Robotics and Autonomous Systems 39 (2002) 205–221

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Foveated active tracking with redundant 2D motion parameters[☆]

Alexandre Bernardino[a,*], José Santos-Victor[a], Giulio Sandini[b]

[a] Instituto Superior Técnico, Instituto de Sistemas e Robótica, 1049-001 Lisboa, Portugal
[b] DIST, University of Genoa, Genoa, Italy

Abstract

This work presents a real-time active vision tracking system based on log-polar image motion estimation with 2D geometric deformation models. We present a very efficient parametric motion estimation method, where most computation can be done offline. We propose a redundant parameterization for the geometric deformations, which improve the convergence range of the algorithm. A foveated image representation provides extra computational savings and attenuation of background effects. A proper choice of motion models and a hierarchical organization of the iterations provide additional robustness. We present a fully integrated system with real-time performance and robustness to moderate deviations from the assumed deformation models. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Visual tracking; Motion estimation; Space-variant imaging; Active vision

1. Introduction

Visual fixation systems have increasingly important applications in robotics. The attempt to use robots on dynamic environments raised the need for better sensors and algorithms to keep track of the changes occurring in the external environment. Ranging from underwater vehicles or airships that must dock or keep their positions relative to a static surface despite external disturbances [15], to land robots that must follow or avoid moving objects [11], visual fixation can play an important role.

In this paper we present a system capable of fixating objects whose changes in appearance follow some a priori geometric deformation model. Often considered geometric models are the *affine*, *scaled-Euclidean* and *Euclidean*. The proposed algorithms cope with any parametric deformation model, including the general projective (planar) model. Many researchers have proposed methods to track the relative motion of planar patches [12]. Many rely on optic flow techniques [1,10] which are known to accumulate errors along time and drift from the expected solution. As in [8], our tracking algorithm is based on an initial template that is registered with the incoming images at every time step, and thus not prone to velocity estimation bias. At the algorithmic level we propose a computational framework with significant improvements over current methods: computational advantages arise from the specification of appropriate motion decomposition rules and the formulation of the optimization strategy based on time-fixed coordinates; convergence range is increased by using a redundant parameterization of geometric deformations; and robustness is improved by a hierarchical organization of the computations and a foveated imaging geometry.

We use a foveated log-polar image geometry, which was first motivated by its resemblance with the

☆ This work was partially funded by projects ESPRIT-LTR 30185—NARVAL.

* Corresponding author.
E-mail addresses: alex@isr.ist.utl.pt (A. Bernardino), jasv@isr.ist.utl.pt (J. Santos-Victor), sandini@dist.unige.it (G. Sandini).

structure of the retina of some biological vision systems and by its data compression qualities [13]. When compared to the usual Cartesian images, the log-polar images allow faster sampling rates on artificial vision systems without reducing the size of the field of view and the resolution on the central part of the retina (fovea) [14]. In the last years, however, it has been noticed that the log-polar geometry also provides important algorithmic benefits [4]. For instance in [2], it is shown that the use of log-polar images increases the size range of objects that can be tracked using a simple translation model. We show that increasing the "order" of the transformation towards more general models, these advantages are still observed.

This paper is organized in the following way: Section 2 describe the visual tracking algorithm, where for the sake of simplicity, the usual Cartesian representation of images is considered. This algorithm is easily extended to the log-polar representation, which is summarized in Section 3. The adopted geometric deformation models and the hierarchical structure of the algorithm are described in Section 4. Experiments with simulated and real setups are presented in Section 5, and finally in Section 6 we draw some conclusions.

2. Visual tracking with redundant 2D motion parameters

The tracking problem is formulated as computing the motion of an image region along time, assuming a geometric model for the 2D image deformations. Our contributions on this topic are based on three aspects: a description of motion as the composition of predicted and residual motion fields; the formulation of the objective function in time-fixed coordinates; and the representation of image deformations in a higher dimensional space (redundant parameterization).

This section is divided into three sections. First we review the parametric motion estimation problem, and develop two similar formulations in parallel. The first formulation is equivalent to the one presented in [8]. The second formulation, that we propose, is obtained from a change of time coordinates. The compared solutions are derived simultaneously and with the same notation, to clearly distinguish between them and realize how the different formulations produce algorithmic changes. In the second subsection we show that our formulation leads to significant computational savings and increased generality of application. Finally, in the third subsection we propose a redundant parameterization for the geometric deformations and show that it extends the convergence range of the algorithm.

2.1. Region based parametric motion estimation

Notation and problem formulation follow those presented in [8]. Let $I_t(\boldsymbol{x})$ denote image brightness of a pixel located at point $\boldsymbol{x} = (x, y)^T$ and at time t. We model image motion by the differentiable and invertible motion field $\boldsymbol{f}(\boldsymbol{x}; \mu)$, where $\mu = (\mu_1, \mu_2, \dots, \mu_n)^T$ is the motion parameter vector. A motion field maps 2D points to 2D points, representing point displacements (motion). The inverse motion field maps back points to their original locations, as illustrated in Fig. 1. The inverse motion field is defined such that

$$\boldsymbol{f}^{-1}(\boldsymbol{f}(\boldsymbol{x}; \mu); \mu) = \boldsymbol{f}(\boldsymbol{f}^{-1}(\boldsymbol{x}; \mu); \mu) = \boldsymbol{x}.$$

Solving the "tracking problem" consists in recovering the motion parameter vector for each time instant. The ground truth value is denoted by $\mu^*(t)$ and the corresponding estimate by $\mu(t)$. Initially, at time $t = 0$, a set of image pixels is selected, defining a reference region $\mathbb{R} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_m\}$. The reference template $R(\boldsymbol{x})$

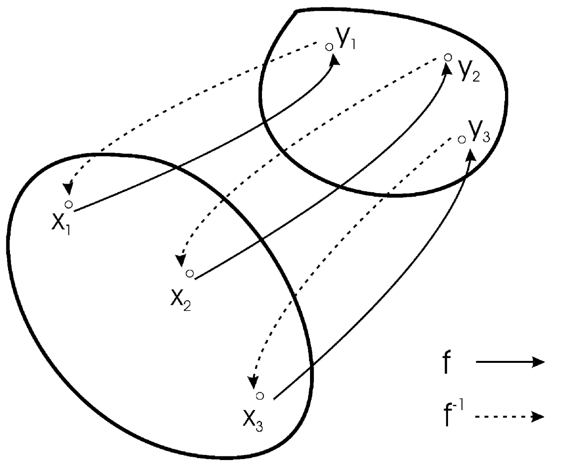


Fig. 1. Point coordinates x_i are mapped to points y_i according to motion field f, and can be mapped back to original coordinates with the inverse motion field f^{-1}.

is defined as the pixel gray-level values of region $\mathbb{R}$ at time $t = 0$:

$$R(\boldsymbol{x}) = I_0(\boldsymbol{x}) \quad \text{for all } \boldsymbol{x} \in \mathbb{R}.$$

With the brightness constancy assumption [9], all changes in image brightness in subsequent time steps are described by the motion parameters $\mu^*(t)$ and the motion field $\boldsymbol{f}$

$$I_0(\boldsymbol{x}) = I_t(\boldsymbol{f}(\boldsymbol{x}; \mu^*(t))) \quad \text{for all } \boldsymbol{x} \in \mathbb{R}$$

or equivalently by the *inverse* motion field $\boldsymbol{f}^{-1}$

$$I_t(\boldsymbol{x}) = I_0(\boldsymbol{f}^{-1}(\boldsymbol{x}; \mu^*(t))) \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^* = \boldsymbol{f}^{-1}(\mathbb{R}, \mu^*).$$

2.1.1. Optimization framework

Usual optimization techniques can recover the motion parameters by minimizing a least squares objective function. This function can either express the difference between the reference template and the current image transformed by the motion field:

$$O_1(\mu) = \sum_{\boldsymbol{x}\in\mathbb{R}} [I_t(\boldsymbol{f}(\boldsymbol{x}; \mu)) - I_0(\boldsymbol{x})]^2 \tag{1}$$

or express the difference between the current image and the *inverse* transformed reference template:

$$O_2(\mu) = \sum_{\boldsymbol{x}\in\mathbb{R}^*} [I_t(\boldsymbol{x}) - I_0(\boldsymbol{f}^{-1}(\boldsymbol{x}; \mu))]^2.$$

Remark 1. In the first formulation the optimization parameters are contained in a *time-varying* entity while in the second formulation they parameterize the *time-fixed* reference template. Notwithstanding, both formulations are equivalent.

2.1.2. Motion decomposition

An usual assumption in this kind of problems is to consider that motion is "smooth", i.e. it is continuous and does not suffer abrupt changes. This is not an unrealistic assumption since motion arises from the displacement of physical objects, which is constrained by inertial physical laws. This fact provides a starting point for the search of target motion at time instant t, based on information of past time steps. We will denominate this starting point as "initial guess" or "motion prediction" and represent it as $\bar{\mu}$. It can be obtained simply as the estimate of the motion field parameters in the previous time instant $\mu(t-1)$ or by a suitable prediction based on the past time information and/or motion model, like in a Kalman filter [5]. In general, the prediction does not coincide with the true parameters and a residual error $\tilde{\mu}$ remains. This residue is also called "innovation term" since it contains the component of motion that cannot be predicted and must be computed by image processing algorithms. Using these two components (prediction $\bar{\mu}$ and innovation $\tilde{\mu}$), we define the composition rule that generates the full motion field:

$$\boldsymbol{f}(\boldsymbol{x}; \mu) = \boldsymbol{f}(\boldsymbol{f}(\boldsymbol{x}; \bar{\mu}); \tilde{\mu}).$$

Given the knowledge of the predicted motion vector at time t, we can recast the tracking problem as one of determining the "innovation term" $\tilde{\mu}(t)$. This can be obtained by minimizing a least-squares objective function:

$$O_1(\tilde{\mu}) = \sum_{\boldsymbol{x}\in\mathbb{R}} [I_t(\boldsymbol{f}(\boldsymbol{f}(\boldsymbol{x}; \bar{\mu}); \tilde{\mu}))) - I_0(\boldsymbol{x})]$$

or equivalently

$$O_2(\tilde{\mu}) = \sum_{\boldsymbol{x}\in\tilde{\mathbb{R}}} [I_t(\boldsymbol{f}(\boldsymbol{x}; \bar{\mu})) - I_0(\boldsymbol{f}^{-1}(\boldsymbol{x}; \tilde{\mu}))].$$

In the latter case the region of summation is given by $\tilde{\mathbb{R}} = \boldsymbol{f}^{-1}(\mathbb{R}, \tilde{\mu})$. It is easy to show that $O_1 = O_2$. To simplify notation, we define the following vectors:

$$\boldsymbol{I}_t(\mu) \triangleq \text{vec}[I_t(\boldsymbol{f}(\boldsymbol{x}; \mu))],$$
$$\bar{\boldsymbol{I}}_t(\tilde{\mu}) \triangleq \text{vec}[I_t(\boldsymbol{f}(\boldsymbol{f}(\boldsymbol{x}; \bar{\mu}); \tilde{\mu}))],$$
$$\boldsymbol{I}_0(\tilde{\mu}) \triangleq \text{vec}[I_0(\boldsymbol{f}^{-1}(\boldsymbol{x}; \tilde{\mu}))],$$

where the operator "vec" represents the stacking of all image pixels into a long column vector. The objective functions can be rewritten as

$$O_1(\tilde{\mu}) = \sum_{\mathbb{R}} [\bar{\boldsymbol{I}}_t(\tilde{\mu}) - \boldsymbol{I}_0(\boldsymbol{0})]^2,$$
$$O_2(\tilde{\mu}) = \sum_{\mathbb{R}'} [\bar{\boldsymbol{I}}_t(\boldsymbol{0}) - \boldsymbol{I}_0(\tilde{\mu})]^2.$$

We call $\boldsymbol{I}_t(\mu)$ the *registered image*. Image $\bar{\boldsymbol{I}}_t(\boldsymbol{0}) = \boldsymbol{I}_t(\bar{\mu})$ is called the *predicted registration* and can be computed by rectifying the acquired image $I_t(\boldsymbol{x})$ with the predicted motion vector $\bar{\mu}$.

Remark 2. In both objective functions one of the images is known and fixed while the other is a variable dependent of the optimizing parameters $\tilde{\mu}$.

2.1.3. Local approximation

Assuming small magnitude for the components of $\tilde{\mu}$ we can make the following approximations:

- The regions of summation $\tilde{\mathbb{R}}$ and $\mathbb{R}$ are similar:

  $$\tilde{\mathbb{R}} \approx \mathbb{R}.$$

- The image $\bar{\boldsymbol{I}}_t(\tilde{\mu})$ at *fixed time t* can be approximated by a first-order McLaurin series expansion with respect to the motion parameters:

  $$\bar{\boldsymbol{I}}_t(\tilde{\mu}) \approx \bar{\boldsymbol{I}}_t(\mathbf{0}) + \bar{\boldsymbol{M}}_t \cdot \tilde{\mu}, \tag{2}$$

  where the matrix $\bar{\boldsymbol{M}}_t$ is the $m \times n$ matrix of partial derivatives of the *predicted registration* $\bar{\boldsymbol{I}}_t(\mathbf{0})$ written in column form:

  $$\bar{\boldsymbol{M}}_t = \left[\frac{\partial \bar{\boldsymbol{I}}_t}{\partial \tilde{\mu}_1}(\mathbf{0}) | \cdots | \frac{\partial \bar{\boldsymbol{I}}_t}{\partial \tilde{\mu}_n}(\mathbf{0})\right].$$

- The image $\boldsymbol{I}_0(\tilde{\mu})$ can be approximated by a first-order McLaurin series expansion:

  $$\boldsymbol{I}_0(\tilde{\mu}) \approx \boldsymbol{I}_0(\mathbf{0}) + \boldsymbol{M}_0 \cdot \tilde{\mu},$$

  where the constant matrix $\boldsymbol{M}(\mathbf{0})$ is the $m \times n$ matrix of partial derivatives of the *reference image* $\boldsymbol{I}_0(\mathbf{0})$ written in column form:

  $$\boldsymbol{M}_0 = \left[\frac{\partial \boldsymbol{I}_0}{\partial \tilde{\mu}_1}(\mathbf{0}) | \cdots | \frac{\partial \boldsymbol{I}_0}{\partial \tilde{\mu}_n}(\mathbf{0})\right].$$

With these assumptions we can rewrite the objective functions as follows:

$$O_1(\tilde{\mu}) = \sum_{\mathbb{R}} [\boldsymbol{D}_t + \bar{\boldsymbol{M}}_t \cdot \tilde{\mu}]^2,$$
$$O_2(\tilde{\mu}) = \sum_{\mathbb{R}} [\boldsymbol{D}_t - \boldsymbol{M}_0 \cdot \tilde{\mu}]^2, \tag{3}$$

where $\boldsymbol{D}_t = \bar{\boldsymbol{I}}_t(\mathbf{0}) - \boldsymbol{I}_0(\mathbf{0})$ is the error between the predicted registration and the reference template.

Remark 3. In the first formulation the matrix of partial derivatives is time-varying while in the second formulation it is fixed for all time instances.

2.1.4. Computing the solution

Since both objective functions are quadratic functions of the residual motion parameters, solutions to the optimization problem can be obtained in closed form by solving the set of equations $\nabla O = 0$. The solution yields in the first case

$$\tilde{\mu} = -(\bar{\boldsymbol{M}}_t^{\mathrm{T}} \bar{\boldsymbol{M}}_t)^{-1} \bar{\boldsymbol{M}}_t^{\mathrm{T}} \boldsymbol{D}_t \tag{4}$$

and in the second formulation

$$\tilde{\mu} = (\boldsymbol{M}_0^{\mathrm{T}} \boldsymbol{M}_0)^{-1} \boldsymbol{M}_0^{\mathrm{T}} \boldsymbol{D}_t. \tag{5}$$

Care should be taken with possible singularities in the motion covariance matrices $\bar{\boldsymbol{M}}_t^{\mathrm{T}} \bar{\boldsymbol{M}}_t$ and $\boldsymbol{M}_0^{\mathrm{T}} \boldsymbol{M}_0$. This may happen when there is not sufficient texture in the interest region and certain object motions are not observable from the image gray-level variations. This is a generalization of the aperture problem [9].

2.2. Minimizing online computations

The work presented in [8] is based on an objective function of type O_1. Although derived with a different motion decomposition rule and temporal analysis, the objective function is equivalent to (3). Functions O_1 and O_2 have a similar aspect but the former contains a Jacobian matrix $\bar{\boldsymbol{M}}_t$ that is time dependent while the latter contains a *constant* Jacobian matrix $\boldsymbol{M}_0$. We propose a formulation based on objective function O_2. An objective function of this type is computationally advantageous since the Jacobian matrix can be computed at initialization of the reference template while the previous formulation requires the computation of the Jacobian (or part of it) at run time.[1]

2.2.1. Algorithmic efficiency

In an algorithmic point of view, there are some operations that can be performed in a *offline* phase

[1] In [8], the Jacobian is decomposed in the product of: image spatial gradient; motion field spatial derivatives; and motion field derivatives with respect to the motion parameters. Image spatial gradients can be calculated offline, on the reference template. Additionally, for some particular motion models, the Jacobian matrix can be written as a product of a constant $m \times k$ matrix and a time-varying $k \times n$ matrix, saving some online computation. However, there is always part of the Jacobian that must be computed online.

Table 1
Functional comparison between the proposed tracking algorithms

Framework 1	Framework 2
Offline steps	
Define the target region	Define the target region
Acquire and store the reference template	Acquire and store the reference template
–	Compute M_0
–	Compute $M_0^+ = (\boldsymbol{M}_0^{\mathrm{T}}\boldsymbol{M}_0)^{-1}\boldsymbol{M}_0^{\mathrm{T}}$
Online steps	
Acquire new image	Acquire new image
Use a suitable motion prediction $\bar{\mu}$ to rectify the target region into the current image	Use a suitable motion prediction $\bar{\mu}$ to rectify the target region into the current image
Compute $\boldsymbol{D}_t$ by taking the difference between the predicted registration and the reference template	Compute $\boldsymbol{D}_t$ by taking the difference between the predicted registration and the reference template
Compute $\bar{\boldsymbol{M}}_t$	–
Compute $\bar{\boldsymbol{M}}_t^+ = (\bar{\boldsymbol{M}}_t^{\mathrm{T}}\bar{\boldsymbol{M}}_t)^{-1}\bar{\boldsymbol{M}}_t^{\mathrm{T}}$	–
Compute $\tilde{\mu} = \bar{\boldsymbol{M}}_t^+\boldsymbol{D}_t$	Compute $\tilde{\mu} = \boldsymbol{M}_0^+\boldsymbol{D}_t$
Compute μ by composing transformations $\bar{\mu}$ and $\hat{\mu}$	Compute μ by composing transformations $\bar{\mu}$ and $\hat{\mu}$

(initialization) and other are performed *online* (at each time step). Since the derivation of the algorithm is based on local linearization, for good run-time performance the sampling period should be kept at minimum. Therefore, the online computation should be as fast as possible. Table 1 describes two computational algorithms that implement the solutions expressed in Eqs. (4) and (5).

2.2.2. Image warping

Image warping is the process of obtaining a new image by applying a geometric transformations (motion fields) to an original image. This process is needed to compute the *predicted registration* image. It is also used to compute image partial derivatives. A very simple means of implementing this procedure is by using look-up tables expressing the correspondences between pixel locations in the original and target image regions. More sophisticated methods can employ some kind of interpolation to improve the resulting image quality. In terms of computational complexity, image warping is $\mathrm{O}(m)$, where m is the number of pixels in the interest region.

2.2.3. Discrete derivatives

In both formulations we must compute partial derivatives of images with respect to the motion parameters. In framework 1 we must compute:

$$\bar{\boldsymbol{I}}_t^{(i)}(\mathbf{0}) = \left.\frac{\partial \bar{\boldsymbol{I}}_t(\tilde{\mu})}{\partial \tilde{\mu}_i}\right|_{\tilde{\mu}=0}, \quad i = 1, \dots, n$$

and, in framework 2:

$$\boldsymbol{I}_0^{(i)}(\mathbf{0}) = \left.\frac{\partial \boldsymbol{I}_0(\tilde{\mu})}{\partial \tilde{\mu}_i}\right|_{\tilde{\mu}=0}, \quad i = 1, \dots, n.$$

A possible way to obtain discrete approximations to the partial derivatives consists in applying the formulas:

$$\bar{\boldsymbol{I}}_t^{(i)}(\mathbf{0}) \approx \frac{\bar{\boldsymbol{I}}_t(h \cdot \boldsymbol{e}_i) - \bar{\boldsymbol{I}}_t(0)}{h},$$
$$\boldsymbol{I}_0^{(i)}(\mathbf{0}) \approx \frac{\boldsymbol{I}_0(h \cdot \boldsymbol{e}_i) - \boldsymbol{I}_0(0)}{h},$$

where h is a "small" constant and $\boldsymbol{e}_i$ is a vector with value 1 at position i and 0 at all other positions. Using this method, to compute each partial derivative we must perform one image warping and one image difference. Therefore, the computation of $\boldsymbol{M}_0$ has complexity $\mathrm{O}(n \times m)$.

2.2.4. Online complexity

Let us concentrate on the online complexity of the algorithms, which is the most important for real-time

Table 2
Online computational complexity

Step	Operation	Algorithm	Complexity
1	Acquire image	1, 2	–
2	Rectify image	1, 2	$\mathrm{O}(m)$
3	Compute $\boldsymbol{D}_t$	1, 2	$\mathrm{O}(m)$
4	Compute $\bar{\boldsymbol{M}}_t$	1	$\mathrm{O}(m \times n)$
5	Compute $\bar{\boldsymbol{M}}_t^{\mathrm{T}}\bar{\boldsymbol{M}}_t$	1	$\mathrm{O}(m \times n^2)$
6	Compute $(\bar{\boldsymbol{M}}_t^{\mathrm{T}}\bar{\boldsymbol{M}}_t)^{-1}$	1	$\mathrm{O}(n^3)$
7	Compute $\bar{\boldsymbol{M}}_t^{+}$	1	$\mathrm{O}(m \times n^2)$
8	Compute $\tilde{\mu}$	1, 2	$\mathrm{O}(m \times n)$
9	Compute μ	1, 2	$\mathrm{O}(n)$

performance. Table 2 presents the required online operations as well as the computational complexity of each step. Assuming fixed dimension for the motion parameter vector, the overall complexity for each case is $\mathrm{O}(m)$. However, we can observe that algorithm 2 saves a great amount of computation because it does not need to compute online steps 4–7 (they are computed offline). Fig. 2 presents the total number of arithmetic operations performed on the online steps of each algorithm as a function of the number of pixels. Data is obtained for $n = 8$ (planar motion model). The gain in efficiency is obvious. Algorithm 2 online performance is about 15 times faster.

2.3. Redundant parameterization of geometric deformations

The motivation to propose a redundant parameterization for the motion vector comes from realizing that representing image deformations in a Taylor series expansion like Eq. (2) may not be very powerful. Considering that image regions may have hundreds of pixels (i.e. a vector in a very high dimension space) and that common motions models have few parameters, then representing one image by a linear combination of a few basis images (the partial derivatives) can lead to very bad approximation. Obviously, the approximation quality depends on image

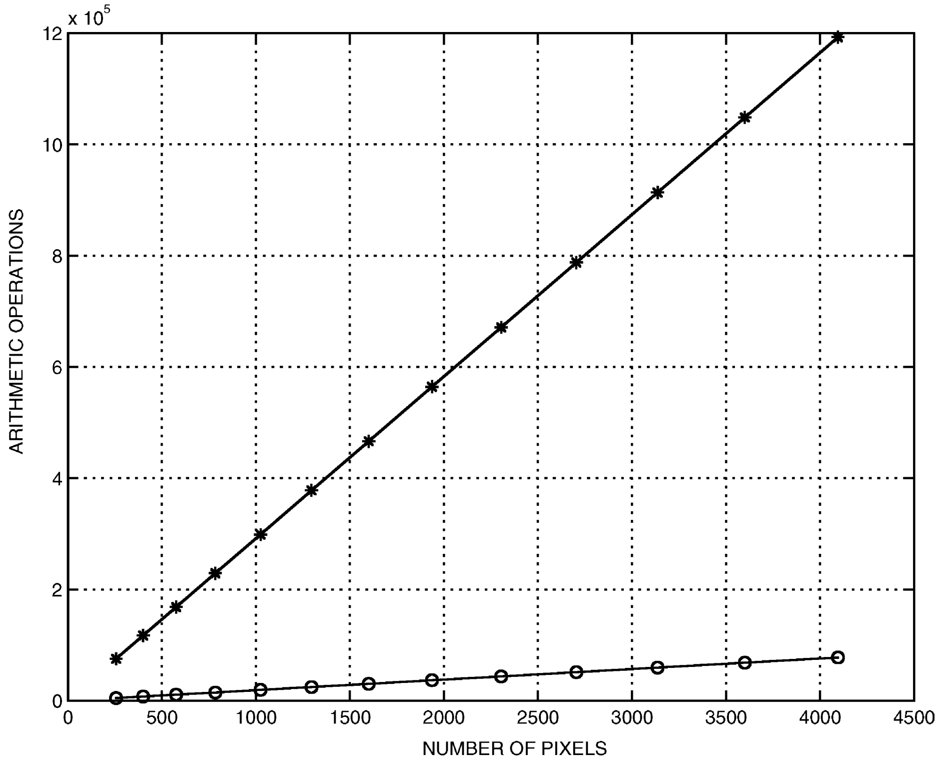


Fig. 2. Number of operations for algorithms 1 ($*$) and 2 (○) as function of region dimension. Results obtained with $n = 8$.

texture content but in general the approximation is only valid for very small perturbations. We propose to improve the approximation by enriching the linear combination with discrete derivatives at several directions and scales. A similar approach is proposed in [6].

Let us define a set of redundant sample vectors $\mathbb{V} = \{\tilde{\mu}_i, i \in (1, \dots, s)\}, s \gg n\}$. This set must be complete, such that any motion vector can be represented as a linear combination vectors on $\mathbb{V}$:

$$\tilde{\mu} = \sum_{i=1}^{s} k_i \cdot \tilde{\mu}_i. \tag{6}$$

Since this new basis is redundant, multiple solutions may exist for the coefficients of the linear combination. Coefficients, $\boldsymbol{k} = (k_1, \dots, k_s)^{\mathrm{T}}$ represent a new set of parameters for the geometric deformation and implicitly define a new representation for image warping:

$$\boldsymbol{I}_0(\boldsymbol{k}) = \boldsymbol{I}_0(\tilde{\mu}) = \boldsymbol{I}_0\left(\sum_{i=1}^{s} k_i \cdot \tilde{\mu}_i\right).$$

The discrete partial derivatives of this new representation come:

$$\left.\frac{\partial \boldsymbol{I}_0(\boldsymbol{k})}{\partial k_i}\right|_{\boldsymbol{k}=0} = \frac{\boldsymbol{I}_0(h \cdot \tilde{\mu}_i) - \boldsymbol{I}_0(0)}{h}.$$

The discretization step h can be made unity and, in such case, the magnitude of the sample vectors $\tilde{\mu}_i$ represent the discretization scale. With this new parameterization, the image partial derivatives represent the derivatives at multiple directions and scales in the original representation (direction and scale of each $\tilde{\mu}_i$). The proposed motion estimation algorithm can be applied with no changes to this new representation. The Jacobian matrix comes

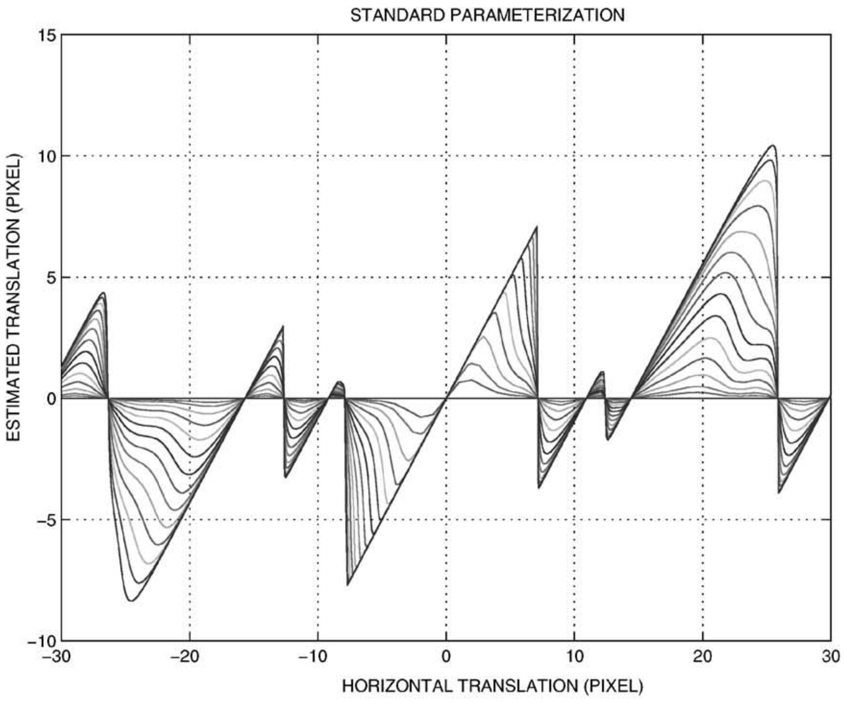
$$\boldsymbol{M}_0 = \left[\left.\frac{\partial \boldsymbol{I}_0(\boldsymbol{k})}{\partial k_1}\right|_{\boldsymbol{k}=0} | \cdots | \left.\frac{\partial \boldsymbol{I}_0(\boldsymbol{k})}{\partial k_s}\right|_{\boldsymbol{k}=0} \right]$$

which has dimension $m \times s$. The optimization process computes a solution for $\boldsymbol{k}$ and the solution for $\tilde{\mu}$ is given by Eq. (6). The computation time increases with the number of sample vectors, but since most of the computations are done offline, real-time performance is still obtained. In our experiments, about 100–150 sample vectors are used and the algorithms run at 25 Hz on a Pentium 400 MHz computer. Notice that, even though the redundant basis motion vectors are linearly dependent, in general this does not happen with the basis images $\boldsymbol{I}_0(\tilde{\mu}_i)$, since they are represented in a much higher dimension space. However, this depends on image texture and care should be taken when running the optimization algorithm. We use a damped least-squares method [3] to compute the Jacobian matrix pseudo-inverse.

One of the advantages of the redundant parameterization over the standard one is the ability to customize the set $\mathbb{V}$ of sample vectors according to the kind and range of expected image deformations. Also with the increase of computational power, we can easily add new sample vectors to improve the estimation results. The algorithm can be customized in order to estimate the larger and more constrained motions in the first iterations and the finer and more generic transformations in the last iterations, which improve its robustness.

2.3.1. Experiment

Let us consider one simple example and compare the performance of the standard and the redundant parameterizations. The experiment consists in simulating image translations from -30 to 30 pixels in small steps (0.1 pixels). For each translation we apply both approaches and compare the solutions. Results are presented in Fig. 3. The first plot is relative to translation estimation with the standard representation. Several curves are presented, corresponding to the estimated translation with different number of iterations $(10, 20, \dots, 150)$. We can observe that in this case the convergence interval is limited to about ± 8 pixels. Another aspect of concern is the convergence speed. The number of iterations depends on the required precision—if translation is small, good estimates can be obtained with a few iterations, but more iterations are required in the limits of the convergence interval. The second plot shows the performance with the redundant parameterization. Again several curves are presented for the evolution of the estimation process with different number of iterations $(2, 4, \dots, 14)$. We can observe that the convergence interval is much larger in comparison with the previous method. Also, the convergence rate is higher—the algorithm reaches a stable solution in about 10 iterations.



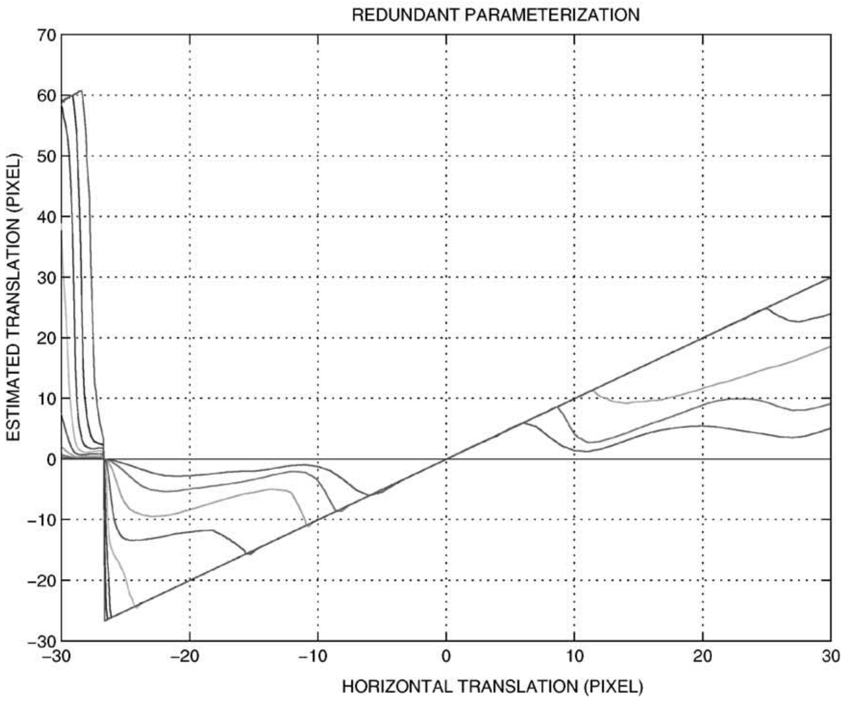


Fig. 3. Comparison of different parameterizations for geometric deformation: standard parameterization (top). Notice the limited convergence range (about ±8 pixels). Different lines correspond to different numbers of iterations $(10, 20, \dots, 150)$. Redundant parameterization (bottom). Different lines correspond to different numbers of iterations $(2, 4, \dots, 14)$.

3. Foveated images

The system described in this paper is based on a log-polar space variant image sampling. One advantage of this kind of sampling is data reduction. This is obtained by reducing the resolution at the image periphery, but still keeping the whole field of view and high resolution in the center (fovea). We map 128×128 Cartesian images to 64×32 log-polar images, achieving eight times increase in efficiency (both storage and speed).

The log-polar transformation, $\boldsymbol{l}(\boldsymbol{x})$, is defined as a conformal mapping from points on the *Cartesian* plane $\boldsymbol{x}=(x,y)$ to points in the *cortical* plane $z=(\xi,\eta)$ [13]:

$$\boldsymbol{l}(\boldsymbol{x})=\begin{bmatrix}\xi\\ \eta\end{bmatrix}=\begin{bmatrix}\log(\sqrt{x^2+y^2})\\ \arctan(y/x)\end{bmatrix}.$$

For active tracking purposes, the main advantage of the log-polar geometry is that objects occupying the central high resolution part of the visual field become dominant over coarsely sampled background elements in the image periphery. This embeds an implicit focus of attention in the center of the visual field where the target is expected to be most of the time.

Although the derivation of the motion estimation algorithm was done considering Cartesian coordinates (x,y), its extension to log-polar coordinates is straightforward. Notice that intuitive deformations in the Cartesian image plane (e.g. translation, rotation, etc.) correspond to not so intuitive deformations in log-polar coordinates (see Fig. 4). Thus we prefer to define 2D transformations in Cartesian coordinates. Then we express the corresponding log-polar deformations in terms of a map between the Cartesian and log-polar motion fields, in the following way:

$$\boldsymbol{f}^{\log}(z;\mu)=\boldsymbol{l}(\boldsymbol{f}(\boldsymbol{l}^{-1}(z);\mu)). \tag{7}$$

In terms of computation complexity, the transformation of an image according to a deformation field is the same in Cartesian and log-polar images. However, since log-polar images have less pixels, these transformations are faster to compute, which is important to achieve high sampling rates and better tracking performance. Each image warping takes about 10 ms on a Pentium 400 MHz computer, including the computation of the log-polar motion field $\boldsymbol{f}^{\log}$.

4. Algorithm implementation

To implement an algorithm based on the proposed method, some design choices must be taken, such as: (i) the deformation model, (ii) the number and distribution of sample vectors $\tilde{\mu}_i$; (iii) the iterative (or not) structure of the algorithm, i.e. the number of iterations and/or the stopping criteria.

The choice of a deformation model depends on the considered application. Target shape and motion should be taken into account when deciding this point. One thing to take into consideration is that more constrained transformations (with less parameters) are more robust to non-modeled aspects of the deformations. Less constrained models (with mode degrees of freedom) are in general less stable but for some applications may be required. Planar surfaces are often used in robotic applications, since they can be found in many human made environments and represent good approximations for other kind of

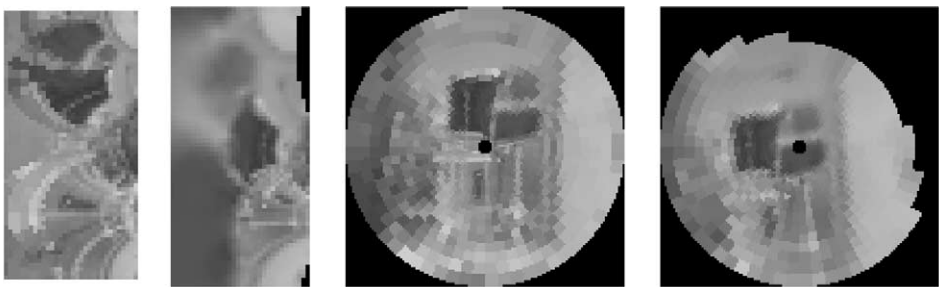


Fig. 4. The original log-polar (on the left) is warped according to a transformation that includes translation and rotation (second from the left). The corresponding retinal images are also shown (right).

surfaces. For example, in aerial mapping or ocean floor exploration, the ground can be approximated by a plane if the camera is distant enough [7]. Therefore, planar surfaces provide an adequate and yet manageable model to work with. In these cases, the use of a projective motion model can provide very good motion estimates which may be needed for precise pose computation or trajectory generation. In this paper we use two types of motion models. In the simulations we use the projective model, thus being able to estimate all deformations of planar surfaces motion. The projective model has 8 degrees of freedom and includes deformations such as translation, rotation, scaling, shear and curl, as illustrated in Fig. 5. In the active tracking tests, we use a rigid motion model, composed by translation and rotation.

In terms of number of iterations, we chose to have a fixed number. Since we are mostly interested in real-time implementations it is more important to have a fixed computation time than a very precise tracking, therefore we fix the number of iterations at 3 for the planar model an at 2 for the rigid model.

Regarding the choice of the sample vectors, and after many experiments we chose to create three sets for the planar model: one with translation vectors, one with affine vectors and one with projective vectors. Each set is composed by 48 vectors with non-uniform distributions, sampling more densely small deformations but still considering large deformations. For instance, the sample translation set is composed by vectors that translate the template by amounts $(x, y) \in \{-6, -3, -1, 1, 3, 6\}^2$. The idea is to have good precision when the deformations are small but still be able to detect large deformations. The three iterations are organized in the following way:

- The first iteration uses *sample translation vectors*. Since in the beginning a large deformation is likely to exist, it is more robust to estimate more constrained transformations. This iteration is intended to center the template with the current image and leave to the next iterations the estimation of the remaining deformations.
- The second iteration uses *sample affine vectors*. This iteration should estimate most of the rotation, scaling and shear present in the transformation.
- The last iteration uses *sample projective vectors*. It should estimate the remaining deformations and

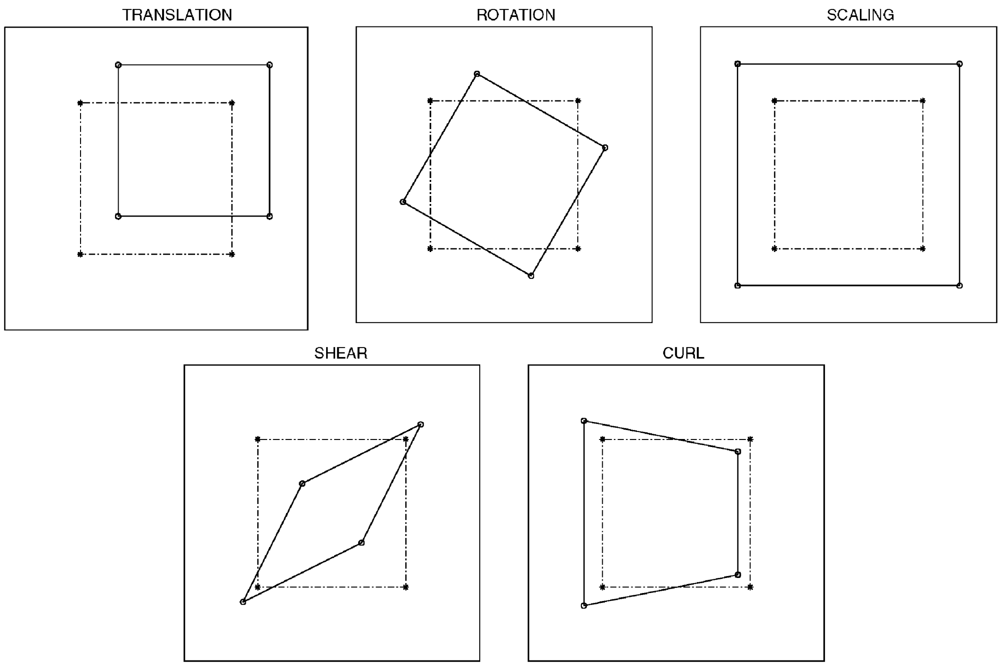


Fig. 5. 2D planar transformations are composed by translation, rotation, scaling, shear and curl.

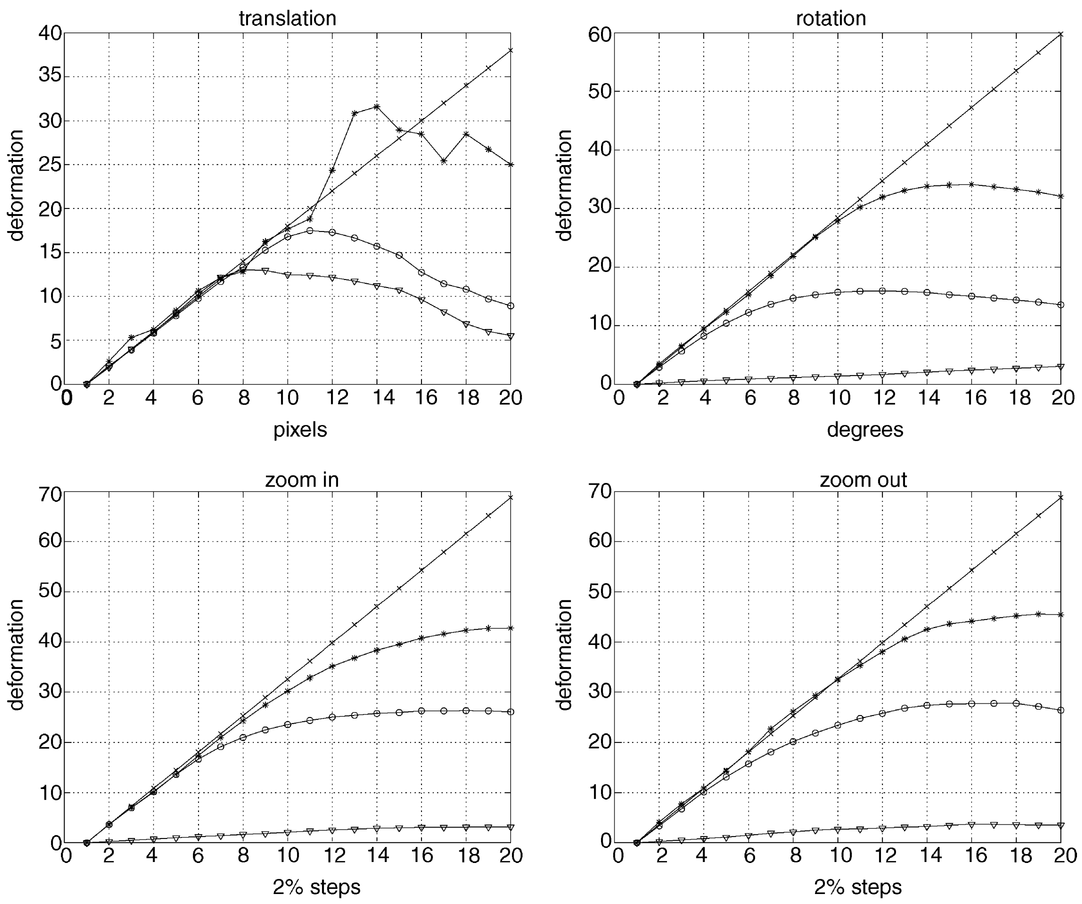


Fig. 6. Performance of the algorithm for translation, rotation, zoom-in and zoom-out transformations. Ground truth (×); first iteration (translation sample vectors) (▽); second iteration (affine sample vectors) (○); third iteration (planar sample vectors) (∗).

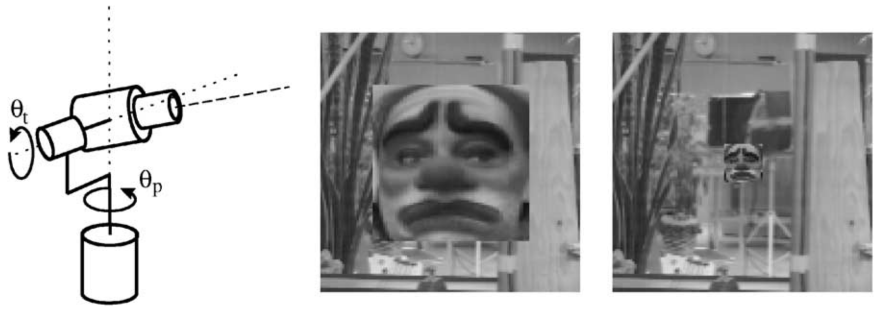


Fig. 7. Pan/tilt camera (left). The axis of rotation are assumed to intersect in the camera optical center. Simulated images with targets of scales 36% and 2.25% (right).

make the final fine adjustment to the template. Since this set spans 8 degrees of freedom, it should be used only with small deformations, otherwise it is likely to produce erroneous results.

Following the same ideas, for the rigid model we use two sets of vectors (one for each iteration). The first one uses the same 48 translation vectors. The second use 24 rotations, also with non-uniform distribution around the origin.

5. Results

Several experiments are shown to evaluate the performance of the proposed methodologies. In particular we are interested in testing the motion estimation algorithm, evaluating the benefits of using foveated images and evaluating the system in real situations, with objects deviating from the assumed deformation models.

In the first experiment (performance evaluation) we simulate camera motions that produce, in the retinal plane, increasing image translations, rotations and scalings. This experiment shows the range limitations of the algorithm, i.e. the maximal image deformations allowed between two consecutive images.

In the second set of experiments we compare between using Cartesian and log-polar images. Again we simulate image motion for objects of different sizes in order to have ground truth data.

In the third experiment we use the real setup and evaluate qualitatively the performance of the full system and its ability to cope with non-modeled aspects.

5.1. *Performance evaluation*

In these experiments we evaluate the algorithm convergence range with translations, rotations and scalings. The algorithm is applied to increasingly bigger deformations and at each iteration the estimated and real transformations are compared. The deformation measure is given by the $L2$-norm of the vector that contains the corner displacements (in pixel) of a polygonal window defining the template. The results are presented in Fig. 6.

With the images and transformations used, the algorithm is able to estimate with precision (error less than half pixel per window corner) about 10 pixels translation, 11 degrees rotation, 18% zoom-in or 24% zoom-out. This corresponds to the biggest motion that the algorithm can cope with, between two consecutive images. It is interesting to note that the sample translation vectors do most of the work when the deformation consists solely of translation but are almost useless when the deformation does not include translations. However, for the problems we are interested with, translation is a dominant deformation and is essential to include translation vectors in the sample set.

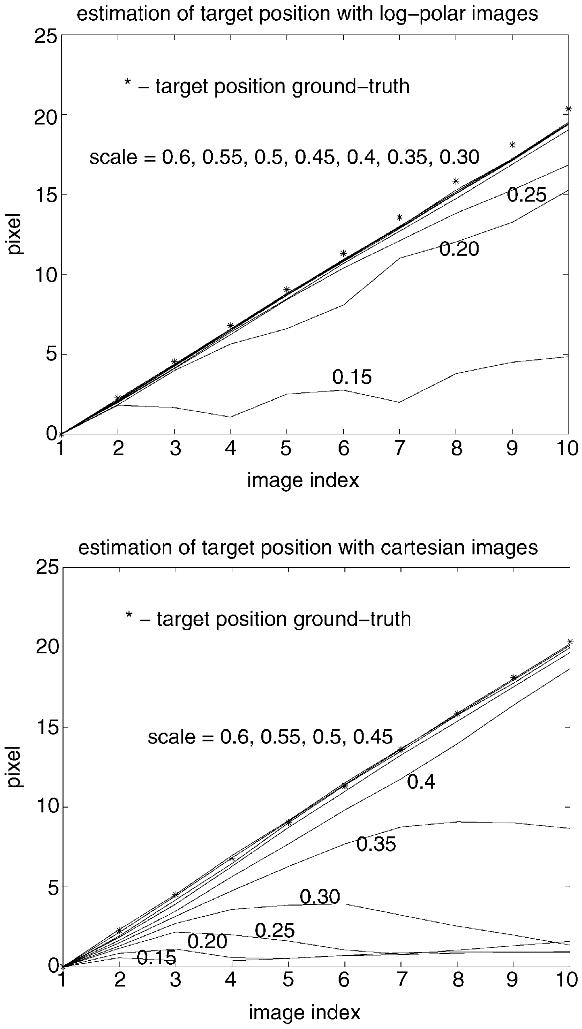


Fig. 8. Comparison between log-polar (top) and Cartesian (bottom) versions of the open-loop experiment. The true and estimated target positions are represented for targets of several dimensions.

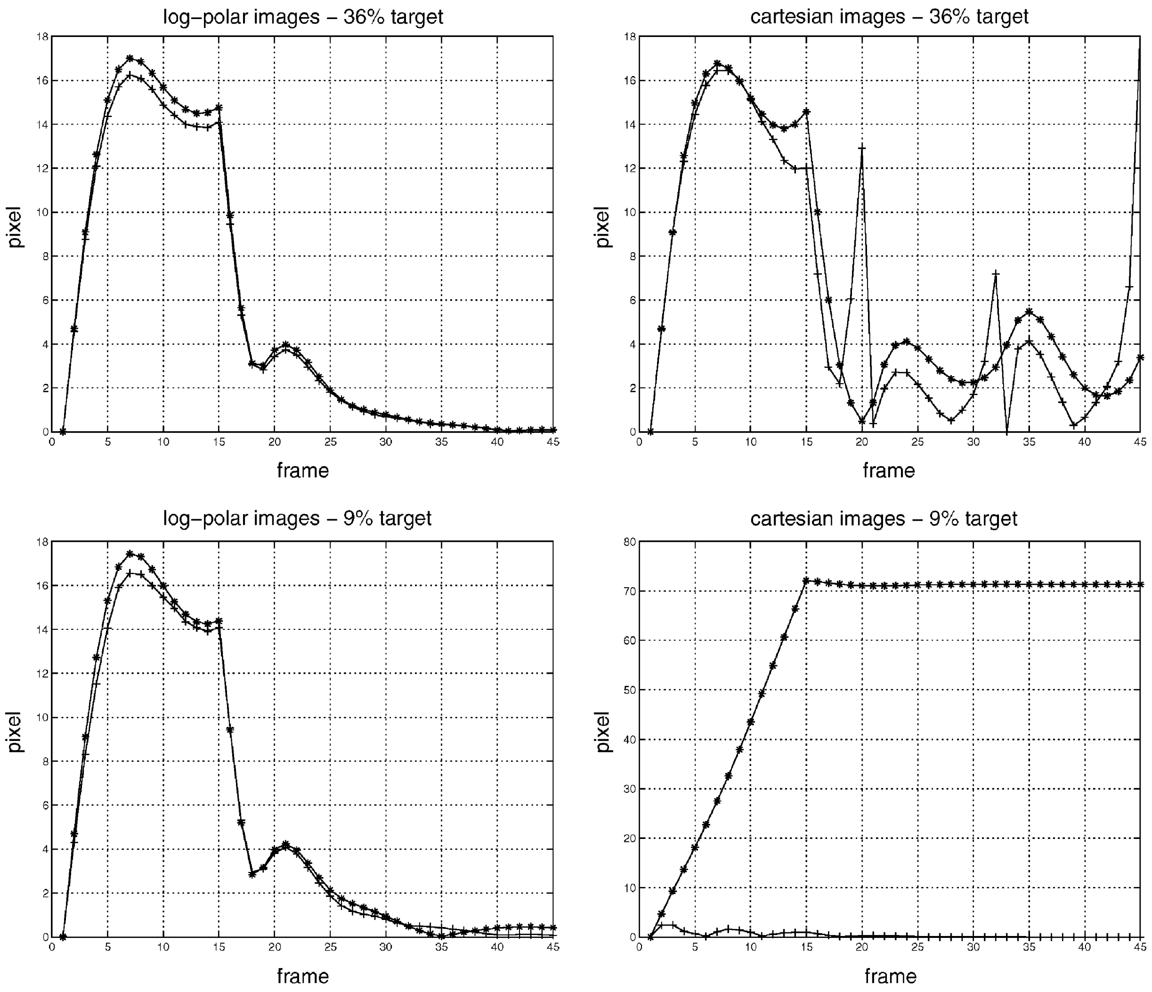


Fig. 9. Estimated (+) versus true (∗) positions of the target. Comparison between log-polar and Cartesian versions of the algorithm with 36% and 9% size objects.

5.2. Advantage of foveated retina

The camera system used in this work has a simple pan/tilt configuration as shown in Fig. 7. The active-tracking goal is to control the pan and tilt angles (θ_p, θ_t) according to the position of the template obtained by the motion estimation algorithm. The purpose is to make the optical axis intersect the center of the target template. When that happens, the image error (x, y) is zero. Otherwise, it is related to the angular position of the target relative to the camera optical axis. Although the real kinematic relations between image error and angular error are non-linear, we will control the pan and tilt angular velocities of the camera with a linear proportional controller on the image error:

$$\dot{\theta}_p = -k_p x, \qquad \dot{\theta}_t = -k_t y.$$

To evaluate the performance of the algorithms with ground truth data we developed a simulator for the system. We assume a simple first order dynamic model for the velocity of the camera joints with a time constant of 200 ms and the sampling frequency is 10 Hz (100 ms), which define a relatively slow dynamics. Therefore, the control is not “one step” but instead has a lag that depends on target velocity and the camera model parameters.

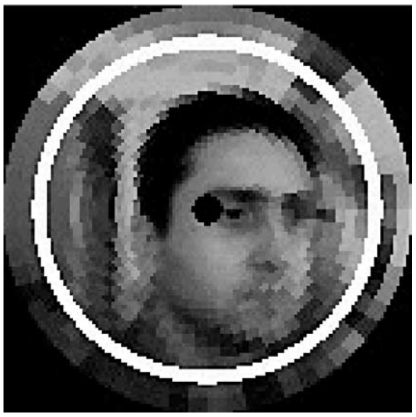


Fig. 10. Reference image for the active tracking experiment. The white circle delimits the region used in the computations. Notice that the selected area contains part of the background.

We use two planar surfaces to simulate the environment: one is the background located 10 m away from the camera and the other is the target at 0.5 m. We tested different scales for the target, from 36 to 2.25% of the full image area (see Fig. 7).

5.2.1. Open-loop test

In this simulated experiment the camera does not move. We compare the use of log-polar and Cartesian images with objects of different sizes. The actual dimension of the object *is not known* a priori, therefore the system selects an initial template that occupies the full image except a small border region in the periphery on the view field. The target translates linearly in 3D space. At each time instance the algorithm estimates target position, which is used as initial guess to the next time step. In Fig. 8 we present plots of the estimated template position for the log-polar and Cartesian versions of the algorithm. From these plots we can observe that the performance of both versions is good for large objects but degrades when target size diminishes. Notwithstanding, the log-polar version copes with smaller objects than the Cartesian version.

5.2.2. Closed-loop test

This is also a simulated experiment and illustrate the integration of motion estimation and active camera control. Simulated camera pan and tilt angles are controlled to keep the observation direction on the center of the target. The target moves with constant velocity during the first 15 time steps and then stops. In this case the displacements can be larger than in the previous experiment because the target is actively kept inside the field of view. Results are shown in Fig. 9. Notice in the plots that a 9% size object is not tracked by the Cartesian algorithm. Even for 36% size, Cartesian tracking is not very stable and sometimes looses

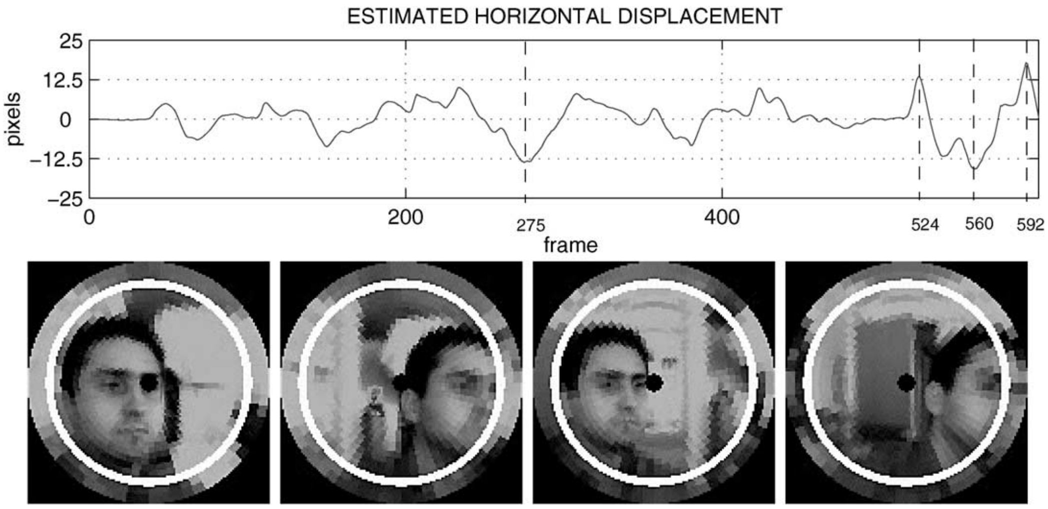


Fig. 11. Estimation of horizontal translation along the active tracking experiment. The frames shown in the bottom correspond to the notable points signaled in the plot.

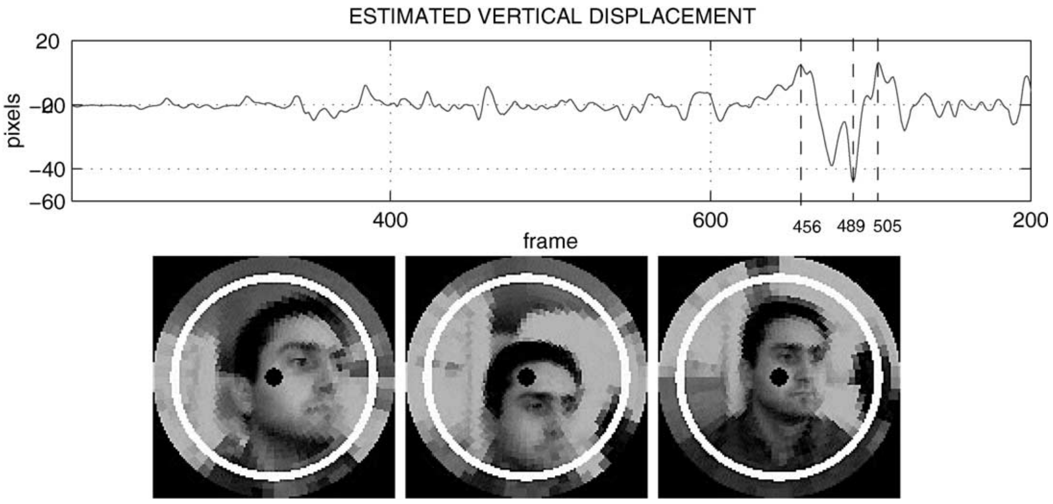


Fig. 12. Estimation of vertical translation along the active tracking experiment. The frames shown in the bottom correspond to the notable points signaled in the plot.

track of target motion. The log-polar algorithm performs very well in both cases, presenting a tracking error less than two pixels in the image plane.

5.3. *Active tracking of real objects*

In this experiment we use a real pan/tilt setup and illustrate qualitatively the performance of the system tracking a face (see Fig. 10). The face moves in front of the system in a natural fashion, performing translations and rotations, both in depth and fronto-parallel to the system. The 2D motion model considered in this case consists in translation and rotation (3 degrees of freedom). We show plots with the estimation of the motion parameters along the sequence. For each plot we signal some special frames, also shown, for which

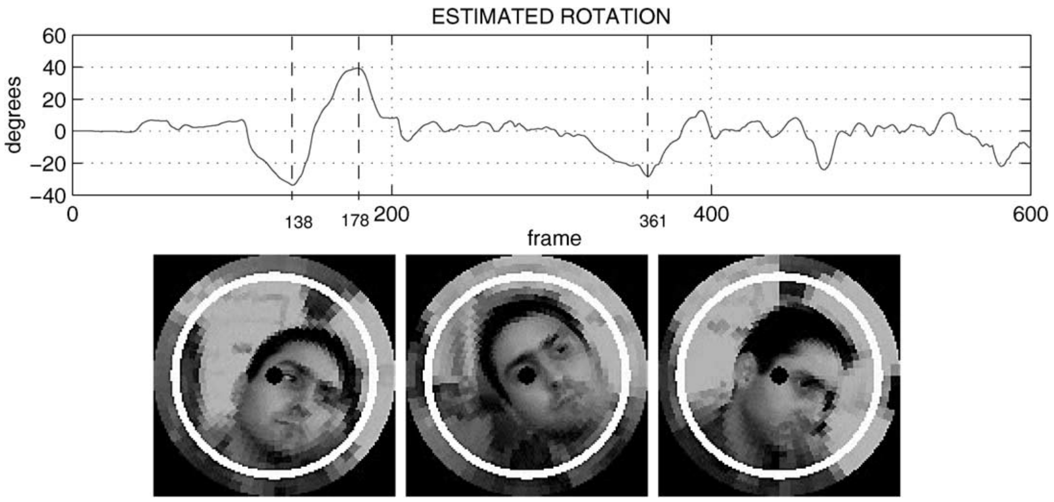


Fig. 13. Estimation of rotation along the active tracking experiment. The frames shown in the bottom correspond to the notable points signaled in the plot.

the motion amplitude is large. Fig. 11 corresponds to the estimation of horizontal translation. Notable points are signaled at frames 275, 524, 560, 592, where target velocity attain high values in the horizontal direction. Figs. 12 and 13 show results for vertical translation and rotation. A qualitative analysis shows good robustness to non-modeled deformations such as pose changes, scale changes and background motion.

6. Conclusion

This paper presented a foveated system capable of tracking of objects in real time. The method is based on parametric motion estimation of a given initial template, using geometric deformation models with redundant parameterizations.

Our results show that motion estimation presents good precision. Although, for active tracking purposes, precision may not be a strong requirement, the method has the advantage of not accumulating errors along time, in opposition to usual optic flow algorithms. Also, precise registration can be used to obtain precise pose or ego-motion measurements, useful in auto-localization applications. The use of log-polar images reduces the computation time and memory storage needs. Additionally, the log-polar geometry implements an implicit focus of attention in the image center. Although not evident from the results shown in this paper, the log-polar geometry reduces the effect of image artifacts not in the center of the image.

The main conclusions to retain from this work are the following:

- A very fast parametric motion estimation algorithm is achieved by using appropriate motion decomposition rules and time coordinates.
- A redundant parameterization of image deformation increases the convergence range and allows easy customization of the algorithm.
- A foveated image geometry attenuates background motion effects.
- The estimation of more constrained motion models in the initial iterations of the algorithm provide additional robustness.
- Active tracking can be achieved with simple motion models and present good tolerance to non-modeled deformations.

Experimental results, obtained in simulated and real setups, support these conclusions. Further work will focus on the integration of other visual cues for target detection and validation (color, edge orientation, etc.).

References

[1] J. Bergen, P. Anandan, K. Hanna, R. Hingorani, Hierarchical model-based motion estimation, in: Proceedings of ECCV, Santa Margherita Ligure, Italy, May 1992, pp. 237–252.
[2] A. Bernardino, J. Santos-Victor, Binocular visual tracking: Integration of perception and control, IEEE Transactions on Robotics and Automation 15 (6) (1999) 1080–1094.
[3] C. Wampler II, Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods, IEEE Transactions Systems on Man and Cybernetics 16 (1) (1986) 93–101.
[4] C. Capurro, F. Panerai, G. Sandini, Dynamic vergence using log-polar images, International Journal of Computer Vision 24 (1) (1997) 79–94.
[5] A. Gelb, Applied Optimal Estimation, MIT Press, Cambridge, MA, 1974.
[6] M. Gleicher, Projective registration with difference decomposition, in: Proceedings of CVPR'97, San Juan, Puerto Rico, June 1997, pp. 331–337.
[7] N. Gracias, J. Santos-Victor, Trajectory reconstruction using mosaic registration, in: Proceedings of SIRS'99, Coimbra, Portugal, July 1999.
[8] G. Hager, P. Belhumeur, Efficient region tracking with parametric models of geometry and illumination, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (10) (1998) 1025–1039.
[9] B. Horn, Robot Vision, MIT Press, Cambridge, MA/McGraw Hill, New York, 1986.
[10] C. Morimoto, R. Chellappa, Fast electronic digital image stabilization, in: Proceedings of ICPR, Vienna, Austria, August 1996.
[11] G. Salgian, D. Ballard, Visual routines for vehicle control, in: D. Kriegman, G. Hager, S. Morse (Eds.), The Confluence of Vision and Control, Springer, Berlin, 1998.
[12] J. Santos-Victor, G. Sandini, Visual behaviors for docking, Computer Vision and Image Understanding 67 (3) (1997) 223–238.
[13] E. Schwartz, Spatial mapping in the primate sensory projection: Analytic structure and relevance to perception, Biological Cybernetics 25 (1977) 181–194.
[14] C. Weiman, Log-polar vision for mobile robot navigation, in: Proceedings of the Electronic Imaging Conference, Boston, MA, November 1990, pp. 382–385.
[15] H. Zhang, J. Ostrowski, Visual servoing with dynamics: Control of an unmanned blimp, in: Proceedings of ICRA'99, Detroit, MI, May 1999.

Alexandre Bernardino was born in Lisboa, Portugal, in 1971. He received the M.Sc. degree in Electrical and Computer Engineering in 1997 from the Instituto Superior Técnico (Lisboa), and is working towards the Ph.D. degree in Computer Vision. He is also a teaching assistant at the Instituto Superior Técnico and a research assistant at the Computer Vision Laboratory at ISR-Lisboa. He has participated in several national and international research projects and published many articles in international journals and conferences. His main research interests focus on robot and cognitive vision, machine learning and real-time control.

José Santos-Victor received the Licenciatura degree in Electrical and Computer Engineering in 1988 from Instituto Superior Técnico (IST—Lisbon, Portugal) and the M.Sc. and Ph.D. in 1991 and 1995 all from IST, in the areas of Active Computer Vision and Robotics. He is an Assistant professor at the Department of Electrical and Computer Engineering of IST and a researcher of the Institute of Systems and Robotics (ISR), where he coordinates the Computer and Robot Vision Lab—VisLab. He is the scientific responsible for the participation of IST in various European and National research projects in the areas of Computer Vision and Robotics. His research interests are in the areas of Computer and Robot Vision, particulary in the relationship between visual perception and the control of action, in (land, air and underwater) mobile robots.

Giulio Sandini is full professor of “natural and artificial intelligent systems” at the Engineering School of the University of Genova in Italy. The leading theme of his research activity has been *visual perception and sensorimotor coordination* from a biological and an artificial perspective. After graduating in Electronic Engineering he spent many years in a neurophysiology laboratory studying human visual perception and the properties of visual cortical neurons in animals. He was also involved, at the Department of Neurology of the Harvard University, in clinical studies and in the developments of new computerized techniques for the analysis of brain electrical activity. In the last years the main topic of his research activity has been the study of human sensorimotor and cognitive development through the use of artificial systems. Is the funding director of the LIRA-Lab (www.lira.dist.unige.it).