Recovering Shapes and Detecting Movements using Fixation

Carlos Paredes * 
MSC student of Elect. Eng. Dept. 
Instituto de Sistemas e Robótica 
Universidade de Coimbra 
Largo Marquês de Pombal 
3000 Coimbra 
E-Mail: paredes@isr.uc.pt

Jorge Dias 
Professor of Electrical Eng. Dept. 
Instituto de Sistemas e Robótica 
Universidade de Coimbra 
Largo Marquês de Pombal 
3000 Coimbra 
E-Mail: jorge@isr.uc.pt

A. Traça de Almeida 
Professor of Electrical Eng. Dept. 
Instituto de Sistemas e Robótica 
Universidade de Coimbra 
Largo Marquês de Pombal 
3000 Coimbra 
E-Mail: aalmeida@isr.uc.pt

Abstract

This paper is about visual fixation strategies for artificial vision systems. The goal is to develop a set of image processing operations, combined to form a robust fixation algorithm, useful for solving problems like 3D structure recovering, movement detection and object tracking. No innovating image processing techniques are used. Instead, popular methods with well known performances were selected. An important requisite of the work that originated this paper is to generate a real-time implementation of the algorithm and use it on both active vision systems developed at the Systems and Robotics Institute in Coimbra. The simplicity and light computational burden is a common feature to all the practical solutions presented, with the algorithms being implemented using Digital Signal Processing technology (known as DSP).

keywords: Visual fixation, Log-polar plane, Edge extraction, Matching, Optical flow, Tracking

1 Introduction

Fixating is an important part of the human vision system. Humans tend to focus their attention somewhere on the scene around them, whatever the task they have at hand. Either facing a moving or stopped target, analyzing or just looking at it, both eyes are kept turned to the same part of the scene and keeping the object's image in the center of the retina (fovea). That is what we mean by fixation: keeping the same target in the center of both eyes. Translated as a computational procedure, fixating is obtained by keeping the same pattern in the center of two images, taken with stereo cameras mounted on some kind of artificial vision system. That is the central idea for this paper: the description of a computational procedure for visual fixation using an existing active vision system [1]. The procedure will be used to perform tasks like 3D structure recovering, movement detection and object tracking, using only the principles behind fixation.

2 Fixation Process

If we want a robust fixation algorithm, we must first understand how fixation works. Figure 1 shows the three stages of a general (i.e. both biological and computational) fixation process. The evolution of the process is as follows:

- During the REST state, the observer is not fixating on any particular subject. In human terms, this is what we call having a vacant look. To leave this state, the scene must change in some way. A common scene change is the appearance of some kind of movement.

- In the SELECTION state, the target responsible for the scene change becomes the fixation subject. The usual reaction is to immediately turn the eyes or cameras in the target's direction. This action is known as a saccade movement.

- The target may or may not be a suitable and/or interesting fixation subject. So, the system may
either proceed to the **FIXATION** state, or return to the **REST** state, respectively.

This is a simplified description of a general fixation procedure. Our goal is to build a computational system able to perform all the stages described above. This work aims to improve an existing functional system developed at our Institute [2]. In particular, the paper describes a more robust fixating/tracking strategy, based on better processing hardware and improved algorithms.

3 Hardware

Two requisites of the fixation algorithm are the simplicity and light computational burden of the several tasks. The global procedure must have real-time performance while running on the available hardware. Hence, it is important to know beforehand what is the processing power. Figure 2 shows the complete setup of one of the active vision systems and the available processing hardware. The system is driven by five stepper motors, allowing the individual rotation of the two CCD cameras (the vergence movement), the variation of the baseline, and the rotation and inclination of the system (the pan and tilt movements). The main processing unit is a PC, responsible for the user interface, the control of the stepper motors and the interaction with the DSP units. Those units are two TMS320C40 processors from Texas Instruments (referred to as C40 - see figure 3). They are responsible for all the image processing, including sampling the PAL video signals from the two cameras, executed by an independent framegrabber controlled by one of the C40 processors. The described setup allows for parallel processing/execution of all the main blocks of the system, therefore increasing the total throughput.

Figure 3: The DSP board. There are two C40 processors on the board. One combines fast processing and large memory capacity. The other supports a programmable resolution 8-bit monochrome framegrabber, with a $1024 \times 1024 \times 8$-bit RGB graphics display section and a 4-bit overlay plane. The video digitizer accepts both PAL and NTSC signals.

4 REST state: Log-Polar Plane

The **REST** state is the easiest processing stage. Detecting scene changes can be performed by simple methods like the computation of the difference between consecutive images in time. But, even when using the differences method, some questions come to mind: should we search the entire image area? should we use the maximum available resolution?

Using the human example, we can answer both questions. We should search the largest scene area we have access to, but at a higher resolution at the image center than at the image border. In other words, we want to cover as much of the scene as possible, and react to all the changes happening at the image center, but only to large changes if they appear at the image border. We can meet both conditions using the *log-polar* transformation.

The log-polar mapping approximates the topological transformation of the retinal image into its cortical projection, occurring in the human vision system [5] [6]. The result is the transformation of an uniformly sampled image to an image in the log-polar plane, where each point $(\rho, \theta)$ is equal to the original image’s pixel located at $(x, y)$, given by:

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

(1)

There are other more complex transformations, involving sub-pixel sampling and average computing, for example. The log-polar plane has several interesting properties, not explored here (see the cited texts). Note that, although even simple transformations like equations 1 contain computationally heavy functions, the coordinates will not be repeatedly computed, but stored in a buffer table. Those values will then be used to apply the log-polar transformation to the regularly sampled images, without significant loss of speed.
Figure 4: The test image (left) and the results of the Sobel operator before (center) and after (right) the algorithm.

5 SELECTION state: Edge Extraction

Once a change is detected, we must decide if we want to fixate the target that originated the change, or simply ignore it. That decision should be based on both texture and dimension parameters. The latter is due mostly to the physical limits of the system. The former is crucial to the performance of the methods used during the FIXATION state (described below).

To have a fair idea of those parameters, we extract the target edges. Our goal is not to reconstruct the complete geometrical model, but only to obtain enough feature information for the FIXATION stage. Edge extracting can be devided into two major steps: Extracting edge pixels from the image and analytically define the edges using those pixels.

There are many known methods to perform both steps [7]. The Sobel operator is one of the most common pixel extracting technics, due to a low computational weight. The results generated by the Sobel operator are usually thick curves, closely related to the smoothness of the image edges, hence creating a problem for the second step of the edge detection task. To obtain thin edges, we use the following algorithm (the results are shown in figure 4):

- For all the pixels in the image, compare the gradient magnitude with the eight neighboring pixels. If three or more neighbors have greater or equal magnitudes, the pixel is not an edge.

We can now use the Hough transform, for example, to mathematically define the major edges of the target. As stated above, we are not interested in models, but in features like line intersections, that we may use as the actual fixation point.

6 FIXATION state: Matching

The traditional method to reach the fixation goal is matching. Matching can be defined in many ways, being the correlation of image areas one of them. The correlation is based on a function that weights the pixels in both areas against each other, returning a measure of their similarity. Fast measuring methods are the sum of square differences (SSD) and the sum of the modulus of the differences (SMD). Let \( l(x, y) \) and \( r(x, y) \) be the left and right cameras' images respectively and \( A \) the area we want to correlate, the equations used to implement those methods are shown below (low results indicate high similarity):

\[
C_{ssd} = \sum_{i,j \in A} (l(i, j) - r(i, j))^2
\]

\[
C_{smd} = \sum_{i,j \in A} |l(i, j) - r(i, j) |
\]

Without additional processing, those functions are very sensitive to images with different mean light intensity. Fortunately, we want to use them to compare images captured at the same instant in time. Hence, their intensity will be similar if the cameras have equal iris apertures.

Correlation methods are not suitable to compare areas without significant texture. This is why the SELECTION stage must filter out all the low texture targets. Another problem of using correlation to fixate appears when the target changes its orientation and/or position. The fixation algorithm must be prepared to cope with those changes and still keep fixating on the target. In other words, the matching process must be adaptive. The feature information extracted during the SELECTION stage will be useful here, because it will allow the implementation of a more intelligent matching scheme.

7 FIXATION state: Optical Flow

Although matching proved to be the best solution for the FIXATION state, applying the correlation function to the entire image, even when using fast measuring methods, results in a serious computational burden. To increase the processing speed, we must reduce the image areas where the target will be searched. To do that, we compute the movement that occurred at the previous target location, and search for the target, in the current image, only in the direction of the detected movement.

Direct extraction of movement from images is usually accomplished using optical flow technics [3]. They can be used simply to find movements in an image, or to evaluate those movements in terms of the relative velocity field [4].

Common movement extraction technics assume very small shifts of the target in the images. In our case, we cannot make that assumption. The solution for this problem is to compute the movement at lower image resolutions, hence increasing the allowed image shift of the target. Recalling that our objective is the reduction of the search areas, we do not need correct measurements of the image velocity field, but only a fair estimation of the moving direction of the target.
Also, we want to obtain the movement of the target, not all the movements in the image. So, only a limited area around the previous position of the target is selected for movement detection purposes. Of course, this also results in the increase of the computation speed.

8 Structure Recovering

We have now introduced fast computational approaches for the three fixation states described. Before moving on to the complete control strategy, a simple application of the algorithm is shown. Using only the matching and edge extraction steps, we will help a human operator to recover the 3D structure of an object. It is, therefore, a semi-autonomous application.

Let's start with the mathematical model of the active vision system (see figure 5). There are five referentials in the model:

- \([W]\) - The world, located at the center of the bottom of the system.
- \([P]\) - The pan, coincident with the world referential when the pan angle is zero.
- \([T]\) - The tilt, located over the vertical rotation axis \((Z_{[P]}).\)
- \([R]\) and \([L]\) - The right and left cameras, located over the \(ZX\) plane of the tilt referential.

To transform a point \([X \ Y \ Z]^T\) across the five referentials, we can use the following relations (the physical dimensions are in centimeters):

\[
\begin{bmatrix}
X' \\
Y' \\
Z'
\end{bmatrix}
= \begin{bmatrix}
\cos(\theta_P) & -\sin(\theta_P) & 0 \\
\sin(\theta_P) & \cos(\theta_P) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}
+ \begin{bmatrix}
14.55 \\
0 \\
20.6
\end{bmatrix}
\tag{4}
\]

\[
\begin{bmatrix}
X' \\
Y' \\
Z'
\end{bmatrix}
= \begin{bmatrix}
\cos(\theta_L) & -\sin(\theta_L) & 0 \\
\sin(\theta_L) & \cos(\theta_L) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}
+ \begin{bmatrix}
-14.55 \\
0 \\
20.6
\end{bmatrix}
\tag{5}
\]

\[
\begin{bmatrix}
X' \\
Y' \\
Z'
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(\beta) & -\sin(\beta) \\
0 & \sin(\beta) & \cos(\beta)
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}
+ \begin{bmatrix}
0 \\
0 \\
25.7
\end{bmatrix}
\tag{6}
\]

\[
\begin{bmatrix}
X' \\
Y' \\
Z'
\end{bmatrix}
= \begin{bmatrix}
\cos(\alpha) & -\sin(\alpha) & 0 \\
\sin(\alpha) & \cos(\alpha) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}
+ \begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix}
\tag{7}
\]

With that, we have all that we need to implement the structure recovering task. As stated, it is a very simple application. It works through the following steps (see figure 6):

1. To define each 3D point, the active vision system is guided by a human operator using a track-ball. The operator is expected to control the system until it is approximately fixated on a selected point of the object. This is the non-automatic step of the task.

2. Once positioned, the system uses the edges and matching information to, under an allowed degree of liberty, adjust its orientation and correctly fixate the point. In practice, no physical adjustments are made. Only image processing operations are involved.

3. Finally, a 3D point is estimated and added to the overall model. Of course, the resulting model depends on the number of points defined. Figure 7 shows a PC monitor defined with eight points.

9 Object Tracking

Tracking and fixating processes are closely related. Although our primary objective is to fixate the target,
we want to be able to control the active vision system in a way that, if the target changes its position in space, we may continue to fixate it. If we are able to do that, we will be tracking the target.

Some authors prefer position based tracking methods, while others use velocity based technics. In our case, the active vision system forces us to perform position based tracking, since the available controllers do not allow for on-the-fly control. Hence the dominant matching/correlation method in the FIXATION state. It is possible that the optimal control solution, if any, is formed by both position and velocity information. That assumption originated the introduction of optical flow information in the algorithm, even if just to improve the matching process performance.

The complete tracking process is shown in Figure 8 and is described by the following steps:

- **REST state:**
  1. The camera’s images are transformed to the log-polar plane.
  2. The images are subjected to a movement analysis (image difference, for example).
  3. Depending on the location of the detected movement, the active vision system may perform a saccade movement.

- **SELECTION state:**
  1. Images’ edges are extracted and a mathematical formulation of the features is computed.
  2. If there is no suitable fixation point, the system returns to the REST state.
  3. If the target’s features are not equivalent on both the left and the right images, the system returns to the REST state. This may be decided using correlation methods.

- **FIXATION state:**
  1. An estimation of the target movement is computed between the previous and the current images.
  2. Matching methods are used to locate the current position of the target. The correlation is the mandatory method, but may be complemented by an edge detection scheme similar to the one described for the SELECTION state. This allows the algorithm to compensate for possible changes in the target’s orientation, distance, etc.
  3. A 3D point in world coordinates is computed from the estimated position, using the active vision system model.
  4. The vergence angle (position of the cameras) is immediately updated, attempting to center the target in both images. The motors driving the cameras are the fastest and are given priority by the control scheme.
  5. The pan and tilt motors are controlled in
order to keep the head setup facing the target. The idea is to compensate the movement made by the vergence motors, allowing them enough degree of freedom to track an escaping target.

This ends the description of the overall control scheme. The major differences between the method described here and the original method [2] are:

- The use of specific image motion analysis. The introduction of optical flow techniques helped to speed up the computational tasks.
- Careful target selection and matching. The cooperation between pure pattern matching and feature extraction improved the robustness of the system.
- Fast computational algorithms. The use of better hardware resulted in higher sampling frequencies and consequent performance improvements.

10 Conclusion

This paper described a practical approach to a robust visual fixation algorithm. The same set of image processing operations (log-polar mapping, edge extraction, matching and optical flow) can be used to solve tasks like 3D structure recovering (using matching and edge extraction), movement detection (using log-polar mapping) or object tracking (using all the operations). All the suggested image processing methods and algorithms were implemented and tested in real-time conditions.

References


