)}^2, \tag{8}$$

where $r_{o(1)}^2 \leq ... \leq r_{o(n)}^2$ are the ordered squared residuals, and h is equal to $n - [n/2]$ for a breakdown point of 0.5. This estimator is very similar to LS,

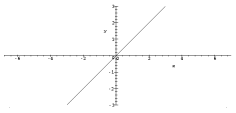
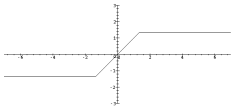
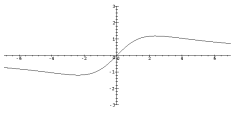
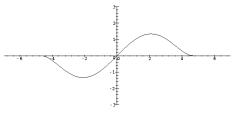
type	$\rho(x)$	$\psi(x)$	graphic representation of $\psi(x)$
LS	$x^2/2$	x	
Huber $\begin{cases} \text{if } \lvert x\rvert \le c \\ \text{if } \lvert x\rvert > c \end{cases}$	$\begin{cases} x^2/2 \\ c(\lvert x\rvert - c/2) \end{cases}$	$\begin{cases} x \\ c * \text{sgn}(x) \end{cases}$	
Cauchy	$\frac{c^2}{2}\log\left(1+\left(\frac{x}{c}\right)^2\right)$	$\dfrac{x}{1+\left(\frac{x}{c}\right)^2}$	
Tukey $\begin{cases} \text{if } \lvert x\rvert \le c \\ \text{if } \lvert x\rvert > c \end{cases}$	$\begin{cases} \frac{c^2}{6}\left[1-\left(1-\left(\frac{x}{c}\right)^2\right)^3\right] \\ c^2/6 \end{cases}$	$\begin{cases} x\left(1-\left(\frac{x}{c}\right)^2\right)^2 \\ 0 \end{cases}$	

Table 1: A few commonly used M-estimators.

except that the largest squared residuals are not taken into account. It has therefore a better relative efficiency than LMS.

Finally, as LMS and LTS systematically discard half of the data, [Rousseeuw 87] propose to refine the result of these estimators by performing a *reduced least squares* (RLS), that is a LS which includes all the data whose residual is lower than a threshold:

$$\min_{\mathbf{p}} \sum_{i=1}^{n} w_i r_i^2, \tag{9}$$

where

$$w_i = \begin{cases} 1 \text{ if } \lvert r_i\rvert \le 2.5\ \hat{\sigma}, \\ 0 \text{ if } \lvert r_i\rvert > 2.5\ \hat{\sigma}\ . \end{cases} \tag{10}$$

$\hat{\sigma}$ is an approximation of the standard deviation of the residual errors, and itself has to be estimated in a robust way. Precisely, we compute $\hat{\sigma}$ by the

equation

$$\hat{\sigma} = 2.6477 \sqrt{\frac{1}{h} \sum_{i=1}^{h} r^2_{o(i)}}, \tag{11}$$

where h is equal to $n - [n/2]$. The factor 2.6477 is introduced because $\frac{1}{h} \sum_{i=1}^{h} r^2_{o(i)}$ is estimated to be approximately $1/2.6477^2$ when the residuals are random numbers sampled from the gaussian normal distribution $N(0, 1)$.

Minimization

Minimizations (5), (8) or (9) can be performed by standard techniques using an initial estimate of $\mathbf{p}$: a very simple approach like Powell's method [Press 88] proved to be sufficient in our case, and relatively fast to compute. For temporal registration, the initial estimate of $\mathbf{p}$ is the pose computed for the previous frame. For the first frame, the pose is initialized by the method of Dementhon and Davis [Dementhon 95] applied on a set of points (this method does not require any initial estimate).

4 Pose computation from various features

Most pose estimation algorithms use simple features: points [Haralick 89, Dementhon 95], lines [Dhome 89, Shakunaga 93, Kumar 94] or circles [Ferri 93]. But only few papers have been devoted to the 2D/3D registration of curves. Kriegman [Kriegman 90] proposed an algebraic method to compute the pose of a curved object from the observation of its occluding contours (it can be easily applied to perform 2D/3D registration). Unfortunately, this method only deals with surfaces or curves that can be described by a collection of parametric patches (ratio of polynomials). Moreover, the use of elimination theory to compute the pose turns out to be very expensive. The 2D/3D registration problem has also been considered in [Feldmar 94], for medical tasks. Starting from a gross estimation of the pose, an ICP algorithm is used to perform registration. An extended Kalman filter is used to discard outliers by performing χ^2 tests. Nevertheless, since the results depend on the order of processing of

the measurements, the estimation process may be trapped into a local minimum, especially if the initial estimate is not very accurate. Such a process is therefore not really robust against gross errors.

The purpose of our algorithm is to compute the pose from the 2D/3D matching of n various features (points, lines and free-form curves) described by chains of points. As in section 3, it is assumed we know an initial estimate of $\mathbf{p}$ and the intrinsic parameters of the camera. Let:

- C_i be a 3D curve, described by the chain of 3D points $\{M_{i,j}\}_{1 \leq j \leq l_i}$ (note that C_i can be any 3D feature, including points and lines),
- c_i be the projection of C_i in the image plane, described by the chain of 2D points $\{m_{i,j}\}_{1 \leq j \leq l_i}$, where $m_{i,j} = Proj(\mathbf{R}M_{i,j} + \mathbf{t})$,
- c'_i be the detected curve (tracked curve) corresponding to C_i, described by the chain of 2D points $\{m'_{i,j}\}_{1 \leq j \leq l'_i}$.

A simple solution would be to perform the one stage minimization

$$\min \sum_{i,j} \rho(d_{i,j}) \tag{12}$$

(if a M-estimator is used), where $d_{i,j} = Dist(m'_{i,j}, c_i)$ and $Dist$ is a function which approximates the Euclidean distance from a point to a contour (see appendix A).

Unfortunately, this method is unsatisfactory for several reasons. First, it makes no distinction between local errors (when a feature is only partially well localized), and gross errors (when the position of a feature is completely erroneous), whereas these two kinds of errors are not identical: not treating them separately induces a great loss of robustness and accuracy. Moreover, this method does not consider a feature to be an independent entity, but merges it into the set of all the points belonging to the various features. That has at least two consequences: first, the larger a feature is, the greater is its weight in the computation of the pose, whereas in some cases a single point can have the same importance as a large curve (in particular a corner is often a safe feature). Secondly, it does not allow us to detect and discard a complete feature for which tracking has failed.

To avoid this, we propose to use robust estimators in a two stage process: a **local stage**, which computes a robust residual for each feature, and a **global stage** which minimizes a robust function of these residuals. The local stage will reduce (or even suppress) the influence of erroneous sections of the contours (see feature 4 on Figure 6.b), whereas the global stage will discard the *feature outliers*, *i.e.* contours which are completely erroneous, or which contain a too large portion of erroneous points (see feature 5 on Figure 6.b).

The local stage

The residual error r_i of curve C_i is computed by a robust function of the distances $\{d_{i,j}\}_{1 \leq j \leq l'_i}$. Different estimators may be used here: a M-estimator gives

$$r_i^2 = \frac{1}{l'_i} \sum_{j=1}^{l'_i} \rho(d_{i,j}) \tag{13}$$

whereas a LTS gives

$$r_i^2 = \frac{1}{h} \sum_{j=1}^{h} d_{i,o(j)}^2, \tag{14}$$

where $h = l'_i - [l'_i/2]$.

The choice of the estimator can have a great influence over the final result: for the local stage, we prefer to use a M-estimator rather than a LTS which is too restrictive. Indeed, as a LTS only uses a part of the data, using it on the local stage often leads to a local minimum, where only a part of the projected features is superimposed on the corresponding 2D features (see feature 8 of Figure 7.a). As the residuals of the points belonging to the superimposed sections are much smaller than the other ones, even a RLS is insufficient to leave this local minimum.

Among the M-estimators presented in Table 1, some are more restrictive than others: when Tukey's influence function is null for residuals larger than c, Cauchy's remains larger than zero while decreasing, whereas Huber's remains constant, equal to c. That's why we have chosen M-Huber for the local stage, which has proved to be a good choice in our experiments.

The global stage

As in section 3, we minimize a robust function of the residuals r_i (expressions (5), (8) or (9)), by using Powell's method initialized with the pose computed for the previous frame (except for the first frame where the pose is initialized by Dementhon and Davis [Dementhon 95] applied on four points - see section 6 for more details).

Once again, the choice of the estimator is crucial. Using a LTS can be a good choice if the segmentation is very poor, with often more than 20% feature outliers. However, this choice can also lead to a local minimum (in our experiment, this occured in particular when the depth information did not belong to the "good" half of the data kept by LTS - see Figure 7.b). Moreover, if the number of matched features is relatively small (typically lower than 10), then the least squares performed by LTS on half of the features easily induces very small residuals for these features, and hence does not enable RLS to take the other features into account. For this reason, we prefer to use M-Tukey, which is restrictive enough to suppress the influence of outliers, but which takes all the data into consideration.

Thus, the pose is computed by using M-Tukey for the global stage and M-Huber for the local stage (with constants k set to 4 and 2, respectively). In the following, this will be denoted by M-Tukey(M-Huber).

Discarding of feature outliers

Once a first estimation of the pose has been done using the previous method, the detection of feature outliers can be performed easily: as they should not have influenced the estimation, their residual must be much larger than the other ones. We therefore only have to compare them with the standard deviation of all the residuals: if $r_i > 2.5\ \hat{\sigma}$ (where $\hat{\sigma}$ is given by equation 11), then the feature is removed from the set $\mathcal{S}$ containing the tracked features used to compute the pose.

In order to refine the pose computed by M-Tukey(M-Huber), we can then perform a least squares estimator on the retained features (with still M-Huber for the local stage). Finally, the $R-ESTIM$ robust estimation algorithm

used in our experiments is the following one:

- **Step 1:** M-Tukey(M-Huber),
- **Step 2:** Discarding of feature outliers,
- **Step 3:** LS(M-Huber) on the retained features.

This algorithm has proved to be very robust and reliable in our experiments.

5 Updating the tracked features

When the camera undergoes a large motion, new features may appear while old ones disappear. It is therefore essential to dynamically update the set $\mathcal{S}$ of tracked features throughout the sequence.

In a preliminary step, we chose features of the 3D model that give rise to easily detectable 2D features. Given a robustly computed pose, we are able to determine which part of these features projects in the image, and then update $\mathcal{S}$ consistently.

When a new feature appears, we have to determine the corresponding 2D feature that will be tracked in the subsequent frames (we call this task *update a feature*). The method used to update a feature is as follows:

- **Step 1:** Perform a classical Canny edge detection [Canny 86].
- **Step 2:** Consider contours that are close enough to the projection of the 3D feature (for example, the first ten contours sorted according to the increasing size of their robust residual, given by equation (14)).
- **Step 3:** For each of these contours c, compute the pose corresponding to $\mathcal{S} \cup c$ (by $R - ESTIM$), and see if c is discarded or not. If c is kept, eliminate the erroneous sections of this contour by comparing distances $d_{i,j}$ to their standard deviation.

- **Step 4:** If no contour is retained, then return fail, else try to concatenate all the kept sections and return the longest concatenated contour (the image feature can therefore be made up of several chains).

This method is illustrated by Figure 4. It is also used to update features that are discarded by $R-ESTIM$ during the pose computation process. Moreover, a phenomena often occurs when tracking is performed over a long sequence: since the tracking (in fact the snake process) tends to decrease the length of the feature, this one may become too small to be correctly tracked (because of the level of noise in the image, the length must be sufficient to allow the optic flow to be significantly computed). This method is therefore also used to update these features.

6 An augmented reality application: the bridges of Paris

We present in this section an application of our method to an augmented reality application: the illumination of the bridges of Paris. Our motivation came from the encrustation of a model illuminated synthetically in its real environment. Decision-makers of the Paris city administration were willing to have a new lighting system for a number of bridges of the Seine around the "Ile de la Cité". They wanted to test several candidate illumination projects and be able to choose on computer simulations alone what project was the best. Most importantly, they wanted to evaluate the influence of this illumination on the surrounding elements.

A 300-image panoramic sequence of the Pont Neuf was shot at dusk time from another bridge (see Figure 9.a). The fact that the images are dark and noisy has a strong influence on the quality of the segmentation (see Figure 3.a, which shows the edge map of the first image of the sequence) and consequently on the strategies used for the tracking. It also restricts the number of reliable features that we are able to use. The reader should keep this in mind when visually evaluating the final composition.

It should also be noted that the modeling of the bridge was done mostly manually using information from architectural maps. It is very unreliable at

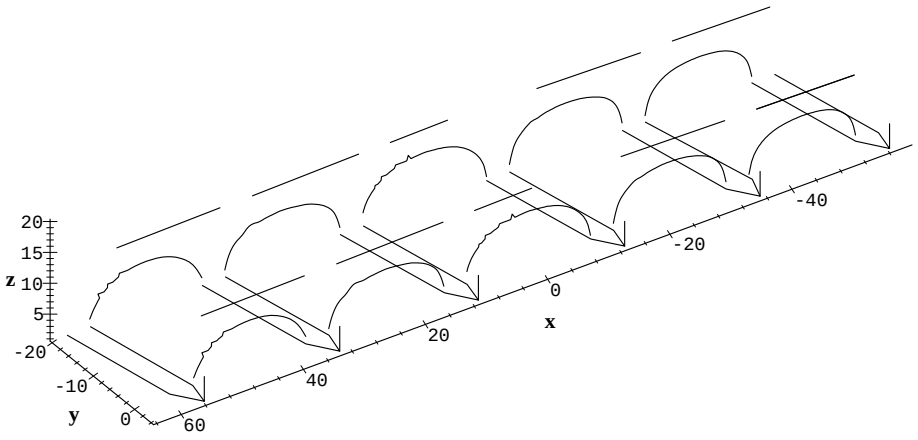


Figure 2: The complete wireframe model of the bridge. The poor quality of this modelling is pretty blatant at certain places.

places (*e.g.* the staircase effect on some of the arches - see Figure 2). Much better results could certainly be obtained with a more precise modeling, using a laser for instance. Besides, some features that are tracked in the sequence are often made up with contours which are the intersection of the bridge with the river surface (see for instance the piers). The elevation of the river surface is of course not known in the model but we can compute it off-line from a set of point correspondences, including points at the river surface: a first approximation h_0 of the elevation can be computed because the dimensions of the bridge are known. It is then refined in the following way: from the estimation h_0, we compute for each elevation h in the $[h_0 - 50cm, h_0 + 50cm]$ range the corresponding viewpoint. The correct elevation is the one for which the reprojection error is the smallest.

The intrinsic parameters of the camera are obtained by calibration on a reference object close to the observer (in the 3 meters range) while the scene under consideration is far from the camera (in the 300 meters range), which induces a lot of noise on these parameters. The number of features that we are able to detect and track over the sequence is usually small (typically, between 6 and 8 curves).

Initialization

At the beginning of the process, the user points out four points in the image and their corresponding 3D points. They are used to compute the initial pose with method of [Dementhon 95] (Figure 3.b). The updating process is then used to automatically determine the features that will be tracked (Figure 3.c). Figure 4 details the updating process for one of these features (a pier of the bridge).

Tracking Stage

Figure 5.a shows the set of tracked features for the 10^{th} image. The reader may notice that feature 4 is not fully correct. This is mainly due to the snake process, that makes the snake attracted by high gradients. However, as a great part of the feature correctly matches the feature, it is retained in the set of tracked features.

The result of the tracking in the 12^{th} image is shown in Figure 5.b. Except for feature 5, the other primitives are well tracked. The error on feature 5 is due to the failure of the prediction step in the tracking process due to noise in the image. Hence, the snake converges towards an erroneous contour.

Robust pose computation

The projection of the model features using the pose computed for the 10^{th} frame is shown in Figure 6.a.

Figure 6.b shows the reprojection of the model after the robust pose computation. Despite the bad accuracy of the model, the result is visually convincing. In order the reader to be aware of the parts of the curve which are less taken into account in the computation, we have drawn in black the points for which the residual is greater than c (c is defined in Table 1). Roughly speaking, these points are the ones for which the weight in the computation is decreased because their residual is too large. It must also be noticed that feature 5 is considered as an outlier and is removed from the set of tracked features.

Figure 7 shows the result we would obtain by using a LTS on the local stage (7.a) or on the global stage (7.b). In the both cases, we fall into a local minimum.

a.

b.

c.

Figure 3: Initialization of the loop. a: Edge map of the 1^{st} image. b: Projection of the 3D features (black dashed lines) after computation of the pose by using Dementhon and Davis' method applied to four points (white crosses). c: 2D features that will be tracked are automatically determined (white lines). White dashed lines correspond to the projections of their corresponding 3D feature.

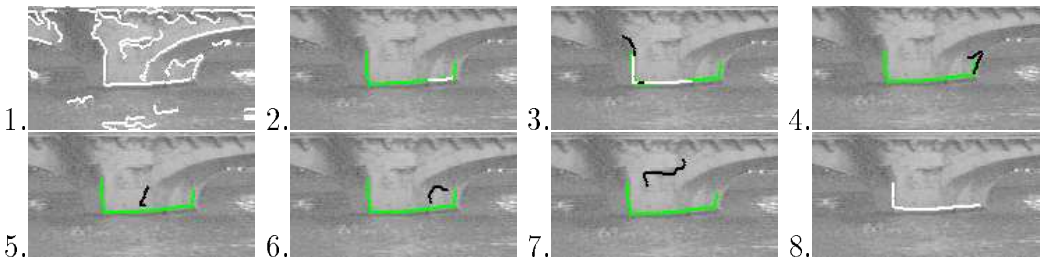


Figure 4: Detail of the updating process for the left pier of the 1^{st} image. 1: Edge map around the projected feature. 2 to 7: Keeping of the coherent sections for the six closest contours c, once $R-ESTIM$ has been applied on $\mathcal{S} \cup c$. Projected 3D features are drawn in green, kept sections of 2D contours in white and discarded sections in black (feature outliers are entirely drawn in black). 8: Final result: the returned feature is made up of the kept sections.

Updating the set of tracked features

A new feature (5) has been added to the set of tracked feature (Figure 8.a). Note that in this case, this new feature replaces the one which has been mis-tracked. Using this new set of features, the pose is re-estimated (Figure 8.b). Figure 9.b shows the result of the composition of the computer-generated and the video image for the 60^{th} image (precisions about this composition may be found in **??**).

The final sequence

The interested reader should visualize an extract of the results obtained for the whole sequence. The first one[1] shows the pose computed by $R-ESTIM$ on step 2 (after tracking) and step 4 (after updating) of each iteration (Figure 1). Conventions used for the lines are the ones of Figure 6. The second sequence[2] shows the reprojection of the model after the last step of each iteration.

Some comments are in order. Considering the conditions that we have already mentioned (bad segmentation due to darkness, poor modeling), the

[1] `http://www.loria.fr/~gsimon/sequence1.mpg`
[2] `http://www.loria.fr/~gsimon/sequence2.mpg`

a.

b.

Figure 5: Tracking for the 12^{th} image. a: The set of features after processing of the 10^{th} image. b: Result of the tracking in the 12^{th} image.

a.

b.

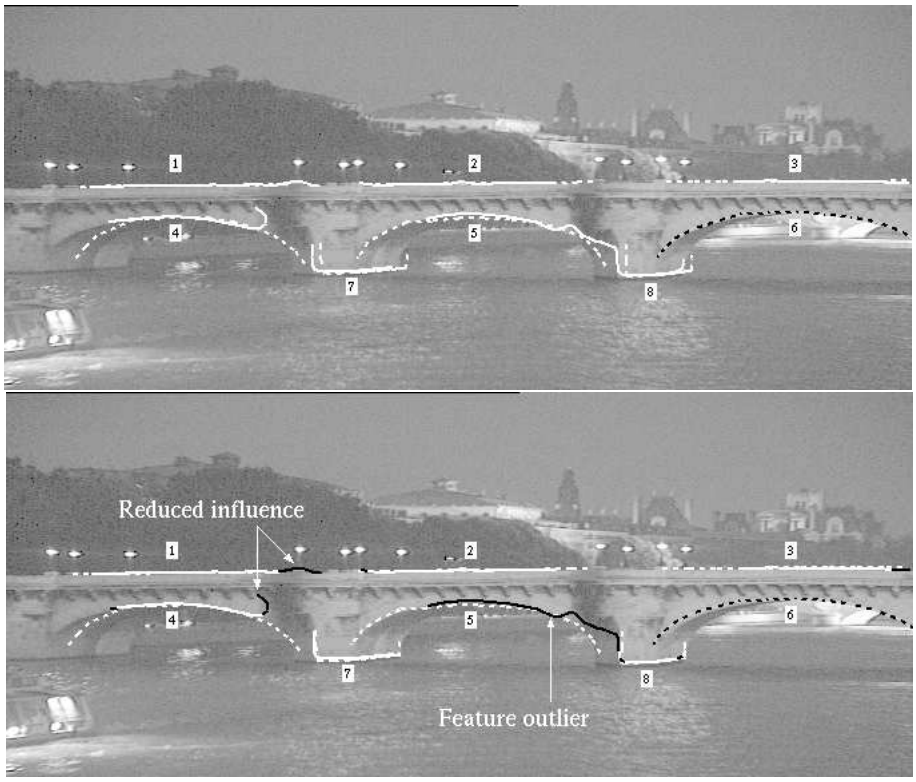


Figure 6: Pose computation for the 12^{th} image. a: Projection of the 3D features using the pose computed for the 10^{th} image. White lines correspond to the tracked features, white dashed lines to their 3D correspondent and black dashed lines to the not (yet) used features. b: Reprojection of the 3D features after the robust pose computation. Sections for which residuals are greater than c and feature outliers are drawn in black.

a.

b.

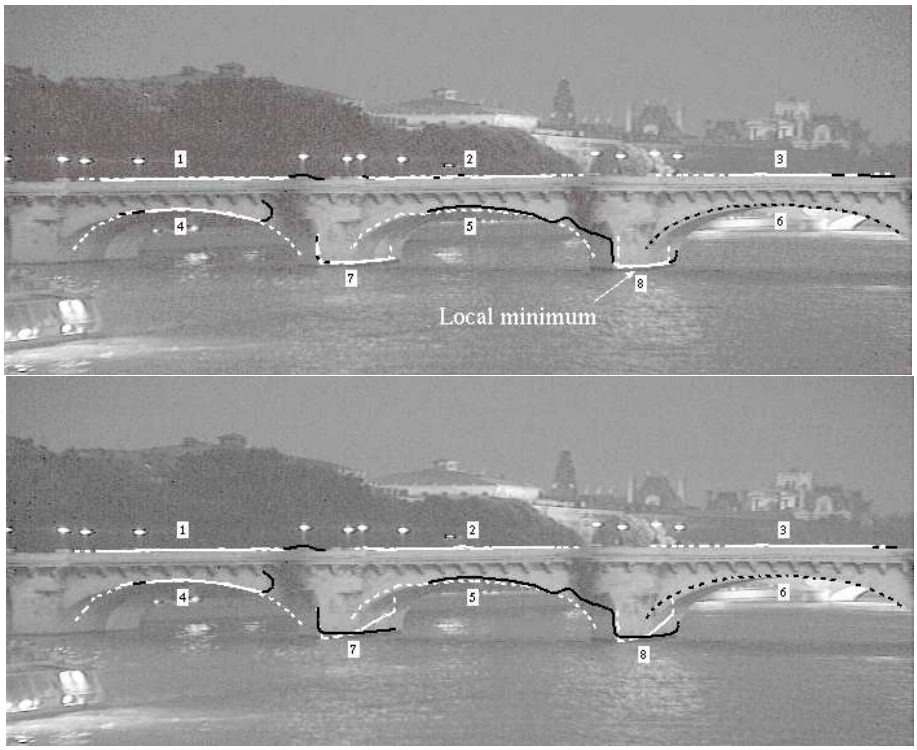


Figure 7: Result of the LTS estimator applied to the local stage (a) and to the global stage (b) for the 12^{th} image. a: Tukey(LTS). Sections for which the residuals are greater than $\hat{\sigma}$ are drawn in black. Local minimum is obtained for feature 8. b: LTS(Huber). Discarded features are drawn in black. We fall in a local minimum because the depth information (features 7 and 8) does not belong to the "good" half of the data.

a.

b.

Figure 8: Updating of the set of tracked features for the 12^{th} image. a: A new feature has been added. b: Reprojection of the model after updating.

a.

b.

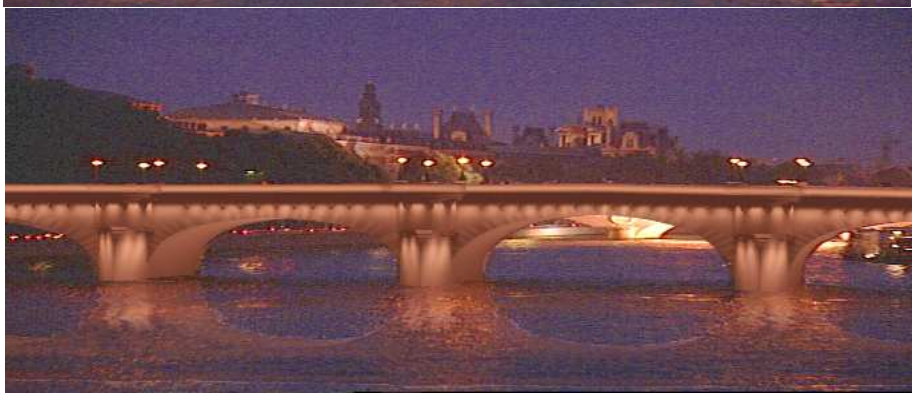


Figure 9: a: The 60^{th} image before composition. b: Result of the final composition.

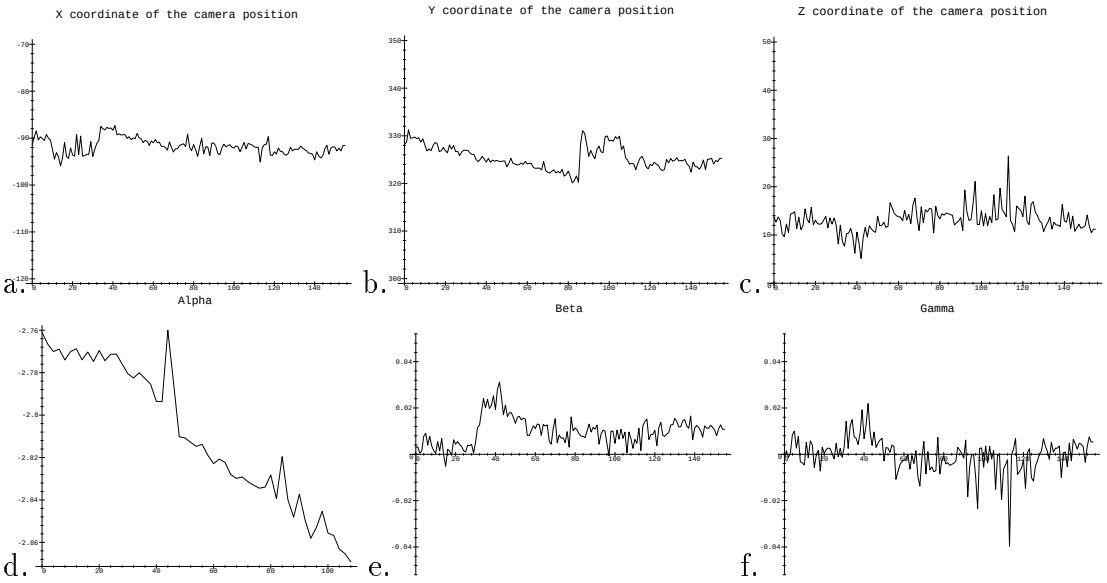


Figure 10: Evolution of the pose (translation and euler angles) over the panoramic sequence.

overall visual impression is quite satisfactory. We show in Figure 10 the evolution of the pose over the sequence. As the sequence has been shot with a camera put on a tripod, the translation part of the pose expressed in the bridge frame should be constant (Figure 10.a,b,c - see Figure 2 for the axes). The reader can see that the t_x and t_y coordinates of the translation are nearly constant whereas the t_z coordinate is less stable. This is mainly due to the strong imprecisions on the 3D model that mainly affect the height of bridge and to the experimental determination of the elevation of the river surface.

The Euler angles of the rotation part of the pose are shown in Figure 10.d,e,f. The α angle, which is the rotation around the z axis regularly evolves as the camera turns. The two other angles are roughly constant. The standard deviation of the camera parameters that should be constant on the panoramic sequence are shown in Table 2.

Parameter	*Mean*	*Standard deviation*
t_x (m)	-91.686634	1.678529
t_y (m)	325.359777	2.339981
t_z (m)	13.092605	2.547535
β (rad)	0.009910	0.005982
γ (rad)	-0.000236	0.005982

Table 2: Mean and standard deviation of the camera parameters.

7 Conclusion and possible improvements

This paper has presented a model registration system capable of tracking an object, the model of which is known, in an image sequence. Once initialized with known camera parameters, a user specified set of 3D features, and four 2D/3D points matched in the first image, the system is able to process a complete sequence in a really autonomous way. Its reliability has been shown on a 300-image sequence of a bridge shot at dusk time (segmentation was hence very poor), for an augmented reality project.

The heart of our system is the pose computation method, which handles various features (points, lines and free-form curves) in a very robust way. It consists in using robust estimators in a two-stage process (a robust residual is computed for each feature and a robust function of these residuals is minimized) and is therefore able to give a correct estimate of the pose even when tracking errors occur. This method is also used to update the set of visible features along the sequence, by finding the 2D contours that are tracked when new 3D features become visible.

Several directions for further investigation are suggested:

- Currently, our algorithms are not processed in real time. It was not a priority for us, on account of the kind of our applications: augmented reality projects, whose goal was to encrust a very realistic synthetical object in a pre-acquired sequence (synthesis process was anyhow much slower than registration task). However, in order to use our method for real time applications, we can greatly improve its speed by processing both the tracking task and the pose estimation in a parallel way (the

snake process and also the computation of the feature residuals can be executed at the same time for all the features).

- In this paper, points were used to compute the initial pose and find the 2D features to be tracked in the first image, but only curves were tracked into the sequence. As our pose estimation method admits points, we could also track them along the sequence by using a corner tracker or other kind of point tracking.
- In this work, the model is structured in a set of features that are chosen off-line by the user. We could think about an automatic way of finding the more pertinent 3D features (with regard to criteria such as total length or number of represented 3D directions), and eventually on-line construct and verify these features.

References

[Berger 94] 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.

[Berger 96] M.-O. Berger, C. Chevrier and G. Simon. Compositing Computer and Video Image Sequences: Robust Algorithms for the Reconstruction of the Camera Parameters. In *Computer Graphics Forum, Conference Issue Eurographics'96, Poitiers, France*, volume 15, pages 23–32, August 1996.

[Canny 86] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on PAMI*, 8(6):679–698, 1986.

[Dementhon 95] D. Dementhon and L. Davis. Model Based Object Pose in 25 Lines of Code. *International Journal of Computer Vision*, 15:123–141, 1995.

[Dhome 89] M. Dhome, M. Richetin, J.T. Lapresté and G. Rives. Determination of the Attitude of 3-D Objects from a Single Perspective View. *IEEE Transactions on PAMI*, 11(12):1265–1278, 1989.

[Faugeras 86] O. D. Faugeras and G. Toscani. The Calibration Problem for Stereo. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL (USA)*, pages 15–20, 1986.

[Feldmar 94] J. Feldmar, N. Ayache and F. Betting. 3D-2D projective registration of free form curves and surfaces. Rapport de recherche 2436, INRIA, 1994.

[Ferri 93] M. Ferri, F. Mangili and G. Viano. Projective Pose Estimation of Linear and Quadratic Primitives in Monocular Computer Vision. *CVGIP: Image Understanding*, 58(1):66–84, July 1993.

[Gennery 92] D. Gennery. Visual Tracking of Known Three Dimensional objects. *International Journal of Computer Vision*, 7(3):243–270, 1992.

[Haralick 89] R. M. Haralick, H. Joo, C. N. Lee, X. Zhuang, V.G. Vaidya and M. B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6), 1989.

[Huber 81] P. J. Huber. *Robust Statistics*. Wiley, New York, 1981.

[Kim 89] D. Y. Kim, J. J. Kim, P. Meer, D. Mintz and A. Rosenfeld. Robust computer vision: A least median of squares based approach. In *The DARPA Image Understanding Workshop*, Palo Alto, CA, May 1989. Kaufman Publishers.

[Koller 92] D. Koller, K. Daniilidis and H. H. Nagel. Model-Based Object Tracking in Traffic Scenes. In *Proceedings of Second European Conference on Computer Vision, Santa Margherita Ligure*

(Italy), volume 588 of *Lecture Notes in Computer Science*, pages 437–452, 1992.

[Kriegman 90] D. Kriegman and J. Ponce. On Recognizing and Positioning Curved 3D objects from Image Contours. *IEEE Transactions on PAMI*, 12(12):1127–1137, December 1990.

[Kumar 94] R. Kumar and A. Hanson. Robust Methods for Estimating Pose and a Sensitivity Analysis. *CVGIP: Image Understanding*, 60(3):313–342, 1994.

[Lowe 92] D. Lowe. Robust Model based Motion Tracking Trough the Integration of Search and Estimation. *International Journal of Computer Vision*, 8(2):113–122, 1992.

[Press 88] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling. *Numerical Recipes in C, The Art of Scientific Computing.* Cambridge University Press, 1988.

[Ravela 96] 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.

[Rousseeuw 87] P. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection.* Wiley Series in Probability and Mathematical Statistics. Wiley, 1987.

[Shakunaga 93] T. Shakunaga. Robust Line Based Pose Enumeration From a Single Image. In *Proceedings of 4th International Conference on Computer Vision, Berlin (Germany)*, pages 545–550, 1993.

[Uenohara 96] M. Uenohara and T. Kanade. Vision based object registration for real time image overlay. *Journal of Computers in Biology and Medecine*, 1996.

[Zhang 95] Z. Zhang, R. Deriche, O. Faugeras and Q. Luong. A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry. *Artificial Intelligence*, 78:87–119, October 1995.

Appendix

A Approximation of the distance from a point to a contour

The Euclidean distance $d_{i,j}$ from a point $m'_{i,j}$ to a curve c_i is estimated as follows (see Fig.11):

1. a first approximation of $d_{i,j}$ is obtained by taking the Euclidean distance from $m'_{i,j}$ to its closest point belonging to c_i:

$$d^0_{i,j} = \min_{1 \leq k \leq l_i} ||m'_{i,j} - m_{i,k}||. \tag{15}$$

2. Let k_0 be the subscript realizing the minimum of equation (15). If $k_0 = 1$ or $k_0 = l_i$, then we take $d_{i,j}$ equal to $d^0_{i,j}$, else we refine $d^0_{i,j}$ as follows: suppose we rename points m_{i,k_0-1}, m_{i,k_0}, m_{i,k_0+1} into (respectively) A, B, C, let θ be the angle between vectors $\overrightarrow{AB}$ and $\overrightarrow{AC}$, and let ϵ be an arbitrarily small positive real.

   - if $|\sin\theta| \leq \epsilon$ (that is, points A, B, C are aligned), then we take $d_{i,j}$ equal to the Euclidean distance from $m'_{i,j}$ to line (AC);
   - if $|\sin\theta| > \epsilon$, then we take $d_{i,j}$ equal to the Euclidean distance from $m'_{i,j}$ to the circle passing through A, B and C.

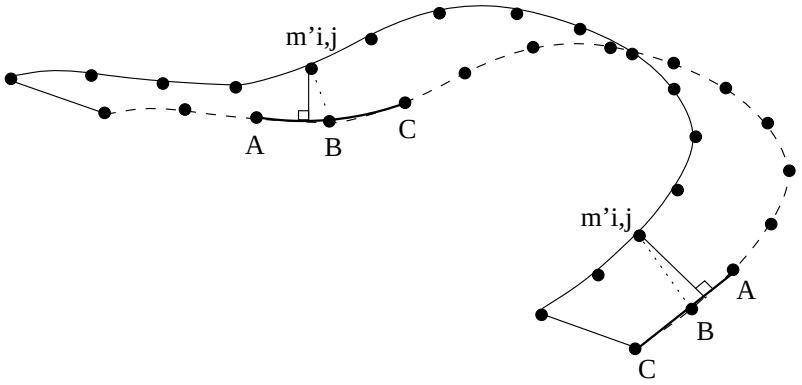


Figure 11: Distance from a point to a curve. Distances are first approximated by the dotted segments length, and then refined by the solid segments length.

Unit´e de recherche INRIA Lorraine, Technopˆole de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhˆone-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399