INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

From Pixels to Objects - Enabling a spatial model for humanoid social robots

Dario António Bacellar Figueira

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Júri

Presidente: Doutor Carlos Silvestre
Orientador: Doutor Rodrigo Ventura
Co.Orientador: Doutor Manuel Cabido Lopes
Vogal: Doutor Alexandre Bernardino

Setembro de 2008

Resumo

Este trabalho adiciona o conceito de objecto ao sistema de atenção baixo-nível do robot humanóide iCub. Os objectos são definidos como conjuntos de pontos-chave SIFT. Quando o robot encontra um objecto desconhecido, a uma distância pequena dos seus olhos, este guarda o conjunto de pontos-chave existentes a pouca distância, usando para isso a percepção de profundidade. Quando objectos previamente guardados são vistos pelo robot, são reconhecidos e mapeados em um sistema de coordenadas egocêntrico. Este mapeamento é persistente no sentido que a posição dos objectos e sua identidade permanecem em memória mesmo quando estes não se encontram no campo de visão do robot. Os pontos-chaves são guardados e reconhecidos de forma totalmente automática. Este trabalho estabelece as fundações para ligar o sistema de atenção baixo-nível com a informação alto-nível, orientada a objectos, proveniente de utilizadores humanos.

Palavras-Chave: Reconhecimento de Objectos, Cálculo de Profundidade; Visão Stereo; Mapa de Saliência.

Abstract

This work adds the concept of object to an existent low-level attention system of the humanoid robot iCub. The objects are defined as clusters of SIFT visual features. When the robot first encounters an unknown object, found to be within a certain (small) distance from its eyes, it stores a cluster of the features present within an interval about that distance, using depth perception. Whenever a previously stored object crosses the robot's field of view again, it is recognized and mapped into an egocentrical frame of reference. This mapping is persistent, in the sense that its positions are kept even when the objects are not in the robot's field of view. Features are stored and recognized in a bottom-up fashion. This work creates the foundation for a way of linking the bottom-up attention system with top-down, object-oriented information provided by humans.

Keywords: Object Recognition, Depth Perception; Stereo Computer Vision; Salience Map

Acknowledgments

Firstly, i would like to express my gratitute for the steady support and insights given by my supervisors Professor Rodrigo Ventura and Doutor Manuel Lopes. I know that their support was a key-factor to the completion of this work. Then i would like to gratefully acknowledge the contribution of Alexandre Bernardino's reviews, comments and insights in the matter of Depth Perception. Furthermore, I would like to acknowledge, with thanks, the help of Jonas Ruesch regarding the underlying attention system used for this work [1].

Contents

List of Tables

List of Figures

Acronyms

SIFT - *Scale Invariant Feature Transform*.
IOR - *Inhibition-Of-Return*.
FPS - *Frames-Per-Second*.
SURF - *Speeded Up Robust Features*.
GUI - *Graphical User Interface*.

Chapter 1

INTRODUCTION

"There is an uncertainty relationship between truth and clarity" - Niels Bohr

For humanoid social robots to autonomously interact with our human structured environments, they must be endowed with the capacity of perceiving objects as such, as separate entities of the environment. However, robot sensory apparatuses only provide raw sensory data. Taking vision as a sensor, how can it bridge the gap between raw pixels and the concept of objects? Moreover, how can it realize their relative positions within the surrounding environment, even when they are temporarily out of the cameras field of view?

The system presented here addresses these problems, by taking a bottom-up, developmental approach. The developed module builds upon an existing low-level attention system [1]. That work provides a salience map with respect to a robot-centric coordinate system (ego-sphere). This saliency map, together with a inhibition of return mechanism (IOR), allows the robot to saccade from salient point to salient point. However, these salience points correspond to preattentive features, e.g., movement, color, and shape, not incorporating the concept of object.

The goal of this work is to endow the robot with the capability of learning and recognizing objects, mapping them in its surroundings, and thus enabling the robot to change its attention focus between recognized objects. By integrating this capability into the existing architecture, the attention module will be able to acknowledge the salience of known objects, because they are recognized as such. Moreover, the capability of recognizing objects *per si* paves the way for higher level modules, to implement complex cognitive functions, such as language, linking symbols to actual physical things in the environment.

So, although the practical objective of this work is to enable a robot to look at what we consider being important objects, the ultimate goal is to study cognition. Cognition is the mental process of knowing, including processes such as perception, awareness, reasoning and judgment. Although since before 1965's TV series and movies, like "Lost in Space", science-fiction has for many years presented us with robots that can navigate their environments, recognizing and manipulating objects, robots these days are still not truly cognizant. With this

work we aim to come one step closer to that reality. A reality that in practical terms will mean a deeper involvement of robots in our lives, taking over repetitive or dangerous task, assisting our elderly and or simply providing friendly company and barter as communications become smoother and smoother between humans and robots.

There are at least two major ways of approaching cognition: the top-down approach, and the bottom-up one. With the top-down approach we start with the deliberative levels [2] and go down from there until we reach the raw sensors levels. In the Bottom-Up approach [3] we start with perception, extracting ever more abstract information from the data until we reach the deliberative levels. In this work we build upon the Bottom-Up approach, our goal being to lay another brick on this approach, creating an abstraction level just above the vision primitives' level, introducing symbols in the architecture where before there was only pixel values and per-pixel saliency metrics.

Neuroscience and robotics have had closing bonds over the years, as the second provides the first with models in which to test theories, and the first provides the second with insights in how to help robots interact with their environments. With our work we are creating yet another model that will help robots interact with their environment and at the same time give grounds for neuroscience to delve further in the realms of robotics.

The robot considered here is the iCub humanoid robot[1]. However, just the head and torso modules were employed in the experimental part of this work. While the head has 6 degrees of freedom, the torso is fixed. Being the coordinate system anchored to the torso, it is a robot-centered coordinate system of the world.

This introduction gives the motivation for this work and presents an overview of the problem at hand and how it was generally solved. This then sets the stage for the actual presentation of the solution developed in this master thesis.

1.1 Modeling Space

To model space, and to fulfill the goal of helping an agent commute automatically its attention focus from recognized object to recognized object, we chose to model the environment as a salience map [4]. We project the surrounding space and objects into a sphere centered in the neck of the robot, an egocentric sphere or ego-sphere, as defined in [1] (Figure 1.1).

A spatial model for the robot is here understood as a model representing the environment surrounding it, namely the known objects, together with their physical positions relative to the robot.

In this work we add to the spatial saliency map implemented in [1] and endow the system with an interpretation of its surroundings. The system now maps known visual objects, so it can know where they are after looking away. Instead of the short-time memory of the previous works [1] [4], the system remembers where important objects are at longer time scales.

We implemented an algorithm to automatically store in a database new objects that are close to the eyes of the robot (the cameras). To do so, we

[1] http://www.robotcub.org/

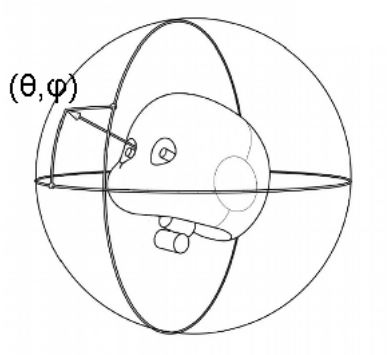


Figure 1.1: Ego-sphere : a spherical map of the surroundings; Horizontal axis: azimuth; Vertical axis: elevation.

compute a depth map [5] [6] of the image to determine if something is in close proximity and under the robot's scrutiny. If so, its representation is stored to a database to be later remembered and recognized.

We chose the Scale Invariant Feature Transform (SIFT) [7] algorithm to enact our recognition. We also used this algorithm to match corresponding points in pairs of stereo images, thus computing the disparity or depth of these points. By using this algorithm to detect distance we defined a new object as a cluster of SIFT features in close proximity to the cameras, while already stored objects are detected continuously in the input images by the SIFT algorithm. Finally, the recognized objects in the robots field of vision are inserted in the egocentric map.

In the area of implemented robot architectures that recognize objects in the surrounding very little is published. In [8] a robot in an office environment uses eight colors and size (in pixels) to describe everyday objects on top of desks. In this work we implement a richer way of describing objects.

1.2 Contributions

The novel contribution of this work is:

- The use of SIFT features to calculate disparity;
- Integration of the open source modules SIFT, and ego-sphere;
- Addition of an Object Map to the ego-sphere's Visual Map and Auditory Map.

1.3 Structure

In the next chapter we shall describe our spatial model in a general sense, leaving out the implementation details that are discussed at length in the third

chapter. Then, in the forth chapter, we describe an experiment in which the functioning of our work is illustrated and evaluated. Finally, we finish with our conclusions and future work.

Chapter 2

ARCHITECTURE

In this chapter we describe the modular architecture employed.

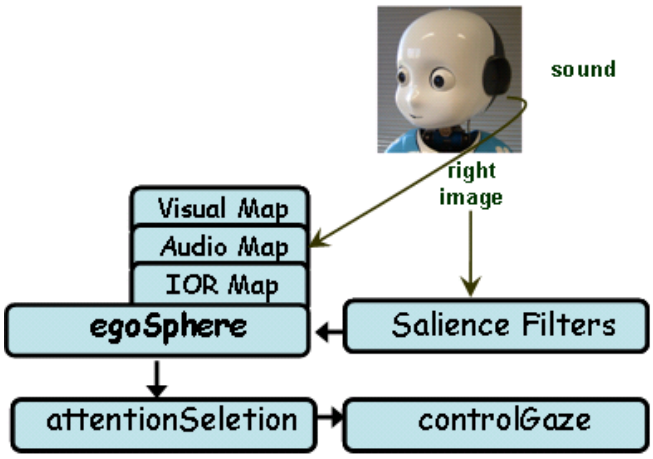


Figure 2.1: Previous system architecture: Salience filters pre-process the image; the ego-sphere aggregates the visual and sound saliency information into a multimodal ego-centric frame of reference; the attentionSelection picks the maximum from the ego-sphere; and the controlGaze sets the robot in motion to look at that maximum.

Previously, Jonas *et. al.* introduced the architecture, displayed in Figure 2.1. It has several interconnected modules to form a sensing-deliberation-actuation chain. It is motivated on the Itti and Koch model [9] where stimuli from various sources is represented and combined in a single salience map. Likewise, the salience filters process the visual information to produce salience maps that emphasize several aspects of the image, *i.e,* color, movement, intensity, and the presence of faces. These maps are combined, in the ego-sphere, with

sound stimuli captured by the robot's microphones to form a single, multimodal, egocentric saliency map. In addition, the ego-sphere keeps a short-memory of the previously looked upon positions, in the form of an inhibition-of-return map (IOR) [1]. This map has values from one to zero, where one means unvisited, and zero implies previous exploration. The IOR map multiplies pixel-by-pixel the final egocentric saliency map, reducing the interest levels of the already observed locations. This map "decays" (returns to one) over time, therefore a region of inhibition (a circular area of zero values) is added to the map only after a pre-set number of frames of continuous scrutiny by the robot, preventing a rapid come-back of the robot's attention focus due to the decay. The resulting egocentric map is processed and its maximum is selected as the point that wins the robots attention next. Finally the robot's visual focus is switched to the new selected attention point.

The resulting behavior is the capability of the robot to fully explore its environment, in a natural [10] non-preprogrammed fashion, without being stuck on the absolute saliency maxima of the salience maps.

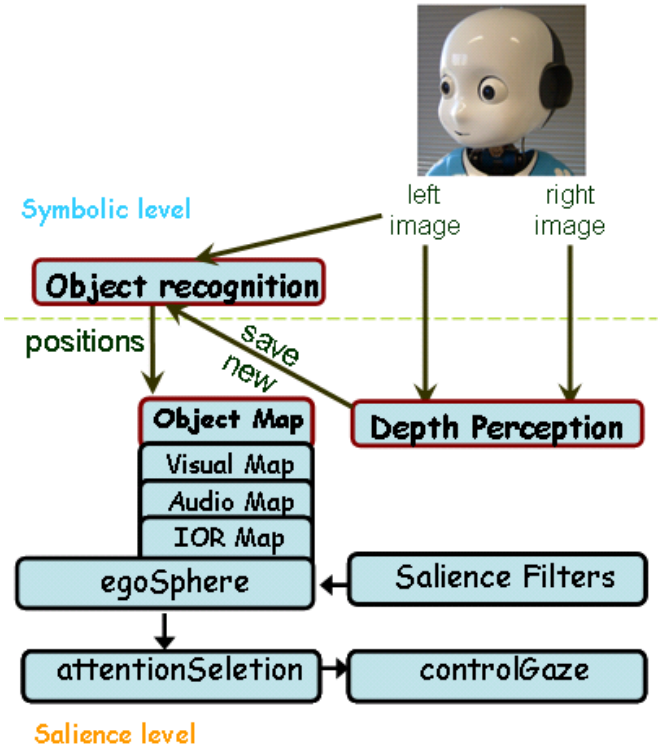


Figure 2.2: System architecture, after introducing the modules presented here (red border): object recognition and depth perception.

Our work, highlighted in Figure 2.2, adds a level of abstraction to the architecture: the concept of object, *i.e.,* the agent can now learn what are and recognize objects. But since this work is to be used by an autonomous agent, the problem of automatizing the decision of *when* and *what* to save to the

database, arises. This problem is solved by using a special depth filter that calculates the depth, and segments the object in the image. When an object is detected in close proximity it triggers the creation of a new object in the memory, if it's not already in memory.

The previous architecture provides the robot with a rapid decaying pictoral map of the surroundings, which prompts the robot to a natural exploratory behavior. Our addition adds general objects to the list of attention-capturing things by the robot. We expect the robot to still look at, for instance, rapid movement, or sound sources, and when the environment is relatively calm, we expect it to return its attention to the known objects in the area.

Although at the present there is no input to describe the objects stored, the objects are stored just with a "label" that is simply a number (the order of appearance). When the robot has the possibility to ask humans around him for names to the objects it is discovering and storing, new possibilities for the creation of other modules that rely on the existence of these named objects arise.

In sum, our work receives images from two stereo cameras, detects and learns new objects in proximity, recognizes already learnt objects in the left image (arbitrarily chosen, either left or right would do) and outputs the positions of the objects recognized in the present frame. This is seamlessly integrated with the present architecture by adding salience peaks in the ego-sphere where objects are located. By doing so we take advantage of the implemented architecture, allowing the robot to look at the objects it has learnt so far. Next chapter provides the reader with the details on how this new capabilities were implemented.

Chapter 3

IMPLEMENTATION

In this chapter we provide the details to perform object segmentation and recognition.

Many different approaches have been used in computer vision to enable recognition, for instance, receptive field histograms has been used successfully by Schiele [11], although it does not work with cluttered or occluded images. Many have benefited from David Lowe's Scale Invariant Feature Transform (SIFT) [7], while others have used Speeded Up Robust Features (SURF) [12], a technique that is similar to the SIFT algorithm in many ways, faster to compute and match, albeit with lower recognition rates. Our approach, the SIFT algorithm, solves both problems of object segmentation and recognition. We chose SIFT for reasons such as invariance to scale and excelling in cluttered or occluded environments (as long as three SIFT features are detected, the object is recognized). And while the SURF algorithm is faster and performs generally well, SIFT's recognition results are still superior [13]. The setback about using this algorithm is that it takes a lot of processing time, the most efficient implementations are not able to run it in real time (24 (FPS)) [14].

3.1 SIFT

SIFT [7] is an algorithm that extracts, features from an image. These features are not only scale invariant features, but also affine transformations invariant (*e.g.,* rotations invariant). Furthermore, they are robust to changes in lighting, robust to non-extreme projective transformations, robust up to 90% occlusion and are minimally affected by noise.

To find points that are repeatably detected and scale-invariant David Lowe looked for the *extrema* of a space invariant to scale (called scale-space). This scale-space is approximated by the computation of the Laplacian of the image convoluted with a gaussian function of variable scale and is divided into octaves (doublings of the scale). The Laplacian is then also approximated by the efficient calculation of the difference of such convoluted images, as illustrated in Figure 3.1.

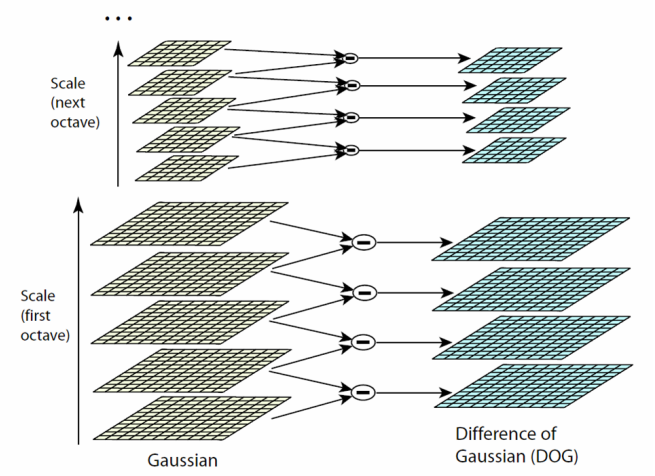


Figure 3.1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.

After having detected the *extrema* of the generated Difference-of-Gaussians, these points are encoded into rotation invariant, robust to changes in illumination, robust to projective changes in perspective, and robust to noise descriptors. These descriptors are created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left of Figure 3.2. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right of Figure 3.2, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. The figure shows a 4x4 descriptor array computed from an 16x16 set of samples.

The descriptors are made rotation invariant by computing an orientation to the detected *extrema* from local image pixel differences, and then making all the gradient orientations relative to this point orientation. Furthermore, the recognition is made robust up to 90% occlusion by storing the relative positions between detected points and needing only the detection of three such points for reliable recognition.

We use the SIFT algorithm to enable the recognition in our system because of all these powerful characteristics. Due to the nature of the SIFT features, its second drawback is the inability to extract features from a texture-less object, as is shown in Figure 3.3, few or no features, in yellow dots, are found in areas with homogeneous color, such as on the table, on the ground, or on the wall. Moreover, since this algorithm was designed to work with planar surfaces, to be able to correctly recognize 3D objects, several snapshots of the objects

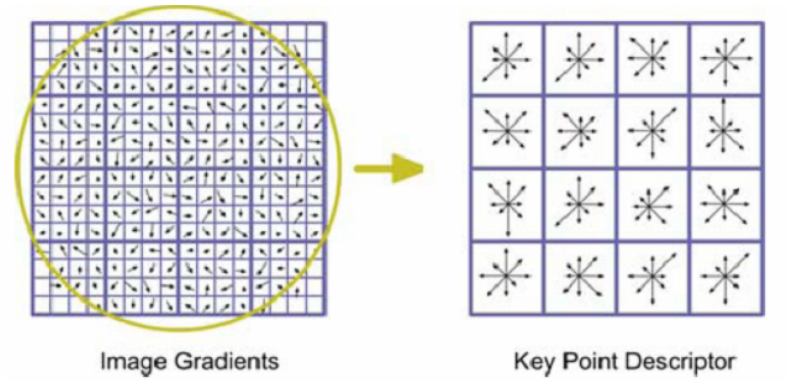


Figure 3.2: A keypoint descriptor is computed from the histogram of the gradient directions in 16x16 windows, shown on the left, weighted with by a gaussian, indicated by the overlaid circle, centered in the detected maximum or minimum, forming a 128 dimension vector from the orientation histograms shown on the right.

in different perspectives need to be taken. An object is represented in the database by several sets of SIFT features, together with the relative positions of the features to each other. Each set represents a different snapshot of the same object.

The SIFT Algorithm was created by Lowe [7] but the implementation was not publicly disclosed. The actual implementation of the SIFT algorithm, in C++, that we use in our work, was written by Hess[1].

3.2 Depth Perception

The problem of automatizing the decision of *when* and *what* to save to the database is solved by calculating the depth, in the image, of the extracted features. For that, we assume two parallel and horizontally aligned cameras, described by the pin-hole model[15]. To calculate depth we require the knowledge of these necessary camera parameters:

- the focal length f;
- the distance between the two cameras β;
- and how much one pixel corresponds in distance in the camera sensor γ.

Also, we need to correctly match a point of the environment, seen in both stereo images, with pixel coordinates (x_1, y_1) in the first image and (x_2, y_2) in the second. The point's coordinates in the camera references are (X_1, Y_1, Z_1) for

[1] http://web.engr.oregonstate.edu/∼hess/

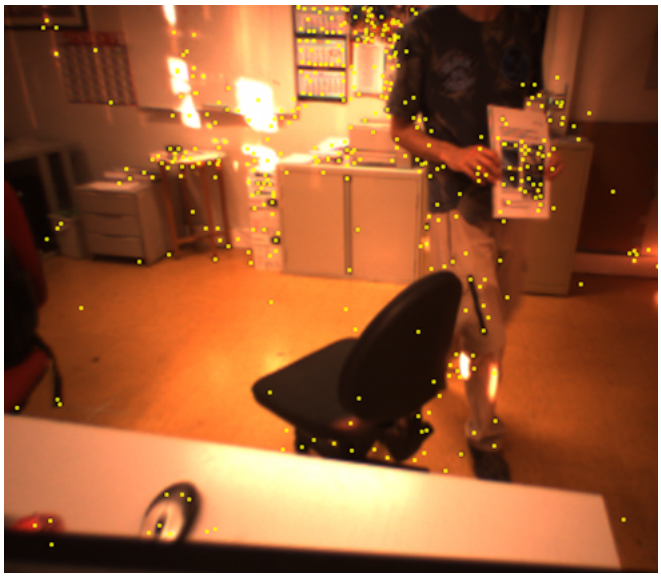


Figure 3.3: Example of SIFT feature extraction; the yellow dots correspond to the extracted features positions.

the first camera and (X_2, Y_2, Z_2) for the second. Armed with all this information we can calculate how far away the matched point is (depth Z) by the following equation, illustrated in Figure 3.2:

$$\begin{cases} \gamma x_1 = f\frac{X_1}{Z_1} \\ \gamma x_2 = f\frac{X_2}{Z_2} \end{cases} \equiv \begin{cases} Z_1 = Z_2 = Z \\ Z\gamma(x_1 - x_2) = f(X_1 - X_2) \\ X_1 - X_2 = \beta \end{cases} \equiv Z = \frac{f\beta}{\gamma(x_1 - x_2)} \quad (3.1)$$

Therefore the common way to determine depth, with two stereo cameras, is by calculating disparity. Disparity is simply the subtraction of the 2D coordinates of corresponding points (from one image to the other, in this work from the left to right). A dense disparity map consists in a matrix, of the size of the images, in which the value of each coordinate indicates how many pixels had to be shifted until we find the match of our current pixel in the other image. Some ways to match corresponding points can be: pixel by pixel probabilistic matching with a Bayesian formulation [6]; or histogram matching of the neighborhood of the pixel [16].

Our first attempt to use depth perception used the algorithm described in [6] to compute a dense disparity map. This approach gives a detail of one pixel resolution illustrated in Figure 3.5(a) and 3.5(b), that we do not require. To determine if there is one very close region or blob, and then segment it, a sparse disparity map is sufficient. So, after determining that our first approach required very fine calibration of the cameras and rectification of the stereo images, we decided to pursue the sparse disparity map approach.

The SIFT features, with their invariance and robustness, enact a way to solve the problem of matching corresponding points in stereo images. We

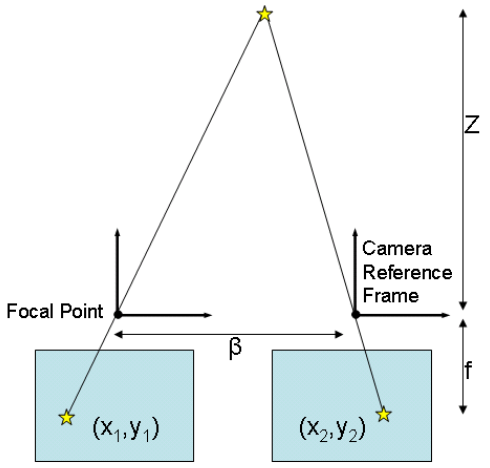


Figure 3.4: Simple camera model to calculate depth, f: focal distance, β: distance separating the parallel cameras, γ: pixel-to-meter ratio in the camera sensors, (x_1, y_1): pixel coordinates of point we wish to calculate depth, Z: depth.

(a) Original image (b) Depth-Perception image

Figure 3.5: Depth-Perception results after exceptional camera calibration

generate a sparse disparity map by extracting the SIFT features from stereo images, and look for matches between both sets. Usually the SIFT features are matched to the database of stored features belonging to objects. Here we are using the same mechanism but using one stereo image as the database where the other stereo image tries to find SIFT matches. Assuming that the robot's eyes are roughly aligned in the horizontal (*i.e.,* mis-alignment of under 30 pixels) we compute the disparity between matching features from the pair of stereo images. Matches that have a high horizontal disparity are assumed to be part of an object in close proximity to the robot's face and matches with low horizontal disparity belong to the background. Matches with high vertical disparity or negative horizontal disparity are outliers, *i.e.,* bad matches.

Using a batch of real images, sampled in Figure 3.6, we get the following results summarized in Table 3.1 and fully displayed in Tables A.1 and A.2 of the Appendix. In the first column we have the number of features detected in the left image, in the second column we display the number of matches found between the left image features and the right image features, while on the third column we show how many of those matches were bad matches, outliers. In the last row we display the percentage of matches and outliers, averaged for all images.

Table 3.1: Summary of Disparity Matching results.

images	features	matches	outliers
total	27564	9545	165
average %	-	34,1	0,6

Figure 3.6: Sample of images used to generate the Disparity Match results.

We operate on the assumption that the agent will have arms and hands, and will use them to manipulate objects when exploring its environment. With that in mind, the threshold distance to save new objects is related to the arms length, or the expected object position relative to the agents eyes when a new object is being inspected by the robot.

Comparing the extracted features of different images in different resolutions, a threshold for the horizontal disparity T_h was found empirically to be the width of the image divided by 6,4. Moreover, the vertical threshold T_v to determine outliers was also empirically found to be the height of the image divided by 16.

If the matches between detected features are "close enough" (each match having its horizontal disparity greater than the threshold), the group is stored to the database as a new object. Only the features that are correctly matched between the two stereo images with high horizontal disparities are stored, because only these features are believed to belong to the close object. For instance, the features from the background being seen by a hole in the object are discarded.

Figure 3.7(a) and Figure 3.7(b) show in blue crosses the features that are correctly matched between the two stereo images as being the same, and therefore store in the database as a new object (only if not recognized as part of an already known object).

3.3 Recognition

To decide upon the presence of an object in the image, SIFT relies on a voting mechanism that is implemented by a Hough transform. Defining pose as the position, rotation and scale of an object, each match votes on an object-pose pair in the image. The Hough transform is computed to identify clusters of matches belonging to the same object. Finally, a verification through least-mean-squares is conducted for consistent pose parameters along all matches (verifying if the matches found have correct relative positions). This part of the algorithm presented in [7] was implemented by Damas [17].

After experimenting with several objects, having the robot store them to the database and then holding them farther and farther away, the algorithm is able to recognize them until roughly two meters away, when the number of extracted features is too few for object recognition. As an example, in the learning process many features are stored in the database, shown in blue crosses in Figure 3.8(a). But when the object is far away only a few are extracted, as shown in yellow in Figure 3.8(b). Fewer still, depicted in purple filled squares, are matched to the database. But still, those few are enough to recognize the object (encased in a red frame) in Figure 3.8(b). Also after experimentation with several resolutions, we confirmed that lower resolutions lead to less features being extracted and consequently poorer recognition, but, on the other hand, the higher the resolution the longer the extraction process takes. We, then, used images of 640x480 resolution as a trade-off between this two issues. Table 3.2 illustrates this trade-off, in the first column we have the image resolution, in the second column we display an average of the number of features extracted from such image sizes, and in the third column we show the average of the time these experiments took taken on a same computer.

(a) Left image

(b) Right image

Figure 3.7: Matching SIFT features in a stereo images: features in yellow; matched features in blue.

3.4 Database and Mapping

New objects are stored into a database, which links object identifiers (labels) to sets of SIFT features. Each set contains a label, if the labels are the same then the different sets are considered to be of the same object. When known

Table 3.2: Image size versus SIFT processing time and number of features.

Image size	features	time (ms)
320x240	1201	800
640x480	4474	1470
800x600	6828	1920
1024x780	10769	2630

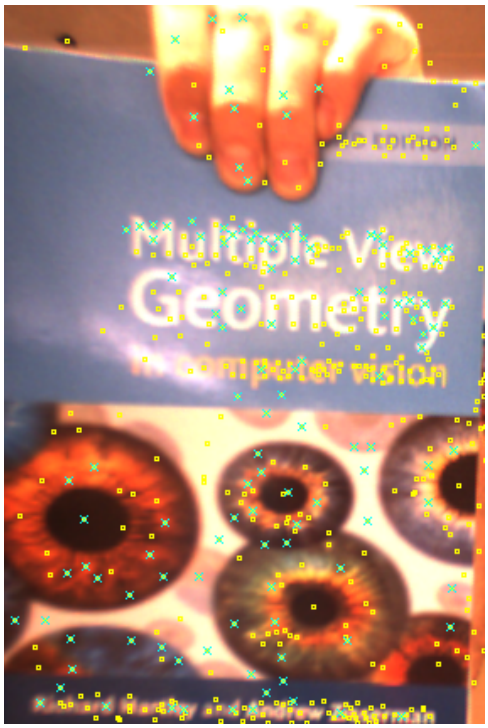
objects are encountered in the environment, their positions are mapped into the ego-sphere [1]. Thus, an object representation is stored in the database, while their positions, whenever recognized by the robot, are represented solely in the ego-sphere.

The egocentric saliency map used for attention selection is obtained from the composition of several specialized maps: a visual map (M_{vis}), containing saliency information extracted from visual features (e.g., motion, color), and an auditory map (M_{aud}), obtained from sound stimuli captured by the robot's microphones [1]. These maps cover the entire space surrounding the robot with a spherical coordinate system (azimuth $\vartheta \in [-180\,^\circ; 180\,^\circ]$ and elevation $\varphi \in [-90\,^\circ; 90\,^\circ]$). The saliency information stored in these maps is continuously decayed ($M_{vis}(k+1) = d_{vis}.M_{vis}(k)$), according to a forgetting factor ($d_{vis} = d_{aud} = 0,95$). This factor coupled with a maximum frame-rate of 20 FPS, yields a half-life of less than a second, 14 frames.

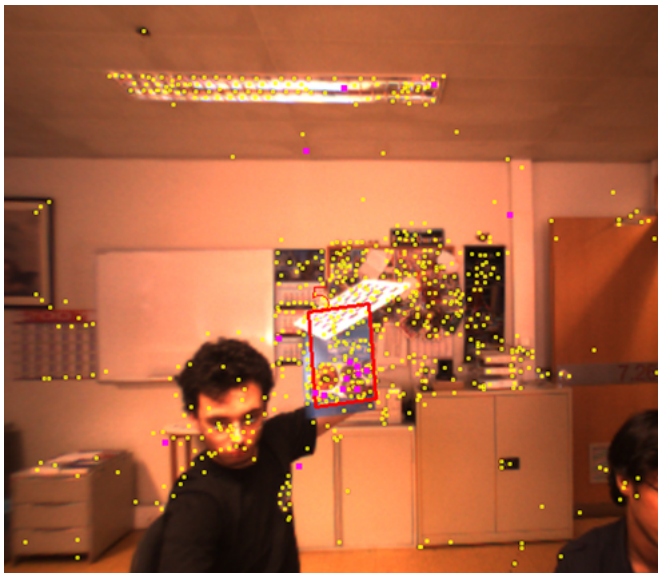
In order to integrate the system described in this paper with the attention selection mechanism, the recognized objects are projected onto a third map (an object map M_{obj}). This map, combined with the other two, contributes for the ego-centric saliency map ($M_{ego} = max(M_{vis}, M_{aud}, M_{obj})$. As the others, this map is also subject to a continuous decay of its information, albeit with a much longer forgetting factor ($d_{obj} = 0,9995$). How long should the robot remember where objects of interest were? How long before such information is unreliable? Those are not trivial questions to answer. Therefore, to fulfill the practical goal of this thesis, of helping a robot switch its attention focus from recognized object to recognized object, even when such objects are not continuously in the robots field of view, this simple decaying memory with such a forgetting factor, that gives an half-life of little over one minute, is sufficient.

To verify the repeatability of the mapping coordinates (x, y) of an object in the image to coordinates in the ego-sphere (ϑ, φ) several experiments were conducted. An object was left on the table in front of the robot, while the robot's head slowly turned. From these experiments we conclude that when the object is away from the limits of the image, it is repeatedly mapped to the same location with an error under one degree elevation and two degrees azimuth. When on the verge of leaving the image, the error in mapping jumps up to two degrees elevation and four degrees azimuth.

The objects are mapped into the ego-sphere as gaussian peaks in salience. To cope with the mapping error, the gaussian parameters used were $\sigma_\vartheta = 30$ and $\sigma_\varphi = 15$.



(a) Object stored to database; Yellow: SIFT features, blue: SIFT features stored to the database



(b) Red: recognized object, yellow: SIFT features, purple: SIFT features matched with the database

Figure 3.8: Recognition of stored object in the environment

Chapter 4

RESULTS

In this chapter we describe the resultant robot behavior due to the contribution of our work.

One of the experiments set up to show the correct recognition and mapping consisted of:

- Showing two objects, in turn, for it to learn and store to the database, a book and a magazine cover shown in Figure 4.1;
- Then setting them up in front of him separated wide enough so that when the robot's attention would be on one of this objects its field of vision will not cover the second object as well;
- Observing the resulting behavior.

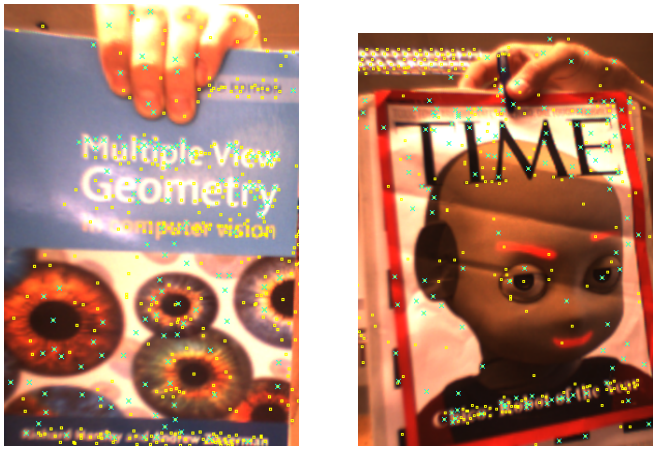
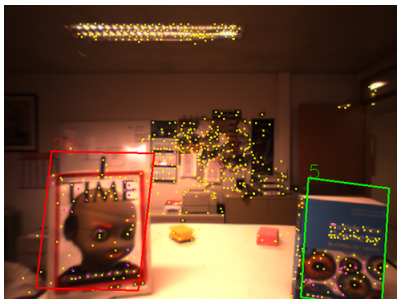


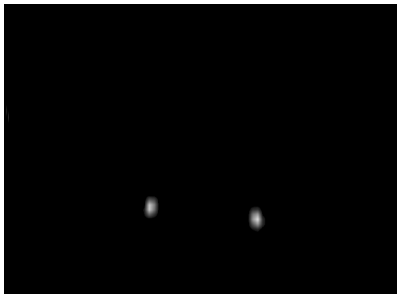
Figure 4.1: Stored objects in database.

The robot, upon recognizing the both previously known objects in Figure 4.2(a), adds interest peaks in the ego-sphere [1] (Figure 4.2(b)). The Attention Selection [1] module tells the robot where to look, and it fixes its attention in the first

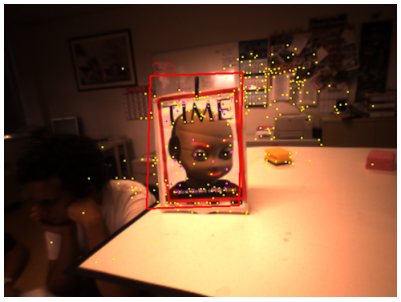
object (Figure 4.2(c)). After some time of intense scrutiny a inhibition region is added to the Inhibition-of-Return map [1] (Figure 4.2(e)) which nudges the attention selector to continue exploring the environment of interesting points, the other recognized objects. The Attention Selection module then indicates the robot to look at the now most salient region in the memory, the second blob in the ego-sphere, the second object in Figure 4.2(f).



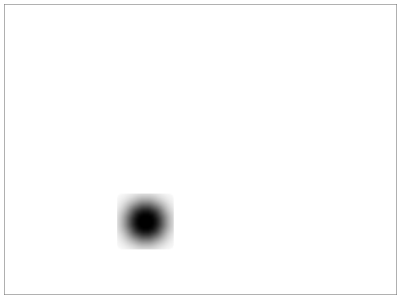
(a) Recognizing objects in the environment. Red: recognized first object, green: recognized second object



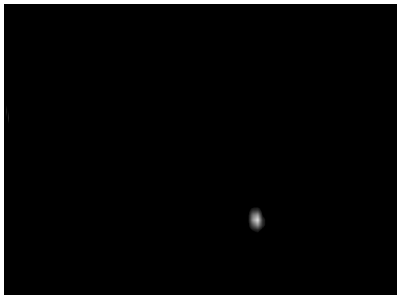
(b) Mapping recognized objects into ego-sphere salience peaks. White: recognized objects



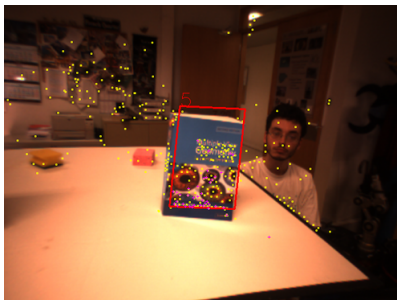
(c) Looking at maximum salience point of the ego-sphere. Red: recognized object



(d) Becoming "bored" of staring at the same point after some time. Black: Inhibition-of-return blob



(e) Inhibiting salience of visited location. White: salience peaks



(f) Looking at current maximum salience point of the ego-sphere. Red: recognized object

Figure 4.2: Example of the exploratory behavior.

Chapter 5

CONCLUSION

The goal of the presented work was to endow the robot with the capability of learning and recognizing objects, mapping them in its surroundings, and thus enabling the robot to change its attention focus between recognized objects.

We implemented a spatial model of the environment, we mapped the recognized objects of the surroundings and introduced salience peaks on the egosphere [1] in their positions. The robot iCub now also explores its environment focusing its attention in the recognizable objects around him.

With the long-term memory implemented, the objective of making this spatial model non-dependent on the robot's field of vision is achieved. As depicted in the results, the robot returns its focus to previously observed objects that were at the moment not in its line of sight.

By using the SIFT algorithm we were able to make this work as robust as currently possible to adverse visibility conditions.

Many recognition algorithms, *i.e.,* [18] [19] [8], to detect certain shapes and certain colors, are already in use. We conclude that the SIFT algorithm chosen to enact recognition complements this many other algorithms in the way that it is unable to detect simple colored objets like a bland blue ball, while successfully recognizing what was before out of recognition-reach: complex shapes and drawings.

This work made a novel contribution by using the SIFT features to calculate disparity and by integrating the SIFT algorithm with the egosphere to recognize objects. We also improved the ego-sphere by creating a new map only for objects.

The software implementation has been made publicly available in the iCub software repository.

5.1 Future Work

Although the objectives were achieved, we were not able to make this algorithm work in real-time due to the heavy computation required for the extraction of SIFT features and matching.

Since it's publication in 2004, the SITF algorithm has enjoyed vast popularity. Many implementations have been developed, and many of these have focused on speed. Still, the algorithm itself is as efficient as it can be and does not suffer further improvement. Therefore, in the future, to decrease the high computation time of extracting the SIFT features, the obvious approach is to use a Graphics Processing Unit, as was done by Wu [14].

The current method of choosing which features to save, finding the matches between the two stereo images and then calculating their disparity, only captures, as shown in Table 3.1, roughly one third of the features extracted. Before trying this method we experimented with dense depth perception maps, trying to extract "big", "close" blobs. The state-of-the-art in dense depth perception was found to be very noisy and needing exceptional calibration of the cameras. We gave up on the dense disparity maps because of these reasons but in the future, if depth-perception can be reliably and easily calculated for two stereo cameras, it could wield better results for us, because it would enable us to save all the extracted features on the "close blob", the close object.

At the present time we only use the positions of the recognized objects in our work. The also provided names of the objects, and average disparity of the recognized matched features, offer the foundation for plenty ideas for future work. One avenue is tracking specific objects in the environment. Another, is the search for specific objects in the surroundings to ascertain its existence or not. Additionally, the estimation of how far an object actually is, in absolute terms, is another avenue of possible work to be done.

Finally, during the course of this work it was verified the need to save several snapshots of an 3D object of its different sides to be able to correctly recognize it in any given position or orientation. As this was verified, a difficulty arose: "how does the agent know that this new snapshot of an object is a new side of an already known thing?" Three avenues for future work open:

- One, if the robot has arms, while exploring its environment it is expected to pick up objects and closely examine them. Then when saving the first view of a new object in its grasp, it will know it is grasping the new object and will rotate it itself, always knowing it has not relinquished hold of its source of interest and saving the following snapshots under the same name tag as the first snapshot.

- The second possibility is using the algorithm developed by Stosic[20], that while tracking a set of features in a 3D object that is rotating, recognizes the rotation movement. So, when the agent saves the first image of a new object it would expect it to rotate, would track the already known features for that movement, and would save new snapshots at every X angles of rotation.

- A third and complementary way to both previous solutions is the agent simply asking the users. Take the example of the RobotCub project, the objective is to create the robotic equivalent of a 2 year-old child. A child doesn't learn by herself alone, she needs teachers, or at least someone to answer when she asks "what is this?". Under this light, more work could be done integrating a speech synthesizer to allow the agent to actually ask this questions out loud. And additionally a speech recognizer to

enable the users to simple answer the agents questions out loud as well, allowing a smoother integration of the robot in its social environment.

Bibliography

[1] J. Ruesch, M. Lopes, A. Bernardino, J. Hornstein, J. Santos-Victor, and R. Pfeifer, "Multimodal saliency-based bottom-up attention, a framework for the humanoid robot icub," *IEEE International Conference on Robotics and Automation Pasadena, CA, USA, May, 2008*, May 2008.

[2] D. Vernon, G. Metta, and G. Sandini, "A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 2, pp. 151–180, April 2007.

[3] R. A. Brooks, "Intelligence without reason." Morgan Kaufmann, 1991, pp. 569–595.

[4] A. Dankers, N. Barnes, and A. Zelinsky, "A reactive vision system: Active-dynamic saliency," *Proc. of the 5th International Conference on Computer Vision Systems, ICVS, Bielefeld, Germany*, March 2007.

[5] K. Muhlmann, D. Maier, J. Hesser, and R. Manner, "Calculating dense disparity maps from color stereo images, an efficient implementation," *IJCV*, vol. 47, no. 1-3, pp. 79–88, April 2002.

[6] A. Bernardino and J. Santos-Victor, "A binocular stereo algorithm for log-polar foveated systems," *Biological Motivated Computer Vision, Tuebingen, Germany*, November 2002.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, November 2004.

[8] H. Takeda, A. Ueno, M. Saji, T. Nakano, and K. Miyamato, "A robot recognizing everyday objects-towards robot as autonomous knowledge media," *Intelligent Robots and Systems (IROS)*, vol. 2, pp. 1107–1112, 2000.

[9] L. Itti and C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention," *Vision Research*, vol. 40, no. 10-12, pp. 1489–1506, May 2000.

[10] S. Grossberg, K. Roberts, M. Aguilar, and D. Bullock, "A neural model of multimodal adaptive saccadic eye movement control by superior colliculus," *The Journal of Neuroscience*, vol. 17, no. 24, December 1997.

[11] B. Schiele and J. L. Crowley, “Object recognition using multidimensional receptive field histograms,” in *European Conference on Computer Vision*, 1996, pp. I:610–619.

[12] H. Bay, T. Tuytelaars, and L. V. Gool, “Surf: Speeded up robust features,” *Computer Vision (ECCV)*, vol. 3951, pp. 404–417, 2006.

[13] J. Bauer, N. Sunderhauf, and P. Protzel, “Comparing several implementations of two recently published feature detectors,” in *International Conference on Intelligent and Autonomous Systems (IAV)*, Toulouse, France, 2007.

[14] C. Wu, “SiftGPU: A GPU implementation of david lowe’s scale invariant feature transform (SIFT).”

[15] R. Hartley and A. Zisserman, *Multiple View Geometry in computer vision*, 2nd ed. Cambridge University Press, 2003.

[16] K. Prazdny, “Detection of binocular disparities,” *Biological Cybernetics*, vol. 52, no. 2, pp. 93–99, June 1985.

[17] B. Damas, “Projecto de visao computacional,” Tech. Rep., 2006.

[18] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

[19] A. Lugos, “Color sensor system for the recognition of objects with colored surfaces,” U.S. Patent 4 917 500, April, 1990. [Online]. Available: http://www.freepatentsonline.com/4917500.html

[20] P. M. Q. Aguiar, J. M. F. Xavier, and M. Stosic, “Globally optimal solution to exploit rigidity when recovering structure from motion under occlusion,” *ICIP*, October 2008.

Appendix A

Application Manual

A.1 Software Architecture

Our work has two image input ports, the two stereo images, port “left” for the left input image, port “right” for the right input image.

Two image output ports, port “left” for the left input image overlaid with colored borders where objects were recognized, port “right” for an image of the last stored object to the database.

One bottle output port, with the information regarding the positions in the image of the recognized objects. An example of a bottle follows:

- `(size 2) (encoders 23.0 -19.5 12.4 -12.4 0.0 0.3) (time 3129831)`
  `(0 (label object1) (x 145) (y 234) (azimuth 13.4) (elevation -10.7))`
  `(1 (label blue-ball)(x 45) (y 24) (azimuth 3.4) (elevation -19.7))`

This bottle has the following configuration:

(size “int”) `number of recognized objects in present image;`

(encoders “double” “double” “double” “double” “double” “double”) `the encoder values of the robot’s neck and eyes at the moment of the image frame;`

(time “double”) `the result of cvGetTime();`

(”n” “bottle”) `numbered bottle containing the coordinates and name of an object found;`

(label “string”) `label of the object;`

(x “int”) `X coordinate of the object in the image;`

(y "int") `Y coordinate of the object in the image;`

(azimuth "double") `Azimuth angle in the ego-sphere, correspondent to the point (X,Y) in the camera frame.`

(elevation "double") `Elevation angle in the ego-sphere, correspondent to the point (X,Y) in the camera frame.`

...
and so on for all the objects recognized in the image.

A.1.1 Application

To integrate this work with the motor control and "attention selector" of iCub, a new input port was added to the ego-sphere [1]. This port receives the bottle generated by our work, which then leads to adding to the ego-sphere 'blobs' where the recognized objects are located.

All the start up scripts for the modules and the GUI are disclosed in the iCub repository.

To set up this application, first start, in the iCub computer at *$ICUB_ROOT / app / attention_objects* the camera drivers and the motor controls:

- *startCamLeft.bat*;
- *startCamRight.bat*;
- *startControlGaze.bat*.

After compiling iCub, at *$ICUB_ROOT/app/atention_objects*, start these modules preferably in different machines:

- the camCalib" for the right camera, *startCamCalibRight.sh*;
- the camCalib" for the left camera, *startCamCalibLeft.sh*;
- the ego-shere" and attentionSelection" module, *startEgoSphere.sh* and *startAttentionSelection.sh* in the same machine.

Then, in the *$ICUB_ROOT/src/SiftObjectRepresentation* folder, after compiling, run the "SiftObjectRepresentation" module, with this command line:

`YARP_Disparity_SiftCV --name chica/sift --file $ICUB_ROOT / app / attention_objects / conf / icubEyes.ini | rdin --name chica/sift`

The ports can be either connected manually or using a GUI. In the manual case:

- Connect the output ports of the camCalib modules to the input ports of the SiftObjectRepresentation module;
- Connect the output bottle port of the SiftObjectRepresentation module to the input bottle port of the ego-sphere.

In the GUI case, in *$ICUB_ROOT/app/attention_objects* start it with *startGui.bat* and connect the ports with mouse clicks.

A.1.2 SIFT

All the parameters that can be adjusted in the SIFT algorithm described in [7] are defined in *siftRH.h*.

Table A.1: Disparity Matching results.

image	sifts	matches	outliers
bola_azul_a	412	43	1
bola_azul_b	404	102	3
bola_azul_c	482	152	2
bola_azul_d	577	224	3
bola_azul_e	600	265	2
bola_azul_f	606	274	2
bola_azul_g	634	300	3
bola_a	332	61	3
bola_b	494	99	2
bola_c	545	120	3
bola_d	615	161	2
bola_e	575	178	3
bola_f	626	191	8
bola_g	623	243	3
bola_h	640	244	6
dario_a	362	92	3
dario_b	409	138	3
dario_c	463	161	5
dario_d	513	208	1
dario_e	548	208	2
dario_f	558	251	2
icubTIME_a	362	155	7
icubTIME_b	367	158	2
icubTIME_c	430	187	4
icubTIME_d	461	216	8
total	27564	9545	165
average %	-	34,1	0,6

Table A.2: Disparity Matching results.

image	sifts	matches	outliers
icubTIME_e	477	200	3
icubTIME_f	477	206	5
icubTIME_g	471	184	4
luso_a	444	23	1
luso_b	546	71	4
luso_c	565	119	3
luso_d	581	198	1
luso_e	681	219	2
luso_f	716	265	6
luso_g	659	269	4
luso_h	636	232	8
luso_i	621	262	3
retinas_a	695	179	10
retinas_b	806	250	2
retinas_c	752	267	4
retinas_d	747	272	1
retinas_e	749	253	3
retinas_f	778	264	4
retinas_g	746	312	2
retinas_h	724	272	3
retinas_i	712	268	2
retinas_j	657	250	3
retinas_k	686	279	4
total	27564	9545	165
average %	-	34,1	0,6