Normal Optical Flow based on Log-Polar Images

Inácio Sousa, Jorge Dias, Helder Araújo*

Institute of Systems and Robotics (ISR),
Electrical Engineering Department, University of Coimbra,
Largo Marquês de Pombal, 3000 Coimbra - PORTUGAL;
Phone: +351 (39) 34576/34884;
Fax: +351 (39) 35672;

Abstract. This paper addresses the problem of computing the normal optical flow in log-polar images. The log-polar allows the detection of a velocity field growing in magnitude from the center to the peripheral image zone. The mapping between the image plane and the log-polar plane allows the speed up of the optical flow computation, while it gives an excellent sampling structure to perform the detection of small central motion and large peripheral motion. The method described on the paper also explores multi-resolution technique in order to check the coherency of the velocity field. In the results the method is compared with other algorithms based on the brightness constraint and the affine velocity model. This algorithm is currently used to generate a controlling feedback signal for navigation control of an autonomous robot with an active vision system.

1 Introduction

Considerable efforts have been applied in the development of different optical flow techniques (or image velocity estimation) to solve the problem of tracking a moving object, or performing navigation using an autonomous robot. These problems can be hard to solve, since the information available in the image plane is the projection of the 3D object’s velocity, and in some situations can not be totally estimated.

The gradient-based techniques exploits the so-called “brightness constraint” equation:

$$\nabla I(x, y, t) \cdot (I_x, I_y, I_t) = 0 \quad (1)$$

where \( V(x, y, t) = (I_x, I_y, I_t) \) is the "optical flow" and \( \nabla I(x, y, t) = (I_x(x, y, t), I_y(x, y, t), I_t(x, y, t)) \), with \( I_x(x, y, t) = \frac{\partial I(x, y, t)}{\partial x}, I_y(x, y, t) = \frac{\partial I(x, y, t)}{\partial y}, I_t(x, y, t) = \frac{\partial I(x, y, t)}{\partial t} \). Introducing the variables \( u = \frac{\partial x}{\partial t} \) and \( v = \frac{\partial y}{\partial t} \), we can write the equation (1) as,

$$\begin{bmatrix} I_x & I_y \\ I_t \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -I_t \quad (2)$$

which represents a line equation in the \((u, v)\) space. The equation (1) is the result of the Taylor series expansion

![Diagram](image)

Fig. 1.: The brightness constraint representation in the \((u, v)\) space and the aperture problem representation for the equation

$$I(x + u \Delta t, y + v \Delta t, t + \Delta t) = I(x, y, t),$$

* Email:sousa@isr.uc.pt
and it will be a valid equation only for a small \( \Delta t \) time. This constraint assumes that the image brightness \( I(x, y, t) \) will have a small change in the elapsed time \( \Delta t \) between the two consecutive images, and \( I(x, y, t) \) is a differentiable function. The equation (2) is the vectorial version of the brightness constraint and, if we only use this restriction for “optical flow” estimation, only the projection of the “optical flow” in the gradient direction can be calculated. This is related with the so-called aperture problem (see Fig. 1). From the figure we conclude that only the component of the apparent velocity which is normal to the object edge could be detected. This component is the “normal optical flow” and is given by the equation:

\[
V_{\text{normal}} = -\frac{I_x}{\sqrt{I_x^2 + I_y^2}} = -\frac{I_x}{\|\nabla I\|} \tag{3}
\]

derived from (2). The relationship between the normal optical flow calculation with the normal velocity field can be achieved. Let us consider a point \((x, y)\) with optical flow \( \mathbf{V} = (u, v)^T \), the local normalized gradient is then: \( \mathbf{n} = \frac{\nabla I}{\|\nabla I\|} = \frac{1}{\sqrt{I_x^2 + I_y^2}} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \) so the normal velocity field is equal to:

\[
u_n = \mathbf{n}^T \mathbf{V} = \frac{1}{\|\nabla I\|} (I_x u + I_y v). \tag{4}
\]

If the both fields are equal its subtraction be zero:

\[
u_n - V_{\text{normal}} = \frac{1}{\|\nabla I\|} (I_x u + I_y v + I_x) = \frac{1}{\|\nabla I\|} \frac{dI}{dt}. \tag{5}
\]

From (5) we could conclude, the two fields are similar only if the local gradient is high, or the total derivative of 1 is close to zero.

The normal flow gives a useful estimation of the motion, and Cornelia Fermuller and Yiannis Aloimonos used it in order to estimate the 3-D motion and the FOE to perform tracking [2].

David Coombs and others [5], also reported a process to perform real time obstacle detection based on peripheral and divergence flow from the normal flow. They use the peripheral flow to steer the robot through the corridor, and the divergence flow to estimate the time to collision. They reported experiences with a robot running at 30cm/s for as long as 20 minutes without collision.

Using only the brightness constraint for optical flow estimation will conduct to algorithms with unstable solutions since the constraint didn’t check the local motion coherency by itself. Better solutions are obtained by minimizing a functional that controls the balance between the values obtained from the optical flow.

An example using other techniques was purposed by Peter Nordlund and Tomas Uhlin [6] where they presented an on-line algorithm for estimating and pursue a moving object. Their system is able to maintain the object centered in the image, by controlling a robot-head. The motion detection method, is based on the affine velocity model using an image stabilization process before computing the motion.

J.L. Barron and others [4], tested different optical flow techniques on real and synthetic image sequences. They reported some results from techniques based on differential, region matching, energy, and phase methods.

The algorithm described in this article is currently being used to generate a controlling feedback signal for navigation control for an autonomous robot with an active vision system as described on [7].

2 Log-polar Mapping

2.1 Log-polar mapping, the continuous case

A mapping between the image plane and the log-polar plane can be made and the transformation between the two planes in the most simple form are resumed to a transformation from cartesian to polar coordinates, followed by the application of the logarithmic function to the radial \( \rho \) coordinate (\( \rho > 0, \theta \in [0..2\pi] \), \( a, b > 1 \)).

\[
\begin{align*}
x(\rho, \theta) &= a^\rho \cos(\theta) \\
y(\rho, \theta) &= b^\rho \sin(\theta)
\end{align*}
\]

(6)

The constants \( a \) and \( b \) will control the transformation evolution in the radial direction, and a circle (\( a = b \)) or ellipse (\( a \neq b \)) mapping can be achieved for particular values. The Fig. 2 shows a graphical interpretation for the log-polar mapping for \( a = b \). Differentiating the (6) equation in order to \( t \), will result in the following equation, which represents the relation from the velocity field on the image log-polar plane to the image cartesian plane:

\[
\begin{bmatrix}
\frac{\partial x(\rho, \theta, t)}{\partial t} \\
\frac{\partial y(\rho, \theta, t)}{\partial t}
\end{bmatrix} = \begin{bmatrix}
a^\rho \ln(a) \cos(\theta) & -a^\rho \sin(\theta) \\
b^\rho \ln(b) \sin(\theta) & b^\rho \cos(\theta)
\end{bmatrix} \begin{bmatrix}
\frac{\partial \rho(t)}{\partial t} \\
\frac{\partial \theta(t)}{\partial t}
\end{bmatrix}
\]

(7)

278
Using the notation \( \dot{\rho} = \frac{\partial \rho}{\partial \xi} \), and performing the inverse transformation of (7) will result in the relation from the field on the image cartesian plane to the image log-polar plane:

\[
\begin{bmatrix}
    \dot{\rho} \\
    \dot{\theta}
\end{bmatrix} = \frac{1}{\cos(\theta)^2 \ln(\frac{b}{a}) + \ln(b)} \begin{bmatrix}
    \cos(\theta) & \sin(\theta) \\
    \frac{a^2}{\sin(\theta)} & \frac{b^2}{\sin(\theta)}
\end{bmatrix}
\begin{bmatrix}
    \dot{x} \\
    \dot{y}
\end{bmatrix}
\]  

(8)

and as can be seen, the optical flow detection on the log-polar plane is equal to the flow on the image plane, but divided by the radial distance \( a^2 \) and \( b^2 \) from the center (the matrix is just a rotation transformation). This property is the base for our algorithms, because, we can detect growing flows from the center toward the periphery, while with other techniques, it will be very difficult.

2.2 Log-polar mapping, the discrete case

In the discrete case, each pixel \((\rho, \theta)\) in the log-polar plane will correspond one pixel \((x, y)\) in the image plane. The intensity value of the pixel \((\rho, \theta)\) in the log-polar plane is obtained by the application of a gaussian mask with a kernel of \(5 \times 5\) centered at the pixel \((x, y)\). The relation between the \((\rho, \theta)\) and \((x, y)\) is based on the following equations (for \( b = a \)):

\[
x = a^\rho \cos(\frac{2\pi \theta}{N_{ang}}) \\
y = a^\rho \sin(\frac{2\pi \theta}{N_{ang}})
\]  

(9)

where \( N_{ang} \) denotes the number of samples in the angular direction, and the value of \( \rho_{fovea} \) is a constant value in order to avoid the pixel sobereposition for \( \rho = 0 \). This value should be chosen equal or greater to the minimum angular sampling period to cover all the circle perimeter without generating oversampling in the image plane and it is given by

\[
\rho_{fovea} = \log_a(\frac{N_{ang}}{2\pi}).
\]  

(10)

The application of this constraint results in a central circle in the image plane which is not mapped for the log-polar plane, as can be seen in the Fig. 2. The dimension of this circle can be controlled by changing the number of samples in \( \theta \) direction \( (N_{ang}) \) and the value for the base of the logarithmic function. The constant value \( a \) will control the evolution of the radius of sampling in the \( \rho \) direction. From the application of the equations (9) for \( \theta = 0, ..., N_{ang} \) and \( \rho = 0, 1, 2, 3, ... \) will result the sampling structure represented in the Fig 2. Another important feature, is the fact that the resolution in the \( \theta \) direction of the log-polar is constant for all radius, which results an image in the log-polar plane with more information in the central zone and decreasing toward the periphery.

The log-polar has many interesting properties that make its application a very useful tool in the vision field. Some of these are the rotation and scaling invariance (with respect to the center of the image plane). But, the most important one is the motion field projection in the log-polar plane when the observer makes rotational or translational movements. We are interested here, in what happens to the optical flow field for some motion types. Notice that the optical flow generated in the images captured by a camera with translational velocity just along the optical axis will generate a divergence flow growing from the image center. In this case the focus of expansion (FOE) of the optical flow field is in the image center, see Fig. 3). This vector field in log-polar will appear as a constant field with the same angular orientation, as can be seen in Fig 3, and if we perform a histogram in the \( \rho \) direction will result in a constant level for all \( \theta \) entries. This kind of flow is of most concern in our experiments.
Fig. 3.: Relation between the optical flow for different types of motion. From the left to the right is represented the optical flow for different types of motion. For each type of motion, the optical flow is represented in the cartesian plane and in the log polar. The left images are the representation of a divergence flow growing from the image center, and the flow in log-polar, will be in this case a constant flow. The central and right images corresponds to a translational and vertical flow growing from the center and this flow has a constant value on the log-polar plane, but with different orientation.

3 Motion Detection

Three methods for motion detection are implemented for estimation of optical flow and normal optical flow. In our experiments the three methods are applied on the log-polar plane. Two of them are concerned with the optical flow estimation, while the other is for computation of the normal flow.

For every point in the image, the gradient is computed using a 3 x 3 spatial gradient operator for the estimation of the \( I_p(\rho, \theta, t) \) and \( I_\theta(\rho, \theta, t) \).

In practice, only the pixels with gradient and temporal derivative above a threshold will be considered, in order to avoid the errors caused by the noise associated with differential calculations. This process we call the pre-processing stage and, is just used for the normal flow algorithm.

3.1 Normal flow algorithm

The normal optical flow is determined by exploiting the brightness constancy constraint and is defined as:

\[
V_{\text{normal}}(\rho, \theta, t) = -\frac{I_\theta}{\nabla I \cdot \tau}(\frac{\partial \rho}{\partial t}, \frac{\partial \theta}{\partial t}).
\]  

(11)

This vector will be generated as a direct result of the results at the end of the pre-processing stage.

3.2 Affine model algorithm

The second method exploits the "brightness constancy constraint" in conjunction with an affine transformation between the consecutive images [6]. The image affine velocity model is generated based on the following relation:

\[
[V_p \ V_\theta]^T = [a_1 + a_2 \rho + a_3 \theta \ b_1 + b_2 \rho + b_3 \theta]^T.
\]

The number of parameters to estimate in the above equation is 6 and are the result of a least square minimization process for the equation, \( \min(\sum (I_p \cdot V_p + I_\theta \cdot V_\theta + I_t)^2) \), and can be computed by the following relation:

\[
\begin{bmatrix}
\sum I_p^2 & \sum I_p I_\rho & \sum I_\rho^2 \\
\sum I_p I_\rho & \sum I_\rho I_\theta & \sum I_\theta^2 \\
\sum I_\rho^2 & \sum I_\rho I_\theta & \sum I_\theta^2
\end{bmatrix} \begin{bmatrix}
a_1 \\
a_2 \\
a_3 \\
b_1 \\
b_2 \\
b_3
\end{bmatrix} = \begin{bmatrix}
-\sum I_p I_\theta \\
-\sum I_\rho I_\theta \\
-\sum I_\theta I_\theta \\
-\sum I_\rho I_p \\
-\sum I_\rho I_p \\
-\sum I_\rho I_\theta
\end{bmatrix}
\]

If the determinate of the symmetrical matrix is zero or below a threshold, we will assume the velocity model:

\[
[V_p \ V_\theta]^T = [a_1 \ b_1]^T.
\]

In this case there is only two parameters to estimate:

\[
\begin{bmatrix}
\sum I_p^2 & \sum I_\rho I_\rho & \sum I_\theta I_p \\
\sum I_p I_\rho & \sum I_\rho I_\theta & \sum I_\theta I_\theta
\end{bmatrix} \begin{bmatrix}
a_1 \\
b_1
\end{bmatrix} = \begin{bmatrix}
-\sum I_p I_\theta \\
-\sum I_\rho I_\theta
\end{bmatrix}
\]

(12)

If the determinate of (12) still below a threshold or null, we will assume that the velocity is null. In the above equations the sum will be performed over a 23 x 36 pixels region to compute the 6 or 2 parameters (this is chosen in this way to have the same division in squares in both directions on the log-polar image plane).
3.3 Horn and Schunck algorithm

The third method, used is the Horn and Schunck process [4]. This process combines the "brightness constancy constraint" with a global smoothness term to constrain the estimated velocity field. This method defines "the departure from smoothness" measure \( s(\rho, \theta) \) and the error in the optical flow constraint equation as:

\[
s(\rho, \theta) = \frac{1}{2}((V_{\rho}(\rho, \theta + 1, \theta) - V_{\rho}(\rho, \theta))^2 + (V_{\theta}(\rho + 1, \theta) - V_{\theta}(\rho, \theta))^2 + (V_{\rho}(\rho, \theta + 1) - V_{\rho}(\rho, \theta))^2 + (V_{\theta}(\rho + 1, \theta) - V_{\theta}(\rho, \theta))^2)
\]

\[
c(\rho, \theta) = \frac{1}{2}(I_{\rho}V_{\rho}^k + I_{\theta}V_{\theta}^k + I)
\]

The main goal is to find the resulting values that minimize the equation

\[
\sum [s(\rho, \theta) + \lambda c(\rho, \theta)]
\]

The iterative equations used to obtain the image velocity are then:

\[
V_{\rho}^{k+1} = V_{\rho}^k - \frac{I_{\rho} - I_{\theta}(V_{\theta}^{k+1} + V_{\theta})}{1 + \lambda I_{\rho}^2}
\]

\[
V_{\theta}^{k+1} = V_{\theta}^k - \frac{I_{\theta} - I_{\rho}(V_{\rho}^{k+1} + V_{\rho})}{1 + \lambda I_{\theta}^2}
\]

where \( k \) denotes the iteration number, \( V_{\rho}^0 \) and \( V_{\theta}^0 \) denote initial velocity estimates set to zero and \( V_{\rho}^k \) and \( V_{\theta}^k \) denote neighbor averages. In our experiments, these averages are taken from the 8 pixels surrounding the \((\rho, \theta)\) pixel. The results reported had used 5 iterations with \( \lambda = 3.5 \).

3.4 Post-processing stage

![Fig. 4.: The optical flow coherency checking stage](image)

The post-processing stage, will be the checking for coherency based on a gaussian pyramid. To check the estimation of the normal/optical flow consistency a gaussian pyramid is used with three levels. First the flow is checked only in the current level, by using a region of \( 4 \times 4 \) (the region is dependent on the log-polar sampling structure). The flow \((V_{\rho}, V_{\theta})\) will be the most voted flow in the region.

This flow will have the radial and angular components given by the cell \( V_{\rho}(\rho_l, \theta_l) \) and \( V_{\theta}(\rho_l, \theta_l) \), where \( \rho_l \) and \( \theta_l \) are calculated as:

\[
\theta_l = \text{index}_{\text{radial}}(\max \{ \text{hist}_{\text{radial}}(\theta) \})
\]

\[
\rho_l = \text{index}_{\text{angular}}(\max \{ \text{hist}_{\text{angular}}(\rho) \})
\]

where \( \text{hist}_{\text{radial}}(\theta) = \sum_{\theta=0}^{3} V_{\rho}(\rho, \theta) \) and \( \text{hist}_{\text{angular}}(\rho) = \sum_{\rho=0}^{3} V_{\theta}(\rho, \theta) \). This process is illustrated in Fig. 4.

Climbing one level in the pyramid results in an image with half the dimensions, with each pixel being obtained by the mean of 2 \times 2 group of pixels of the image directly below it. The resulting image will be then filtered with a gaussian function \((\sigma = 2 \text{ pixels})\) using a 5 \times 5 convolution kernel, to smooth the image. This kernel is designed to low-pass filter the image as illustrated in Fig. 2. We check the motion in different levels. Checking through the gaussian pyramid will work in following way: all the pixels will be scanned in the 3 levels to check the orientation and magnitude consistency. The resulting flow value is available through a voting process based on the best of two (if they are closer from each other), and if not a null value will be assumed.

The values of optical flow in each image point are continuously tracked by an alfa-beta tracker. This is just for better estimation and is also useful when the region does not have enough texture to estimate the local optical flow. In that case the pixels exhibit null flow and it will be given by the prediction of an alfa-beta filter (just for 3 consecutive missing samples).

In each level an equal process as described in (15) will be made, but with the difference in the region size (the region will have half the dimensions of the below level).
4 Experimental Results

Fig. 5.: Evolution of the Normal flow algorithm. From left to right, is represented the scene evolution, and the respectively flow on the log-polar plane.

The examples in Fig. 5, 6 and 7 are the results from the algorithms for normal optical flow, the optical flow using the affine model and the Horn and Schunck method. The images are sintetized and simulating a camera translating in the direction of the optical axis. From the pictures, we can see that the motion was detected by all methods and the best coherence was achieved by the affine model. However the other two algorithms have also a good behavior but from the three methods the normal optical flow is the fastest method. The performance given by the normal flow algorithm are mainly due to pre- and post-processing stages. If we use the same methodology on the affine and Horn and Schunck algorithms the results are very similar but the execution time grows also.

The robustness of a motion detection algorithm can be tested on situations where the amplitude of image velocity field grows continuously with time. That will show if the algorithms are able to continuously tracking the optical flow field. We tested the same situation for the three methods by simulating a continuously growing linear velocity. In this case the visual scene is oblique, and the Fig 8 shows the temporal evolution of the $\rho$ velocity component for the three methods.

Fig. 6.: Evolution of the affine model algorithm

Fig. 7.: Evolution of the Horn and Schunck algorithm

5 Summary and Future Work

This work described an application of the normal/optical flow methods on the log-polar plane, and from the results we take the following conclusions:

1. the log-polar image resolution is below than the image plane resolution, so working in the log-polar will save some time in the computations,
2. the image structure is good enough to detect the optical flow generated by a moving observer.

Currently we are addressing the application of this method to control the motion of a moving robot. The main idea, is to achieve an equilibrium between the flow in the left and right cameras. The problem resolution is however very difficult when the observer is moving, because the independent image motion has to be discriminated from the self-induced movement and can be solved by using visual fixation [7] [3].

282
Fig. 8.: Temporal evolution of the \( p \) velocity for 2 pixels (\(+\)-Normal flow *-Affine model o-Horn and Schunck), and the simulated camera's linear velocity

A Real Time Implementation

To perform the real time implementation, we use an image processing board, with two C40s processors. Only one of the C40 processor, has image acquisition capabilities, and is designated as the master C40, and the other is the slave processor, as can be seen in Fig 9. The host computer (a PC in our case), communicates with the master C40, through a FIFO memory channel, and the booth processors can perform transfers between them through a DMA channel for big transfers, and a Port channel for commands purposes. The time wasted to transfer data between the C40s, is almost null, because the DMA channel has a big bandwidth and the C40s can receive and send data while they are processing. The frame grabber can do acquisitions at a video rate frequency (25Hz), and has multiple video entries, with the capability of switching from one of them, but just one of the entries can be captured at a time. Since we have two cameras, and the analogue video format is interlaced PAL signal, to speedup the acquisition process, we only use one of the fields for each camera, to get an image acquisition rate of 25Hz, for both cameras. For this process work, the two cameras

![Image Processing Board](image)

Fig. 9.: The Image Processing Board (left) is used in the autonomous Active Vision system (right) for development of navigational tasks based on visual information.

must be synchronized. The acquisition time, can be hidden, in the optical flow calculation time, since multiple buffers can be defined, and the C40 allow us to use a video acquisition interrupt, which is activated at the end of the video field, and we can guarantee then, that for each camera the signal is always taken from the same field (the odd or even it doesn't matter). The interrupt after switching the video input, will perform the registration of the current time (care must be taken in the implementation of the interrupt, since it will run at a rate of 50Hz). When a new cycle processing arrives, the master C40 starts the log-polar sampling of the left image, and will download it to the slave, after that we will perform a log-polar sampling of the right image, then they will start the normal optical flow calculations. When the results are ready, the slave will upload to the master that uploads to the host. The log-polar sampling is achieved by a sampling table, with the position of the cartesian plane where the sample must be taken. The surrounding pixels will be used to perform a special gaussian convolution, based only in shifts, and is supported by the following kernel:

\[
\begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1
\end{bmatrix}
\]

We use a table to speedup the normal flow calculations. In this table for each entry, we save the result of the following equation:

\[
\frac{E_1}{E_1^2 + E_2^2}
\]
Fig. 10.: Testing scene: Left - rotational movement and translational, the others the movement is just rotation. Top: One Pyramid level, Down: Two pyramid levels. The algorithm is the Normal Flow.

for $E_1, E_2 = 0, \ldots, 256$.

In this implementation we only use a two level pyramid approach as described earlier. The image in the first level of the pyramid have a size of 110x150 pixels. The time spending in one processing cycle is 0.1 seconds, which give us a frame rate of 10Hz. The Fig 10 shows the results given by this process.

References

2. C. Fermuller and Y. Aloimonos, The Role of Fixation in Visual Motion Analysis, Computer Vision Laboratory, Center for Automation Research, University of Maryland, 1992.

This article was processed using the \LaTeX \text{macro} package with SIRS96 style

284