Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems
September 28 - October 2, 2004, Sendai, Japan

Real-time image-based tracking of planes using Efficient Second-order Minimization

Selim BENHIMANE and Ezio MALIS
I.N.R.I.A. Sophia-Antipolis, FRANCE

***Abstract*— The tracking algorithm presented in this paper is based on minimizing the sum-of-squared-difference between a given template and the current image. Theoretically, amongst all standard minimization algorithms, the Newton method has the highest local convergence rate since it is based on a second-order Taylor series of the sum-of-squared-differences. However, the Newton method is time consuming since it needs the computation of the Hessian. In addition, if the Hessian is not positive definite, convergence problems can occur. That is why several methods use an approximation of the Hessian. The price to pay is the loss of the high convergence rate. The aim of this paper is to propose a tracking algorithm based on a second-order minimization method which does not need to compute the Hessian.**

I. Introduction

Visual tracking has many important applications and, in particular, it is the core of a vision-based control system in robotics [1]. When considering real-time robotic applications, the main requirements of a tracking algorithm are efficiency, accuracy and robustness. In this paper, we focus mainly on efficiency and accuracy of visual tracking methods. Visual tracking methods can be classified into two main groups. The first group is composed of methods that track local features such as line segments, edges or contours across the sequence [2]–[4]. These techniques are sensitive to feature detection and cannot be applied to complex images that do not contain special sets of features to track. The second group is composed of methods that make only use of image intensity information. These methods estimate the movement, the deformation or the illumination parameters of a reference template between two frames by minimizing an error measure based on image brightness. Many approaches have been proposed to find the relationship between the measured error and the parameters variation. Some methods learn this relationship in an off-line processing stage: difference decomposition [5] [6], active blobs [7], active appearance models [8]. Although these methods solve the problem, they can not be used in some real-time robotic applications where the learning step can not be processed on-line. For example, consider a robot moving in an unknown environment that needs to track instantaneously an object suddenly appearing in its field of view. Alternatively, there are methods that minimize the sum-of-squared-differences (SSD) between the reference template and the current image using parametric models [9]–[12]. Many minimization algorithms could be used to estimate the transformation parameters. Theoretically, the Newton method has the highest local convergence rate since it is based on a second-order Taylor series of the SSD. However, the Hessian computation in the Newton method is time consuming. In addition, if the Hessian is not positive definite, convergence problems can occur. In this paper, we propose to use an efficient second-order minimization method (ESM) [13] to solve the problem. The ESM method has a high convergence rate like the Newton method, but the ESM does not need to compute the Hessian. Theoretical analysis and comparative simulations with other tracking approaches show that the method has a higher convergence rate than other minimization techniques. Consequently, the ESM algorithm tracks with higher inter-frame movements.

II. Notations and definitions

Let I be a $(n \times m)$ image template, which can be considered as a matrix. Let $\mathbf{p} = (u,v)$ be the (2×1) vector containing the coordinates of a pixel: $(u,v) \in \{1,2,...,n\} \times \{1,2,...,m\}$. Thus, $I(\mathbf{p})$ is the intensity of the pixel $\mathbf{p}$. Let $\mathbf{w}(\mathbf{x})$ be an image transformation operator (e.g. a translation, an affine or a projective transformation): $\mathbf{p} \mapsto \mathbf{w}(\mathbf{x})(\mathbf{p})$. The p parameters of the transformation $\mathbf{w}(\mathbf{x})$ are in the vector $\mathbf{x} \in \mathbb{R}^p$. Let $\circ$ be a binary operation such that $(\mathbb{R}^p, \circ)$ is a transformation group and the map $\mathbf{w}$ is a group action. Let $\mathbf{e}$ be the identity element of the transformation group. We have the following properties:

- $\mathbf{w}(\mathbf{e})$ is the identity map i.e. $\forall \mathbf{p} \in \mathbb{R}^2$

$$\mathbf{w}(\mathbf{e})(\mathbf{p}) = \mathbf{p} \tag{1}$$

- the composition of actions corresponds to the action of the composition i.e $\forall \mathbf{p} \in \mathbb{R}^2$, $\forall \mathbf{x}_1$, $\mathbf{x}_2 \in \mathbb{R}^p$

$$\mathbf{w}(\mathbf{x}_1)(\mathbf{w}(\mathbf{x}_2)(\mathbf{p})) = \mathbf{w}(\mathbf{x}_1 \circ \mathbf{x}_2)(\mathbf{p}) \tag{2}$$

- the inverse of an action corresponds to the action of the inverse i.e $\forall \mathbf{x} \in \mathbb{R}^p$

$$(\mathbf{w}(\mathbf{x}))^{-1} = \mathbf{w}(\mathbf{x}^{-1}) \tag{3}$$

Let $q = nm$ be the total number of pixels in the image I, and consider the $(q\times1)$ vector $\mathbf{s}(\mathbf{x})$ obtained by rearranging the entries of the image matrix I after warping with $\mathbf{w}(\mathbf{x})$:

$$\mathbf{s}(\mathbf{x}) = (I(\mathbf{w}(\mathbf{x})(\mathbf{p}_1)), I(\mathbf{w}(\mathbf{x})(\mathbf{p}_2)), ..., I(\mathbf{w}(\mathbf{x})(\mathbf{p}_q))) \tag{4}$$

where $\forall k \in \{1,2,...,q\}$, $\mathbf{p}_k \in \{1,2,...,n\} \times \{1,2,...,m\}$. We will note $\mathbf{J}(\mathbf{x})$ the $(q \times p)$ Jacobian matrix:

$$\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{s}(\mathbf{x})}{\partial \mathbf{x}}$$

and $\mathbf{H}_k(\mathbf{x})$ the k-th $(p \times p)$ Hessian matrix:

$$\mathbf{H}_k(\mathbf{x}) = \frac{\partial^2 s_k(\mathbf{x})}{\partial \mathbf{x}^2} \qquad \forall k \in \{1,2,...,q\}$$

Note that there exist q Hessian matrices, one per pixel.

0-7803-8463-6/04/$20.00 ©2004 IEEE

III. Theoretical background

Suppose we have selected a $n \times m$ template which can be attached to an object in a reference image. Suppose that, without loss of generality, the template has been transformed into itself by the identity map $\mathbf{w}(\mathbf{e})$. Then $\mathbf{s}(\mathbf{e})$ is the (q×1) vector containing the intensities of the template. Suppose that the current image of the same object can be obtained by warping the template with a given motion model (e.g. a projective transformation). Let $\mathbf{x}_c$ be the vector containing the unknown current transformation parameters. The current image can be reshaped into the (q×1) vector $\mathbf{s}(\mathbf{x}_c)$. The goal of template-based tracking algorithms is to find $\Delta\mathbf{x}$ such that $\mathbf{s}(\Delta\mathbf{x} \circ \mathbf{x}_c)$ coincides with the given template image $\mathbf{s}(\mathbf{e})$. The optimal $\Delta\mathbf{x}$ is $\Delta\mathbf{x} = \mathbf{x}_c^{-1}$. The vector $\Delta\mathbf{x}$ can be estimated by minimizing the function $f(\Delta\mathbf{x})$ which is the SSD between $\mathbf{s}(\mathbf{e})$ and $\mathbf{s}(\Delta\mathbf{x} \circ \mathbf{x}_c)$. Let s_k be the k-th entry of vector $\mathbf{s}$, the function f is:

$$f(\Delta\mathbf{x}) = \frac{1}{2}\sum_{k=1}^{q}(s_k(\Delta\mathbf{x} \circ \mathbf{x}_c) - s_k(\mathbf{e}))^2 = \frac{1}{2}\|(\mathbf{s}(\Delta\mathbf{x} \circ \mathbf{x}_c) - \mathbf{s}(\mathbf{e}))\|^2$$

The minimization of this non-linear function is done iteratively. At each iteration, we estimate the state update $\Delta\mathbf{x}$ by finding the minimum of the cost function. The necessary condition for finding the minimum of the cost function is:

$$\frac{\partial f(\Delta\mathbf{x})}{\partial \Delta\mathbf{x}} = 0 \tag{5}$$

This is a non-linear equation. We can linearize it by approximating the derivative of f about $\Delta\mathbf{x} = \mathbf{e}$ as follows:

$$\frac{\partial f(\Delta\mathbf{x})}{\partial \Delta\mathbf{x}} \approx \mathbf{g}^\top + \Delta\mathbf{x}^\top \mathbf{S} \tag{6}$$

where, after setting $\Delta\mathbf{s} = \mathbf{s}(\mathbf{x}_c) - \mathbf{s}(\mathbf{e})$ we have:

$$\begin{aligned}
\mathbf{g} &= \left.\frac{\partial f(\Delta\mathbf{x})}{\partial \Delta\mathbf{x}}\right|_{\Delta\mathbf{x}=\mathbf{e}}^{\top} = \mathbf{J}^\top(\mathbf{x}_c)\Delta\mathbf{s} \\
\mathbf{S} &= \left.\frac{\partial^2 f(\Delta\mathbf{x})}{\partial \Delta\mathbf{x}^2}\right|_{\Delta\mathbf{x}=\mathbf{e}} = \mathbf{J}^\top(\mathbf{x}_c)\mathbf{J}(\mathbf{x}_c) + \sum_{k=1}^{q}\mathbf{H}_k(\mathbf{x}_c)\Delta s_k
\end{aligned}$$

where Δs_k is the k-th entry of vector $\Delta\mathbf{s}$. The standard Newton minimization method solves equation (5) using the approximation (6) as follows: $\Delta\mathbf{x} = -\mathbf{S}^{-1}\mathbf{g}$. The Newton method has a quadratic convergence in a neighborhood of $\mathbf{e}$. If the function f is convex quadratic, then we can estimate $\Delta\mathbf{x} = \mathbf{x}_c^{-1}$ in only one iteration since the approximation in equation (6) becomes an equality. If the function f is not convex quadratic, convergence problems may occur if the Hessian is not positive definite. To overcome convergence problems, several methods approximate the Hessian $\mathbf{S}$ with a positive definite matrix $\widehat{\mathbf{S}}$. For example:

$\widehat{\mathbf{S}} = \alpha\,\mathbf{I}$	Steepest Descent
$\widehat{\mathbf{S}} = \mathbf{J}^\top\mathbf{J}$	Gauss-Newton
$\widehat{\mathbf{S}} = \mathbf{J}^\top\mathbf{J} + \alpha\,\mathrm{diag}(\mathbf{J}^\top\mathbf{J})$	Levenberg-Marquandt

Approximating the Hessian has the advantage of reducing the computation cost. However, the approximated Hessian must be updated at each iteration. In order to further reduce computation, we can use the following function:

$$\widetilde{f}(\Delta\mathbf{x}) = \frac{1}{2}\sum_{k=1}^{q}(s_k(\Delta\mathbf{x}) - s_k(\mathbf{x}_c))^2 = \frac{1}{2}\|(\mathbf{s}(\Delta\mathbf{x}) - \mathbf{s}(\mathbf{x}_c))\|^2$$

Similarly to the function f, in this case the Newton minimization method computes $\Delta\mathbf{x} = -\widetilde{\mathbf{S}}^{-1}\widetilde{\mathbf{g}}$, where:

$$\begin{aligned}
\widetilde{\mathbf{g}} &= \left.\frac{\partial \widetilde{f}(\Delta\mathbf{x})}{\partial \Delta\mathbf{x}}\right|_{\Delta\mathbf{x}=\mathbf{e}}^{\top} = -\mathbf{J}^\top(\mathbf{e})\Delta\mathbf{s} \\
\widetilde{\mathbf{S}} &= \left.\frac{\partial^2 \widetilde{f}(\Delta\mathbf{x})}{\partial \Delta\mathbf{x}^2}\right|_{\Delta\mathbf{x}=\mathbf{e}} = -\mathbf{J}^\top(\mathbf{e})\mathbf{J}(\mathbf{e}) - \sum_{k=1}^{q}\mathbf{H}_k(\mathbf{e})\Delta s_k
\end{aligned}$$

This function permits to compute once for all the matrices $\mathbf{J}$ and $\mathbf{H}_k$ since they are estimated in $\mathbf{e}$. The price to pay is a reduction of the convergence neighborhood.

IV. A different second-order approximation

The Newton minimization method performs a second-order approximation of the cost function by using its Hessian matrix. In this paper, we use the efficient second order approximation proposed in [13]. Consider the second-order Taylor series of the vector function $\mathbf{s}(\mathbf{x})$ about $\mathbf{x}_c$ and then evaluated at $\mathbf{e}$:

$$\mathbf{s}(\mathbf{e}) \approx \mathbf{s}(\mathbf{x}_c) + \mathbf{J}(\mathbf{x}_c)\Delta\mathbf{x} + \frac{1}{2}\mathbf{M}(\mathbf{x}_c, \Delta\mathbf{x})\Delta\mathbf{x} \tag{7}$$

where $\mathbf{M}(\mathbf{x}_c, \Delta\mathbf{x})$ is a $(q \times p)$ matrix that depends on $\Delta\mathbf{x}$ and on the q Hessian matrices of $\mathbf{s}(\mathbf{x})$ evaluated at $\mathbf{x}_c$. Using the first order series of the matrix $\mathbf{J}(\mathbf{x})$ about $\mathbf{x}_c$ and then evaluated at $\mathbf{e}$, we obtain:

$$\mathbf{J}(\mathbf{e}) \approx \mathbf{J}(\mathbf{x}_c) + \mathbf{M}(\mathbf{x}_c, \Delta\mathbf{x}) \tag{8}$$

Plugging equation (8) in equation (7), we obtain:

$$\mathbf{s}(\mathbf{e}) \approx \mathbf{s}(\mathbf{x}_c) + \frac{1}{2}(\mathbf{J}(\mathbf{e}) + \mathbf{J}(\mathbf{x}_c))\Delta\mathbf{x} \tag{9}$$

The equation:

$$\Delta\mathbf{s} \approx -\frac{1}{2}(\mathbf{J}(\mathbf{e}) + \mathbf{J}(\mathbf{x}_c))\Delta\mathbf{x} \tag{10}$$

is a second order approximation of $\Delta\mathbf{s}$ for small $\Delta\mathbf{x}$. Without computing the Hessian, we obtain an efficient second-order approximation of $\Delta\mathbf{s}$ (i.e. only using first order derivatives). Thus, the efficient second-order minimization (ESM) proposed in [13] can be used to improve template-based visual tracking. Similarly to [10] or [12], the Jacobian $\mathbf{J}(\mathbf{e})$ is constant and can be precomputed while similarly to [9], $\mathbf{J}(\mathbf{x}_c)$ is updated depending on $\mathbf{x}_c$. Let us suppose for the moment that $\mathbf{J}(\mathbf{x}_c)$ can be computed or approximated. The displacement can be obtained by computing the pseudo-inverse of the mean of the Jacobians:

$$\Delta\mathbf{x} \approx -2(\mathbf{J}(\mathbf{e}) + \mathbf{J}(\mathbf{x}_c))^{+}\Delta\mathbf{s} \tag{11}$$

If $\mathbf{s}(\mathbf{x})$ is quadratic in $\mathbf{x}$, then the equation (10) is not an approximation any more. Thus, we can estimate the true parameters of the warping in only one iteration. If the vector function $\mathbf{s}(\mathbf{x})$ is not quadratic, we can expect an improvement over the standard Newton second-order minimization method since we do not need the Hessian of the cost function to be definite positive in order to converge.

V. Advantages of the ESM algorithm

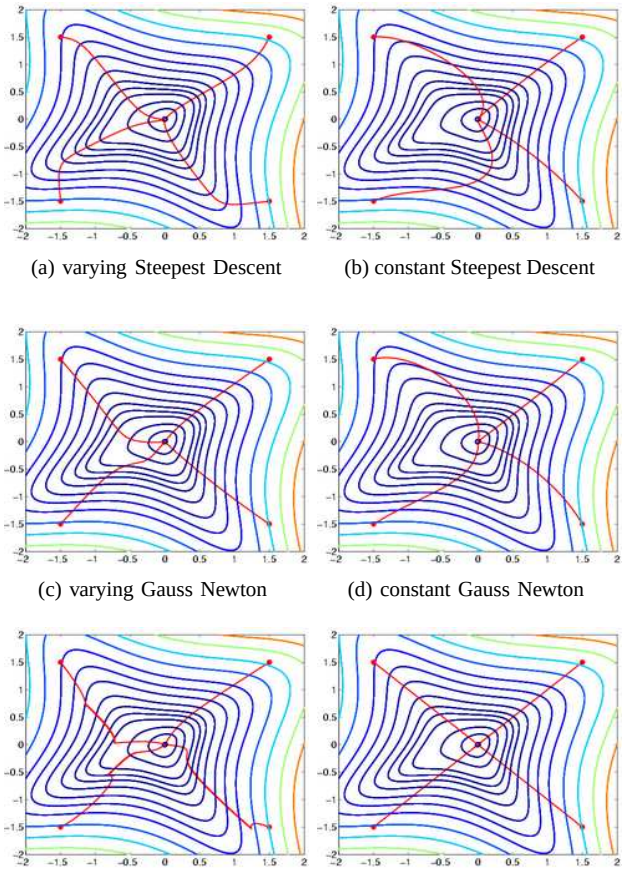
The main advantage of having a second order approximation of the parameter displacement is the high convergence rate. Another advantage is the avoidance of local minima close to the global one (i.e. when the second-order approximation is valid). We show now these advantages with the help of two simple examples.

A. High convergence rate

Consider a (4×1) vector function $\mathbf{s}(\mathbf{x})$ quadratic in a (2×1) parameter vector $\mathbf{x}$. The simulation is repeated 4 times with different starting points: $\mathbf{x}_c \in \{(\pm1.5, \pm1.5)\}$. Suppose we can measure the constant Jacobian $\mathbf{J}(\mathbf{e})$ and the varying Jacobian $\mathbf{J}(\mathbf{x}_c)$. The results for 6 different minimization methods are given in Figure 1. The contours represent isolines of the SSD (i.e. the cost function has the same value for each point of the contour) while the red lines represent the paths for each starting point. Obviously, the ideal path (i.e. the shortest one) would be a straight line from $\mathbf{x}_c$ to $\mathbf{e}$. Figure 1(a) shows that the varying Steepest Descent method moves always in a direction perpendicular to the isolines. For this reason, it has a slow convergence rate and cannot reach the minimum following a straight line. The paths for the constant Steepest Descent method are even longer (see the path lengths in Figure 1(b)). The constant (Figure 1(d)) and the varying (Figure 1(c)) Gauss-Newton methods performs better than the constant and the varying Steepest Descent methods respectively. In fact, in the constant and the varying Gauss-Newton methods a rough approximation of the Hessian is used. Ill conditioned and indefinite Hessian matrix causes the oscillations of the Newton method in Figure 1(e). Finally, the ESM method gives the best solution since the paths in Figure 1(f) are straight lines. Indeed, when the function $\mathbf{s}(\mathbf{x})$ is exactly quadratic we can correctly estimate the displacement in only one step and thus the correct descent direction regardless of the shape of the isolines.

B. Avoiding local minima

In the second simulation, we choose a different quadratic function $\mathbf{s}(\mathbf{x})$ such that the corresponding SSD cost function has a local minimum very close to the global minimum. The Newton method and methods with varying Jacobian fall into the local minimum when the starting point is close to it (see Figures 2(a), 2(c) and 2(e)). In that case, methods with constant Jacobian can eventually diverge (see Figures 2(b) and 2(d)). Indeed, the constant Jacobian approximation is valid only in a neighborhood of the true solution. On the other hand, the ESM method follows the shortest path (see Figure 2(f)). Thus, if $\mathbf{s}(\mathbf{x})$ is locally quadratic the ESM method is able to avoid local minima. Obviously, if the local minimum is far from the true minimum the second-order approximation is not valid any more.

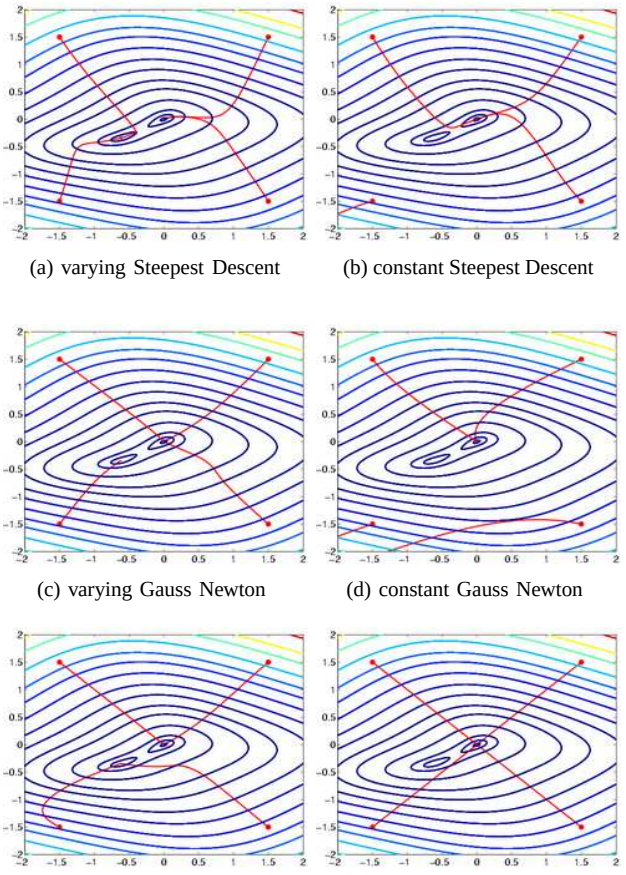


(a) varying Steepest Descent (b) constant Steepest Descent

(c) varying Gauss Newton (d) constant Gauss Newton

(e) Newton (f) ESM

Fig. 1. Comparing the behavior of 6 different minimization methods.



(a) varying Steepest Descent (b) constant Steepest Descent

(c) varying Gauss Newton (d) constant Gauss Newton

(e) Newton (f) ESM

Fig. 2. Comparing the behavior of 6 different minimization methods.

VI. Application to homographic motion model

In this section, we show how the ESM algorithm can be applied when the motion model is a homography. Obviously, it is also possible to add priors on the warping parameters by considering special cases of the homographic model such as the affine model or simple translations.

A. Homography warping

Let us suppose we want to track a planar object. When the object moves in the 3D space, the image transformation of the object is a homography. We need $p = 8$ parameters to define a homography. The homography is modeled by a (3×3) matrix $\mathbf{H}(\mathbf{x})$:

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} h_{11}(\mathbf{x}) & h_{12}(\mathbf{x}) & h_{13}(\mathbf{x}) \\ h_{21}(\mathbf{x}) & h_{22}(\mathbf{x}) & h_{23}(\mathbf{x}) \\ h_{31}(\mathbf{x}) & h_{32}(\mathbf{x}) & h_{33}(\mathbf{x}) \end{bmatrix}$$

where $\mathbf{x} = (x_1, x_2, ..., x_8)$ is the vector containing the homography parameters. A transformed image point can be written as follows:

$$\mathbf{p}' = \mathbf{w}(\mathbf{x})(\mathbf{p}) = \begin{bmatrix} \frac{h_{11}(\mathbf{x})u + h_{12}(\mathbf{x})v + h_{13}(\mathbf{x})}{h_{31}(\mathbf{x})u + h_{32}(\mathbf{x})v + h_{33}(\mathbf{x})} \\ \frac{h_{21}(\mathbf{x})u + h_{22}(\mathbf{x})v + h_{23}(\mathbf{x})}{h_{31}(\mathbf{x})u + h_{32}(\mathbf{x})v + h_{33}(\mathbf{x})} \end{bmatrix}$$

B. Homography parameterization

Choosing a homography parameterization consists in choosing the functions $h_{ij}(\mathbf{x})$ where $(i, j) \in \{1,2,3\} \times \{1,2,3\}$. There exists many parameterizations of the homography since it is defined up to a scale factor. Thus, a constraint on the matrix $\mathbf{H}$ must be added. Usually, $\mathbf{H}$ is chosen in $P(2)$ (the projective group) by setting $h_{33}(\mathbf{x}) = 1$. However, it is possible for big displacements of the camera to find $\mathbf{x}$ such that $h_{33}(\mathbf{x}) = 0$. For this reason, we prefer to choose $\mathbf{H}$ such that its determinant is equal to 1. This choice is well justified since $det(\mathbf{H}) = 0$ corresponds to the singular case when the observed plane passes through the optical center of the camera. In this singular case, all the points of the plane project into a line. The additional constraint is then $\mathbf{H} \in SL(3)$ (the Special Linear group of dimension 3). $SL(3)$ is the (3×3) group of matrices that have the determinant equal to 1. Let $sl(3)$ be the Lie algebra associated to this group. The matrices in this algebra have their trace equal to 0. The parameterization of $\mathbf{H}$ is done as follows. Let $\mathbf{G}_i$ $(\forall i \in \{1, 2, ...8\})$ be a basis of $sl(3)$ (i.e. linear independent constant matrices such that $trace(\mathbf{G}_i) = 0$). The matrix $\mathbf{A}(\mathbf{x})$ can be written as linear combination of the matrices $\mathbf{G}_i$:

$$\mathbf{A}(\mathbf{x}) = \sum_{i=1}^{8} x_i \mathbf{G}_i$$

The matrix $\mathbf{A}(\mathbf{x})$ is a matrix in $sl(3)$ and $\mathbf{H}(\mathbf{x}) = \exp(\mathbf{A}(\mathbf{x}))$. For this parameterization, we have $\mathbf{H}(\mathbf{e}) = \mathbf{I}$, $\mathbf{H}(\mathbf{x}_1 \circ \mathbf{x}_2) = \mathbf{H}(\mathbf{x}_1)\mathbf{H}(\mathbf{x}_2)$ and $\mathbf{H}^{-1}(\mathbf{x}) = \mathbf{H}(\mathbf{x}^{-1}) = \mathbf{H}(-\mathbf{x})$. Note that, we have $\mathbf{x}^{-1} = -\mathbf{x}$. We will show in the following section that this parameterization simplifies the computation of the Jacobian and is necessary to obtain the best performances from the ESM algorithm.

C. Computing the Jacobians

In order to be able to use equation (11), we need to compute $\mathbf{J}(\mathbf{x})$ for $\mathbf{x} = \mathbf{e}$ and $\mathbf{x} = \mathbf{x}_c$. Using equation (4), the k-th row $\mathbf{r}_k(\mathbf{x})$ of the matrix $\mathbf{J}(\mathbf{x})$ can be written as follows:

$$\mathbf{r}_k(\mathbf{x}) = \left. \frac{\partial I(\mathbf{w}(\mathbf{y})(\mathbf{p}_k))}{\partial \mathbf{y}} \right|_{\mathbf{y}=\mathbf{x}}$$

where $\mathbf{p}_k$ are the image coordinates of the k-th pixel $\forall k \in \{1, 2, ..., q\}$. Using the properties (1, 2, 3) of $\mathbf{w}(\mathbf{x})$, we have:

$$\mathbf{r}_k(\mathbf{x}) = \left. \frac{\partial I(\mathbf{w}(\mathbf{x})(\mathbf{w}(\mathbf{x}^{-1} \circ \mathbf{y})(\mathbf{p}_k)))}{\partial \mathbf{y}} \right|_{\mathbf{y}=\mathbf{x}} \quad (12)$$

which can be written as the following product:

$$\mathbf{r}_k(\mathbf{x}) = \left. \frac{\partial I(\mathbf{w}(\mathbf{x})(\mathbf{p}))}{\partial \mathbf{p}} \right|_{\mathbf{p}=\mathbf{p}_k} \left. \frac{\partial \mathbf{w}(\mathbf{x})(\mathbf{p}_k)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{e}} \left. \frac{\partial (\mathbf{x}^{-1} \circ \mathbf{y})}{\partial \mathbf{y}} \right|_{\mathbf{y}=\mathbf{x}} \quad (13)$$

The (1×2) vector $\left. \frac{\partial I(\mathbf{w}(\mathbf{x})(\mathbf{p}))}{\partial \mathbf{p}} \right|_{\mathbf{p}=\mathbf{p}_k}$ contains the gradient of the image (after being warped with $\mathbf{w}(\mathbf{x})$) computed at $\mathbf{p}_k$. Thus, if $\mathbf{x} = \mathbf{e}$ then $\left. \frac{\partial I(\mathbf{w}(\mathbf{e})(\mathbf{p}))}{\partial \mathbf{p}} \right|_{\mathbf{p}=\mathbf{p}_k}$ corresponds to the gradient computed at $\mathbf{p}_k$ in the reference image while if $\mathbf{x} = \mathbf{x}_c$ then $\left. \frac{\partial I(\mathbf{w}(\mathbf{x}_c)(\mathbf{p}))}{\partial \mathbf{p}} \right|_{\mathbf{p}=\mathbf{p}_k}$ corresponds to the gradient computed at $\mathbf{p}_k$ in the current warped image. It is very important to notice that the gradient can be computed without explicitly knowing the current parameters $\mathbf{x}_c$. The (2×p) Jacobian $\left. \frac{\partial \mathbf{w}(\mathbf{x})(\mathbf{p}_k)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{e}}$ does not depend on $\mathbf{x}$ and can be precomputed as in [10] or [12]. Finally, the third (p×p) Jacobian generally depends on $\mathbf{x}$. However, thanks to the parameterization of the homography using Lie Algebra, we have the following simplification formula:

$$\left. \frac{\partial (\mathbf{x}^{-1} \circ \mathbf{y})}{\partial \mathbf{y}} \right|_{\mathbf{y}=\mathbf{x}} \mathbf{x} = \mathbf{x} \quad (14)$$

This formula shows that we do not need to know $\mathbf{x}$ to compute the Jacobian. It will be used in the next section for computing the second-order approximation. This explains why the homography parameterization using Lie algebra is a key point of our algorithm. Another parameterization can imply a Jacobian matrix depending on $\mathbf{x}$. As a consequence, in that case, the Jacobian can only be approximated.

D. Computing the second-order approximation

Consider the second order approximation in equation (10), the i-th entry of the vector $\Delta\mathbf{s}$ can be written:

$$\Delta s_k \approx -\frac{1}{2}\mathbf{r}_k(\mathbf{e})\Delta\mathbf{x} - \frac{1}{2}\mathbf{r}_k(\mathbf{x}_c)\Delta\mathbf{x}$$

Using equation (13) we have:

$$\mathbf{r}_k(\mathbf{e})\,\Delta\mathbf{x} = \left. \frac{\partial I(\mathbf{p})}{\partial \mathbf{p}} \right|_{\mathbf{p}=\mathbf{p}_k} \left. \frac{\partial \mathbf{w}(\mathbf{x})(\mathbf{p}_k)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{e}} \Delta\mathbf{x}$$

From equation (14), since $\Delta\mathbf{x} = \mathbf{x}_c^{-1} = -\mathbf{x}_c$ we obtain:

$$\mathbf{r}_k(\mathbf{x}_c)\,\Delta\mathbf{x} = \left. \frac{\partial I(\mathbf{w}(\mathbf{x}_c)(\mathbf{p}))}{\partial \mathbf{p}} \right|_{\mathbf{p}=\mathbf{p}_k} \left. \frac{\partial \mathbf{w}(\mathbf{x})(\mathbf{p}_k)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{e}} \Delta\mathbf{x}$$

Then the second-order approximation can be written:

$$\Delta s_k \approx -\frac{1}{2}\left(\left.\frac{\partial I(\mathbf{p})}{\partial \mathbf{p}}\right|_{\mathbf{p}_k} + \left.\frac{\partial I(\mathbf{w}(\mathbf{x}_c)(\mathbf{p}))}{\partial \mathbf{p}}\right|_{\mathbf{p}_k}\right)\left.\frac{\partial \mathbf{w}(\mathbf{x})(\mathbf{p}_k)}{\partial \mathbf{x}}\right|_{\mathbf{e}}\Delta\mathbf{x}$$

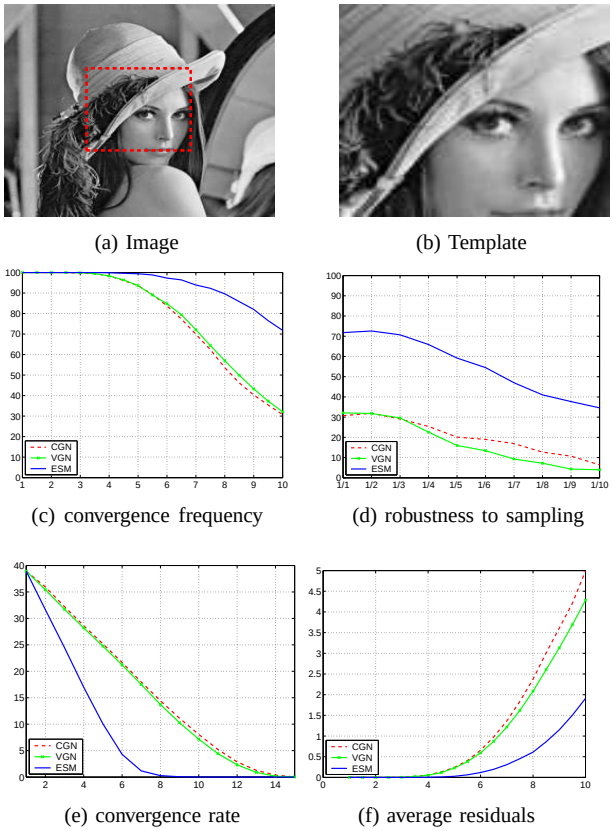
This equation shows that we can compute the second-order approximation in a very efficient way. Rearranging the rows of the Jacobians into a single matrix $\mathbf{J}(\mathbf{e}) + \mathbf{J}(\mathbf{x}_c)$ we can compute the state update as in equation (11).

VII. Experimental results

In order to have a ground truth, the ESM algorithm has been tested by warping a static image. In real conditions (with noise and light changes), it has been tested on a moving planar object and for tracking the back of a car.

A. Static object

We compare the ESM method with the constant Gauss Newton method (CGN) proposed in [12] and with the varying Gauss Newton method (VGN) proposed in [11]. We have used the Matlab software available on the web page of Simon Baker at the Robotics Institute of the Carnegie Mellon University (http://www.ri.cmu.edu/people/baker_simon.html). Thus, the performance of the algorithms have been compared with the same experimental setup. We use the well known Lena image shown in Figure 3(a). The (100×100) template illustrated in Figure 3(b) has been selected in the center of the image. The computational complexity of the ESM algorithm is equivalent to the VGN method which is higher than the CGN method. In order to have the same execution time per iteration, we can use a smaller subset (25 %) of the template for computing the Jacobians and the estimated displacement. The template is warped 1000 times using different random homographies. Similarly to [12], the homography is computed by adding a Gaussian noise to the coordinates of the 4 corners of the template. The standard deviation σ of the Gaussian noise is increased from 1 to 10. Figure 3(c) plots the frequencies of convergence (% over 1000 tests). As σ grows, the frequency of convergence of the CGN and the VGN methods decay quicker than the frequency of convergence of the ESM method. At the final $\sigma = 10$, the frequency of convergence of the CGN and the VGN methods are only 30% while the frequency of convergence of the ESM method is 70%. Figure 3(e) shows the average convergence rate (over the converged tests) of the algorithms for $\sigma = 10$. The initial SSD is the same for the three algorithms but the speed of convergence of the ESM method is much higher. This means that we can perform real-time tracking at higher rates. Since our objective is to track objects in real-time, it is very important to measure the residuals after each minimization. Indeed, since the number of iterations is fixed by the frame rate, the error will cumulate. Figure 3(f) plots the average residual over all the tests for which the algorithms did not diverge (we consider that the algorithm diverges when the final SSD is bigger than the initial SSD). Obviously the SSD increases with the amplitude of the initial displacement. However, the ESM method performs much better than the CGN method and the VGN method. Finally, we have tested the robustness of the algorithms to sampling. Figure 3(d) plots the frequency of convergence for $\sigma = 10$ against the sampling rate r between the size of the subset used in the algorithm and the size of the template e.g. for 1/1 we use all the template while for 1/10 we use 1 % of the template (1 pixel used every 10 pixels of the image per line and per column). The ESM algorithm is more robust to sampling. For $r = 1/10$, the frequency of convergence of the ESM method is almost the same as the two other methods without sampling. Thus, we can obtain the same frequency of convergence with a faster algorithm.



(a) Image (b) Template

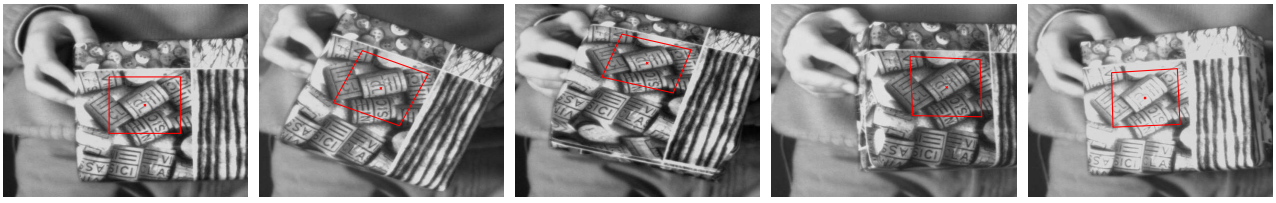
(c) convergence frequency (d) robustness to sampling

(e) convergence rate (f) average residuals

Fig. 3. Comparison between ESM and standard methods.

B. Moving object

The second-order tracking has been tested on sequences with moving planar objects. Five images have been extracted from each sequence and they are shown in the first row of Figure 4 and Figure 5. In the first experiment, the template to track is a (150×150) window shown in Figure 4(f). The red windows in the first row of Figure 4 are warped back and shown in the second row of Figure 4. Despite illumination changes and image noise, the warped windows are very close to the reference template proving that the tracking is accurately performed. During the sequence, a generic projective motion and several light variations have been observed. For example, Figures 4(b) and 4(c) shows translations and rotation around the $\vec{z}$ and $\vec{x}$ axis respectively, while Figure 4(d) and 4(e) shows a rotation around the $\vec{y}$ and varying illumination (the image becomes darker, the image becomes lighter). In the second

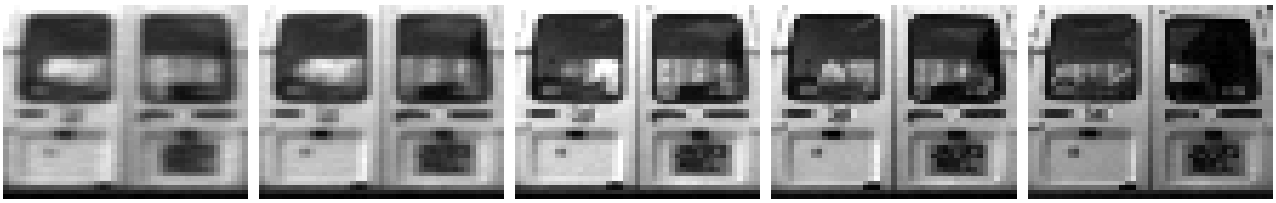


(a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4 (e) Image 5

(f) Window 1 (g) Window 2 (h) Window 3 (i) Window 4 (j) Window 5

Fig. 4. Tracking a template on a planar object.

(a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4 (e) Image 5



(f) Window 1 (g) Window 2 (h) Window 3 (i) Window 4 (j) Window 5

Fig. 5. Tracking a template on the back of a car.

experiment, we track a (43×43) template on the back of a car with a camera mounted on another car (see Figure 5(a) to (e)). Again, the tracking is accurately performed (see Figure 5(f) to (j)) in spite of the template changes due to people movement that we can see through the window of the car.

VIII. Conclusion

In this paper, we have proposed a real-time algorithm for tracking planar targets. We perform an efficient second-order approximation of the error using only first order derivatives (the ESM algorithm). This avoids the computation of the Hessian of the cost function. At the same time, the second order approximation allows the tracking algorithm to achieve a high convergence rate. This is very important if we want to track objects in real-time. Despite the ESM algorithm deals only with changes of the template due to the 3D motion of the plane, it can be extended in order to take into account illumination changes or transformed into a robust algorithm in order to take into account partial occlusions.

References

[1] S. Hutchinson, G. Hager, P. Corke, “A tutorial on visual servo control”, *IEEE TRA*, vol. 12, n. 5, p. 651-670, 1996.

[2] M. Isard and A. Blake, “Contour tracking by stochastic propagation of conditional density”, in *ECCV*, p. 343-356, 1996.

[3] P. Torr and A. Zisserman, “Feature based methods for structure and motion estimation”, in *Int. Work. on Vis. Alg.*, p. 278-295, 1999.

[4] T. Drummond and R. Cipolla, “Visual tracking and control using Lie algebras”, in *IEEE CVPR*, p. 652-657, 1999.

[5] M. Gleicher, “Projective registration with difference decomposition”, in *IEEE CVPR*, p. 331-337, 1997.

[6] F. Jurie and M. Dhome, “Hyperplane approximation for template matching”, *IEEE TPAMI*, vol. 24, n. 7, p. 996-1000, 2002.

[7] S. Sclaroff and J. Isidoro, “Active blobs”, in *IEEE ICCV*, p. 1146-1153, 1998.

[8] T. Cootes, G. Edwards, and C. Taylor, “Active appearence models”, in *ECCV*, p. 484-498, 1998.

[9] B. Lucas and T. Kanade, “An iterative image registration technique with application to stereo vision,” in *JCAI*, p. 674-679, 1981.

[10] G. Hager and P. Belhumeur, “Efficient region tracking with parametric models of geometry and illumination”, *IEEE TPAMI*, vol. 20, n. 10, p. 1025-1039, 1998.

[11] H. Shum and R. Szeliski, “Construction of panoramic image mosaics with global and local alignment”, *IJCV*, vol. 16, n. 1, p. 63-84, 2000.

[12] S. Baker and I. Matthews, “Equivalence and efficiency of image alignment algorithms”, in *IEEE CVPR*, p. 1090-1097, 2001.

[13] E. Malis, “Improving vision-based control using efficient second-order minimization techniques”, in *IEEE ICRA*, p. 1843-1848, 2004.