Universidade de Aveiro
2007

Departamento de Engenharia Mecânica

Miguel Armando Riem de Oliveira

Desenvolvimento de um Sistema de Visão Foveada para o Seguimento de Alvos Móveis em Ambientes Dinâmicos

Development of a Foveated Vision System for the Tracking of Mobile Targets in Dynamic Environments

Universidade de Aveiro
2005

Departamento de Engenharia Mecânica

Miguel Armando Riem de Oliveira

Desenvolvimento de um Sistema de Visão Foveada para o Seguimento de Alvos Móveis em Ambientes Dinâmicos

Development of a Foveated Vision System for the Tracking of Mobile Targets in Dynamic Environments

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob a orientação científica do Dr. Vitor Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro

Obrigado Anita e Laura

o júri

presidente

Prof. Dr. José Joaquim de Almeida Grácio
professor catedrático da Universidade de Aveiro

Prof. Dr. Armando José Formoso de Pinho
professor associado da Universidade de Aveiro

Prof. Dr. Vitor Manuel Ferreira dos Santos
professor associado da Universidade de Aveiro

agradecimentos

Os meus agradecimentos ao Professor Vitor Santos pelo apoio incansável e motivação constante, e também pela compreensão e amizade.

Também aos meus companheiros de estudo noctívagos, Rui Cancela e David Gameiro, sem os quais não teria conseguido terminar.

E também aos companheiros diurnos, Eduardo Durana, Milton Ruas e Caetano Ferreira.

À Ana e à Laura, o meu obrigado por tudo, simplesmente tudo.

À minha familia também o meu muito obrigado.

palavras-chave

Visão por Computador, Percepção Activa, Visão Foveada, Controlo da Foveação, *Haar Features*, Busca Visual, Mecanismos de Atenção, Seguimento Visual, Filtro Gaussiano e Computação Rápida de Filtro Gaussiano.

resumo

Este trabalho descreve um sistema baseado em percepção activa e em visão foveada, projectado para identificar e seguir objectos móveis em ambientes dinâmicos. O sistema inclui uma unidade *pan & tilt* para facilitar o seguimento e manter o objecto no centro do campo visual das câmaras, cujas lentes grande-angular e tele-objectiva proporcionam uma visão periférica e foveada do mundo, respectivamente. O método *Haar features* é utilizado para efectuar o reconhecimento dos objectos. O algoritmo de seguimento baseado em *template matching* continua a perseguir o objecto mesmo quando este não mais está a ser reconhecido pelo módulo de identificação. Algumas técnicas utilizadas para melhorar o *template matching* são também apresentadas, nomeadamente o Filtro Gaussiano e a Computação Rápida de Filtro Gaussiano. São indicados resultados relativos ao seguimento, identificação e desempenho global do sistema. O sistema comporta-se muito bem, mantendo o processamento de, pelo menos, 15 fotogramas por segundo em imagens de 320x240, num computador portátil normal. São também abordados alguns aspectos para melhorar o desempenho do sistema.

keywords

Computer Vision, Active Perception, Foveated Vision, Foveation Control, Haar Features, Visual Search, Attention Mechanisms, Visual Tracking, Gaussian Filtering and Fast Gaussian Computation.

abstract

This work describes a system based on active perception and foveated vision, intended to identify and track moving targets in dynamic environments. The full system includes a pan and tilt unit to ease tracking and keep the interesting target in the two cameras' view, whose wide / narrow field lenses provide both a peripheral and a foveal view of the world respectively. View-based Haar-like features are employed for object recognition. A template matching based tracking technique continues to track the object even when its view is not recognized by the object recognition module. Some of the techniques used to improve the template matching performance are also presented, namely Gaussian Filtering and Fast Gaussian computation. Results are presented for tracking, identification and global system's operation. The system performs well up to 15 frames per second on a 320 x 240 image on an ordinary laptop computer. Several issues to improve the system's performance are also addressed.

Symbols and Acronyms Convention

(By order of appearance)

DOF(s)	Degrees of freedom.
$MC\left(\vec{x}\right)$	Mass center.
M_{line}	Map of line values of the image's pixels.
M_{col}	Map of column values of the image's pixels.
$\otimes$	Pixel by pixel matrix multiplication.
OpenCV	Open Source Computer Vision Library.
T_c	Macro for template used for comparison.
$T_{(x,y)}$	Intensity values of template T_c.
w	Width of T_c.
h	height of T_c.
$\vec{r}$	Image based pixel coordinates' T_c's upper left corner position.
$I(x,y,t)$	Image's x, y pixel value at instant t.
$\delta = (\varepsilon, \eta)$	Pixel / feature displacement.
D	Deformation matrix.
$\mathbf{I}$	Identity matrix.
W	Width of Image.
H	Height of Image.
$R(x,y)$	Result of template matching operation.
$\mathrm{Re}cSum(x,y,w,h)$	Haar feature sub window.
$SAT(x,y)$	Summed Area Table.
TMR	Template matching results.
$P(\vec{r}^{N} \mid \vec{r}^{N-1})$	Probability of the template being at position $\vec{r}^{N}$ given that it was at $\vec{r}^{N-1}$.
$G(a,b)$	Gaussian matrix calculation.
GM	Gaussian matrix.
EGM	Extended Gaussian matrix.

PTU	Pan and Tilt Unit.
fps	Frames per second.
$\vec{v}$	Current object coordinates in the peripheral image.
$\hat{\vec{v}}$	Desired object coordinates in the peripheral image.
P_{error} / T_{error}	Pan / tilt error.
$P_{position} / T_{position}$	Pan / tilt position.
P_{speed} / T_{speed}	Pan / tilt speed.
α_p / α_f	Peripheral / foveated horizontal axis scaling factor.
β_p / β_f	Peripheral / foveated vertical axis scaling factor.
M / m	3D / 2D point.
$\tilde{M} / \tilde{m}$	3D / 2D point in homogeneous coordinates.
R / t	Rotation and translation extrinsic camera parameters' matrixes.
A	Intrinsic camera parameters' matrix.
γ	Camera's skewness intrinsic parameter.
x_0, y_0	Camera's principal point coordinates.
$\hat{R} / \hat{t}$	R / t transformation from foveated to peripheral coordinates.
Z	Depth or object's distance from camera.
TDS(i)	Training dataset i.
PDS(i)	Performance dataset i.
sf	Haar detection scaling factor.
$Ci_{sf=j}$	Cascade i with scaling factor j.
OCR	Optical character recognition.

Table of Contents

List of Figures

List of Tables

1 Introduction

"Recent years have witnessed a dramatic increase in the use of computer vision in embedded systems. Computer vision was successfully used in various mission-critical systems from the landing of the recent rovers on mars to computer-aided surgery". This citation was found in the call for papers of the first IEEE workshop on Embedded Computer Vision 2005. It clearly shows how computer vision systems are swiftly gaining importance in the world today. The pool of applications is immense, ranging from medical imaging to the robots that travel to Mars, passing through industry applications and autonomous vehicles' guidance.

This work intends to develop an active foveated vision unit, study the techniques used to solve the problems that this technology carries and to find possible innovative applications for such a system. Active vision concerns of using visual feedback to execute servo motor control, which enables directing the cameras towards particular areas of attention. In this case we have employed a pan and tilt servo controlled unit to provide for active vision capabilities. Foveated vision was implemented by means of two, different focal length, cameras that provided both a peripheral and a foveated view of the world.

The work comprehended the following stages:

a) An analysis of the current state of the art in the technologies that were employed.
b) The development of low level software for hardware interaction and basic image processing algorithms.
c) Development of a controller that translates a visual target position into position and speed values to the pan and tilt.
d) The development of algorithms that could handle the visual search problematic, i.e., deployment of attention techniques.
e) The development of visual tracking algorithms.
f) The applications of state-of-the-art object recognition techniques, namely Haar-like features.
g) The integration of the previous items into a modular software architecture that could identify and track moving objects in dynamic environments.

Some of the results here presented were already published in [Oliveira and Santos, 2007].

2 The State of the Art

There are many applications using computer vision, and so it is not possible to provide a complete overview of the whole computer vision problematic and applications. This project is based on active vision. Active vision concerns not only processing the information provided by the cameras, but also on making use of that information to direct the cameras' view towards interesting features in the visual field. This technology is also used by biological vision systems, like, for example, human.

In fact, human eyes have a multitude of movements. According to [Laschi, 2006] there are mainly five types of eye movements: saccades, vergence, pursuit, vestibulo-ocular reflex and opto-kinetic response. Saccades are ballistic eye movements, i.e., cannot be interrupted or redirected while they occur. They are very fast, up to 700 degrees per second, and last from 30 to 120 milliseconds, and are used to move the fovea to the next object or region of interest. They present a latency of around 150 milliseconds. Vergence changes the angle between the eyes viewing axes. It is a slow movement that can take up to a second and can be interrupted while on course. Pursuit is a smooth eye movement that enables visual tracking. Obviously, a human vision system possesses extra redundancy, namely because of the combination of head and eye movements. However, head movements cannot be entirely dedicated to visual analysis/tracking. For example, the head may turn in order to better locate the source of a sound. Visual processing is maintained during head turns by means of the vestibulo-ocular reflex. This reflex stabilizes image on the retina during head movement by producing an opposite direction eye movement. Vestibulo-ocular reflex's latency is around 14 milliseconds and is performed without visual feedback (it occurs even in total darkness). The opto-kinetic response allows the eyes to track objects in motion while the head remains still. They make use of visual feedback and therefore have a larger latency when compared to vestibulo-ocular reflex. There are some other secondary eye movements, as pointed out by [Laschi, 2006]. Torsional eye movements rotate the eye around the viewing axis and are employed to compensate for body rotation. They present a range of 15 degrees. Fixations are alternated with saccades and keep the eye motionless. They typically last from 200 to 600 milliseconds and allow all of the scene information to be processed (used, for example, to read). As seen,

there is a wide variety of eye movements with multiple purposes, i.e. tracking, foveation, quick shift of the object of attention or compensating other degrees of freedom (DOFs). This endorses the idea that active vision has evolved in biological organisms to a high degree of complexity.

In robotics applications, active vision is only now starting to spread. It has been implemented in many humanoid robots, such as the ARMAR III, from Universität Karlsruhe, in Germany, that presents 4 DOFs at the neck combined with 2 more DOFs at the eyes. Altogether, 6 DOFs provide a redundant mechanism which is being employed to mimic natural human motions [Asfour *et. al*., 2007][Azad *et. al*., 2007].

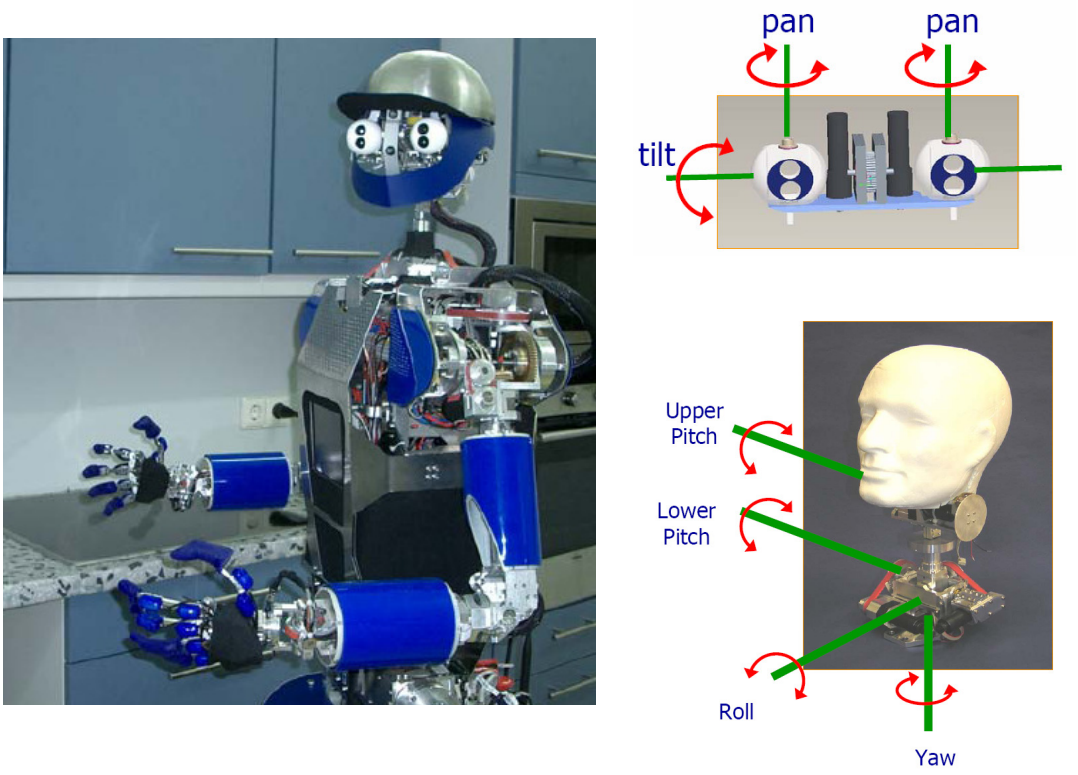

Figure 1. ARMAR III humanoid robot (left) active vision mechanisms. 4 DOFs at the neck (bellow right) plus 2DOFs at the eyes (top right). All pictures taken from [Asfour, 2006].

Italy's Scuola Superiore Sant'Anna also proposes a robotic head mechanism, shown on Figure 2. They have successfully mimicked human saccadic eye movements and proposed a way of dealing with the neck and eyes redundancy [Laschi, 2006].

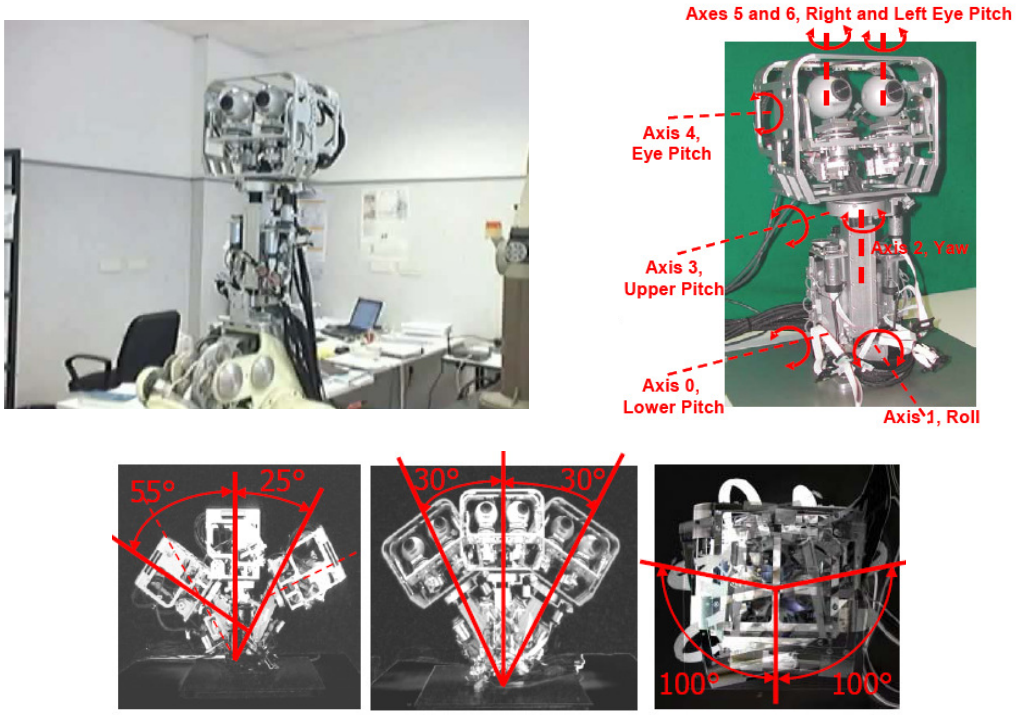

Figure 2. SSSA Robot Head (top left) and axes indication (top right and bottom). Pictures taken from [Laschi, 2006].

In Japan, there are also some examples of active perception (Figure 3).

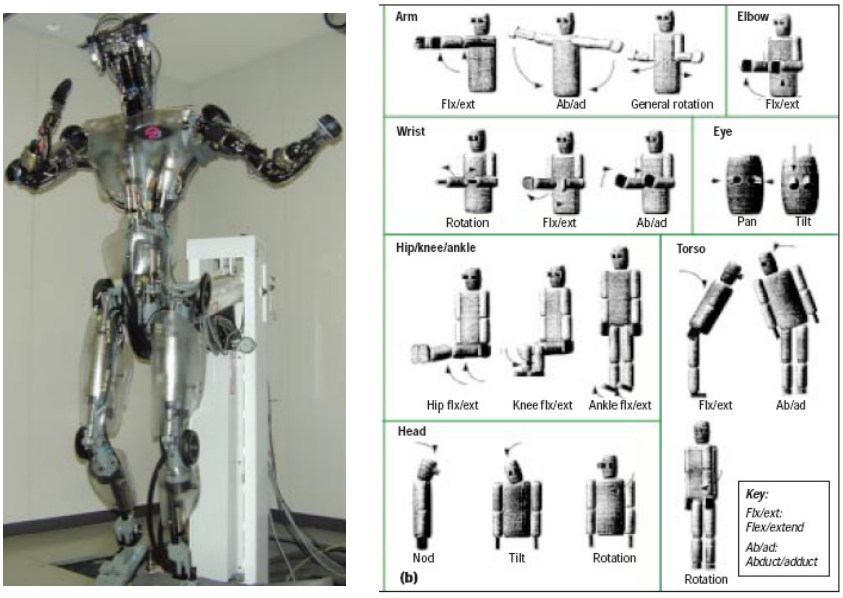

Figure 3. DB humanoid robot full body (left) and robot's joints (right). Notice the 5 combined DOFs per eye. All pictures taken from [Atkeson *et. al.,* 2000].

Japan's ATR Computational Neuroscience Laboratories humanoid robot DB, i.e. Dynamic Brain, also uses redundant degrees of freedom to actively recognize and pursue objects of interest. Another DB project uses active vision to capture and learn movement sequences from observing humans [Ude *et. al*., 2004 a].

In the United States, MIT's COG project, a human-like robotic torso was developed to "study theories of cognitive science and artificial intelligence.", by creating a "robot which is capable of interacting with the world — including both objects and people — in a human-like way, so that we may study human intelligence by trying to implement it." [COG homepage]. It also takes advantage of active perception by making use of its 7DOFs from the neck up.

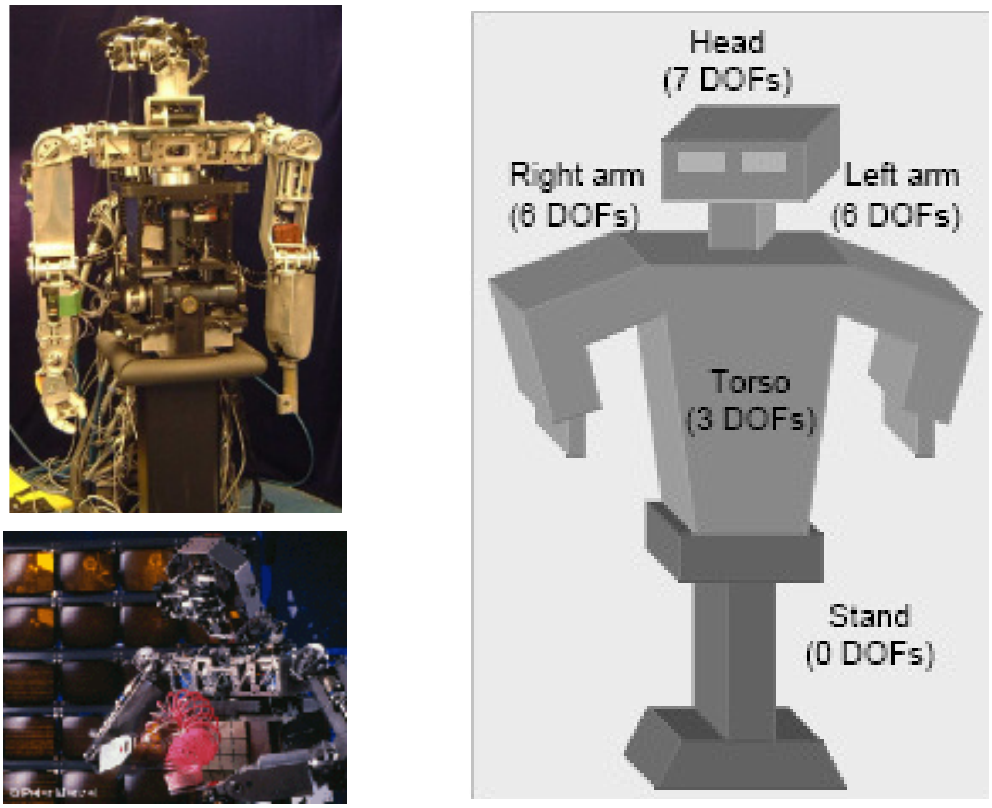

Figure 4. Full view of COG (top left) and respective DOFs (right), both from [Fitzpatrick *et. al.*, 2003].COG interacting with the world (bottom left), taken from [COG homepage].

As seen before, humanoid robots make use of state of the art active vision systems to perceive the world and act accordingly. However, active vision is used by many other robotic applications, like autonomous guided vehicles. Out of these, the most advanced would be the Mars Exploration Rovers built by NASA's Jet Propulsion Laboratory. These Rovers have 8

onboard cameras. Two of them, called *Science Pancams*, are mounted on top of a mast, which in turn is an actuated device with 2 DOFs, therefore providing pan and tilt movement capabilities to the visual sensors. These cameras carry a multiple filters wheel, allowing multispectral imaging capabilities. One of these filters is a color solar-imaging filter, which, combined with the active vision capabilities, allows the Rover to point the cameras at the Sun. This, combined with the time of day, enables the Rover to have an absolute heading sensor… in Mars.

Figure 5. NASA's Mars Exploration Rover (left) also makes use of active perception. The mast (right), where two high resolution cameras are mounted, rotates and tilts. Pictures taken from [MERs homepage].

It seems clear that many of the most advanced robots in the world are using vision. Furthermore they are employing active vision, by mounting the cameras on some kind of actuated platform. Of course that this sensory configuration accounts for some new, more difficult issues that must be attended, but the key insight here is that an active perception system is much more powerful, despite being obviously more complex, than a rigidly mounted system.

Another core technology that is being employed in this project concerns the use of foveated vision. Foveated vision, from the term fovea, the central highest resolution region of the human eye's retina, explores image resolution disparities in order to process visual information in real time. Computer vision demands high computational power because the amount of raw information, i.e. the pixel's intensity values across multiple channels, in an

image is quite large, but also due to the virtually infinite amount of covert, but deducible, information present. Real time applications have limited time to spare to visual processing before an action becomes mandatory. Limited time implies limited processing or, on the other hand, focalized processing, i.e. processing of the image regions that are in fact important, neglecting others. This decision making process of selection is called attention. [Rothenstein *et. al*., 2006] also highlight the difficulties of real time visual processing: "Those who build complete vision application systems invoke attentional mechanisms because they must confront and defeat the computational load in order to achieve the goal of real-time processing (…). But the mainstream of computer vision does not give attention processes, especially task-directed attention, much consideration"

In real applications, the foveal area of a given image region must have a high resolution, while, for real time demands, the rest of the image should have a coarse resolution. This can be achieved by one of three distinct manners, either a) the cameras provide very high resolution images overall, which is then downgraded at non foveal regions or b) the cameras have varying resolution CCDs or c) two cameras per visual element are assembled and used as a single visual perception entity. In this last case, the two cameras must have different lenses, one with a wide lens and the other with a narrow field lens, in order to provide both a peripheral and a foveal view of the world.

The first approach (a) is not very common one since that, by applying a posterior resolution downgrade, it enforces the acquisition of a very high resolution image even in the regions of the image where this is not desired, only to discard them with a log-polar transformation soon after. However, Scuola Superiore Sant'Anna has made several tests with these techniques, mainly to study digital conversion of a standard image to a foveal image by using a log-polar transformation.

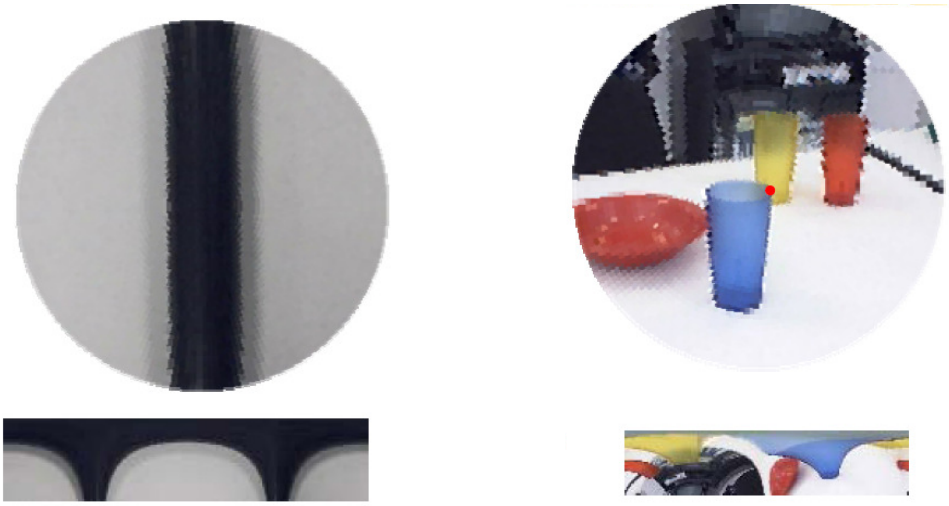

Figure 6. Foveal images (top) obtained by transforming standard images using the log-polar transformation (down). Pictures taken from [Laschi, 2006].

The second approach (b) is based on retina-like Giotto cameras. Giotto cameras use a circular geometry CCD whose resolution is maximum at the center of the image and decreases towards the periphery. These cameras are implemented in the WE-4 robotic head of the Takanishi Lab, Waseda, Japan.

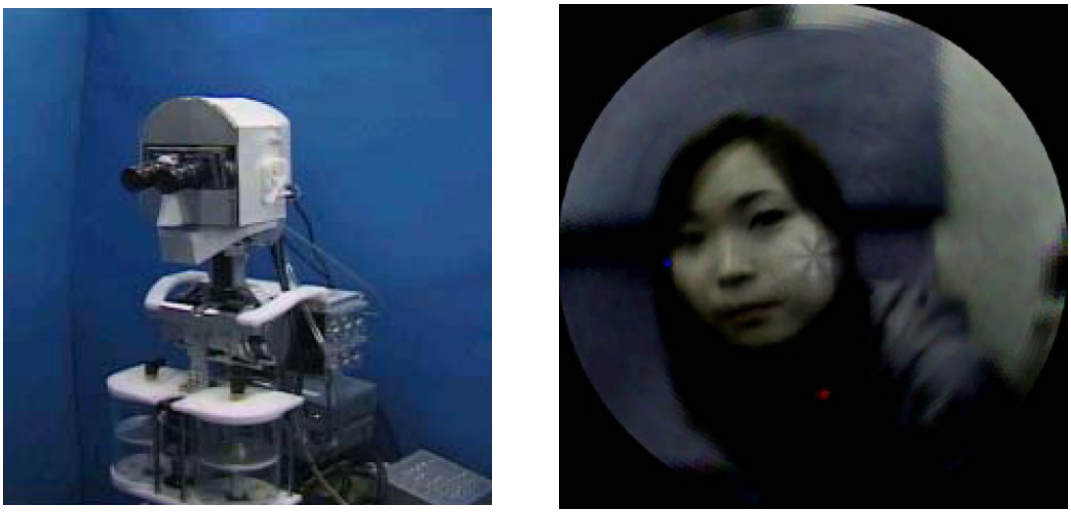

Figure 7. WE-4 robotic head with two foveal resolution Giotto cameras (left) and a view of those cameras (right). Pictures taken from [Laschi, 2006].

The third approach (c) dictates the use of a pair of cameras per visual sensing unit. It is the most commonly used approach. [Ude *et. al*., 2006] have chosen this alternative "While log-

polar-sensors and lenses with space-variant resolution are conceptually appealing, they are difficult to construct and prevent us from using high-quality standard cameras and lenses. High-definition cameras are problematic for real-time humanoid vision because of the form factor and bandwidth. We therefore follow the first approach and use two cameras per eye to realize foveation." [Ude et. al., 2006].

Figure 8. Two cameras per eye DB humanoid robot's peripheral (left) and foveal (right) view. Pictures taken from [Ude et. al., 2006].

In fact, many researchers have applied this technique to solve the foveal view problem (Figure 9).

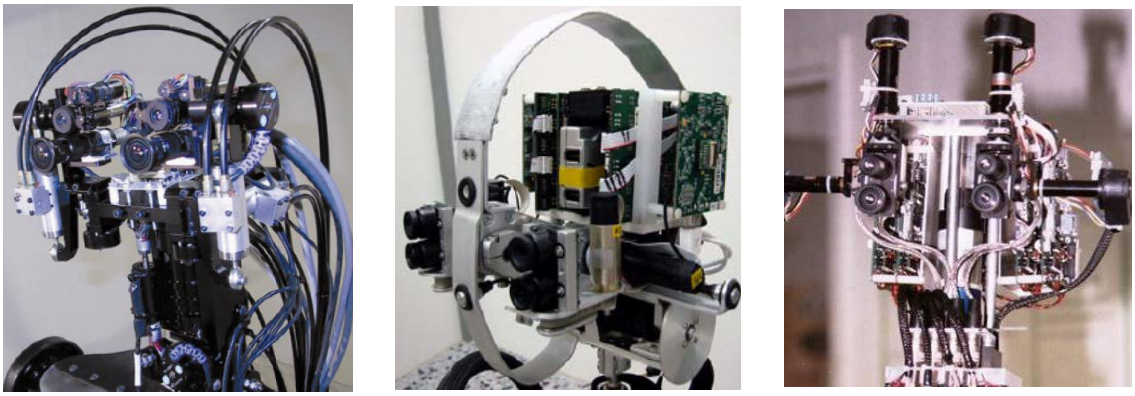

Figure 9. Dual camera foveation systems. DB (left, from [Ude et. al., 2006]), ARMAR III, (middle, from 1 [Asfour, 2006]) and COG (right, from [COG homepage]).

This chapter reviewed state of the art technologies used to employ active foveated vision. Based on the analysis exposed, it was decided to mount a dual camera foveation setup on a pan and tilt servo controlled unit. The following chapters will focus in detail the problems that arise from this particular setup and the techniques that were employed to solve them.

3 Visual Search

Visual Search is a mechanism that occurs during the pre-attentive stage of a vision system and is responsible for the selection of particular features for posterior in depth processing. Studies of biological vision systems suggest that only a small set of basic features like color, size, motion, and orientation are used in a visual search. Also, pre-attentive processing (where attention plays its role) produces reaction times that are somewhat independent of the amount of objects in display, as stated in [Wolfe, 2003]. This weak or inexistent correlation between the amount of features and the reaction times in pre-attentive processing seems to imply that they "… only use information about the categorical status of items. In orientation, that means that it is only easy to find a target if it is uniquely steep, shallow, tilted left or right" [Wolfe, 2003]. In Figure 10 four images are presented. Trying to find a vertical line seems trivial for (a) and (b) whereas the same operation is quite more demanding in (c) and (d).

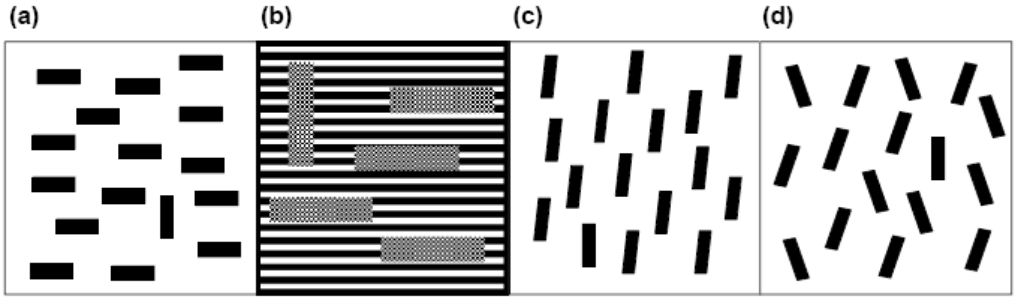

Figure 10. Pre-attentive recognition of a vertical line is easy when its orientation is unique [(a) and (b)] but pre-attentive orientation detection's limitations are brought up if the features orientations are similar [(c) and (d)]. Taken from [Wolfe, 2003].

Another example also present in [Wolfe, 2003] uses pre-attentive color recognition to justify why finding the molecule present in Figure 11 is much easier in (a) than in (b).

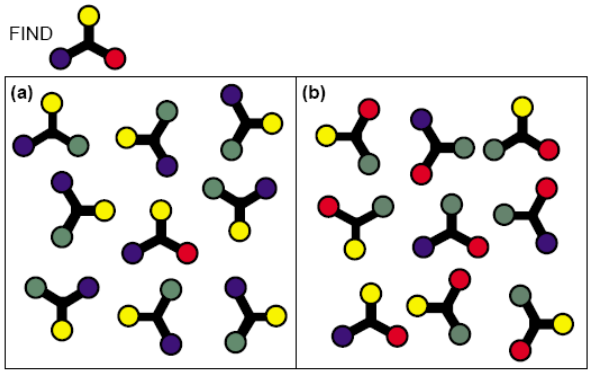

Figure 11. Pre-attentive color recognition allows a much faster finding of the molecule in (a) than in (b) because the red color is unique in (a). Taken from [Wolfe, 2003].

[Ude *et. al*., 2004 b] provide a general description of the visual search phenomenon's implementation issues by stating that "it does not make sense to implement complex search schemes when a humanoid robot looks for a particular feature or object in an unknown dynamic and cluttered environment". Given these clues of the attention mechanisms involved in biological vision systems, it makes sense to attempt one or several simple implementations of attention capturing mechanisms. These should deploy the systems attention to a particular object by foveating it. This attention capturing mechanisms must be simple (fast to compute) and rely on object properties that may be detected without a foveal, high resolution view, of the object. These mechanisms are meant to be used when the system is searching for a known object to track. This chapter intends to describe some simple algorithms that play the role of attention mechanisms and have already been implemented and tested for this purpose. They will be described in detail in the following chapters.

3.1 Color Detection

Simple color recognition may work as a good attention mechanism. [Ude *et. al*., 2004 b] use it as signal detectors that deploy attention on a particular image blob. The object to be followed can be physically tagged with color markers and these will capture the system's attention. Alternatively, the object's dominant color can be set as one worthy of attention. Color recognition is performed in the HSV color space which suits perfectly for color recognition since, unlike RGB, it separates color from light intensity and saturation (Figure 12).

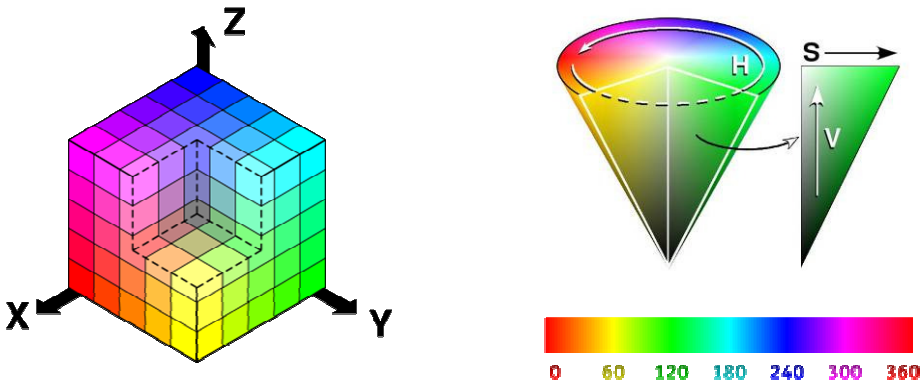

Figure 12. Graphical representations of RGB (left) and HSV (right) color models. Unlike RGB, HSV isolates color information on one single channel (Hue) and is therefore easier to use for color detection. Images taken from Wikipedia *(www.wikipedia.com).*

Therefore, the simplest approach might be to filter all the pixels with Hue values between some pre-defined limits. Using a mask built with these conditions, the pixels with the desired color are filtered.

$$\begin{aligned} &if\,(\min_{Hue} > src_{Hue} > \max_{Hue}) \Rightarrow mask_{val} = 1 \\ &else \Rightarrow mask_{val} = 0 \end{aligned} \qquad (1)$$

Where src_{Hue} is the Hue value for a given pixel and $mask_{val}$ is the value of the Boolean mask that is being built and indeed holds the object being segmented. The undeniable advantage of this method, its simplicity, is on the other hand contradicted by the fact that min_{Hue} and max_{Hue} values have to be precisely tuned. A very restricted interval may discard some of the object's pixels, while a large, undemanding interval fails to filter background noise and does not entirely segment the object. A balance can be achieved if post processing is used to attempt to extract the remaining background pixels with some other criteria. Upon the application of an additional filter to remove isolated pixels, this mechanism usually achieves satisfactory results. Another possibility is to use pixel connectivity to separate the image into several spots and then select the spot with the largest amount of pixels. After some experiments, it was found that the final solution is to find adequate values for max_{Hue} and min_{Hue}, perform an isolated pixel filtering and, finally, calculate the filtered pixels' mass center, i.e. the object's centroid. Mass center works well because the weight of the object's pixels (being far more than the rest, assuming that the color being filtered appears only in the object) pulls it to the object center. The mass center is calculated using the following expression:

$$MC\left(\vec{x}\right)=\begin{bmatrix}\dfrac{\sum_{c=0}^{c=w-1}\sum_{l=0}^{l=h-1} mask_{val}\left(l,c\right)\times l}{n}\\ \dfrac{\sum_{c=0}^{c=w-1}\sum_{l=0}^{l=h-1} mask_{val}\left(l,c\right)\times c}{n}\end{bmatrix} \tag{2}$$

Where *MC* is the vector representing the mass center's coordinates, *l* is the line, *c* the column and *n* represents the total number of filtered pixels.

OpenCV's functions to access image pixels (*cvGet2D*) are quite slow [OpenCV documentation] and only unrestricted pointer based access to pixels is very fast (about 10 times faster than *cvGet2D*). To avoid free pointer access and to keep software structure it is preferable instead to use OpenCV's built-in functions to perform mass center calculation, for which a method has been devised. Two matrices with the same size of the image are defined, where the first, called M_{line}, has for each pixel its corresponding line, and the second, M_{column}, has for each pixel the column index of its location, defining what is frequently called a mesh grid.

$$M_{line}=\begin{bmatrix}0 & 0 & \dots & 0\\ 1 & 1 & \dots & 1\\ \dots & \dots & \dots & \dots\\ h-1 & h-1 & \dots & h-1\end{bmatrix} \qquad M_{column}=\begin{bmatrix}0 & 1 & \dots & w-1\\ 0 & 1 & \dots & w-1\\ \dots & \dots & \dots & \dots\\ 0 & 1 & \dots & w-1\end{bmatrix} \tag{3}$$

Both these matrices are constant for every image size and are therefore created only once at the beginning of the program. No real time computational power is spent in this calculation. To get *MC*, it is simply necessary to sum the pixels for every mesh grid, using the color mask as a conditioner for the operation, and divide the final result by *n*, the total number of pixels of the object ($n=\sum\sum mask_{val}$):

$$MC\left(\vec{x}\right)=\frac{\begin{bmatrix}\sum\sum\left(mask_{val}\otimes M_{line}\right)\\ \sum\sum\left(mask_{val}\otimes M_{column}\right)\end{bmatrix}}{n} \tag{4}$$

Where $\otimes$ is the symbol for pixel multiplication. This procedure avoids user retrieval of the original image information (safe access), using only OpenCV's functions, and so, boosting the time of *MC* calculation when compared to the common OpenCV's image data retrieval functions (8~9 times faster than if using *cvGet2D* with the method of equation (2)).

Regarding color segmentation, there are some operations that are far more complex and effective. Pyramid segmentation [OpenCV documentation] can be used to group pixels with color similarity and divide the image into a set of blobs. Each of these blobs represents a group of connected pixels whose color is similar. However, these methods are obviously more time costing and therefore are not used. An attention mechanism must be a simple and fast process, since it does not try to detect complex features but only some particular attention capturing properties.

Also, finding the mass center of a color filter is, of course, a somewhat statistical process. No high resolution or special details are mandatory. In fact, experiences with this method have proved that no noticeable difference arises from using 640x480 or conversely 320x240 images. Because of this, color detection has been the most widely used attention mechanism and is, at it most recent version, being calculated over 160x120, 3 channel images at a rate of about 4 milliseconds and with no noticeable loss of effectiveness. Of course that color detection highly depends on the camera's internal parameters such as white-balance, saturation and others. It is also sensitive to light conditions change.

3.2 Motion Detection

Another possible attention capturing property is based on movement detection. Motion detection techniques are now quite standard. Some of them have been implemented. Optical flow techniques try to find corresponding pixels (or features, for that matter) in two sequential frames. Using this correspondence a vector for the movement of each tracked pixel/feature can be obtained. [Shi and Tomasi, 1993] present two models for image motion. A given feature may change its position due to object displacement or to object deformation. The correlation of patterns present on two images at instants t and $t+\tau$ satisfies equation (5)

$$I(x, y, t+\tau) = I(x - \varepsilon(x, y, t, \tau), y - \eta(x, y, t, \tau)) \quad (5)$$

where $\delta = (\varepsilon, \eta)$ is the displacement. Equation's (5) law is often violated due to feature occlusion, light conditions change, or even light reflection change (reflection is a function of the viewpoint, that may change when an object is being tracked). However, [Shi and Tomasi, 1993] found that equation (5) is "by and large satisfied at surface markings that are away from occluding contours" and also that "at these locations, the image intensity changes fast with x

and y, and the location of this change remains well defined even in the presence of moderate variations of overall brightness around it". [Shi and Tomasi, 1993]. define affine motion field as

$$\delta = D(x, y) + d \tag{6}$$

where d is the translation of the feature's window center (or displacement vector) and D is the deformation matrix given by

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \tag{7}$$

If a given feature, at image I_t suffers no deformation on its path to image $I_{t+\tau}$ then D takes the form a 2x2 identity matrix $\mathbf{I}$. Hence one can state that

$$I_{t+\tau}\left((\mathbf{I} + D)\vec{x} + d\right) = I_t(\vec{x}) \tag{8}$$

Despite that smaller features make D harder to estimate (low variation in motion), they are still preferable for the reason that the likelihood of the feature containing pairs of patches at different depths and, thus, a depth discontinuity decreases with the decrease of the feature's window size. Because of this, [Shi and Tomasi, 1993], propose to use a pure translation model, neglecting the deformation. Therefore, equation (8) is approximated by

$$I_{t+\tau}\left(\mathbf{I}\vec{x} + d\right) = I_t(\vec{x}) \Leftrightarrow I_{t+\tau}\left(\vec{x} + d\right) = I_t(\vec{x}) \tag{9}$$

According to this formulation, [Shi and Tomasi, 1993], define tracking as "determining the six parameters that appear in deformation matrix D (neglected in this case) and displacement vector d" that best minimize the dissimilarity ε

$$\varepsilon = \int\int_W \left[I_{t+\tau}(\vec{x} + d) - I_t(\vec{x})\right]^2 w(\vec{x}) d\vec{x} \tag{10}$$

The weight $w(\vec{x})$ may be, in the simplest case, equal to 1, or it may be a function of $\vec{x}$ in order to, for example, emphasize the center of the feature window. Usually a Gaussian-like function is used for this. To minimize the dissimilarity ε, we set equation (10)'s differential to zero

$$\frac{\partial \varepsilon}{\partial d} = \int\int_W 2 \times \left[I_{t+\tau}(\vec{x} + d) - I_t(\vec{x})\right] G \times w(\vec{x}) d\vec{x} = 0 \tag{11}$$

Where $G = [\partial I_{t+\tau} / \partial x, \partial I_{t+\tau} / \partial y]^T$ is the spatial gradient of the image's intensity. Rearranging equation (11)

$$\frac{\partial \varepsilon}{2\partial d} = \int\int_W \left[I_{t+\tau}(\vec{x}+d) - I_t(\vec{x}) \right] G \times w(\vec{x}) d\vec{x} = 0 \tag{12}$$

Assuming that the image motion is small, the term $I_{t+\tau}(\vec{x}+d)$ can be approximated by its Taylor series truncated to the linear term

$$I_{t+\tau}(\vec{x}+d) = \frac{f^0\left(I_{t+\tau}(\vec{x})\right)}{0!} \times \left((\vec{x}+d)-\vec{x}\right)^0 + \frac{f^1\left(I_{t+\tau}(\vec{x})\right)}{1!} \times \left((\vec{x}+d)-\vec{x}\right)^1 \tag{13}$$

which results in

$$I_{t+\tau}(\vec{x}+d) = I_{t+\tau}(\vec{x}) + G^T \times d \tag{14}$$

Substituting in equation (12) the result of equation (14) we obtain the following equation

$$\begin{aligned} &\int\int_W \left[I_{t+\tau}(\vec{x}) + G^T \times d - I_t(\vec{x}) \right] G \times w(\vec{x}) d\vec{x} = 0 \Leftrightarrow \\ &\Leftrightarrow \int\int_W \left[I_{t+\tau}(\vec{x}) - I_t(\vec{x}) \right] G \times w(\vec{x}) d\vec{x} = \int\int_W \left[G^T \times d \right] G \times w(\vec{x}) d\vec{x} \end{aligned} \tag{15}$$

Expanding equation (15)

$$\begin{aligned} &\int\int_W \left[I_{t+\tau}(\vec{x}) - I_t(\vec{x}) \right] \begin{bmatrix} G_x \\ G_y \end{bmatrix} \times w(\vec{x}) d\vec{x} = \int\int_W \begin{bmatrix} G_x \\ G_y \end{bmatrix} \left[\begin{bmatrix} G_x & G_y \end{bmatrix} \times \begin{bmatrix} d_x \\ d_y \end{bmatrix} \right] \times w(\vec{x}) d\vec{x} \Leftrightarrow \\ &\Leftrightarrow \int\int_W \begin{bmatrix} G_x^2 & G_{xy} \\ G_{xy} & G_y^2 \end{bmatrix} w(\vec{x}) dx \times \begin{bmatrix} d_x \\ d_y \end{bmatrix} = \int\int_W \left[I_{t+\tau}(\vec{x}) - I_t(\vec{x}) \right] \begin{bmatrix} G_x \\ G_y \end{bmatrix} \times w(\vec{x}) d\vec{x} \end{aligned} \tag{16}$$

[Shi and Tomasi, 1993] have formulated the problem of tracking a moving object in equation (16). They have also defined a criteria for which are be the best features to track based on eigenvalues. Optical flow detection implementation is mainly based on [Stavens, 2007], that formalizes optical flow as the attempt to find the minimum $\varepsilon(\delta_x, \delta_y)$ given by

$$\varepsilon(\delta_x, \delta_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} \left(I_{n-1}(x,y) - I_n(x+\delta_x, y+\delta_y) \right) \tag{17}$$

This is the Lucas-Kanade formulation which assumes that the optical flow is constant in a small image region delimited by the w_x and w_y in order to overcome the aperture problem. The question of which are the best pixels/features to track was approached with the Shi-Tomasi method that considers the best features to be the ones with the highest eigenvalues. There is also the possibility of matching only a sparse set of features instead of all of the image's pixels, hence using a smaller amount of tracking data [Bouget, unknown] [Stavens, 2007].

4 View-based Object Recognition

4.1 Template Matching

Template matching is one of the simplest forms of identification. A template is a matrix that is tested on an image by finding some measure of similarity between the template and the image's pixel values. The template is tested in all possible image locations so, if the image has $W \times H$ pixels and the template $w \times h$, then the matrix that results from the template matching operation should have size:

$$(H - h + 1) \times (W - w + 1) \tag{18}$$

These so-called measures of similarity can be mathematically expressed in several ways.

Using the minimum square differences, the best possible match is obtained for the smallest $R(x, y)$ (smallest difference between image and template).

$$R(x, y) = \sum_{x'=0}^{w-1} \sum_{y'=0}^{h-1} [T(x', y') - I(x + x', y + y')]^2 \tag{19}$$

Where $R(x, y)$ is the resulting value, T the value of the template and I the value of the image, x' and y' the template's line and column coordinates respectively, while x and y are the images line and column coordinates.

Another method is the correlation technique:

$$R(x, y) = \sum_{x'=0}^{w-1} \sum_{y'=0}^{h-1} [T(x', y') \cdot I(x + x', y + y')] \tag{20}$$

In this case, the best match is achieved where $R(x, y)$ is higher (higher correlation). Template matching is a computationally demanding operation since the template is tested on every possible location on the image. However, using OpenCV's template matching function, and Intel Performance Primitives, these operations can be done in real time.

4.2 Haar Features

Haar features were proposed by [Viola and Jones, 2001] as an alternative method for face detection. The general idea was to describe an object as a cascade of simple feature classifiers. This is a very fast method which performs face detection as effectively as any other methods.

As sated in 24 [Viola and Jones, 2001], in the CMU+MIT test set, a reference test set , the method performed 15 times faster than the Baluja-Kanade detector and about 600 times faster than the Schneiderman-Kanade detector.

The classification of images is based on the value of simple features. Features are used instead of simple raw pixel values generally because they can act to encode ad-hoc domain knowledge but also, in this particular case, because they are much faster to process. Later on, [Lienhart and Maydt, 2002] proposed to extend the pool of features by utilizing also 45º rotated features thus "significantly enhancing the expressional power of the learning system and consequently improving the performance of the object detection system" [Lienhart and Maydt, 2002].

The features that were used by [Viola and Jones, 2001] (basic set) and latter by [Lienhart and Maydt, 2002] (extended set) are shown in Figure 13.

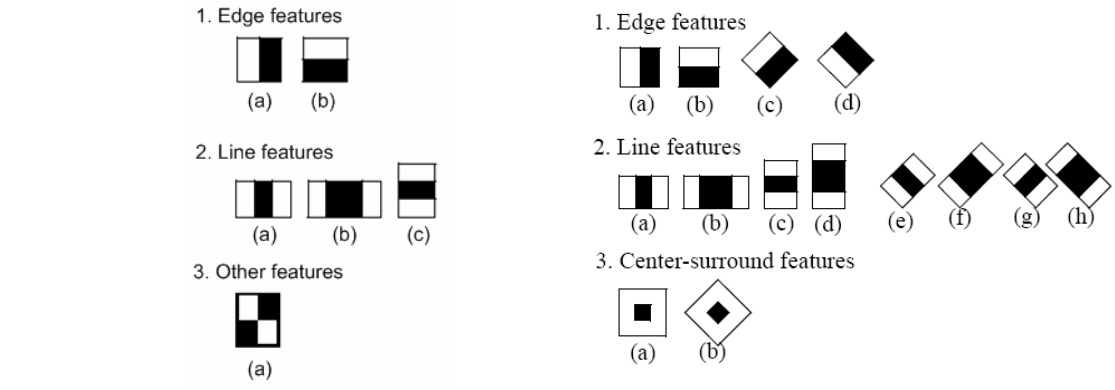

Figure 13. Basic set of Haar features used by [Viola and Jones, 2001]. (left) and extended set applied by [Lienhart and Maydt, 2002] (right).

It is important to emphasize that these are mere prototypes of features. They are then scaled independently in horizontal and vertical directions in order to get an over complete set of features (with the 24x24 window used by 24 [Viola and Jones, 2001] the amount of possible features is around 180000). The result of the application of each feature to a particular image region is given by the sum of the pixels that lie within the black rectangles of the feature subtracted by the sum of the ones overlapping the white rectangles.

$$feature_1 = \sum_{i \in I=\{1,\ldots,N\}} w_i \operatorname{Re}cSum(r_i) \Leftrightarrow \sum_{i \in I=\{1,\ldots,N\}} w_i \operatorname{Re}cSum(x, y, w, h) \qquad (21)$$

The values of N, w_i and of r_i are arbitrarily chosen. In the case of [Lienhart and Maydt, 2002], it has been defined that black rectangles (r_0) have negative weight w_i and white (r_1) have positive weights. Furthermore, the relationship between weights is given by the difference of area occupied by the black and white rectangles.

$$-w_0 \cdot Area(r_0) = w_1 \cdot Area(r_1) \tag{22}$$

Assuming $w_0 = -1$, one can obtain:

$$w_1 = \frac{Area(r_0)}{Area(r_1)} \tag{23}$$

Consequently, for example for feature (2a) of Figure 13, with a height $h = 2$ and width $w = 6$ the outcome of the feature application to a rectangular region positioned at x,y would be:

$$feature_{2a} = -1 \cdot \mathrm{Re}cSum(x, y, 6, 2) + \frac{6 \times 2}{2 \times 2} \cdot \mathrm{Re}cSum(x + 2, y, 2, 2) \tag{24}$$

In order to compute the value of each feature very rapidly an intermediate image representation is calculated. This representation is called integral image or Summed Area Table (SAT). The value of the integral image at coordinates x,y , is given by the sum of all the pixels that are above and to the left of x, y:

$$SAT(x, y) = \sum_{x' \le x, y' \le y} I(x', y') \tag{25}$$

Where $I(x', y')$ is the value of the image.

The value of any $\mathrm{Re}cSum(x, y, w, h)$ can be obtained by simply four lookups at the SAT.

$$\mathrm{Re}cSum(x, y, w, h) = SAT(x + w, y + h) - SAT(x + w, y) - SAT(x, y + h) + SAT(x, y) \tag{26}$$

This procedure is shown on Figure 14.

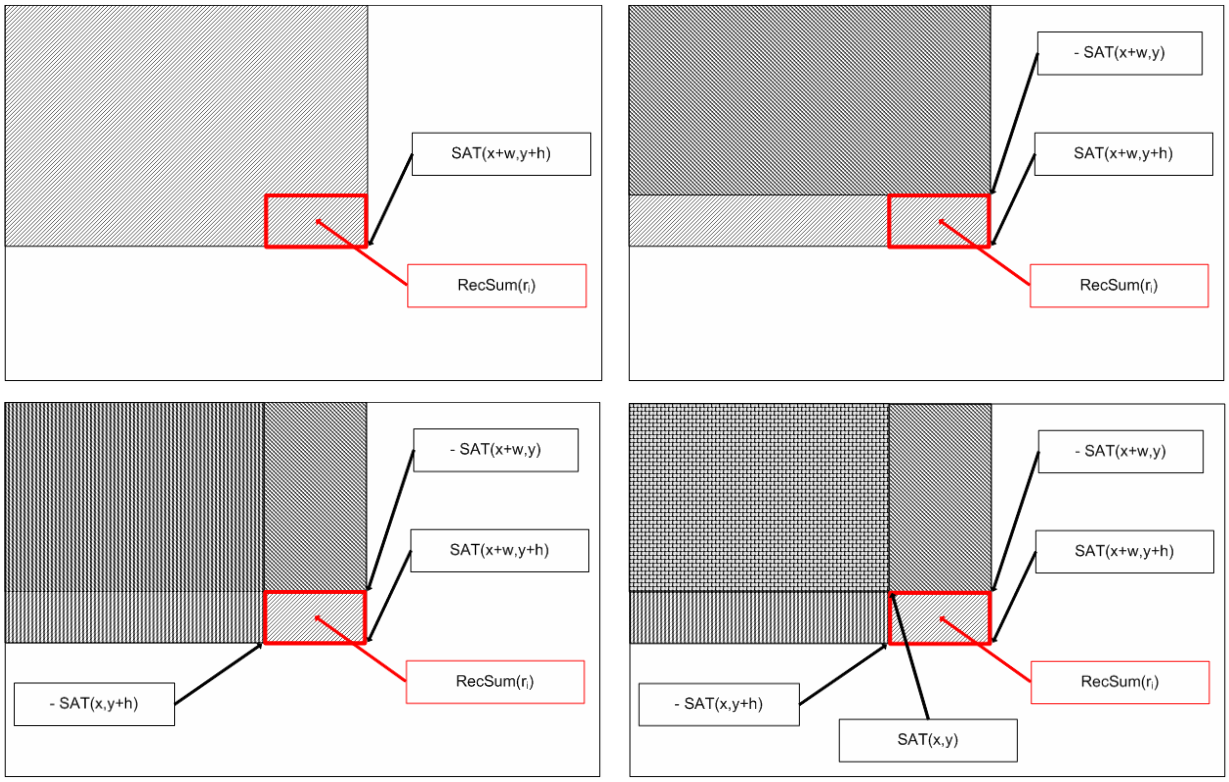

Figure 14. Fast $RecSum(r_i)$ calculation.

Viola *et al.* have set up a framework to combine several features into a cascade. For an object to be recognized, it must pass through all of the stages of the cascade. The cascade is built by supplying a set of positive and negative examples to the training algorithm. The used algorithm is called Adaboost, known for its high performance in what concerns generalization speed. At each stage of the cascade, the machine learning algorithm selects the feature or a combination of features that best separate negative from positive examples, by tuning the threshold classification function. There is a trade off relationship between the number of stages in a cascade and features in each stage and the amount of time it takes to process the cascade. [Viola and Jones, 2001] define, for each stage, a target for the minimum reduction in false positives and a maximum decrease in detection. The mentioned rates are obtained by using a validation set made up of the positive and negative examples.

In order to improve the time performance of the algorithm, [Viola and Jones, 2001] have also presented the notion of attentional cascade. The idea consists of using the first stages of the cascade to effectively discard most of the regions that have no objects. This is done by

adjusting the classifier's threshold so that the false negative is close to zero. By discarding many candidate regions early in the cascade, [Viola and Jones, 2001]. significantly improve the method's performance. In fact, it makes a lot of sense that the detection system is able to quickly discard obvious negative regions of an image using valuable time to better test much more promising regions by submitting them to higher level stages that yield more complex features.

It has already been said that Haar features were developed to perform face detection. However, the framework is all-purpose. Some other approaches have successfully used it for pedestrian detection [Monteiro *et.* al., 2006], hand detection [Chen, 2006] or license plate detection [Dlagnekov, 2005].

5 Visual Tracking

"View-based strategies are receiving an increasing attention because it has been recognized that 3D reconstruction is difficult in practice and also because of some psychophysical evidence for such strategies" [Ude *et. al*., 2004 b]. Therefore, to have a detector that can recognize an object and track it from every possible view is still a very demanding challenge. Furthermore, the dimension of a database that would contain all of the object's possible points of view should be immense just for each single object, unleashing some other problems concerned with real time processing. As an alternative, a method for tracking fully rotating objects based on simple template matching has been developed. The idea is to track a previously classified object. Templates are self-updated when the identification module fails and redefined when they succeed, allowing the object to freely move and rotate overcoming temporary failures of the identification module. Template tracking is very simple. Every new frame is compared to the previous template at every possible location. The best match defines a region of the image that is latter used to provide information for the analysis of the next frame by, somehow, updating .the template. The problem of how to update the template is a difficult one to attend to. [Kaneko and Hori, 2002] have goaled the problem very well: "There is a trade-off relationship between accumulated errors and errors caused by image deformations. If templates are updated frequently, the accumulated errors become large. Conversely, if a template is not updated for a long time, a fatal large error occurs as a result of an image deformation." Kaneko *et al.* approach the problem in a very complex way, defining inclusively an advanced template update criterion. In this study a very simple method is used. The template used for comparison T_c can be the last acquired template, $N=1$, or an average of the latest N templates.

$$T_c = \sum_{n=1}^{N} \frac{1}{N} \times T_n \tag{27}$$

T_c is defined after the intensity values in the image $T_{(x,y)}$, its width (w) and height (h), and also after its last best match position $\vec{r}$.

$$T_c = T_c \left(T_{(x,y)}, T_{width}, T_{height}, T_p \right) \tag{28}$$

Template tracking technique only updates $T_{(x,y)}$ and $\vec{r}$, width and height are not touched. Therefore, this technique requires another module (based on Haar features or any other) to trigger the tracking system by fully defining a starting T_c when a positive identification occurs. Figure 15 shows the tracking of a car's rear, as well as a graph representing the results of the template matching operation (TMR).

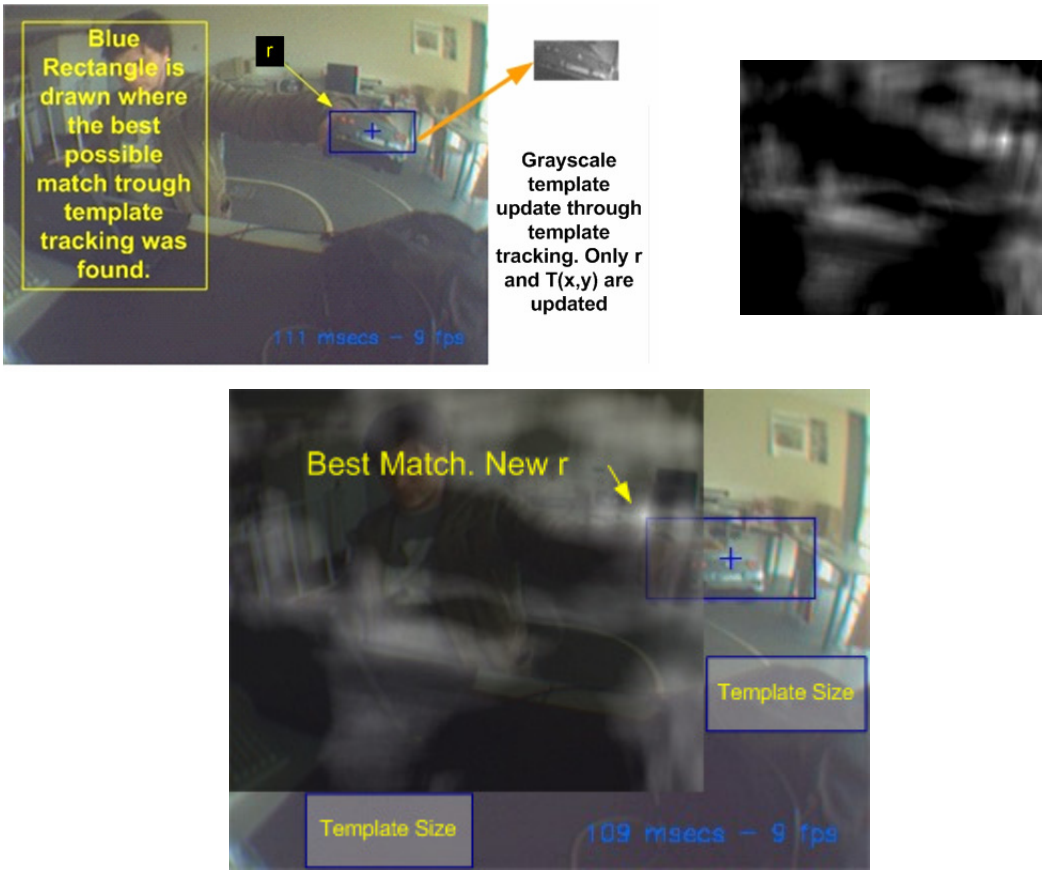

Figure 15. Template tracking (top left) and TMR with the best match in lighter areas (top right). Overlay view (down).

Figure 15's TMR image clearly shows that, despite the fact that the brightest area is around the car's effective position, there are some other areas with a high match result. This observation is related to low car/background contrast. The car's color was purposely chosen to avoid that better results were achieved based on external factors. The same experience works much better for an object of higher contrast.

5.1 Gaussian Conditioning

The template update criterion that is being used is, as mentioned, a very simple one. Perhaps some optimization of this routine could improve the Template Tracking mechanism. Nonetheless, another approach has been attempted. The idea is to condition TMR, i.e. to weight it by some other matrix. This other matrix would supply additional information to the tracking mechanism, other than the absolute measure of similarity inherent to template matching.

Relevant information is related to the positional continuity of objects in time, i.e. objects do not warp from one place to another without following some connecting trajectory between these points. Of course that frame by frame analysis, as is the case, is always a discrete process and warping might occur due to low sampling frequency. However, assuming that the frame rate is high enough, it can be fairly trusted that an object early positioned at some coordinates $\vec{r}^{N-1}$ at instant $N-1$, will have, in the following iteration, a higher probability of its position $\vec{r}^{N}$ being in the neighborhood of $\vec{r}^{N-1}$.

$$P(\vec{r}^{N} \mid \vec{r}^{N-1}) = f\left(\frac{1}{\left\|\vec{r}^{N} - \vec{r}^{N-1}\right\|}\right) \tag{29}$$

In order to embed this "common sense" into our system, it was decided to use a 2D Gaussian probability distribution centered at $\vec{r}^{N-1} = \left[x^{N-1}, y^{N-1}\right]^{T}$, embedded into a matrix with the same size as TMR that will be referred to as Gaussian matrix (GM). GM is calculated as follows:

$$G(a,b) = \frac{1}{2\pi\sigma^2} e^{\left(-\frac{a^2+b^2}{2\sigma^2}\right)} \tag{30}$$

Where $G(a,b)$ is the function's value considering $a = x^{N} + x^{N-1}$, $b = y^{N} + y^{N-1}$ while σ is the standard deviation expressing how "wide" the Gaussian is defined. This enables centering the matrix at the previous template position, as in Figure 16.

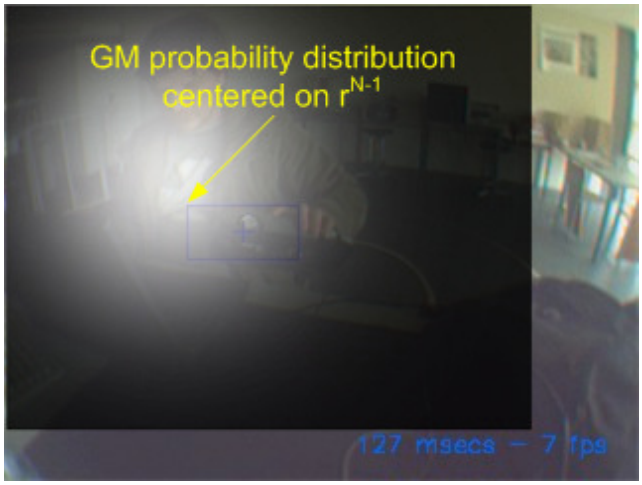

Figure 16. GM overlapped onto its corresponding frame.

After being properly normalized, GM is used to condition TMR by performing a simple pixel by pixel multiplication, thus decreasing the match score for pixels that where distant from the previous $\vec{r}^{N-1}$.

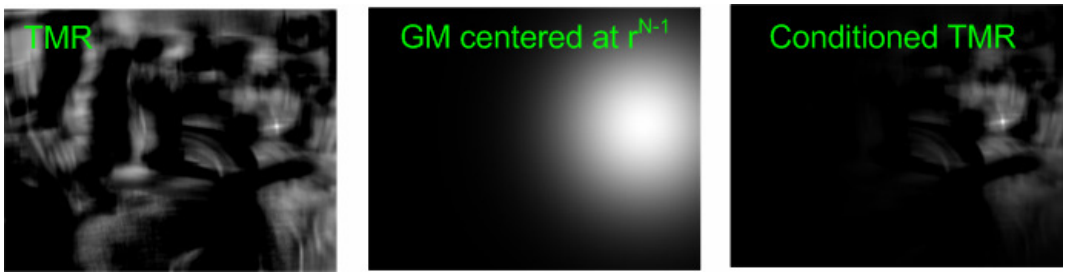

Figure 17. TMR (left), GM centered at $\vec{r}^{N-1}$ (middle) and conditioned GM (right).

This technique was able to considerably improve tracking performance. The phenomena of warping disappeared completely. The tracking works even when objects do not have a high contrast with the background.

5.2 Fast Gaussian Computation

The Gaussian function calculation is not fast to compute, especially considering that any tracking mechanism should be fast when compared to an object recognition technique. Calculating a new GM in all iterations (for every new $\vec{r}^{N}$) would considerably decrease the frame rate. Hence, an alternative method had to be developed. As a part of the program

initialization processes, an extended Gaussian matrix (EGM) is calculated. To calculate EMG's minimum size, note that, by absurd, the smallest template one can use is of size 1×1. Therefore, recovering chapter 4.1 template matching results matrix size equation (18), the biggest size that the template match results matrix can have is $W \times H$, i.e., the image's size. The EGM is also a Gaussian 2D function represented in an image of size $2W \times 2H$, with the function centered in the center of the image. For every given T_p and TMR size, a specific sub-window of the EGM can be used without having to reposition the probability distribution thus recalculating the probability value of every pixel. The EGM sub-window's upper left corner's coordinates are given by:

$$EGM_{upperleft} = \begin{bmatrix} EGM_{height} - \frac{TMR_{height}}{2} - T_{px} \\ EGM_{width} - \frac{TMR_{width}}{2} - T_{py} \end{bmatrix} \tag{31}$$

Its size is equal to the size of TMR. Using this technique, the Gaussian matrix's calculation is limited to the definition of a particular region of interest of the EGM, therefore saving precious computation time (Figure 18).

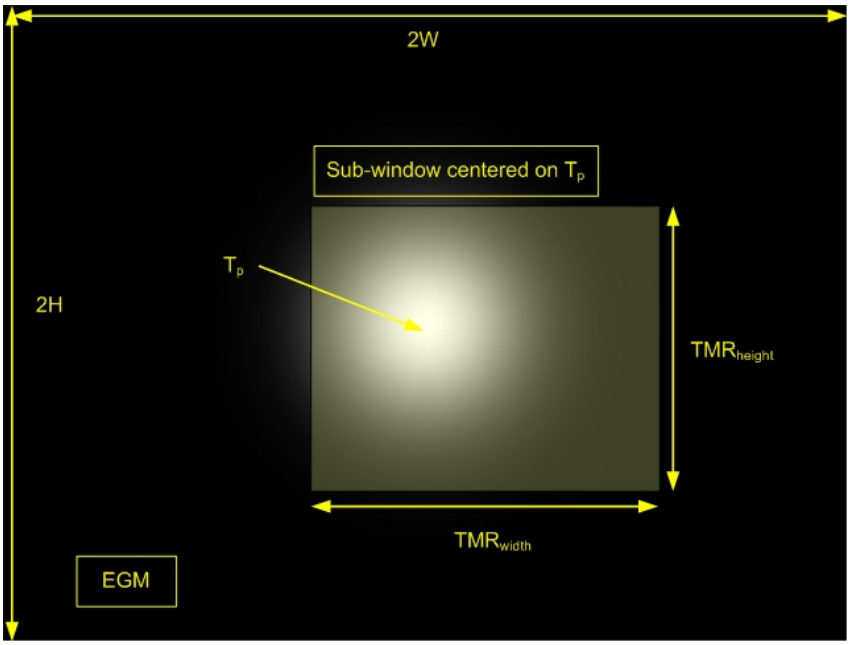

Figure 18. Fast Gaussian computation.

6 Foveation Control

Foveation control is closely related to the term visual servoing: “Machine vision can provide closed-loop position control for a robot end effector - this is referred to as visual servoing” [Hutchinson *et. al.*, 1996]. Foveation control regards the techniques that were used to, having captured the system’s attention, center the object in the foveated image. Although the implementation issues were always handled in a general approach perspective avoiding loss of generality, the solutions here implemented are directed towards specific hardware characteristics. Therefore, this chapter starts with a brief description of the Pan and Tilt unit as well as the cameras setup.

6.1 Pan and Tilt and Cameras Setup

For visual tracking, a servo controlled Pan and Tilt unit is used, model PTU46C from Directed Perception. Its Pan and Tilt axis have 270º/180º positioning range respectively and both support speeds of up to 300º/s. The system allows for position, speed and acceleration control based on RS232 communications.

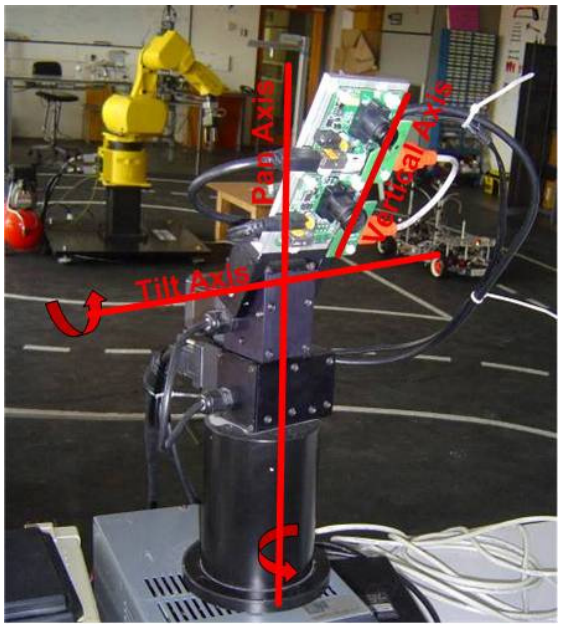

Figure 19. Pan and Tilt setup.

Two low cost Unibrain IEEE1394 cameras are installed on the unit. They are positioned so that both cameras' vertical axes are coincident with the tilt plane, while pan movements shift the image horizontally. With this setup and disregarding lenses distortion it is possible to have independent control of the images' horizontal and vertical axes shift related to pan and tilt movements respectively. The lower camera has a wide lens installed while the upper camera uses a narrow lens. Frame rates of 30 fps with images of 640x480YUV411 or 320x240YUV422 are supported. Similar setups have been proposed by [Ude *et. al*., 2006] in this case for robotic humanoid heads with two cameras per eye.

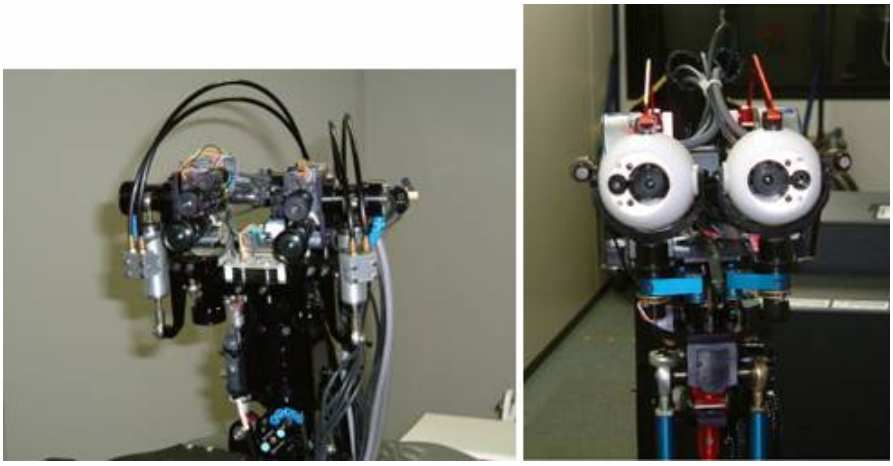

Figure 20. Cameras setup proposed by [Ude *et. al*., 2006].

As can be seen on Figure 20, the registration between foveal and peripheral cameras is proposed either with vertical or horizontal axis coincidence. The mechanism should be able to compensate for the distance (horizontal or vertical) between the cameras.

6.2 Tracking Controller

Object tracking implies following a given object. However, in this approach and for the sake of program modularity, the tracker module does not need to be aware of what it is following nor of its properties. This approach was chosen because it ensures total independence of the tracker module. Its inputs are the current peripheral image coordinates $\vec{v} = \left[x_p, y_p \right]^T$ of the object to track and the desired coordinates, i.e. where the object is and where it ought to be.

Usually, the desired coordinates, named Target $\hat{\vec{v}} = \left[\hat{x}_p, \hat{y}_p \right]^T$ are the image's center. However, that may not always be the case. In this case and because the foveal camera is positioned on top of the peripheral image (in which tracking occurs), $\hat{\vec{v}}$ should be placed above the image's center as well. This problem will be addressed further ahead (chapters 6.4 and 6.5). For now, the important issue is of how to bring the recognized object located at current coordinates $\vec{v}$ to some predefined target coordinates $\hat{\vec{v}}$

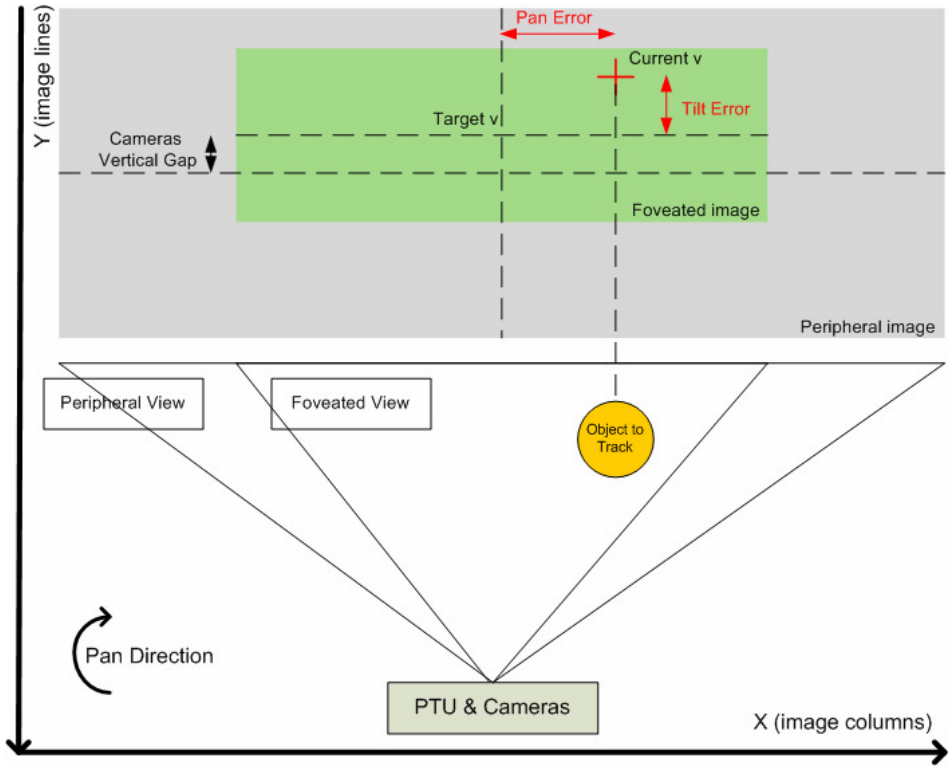

Figure 21. Pan & Tilt controller schematics and controller implementation.

Figure 21 illustrates a schematic of the perception unit. The pan error (P_{error}) is given by:

$$P_{error} = x_p - \hat{x}_p \tag{32}$$

While the tilt error (T_{error}) is defined as:

$$T_{error} = y_p - \hat{y}_p \tag{33}$$

Two independent controllers, one for each image axis, i.e. one for pan control and one for tilt control, are implemented. Nonetheless, both controllers are similar and therefore only the Pan controller will be addressed. Note that the philosophy of the whole implementation is not to use vision for exact metric measurements, but conversely, to use it for some kind of self calibrating control using only correlations between pixel distances. Therefore, there is no need to calculate the world coordinates that correspond to $\vec{v}$ nor to $\hat{\vec{v}}$. Without possessing this information, it is hard to specify the desired angles to pan/tilt based only on the object image coordinates. These angles have to be specified to the hardware unit. To solve this problem, fixed angle values are set. That is, if the object is on the right side of $\hat{\vec{v}}$ ($T_{error} > 0$), the pan position's value ($P_{position}$) is a predetermined one that pans the cameras entirely to the right, while the opposite occurs when the object is on the left.

$$\begin{aligned} &if\,(P_{error} > 0) \Rightarrow P_{position} = 200^{o} \\ &else \Rightarrow P_{position} = -200^{o} \end{aligned} \tag{34}$$

The controller is actually a speed controller (in practice, position only reports the signal of the speed value). Pan speed is controlled based on a PID controller that accounts for P_{error} magnitude, its past history and future trends.

$$P_{speed} = K_p P_{error}^{n} + K_i \sum_{n=0}^{N} P_{error}^{n} + K_d \frac{P_{error}^{n} - P_{error}^{n-1}}{\Delta t} \tag{35}$$

Where K_p, K_i and K_d are the proportional, integral and derivative gains respectively, n is the iteration index, N the max amount of iterations to account for, and Δt corresponds to the time that has elapsed between iterations n and $n-1$. The K_p, K_i and K_d parameters were tuned by the method of empirical calibration. Pan acceleration may also be controlled by a PID controller but for now it is set as a constant high value with acceptable results so far.

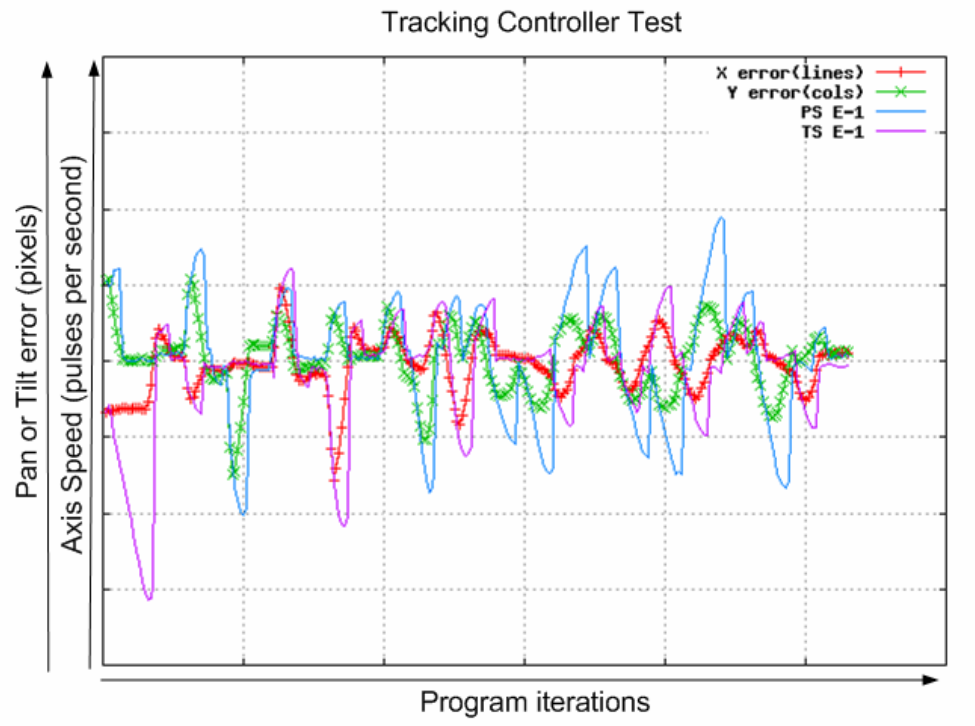

Figure 22. Tracking controller experiment.

Figure 22 shows how P_{speed} and T_{speed} vary when a P_{error} or T_{error} occurs. Independent axis control is noticeable since a T_{error} (x error in Figure 22) implies only that T_{speed} increases. Also, no overshooting is noticeable.

6.3 Estimating Lens Scaling Factors

According to the vendor, the cameras' pixel shape is $5.6\times5.6\mu m$. Considering a resolution of 640×480 pixels, the usable size of the CCD would be:

$$CCD_w \times CCD_h = \left[5.6\times10^{-3}\times640\right]\times\left[5.6\times10^{-3}\times480\right] \simeq 3.58\times2.69mm \quad (36)$$

The vendor also provides the following table.

Table 1. Unibrain cameras lenses.

Lens focal length	Horizontal viewing angle	Vertical viewing angle	Diagonal viewing angle	Size of the reality for D = 0,5 m (see note)		Size of the reality for D = 5 m	
f	*α hor*	*α vert*	*α diag*	*W*	*H*	*W*	*H*
(mm)	(°)	(°)	(°)	(m)	(m)	(m)	(m)
2,10	80,95	65,24	93,70	0,853	0,640	8,53	6,40
2,50	71,27	56,52	83,72	0,717	0,538	7,17	5,38
3,00	61,70	48,26	73,49	0,597	0,448	5,97	4,48
3,60	52,93	40,94	63,78	0,498	0,373	4,98	3,73
4,00	48,26	37,14	58,50	0,448	0,336	4,48	3,36
4,30	45,25	34,71	55,03	0,417	0,313	4,17	3,13
6,00	33,26	25,25	40,94	0,299	0,224	2,99	2,24
8,00	25,25	19,07	31,28	0,224	0,168	2,24	1,68
12,00	16,99	12,78	21,15	0,149	0,112	1,49	1,12
16,00	12,78	9,60	15,94	0,112	0,084	1,12	0,84
Note : each lens has a minimum object distance, that may be longer than 0,5 m							

The peripheral lens has $2.1mm$ focal distance. Hence, the scaling factors (per meter of subject distance) for this camera's horizontal and vertical axis, α_p and β_p respectively, are calculated as follows.

$$\begin{cases} \alpha_p = \dfrac{2\times W_{0.5m}^{fd=2.1mm}}{CCD_w} \simeq 476.0 \\ \beta_p = \dfrac{2\times H_{0.5m}^{fd=2.1mm}}{CCD_h} = 476.2 \end{cases} \tag{37}$$

Which results in a constant scale factor for both directions $\alpha_p \simeq \beta_p \simeq 476.1$.

In the case of the foveal lens, the scale factors are:

$$\begin{cases} \alpha_f = \dfrac{2\times W_{0.5m}^{fd=8mm}}{CCD_w} \simeq 125.0 \\ \beta_f = \dfrac{2\times H_{0.5m}^{fd=8mm}}{CCD_h} = 125.0 \end{cases} \tag{38}$$

Also for this lens, the scale factors are similar $\alpha_f \simeq \beta_f \simeq 125.0$

6.4 Modeling the Foveation Setup Assuming Pinhole Cameras

The aim of this mathematical formulation was to accurately find the values of the peripheral image's target point $\hat{\vec{v}}$ in order to maintain the object in the center of the foveated image. This is a difficult problem to address, since no information regarding the disparity map is present.

[Ude *et. al*., 2006] have formulated the problem by focusing two issues that would need to be considered when analyzing the foveation setup with two cameras:

1) Given a 3-D point that projects onto the center of the foveal image, where will the point be projected onto the peripheral image? This will be the ideal position in the periphery for foveation.

2) If a 3-D point projects onto the peripheral image away from the ideal position described above, how far is the projection of the point from the center of the foveal image?

Both cameras can be modeled based on a standard pinhole camera model. Most of the following deduction as well as assumptions are based on [Ude *et. al*., 2006].

A 3D point is denoted by $M = [X \quad Y \quad Z]^T$ while a 2-D point by $m = [x \quad y]^T$. Let $\tilde{M} = [X \quad Y \quad Z \quad 1]^T$ and $\tilde{m} = [x \quad y \quad 1]^T$ be the homogeneous coordinates of M and m. A 3-D point M and its projection m are related by

$$s\tilde{m} = A[R \quad t]\tilde{M} \tag{39}$$

Where s is an arbitrary scale factor and R and t are the rotation and translation extrinsic parameters. A is the intrinsic matrix and can be expanded as follows

$$A = \begin{bmatrix} \alpha & \gamma & x_0 \\ 0 & \beta & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{40}$$

α and β are the axes scale factors while γ describes the skewness of the two image axes, x_0 and y_0 are the principal point coordinates. Without loss of generality we assume that $x_0 = y_0 = 0$. In most of the standard cameras the principal point i.e., the point that is coincident with the camera's optical axis, does not coincide exactly to the image center in pixel coordinates but it is close to it (about 10 pixels distance in a 640x480 image). The pinhole camera model does not consider effects of lens distortion. In the foveal image distortion is not noticeable because of the long focal lengths. However, this is not the case for the wide lens of the peripheral image whose distortion is quite considerable. This study did not include previous distortion correction of the peripheral image. Distortion is mainly due to radial components, which are higher in the periphery of the image. It is known from previous applications that the target position at the peripheral image should be around some few dozens

of pixels. Therefore, one might assume that this magnitude of values is still very close to the center of the image and so will suffer minor distortion effects. Let A_f, R_f, t_f be the intrinsic and extrinsic foveated camera's parameters, while A_p, R_p, t_p represent the same for the peripheral image. Assuming, as [Ude *et. al.*, 2006], that the world coordinate system is aligned with the coordinates system of the foveated image, then the rotation parameters for the foveated camera are neutral, i.e. $R_f = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix and the translation parameters for the same camera are null, i.e. $t_f = 0$. The translation required to shift from the world/foveated to the peripheral coordinates system is denoted as $\hat{t}$, while $\hat{R}$ respects to the same transformation but in what rotation is concerned, we have

$$R_p M + t_p = \hat{R}\left(M - \hat{t}\right) \tag{41}$$

Expanding equation (41) one can obtain the projection of a 3-D point for the foveal and the peripheral images. The foveal image is the easiest to calculate

$$\begin{bmatrix} x_f \\ y_f \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_f & \gamma_f & 0 \\ 0 & \beta_f & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \right) \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_f X + \gamma_f Y \\ \beta_f Y \\ Z \end{bmatrix} = \begin{bmatrix} (\alpha_f X + \gamma_f Y)/Z \\ \beta_f Y / Z \\ 1 \end{bmatrix} \tag{42}$$

While the peripheral camera's projection is the following

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_p & \gamma_p & 0 \\ 0 & \beta_p & 0 \\ 0 & 0 & 1 \end{bmatrix} \hat{R}\left(M - \hat{t}\right) \tag{43}$$

Replacing $\hat{R}$ for the expansion $r_{i,j}$ then we have

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_p & \gamma_p & 0 \\ 0 & \beta_p & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \left(M - \hat{t}\right) \tag{44}$$

As mentioned before, the objective is to find to which point in the peripheral image corresponds the center of the foveated image (assuming the center is coincident with the principal point). Hence, assuming that M projects into the foveated image's principal point, then $x_f = y_f = 0$. Also, M must be in front of the camera ($Z > 0$). From equation (42) we obtain

$$\begin{bmatrix}0\\0\\1\end{bmatrix}=\begin{bmatrix}(\alpha_f X+\gamma_f Y)/Z\\ \beta_f Y/Z\\ 1\end{bmatrix}\Rightarrow\begin{cases}X=0\\Y=0\\1=1\end{cases} \tag{45}$$

Which means that M is aligned with the foveated camera's optical axis. Substituting in equation (44) one gets the following for the ideal peripheral position $\hat{\vec{v}}=\begin{bmatrix}\hat{x}_p,\hat{y}_p\end{bmatrix}^T$

$$\begin{bmatrix}\hat{x}_p\\ \hat{y}_p\\ 1\\ \end{bmatrix}=\begin{bmatrix}\alpha_p & \gamma_p & 0\\ 0 & \beta_p & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13}\\ r_{21} & r_{22} & r_{23}\\ r_{31} & r_{32} & r_{33}\end{bmatrix}\left(\begin{bmatrix}0\\0\\Z\end{bmatrix}-\hat{t}\right) \tag{46}$$

Considering r_{i*} the i[th] row of $\hat{R}$

$$\begin{bmatrix}\hat{x}_p\\ \hat{y}_p\\ 1\\ \end{bmatrix}=\begin{bmatrix}\alpha_p & \gamma_p & 0\\ 0 & \beta_p & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}-r_{1*}\hat{t}+r_{13}Z\\ -r_{2*}\hat{t}+r_{23}Z\\ -r_{3*}\hat{t}+r_{33}Z\end{bmatrix}=\begin{bmatrix}-r_{1*}\hat{t}\,\alpha_p+r_{13}Z\alpha_p-r_{2*}\hat{t}\,\gamma_p+r_{23}Z\gamma_p\\ -r_{2*}\hat{t}\,\beta_p+r_{23}Z\beta_p\\ -r_{3*}\hat{t}+r_{33}Z\end{bmatrix} \tag{47}$$

which results in

$$\begin{bmatrix}\hat{x}_p\\ \hat{y}_p\end{bmatrix}=\begin{bmatrix}\dfrac{r_{1*}\hat{t}\,\alpha_p+r_{2*}\hat{t}\,\gamma_p-\left(r_{23}\gamma_p+r_{13}\alpha_p\right)Z}{r_{3*}\hat{t}-r_{33}Z}\\ \dfrac{r_{2*}\hat{t}\,\beta_p-r_{23}\beta_p Z}{r_{3*}\hat{t}-r_{33}Z}\end{bmatrix} \tag{48}$$

The first observation is that the foveated camera's intrinsic parameters α_f,β_f and γ_f do not appear in equation (48) and therefore, the ideal position in the peripheral camera is independent of those parameters. The distance to the object Z is, conversely a variable to account for. [Ude *et. al*., 2006] continue this deduction further on expressing the relationship between the displacement of a given M point in the foveated view and its displacement in the peripheral view. In this particular case, however, this analysis is not required because all that is required is to achieve an accurate value of where to position the controller's target in order to best foveate an image.

6.5 Selecting Peripheral Image $\hat{\vec{v}}$ to Center the Object in the Foveal Image

Since cameras are precisely registered, one can assume $\hat{R} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix, i.e. it is assumed with some degree of trust that there is no rotation between both cameras' coordinates' axes. This enables the replacement of equation (48) taking into account that $r_{12} = r_{13} = r_{21} = r_{23} = r_{31} = r_{32} = 0$ and that $r_{11} = r_{22} = r_{33} = 1$

$$\hat{\vec{v}} = \begin{bmatrix} \hat{x}_p \\ \hat{y}_p \end{bmatrix} = \begin{bmatrix} \dfrac{r_{1*}\hat{t}\alpha_p + r_{2*}\hat{t}\gamma_p}{r_{3*}\hat{t} - Z} \simeq \dfrac{r_{1*}\hat{t}\alpha_p}{r_{3*}\hat{t} - Z} \\ \dfrac{r_{2*}\hat{t}\beta_p}{r_{3*}\hat{t} - Z} \end{bmatrix} \tag{49}$$

Assuming a perfect pinhole camera the skewness is neglected, which resulted in the approximation made for $\hat{x}_p$ .in equation (49). The cameras' support structure was especially machined for this application. Having ensured the setup's precision, one can state that the optical axes the cameras are almost perfectly aligned and so that

$$\hat{t} = \begin{bmatrix} 0 \\ \hat{t}_y = 55mm \\ 0 \end{bmatrix} \tag{50}$$

For our foveation setup $\hat{t}_y$ corresponds to the vertical displacement of the cameras' and is approximately 55 millimeters. Substituting equation (50) in equation (49) and recovering chapter 6.3 estimated values of the lenses scaling factors ($\alpha_p \simeq \beta_p \simeq 476.1$) we get

$$\begin{bmatrix} \hat{x}_p \\ \hat{y}_p \end{bmatrix} = \begin{bmatrix} 0 \\ \dfrac{55\beta_p}{-Z} \end{bmatrix} = \begin{bmatrix} 0 \\ \dfrac{55 \times 476.1}{-Z} \end{bmatrix} \tag{51}$$

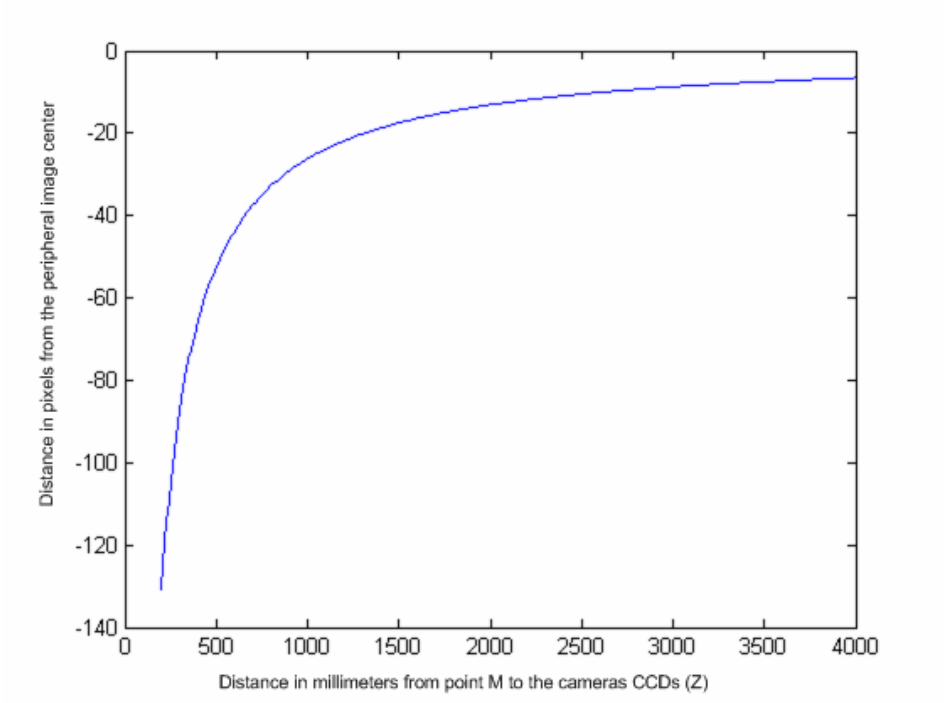

Figure 23. Plotted equation (51) by varying Z from 0.2 to 4 meters. Notice that for distances bellow 1 meter $\hat{y}_p$ varies abruptly and then tends to be more or less constant.

Unlike [Ude *et. al.*, 2006], our foveation setup uses only two cameras (instead of four). Therefore, no disparity map is obtained and consequently there is no information regarding the value of Z. However, this analysis was important to have a clue of the acceptable range of Z values. Looking at Figure 23 we can see that for $Z \geq 1$ meter, $\hat{t}_y$ ranges from -30 to -5 pixels. In 640x480 images this is quite insignificant (6% to 1% of the image's height). The final foveation control assumption is that the unit, with its current setup, should properly foveate objects standing over 1 meter away. A value of $\hat{t}_y = -20$ is used as a constant by the tracking program. If not entirely accurate, this scheme enables the mechanism to, at least, roughly (at most $\max\left[(-5)-(-20),(-30)-(-20)\right]=15$ pixels away from the desired position) point the foveated camera at the object. In fact, although in the possession of a stereo distance measuring setup, [Ude *et. al.*, 2006] also use a constant Z approach for foveation because the alternative strategy "is not practical on highly dynamic humanoid robots because it makes the unrealistic assumption that we can maintain the calibration of the eyes during fast eye

movements.” Also, the core idea of this work is to obtain a self calibration system that does not require complex setup/maintenance procedures. Bearing this in mind, the next step would be to use both cameras to perform foveation control instead of just the peripheral camera. A simple law that would shift the control between the cameras according to the proximity of the object to the target $\hat{\vec{v}}$ could be used. Peripheral camera would roughly bring the object into the foveated image which would then take charge of the foveation control to accurately foveate the object.

7 Software Architecture

This chapter describes how all the previously exposed modules, i.e. attention mechanisms developed to perform visual search, view-based object recognition, object tracking and others are used and integrated into an application. The explanation is mainly based on flowcharts. For a better comprehension of these, the used convention is shown on Figure 24

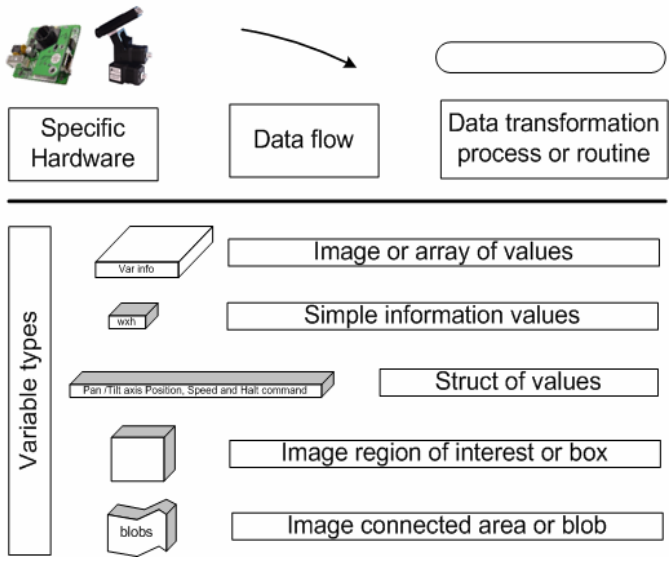

Figure 24. Flowcharts symbol legend.

7.1 Image Acquisition Flowchart

Image acquisition is a fundamental module of the system. Both cameras images are grabbed at a max resolution of 640x480 in YUV422 format. Nonetheless, many of the subsequent modules do not require such a high resolution. Therefore, this module also downsamples the images, making all three resolutions available per captured frame (640x480, 320x240 and 160x120).

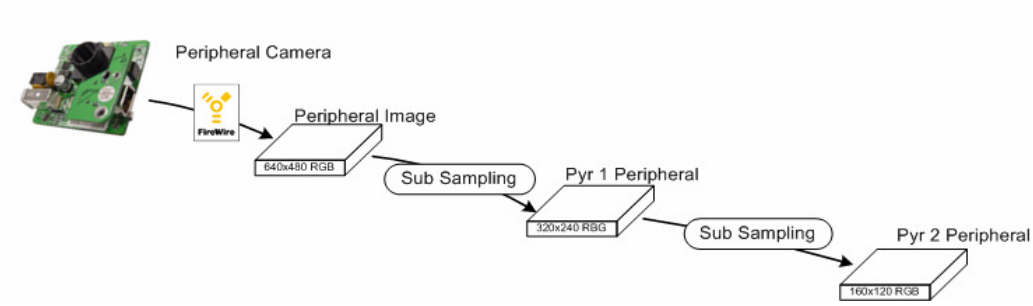

Figure 25. Image acquisition flowchart.

7.2 Visual Search Flowchart

Visual search implemented modules comprehend color detection, motion detection and also the use of Haar features cascades. All three should preferably be processed separately, since their levels of complexity are quite dissimilar.

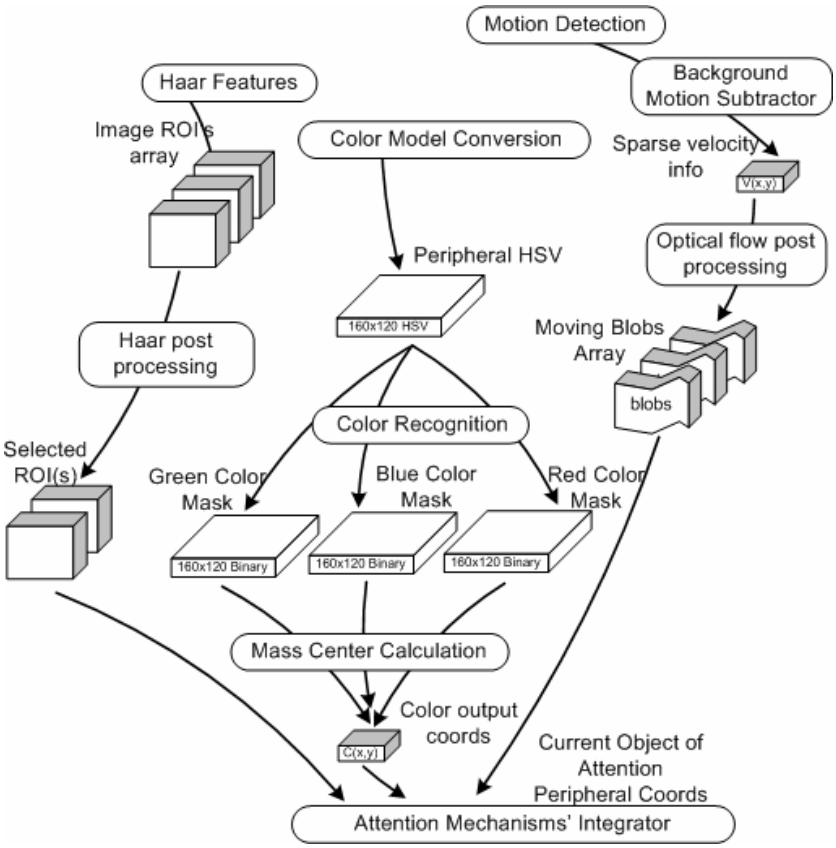

Figure 26. Visual search flowchart.

The Attention mechanisms' integrator module has not yet been developed. As a consequence, only one attention mechanism may work at a time.

7.3 Visual Tracking Flowchart

As previously mentioned, the visual tracking's module requires a full definition of the templates characteristics but, once triggered, it can self update the template's intensity values, which is reflected in the circular loop of Figure 27's flowchart.

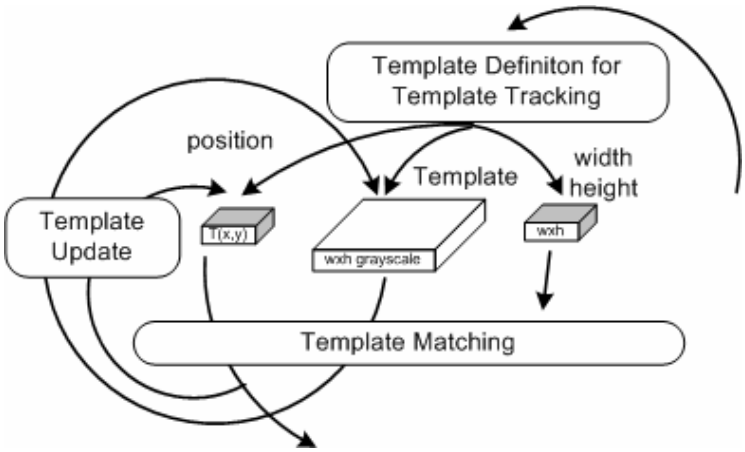

Figure 27. Visual tracking flowchart.

7.4 View-Based Object Recognition Flowchart

Only two recognition techniques were employed: Haar features and template matching. Some experiences were also made with an optical character recognition application. This will be mentioned more in detail in chapter 9.

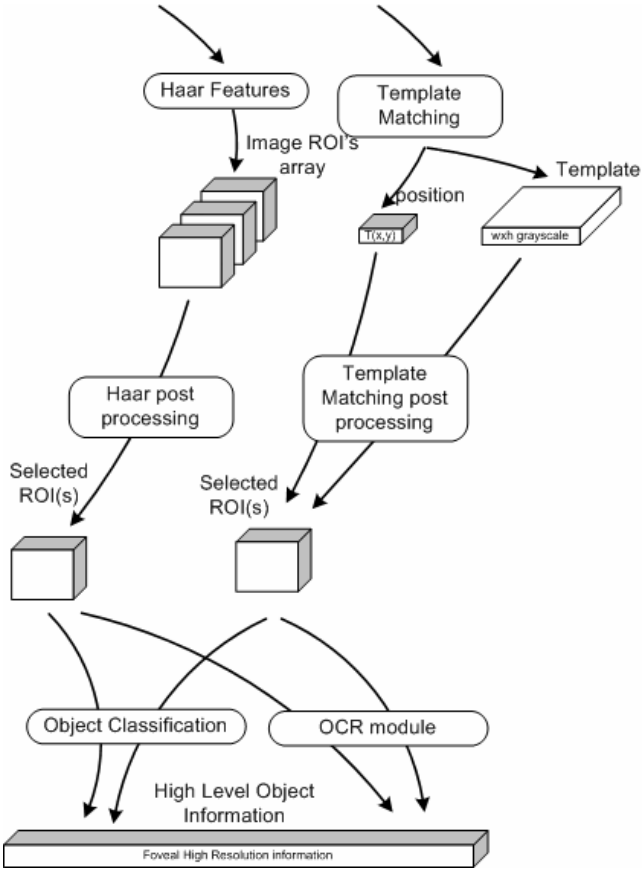

Figure 28. Object recognition flowchart.

7.5 Foveation Control Flowchart

Foveation control aims at positioning an object viewed in the peripheral image in the center of the foveated view. For this a tracking controller module and a communications module were developed (Figure 29).

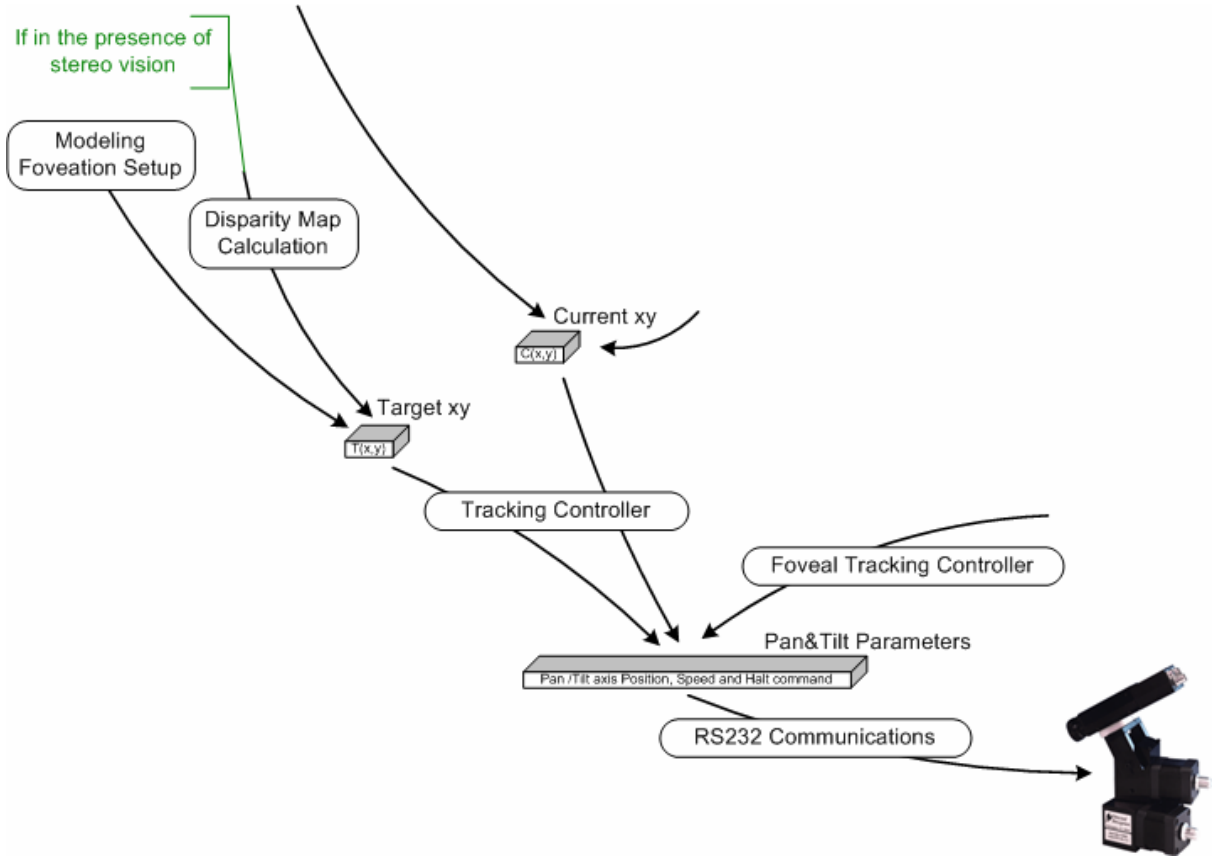

Figure 29. Foveation control flowchart.

7.6 Global Architecture Flowchart

Figure 30 shows the global architecture of the program, explaining the interaction between modulesAll previously mentioned modules are there and when identified, a more holistic perspective can be achieved. Complexity derives from the multitude of crossed interactions between all modules.

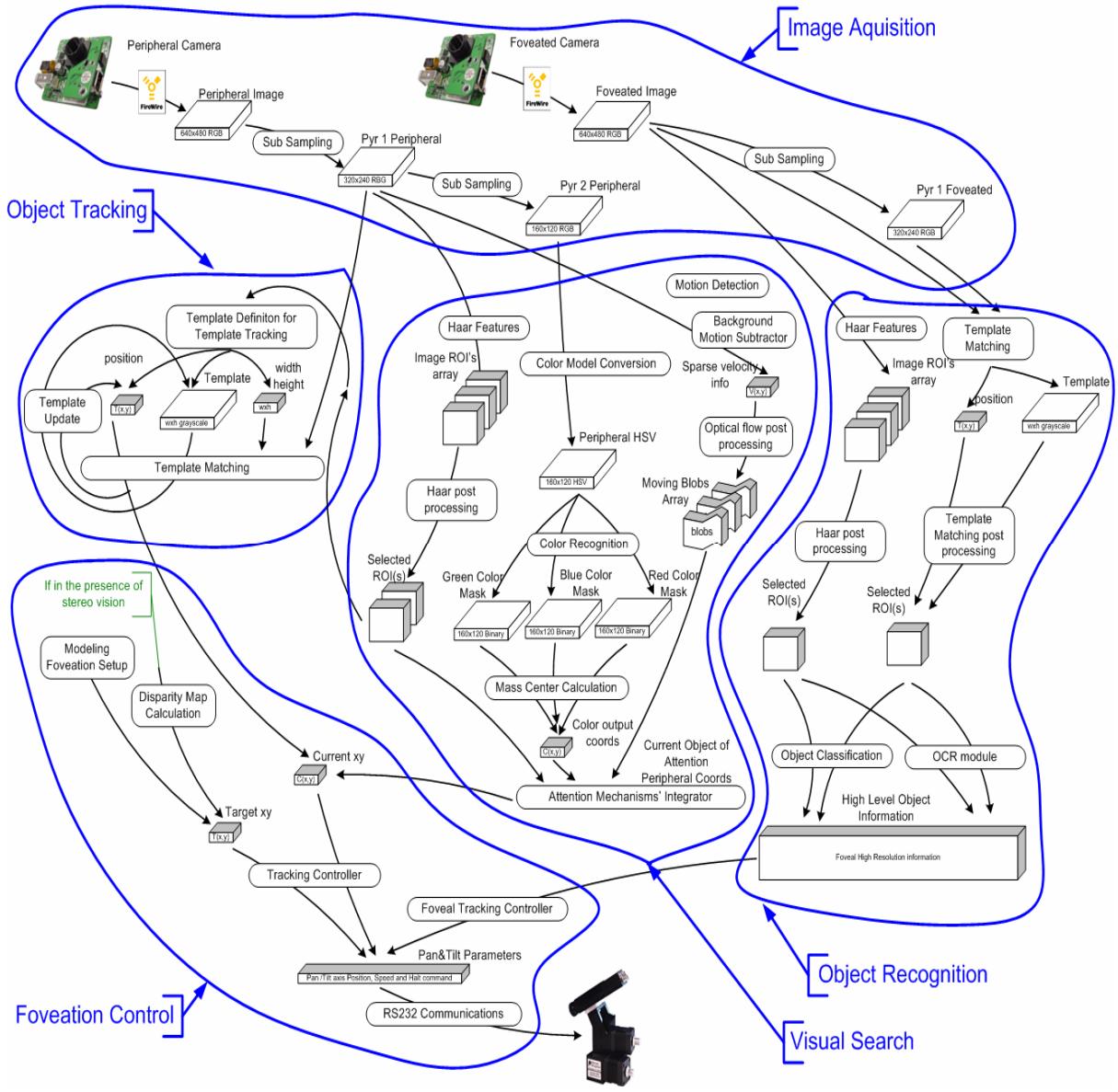

Figure 30. A proposed solution for the integration of the developed modules.

There are five overall modules: image acquisition, visual search, foveation control, object recognition and object tracking. The development of the application intended to follow this model of visual perception. Several class based libraries have been developed. All of the libraries share a common global structure that holds all the information that is relevant, i.e. unimportant local class variables are not included. Also, every developed library includes thread support which enables the launching of a program independent thread and the desired

execution frequency. The architecture supports parallel processing between threads whose priority can be preset. Thread support is provided by QT3, a toolbox very similar to GTK with a vast amount of functionalities. There is but one program whose mission is to setup the initial configuration and to launch the required threads. Figure 31 highlights the threads that are now available to be launched.

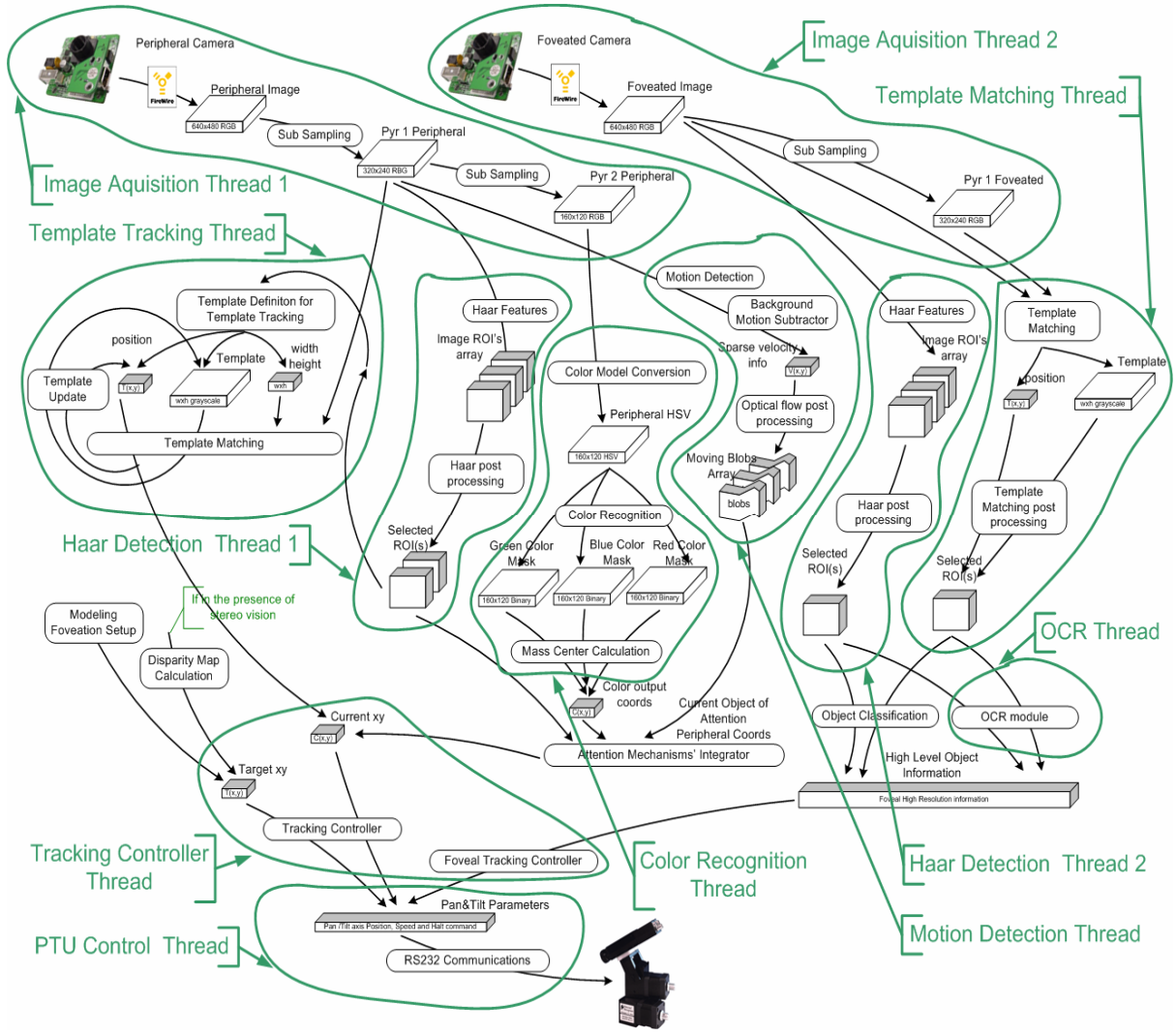

Figure 31. Thread flowchart.

Due to real time demands, not all threads may be running at the same time. However, good results were achieved by trying several combinations of these. Of course that parallel processing brings forward problems due to synchronization of events. The core idea is to have a global pool of information shared by all threads. Threads operate based on this information

and update some other when they are processed. To avoid conflicts arising from simultaneous variable writing and reading, a QT3 based, mutual exclusion system was applied. Variables are locked when accessed by a thread. Some threads do heavy processing over images and for this reason may lock the variable for a long time. To avoid this, a more efficient strategy was implemented. Local copies are made, during which the respective global images are locked. Copying is a very fast process compared to some others and so, the thread will lock the variables for a short period of time, after which it performs the time demanding processing without halting other threads that could require the same variables. All threads are processed at the same time and therefore one admits that the synchronization is performed in a statistical basis, i.e. the best information is always the last updated one.

8 Experimental Results

This chapter will present the most important experiments that were performed. Several practical implementations were attempted in order to test foveation control, either based on Haar features combined with template tracking or color detection. The idea is to try to identify, track and foveate small models of cars. Several Haar cascades were trained to provide robust real time identification of these. Motion detection has also been tested tough it was not integrated into the system as an attentional mechanism. The objective of these experiments was to be able to direct peripheral attention to a particular object, identify it, and then follow it as it moves and rotates freely. At the same time, the object could undergo additional foveal processing.

8.1 Detection of Small Model Car Using Haar Cascades

For experimental purposes, it was decided to try to follow the rear of a small model of a car, whose color was purposely selected to be very similar to the background in the test laboratory, therefore demanding a higher effectiveness of the detection as well as the tracking modules. The contrast between the object and the background is significantly low. Approximately 1400 images of the object were hand labeled. This process consists of, for each training image, defining the region of interest (ROI) where the object is.

Figure 32. Car model used for training (top right). Set of positive images used for training (other).

The training set was converted to grayscale and then reduced to a size of 25x12 using OpenCV's embedded application. Considering [Viola and Jones, 2001] 24x24 face detector,

the elected size appeared to be reasonable. Identification's performance is very effective, including when handling with a large zoom factor. However, when the object is shown at a slightly different angle than the one used for training, the detection rates considerably fall. No quantitative results can be provided since that cascade testing was made only at real time in order to test all of the other programming modules. When this experience was made, thread programming was still not implemented and so the task of saving images slowed the program and disrupted the systems performance.

8.2 A Generalized Cascade for the Detection of Car's Rears

The next attempt was to train a generalized cascade for the detection of car's rears, i.e. to train a Haar cascade that would detect not a particular car model, but a indiscriminate car's rear detector. As mentioned in chapter 4.2, Haar features are first trained to obtain a representation to be used latter for real time object detection. For this purpose several image collections were acquired.

8.2.1 Training Datasets Description

For training purposes, two image datasets were borrowed from the internet and a third was home made. This chapter will describe in detail each set, indicating the number of images per set, their properties and locations were they were taken. Table 2 sums up the training sets information. Training datasets will, henceforth, be named as TDS followed by their respective number.

Table 2. Training datasets description.

Name	Num of Images	Resolution	Format	Location	Authors
TDS 1	1556	variable	png	California	unknown
TDS 2	126	896x592	jpg	California	Weber
TDS 3	1004	752x512	png	Portugal	Oliveira, Santos

California Institute of Technology dataset is composed of 1156 images in *png* format, tough many are very similar (Figure 33). Image resolution is variable. This dataset is used for training and will henceforth be named training dataset 1 (TDS1).

Figure 33. Samples from of California Institute of Technology dataset.

Markus Weber's dataset is not as broad, bearing only 126 images. The resolution is 896 x 592, *jpg* format and the images were taken in the California Institute of Technology parking lots. Some examples are on Figure 34. This will be named training dataset 2 (TDS2).

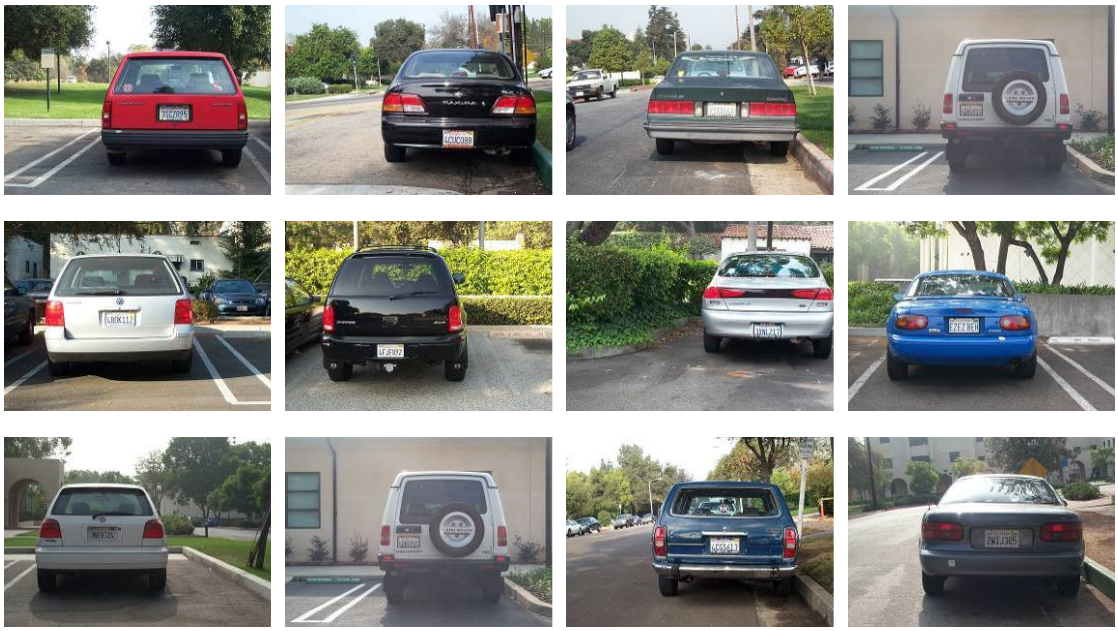

Figure 34. Car dataset taken by Markus Weber, California Institute of Technology.

A third dataset was home made. During a car travel in Portugal (from Algarve to Aveiro) nearly 2 hours of footage was captured. Resolution was 752 x 512. Over 1000 images were extracted from the film. Positive examples were separated and cars were also hand labeled (Figure 35). No rescaling was performed. This will be referred to as training dataset 3 (TDS3).

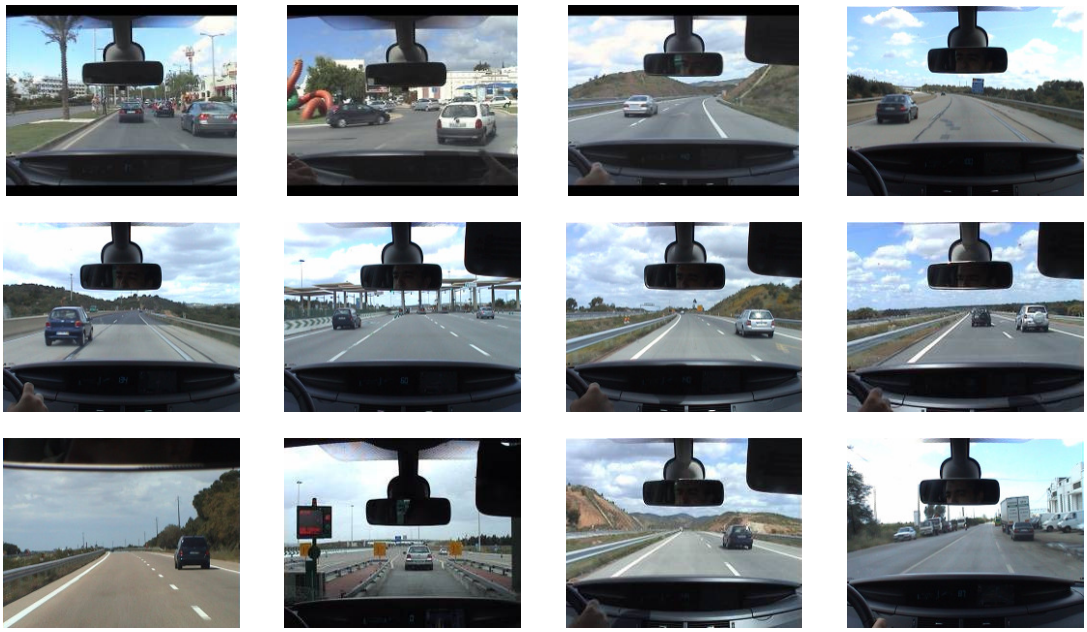

Figure 35. Home made car's rears dataset. From Portuguese roads.

Some of the images were taken during adverse weather conditions, such as rain. Some examples are present on Figure 36. These images are included in TDS3.

Figure 36. Home made car's rears dataset with poor weather conditions.

8.2.2 Test Datasets Description

For the purpose of testing, three separate datasets are used. The first dataset was built by Brad Philip and Paul Updike (Figure 37). It was taken on the freeways of southern California. It is composed of 530 images in *jpeg* format. Resolution is constant at 320x240 pixels. Images are

quite similar to TDS1 but are not included in it. This test dataset will be employed to measure the performance of the cascades and will be mentioned as performance dataset 1 (PDS1).

Figure 37. Example of the car dataset taken by Brad Philip and Paul Updike, California Institute of Technology.

Performance dataset 2 (PDS2) is taken from the footage that provided images for TDS3. The images are not the same although they are similar. PDS2 consists of 105 images, 756x512 of resolution, saved in png format. No demanding weather conditions, city environment or gas stations images were included. The idea was to use a simplified version of the footage. Finally, performance dataset 3 (PDS3) is an extension of PDS2 obtained by including all kinds of images: Poor weather, city, gas stations, bridges etc. (Figure 38).

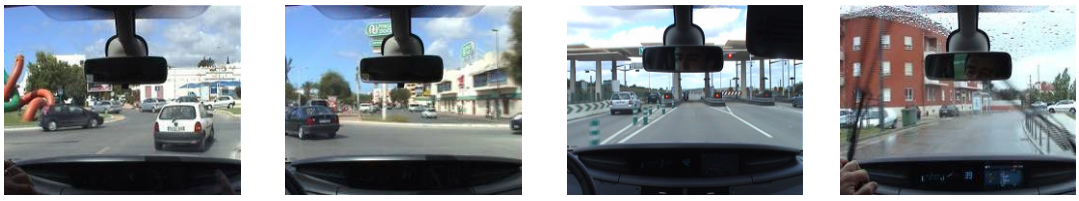

Figure 38. Difficult images in PDS3.

PDS3 is a much harder set. It consists of 232 images with the same resolution and format as the ones of PDS2.

Table 3. Performance datasets description.

Name	Num of Images	Resolution	Format	Location	Authors
PDS 1	530	320x240	png	California	Philip, Updike
PDS 2	105	752x512	png	Portugal	Oliveira, Santos
PDS 3	232	752x512	png	Portugal	Oliveira, Santos

8.2.3 Cascades Description

Having ensured a wide variety of examples, hand labeling was performed over all images both in training and performance sets. A semi-automatic hand labeling application was developed to ease the process by enabling fast mouse selection. It also generates a text file were the ROI is defined for every image. OpenCv's Haar feature tool creates samples by clipping the defined ROIs from TDS images, converting them to grayscale, rescaling them to window size, and inserting them into a random background image. The background image pool, or negative set, has not yet been described. A negative set consists of a set of images where no objects can be found. They haven't been mentioned since they are impaired with their respective TDS, i.e., every TDS also has a set of negative examples, usually road images where no cars are present.

Several cascades were trained using different combinations of TDS, number of stages, features pool, i.e., BASIC for Viola Jones features' poll and ALL meaning Lienhart and Maydt extended set (review chapter 4.2) as well as the number of positive and negative samples generated after the dataset (T. Samples). Also, several window sizes were attempted. Cascades will henceforth be named by C followed by their respective number. A brief description can be found on Table 4.

Table 4. Cascades Description.

Name	Win Size	Training Set(s)	T. Samples (pos/neg)	Nº Stages	Features Set
C1	30x20	TDS1 + 2	unknown	20	BASIC
C2	60x40	TDS1 + 2	1282 / 754	20	BASIC
C3	30x20	TDS3	unknown	20	BASIC
C4	30x20	TDS3	unknown	20	ALL
C5	60x40	TDS3	unknown	20	BASIC
C6	30x20	TDS1 + 2 + 3	1556 / 915	20	BASIC
C7	30x20	TDS1 + 2 + 3	1556 / 915	20	ALL
C8	20x12	TDS1 + 2 + 3	1556 / 915	30	ALL

8.2.4 Performance Tests

OpenCV provides a tool for cascade performance testing. The tool applies the cascade to all test images and compares the algorithm's outcome to the report generated by hand labeling. Hit and false detection rate is generated based on this comparison. In order to assume a given detection as one described in the report, some tolerances are given. These tolerances are positional, i.e. how far is the detection from were it ought to be, and in terms of size, i.e. how bigger/smaller is the detection from what it should be. In order to allow for easy performance comparison, the tolerances employed are the default values of the mentioned tool. PDS1 was tested with several cascades and several scaling factors scaling factor, sf, which is a Haar detection parameter that indicates how much the reference window should be scaled up. Hit rates are quite good (some above 95%) tough false alarm rates, i.e. false positive detections, are quite high (Table 5).

Table 5. Performance results for PDS1.

Name	sf	Hits	Missed	False Alarms	Hit Rate	False Alarm Rate
C1	1.05	508	18	400	0,966	0,760
	1.9	370	156	263	0,703	0,500
C2	1.05	501	25	444	0,952	0,844
	1.5	501	25	193	0,952	0,367
	2.9	311	215	500	0,591	0,951
C3	1.05	440	86	4840	0,837	9,202
	1.9	436	90	1529	0,829	2,907
C5	1.05	449	77	2676	0,854	5,087
	1,9	495	31	953	0,941	1,812
	2,9	397	129	874	0,755	1,662
C6	1.05	442	84	5799	0,840	11,025
C7	1.05	429	97	3385	0,816	6,435
	1,9	396	130	925	0,753	1,759
	2,9	193	333	868	0,367	1,650

The cascades that best perform would be $C1_{sf=1.05}$ and $C2_{sf=1.5}$. The performance curves for both is presented at Figure 39.

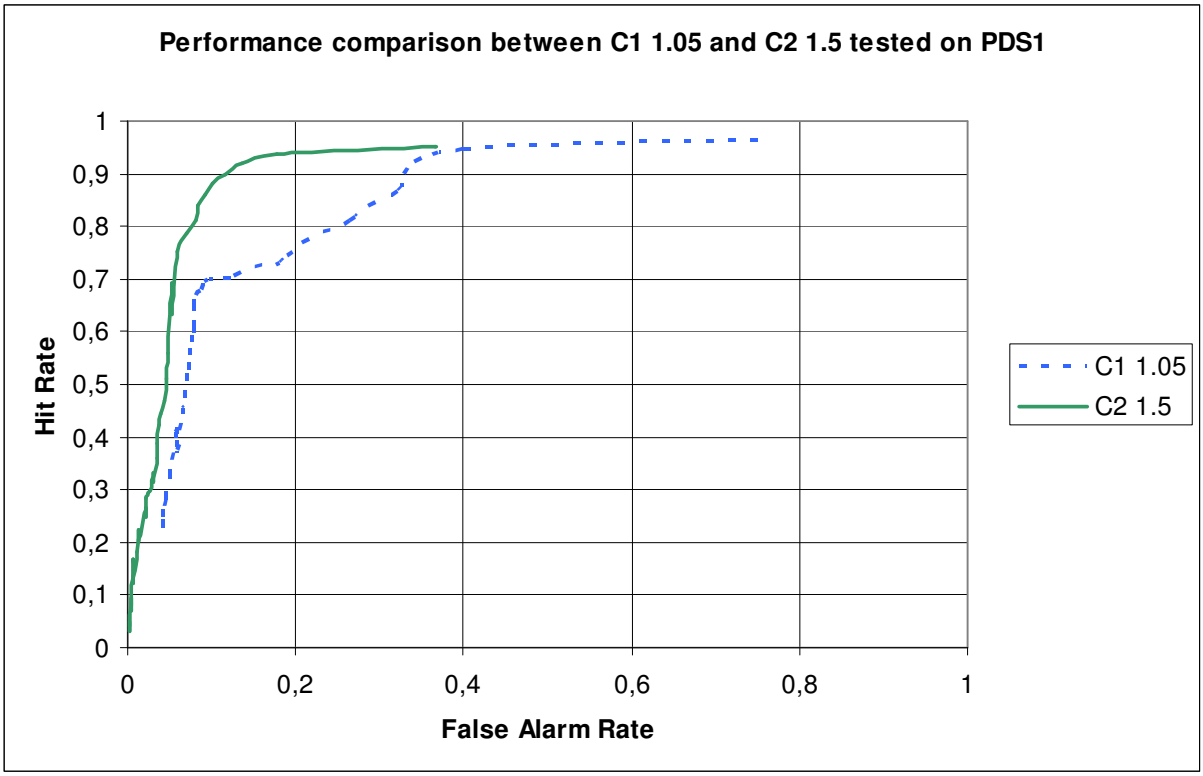

Figure 39. Comparison of the cascades that best performed on PDS1.

Figure 39 shows that cascade $C2_{sf=1.5}$ performs better than $C1_{sf=1.05}$. Cascade $C2_{sf=1.5}$ can achieve the same hit rate as $C1_{sf=1.05}$ at a lower cost, i.e., lower false alarm rate. Some examples of $C2_{sf=1.5}$ detections can be seen on Figure 40.

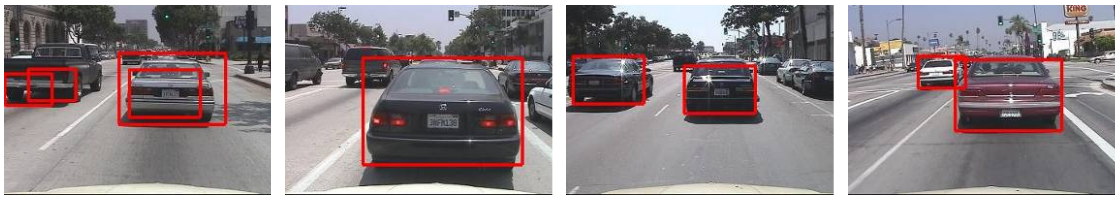

Figure 40. Some detections made by $C2_{sf=1.5}$ on PDS1 dataset.

Regarding PDS2, fewer tests were executed.

Table 6. Performance results for PDS2.

Name	sf	Hits	Missed	False Alarm	Hit Rate	False Alarm Rate
C4	1.05	116	28	2150	0,806	14,931
C5	1.05	79	65	1213	0,549	8,424
C6	1.05	84	60	1081	0,583	7,507

Table 6 clearly shows a much higher false alarm rate's average score. While $C4_{sf=1.05}$ yields the best hit rate, it also has a false alarm rate of 14, i.e. for every detection that should be made, 14 false alarms occur. This number may appear high if the cascade is used for actual detection but may loose relevance if the cascade is to be used as a simple attention mechanism or if further validation tests are to be implemented.

Figure 41. Comparison of the cascades that best performed on PDS2.

Tests with PDS2 were not entirely satisfactory in what concerns false alarm rates. However, this is a very difficult set and some additional procedures could have been implemented to ease the false alarm rate and also, in some cases, improve the hit rate. First of all, the images from PDS3 could be clipped without loss of reliable extrapolation of the algorithm's performance. The upper and the lower parts of these images contain no information on the road (sky/rear mirror and car interior panel). This clipping operation would lower considerably the false alarm rate since many of these false alarms are in these areas of the images (Figure 42).

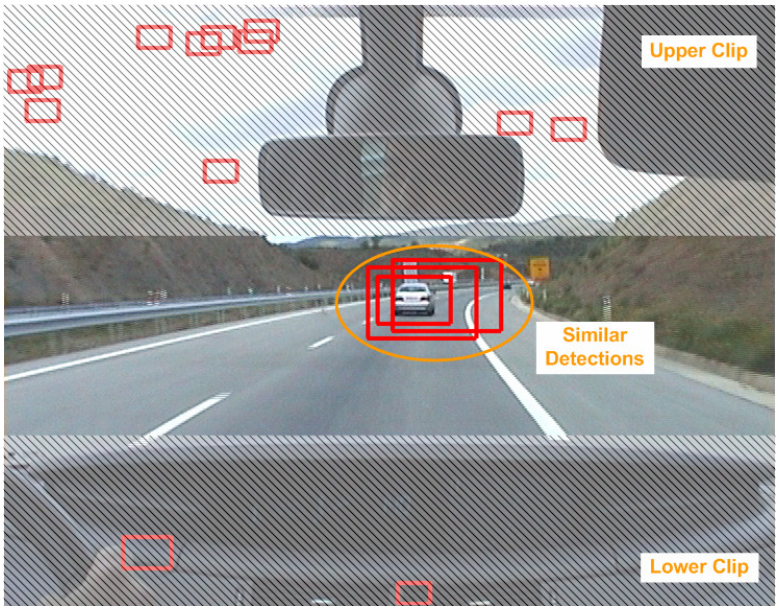

Figure 42. PDS2 apparent problems.

Also, many detections are very close to each other. An algorithm for merging overlapping detections could be easily implemented thus decreasing even more the false alarm rate. In the case of Figure 42, one would go from a situation with 15 false alarms to none, if these procedures were implemented, which would dramatically improve the false alarm rate. Taking the previous considerations into account, it seemed interesting to test some cascades on PDS3, even knowing that it is even more demanding than PDS2. The results are outlined on Table 7.

Table 7. Performance results for PDS3.

Name	sf	Hits	Missed	False Alarm	Hit Rate	False Alarm Rate
C1	1.5	78	249	156	0,239	0,477
C2	1.5	89	238	612	0,272	1,872
C3	1.5	269	58	1510	0,823	4,618
C4	1,05	256	71	4837	0,783	14,792
	1.5	204	123	2028	0,624	6,202
C5	1.5	158	169	1252	0,483	3,829
	1.05	79	65	1213	0,549	8,424
C6	1.5	158	169	1457	0,483	4,456
C7	1.5	115	212	1649	0,352	5,043
C8	1.05	295	32	4119	0,902	12,596
	1,9	30	297	43	0,092	0,131
	2,9	189	138	837	0,578	2,560

Most of the cascades present a poor hit rate. However, cascades $C3_{sf=1.5}$, $C4_{sf=1.05}$ and particularly $C8_{sf=1.05}$ have acceptable hit rates. Of course that the false alarm rates are huge. But

there is the conviction that these rates can be substantially reduced by means of the already mentioned clipping and merging techniques. Performance data was not extracted from $C3_{sf=1.5}$ neither from $C8_{sf=1.05}$ and so Figure 43 presents only the results of $C4_{sf=1.05}$.

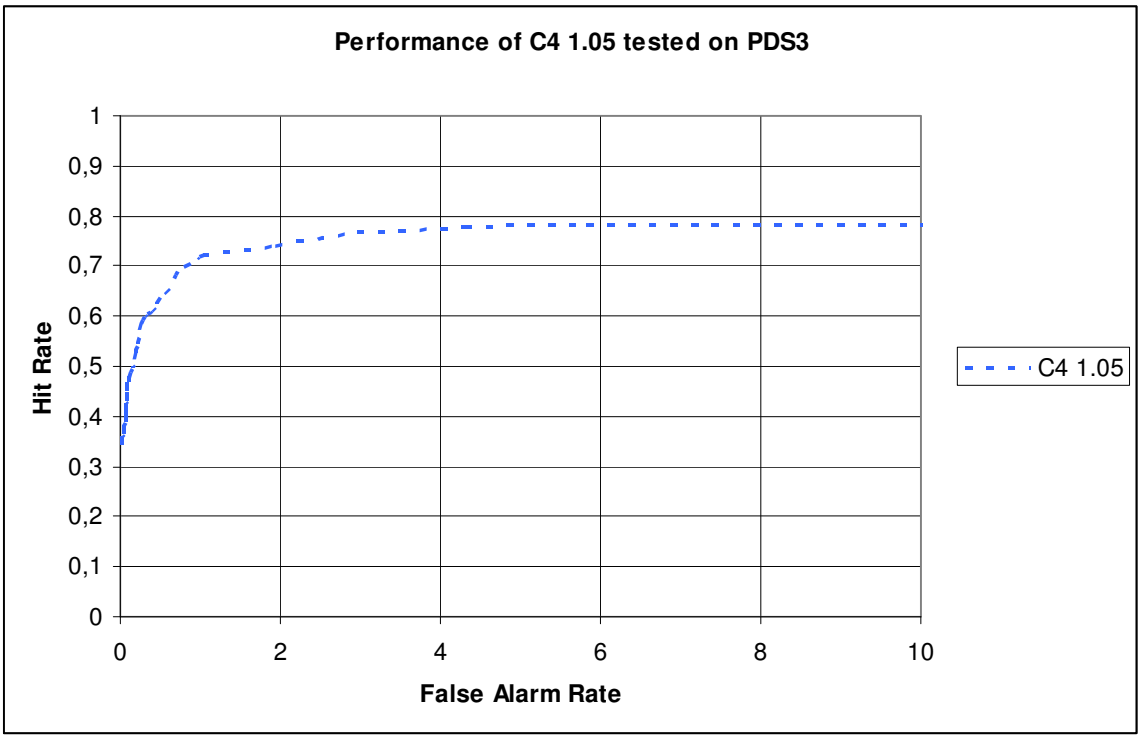

Figure 43. Performance of $C4_{sf=1.05}$ tested on PDS3.

Bearing in mind that false alarm rates could be overstated, and that PDS3 is a set of high complexity including images in the rain, city and other tricky obstacles, the hit rate of $C8_{sf=1.05}$ is quite acceptable.

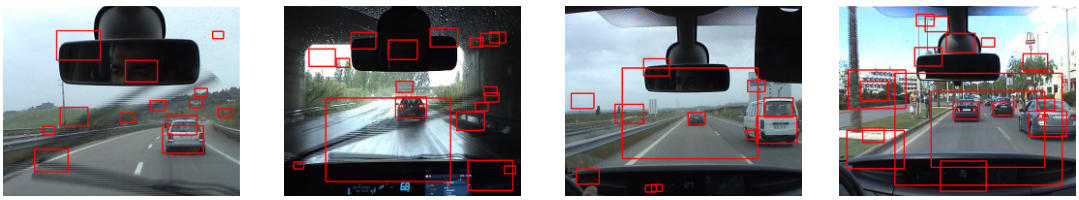

Figure 44. Some detections made by $C8_{sf=1.05}$ on PDS3.

8.3 Color Recognition Based Foveation Control

In order to test the foveation control, a simple color recognition targeting mechanism was employed. In the following images (Figure 45), a red cross represents the target position for foveation and the green cross the red color mask mass center.

Figure 45. Foveation control by red color recognition. Peripheral view.

Foveation can be used for detailed processing. In Figure 46, a contour of the red color mask of the foveal view was being calculated while the peripheral view maintained the foveation control (the bounding box is not always the desired one due to difficulties in the color recognition process).

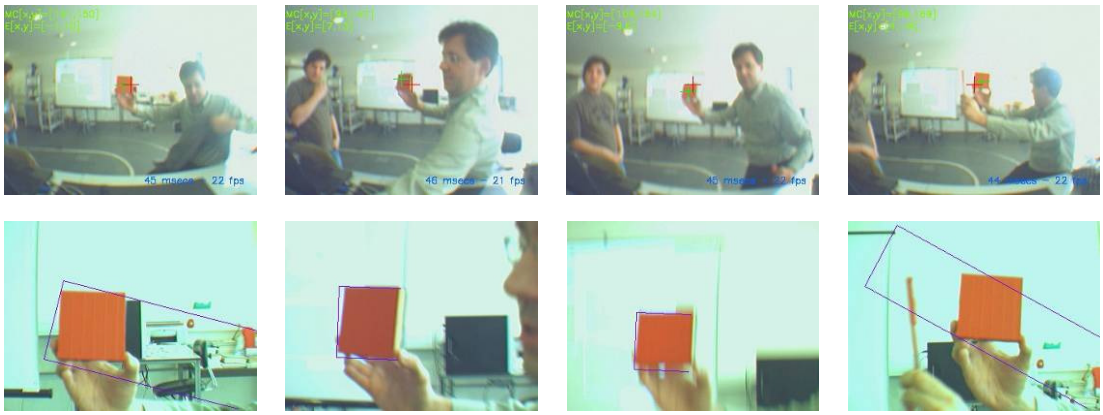

Figure 46. Foveation control by red color recognition. Peripheral view (top row) and corresponding foveal view (bottom row) used to calculate color mask contour.

Chapter 6 has stated that the preferable solution for foveation control would be one that could easily be calibrated, should this procedure be at all necessary. Figure 47 shows how robust foveation control is, since in this sequence the pan and tilt unit was picked up and held upside down. Since both cameras remain properly registered (both are upside down), foveation continues effectively, disregarding global orientation.

Figure 47. Foveation control implementation handles turning pan and tilt upside down.

8.4 Combined Haar Detection Template Tracking Based Foveation Control

This chapter presents the results obtained by performing Haar detection combined with template tracking. Tracking is done whenever Haar detections fail to find a match. Mixing both techniques improves the foveation control largely. The control program is drafted on Figure 48.

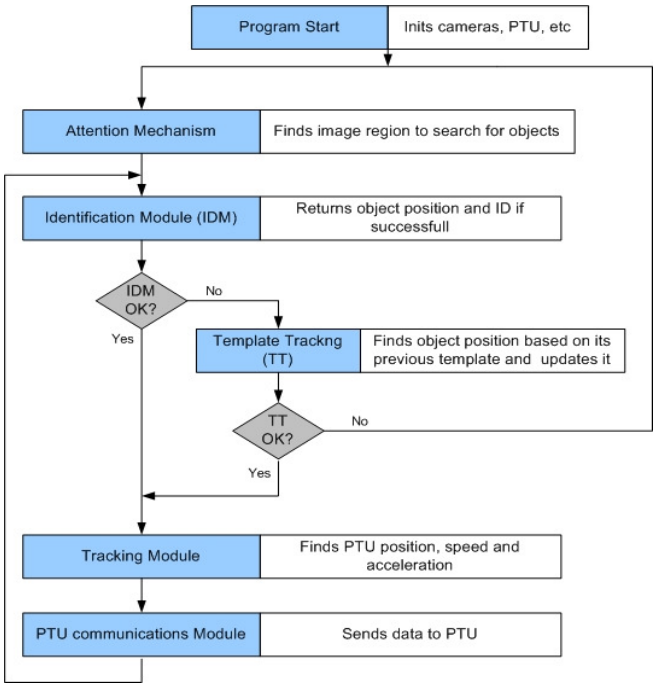

Figure 48. Schematic of how to combine tracking with Haar modules.

Figure 49 shows a sequence where both Haar detections and tracking takes place. The first is represented by a green box around the detected object while tracking template is highlighted by blue boxes.

Figure 49. Combined Haar detection template tracking modules.

Chapter 5 defined that templates width and height are only updated when a Haar detection occurs. Figure 50 shows both the peripheral view and the zoomed template, whose size change due to a different Haar detection is noticeable.

Figure 50. Template updating details.

On Figure 51 another sequence is presented. The object is followed and foveated consistently.

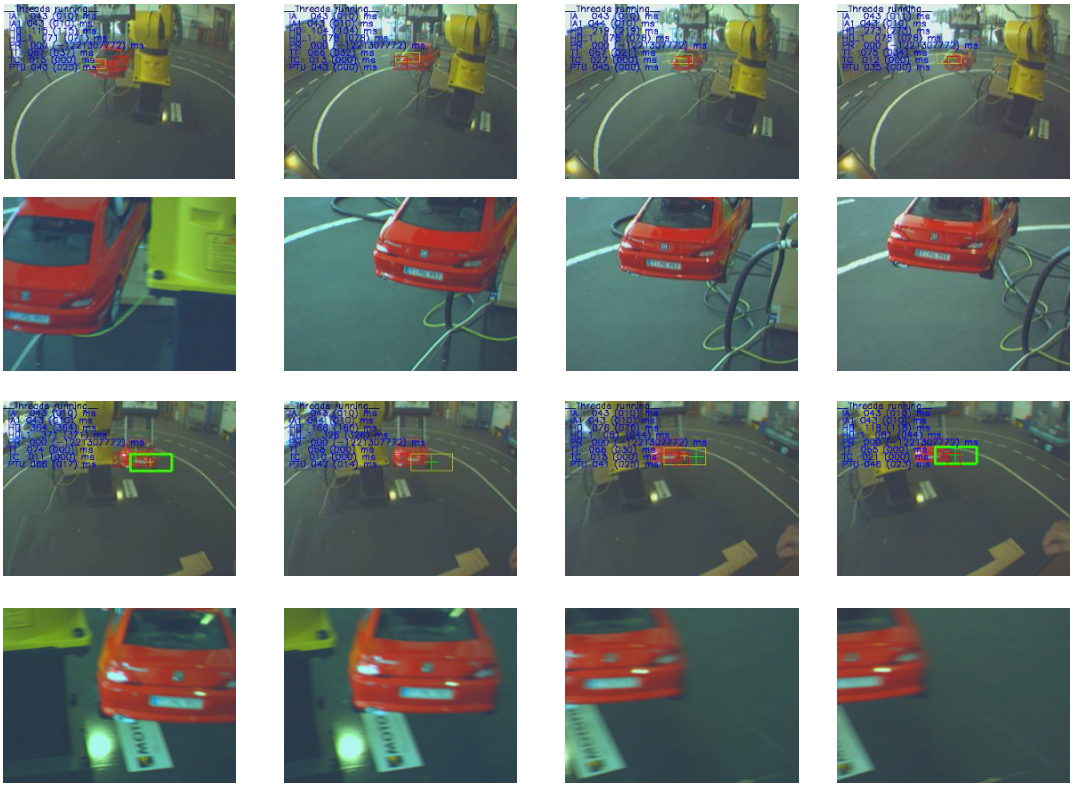

Figure 51. Peripheral template tracking (1st and 3rd row). Foveated view (2nd and 4th row).

Figure 52 shows a complete overview of the tracking process.

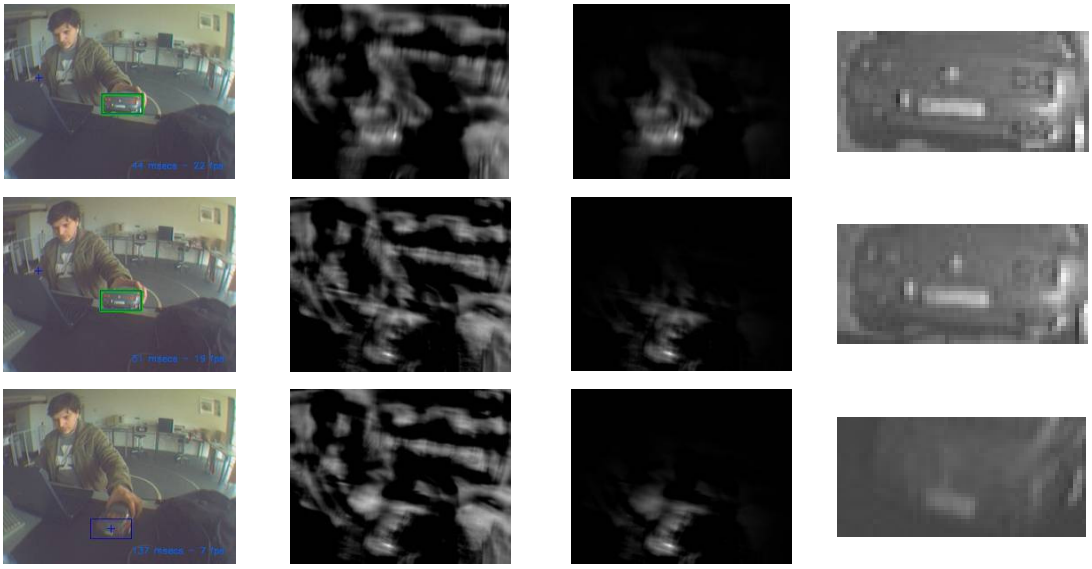

Figure 52. Tracking (1st col.). TMR (2nd col.), conditioned TMR (3rd col.) and Template (4th col.).

Globally, the combination of both techniques was fruitful. The system is now capable of following objects that are identified occasionally by tracking them from the first detection onwards.

8.5 Optical Flow Detection

A motion detection algorithm was also implemented although its integration as an attention mechanism was not fully implemented. Nonetheless, some interesting results were obtained. Figure 53 shows the detected optical flow of a sequence of images. In this sequence the pan and tilt unit was executing the startup routine by covering the entire pan range, i.e. it is not the objects that are moving but the cameras.

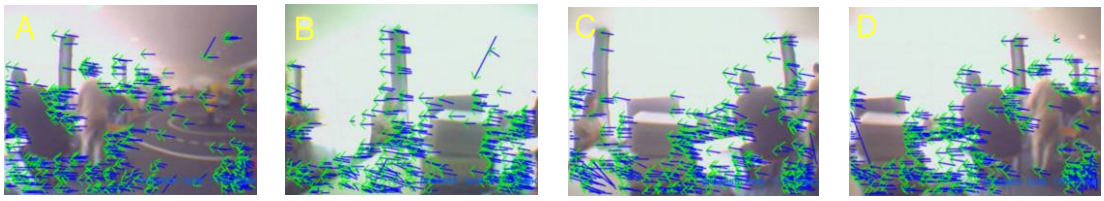

Figure 53. Optical flow detection during a pan and tilt reset routine.

The implemented optical flow algorithm, even tough being a sparse optical flow algorithm, is quite heavy computationally. Therefore, it was necessary to calculate the optical flow of a particular area, i.e., the region were the object of interest is. A simple ROI definition using the template tracking template's position and size information is sufficient to limit the optical flow detection region.

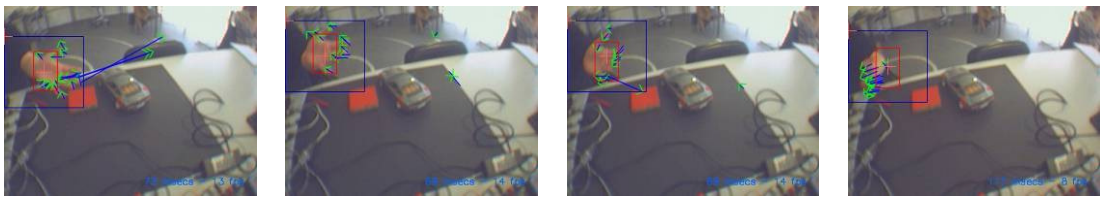

Figure 54. Confined optical flow detection while tracking a hand.

9 Conclusions and Open Issues

This work's primary objective was to study active foveated vision. A pan and tilt unit equipped with two cameras providing both a peripheral and a foveated view was used. The dual camera approach is the most common solution and appears the most feasible one.

The pan and tilt unit is an old model (actually it was a part of the Robuter II robot acquired in 1996) and the latency time between successive commands by the serial line is quite high (80 milliseconds). A better and more up to date hardware would certainly ease the systems interface and possibly the controller's performance. The tracking controller is a simple PID controller, which is not an advanced controller, but the objective was not to focus on this subject but on how to make the most of visual feedback in order to execute a proper foveation control.

Haar features are indeed state of the art detection techniques, which proved to be very fast, allowing for real time execution in both the peripheral and the foveal images, in a standard 1.8 GHz Dual Core Laptop. For this reason they can also be employed as attention mechanisms, by electing a few image ROIs that could then be processed by other slower, yet more effective methods.

The developed tracking technique is not entirely new, but some contributions have been added, namely Gaussian conditioning and especially fast Gaussian computation. In our laboratory, the tracker module usually runs smoothly. Some unimplemented ideas could also improve the tracking module's performance. Namely, to make use of localized optical flow detection to shape the Gaussian filter used for conditioning. For example, in Figure 55 the optical flow represented by the arrows clearly indicates an average movement direction. This information can be utilized to define a Gaussian filter that can have an elliptical footprint instead of circular, as implemented. The ellipse could also be shifted, tilted and skewed by a function of the average optical flow. If the vector is zero, then the elliptic axes would have the same length and the ellipse becomes a circumference, i.e. equivalent likelihood of the object moving (or starting to move) in any direction. If, on the other hand, an object is moving in a given direction axis u (the axis of movement), the ellipse may be skewed by decreasing the v octagonal axis' length and increasing u axis length. The ellipse may also be shifted along the u

axis by repositioning its center in the direction of the movement by a function of the average optical flow magnitude, which works as a measure of the object's speed. The common sense interpretation is related to the inertial property of objects in motion. Objects are more likely to continue on the same movement direction than of changing abruptly the motion vector's direction (Figure 55).

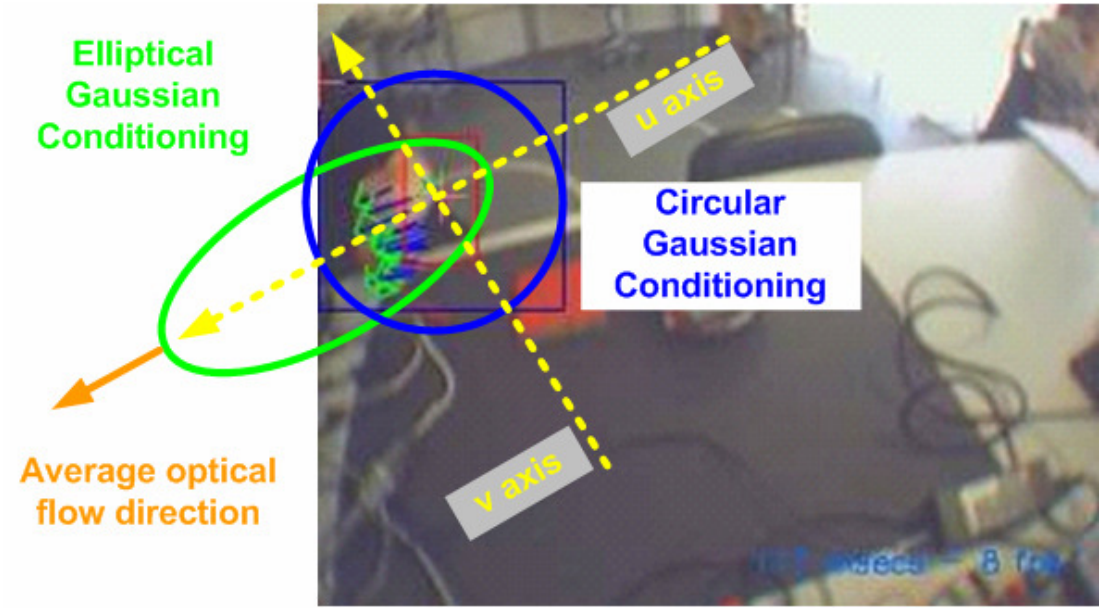

Figure 55. Shaping Gaussian matrix as a function of the optical flow average vector.

There is a strong conviction that this elliptic Gaussian conditioning could further improve the tracking's performance. However it still has not been implemented, especially due to the difficulties in coming up with a technique as transparent as the fast Gaussian computation one, thus enabling real time computation of the elliptical Gaussian filter.

The thread based, distributed, software architecture has proved much more effective than a standard, do-while based cycle, type of program, especially when it comes to asynchronous tasks. It is very useful to be able to define thread priorities, because in fact processes should all be asynchronous. For example, a foveal identification processing takes much longer than an attention based tracking, that, in order to maintain tracking, should not be waiting for the end of other slower processing routines.

Future studies should include distributed foveation control, i.e., to use the foveated image for precise foveation movements. Also, foveated view has not been employed all that much.

Experiences with peripheral Haar / template tracking and foveal Haar detection techniques have been able to haul out the car model license plate, while it is moving around. Then, standard optical character recognition (OCR) applications could read the license plate number. Unfortunately, the used foveal camera's max resolution seems insufficient to hand out a clear enough license plate image. Several open source OCR programs were tested. *Tesseract* and *Gnu OCR* were the ones who performed better.

Figure 56. License plate photo extracted from a foveal view Haar detection (left).
Debug images for Gnu OCR application (right).

Perhaps with better cameras and professional OCR applications a real time license plate recognition system of a dynamic scene (such as cars in motion anywhere on a road track), can be mounted, but for now, the hardware setup allowed to test successfully the systems philosophy, and lead to the conclusion that active, dual camera, foveated vision is a good base to work on several real world vision applications, static or dynamic.

10 References

1 [Asfour, 2006] T. Asfour, *Humanoid Robots: Design Issues and Control*, Presented at International UJI Robotics School on Humanoid Robots, Benicasim, Castelló de la Plana, Spain, 18-22 September 2006

2 [Asfour *et. al*., 2007] T. Asfour, K. Welke, A. Ude, P. Azad, J. Hoeft and R. Dillmann, P. Azad, J. Hoeft and R. Dillmann, *Perceiving Objects and Movements to Generate Actions on a Humanoid Robot*. In IEEE International Conference on Robotics and Automation (ICRA'07), 10-14 April, 2007

3 [Atkeson *et. al.,* 2000] C. G. Atkeson, J.G. Hale, Frank Pollick, M. Riley, S. Kotosaka, S. Schaal, T. Shibata, G. Tevatia, A. Ude, S. Vijayakumar and M. Kawato, *Using Humanoid Robots to Study Human Behavior*, Humanoid Robotics, IEEE Intelligent Systems, 2000;

4 [Azad *et. al*., 2007] P. Azad, A. Ude, T. Asfour and R. Dillmann, *Stereo-based Markerless Human Motion Capture for Humanoid Robot Systems*.. In IEEE International Conference on Robotics and Automation (ICRA'07), 10-14 April, 2007;

5 [Bouget, unknown] J. Bouget, *Pyramidal Implementation of the Lucas Kanade Feature Tracker* Description of the Algorithm. Included into OpenCV distribution.

6 [Chen, 2006] Q. Chen, 2006, Hand Detection with a Cascade of Boosted Classifiers Using Haar-like Features, Discover Lab, SITE, University of Ottawa, May 2, 2006

7 [COG homepage] COG homepage, found at http://www.ai.mit.edu/projects/humanoid-robotics-group/cog/.

8 [Dlagnekov, 2005] L. Dlagnekov, unknown, *License Plate Detection Using AdaBoost*, Department of Computer Science & Engineering, UC San Diego, La Jolla, CA 92093-0114

9 [Fitzpatrick *et. al.*, 2003] Paul Fitzpatrick, Giorgio Metta, Lorenzo Natale, Sajit Rao, Giulio Sandini, Learning About Objects Through Action - Initial Steps Towards Artificial Cognition, Accepted for the IEEE International Conference on Robotics and Automation (ICRA), Taipei, Taiwan, May 12 - 17, 2003;

10 [Hutchinson *et. al*., 1996] S. Hutchinson, G. Hager and P. Corke, 1996, *A Tutorial on Visual Servo Control*, IEEE Transactions on Robotics and Automation, October 1996.

11 [Kaneko and Hori, 2002] T. Kaneko, O. Hori. Template Update Criterion for Template Matching of Image Sequences, *16th International Conference on Pattern Recognition (ICPR'02) - Volume 2, 2002.*

12 [Laschi, 2006] C. Laschi, *Vision and Eye Movements in Humans and Robots,* Presented at

International UJI Robotics School on Humanoid Robots, Benicasim, Castelló de la Plana, Spain, 18-22 September 2006

13 [Lienhart and Maydt, 2002] R. Lienhart and J. Maydt. *An Extended Set of Haar-like Features for Rapid Object Detection*. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

14 [MERs homepage] Mars Exploration Rovers homepage, found at http://marsrovers.nasa.gov/overview/.

15 [Monteiro *et. al*., 2006] G. Monteiro, P. Peixoto, U. Nunes, 2006. Vision-based Pedestrian Detection using Haar-like Features. *Encontro Científico, Festival Nacional de Robótica 2006.*

16 [Oliveira and Santos, 2007] M. Oliveira and V. Santos, *Combining View-based Object Recognition with Template Matching for the Identification and Tracking of Fully Dynamic Targets*, The 7th Conference on Mobile Robots and Competitions, April 2007

17 [OpenCV documentation] OpenCV version 0.9.7 cxcore and cvaux documentation, *Included into OpenCV distribution*.

18 [Rothenstein *et. al*., 2006] A. L. Rothenstein, J. K. Tsotsos, *Attention links sensing to recognition,* unknown, 2006;

19 [Shi and Tomasi, 1993] J. Shi and C. Tomasi, 1993, Good Features to Track, unknown.

20 [Stavens, 2007] D. Stavens, Introduction to OpenCV, Stanford Artificial Intelligence Lab. Found at *http://robots.stanford.edu/cs223b05/schedule.html*., in January 2007.

21 [Ude *et. al*., 2004 a] A. Ude, C. G. Atkeson and M. Riley, *Programming Full-Body Movements for Humanoid Robots by Observation*, Robotics and Autonomous Systems, vol. 47, 2004, pp. 93-108, 2004;

22 [Ude *et. al*., 2004 b] A. Ude, C. Gaskett, G. Cheng, 2004, *Support Vector Machines and Gabor Kernels for Object Recognition on a Humanoid with Active Foveated Vision*, Proceedings of 2004 IEEEIRSI International Conference on Intelligent Robots and Systems, Sendai Japan.

23 [Ude *et. al*., 2006] A. Ude, C. Gaskett, G. Cheng, Foveated vision systems with two cameras per eye, Proc. IEEE Int. Conf. Robotics and Automation, Orlando, Florida, May 2006, pp. 3457-3462, 2006;

24 [Viola and Jones, 2001] P. Viola, M. Jones 2001. *Rapid Object Detection using a Boosted Cascade of Simple Features*, Conference on Computer Vision and Pattern Recognition 2001.

25 [Wolfe, 2003] J. Wolfe, 2003, *Moving towards solutions to some enduring controversies in visual search*, TRENDS in Cognitive Sciences Vol.7 No.2 70 February 2003