ELSEVIER

Image and Vision Computing 21 (2003) 145–160

image AND vision COMPUTING

www.elsevier.com/locate/imavis

Dealing with 2D translation estimation in log-polar imagery

V. Javier Traver*, Filiberto Pla

Computer Vision Group, Departament de Llenguatges i Sistemes Informàtics, Campus Riu Sec, Universitat Jaume I, E12080 Castelló, Spain

Received 26 August 2002; accepted 8 October 2002

Abstract

Log-polar mapping has been proposed as a very appropriate space-variant imaging model in active vision applications. This biologically inspired model has several advantages, and facilitates some visual tasks. For example, it provides an efficient data reduction, and simplifies rotational and scaling image transformations. However, simple translations become a difficult transform due to the log-polar geometry. There is no doubt about the importance of translation estimation in active visual tracking. Therefore, in this work, the problem of translation estimation in log-polar images is tackled. Two different approaches are presented, and their performances are evaluated and compared. One approach uses a gradient descent for minimizing a dissimilarity measure, while the other converts the 2D problem into two simpler 1D problems, by using projections. As the experimental results reveal, this second approach, besides being more efficient, can deal with larger translations than the gradient-based search can.

Keywords: Log-polar mapping; Active vision; Motion estimation; Steepest descent; Projections

1. Introduction

Active vision is an interesting paradigm, which turns problems that are ill-posed in passive vision into well-posed problems [1]. This is possible because active visual systems interact purposefully with the world. On the other hand, much research suggests that foveate sensors are intimately linked with the active vision concept [2,3]. The joint use of these two powerful ideas results in effective and inexpensive active agents, which may mimic the structure, behavior, and performance of the human visual system.

There exist studies showing that the information received by the visual cortex is the result of a log-polar conformal mapping of the retinal stimulus [4–6]. These neurophysiological findings have probably promoted investigations on anthropomorphic sensors in artificial, computer-based vision. Indeed, the log-polar transformation has been adopted in numerous applications and is the subject of current research interest [7–10]. Log-polar images offer a good trade-off among large visual field, reasonably high resolution, and significant data reduction. This is a valuable feature for real-time performance in active vision algorithms. Another well-known advantage brought by the log-polar mapping is that origin-centered rotations and scalings become simple shifts in the cortical image. This is a useful property for scale- and rotation-invariant pattern recognition.

However, despite the many interesting characteristics of the log-polar transform, it has also an important drawback: linear features and translational transformations are distorted by the log-polar mapping, thus adversely complicating the analysis of these patterns. A completely different transform, the *reciprocal-wedge transform*, was proposed to preserve linearity of lines and translations of the original image [11]. Adopting this transform, however, would imply the loss of the interesting properties of the log-polar mapping. Furthermore, the reciprocal-wedge transform seems to have had a much smaller impact in the research community than the log-polar transformation has.

Therefore, to keep the advantages of the log-polar transformation and still be able to estimate linear features, the problem of translation estimation in log-polar images must be addressed. In addition to the general benefits of foveal vision (e.g. drastic reduction in the amount of information to be kept and to be processed), there is still another important advantage of using log-polar images for tracking purposes: an object occupying the central part of the visual field becomes dominant over the coarsely sampled background elements in the image periphery [12].

* Corresponding author. Tel.: +34-964-728-327; fax: +34-964-728-435.
E-mail addresses: vtraver@uji.es (V.J. Traver), pla@uji.es (F. Pla).

0262-8856/03/$ - see front matter © 2003 Elsevier Science B.V. All rights reserved.
PII: S0262-8856(02)00150-6

This makes unnecessary an explicit target segmentation that would otherwise be required in uniformly sampled images.

1.1. Related work

Some work on motion estimation and target tracking using log-polar images has been performed over the last decade. The first efforts, in the late 1980s, are probably due to Weiman and Juday, who exploited the features of the log-polar geometry for centering a target [13]. However, they assumed perfectly segmented objects. Optical flow-based methods can be found in works by Tunley and Young [14], Daniilidis and Krüger [15], or Dias et al. [16]. Even though complex motion models are used, optical flow has a general reputation of being noisy and costly, and special care has to be paid in its computation in space-variant images [17]. Panerai et al. propose a translation estimation algorithm based on the analysis of what translations become in the log-polar plane [18]. Even though no experimental results are given, Traver and Pla [19] have recently re-explored this method, reporting moderate good results. The work by Ahrns and Neumann [20] is close to ours in that they rely on image grey levels directly. Unfortunately, besides the log-polar images themselves, they make use of the cartesian images in their gradient-descent technique for the control of the pan angle of a vision head. Recently, Bernardino et al. presented a system for tracking a surface undergoing planar deformations [21]. Their algorithm is based on registering a given initial reference template, and precomputing a set of sample templates by taking advantage of the knowledge of the kind and range of expected image deformations, as well as the target. Intensive off-line computation is performed for an a priori known reference template. Applying this method in tracking tasks where no model of the target is available would imply the precomputations to be done on-line, which may make the strategy infeasible for real-time performance. Okajima et al. [22] use complex wavelets to estimate the motion parameters of an object rigidly translating in a narrow region of the 3D scene. Unfortunately, the use of wavelets imposes a significant computational burden.

Some researchers (Oshiro et al. [23], Bernardino and Santos-Victor [24]) work with binocular log-polar images. In these works, the target is segmented from the background by means of a zero disparity filter, and the centroid of the resulting target region is used as the tracking error signal. In contrast, we focus our work on monocular log-polar images, make no use of optical flow, assume no model of the target, and restrict to a translational motion model. Additionally, the methods we present here are conceptually simple, and one of them has the important advantage of being computationally very efficient, and able to estimate considerably large translations—an ability not always present in the mentioned literature. This makes the approach very suitable for real-time applications. Despite the simplicity of the motion model considered, translations are the dominant motion component in many interesting situations (traffic scenarios, video-conference, etc.), which render our strategies useful for real-world problems.

The organization of the rest of this paper is as follows. The log-polar transformation, the notation used subsequently, and the effect of translations in the cortical plane are discussed in Section 2. Next, in Sections 3 and 4, the two proposed techniques of translation estimation in log-polar images are described. Very briefly, one of them is based on a gradient descent search, while the other reduces the dimensionality of the problem by using projections. Then, their experimental performance is shown in Section 5, and a comparison of both techniques is presented in Section 6. Finally, overall conclusions are given in Section 7.

2. Log-polar mapping

2.1. Definitions and notation

A *log-polar mapping* commonly used in literature (e.g. Ref. [17]) defines the log-polar coordinates

$$(\xi, \eta) \triangleq \left(\log_a\left(\frac{\rho}{\rho_0}\right), q\cdot\theta\right), \tag{1}$$

with (ρ, θ) being the polar coordinates defined from the cartesian coordinates (x, y) as usual, i.e. $(\rho, \theta) \triangleq (\sqrt{x^2+y^2}, \arctan y/x)$. Because of the discretization, the continuous coordinates (ξ, η) become the discrete ones $(u, v) = (\lfloor\xi\rfloor, \lfloor\theta\rfloor)$, $0 \le u < R$, $0 \le v < S$, with R and S being the number of rings and sectors of the discrete log-polar image, and $q = S/2\pi$ sectors/radian being the angular resolution. The notation $\lfloor z\rfloor$ denotes the common floor operation, i.e. the largest integral value not greater than z. Having chosen R, ρ_0 (the radius of the innermost ring), and $\rho_{\max}$ (the radius of the visual field), the transformation parameter a is computed as $a = \exp(\ln(\rho_{\max}/\rho_0)/R)$. Other log-polar models and further details on their computation can be found in Refs. [25–27].

Given the parameters of the cartesian and log-polar geometries, the log-polar transformation builds the map $\mathscr{L}$, where $\mathscr{L}(i,j)$ is the set of log-polar pixels (u, v) intersecting the cartesian pixel (i, j). If the original cartesian image is sized $M \times N$, $\rho_{\max}$ is defined as $\rho_{\max} = \frac{1}{2}\min(M, N)$, and the log-polar transform is centered at the foveation point $(x_c, y_c) = (M/2, N/2)$.

An example of a log-polar transformation is shown in Fig. 1, from which several observations can be made. First of all, it can be appreciated the much smaller size of the cortical image (Fig. 1(c)) compared to the original uniformly sampled image (Fig. 1(b)), which illustrates the data reduction property. Second, the small arrows radially disposed in the cartesian image become magnified and parallel one to each other (Fig. 1(c)), which demonstrates how rotations become translations along the η log-polar axis. Third, note in the retinal visualization of the cortical image (Fig. 1(d)) how edges near the image center are much

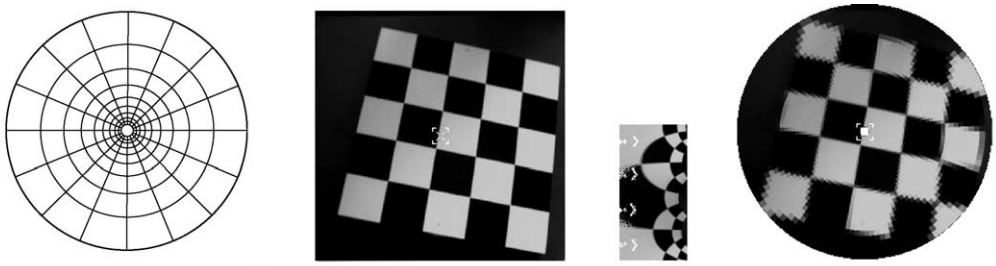


Fig. 1. Log-polar mapping: (a) grid layout example (10 × 16), (b) original cartesian image (256 × 256), (c) cortical image (64 × 128), (d) retinal image (256 × 256) obtained by the inverse mapping from (c).

sharper than edges at the periphery, because of the space-variant resolution.

2.2. Effect of translations in the log-polar plane

A translation in the x and y coordinate axis, (x_0, y_0), can be expressed in the cartesian plane as simply as the linear equation:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}. \tag{2}$$

In the complex logarithmic plane, however, this simplicity disappears, and the motion model would read as follows:

$$\begin{pmatrix} \xi' \\ \eta' \end{pmatrix} = \begin{pmatrix} \log_a\left(\dfrac{\sqrt{(\rho_0 \cdot a^{\xi} \cdot \cos(\eta/q) + x_0)^2 + (\rho_0 \cdot a^{\xi} \cdot \sin(\eta/q) + y_0)^2}}{\rho_0}\right) \\ q \cdot \arctan\left(\dfrac{\rho_0 \cdot a^{\xi} \cdot \sin(\eta/q) + y_0}{\rho_0 \cdot a^{\xi} \cdot \cos(\eta/q) + x_0}\right) \end{pmatrix}. \tag{3}$$

Eq. (3) is obtained by expressing the cartesian coordinates $(x{=}x(\xi,\eta),\ y{=}y(\xi,\eta))$ corresponding to the log-polar coordinates (ξ,η), then adding the motion displacement (x_0,y_0), and finally converting the result $(x{+}x_0,y{+}y_0)$ back to log-polar coordinates.

It is illustrating the visualization of how log-polar images are deformed under translational motion. The same sample image in Fig. 1 undergoes a uniform translation $(x_0, y_0) = (1, -1)$ at each time step in a sequence, and some frames are shown in Fig. 2. A comparison of the translational vectors in the cartesian and log-polar planes is given in Fig. 3. For the sake of clarity, sparse grids have been drawn, the vectors near the center have been avoided, and the log-polar grid has been magnified with respect to the size of the cartesian grid.

3. Gradient-descent search (GDS)

3.1. Motivation

Some authors have studied that the correlation measure between two stereo log-polar images for varying values of the vergence angle has a profile with interesting properties, which are not present when the correlation is computed between cartesian images [24,28]. We show that this idea, used for vergence control, can also be exploited in monocular tracking in log-polar space. The choice of a steepest-descent control is justified by the shape of the resulting correlation surfaces.

In Refs. [24,28] the correlation index $\mathscr{C}$ for the vergence angle ψ in a certain range, gives rise to a one-dimensional (1D) function, $\mathscr{C}(\psi)$. Our approach extends this idea to a 2D function (the correlation surface), which is dependent on the translational components in x and y direction, x_0 and y_0, respectively, $\mathscr{C}(x_0, y_0)$. We can compute the value of a correlation measure $\mathscr{C}$ between one image and versions of this one shifted by (x_0, y_0). Fig. 4 shows an example of the resulting surface $\mathscr{C}(x_0, y_0)$ computed in cartesian (Fig. 4(a)) and log-polar (Fig. 4(b)) domains for the same images. As can be seen in the figure, while both surfaces exhibit a distinguishable minimum, in the case of log-polar domain the location of such a minimum is very close to the actual translational motion undergone in image plane ($x_0 = 5$, $y_0 = 5$), while in the case of cartesian images this minimum is near $(0, 0)$, i.e. far from the right motion parameters. These surfaces have been obtained before and after a foveated target, not occupying the whole image, has performed a small displacement. This interesting feature can be referred to as *implicit focus-of-attention* [24].

Algorithm 1. Gradient-descent-based translation estimation in log-polar images.

Input: Two log-polar images, I_1 and I_2
Output: The estimated translation vector $(\hat{x}_0, \hat{y}_0)$
1: $(x_0^0, y_0^0) \leftarrow (0, 0)$ {*initial guess*}
2: $k \leftarrow 0$ {*iteration number*}

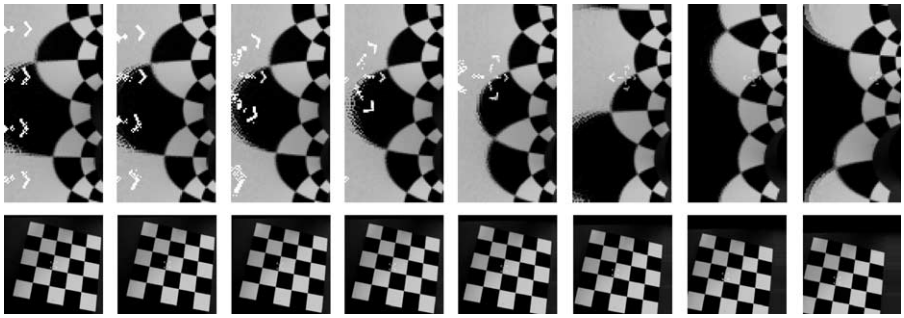


Fig. 2. Illustration of the cortical image deformation under cartesian translation. Log-polar images are in first row, cartesian images are in the second row (shown much more smaller than they are). Columns 2–8 are versions of the original image (in first column) shifted by ($x_0 = s, y_0 = -s$), with s equal to 1, 3, 5, 10, 20, 30, and 40, respectively.

3: $\delta \leftarrow 1$ {*step length*}
4: **while** $(\delta > \delta_{\min}) \wedge (k < k_{\max})$ **do**
5: $k \leftarrow k + 1$
6: $(x_0^k, y_0^k) \leftarrow (x_0^{k-1}, y_0^{k-1}) - g(\nabla\mathscr{C})$ {*estimation update rule*}
7: **if** minimum surpassed **then**
8: $\delta \leftarrow \delta/2$
9: **end if**
10: **end while**
11: $(\hat{x}_0, \hat{y}_0) \leftarrow (x_0^k, y_0^k)$

3.2. The algorithm

As we have shown, the minimum of the correlation surface computed on log-polar images occurs (approximately) at the correct displacement of the target. To estimate the translation parameters (x_0, y_0), our approach consists of finding the location of the minimum of the correlation measure $\mathscr{C}(x_0, y_0)$. The shape of the correlation surfaces seen in Section 3.1 suggests that a gradient-based search could be an adequate search algorithm. The estimation at iteration k, (x_0^k, y_0^k), is updated from the estimation at the previous iteration, (x_0^{k-1}, y_0^{k-1}), by using the gradient, $\nabla\mathscr{C}$, of the correlation measure, as the most promising direction to move, i.e.:

$$(x_0^k, y_0^k) = (x_0^{k-1}, y_0^{k-1}) - g(\nabla\mathscr{C}). \tag{4}$$

A common definition for $g(\cdot)$ is $g(\nabla\mathscr{C}) = \delta \cdot \nabla\mathscr{C}/z.sfnc; \nabla\mathscr{C}|$, i.e. consider the unit vector in the direction of the gradient and move a certain amount δ in the opposite direction. The issue of choosing the value for δ usually involves a trade-off. Then, an adaptive, rather than a fixed step, is called for. One possibility consists of moving 'large' steps when far from the minimum, and 'small' steps closer to it. This is the idea we use: initially $\delta = 1$ and whenever the minimum is surpassed the value of δ is halved. The search may be stopped using some criteria such as that δ is smaller than a given threshold $\delta_{\min}$, or that the search has reached a maximum number of iterations $k_{\max}$. Some other possible convergence criteria are discussed in Ref. [29], p. 309. The steps of the whole process are presented in Algorithm 1. This is a basic formulation over which some variations are possible [29,30].

3.3. Computing the correlation gradient

To evaluate Eq. (4) we need a way to compute the gradient $\nabla\mathscr{C}$. Let I_1 and I_2 be two log-polar images.

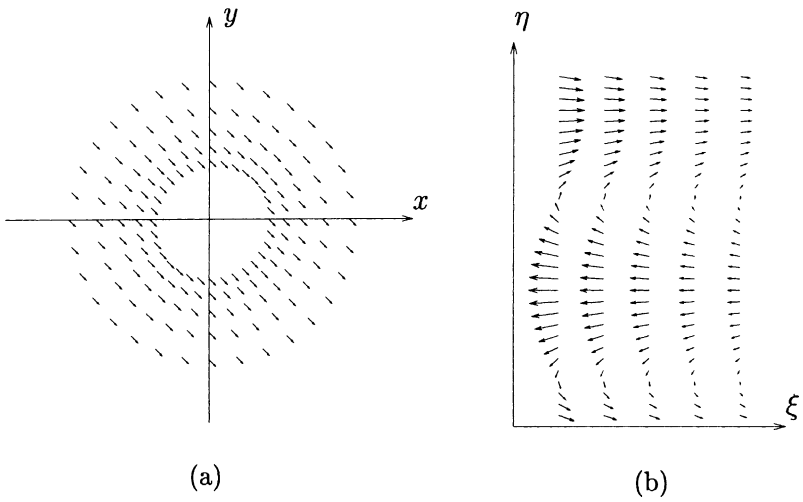


Fig. 3. Translation vectors in the (a) retinal and (b) cortical planes.

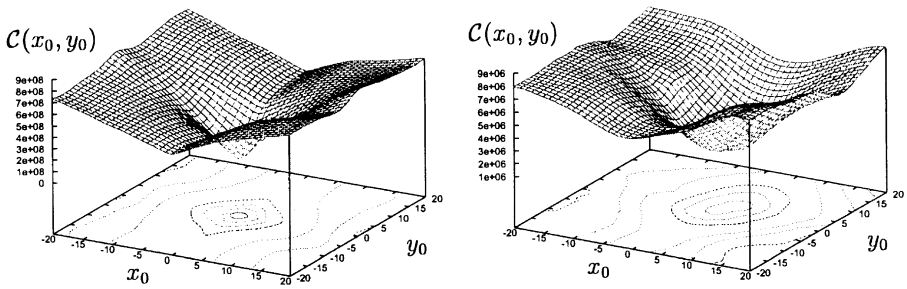


Fig. 4. Example of correlation surface in the (a) cartesian and (b) log-polar spaces.

In the case of the well-known SSD (sum of squared differences) correlation measure [31,34],

$$\mathscr{C}(x_0, y_0) = \sum_{(\xi,\eta)\in\mathscr{D}} (I_2(\xi', \eta') - I_1(\xi, \eta))^2,$$

the gradient $\nabla\mathscr{C} = (\mathscr{C}_{x_0}, \mathscr{C}_{y_0})$ becomes:

$$\begin{pmatrix} \mathscr{C}_{x_0} \\ \mathscr{C}_{y_0} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial\mathscr{C}}{\partial x_0} \\ \dfrac{\partial\mathscr{C}}{\partial y_0} \end{pmatrix} = 2\begin{pmatrix} \displaystyle\sum_{(\xi,\eta)\in\mathscr{D}} \{(I_2(\xi', \eta') - I_1(\xi, \eta))\cdot I_{2_{x_0}}(\xi', \eta')\} \\ \displaystyle\sum_{(\xi,\eta)\in\mathscr{D}} \{(I_2(\xi', \eta') - I_1(\xi, \eta))\cdot I_{2_{x_0}}(\xi', \eta')\} \end{pmatrix}, \quad (5)$$

with $\mathscr{D}$ being a certain set of image pixels (usually, the entire image), and where $I_{2_{x_0}} = I_{2_{x_0}}(\xi', \eta')$ and $I_{2_{y_0}} = I_{2_{y_0}}(\xi', \eta')$ are:

$$\begin{pmatrix} I_{2_{x_0}} \\ I_{2_{y_0}} \end{pmatrix} = \begin{pmatrix} \xi'_{x_0} & \eta'_{x_0} \\ \xi'_{y_0} & \eta'_{y_0} \end{pmatrix} \cdot \begin{pmatrix} I_{2_{\xi'}} \\ I_{2_{\eta'}} \end{pmatrix}. \quad (6)$$

Note that the common notation for partial derivatives, $f_z = \partial f/\partial z$, is used. On the other hand, (ξ', η') depend in a space-variant way on the translational displacement (x_0, y_0). For the sake of simplicity, we use the following approximation, rather than the exact expression in Eq. (3):

$$\begin{pmatrix} \xi' \\ \eta' \end{pmatrix} \approx \begin{pmatrix} \xi \\ \eta \end{pmatrix} + \begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}. \quad (7)$$

Deriving ξ' and η', as defined in Eq. (7), with respect to x_0 and y_0 yields $\xi'_{x_0} = \xi_x$, $\xi'_{y_0} = \xi_y$, $\eta'_{x_0} = \eta_x$, and $\eta'_{y_0} = \eta_y$, which are then used in Eq. (6).

Finally, by taking the partial derivatives of ξ and η, as defined in Eq. (1), with respect to x and y, we get ξ_x, ξ_y, η_x, and η_y, as follows:

$$\begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix} = \begin{pmatrix} \dfrac{\partial\xi}{\partial x} & \dfrac{\partial\xi}{\partial y} \\ \dfrac{\partial\eta}{\partial x} & \dfrac{\partial\eta}{\partial y} \end{pmatrix} = \frac{1}{\rho}\begin{pmatrix} \dfrac{\cos\theta}{\ln a} & \dfrac{\sin\theta}{\ln a} \\ -\sin\theta & \cos\theta \end{pmatrix}. \quad (8)$$

4. Projections-based approach (PBA)

4.1. Introduction

In computer vision, projections are a well-known technique allowing one to address some problems by reducing their dimensionality. As an example in motion estimation, radial projections in the frequency domain are defined in Ref. [32] using conventional images. Our approach works in the spatial domain of space-variant images.

In Ref. [33], a 1D summation, and a search for global minima in a 1D signal are performed to estimate the focus of expansion. In Ref. [16], summations of the components of the optic flow are defined along the radial and angular directions to find the location of the moving target. Like our approach in this section, both Dias et al. and Daniilidis [16,33] work with log-polar images, and perform operations with 1D signals. Unlike our work, these works do not employ projections with the same meaning used in this paper (i.e. sum of gray-level values), and the problem they try to solve is related, but different, to ours.

4.2. Definitions

A *vertical* projection $\mathscr{V}$ at the discrete vertical coordinate i is defined as

$$\mathscr{V}(i) \triangleq \sum_{j'=j_3}^{j_4} \sum_{(u,v)\in\mathscr{L}(i,j')} I(u, v), \quad i \in [i_1, \ldots, i_2]. \quad (9)$$

Likewise, a *horizontal* projection $\mathscr{H}$ at the discrete horizontal coordinate j is defined as

$$\mathscr{H}(j) \triangleq \sum_{i'=i_3}^{i_4} \sum_{(u,v)\in\mathscr{L}(i',j)} I(u, v), \quad j \in [j_1, \ldots, j_2]. \quad (10)$$

The underlying idea in Eqs. (9) and (10) is as follows. For instance, in the case of the vertical projections, $\mathscr{V}(i)$ is the sum of the gray levels of the image at locations (i, j'), with a fixed i and varying j'. If image I was an uniformly sampled

cartesian image, the expression for $\mathscr{V}(i)$ would have the simpler form:

$$\mathscr{V}(i) = \sum_{j'=j_3}^{j_4} I(i,j'), \quad i \in [i_1, \ldots, i_2],$$

i.e. the sum of gray levels along column i.

However, we are dealing with space-variant images with a polar logarithmic geometry. Therefore, to define $\mathscr{V}$ for a log-polar image, we use the map $\mathscr{L}(i,j)$, introduced in Section 2.1, to access the set of log-polar pixels (u,v) corresponding to the cartesian position (i,j). For horizontal projections, the explanation would be analogous.

Fig. 5 shows the directions along which these projections are computed (the concept of support map is introduced below). Note how vertical and horizontal directions in the cartesian domain are mapped to curves in the log-polar image. Notice that the vertical and horizontal projections, as defined in Eqs. (9) and (10), count the same $I(u,v)$ as many times as cartesian pixels (i,j) intersect the log-polar pixel (u,v). In an alternative definition, each pixel (u,v) could be considered only once. Experiments show that both approaches result in similar results, while the latter being somehow more efficient.

Fig. 6 shows graphically the parameters used in Eqs. (9) and (10), as well as some others that will be referred to later in this article. Although the projections have been defined in a general way, the implementation is somehow more specific. Therefore, the *location* parameters i_1 and i_2 for the vertical projections, and j_1 and j_2 for the horizontal projections, are defined symmetrically from the foveation point (x_c, y_c). The same happens for j_3, j_4, i_3 and i_4, which define the *extension* of the projections. In other words, the parameters m_V, n_V, m_H and n_H are used, as shown in Fig. 6, to define these indices, taking (x_c, y_c) as a reference. A further simplification yields $m_V = m_H = m$ and $n_V = n_H = n$. Finally, everything can be summarized in one parameter by making $m = n = w$, with w being the parameter we will use later on, when explaining the experiments and when analyzing the computational cost.

For each vertical projection $\mathscr{V}(i)$, we define the *support map* $\mathscr{M}_v(i)$ as the set of log-polar pixels (u,v) whose image gray level $I(u,v)$ contributes to $\mathscr{V}(i)$. Analogously, support maps $\mathscr{M}_h(j)$ are defined for each horizontal projection $\mathscr{H}(j)$. Therefore, using these support maps, Eqs. (9) and (10) can be redefined, respectively, as:

$$\mathscr{V}(i) \triangleq \sum_{(u,v)\in\mathscr{M}_v(i)} I(u,v), \quad i \in [i_1, \ldots, i_2], \tag{11}$$

and

$$\mathscr{H}(j) \triangleq \sum_{(u,v)\in\mathscr{M}_h(j)} I(u,v), \quad j \in [j_1, \ldots, j_2]. \tag{12}$$

The black and gray areas in Fig. 5 would be the graphic representation of four of such support maps, for two particular values of i and two particular values of j. These maps can be computed once and for all (i.e. in an off-line stage) to avoid repetitive and costly checks during on-line motion estimation.

4.3. The algorithm

Let I_1 and I_2 be two consecutive log-polar images. We add a subindex k to the projection functions $\mathscr{V}$, $\mathscr{H}$, to indicate that they are defined on image I_k, $k \in \{1,2\}$.

Translations in x and y directions (x_0 and y_0) can be estimated by making use of the vertical and horizontal projections, respectively. The principle behind this is that projections are defined in the image plane in a direction

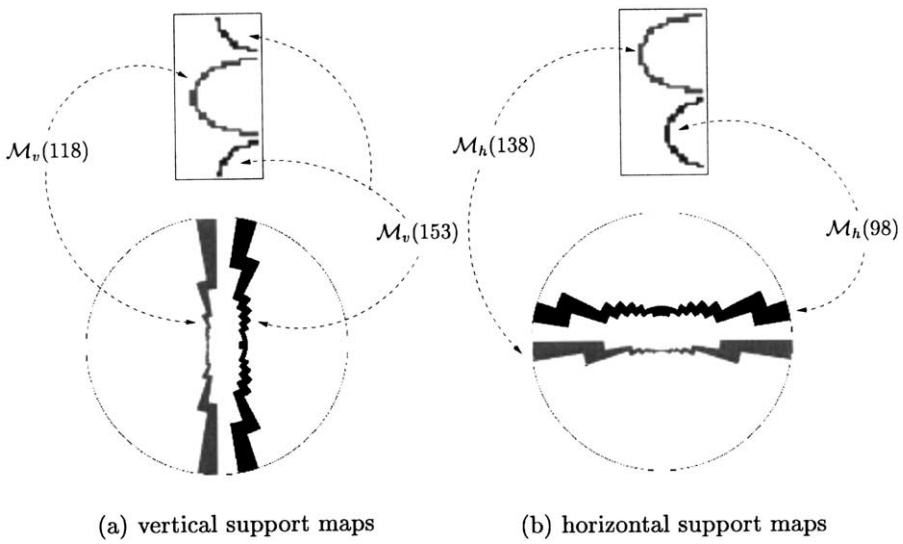
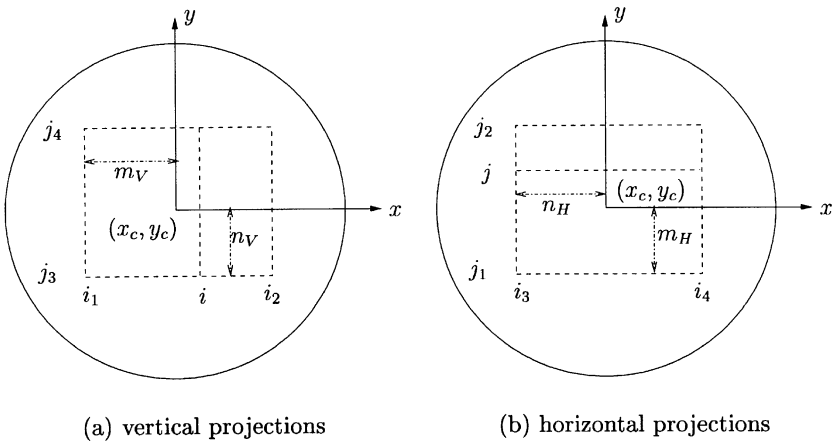


Fig. 5. Examples of projection directions: (a) $\mathscr{M}_v(118)$ and $\mathscr{M}_v(153)$ are two examples of vertical support maps at $i = 118$ and 153, respectively; (b) $\mathscr{M}_h(98)$ and $\mathscr{M}_h(138)$ are two examples of horizontal support maps at $j = 98$ and 138, respectively. These support maps are drawn in the cortical domain (upper part of the figure), and in the retinal plane (lower part of the figure).



(a) vertical projections (b) horizontal projections

Fig. 6. Illustration of the parameters involved in projections definition.

orthogonal to the direction of the motion to be detected. Thus, x_0 and y_0 can be estimated by observing that $\mathscr{V}_1(i) = \mathscr{V}_2(i - x_0)$ and $\mathscr{H}_1(j) = \mathscr{H}_2(j - y_0)$.

Fig. 7 depicts an example of the typical 1D signals we are dealing with, as well as their translations. In relation to this figure, two observations can be made at this point. On the one hand, i and j are not coordinates in the image plane, but indices of the projections, ranging in $[0, 2w - 1]$. On the other hand, vertical (horizontal) projections are normalized, because projections at different horizontal (vertical) locations i (j) are computed over sets of pixels of different cardinality, given the space-variant resolution of the log-polar images.

Algorithm 2. Projections-based translation estimation in log-polar images.

Input: Two log-polar images, I_1 and I_2
Output: The estimated translation vector $(\hat{x}_0, \hat{y}_0)$
1: Compute the vertical projections $\mathscr{V}_1$ and $\mathscr{V}_2$ (using Eq. (11))
2: Compute the horizontal projections $\mathscr{H}_1$ and $\mathscr{H}_2$ (using Eq. (12))
3: Evaluate the 1D displacements among these projections:

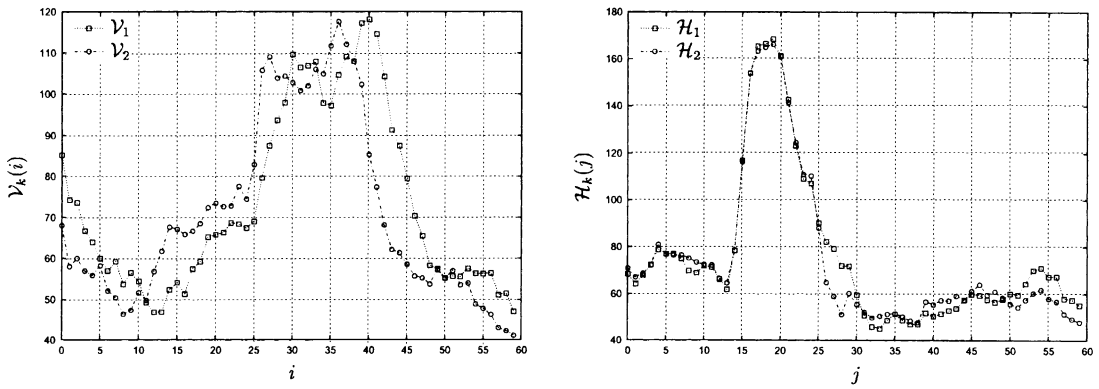
$\hat{x}_0 \leftarrow \text{FIND1DSHIFT}(\mathscr{V}_1, \mathscr{V}_2)$

$\hat{y}_0 \leftarrow \text{FIND1DSHIFT}(\mathscr{H}_1, \mathscr{H}_2)$

4.4. Estimating the 1D shift

Algorithm 2 makes explicit the steps needed to use the projections for motion estimation. However, how to compute the 1D displacement between two 1D signals has been abstracted with the function FIND1DSHIFT(·,·). This is so because there are several possibilities for carrying out this estimation.

If we are to find the shift d between two 1D signals (like those in Fig. 7), S_1 and S_2, i.e. the value d such that $S_1(p) = S_2(p - d)$, one possibility is to evaluate a similarity measure or a correlation function $\mathscr{C}(S_1, S_2; d)$ between both signals at



(a) vertical projections (b) horizontal projections

Fig. 7. Examples of projection signals computed on two consecutive log-polar images where a translation ($x_0 = 4$, $y_0 = 0$) occurred. It can be observed how one vertical projection is a shifted version of the other (due to the horizontal translation), while the horizontal projections (approximately) overlap (due to the absence of vertical translation).

different values d in a discrete set D. In our case, D is the set of integers in a interval $[d_{\min}, d_{\max}]$, which contains the range of expected translations. To speed up the process, D may contain many small displacements, but only a few large displacements. Then, small displacements can be estimated at high accuracy while large displacements can still be detected, but in a coarser manner. This idea is used in Ref. [24] in their *disparity channels* for binocular disparity estimation. A third interesting approach would be to use a 1D gradient-descent approach. This may make the process more efficient, but at some risk of getting stuck at some local minima in case of large displacements. We have also tested this strategy and the results for small-medium translations were good, comparable to those of the global approach described first.

4.5. Adding sub-pixel accuracy

The approaches mentioned in Section 4.4 may yield a translation estimation with only integer accuracy. If sub-pixel accuracy is desired, there are several choices for this. One of them consists of using this integer estimation, say d_0, as the starting point of an iterative hierarchical process. At step k, a sub-pixel precision p_k is defined (e.g. $p_k = p_{k-1}/2$, with $p_0 = 1$). Then, the similarity measure is evaluated at d_k, at $d_k - p_k$, and at $d_k + p_k$, and d_{k+1} is set to the one yielding the optimal confidence. This process is repeated until a given precision p_k has been reached. An alternative consists of computing a weighted average of displacements: several displacements are considered, each weighted by its own confidence measure (see Ref. [34] for details).

5. Experimental results

Algorithms GDS and PBA, introduced in Sections 3 and 4, are experimentally validated in this section, by applying synthetic motion, with available ground-truth data (Section 5.1), and by estimating translations in a real image sequence (Section 5.2).

5.1. Synthetic motion

5.1.1. Setup

To assess the behavior of the algorithms proposed in previous sections, the following steps were carried out. Conventional $M \times N$-sized (cartesian) images are transformed by known motion parameters (x_0, y_0). These images (before and after the transformation) are converted to $R \times S$-sized cortical images by means of the log-polar mapping (Section 2). These log-polar images are input to Algorithms 1 (GDS) and 2 (PBA), which yield an estimate $(\hat{x}_0, \hat{y}_0)$. We set $M = N = 256$, $R = 64$, $S = 128$, and $\rho_0 = 5$.

Given an image, we are interested in studying how translations are estimated for both small and large displacements, as well as in any possible direction. Therefore, we opted to generate translations following a spiral of Archimedes [35] (Fig. 8), which gives us the flexibility to vary the orientation of the translation while increasing the motion magnitude. For the same image, we perform T tests. At each test, numbered t, the motion orientation is $\vartheta = \vartheta(t) = \pi/180\, b_1 t$ radians, and the magnitude is $\varrho = \varrho(t) = b_2 \vartheta(t)$. If we want the spiral to give n_T turns, the constant b_1 can be fixed to $b_1 = 360\, n_T/T$. And if we want the maximum motion magnitude to be $\varrho_{\max}$, then we have that $b_2 = (180/\pi)(\varrho_{\max}/b_1 T)$. Thus, the ground truth motion parameters are: $x_0 = x_0(t) = \varrho(t)\cos(\vartheta(t))$, and $y_0 = y_0(t) = \varrho(t)\sin(\vartheta(t))$. In the experiments below, $T = 100$, $\varrho_{\max} = 30$, and $n_T = 4$.

For GDS, we have $k_{\max} = 40$, $\delta_{\min} = 0.05$; and for PBA, we set $w = 40$, and 1D translations were found using 1D SSD, by testing integer translations in the range $[d_{\min}, d_{\max}]$, with $-d_{\min} = d_{\max} = 30$.

5.1.2. Results

In Fig. 8 the true and estimated spirals corresponding to the 'trajectory' of the motion parameters (it could correspond to the trajectory of a target) are shown. They correspond to the images whose log-polar transformations

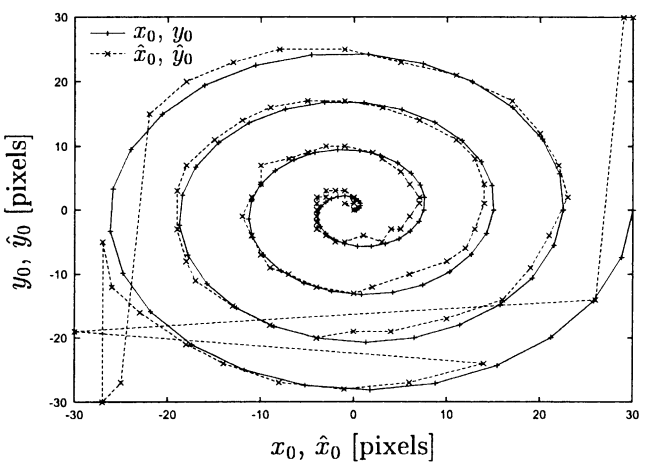


Fig. 8. Cortical (log-polar) images at selected test numbers.

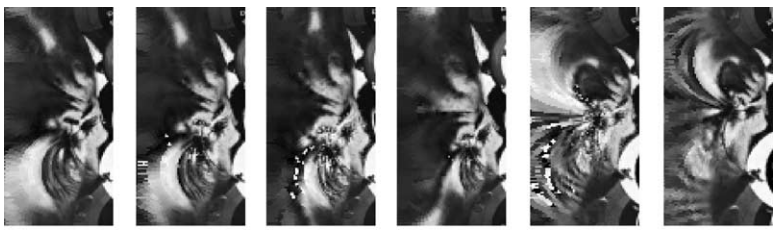
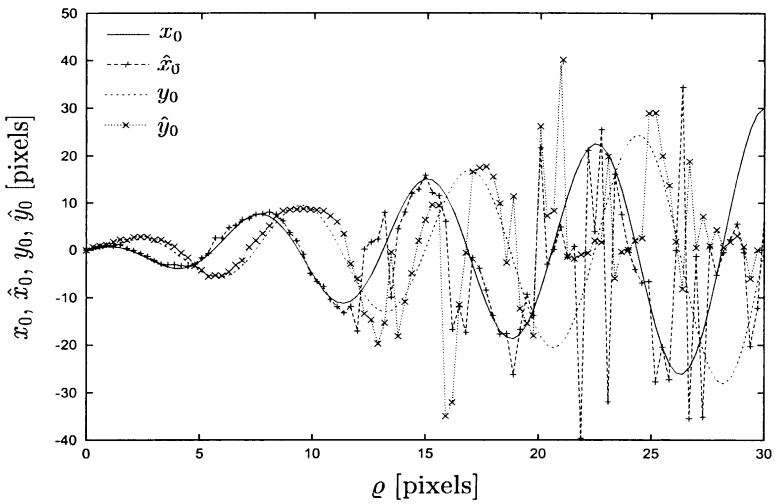
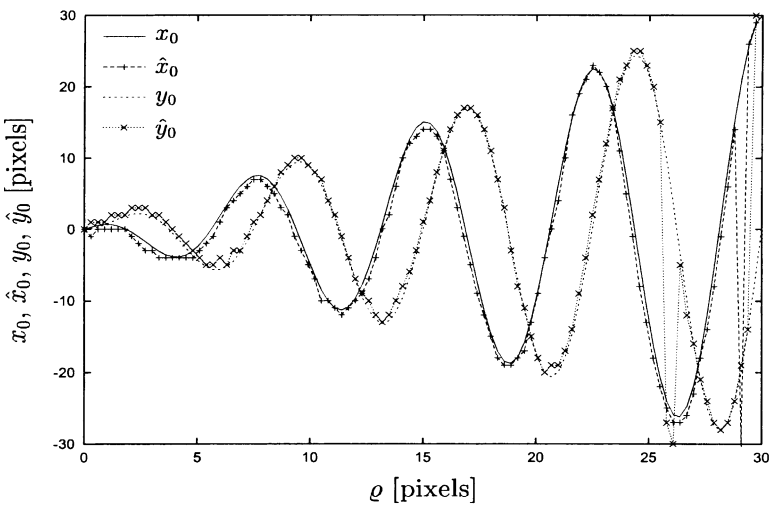


Fig. 9. Trajectory of motion parameters.

at some t are plotted in Fig. 9. The evolution of the true and estimated parameters (x_0, y_0) with the motion magnitude (ϱ), can be seen in Fig. 10. It can be observed that the estimates are generally close to the true translations. However, while this is true in the case of the PBA method for all the range of motion magnitudes tested, it is not in the GDS approach. Certainly, GDS behaves well only for small-medium displacements. As it can be seen in Fig. 10(a), estimates of GDS become unreliable for motions larger than about 10 pixels. In contrast, PBA is robust even for large motions. This poor performance of GDS under large translations can be explained by the fact that any



(a) GDS



(b) PBA

Fig. 10. Evolution of motion parameters with increasing motion magnitude (whole range, $\varrho \in [0, 30]$).

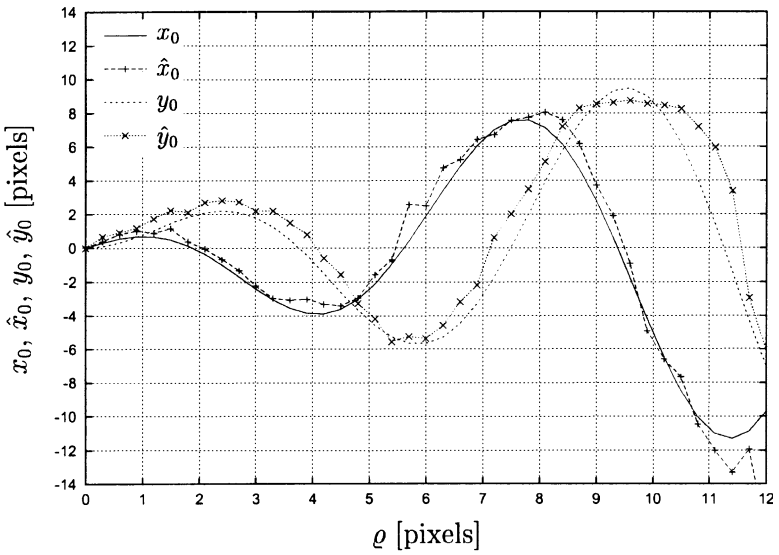
gradient-based search technique is a *local* search by nature, and then only suitable for searching near the goal. Another reason is the approximation made in Eq. (7), which remains reasonably valid for small values of x_0 and y_0.

Both methods can yield spurious erroneous estimates (e.g. see $\hat{y}_0$ around $\varrho \approx 26$, or $\hat{x}_0$ around $\varrho \approx 29$ in Fig. 10(b)). These 'spikes' are not really a problem, because the property of *motion continuity*, in an (active) tracking scenario, makes possible to readily filter them (e.g. by means of a Kalman filter).

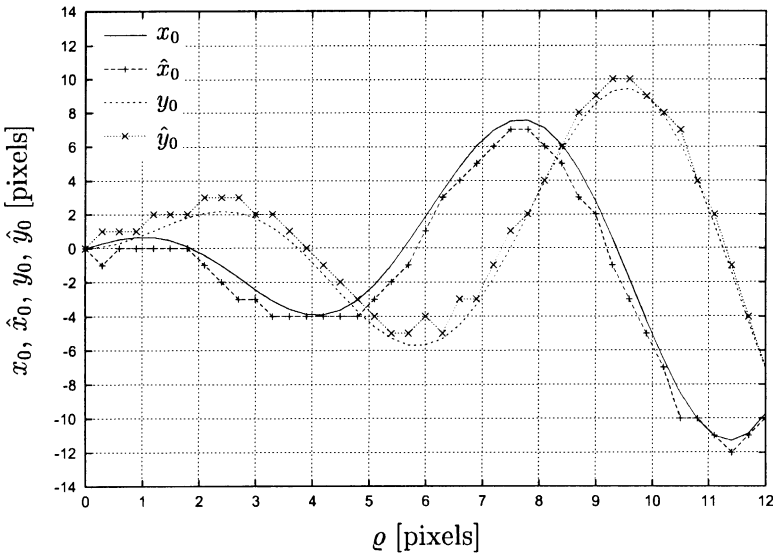
For better viewing the behavior of the algorithms with small-medium displacements, plots in Fig. 10 are enlarged in the range $\varrho \in [0, 12]$, and shown in Fig. 11. Even though only integer accuracy is used in PBA, GDS seems not to exhibit a particularly better estimation accuracy than PBA.

PBA may fail, however, with images having some particular pattern, such as a repetitive texture. This makes projections at different locations have similar values, which, in turn, makes the estimation of 1D shift not very reliable in such cases.

For GDS, a small experiment was conducted to study the influence of the integration domain $\mathscr{D}$, in Eq. (5), on the estimated parameters. Therefore, for the same image pair, GDS was run with fractions $\alpha \in \{0.1, 0.2, \ldots, 0.9, 1.0\}$ of the total number of rings R. Results gracefully deteriorate with smaller integration regions (i.e. with lower α), and



(a) GDS



(b) PBA

Fig. 11. Evolution of motion parameters with increasing motion magnitude (smaller motions, $\varrho \in [0, 12]$).

become particularly bad for $\alpha < 0.3$. This graceful degradation can be explained by the fact that data remaining in the integration region is that corresponding to the highly resolved fovea. For more conclusive results, a more thorough experimental study should be carried out in which the effect of other parameters (such as the size of the log-polar image, $R \times S$, or the value of the innermost radius, ρ_0) is also taken into account.

Fig. 12. (a, b) Some frames of the sequence building01 and their log-polar transform, and frame difference *between* the current image (frame t) *and*: (c) the first image (frame 0); (d) the first image rectified with the translation estimated with GDS; (e) the first image rectified with the translation estimated with PBA. The gray level of the difference images has been inverted.

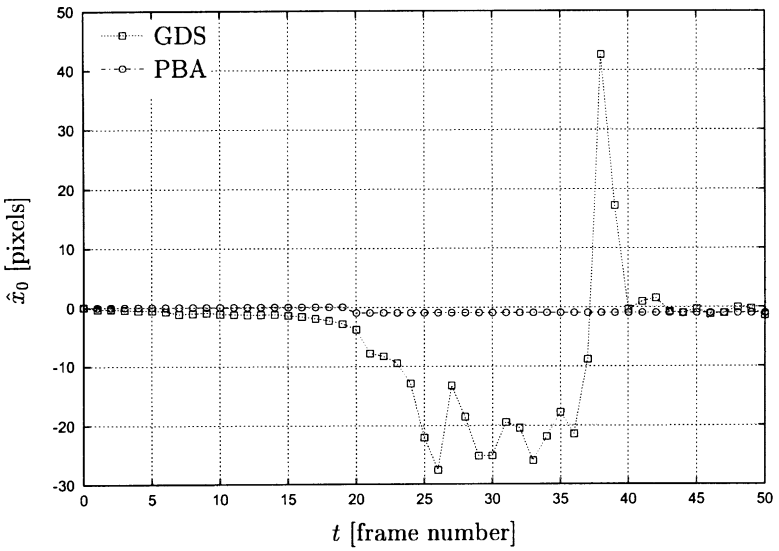
5.2. Real motion

5.2.1. Setup

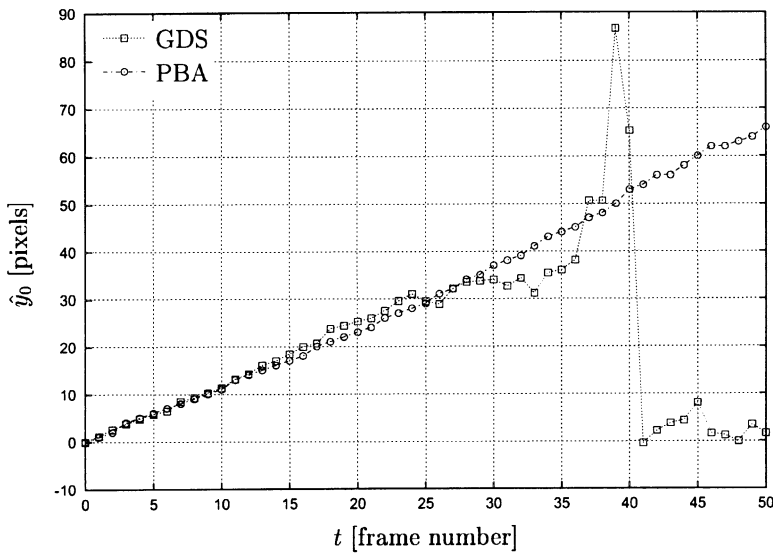
As an example of the effectiveness of the technique in real images, we used the sequence `building01`, publicly available at `http://vasc.ri.cmu.edu/idb/html/motion/`. It is an aerial image of some model buildings, where the dominant motion is a vertical translation (Fig. 12(a) and (b)). However, because different points in the scene are at different distances to the camera, and due to motion parallax, not only the translation is different at different image locations, but also some points are occluded in some frames while visible in others.

The 51 images of the sequence were scaled to size 256×256, and then transformed to log-polar images, with the same parameters as in the synthetic motion ($\rho_0 = 5$, $\rho_{\max} = 128$, $R = 64$, $S = 128$). At each time step t, the input to the GDS and PBA algorithms are the first log-polar image (at $t = 0$) and the current log-polar image at $t \in [0, 51]$. This allows us to test GDS and PBA with increasing motion magnitude.

By observing the sequence (Fig. 12(b)), it is obvious that vertical translation increases monotonically with t, while horizontal translation is zero (or almost). This gives a first clue on which translational estimates can be expected and which ones would clearly be wrong. To learn about the maximum displacement, we manually measured the vertical shift between the first and last frames, and it was found to be about 65 pixels.



(a) horizontal shift, $\hat{x}_0$



(b) vertical shift, $\hat{y}_0$

Fig. 13. Estimated translational parameters in the real-motion experiment.

Therefore, if we want GDS and PBA deal with such big translations, their parameters should be customized to the expected motion range. To that end, we set $k_{\max} = 100$ in GDS, to give the steepest descent greater chances of getting to the minimum. For PBA, we set $-d_{\min} = d_{\max} = 70$, and $w = 100$.

5.2.2. Results

The estimated translation parameters $(\hat{x}_0, \hat{y}_0)$ are plotted in Fig. 13. As expected, $\hat{x}_0 \approx 0$ (Fig. 13(a)); however, this is true for the whole sequence with PBA, but, for GDS, $\hat{x}_0$ becomes noisy during frames 20–40. Similarly, the estimation of $\hat{y}_0$ (Fig. 13(b)) is stable and reasonably good with PBA; however, for GDS, estimates from $t = 30$ onwards become erratic.

These results are in agreement with those obtained with the synthetic-motion experiment, namely: both GDS and PBA can cope with small-medium translations, but for larger translations, PBA behaves much better than GDS. It is also important the fact that both approaches estimate the dominant translation, even when the true deformation is not a *pure* translation (as it was in the synthetic-motion case). This indicates the effectiveness of the proposed techniques in real-world conditions, and their robustness against (moderate) non-modelled image deformations.

When ground-truth data are not available, as it is the case, the RMS (root mean square) image reconstruction error has been proposed to measure the estimation error [36]. We compute the RMS between the cartesian image at $t = 0$ and the different cartesian images at each $t \geq 0$. This error, which we call $\epsilon_\infty(t)$, is taken as a measure of how different images I_0 and I_t are. On the other hand, image $\hat{I}_t$ is the *rectified* (or *reconstructed*) image using the motion estimates $(\hat{x}_0, \hat{y}_0)$, i.e. $\hat{I}_t(x, y) = I_0(x + \hat{x}_0(t), y + \hat{y}_0(t))$. The RMS is then computed between $\hat{I}_t$ and I_t, with translation estimated with GDS and PBA, resulting in ϵ_g and ϵ_p, respectively. Ideally, if the estimates were perfect, $\hat{I}_t(x, y) = I_t(x, y)$, and then $\epsilon_p = \epsilon_g = 0$. These three RMS errors are plotted in Fig. 14. It can be noticed how ϵ_∞ grows with the image displacement, and how $\epsilon_g < \epsilon_\infty$ and $\epsilon_p < \epsilon_\infty$. On the other hand, $\epsilon_g(t) \approx \epsilon_p(t)$, for $0 \leq t < 15$, but $\epsilon_g(t) \gg \epsilon_p(t)$ for $t > 15$, when GDS starts to exhibit a bad performance.

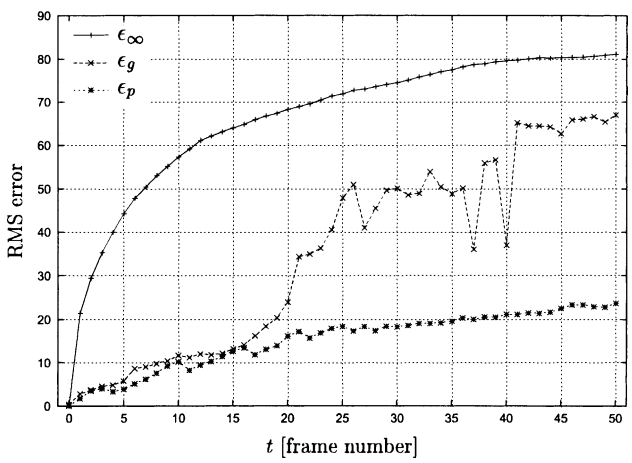


Fig. 14. RMS error evaluated at each frame in the real motion sequence.

In our case, the usefulness of the RMS error as a quantitative measure of the error committed in the translation estimation is limited, for two reasons: (i) not all points in the image undergone the same translation; and (ii) with so big image translations, the gray level of many image points in rectified images are undefined (for those points, $(x + \hat{x}_0, y + \hat{y}_0)$ lying outside the image). These two facts may affect in having a biased RMS error measure.

A qualitative measure of the performance of the algorithms can be visualized by subtracting $\hat{I}_t$ and I_t. In these images (Fig. 12(d) and (e), it is easy to evaluate the goodness of the estimation by observing the degree of *matching* between these pairs of images, and by comparing it with that of the *unregistered* pairs (Fig. 12(c)).

6. Comparing the methods

6.1. Conceptual and implementation complexity

Both methods are quite simple in nature, what makes them attractive, as they involve no (or little) conceptual complexity. The GDS approach carries a higher, but moderate, mathematical complexity (deriving the correlation gradient, computing and evaluating partial derivatives, etc.). In this regard, the PBA strategy is easier, with the added advantage of turning one 2D problem into two equal and independent 1D problems. Some minor difficulty in PBA lies in extracting the sets of pixels from the log-polar map $\mathscr{L}$ over which to perform the projections, i.e. computing the support maps.

6.2. Computational cost

The computational cost of the GDS algorithm is $O(k_{\max}RS)$, where $k_{\max}$ denotes the maximum number of iterations, and $R \times S$ is the size of the log-polar images, which is small compared to conventional cartesian images. The actual number of iterations k depends on the motion magnitude, how fast the descent is performed, and on the particular convergence criteria being used. Because the O-notation is being used, a worst-case analysis should be made. This is why $k_{\max}$ is used in the upper bound of the cost. In practice, the main factor affecting the actual number of iterations is the image displacement. Therefore, the efficiency of the algorithm is guaranteed as long as only small translations of the target onto the image plane occur.

Regarding the initial guess, (x_0^0, y_0^0), in the absence of any other information, it is sensible to use $(0, 0)$. However, during an active tracking, it may be expected the target to move following some dynamics. Therefore, if the target

Table 1
Off-line times in GDS and PBA

	GDS	PBA			
w	–	20	30	40	50
Time (ms)	30	2700	2800	2900	3100

motion can be predicted, we can use the expected future target position as the initial guess for the algorithm. This would help speed up the motion estimation process.

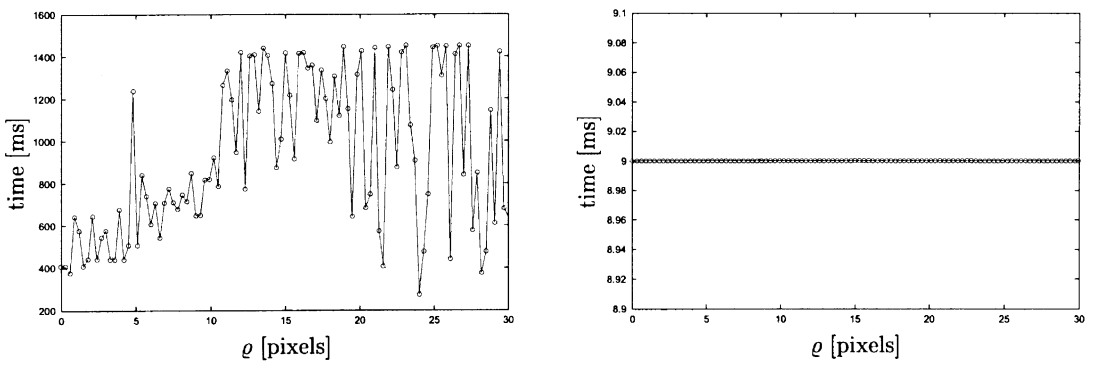
Look-up tables storing the values of the partial derivatives in Eq. (8) can be computed in an off-line stage, thus saving time during the on-line computation of Eqs. (6) and (7).

As for the computational cost of PBA, we have to take into account the number of vertical and horizontal projections, $2m_V$ and $2m_H$, respectively (see Fig. 6), which determine the size of the resulting 1D projection signals. On the other hand, the extent of each projection, i.e. $2n_V$ and $2n_H$, should be considered. The cardinality of the set $\{(u,v) \in \mathscr{L}(i,j)\}$, used in Eqs. (9) and (10) for a given (i,j), depends on the accuracy which the map $\mathscr{L}$ was computed with, and on the resolution of the log-polar image, but it tends to be small. Additionally, we take now into account the simplifications considered in Section 4.2, regarding $m_V = m_H = m$, $n_V = n_H = n$. Therefore, computing a projection is an operation with a cost $O(mn)$. With the additional simplification $m = n = w$, the cost can be expressed as $O(w^2)$. We are assuming the maps $\mathscr{M}_v$ and $\mathscr{M}_h$ are computed off-line, thus saving a precious time for on-line computations. It remains to be considered the cost of estimating the 1D shift. This depends on the particular strategy used, and can be represented by $O(l)$, $l = |D|$. Thus, the total cost (computing the projections plus evaluation the 1D translations) is $O(lw^2)$. Notice that these computations have to be performed twice (once for each kind of projection), but this does not affect the asymptotic cost.

Comparing the asymptotic costs of GDS and PBA, we have that, in general, $w^2 < RS$, and, $k_{\max} \approx l$. Therefore, $lw^2 < k_{\max}RS$, which indicates that PBA's asymptotic cost is lower than GDS'. Regarding the memory requirements, both approaches are asymptotically similar, as they both keep LUTs of sizes of the same order of magnitude. In practice, PBA actually requires more memory than GDS, due to the need to store the support maps.

To compare both approaches in terms of actual running times, we present some figures, obtained with a PC with a Pentium III (500 MHz, 800 Mbytes), and Linux as the operating system. Table 1 shows times corresponding to the initialization stage of the algorithms. Notice the substantial times (around 3 s) that, in the PBA, are involved in the off-line computation of the projections support maps. This also indicates how important this computation step is in order to save time during on-line processing, and make real-time processing possible. In GDS, the savings are somehow less remarkable.

In Fig. 15, we show the times for both algorithms at each of the same tests performed in Section 5.1.2, i.e. with increasing motion magnitude. It is clear from the plots that, while the cost in PBA is constant, in GDS it depends on the magnitude of the displacement: the greater the displacement the more iterations are generally needed. On the other hand, PBA is very efficient, as only 9 ms per step are needed, while around 500 ms are spent in each step with GDS (for the smaller displacements). As pointed out before, using the motion estimates at one step to initialize the descent at the following step is a possible measure to stabilize the times needed in GDS. Two observations should be done, however. On the one hand, the improvements are not very significant. On the other hand, care has to be taken when doing this: if estimates are not filtered, the above measure may be counterproductive as spurious erroneous estimates (such



(a) GDS (b) PBA

Fig. 15. On-line times in GDS and PBA.

as those we have mentioned before) would propagate in subsequent steps.

6.3. Translation estimation performance

For small-medium displacements (say, up to 10 pixels), both approaches yield very similar results. However, GDS cannot effectively deal with larger translations, while PBA can robustly tackle these larger retinal shifts, provided its parameters have properly been set up (e.g. it would not be possible to detect an absolute displacement of 25 pixels with only $w = 10$, or with $[d_{\min}, d_{\max}] = [-15, 15]$, for instance).

7. Conclusions

A brief presentation of the log-polar mapping has been given. Then, the importance of translation estimation and its difficulty in cortical images has been shown. Next, two approaches (GDS and PBA) that effectively deal with translation estimation in the complex logarithmic domain have been introduced, and experiments with both synthetic and real motion estimation are reported.

Both strategies have been proven to be efficient. However, even when their asymptotic cost may be similar, PBA turns out to be a much more efficient approach. Furthermore, in our implementation, PBA has a fixed cost, while GDS has a varying cost, which depends on the convergence speed which, in turn, is a function of the translational magnitude. On the other hand, carrying out some off-line computations has been shown to be crucial in order to run these algorithms in real time.

As for their abilities to estimate translation, both approaches exhibit a similar behavior with small-medium displacements. GDS, though, becomes unreliable for larger displacements, whereas PBA is able to deal with a larger range of motions. PBA can be tuned to get a trade-off between the range of motions to be estimated and the allowable computational cost.

Both methods may provide spurious estimates. However, filtering techniques (e.g. a Kalman filter) can effectively deal with them. The continuity of target motion can be exploited not only to deal with the estimation variance, but also to make the process faster. Filtered estimates at one step in the flow of frames can be used as initial guesses to find the estimate at the following time step. This makes sense in active visual tracking applications.

Acknowledgements

This work is partially supported by projects GV97-TI-05-27 from the Conselleria d'Educació, Cultura i Ciència, Generalitat Valenciana, and CICYT TIC98-0677-C02-01 from the Spanish Ministerio de Educación y Cultura.

References

[1] J.Y. Aloimonos, I. Weiss, A. Bandyopadhyay, Active vision, International Journal of Computer Vision (1988) 333–356.
[2] F.L. Lim, G.A.W. West, S. Venkatesh, Tracking in a space variant active vision system, International Conference on Pattern Recognition (ICPR), Vienna, Austria (1996) 745–749.
[3] M. Balsi, Cellular neural networks defined on log-polar grids for artificial vision, European Conference on Circuit Theory and Design (ECCTD), Budapest, Hungary (1997).
[4] E.L. Schwartz, Spatial mapping in the primate sensory projection: analytic structure and relevance to perception, Biological Cybernetics 25 (1977) 181–194.
[5] S. Shah, M.D. Levine, Information processing in primate retinal cone pathways: a model, Technical Report TR-CIM-93-18, Center for Intelligent Machine, McGill University, Montreal, Québec, Canada, December 1993.
[6] R.M. Hodgson, R.I. Chaplin, W.H. Page, Biologically inspired image processing, International Conference on Image Processing and its Applications (IPA) (1995) 11–15.
[7] H. Yamamoto, Y. Yeshurun, M.D. Levine, An active foveated vision system: attentional mechanisms and scan path convergence measures, Computer Vision and Image Understanding (CVIU) 63 (1) (1996) 50–65.
[8] Z. Mikrut, B. Czwartkowski, Log-Hough space as input for a neural network, Fourth Conference on Neural Networks and their Applications, Zakopane, Poland (1999).
[9] J.A. Boluda, F. Pardo, T. Kayser, J.J. Pérez, J. Pelechano, A new foveated space-variant camera for robotic applications, IEEE International Conference on Electronics Circuits and Systems, Rodos, Greece (1996) 680–683.
[10] G. Sandini, P. Questa, D. Scheffer, B. Dierickx, A. Mannucci, A retina-like CMOS sensor and its applications, Proceedings of the First IEEE SAM Workshop, Cambridge, USA (2000).
[11] F. Tong, Z.-N. Li, Reciprocal-wedge transform for space-variant sensing, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 17 (5) (1995) 500–511.
[12] A. Bernardino, J. Santos-Victor, Binocular tracking: integrating perception and control, IEEE Transactions on Robotics and Automation 15 (6) (1999) 1080–1094.
[13] C.F. Weiman, R.D. Juday, Tracking algorithms using log-polar mapped image coordinates, Intelligent Robots and Computer Vision VIII: Algorithms and Techniques, SPIE (1989) 843–853.
[14] H. Tunley, D. Young, Dynamic Fixation of a Moving Surface using Log Polar Sampling, British Machine Vision Conference, vol. 2, BMVA Press, 1994, pp. 579–588.
[15] K. Daniilidis, V. Krüger, Optical flow computation in the log-polar plane, in: V. Hlaváč, R. Šára (Eds.), International Conference on Computer Analysis of Images and Patterns (CAIP), Springer, Berlin, 1995, pp. 65–72.
[16] J. Dias, H. Araujo, C. Paredes, J. Batista, Optical normal flow estimation on log-polar images. A solution for real-time binocular vision, Real Time Imaging 3 (3) (1997) 213–228.
[17] V. Krüger, Optical flow estimation in the complex logarithmic plane, Master's thesis, Lehrstuhl für Kognitive Systeme, Institutus für Informatik and Praktische Mathematik, 1995.
[18] F. Panerai, C. Capurro, G. Sandini, Space variant vision for an active camera mount, Technical Report TR l/95, LIRA, DIST, University of Genova, Italy, February 1995.
[19] V.J. Traver, F. Pla, Translation estimation in log-polar images, in: J.S. Sánchez, F. Pla (Eds.), Spanish Symposium on Pattern Recognition and Image Analysis (SNRFAI), Castellón, Spain, vol. II, 2001, pp. 281–286.

[20] I. Ahrns, H. Neumann, A view-based approach for real-time fixation using log-polar mapping, in: C. Freksa (Ed.), Proceedings of Artificial Intelligence, 1998, pp. 89–96.
[21] A. Bernardino, J. Santos-Victor, G. Sandini, Tracking planar structures with log-polar images, Symposium on Intelligent Robotic Systems, Reading, UK (2000) Also, as VisLab TR 06/2000.
[22] N. Okajima, H. Nitta, W. Mitsuhashi, Motion estimation and target tracking in the log-polar geometry, 17th Sensor Symposium, Kawasaki, Japan (2000).
[23] N. Oshiro, N. Maru, A. Nishikawa, F. Miyazaki, Binocular tracking using log polar mapping, IEEE/RSJ International Conference on Intelligent Robots and Systems, Osaka, Japan 2 (1996) 791–798.
[24] A. Bernardino, J. Santos-Victor, Visual behaviors for binocular tracking, Robotics and Autonomous Systems 25 (1998) 137–146.
[25] M. Bolduc, M.D. Levine, A review of biologically motivated space-variant data reduction models for robotic vision, Computer Vision and Image Understanding (CVIU) 69 (2) (1998) 170–184.
[26] R. Alan Peters II, M. Bishay, T. Rogers, On the computation of the log-polar transform, Technical Report, School of Engineering, Vanderbilt University, March 1996.
[27] F. Jurie, A new log-polar mapping for space variant imaging. Application to face detection and tracking, Pattern Recognition 32 (1999) 865–875.
[28] C. Capurro, F. Panerai, G. Sandini, Dynamic vergence using log-polar images, International Journal of Computer Vision 24 (1) (1997) 79–94.
[29] S.S. Rao, Optimization: Theory and Applications, Second ed., Wiley Eastern Ltd, 1984.
[30] W.H. Press, S.A. Teukolsky, W.T. Vetterlng, B.P. Flannery, Numerical Recipes in C. The Art of Scientific Computing, Cambridge University Press, Cambridge, 1992.
[31] L.G. Brown, A survey of image registration techniques, ACM Computing Surveys 24 (4) (1992) 325–376.
[32] L. Lucchese, A frequency domain technique based on energy radial projections for robust estimation of global 2D affine transformations, Computer Vision and Image Understanding (CVIU) 81 (2001) 72–116.
[33] K. Daniilidis, Attentive visual motion processing: computations in the log-polar plane, Computing 11 (1996) 1–20.
[34] A. Giachetti, Matching techniques to compute image motion, Image and Vision Computing (IVC) (18) (2000) 247–260.
[35] J. Harris, H. Stocker, Handbook of Mathematics and Computation Science, Springer, Berlin, 1998.
[36] T. Lin, J.L. Barron, Image reconstruction error for optical flow, Vision Interface (VI), Banff, Canada (1994) 73–80.