UAlg FCT
UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

University of the Algarve

Faculty for Sciences and Technology

ACTIVE VISION IN ROBOT COGNITION

MÁRIO ALEXANDRE NOBRE SALEIRO

THESIS
PhD in Informatics Engineering

Work developed under the orientation of:
PROF. DOUTOR JOHANNES MARTINUS HUBERTINA DU BUF
PROF. DOUTOR JOÃO MIGUEL FERNANDES RODRIGUES

2016

UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

University of the Algarve

Faculty for Sciences and Technology

ACTIVE VISION IN ROBOT COGNITION

MÁRIO ALEXANDRE NOBRE SALEIRO

THESIS
PhD in Informatics Engineering

2016

DECLARATION

I hereby declare to be the author of this work, which is original and unpublished. Authors and works consulted are properly cited in the text and appear in the included reference list.

Declaro ser o autor deste trabalho, que é original e inédito. Os autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

COPYRIGHT

Faro, 2016

Mário Saleiro

I would like to thank my parents for always supporting me and for giving me the proper conditions and motivation to get to this point, for teaching me how to define my objectives, for teaching me how to work hard to accomplish them, and for giving all the education that made me the person I am today. I would also like to thank my brother for teaching me how to program when I was still in elementary school, which probably contributed a lot to my interest in electronics and computer science. A big thank you to my girlfriend, Bruna for being so supportive and so patient during these 4 years. A very special thank you to all of you.

I would also like to thank my supervisors, Prof. Dr. João Rodrigues who has been supervising and teaching me since I started my MSc. and to Prof. Dr. Hans du Buf, who provided me with all the lab conditions I needed to develop my work and helped turning my publications much more interesting to read. Also, a big thank you to my lab colleagues Miguel Farrajota, Jaime Martins, David Lobato and especially to Dr. Kasim Terzić, who was like a third supervisor to me and from whom I learned a lot.

Also, a thank you to all my previous professors, to all the people at eLab Hackerspace with whom I had great moments and learned a lot and to all my friends who have been supporting me all the time.

THANK YOU!

Acknowledgements

The research published in this thesis was supported by the Portuguese Foundation for Science and Technology (FCT) through ISR/LARSyS pluriannual funding (UID/EEA/50009/2013), FCT PhD grant to Mário Saleiro (SFRH/BD/71831/2010) and EU project NeuralDynamics (FP7-ICT-2009-6. PN:270247).

I would also like to thank immensely for their contributions:

- Dr. Kasim Terzić, for writing part of the introduction of Chapter 3 and developing the code for visual saliency mentioned in Chapter 4 (Section 4.5).

- Miguel Farrajota for the work done in Chapter 2 (optical flow, face and hand detection, in Section 2.2.4).

- Sai Krishna for the work done in Chapter 2 (visual saliency, in Section 2.2.3).

- Prof. Dr. João M. F. Rodrigues and Prof. Dr. Hans du Buf, for reviewing my publications and this thesis, making them much more interesting to read.

Abstract

As technology and our understanding of the human brain evolve, the idea of creating robots that behave and learn like humans seems to get more and more attention. However, although that knowledge and computational power are constantly growing we still have much to learn to be able to create such machines. Nonetheless, that does not mean we cannot try to validate our knowledge by creating biologically inspired models to mimic some of our brain processes and use them for robotics applications.

In this thesis several biologically inspired models for vision are presented: a keypoint descriptor based on cortical cell responses that allows to create binary codes which can be used to represent specific image regions; and a stereo vision model based on cortical cell responses and visual saliency based on color, disparity and motion. Active vision is achieved by combining these vision modules with an attractor dynamics approach for head pan control.

Although biologically inspired models are usually very heavy in terms of processing power, these models were designed to be lightweight so that they can be tested for real-time robot navigation, object recognition and vision steering. The developed vision modules were tested on a child-sized robot, which uses only visual information to navigate, to detect obstacles and to recognize objects in real time. The biologically inspired visual system is integrated with a cognitive architecture, which combines vision with short- and long-term memory for simultaneous localization and mapping (SLAM). Motor control for navigation is also done using attractor dynamics.

KEYWORDS: Keypoint Descriptors; Stereo Vision; Active Vision; Cognitive Robotics; Object Recognition; SLAM.

Resumo

À medida que a tecnologia e a nossa compreensão do cérebro humano aumenta, a ideia de criar robôs que se comportam e aprendem como os seres humanos começa a ganhar mais e mais atenção. Contudo, apesar de esse conhecimento e do poder computacional estarem constantemente a aumentar, ainda temos muito para aprender até sermos capazes de criar tais máquinas. No entanto, isso não nos impede de tentar validar o nosso conhecimento ao criar modelos biologicamente inspirados que imitem alguns dos processos do nosso cérebro e usá-los para aplicações robóticas.

Nesta tese são apresentados diversos modelos biologicamente inspirados para visão: um descritor de *keypoints* baseado em respostas das células da área V1 do córtex visual que permite a criação de códigos binários que podem ser usados para representar regiões específicas das imagens; um modelo de visão estéreo também baseado em respostas de células da zona do córtex visual V1 e um modelo de saliência visual baseado em cor, disparidade e movimento. Combinando estes modelos de visão com um modelo de atractores dinâmicos para controlo da rotação da cabeça criou-se um sistema de visão activa.

Apesar de os modelos computacionais biologicamente inspirados serem geralmente bastante pesados em termos de processamento, os modelos apresentados foram concebidos para que fossem suficientemente leves em termos de poder de processamento de maneira a poderem ser usados em robôs para navegação, reconhecimento de objectos e para foco-de-atenção em tempo real. Os modelos de visão desenvolvidos foram testados num robô com o tamanho de uma criança, que usa apenas informação visual para navegar, detectar obstáculos e reconhecer objectos em tempo real. O sistema de visão biologicamente inspirado é ainda integrado com uma arquitectura cognitiva, que combina os processos visuais com estruturas de memória a curto e longo prazo para SLAM. O controlo dos motores para navegação também é feito utilizando atractored dinâmicos.

PALAVRAS-CHAVE: Descritores de Keypoints; Visão Stereo; Visão Activa; Robótica Cognitiva; Reconhecimento de Objectos; SLAM.

Publications

Some of the thesis contents and figures have appeared previously in the following publications (or are being prepared for submission):

- PUBLISHED IN CONFERENCE PAPERS
  - Saleiro, M., Rodrigues, J. and du Buf, J.M.H. (2009) Automatic hand or head gesture interface for individuals with motor impairments, senior citizens and young children. In Proc. Int. Conf. on Software Development for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2009), Lisbon, Portugal, June 3-5, pp. 165–171.
  - Saleiro, M., Rodrigues, J.M.F. and du Buf, J.M.H. (2010) Cognitive robotics: a new approach to simultaneous localisation and mapping, In Proc. 16th Portuguese Conf. on Pattern Recogn. (RECPAD 2010), Vila Real, Portugal, October 29, 2 pp.
  - du Buf, J.M.H., Barroso, J., Rodrigues, J.M.F., Paredes, H., Farrajota, M., Fernandes, H., José, J., Teixiera, V., Saleiro, M. (2010) The SmartVision navigation prototype for the blind, In Proc. Int. Conf. on Software Development for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2010), Oxford, United Kingdom, 25-26 November, pp. 167–174.
  - Saleiro, M., Rodrigues, J.M.F. and du Buf, J.M.H. (2012) Minimalistic Vision-based Cognitive SLAM, In Proc. 4th Int. Conf. on Agents and Artificial Intelligence, Special Session Intelligent Robotics (ICAART-SSIR2012), Vilamoura, Portugal, 6-8 Fev., Vol. 1, pp. 614–623
  - Farrajota, M., Saleiro, S., Terzić, K., Rodrigues, J.M.H, du Buf, J.M.H (2012) Multi-scale cortical keypoints for realtime hand tracking and gesture recognition, In Proc 1st Int. Workshop on Cognitive Assistive Systems: Closing the Action-Perception Loop (ISBN 978-972-8822-26-2), in conjunction with IEEE/RSJ Int. Conf. on

Intelligent Robots and Systems (IROS2012), Vilamoura, Portugal, 7-12 Oct., pp. 9–15

- Terzić, K., Lobato, D., Saleiro, S., Martins, J., Farrajota, F., Rodrigues, J. M.F., du Buf, J. M. H. (2013). Biological Models for Active Vision: Towards a Unified Architecture. In M. Chen, B. Leibe, & B. Neumann (Eds.), Computer Vision Systems SE - 12, vol. 7963, pp. 113–122. Springer Berlin Heidelberg.
- Saleiro, M., Farrajota, M., Terzić, K., Rodrigues, João M.F., du Buf, J.M.H (2013) A biological and realtime framework for hand gestures and head poses, In C. Stephanidis and M. Antona (Eds.) Universal Access in Human-Computer Interaction. Design Methods, Tools, and Interaction Techniques for eInclusion SE – 60, vol. 8009, pp. 556-565
- Terzić, K. and Lobato, D. and Saleiro, M. and du Buf, J.M.H. (2014) A fast neural-dynamical approach to scale-invariant object detection, Proc. Int. Conf. On Neural Information Processing, LNCS vol. 8834, pp. 5611-518.
- Saleiro, M., Terzić, K., Lobato, D., Rodrigues, J.M.F., du Buf, J.M.H. (2014) Biologically inspired vision for indoor robot navigation, In Campilho, Aurélio, Kamel, Mohamed (Eds.): Image Analysis and Recognition, Part II, vol. 8815, pp 469–477, Springer International Publishing
- Cardoso, P.J.S., Rodrigues, J.M.F., Carlos, L., Mazayev, A., Ey, E., Corrêa, T., Saleiro, M. (2015) A freehand system for the management of orders picking and loading of vehicles, In M. Antona and C. Stephanidis (Eds.): Universal Access in Human-Computer Interaction 2015, Part IV, LNCS 9178, pp. 422–431
- Saleiro, M., Farrajota, M., Terzić, K., Krishna, S., Rodrigues, J.M.F., du Buf, J.M.H. (2015) Biologically inspired vision for human-robot interaction, In M. Antona and C. Stephanidis (Eds.): Universal Access in Human-Computer Interaction 2015, Part II, LNCS 9176, pp.

505–517.

- PUBLISHED IN JOURNALS:
  - du Buf, J.M.H., Barroso, J., Rodrigues, J.M.F., Paredes, H., Farrajota, M., Fernandes, H., José, J., Teixiera, V., Saleiro, M. (2011) The SmartVision navigation prototype for blind users, JDCTA: International Journal of Digital Content Technology and its Applications, Vol. 5, No. 5, pp. 351–361
- SUBMITTED OR IN FINAL PREPARATION FOR SUBMISSION:
  - Saleiro, M., Terzić, K., Rodrigues, J.M.F., du Buf, J.M.H. (2016) BINK: Biological Binary Keypoint Descriptor. Submitted to Biosystems (Ref: `BIO_2016_D134`)
  - Saleiro, M., Terzić, K., Krishna, S., Rodrigues, J.M.F., du Buf, J.M.H., Biologically Inspired Active Vision for Robot Head. In final preparation for Biologically Inspired Cognitive Architectures, Elsevier.
  - Saleiro, M. Terzić, Rodrigues, J.M.F., du Buf, J.M.H., A cognitive robot with biological vision for SLAM. In final preparation for Cognitive Systems Research, Elsevier.

Contents

Acknowledgments **I**

Abstract **III**

Resumo **V**

Publications **IX**

List of figures **XV**

1 Introduction **1**

1.1 Scope of the thesis 1

1.2 Human visual system 4

1.2.1 Visual attention models 5

1.2.2 Object recognition 7

1.2.3 Object recognition by humans 8

1.2.4 Object recognition methods 10

1.3 Cognitive robotics 12

1.3.1 Simultaneous localization and mapping 13

1.3.2 Cognitive memory 16

1.3.3 Types of memory 18

1.3.4 Cognitive memory in robotics 18

1.3.5 Task realization and management 20

1.4 Overview of the thesis 21

2 Biologically inspired vision for human-robot interaction **25**

2.1 Introduction 25

2.2 Biologically inspired HRI system 27
2.2.1 Keypoint descriptors for gesture recognition 27
2.2.2 Stereo vision for robot navigation and obstacle detection 30
2.2.3 Visual saliency 31
2.2.4 Face and hand detection 33
2.2.5 Gesture recognition 36
2.3 Tests and results 37
2.4 Discussion 39

3 Learning biological binary keypoint descriptors 41

3.1 Introduction 42
3.2 Related work 45
3.3 Method 47
3.3.1 Training and evaluation 48
3.3.2 V1 cortical cells 49
3.3.3 Learning the projection matrix 50
3.3.4 Learning the optimal binarization thresholds 51
3.4 Evaluation 51
3.4.1 Cell selection: even simple, odd simple or complex? . . . 52
3.4.2 Scale comparison 55
3.4.3 Pooling region size 56
3.4.4 Pooling step selection 57
3.4.5 Dimensionality of the descriptor 58
3.4.6 Combining different scales 59
3.4.7 Testing HMAX features 60
3.4.8 Comparison with other biologically inspired local descriptors 61
3.4.9 Comparison with the SIFT-based LDAHash local descriptor 63
3.4.10 Computation time 64
3.5 Applications 64
3.5.1 Optical flow 65

3.5.2 Stereo vision . . . 67

3.5.3 Object recognition . . . 68

3.6 Conclusions . . . 69

4 Biologically Inspired Active Vision for Robot Head 71

4.1 Introduction . . . 72

4.2 Related work . . . 74

4.3 Biologically inspired architecture of the active vision system . . 79

4.4 Multiscale binocular stereo cells . . . 81

4.4.1 LAB image resizing . . . 83

4.4.2 Binary masks . . . 84

4.4.3 Binary pixel descriptors . . . 84

4.4.4 Pixel correspondence . . . 86

4.4.5 DoG filtering . . . 86

4.4.6 Disparity map . . . 87

4.5 Saliency . . . 89

4.6 Object Recognition . . . 91

4.7 Attractor dynamics for robot head pan control . . . 92

4.8 Evaluation and tests . . . 97

4.8.1 The robotic platform . . . 97

4.8.2 Middlebury dataset for stereo evaluation . . . 98

4.8.3 Test results of the active vision system . . . 100

4.9 Discussion . . . 104

5 A cognitive robot with biological vision for SLAM 107

5.1 Introduction . . . 108

5.2 Cognitive robot framework . . . 109

5.2.1 Robotic platform . . . 111

5.2.2 Active vision system . . . 113

5.3 Memory and mapping . . . 116

5.3.1 Short- and long-term memory . . . 117

5.3.2 Task management . . . 120

5.4 Attractor dynamics for differential motor control . . . 122

5.5 Conclusions 123

6 Concluding Remarks 127

6.1 Summary 127

6.2 Integrated architecture 129

6.3 Achievements 129

6.4 Final considerations for future research 130

Bibliography 132

List of Figures

1.1 Example of a cognitive model. 17

2.1 Example of descriptor matching and descriptor comparison. . . 29

2.2 Disparity and Saliency maps. 32

2.3 Person coded by HOG-like features and with detected face and hands. 34

2.4 HOG-like feature parameter performance. 37

2.5 Robotic platform, gesture recognition and hand and head detection. 38

3.1 Sample patches from the Notre Dame and Yosemite datasets. . . 47

3.2 Comparison of results using different types of cells. 54

3.3 Comparison of the best results for each of the scales tested. . . . 55

3.4 Evaluation of performance for pooling regions with different sizes. 56

3.5 Evaluation of performance of 128-bit descriptors generated from even simple cell responses 57

3.6 Comparison of results using 4 different numbers of bits at scales $\lambda = 4$ and $\lambda = 6$ 58

3.7 Comparison of results between the best 128-bit single-scale results 59

3.8 Testing HMAX features. 61

3.9 Comparison of our descriptor performances with the other biologically inspired descriptors BRIEF and BRISK 62

3.10 Comparison of our descriptors with the non-biological descriptor LDAHash 63

3.11 Example of optical flow determination using cortical keypoints and keypoint descriptors. 66

3.12 Example of stereo matching using our descriptor 67

3.13 Example of object recognition using our keypoint descriptors . . 69

4.1 System architecture 80

4.2 Vision component overview 81

4.3 Descriptor sampling pattern 85

4.4 Example showing the use of the DoG to model the inhibition and excitation interactions between cells tuned to different disparities 87

4.5 Examples of saliency maps 90

4.6 Example of object recognition 92

4.7 Example of vision steering using attractor dynamics 96

4.8 Child-sized robot used for the active vision tests. 97

4.9 Middlebury Stereo evaluation 99

4.10 Playtable image from Middlebury training dataset (version 3) and Djembe image from Middlebury the testing dataset. 100

4.11 Sequence of images showing the movement of the head to focus the salient regions on the table top setup 101

4.12 Sequence of images showing the movement of the head to focus the salient regions on the corridor setup 102

4.13 More examples of the robot recognizing objects. 103

4.14 Sequence example of a tracked object. 104

5.1 Child-sized robot used for the active vision tests and robot monitoring panel 112

5.2 Example of images captured during navigation: disparity map, saliency map and object recognition 116

5.3 STM and LTM maps 118

5.4 LTM map acquired during navigation 119

5.5 Task management system. 121

5.6 Example of attractor dynamics during robot navigation 124

Chapter 1

Introduction

Abstract: This chapter introduces the scope of this thesis, namely, an integrated cortical architecture for robot vision and cognitive robotics, relying on biological vision principles and properties.

1.1 Scope of the thesis

Throughout history, humans have created tools, and later machines, to improve and accelerate daily tasks. In the same way that the human being has evolved, its creations have also evolved to the point where we find ourselves today, where most of the harder, longer and repetitive tasks could be performed by robots. However, the robots that have been developed in the past decades were dedicated to industrial environments, where they realize specific tasks under human supervision. The term “autonomous robot” is used many times but the truth is that truly autonomous robots still hardly exist. They are autonomous only in the sense that they are able to execute a previously programmed task on their own. They are not quite able to adapt to changes in the workspace. In some cases, very small changes in the workspace may be enough to make a robot fail in its task.

There has been much research and advances in artificial intelligence to increase the autonomous component of robots so that they can learn by themselves while interacting with the surrounding environment. A big part of that research can provide good solutions for specific cases but there may still fail when the changes in the workspace increase. However, we know of a system in nature that is capable of learning and solving a large number of highly complex problems: the human brain. This system is capable of processing the various stimuli (sounds, images, smells, etc.) of the surrounding environment in an extremely fast and efficient way. On the other hand, we must take into account that the creation of a computational model that is able to simulate the human brain in all its components and with all its capabilities is extremely complex. Although some are optimistic [Kurzweil, 2006], this will certainly take many more years since we do not even completely know how our brain works.

Considering the human brain as a model when we think of autonomous robots, it is important to focus on one of its features: cognition. The word "cognition" generally refers to the faculty of mental activities of human beings dealing with abstraction of information from the real world, their representation and storage in memory, as well as automatic recall [Patnaik, 2007]. It is believed that through cognition we may overcome the limitations of classical artificial intelligence, hence allowing us to create cognitive robots: robotic systems that, instead of being pre-programmed to execute a task, are able to execute it by learning from its past experiences and emotions [Ratanaswasd et al., 2005].

Cognition implies an apprenticeship and any process of learning begins with the acquisition of information. In the case of human beings, that acquisition of information begins in the senses. Concerning the five human senses, we may consider vision as the most important one in the cognitive process. Using only the visual system we are able to immediately identify objects, places, other people and other animals. Vision also allows us to have notions of distance and size. Another factor that leads us to consider vision as one of the most important senses is the fact that it is closely connected to memory: it is much easier to recognize someone by the face than by the voice; it is also easier to recognize an object by looking at it than by what we feel when we touch it. Doing this analysis in a reverse direction, when we think about someone we know, we easily remember the face. However, although we can easily remember some things and create mental representations of objects, places and people, we cannot remember everything to the very finest detail. Some things remain in our memory for our entire life whereas other things may not even remain there for a few seconds. This duality in the storage of information in our brain has been studied by researchers and cognitive psychology scientists for a very long time. They usually consider the existence of two main types of memory: short-term memory and long-term memory [Patnaik, 2007]. Some even consider the existence of a third type of memory: sensory memory, which consists of the immediate perception of the surrounding environment [Brady et al., 2008].

A very simple example of the distinction between the several types of memory is the observation of an image. In the immediate moment in which we observe it, we are able to describe everything to the finest detail, since all the captured information is still present in sensory memory. After a few seconds we are able to describe less details, since only the most important details were moved to short-term memory. Finally, if we are going to describe the same image after a few days, we are only able to give a short description of the scene without many details when compared to the descriptions based on the short-term and sensory memories. In this triadic memory system there is a process of filtering the information in such a way that only important things remain in memory. However, we may not say that visual long-term memory cannot have a high degree of detail: apart from being able to remember specific scenes, we are also able to remember details that allow us to distinguish objects of the same type and their states. The detail of the stored information depends on our attention and concentration while observing the world around us. Although we normally suppress most of the details, we are able to store lots of detailed information when we have to [Brady et al., 2008].

In the same way that we are naturally capable of selecting and storing only the relevant information that comes from perceptions, we are also able to direct our attention to regions that are relevant, discarding the rest. This ability to filter out important information is something that reflects our brain's efficiency. If our brain would analyze all the information provided by our senses it would be constantly busy [Rensink, 2000]. As an example, when we observe a scene we easily identify several objects without analyzing everything in our field of view. The same does not happen when we analyze images using classical computer vision algorithms. In these, the images are normally processed pixel by pixel, which makes the processing extremely heavy and slow when compared to human capabilities.

Another problem that emerges when we try to model human cognition is object recognition. This topic has been widely studied but, as for most of the problems already described, it does not have a solution that allows for generalization. Most used recognition methods can only deal with objects which already exist in training databases [Forssén et al., 2008].

The implementation of cognitive systems in robots, even if only very rudimentary, may yield a new generation of robots that may, in a certain way, adapt to changes in the workspace and interact with humans. By implementing a cognitive system, human-machine interaction may become much more dynamic and would allow the system to evolve.

There are countless applications for cognitive robots, such as assistance robots for elderly or handicapped people. From tasks that may seem simple, like navigation in dynamic environments, other applications may emerge like guiding robots for blind people in airports or shopping malls, surveillance robots, etc. Despite the countless applications, the pursuit of cognition in robots is still in its first steps. However, the increasing exchange of information between engineers, psychologists and neuroscientists, among others, certainly will result in the creation of biological models that will get closer and closer to the cognitive system of which we know that it works but still not how it works: our own brain.

1.2 Human visual system

For a better comprehension of the text, in this section we briefly introduce some main aspects of the Human Visual System (HVS), namely, the visual attention models and object recognition. The state-of-the-art about each chapter topic in done within the chapter.

1.2.1 Visual attention models

Most of the scenes that we observe are too complex to be totally perceived. The human brain, although being extremely complex and having great capabilities, also has its limitations and it is not able to instantly process all the sensory information. Hence, we may say that its great capabilities are also due to the ability of selecting and using only the necessary and essential sensory information. In this process, humans must select the essence of the scenes they observe – gist – and process this sequentially [Martins et al., 2009].

This selection process is not only performed by humans. It is also a very important process for the survival of animals, allowing them to quickly detect their preys and their predators. However, although this process is just a small part of the visual system, when combined with other processes it allows us to draw some conclusions on how it works [Martins et al., 2009]: (1) very quick extraction of the essence of the perceived scene, (2) also very fast extraction of the essence of some objects and drafting a spatial layout map, in parallel with (3) the construction of a saliency map for Focus-of-Attention (FoA), and finally (4) sequential analysis of different regions of the observed scene for precise object recognition, using peaks and regions of the saliency map with Inhibition-of-Return (IoR), in order not to process the same region more than once.

The last step (4) may occur in a conscious (directed or overt) or unconscious (covert) way [Martins et al., 2009]. Focus-of-Attention [Itti and Koch, 2001] is a cognitive process that may be described as the ability to focus one or more senses on a specific object or sound. FoA is directly related to the saccadic movements of the eyes (saccades), which occur countless times per minute without us being aware of this. However, the existence of information about the place where something is going to happen may influence our movements [Masciocchi et al., 2009]. When we look at an image, our eyes will sequentially focus the most salient regions by descending order of conspicuity, because some regions will have a bigger concentration of salient points than others.

The regions with higher concentration of points usually correlate with the detected salient regions. The grouping of these salient and spatially organized regions we call a saliency map. Saliency maps represent the instantaneous saliency of the different parts of the visual input. Such maps can be built from several features, such as color, orientation, intensity, among others [Itti and Koch, 2001].

The identification of salient regions in the periphery of the field of view may also cause movements of the head, for directing the field of view towards regions of higher interest. The fact that visual attention guides our field of view towards areas with higher saliency makes it one of the most important mechanisms for perceiving the surrounding environment [Ruesch et al., 2008].

As already mentioned, saliency maps may be built from several features, such as keypoints, contours, color, texture, orientation, movement, disparity, etc. However, each of these features results in a different saliency map, making it necessary to combine multiple saliency maps into one "master" map. The result can be the intersection of the multiple maps or their union [Martins et al., 2009]. However, in some cases it is possible to obtain good results using only a small amount of features. An example of this is the saliency and segregation algorithm proposed by Martins et al. [2008] which is based on color, one of the most relevant features.

One of the reasons of the above algorithm is the fact that the color regions are normally very well defined because of differently colored objects and backgrounds. This makes it easy to delimit them and segregate them from the rest of the image. Later, Martins et al. [2009] presented a biologically plausible method to obtain a saliency map not only based on color but also on texture. The combination of color saliency with texture saliency makes the method more complete, since it can highlight other types of features in images, therefore making it more robust to multiple environments.

A very different attention algorithm for a robot was developed by Meger et al. [2008]. Their algorithm uses stereo vision to determine the depth of the field in order to highlight objects that are not in the ground plane. This method implies the use of a stereo camera to generate the disparity maps. Apart from the saliency map obtained from disparity, a color-based saliency map is also created and merged with the previous one. The combined saliency map is obtained by the intersection of the two maps.

Another algorithm to create saliency maps was proposed by Butko et al. [2008]. Their algorithm applies an approximation to a Bayesian model, optimized to be used in real time with low computational cost, to direct the cameras of a robot for the development of social robots.

In general, visual attention algorithms are very similar with some common aspects, even if they are based on different features or combinations of multiple features. Color is one of the most used features due to its relevance and to the good results that can be obtained. However, other aspects such as texture, orientation or disparity may also provide very interesting and useful results. Once a saliency map is computed we can then apply object recognition algorithms to the most salient regions.

1.2.2 Object recognition

Object recognition is one of the major topics in computer vision since it is connected to all the branches of computer vision. Throughout times significant research has been done to develop representation methods and algorithms to recognize images captured under multiple conditions (different points of view, different illumination conditions, occlusions, etc.). In some cases of very specific objects, such as fingerprints, faces and traffic signs, substantial success rates have been accomplished. One of the objectives of object recognition is also to perform categorization.

In most applications involving artificial vision it is necessary to make a distinction between the detection and the recognition of one or several objects. As an example, in an industrial environment the objects to detect are previously well defined and sometimes the position where they will appear is also known. This makes the recognition task much easier and straight-forward [Ohali, 2011]. If the objects can appear in any position, the task becomes a lot more difficult. In the case that many different objects may appear and that they may appear with different views and in different backgrounds, it becomes extremely hard to perform object recognition using the methods from classical computer vision. An aspect that makes it even harder is if the objects are partially occluded.

Many object recognition methods are based on the use of big image data sets, containing multiple views of each object to be detected [Meger et al., 2008]. However, the efficiency of the algorithms decreases and the processing time increases with the increase of the number of objects in the data sets, since the number of comparisons to be made between the objects from the data set and the captured image grows. Another problem emerges from the small variations that objects may have, even if they belong to the same type of objects, i.e. class or category. Taking all of this into account, it is not hard to conclude that we are still far away from having an object recognition system which comes close to our visual system.

We, human beings, are able to detect and recognize an infinity of objects almost instantly, regardless of the variations caused by illumination, position or occlusion [Al-Absi and Abdullah, 2009]. By peeking into a room in a house we instantly realize whether it is a living room, a bedroom or an office, since we immediately recognize the kinds of objects in that room. This allows us to draw conclusions about the type of room that we are looking at [Vasudevan et al., 2007] or even predict which other objects we may expect in that room, even if we have not yet seen them.

In the next section we will focus on the human visual system component that relates to object recognition. After that some object recognition methods will be described.

1.2.3 Object recognition by humans

Object recognition is an ability that we possess since early childhood. With a simple observation of an object, we are able to identify and categorize it irrespective of the countless changes that may occur due to illumination, position, orientation or occlusion. It is therefore a great challenge to develop vision systems that may get close to the cognitive capabilities of human beings. The main difficulties in the development of such methods lie in the variations mentioned above and in the challenge of creating a generalization of an object from a group of sample images.

We still do not completely know how our visual system works. However, it has been the object of many studies from which some theories about its inner workings have arisen. Some of the studies were made by recurring to expensive equipment [Rodrigues, 2008] which allowed researchers to measure the activity levels in certain brain regions when objects or scenes were shown to the test subject.

On the other hand, other studies were a lot simpler, consisting of the analysis of certain human behaviors in an attempt to understand which features have the biggest importance in the recognition of objects and how they are stored in memory. One of those studies was done by Vasudevan et al. [2007]. It allowed to conclude that the structure of the objects was a fundamental aspect to perform recognition. The structure of an object may be represented by points, lines and contours. There are some methods that use this kind of information to perform object recognition. These will be described in the next section.

Regarding the recognition process, for some time we used to think that our vision system applies a sequence of processes (detection, segregation, categorization and recognition). According to recent research [Bar et al., 2006] these processes cannot be completely sequential. They must occur in "parallel," at least partially. It was a common habit to think that to recognize an object we first had to isolate it from the background. However, recent research suggests that the categorization of objects occurs before their segregation. In other words, when we realize what the object that we observe is, the brain already knows which object it is [Rodrigues and du Buf, 2009a].

Apart from all these breakthroughs and research, we still do not know the exact order in which processes occur, as well as in which cases there is parallel processing. We are only certain that object recognition cannot be seen as a simple task. This process must be seen as a task with several levels [Rodrigues and du Buf, 2009a], and it is possible that at each level multiple processes occur simultaneously.

The breakthroughs that have been achieved have inspired the creation of several biological models which try to mimic human vision. However, these models demand a large processing power, making it mandatory to use high performance multi-core processors. These processing demands make it difficult to apply this type of models in real-time applications, such as in robotics.

1.2.4 Object recognition methods

There are several methods for object recognition that can be split into two main types: dedicated object recognition methods and general object recognition methods. The dedicated methods are developed with the goal of recognizing a limited number of objects and, as such, they are optimized to detect those specific objects. These methods are usually applied in industrial environments to inspect products and to monitor processes [Ohali, 2011]. General object recognition methods may work on a vast majority of applications, despite of having a bigger computational cost [Lowe, 1999; Evans, 2009].

Object recognition methods rely on the extraction and recognition of image regularities taken under different illuminations and positions. In other words, most algorithms use certain representations and models to capture those characteristics, making it easier to execute procedures to identify the objects [Lowe, 2004]. The representations may be 2D or 3D geometric models. After the extraction of the fundamental features of the image the recognition procedure is performed. It is based on the comparison of the models or object representations with the test image [du Buf et al., 2010].

The most common features on which object recognition algorithms are based are geometry, aspect and interest points. To analyze object geometry, some perspective invariant geometric primitives (lines, circles, etc.) are extracted. Algorithms devoted to aspect are based on patterns of the objects (features, textures, histograms, etc.) [Shotton et al., 2008]. Finally, the methods based on interest points search for certain points in images that are invariant to changes provoked by illumination or scaling. One of the most used representation methods for vision applications is SIFT (Scale-Invariant Feature Transform), proposed by Lowe [1999].

Using the SIFT representation, a recognition system was developed which consists of transforming an image into a collection of vectors of scale- and rotation-invariant keypoints. These keypoints are also partially invariant to changes in illumination and 3D projections. The fact that keypoints are invariant to a wide range of changes resulted in the creation of a robust method for object recognition, even in cases of occlusion of parts of the objects or in cluttered environments and with complex backgrounds. In the keypoint matching process, every cluster of keypoints with at least three matches can be accepted as a detected object [Ramisa et al., 2008].

The enhanced SIFT method, proposed by Lowe [2004], was tested and analyzed by Ramisa et al. [2008]. It was compared with the "bag of features" method proposed by Nistér and Stewénius [2006]. From the results obtained by Ramisa et al. [2008], it was verified that for textured objects, being the texture uniform or not, the results are similar for both algorithms. However, we must note that the SIFT method was not able to detect untextured objects in the experiments.

There is another type of descriptors based on SIFT, called SURF (Speeded-Up Robust Features) [Bay et al., 2008]. The SURF algorithm was developed with the purpose of being more efficient in terms of processing cost, therefore more suitable for real-time applications. According to the authors, the SURF algorithm is faster, more robust and more precise than SIFT.

Another method that is not only suitable for object recognition, but also for categorization, is the one proposed by Rodrigues and du Buf [2009a]. This one is biologically inspired, it is a cortical model. The method is based on multi-scale features: lines, edges and keypoints are extracted by using the responses of simple, complex and end-stopped cells in cortical area V1. The keypoints are used to build saliency maps that are then used for Focus-of-Attention.

The latter algorithm allows us to obtain 2D translation, rotation and scale invariance through the dynamic mapping of saliency maps based on information provided by the multi-scale keypoints. The model is functional and is split into two parts: keypoints are used for object recognition and lines and edges for categorization. Apart from this division in two parts, there is also a progression of detail in the data flows, starting in both cases with a coarse scale (less detail) and progressively using finer scales (more detail).

Recently some new descriptors have been developed, but unlike SIFT and

SURF, which are floating-point descriptors, most of these are binary. The interest in binary descriptors was powered by the increase of mobile devices, which had limitations in terms of processor and memory, and which needed faster and lightweight descriptors for mobile applications. Binary descriptors are much faster to match than floating-point descriptors because the matching is done by calculating the Hamming distance. Some of these descriptors are simple binarizations of existing floating-point descriptors (e.g. LDAHash is based on SIFT features [Strecha et al., 2012]). With the growing interest in binary descriptors came some that have a biological inspiration, like BRISK [Leutenegger et al., 2011], FREAK [Alahi et al., 2012] or BRIEF [Calonder et al., 2012]. However, their performance lags behind non-biological binary descriptors, even though the biological ones use more bits than the non-biological descriptors.

1.3 Cognitive robotics

Cognitive robotics is an approach to robotics that is still in its initial steps. Most of the research in robotics focuses on the realization of specific tasks, not implying a need for an architecture that allows the robot to learn and evolve while performing several tasks. Most of the time, robots use several types of sensors (infra-red, ultrasonic, laser, odometric, etc.) which allow them to precisely perform tasks that they are specifically programmed to perform. However the kind of data used with those types of sensors has not much to do with the type of data that we use to perceive the world around us [Montemerlo et al., 2002].

Although our visual system allows us to make coarse estimations of distances and to know if the objects that we detect and recognize are close or far away, it does not provide us with very precise distances.

If we consider these limitations when trying to program a robot to solve the classical problem of SLAM (Simultaneous Localization and Mapping) we realize that it becomes a really difficult task to accomplish [Davison et al., 2007]. Without precise metrics it becomes difficult to build a map of the surrounding environment.

To solve this problem we have to make the robot proceed as a human and perceive the surrounding environment using vision to recognize objects in order to create references in space, so that the robot can calibrate its position. However, in an attempt to mimic the human visual system, before performing object recognition it becomes necessary to apply a visual saliency model [Itti et al., 1998] in order to select the regions of the image that may contain the most relevant objects and to recognize them fast, and ignore the non-relevant information.

Another problem to be solved is to create a system that allows a robot to adapt to changes in the surrounding space, like we do, due to the way our memory works [Deco and Rolls, 2004; Brady et al., 2008]. Although we have a great ability to store information in our brain, we only remember the most important information. In classical robotics, enormous amounts of data are stored in electronic storage devices, and traditional SLAM systems make use of all that information [Montemerlo et al., 2002]. However, when we consider the case of a robot that may be able to adapt to changes of the surrounding environment, the continuous storage of information becomes unreliable since many of the stored data may become redundant or obsolete after some time.

It is also necessary to conceive a way to make the robot's architecture scalable. When thinking about cognition, we must consider to implement several action levels, so that the robot may build complex tasks by combining simpler ones, which may be common to several other tasks [Ratanaswasd et al., 2005]. In non-cognitive robotics, robots are only provided with predefined, complex operation cycles.

In short, there are five major distinct problems that, however, are connected to each other: (a) SLAM; (b) memory organization; (c) attention models; (d) object recognition and (e) multi-level task scheduling.

1.3.1 Simultaneous localization and mapping

The ability to simultaneously localize the robot and map the surrounding near space is a fundamental pre-requisite to develop truly autonomous robots [Se et al., 2001; Montemerlo et al., 2002]. In order to navigate in unknown locations, the robot must be able to perceive them well enough to navigate safely.

Mapping in mobile robotic applications is something that has often been applied and resulted in many different models of maps [Thrun et al., 2000; Davison et al., 2007]. The models may differ in several ways, such as the number of dimensions or the quantity and type of used references. The types of mapping may be classified according to several features, but there are two main features. (a) The number of dimensions (2D or 3D) [Dellaert and Stroupe, 2007; Davison et al., 2007]. Each type has its advantages and disadvantages. The 2D maps are simpler, but do not allow for a correct localization of specific references in the map since there may be two references in the same spot but at different heights [Se et al., 2001]. On the other hand, 3D maps do not have that issue but they are much more complex. (b) The type of used references (landmarks). A map can be metric or topological [Thrun et al., 1998; Tomatis and Nourkbakhsh, 2001] or, in some cases, hybrid [Buschka and Saffioti, 1998]. In metric maps the representation of the surrounding world is based on a set of coordinates arranged in a Cartesian coordinate system [Thrun et al., 1998]. This kind of mapping is normally used when a robot has an odometry[1] system and multiple sensors that allow it to measure distances. An example of this kind of mapping is FastSLAM [Montemerlo et al., 2002], in which the map is built by an odometry system with laser sensors, registering at every point of its path the distances to detected obstacles. However, errors in the odometry system and in the sensors cause loss of precision over time. On the other hand, topological maps are built by establishing connections between reference points, without any precise metrics [Tomatis and Nourkbakhsh, 2001]. As such, these maps cannot provide an exact location, neither of the robot nor of the reference points.

Until here only the basic types of mapping were referred to. However, a large number of variations of the described methods have been implemented. One of those was proposed by Se et al. [2001] and consists of the use of stereo vision computed from a group of scale and rotation invariant keypoints. These

[1] Odometry refers to measuring speeds and distances.

points are acquired using the SIFT descriptors [Lowe, 1999]. Since these points are invariant to several transformations, they provide good references for a mobile robot. Using vertical and horizontal disparity (3-camera system) they were able to know the keypoints' location in three-dimensional space for their mapping. A very similar approach, also using a 3-camera system was proposed by Saeedi et al. [2003].

Another variant, proposed by Tomatis and Nourkbakhsh [2001], consists of a hybrid architecture which combines topological and metric maps in a SLAM system, creating a compact model which does not require highly consistent information on a global scale. The maps used in this approach consist of a group of nodes (openings) connected with each other, the list of reference points (corners) being placed between each pair of two nodes. The nodes may be transitions to another division (room) or a connection to another hallway.

Topological maps also fit in the work of Vasudevan et al. [2007], who confirmed the existence of a hierarchical sequence of structural elements when people describe how to go from one place to another. They also verified the use of reference points such as doors and walls which delimit a place or a room.

Another hybrid concept, presented by Kawamura et al. [2002] considers egocentric navigation. This concept possesses a metric component very different from most methods. In this case, at the beginning of navigation the robot already has a sketch of the map in its memory, with reference points placed at their corresponding locations. The initial sketch does not require precise metrics since the robot's navigation is done in a topological way. In addition to the sketch, the sequence of reference points that the robot must go through to reach its destination is also placed in memory.

An interesting aspect involved in this method is the sensory egosphere, which consists of the useful information in the environment close to the robot at each moment. This concept is an approach to short-term memory, which will be described in the following sections. One example of this concept can be seen in Fig. 1 from Kawamura et al. [2002]. It shows a sketch of a map without any metric precision that allows the robot to navigate by using the annotated reference points. The path used by the robot to go from point A to point B is also sketched.

There are also methods which explore a combination of the robot's "perceptions" and its motor actions [Zetzsche et al., 2008], creating a hybrid architecture which is also related to human behavior, since our movements are directly related to our vision. According to this model, when the robot finds itself exploring the surrounding environment, instead of doing a random exploration, it goes towards the paths where it may find a larger amount of information, which is what we, humans, do. To perform such exploration, the authors combine the mapping system with a model of visual attention, that resembles saccadic eye movements, in order to find regions of interest.

More recently, Ball et al. [2013] released a new SLAM system, OpenRatSLAM, successor of RatSLAM [Milford and Wyeth, 2010]. They are both based on two types of cells that exist in rat brains, the hippocampus: local view cells and pose cells, which are similar in characteristics to navigation neurons found in many mammals, called grid cells. OpenRatSLAM is composed of four nodes: (a) a pose cell network, that manages the energy packet that represents pose in response to odometric and local view connections; (b) local view cells that determine whether a scene given by the current view is novel or familiar by using image comparison techniques; (c) experience map; and (d) visual odometry, which provides an odometric estimate based on changes in the visual scene. Excellent results have been obtained with OpenRatSLAM in different experimental setups: it was successful in mapping the suburbs of St. Lucia, in Australia, using only a laptop on top of a car, and in mapping a small controlled environment using a rat-sized robot.

1.3.2 Cognitive memory

Throughout times, several models for cognition were developed. In a general way, a cognition model is composed of a group of mental states, by their transitions and also by cognitive memory which is connected to the various cycles in the model. This interconnection is a fundamental part of the cognition model. An example of a cognitive model was proposed by Patnaik [2007]. It contains a total of seven mental states plus memory: sensoring and acquisition, reasoning, attention, recognition, learning and planning, action and coordination (see Fig. 1.1).

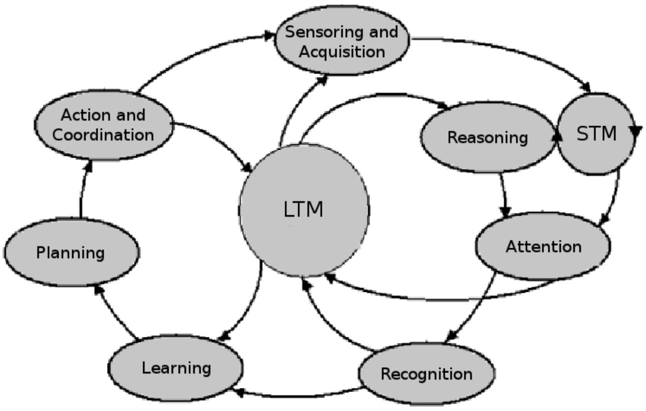


Figure 1.1: Example of a cognitive model in which it is possible to verify the existence of three cycles: acquisition, perception, and learning and coordination. The main block of the system is long-term memory (LTM), the second main block being the short-term memory (STM). Adapted from Fig. 1.1 in Patnaik [2007].

The central role of cognitive memory is evident in this model, with a stronger emphasis on long-term memory. There are three cycles in the model, the long-term memory being the connection between all three. The acquisition cycle is composed by the short- and long-term memories, by sensoring and acquisition, and by the attention states [Patnaik, 2007]. The perception cycle contains the states of reasoning, attention and recognition, and the long-term memory. Finally, the third cycle is composed by the states of learning, planning, action and coordination, and also by long-term memory.

The process of cognition starts in the acquisition cycle, in which a human being acquires information by using its senses and puts it in short-term memory. The information will then be processed and will be sent, or not, to long-term memory. In the perception cycle the brain processes previously acquired information, performing the recognition of information and creating higher level information which is also stored in long-term memory. At last, the cycle of learning and coordination uses the information stored in long-term memory in order to proceed to learning, action and coordination.

Taking into account the evident importance of memory in the cognitive process, it becomes necessary to study it more deeply in order to find out how to implement a similar structure on a mobile robot.

1.3.3 Types of memory

The human brain has an enormous capacity for storing information [Brady et al., 2008]. However, some of the acquired information is stored only for a short time while the rest may be stored for a very long time. This duality evidences the existence of more than one type of memory in the human brain. Much research has been done to identify how many types exist and what their purpose is.

We can say that there are three main types of memory in our brain [Brady et al., 2008; Smith et al., 2008]. They can be distinguished by the duration that the information may remain in them, and also by their source of information. Sensory memory is only used to store information provided by the senses, quickly fading unless that information is being focused by the attention system. Short-term memory consists of our brain's working memory and contains only the information that is being processed. Long-term memory stores all persistent information. The information stored in this last type of memory may remain there for a very long time, even if it is not regularly used [Smith et al., 2008]. The capacity of short-term memory is very small if we take into account the complexity of the tasks that we perform. Recent research suggests that the information stored in short-term memory consists only of references to information stored in long-term memory [Smith et al., 2008].

1.3.4 Cognitive memory in robotics

The use of multiple types of memory (sensory, STM and LTM) in robotics is not something common. In most applications information is accumulated over time, disregarding its importance for the task that the robot must perform. An example of this is reflected in the work of Lowe [1999], where a large number of keypoints must be stored to allow the robot to navigate in an unknown environment. However, there are a few attempts to model the inner workings of human memory in robotic applications.

An example of an implementation of a memory management system was proposed by Kawamura et al. [2002]. The authors considered the existence of two types of similar storage structures: the robot's sensory egosphere and the sensory egosphere of the landmarks that the robot uses to locate itself. The first one corresponds to STM and the second to LTM. While the robot roams around, STM will constantly be changing, in contrast to what happens to the landmarks' egospheres that are stored in LTM for later use. This storage is done spatially, in order to create a map with relative positions of landmarks. The dual memory structure is simple but it was mainly designed for the localization and mapping problem, therefore lacking some other features which are required for the cognitive process.

Another, more complex, approach to cognitive memory was developed by Ratanaswasd et al. [2005]. The authors used memory structures to keep information for immediate tasks and also to store experiences that may be used during a decision process. Again, two memory structures were used: STM and LTM.

STM consists of the storage of sensory information in a sensory egosphere. The amount of information stored in the sensory egosphere is very small, containing only simple information like keywords or colors. Another feature of STM is the fact that the stored information will fade over time, which also happens in our brain. In turn, LTM stores information that may be used in the future. In this case, LTM can be considered to be split into three parts: process memory, which stores primitives of movement and behavior; semantic memory, which is a database with information about the existing objects in the surrounding environment; and episodic memory, that stores past experiences, such as objectives or sequences of tasks that have been performed before. The grouping of these memory structures and the inherent flow of information between them is called Working Memory System.

1.3.5 Task realization and management

Another feature of human cognition is the fact that we can plan and execute sequences of actions in order to achieve an objective. Even in cases when the sequence of actions is not consciously accessible, often we are able to infer the sequence of actions to be realized, taking into account previous experiences [Meinert, 2008]. A relevant characteristic is the capacity of building complex sequences of actions by aggregating simpler tasks which, in turn, are built from even simpler tasks. We can say that there are several levels of complexity involved in our actions. Taking these features into account, we may conclude that it is necessary to implement a task management system that consists of multiple complexity levels and the aggregation of already known, simple tasks into more complex ones.

There has not been a lot of effort in the development of systems of this kind: it is still a big challenge. In traditional robotics, the execution of predefined procedures was enough for most tasks. The remaining cases could be solved by using artificial intelligence methods which could provide the ability to learn to realize the requested task. However, that kind of learning would always be based on probabilities and mathematical calculus that have no resemblance with the human way of learning. Another problem of this type of learning is that it is not appropriate for dynamic environments which are constantly changing.

According to Ratanaswasd et al. [2005], any actions must be carefully selected using information from past experiences, from the task itself and from the workspace. The authors presented a cognitive control model that joins the short- and long-term memories with a procedural memory (also with long term characteristics) in which the procedures of some tasks to be realized are stored. The stored procedures can then be combined with other tasks to realize more complex tasks. The management of the planning and the execution of tasks is done by a Central Executive Agent. The choice of tasks to be realized is then made through a reward system: the chosen tasks to be executed are the ones that are expected to provide the biggest rewards.

Another management, planning, selection and execution system was proposed by Alami et al. [2006]. This system has also a component which is responsible for controlling and monitoring the behavior of the robot. Also in this case, there is a series of simple tasks that are chained together to accomplish other tasks.

A slightly different way of trying to solve this problem was developed by Jung et al. [2007]. The authors created a database of knowledge and rules in specific scripts. The bigger the amount of stored scripts, the more flexible the robot will be and more tests can be executed. Flexibility comes from the fact that a task can be decomposed into basic scripts which can be recombined to perform a different task. However, there is also flexibility in the insertion of scripts in the robot's memory, making it easier to increase the robot's knowledge base. Besides that, by using this kind of system it becomes possible to contextualize the knowledge. All scripts include a pre and a post condition which allow the management system to start or terminate them. There is also a condition related to the workspace, so that the robot can select the information accordingly.

Apart from these three characteristics, the scripts also have sequences of scenes with pre and post conditions which, in turn, result in actions that also have their own pre and post conditions. Actions consist of sequences of primitive actions. This way of structuring the task realization is very versatile and allows the robot to have a behavior that follows some sort of narrative or a logic chain of causes and effects.

1.4 Overview of the thesis

This thesis is divided into six chapters, each one corresponding to a specific subject. It is important to stress, that some sections inside the chapters may be slightly repeated because they were based on publications that are already published or will be published soon.

The chapters are organized as follow:

- Chapter 2 (page 25) presents the initial studies, describing the use of biologically inspired vision for human-robot interaction that makes use of the stereo vision algorithm for robot navigation and obstacle/human detection and the initial approach to the biological descriptor presented in Chapter 3, in this case for gesture recognition. We also integrate head and hand detection models using SVMs trained on head and hand datasets. It also describes an improved saliency algorithm that integrates motion. The whole system is tested on a mobile robot platform that was used for testing all the work described in this thesis. This chapter was partially published in Saleiro et al. [2013b] and fully published in Saleiro et al. [2015].

- Chapter 3 (page 41) describes the process of learning binary biological local descriptors (improving the descriptor presented in Chapter 2). It explores how we can use cortical cell responses from cortical area V1 to code image areas around keypoints, which can also be extracted using the same cell responses. This allows for a completely biologically inspired keypoint extraction and description. The performance of the obtained descriptors is very promising, outperforming, by far, any other biologically inspired keypoint descriptor and also outperforms a SIFT-based non-biological descriptor. We also explore how the binary descriptors can be used for early vision processes like optical flow, stereo vision or object recognition. This chapter has been submitted to BioSystems international journal.

- Chapter 4 (page 71) describes a fast biological active vision model comprising stereo vision, saliency and object recognition. The included biologically inspired stereo model was specifically designed for real-time robotics applications and when evaluated using stereo testing datasets proved to outperform other biologically-inspired stereo models. The resulting disparity maps are also used as input for the saliency algorithm described in Chapter 2 that also uses color and motion as input. The saliency algorithm is used to select interesting image regions for further object recognition using the state-of-the-art biological descriptor presented in Chapter 3. Motor control of the robot head is done using attractor dynamics. The whole system is successfully tested in a child-sized robot in a real environment. An initial version of this chapter was fully published in Saleiro et al. [2014], and now is being prepared an updated version for a journal.

- Chapter 5 (page 107) describes a cognitive robot architecture with the biologically inspired active vision system presented in Chapter 4 for SLAM. It explores the use of cognitive robot framework that comprises the integration of biological vision models with motor control for active vision. The cognitive robot framework also includes a memory management system that mimics the human memory model and a task management system. Motor control of the mobile robot platform is made using attractor dynamics. An initial version of this chapter was partially published in Saleiro et al. [2012] and Terzić et al. [2013a] and is being prepared for publication in an international journal.

- Chapter 6 (page 127) provides a summary of major achievements, concluding remarks and ideas for future research.

Chapter 2

Biologically inspired vision for human-robot interaction

Abstract: Human-robot interaction is an interdisciplinary research area that is becoming more and more relevant as robots start to enter our homes, workplaces, schools, etc. In order to navigate safely among us, robots must be able to understand human behavior, to communicate, and to interpret instructions from humans, either by recognizing their speech or by understanding their body movements and gestures. We present a biologically inspired vision system for human-robot interaction which integrates several components: visual saliency, stereo vision, face and hand detection and gesture recognition. Visual saliency is computed using color, motion and disparity. Both the stereo vision and gesture recognition components are based on keypoints coded by means of cortical V1 simple, complex and end-stopped cells. Hand and face detection is achieved by using a linear SVM classifier. The system was tested on a child-sized robot.

2.1 Introduction

It is expected that, in the future, robots will employ a growing number of roles in society. Currently robots are mainly used in factory automation, but they are also deployed in service applications, medical assistance [Messias et al., 2014], schools [Saleiro et al., 2013a] and entertainment, among other application fields. As they start to roam among us, there will be a need to interact with them in easy and natural ways. Human-robot interaction (HRI) research is therefore attracting more and more attention. Researchers are trying to develop new, easy and natural ways of programming robots, either by teaching them by manipulating a robot's hardware manually [Kronander and Billard, 2014] or by creating programming interfaces so simple that even children can use them at school [Saleiro et al., 2013a]. However, programming a robot still requires some skill that must be learned. In a world where robots navigate next to us, it will be necessary to be possible to interact with them effortlessly, using voice commands or gestures. In an ideal situation, robots should even be able to perceive some of our intentions by analyzing the motions of our body.

The analysis and recognition of static hand gestures for HRI has been an interesting research area for some time and there have been many approaches. Some are intrusive, requiring the user to use specially designed gloves [Kumar et al., 2012], while others are less intrusive, but still require specific hardware like Leap Motion [Han and Gold, 2014] or Microsoft Kinect [Qian et al., 2013]. Other approaches rely on simple cameras and computer vision methods. A simple solution can be based on skin color segmentation and matching of previously stored gesture templates [Saleiro et al., 2013b]. More complicated is to extract the skeleton of the hand [Ionescu et al., 2005] and use this for matching. For dynamic gestures there are methods that perform tracking and motion detection using sequences of stereo color frames [Ghaleb et al., 2014], or gestures are characterized by using global properties of trajectories described by a set of keypoints [Suk et al., 2010]. Although many solutions may work quite well for a specific type of application, they are too simple for more complex gestures. In addition, they are often limited by fixed hardware devices or lighting conditions. In order to use gesture analysis and recognition for human-robot interaction, a system that is able to work under most lighting conditions and almost anywhere is needed.

In our previous work [Saleiro et al., 2013b] we developed a biological and real-time framework for detecting and tracking both hand and head gestures.

In this chapter we present an extension of the system and also add new features to improve the system for human-robot interaction. The previously developed framework is based on multi-scale keypoints detected by means of models of cortical end-stopped cells [Rodrigues and du Buf, 2006, 2009]. We have improved the previous annotation of keypoints by using a fast binary descriptor that allows for fast and robust matching. We also added a combined disparity and motion saliency process so that the robot can initially focus on the hands of the user and track them. Gesture recognition is performed by matching the descriptors of the detected keypoints with the descriptors of previously stored templates. We also integrated a head and hand detector based on a linear SVM classifier.

The robot uses stereo vision for navigation, which also allows it to detect obstacles. Every time it finds an obstacle in front of it, it looks up and searches for the user's head in order to start the interaction. The robot attempts to center its own camera on the user's face, and then starts performing hand detection and gesture recognition. To detect a face/hand we employ a modified HOG (Histogram of Oriented Gradients) descriptor combined with responses of complex cells and a linear SVM to code the shape. The face and hand detectors were trained and evaluated on the FaceScrub dataset Ng and Winkler [2014] and the Oxford hand dataset [Mittal et al., 2011], respectively. The developed HRI system does not need any prior calibration and has been designed to run in real time.

2.2 Biologically inspired HRI system

In this section we describe all components of the developed system: (a) keypoint descriptor, (b) stereo vision for navigation and obstacle detection, (c) visual saliency, (d) face and hand detection, and finally (e) gesture recognition.

2.2.1 Keypoint descriptors for gesture recognition

In cortical area V1 there are simple, complex and end-stopped cells [Rodrigues and du Buf, 2009], which are thought to be responsible for part of the process of coding the visual input: they serve to extract multi-scale line, edge and keypoint information (keypoints are line/edge vertices or junctions and also blobs). In the present section we briefly describe multi-scale keypoint detection and the fast binary descriptor that we designed for matching keypoints. The descriptor is also based on V1 cell responses.

Keypoint detection: Responses of even and odd simple cells, which correspond to the real and imaginary parts of a Gabor filter [Rodrigues and du Buf, 2009], are denoted by $R^E_{s,i}(x,y)$ and $R^O_{s,i}(x,y)$, i being the orientation (we use $4 \leq N_\theta \leq 12$). The scale s is defined by λ, the wavelength of the Gabor filters, in pixels. We use $4 \leq \lambda \leq 12$ with $\Delta\lambda = 4$. Responses of complex cells are obtained by computing the modulus $C_{s,i}(x,y) = [\{R^E_{s,i}(x,y)\}^2 + \{R^O_{s,i}(x,y)\}^2]^{1/2}$. The process of edge detection is based on responses of simple cells: a positive or negative line is detected where R^E shows a local maximum or minimum, respectively, and R^O shows a zero crossing. In the case of edges the even and odd responses are swapped. Lateral and cross-orientation inhibition are used to suppress spurious cell responses beyond line and edge terminations, and assemblies of grouping cells serve to improve event continuity in the case of curved events. On the other hand, keypoints are based on cortical end-stopped cells [Rodrigues and du Buf, 2006] and they provide important information because they code local image complexity. Since keypoints are caused by line and edge junctions, they are usually located at interesting locations of the image. When combined with a proper descriptor, they can be very useful for object categorization [Terzić et al., 2013a]. There are two types of end-stopped cells, single and double. These are applied to $C_{s,i}$ and are combined with tangential and radial inhibition schemes to obtain precise keypoint maps $K_s(x,y)$. For a detailed explanation with illustrations see Rodrigues and du Buf [2006] and Terzić et al. [2013a].

Binary keypoint descriptors: Creating a biological descriptor for keypoints is not a trivial task, mainly because responses of simple and complex cells, which code the underlying lines and edges at vertices, are not reliable due to response interference effects [du Buf, 1993]. Therefore, the responses in a neighborhood around a keypoint must be analyzed, the neighborhood size

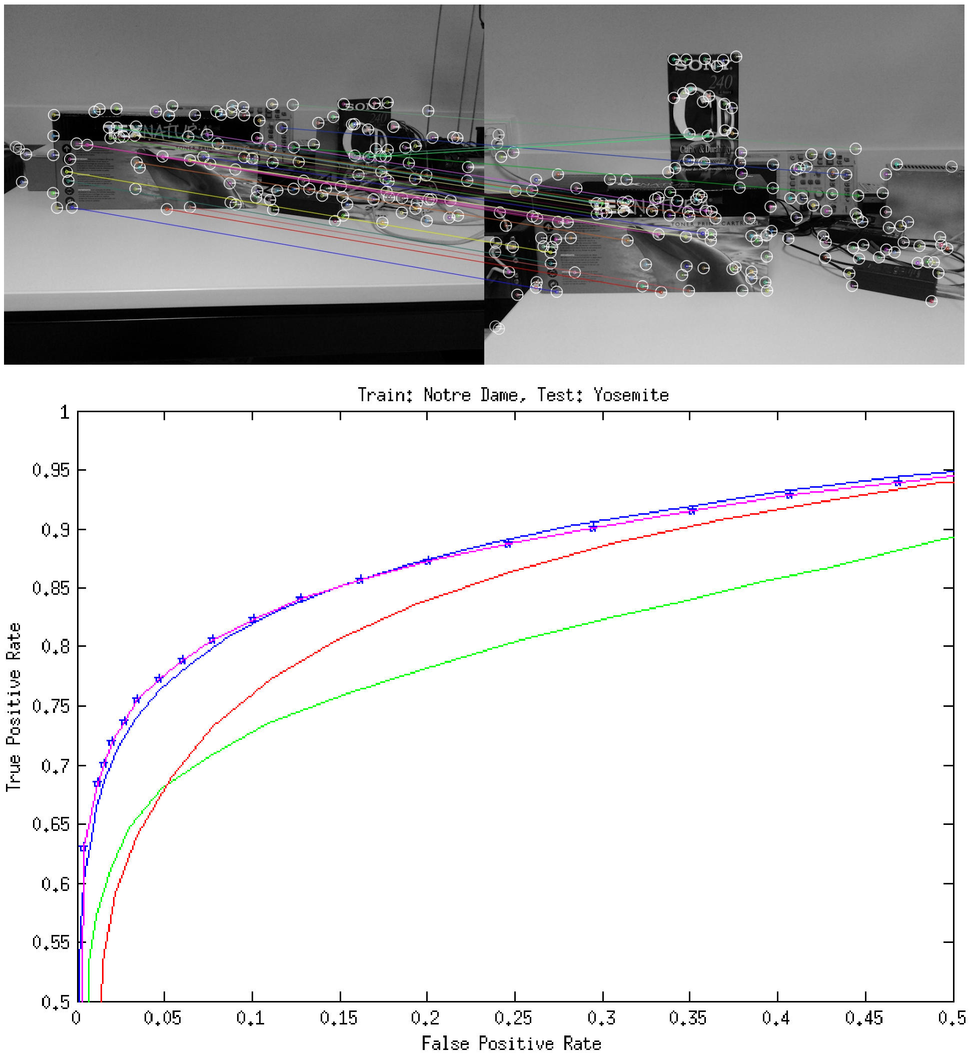


Figure 2.1: Top: Example of matching. Bottom: comparison between our descriptor (blue), BRISK (green), BRIEF (red) and LDAHash (pink) over 200,000 patches.

being proportional to the scale of the cells. In our approach we developed a 128-bit binary keypoint descriptor based on the responses of simple cells, each bit coding the activation level of a single cell. Also, from a computational point of view, a binary descriptor is much faster to compute and to match than a floating-point one. This method is an improvement of the previous method [Saleiro et al., 2014].

We start by applying maximum pooling in a circular region around each keypoint, followed by zero-mean normalization and extraction of the maximum cell responses in 8 filter orientations. Then we combine the extracted values by a weighted sum, using a weight matrix previously learned using LDAHash [Strecha et al., 2012] on the Notredame dataset [Brown, 2011]. Finally, we apply a threshold vector, also previously trained on the Notredame dataset, to set each of the 128 bits to 1 or 0. The resulting descriptor is a huge improvement of the previous one [Saleiro et al., 2014]. It is comparable to the SIFT-based LDAHash descriptor in terms of performance when tested on the Yosemite dataset [Brown, 2011]. The developed descriptor significantly outperforms other biologically-inspired descriptors. In terms of processing time, the descriptor is also very fast to compute and to match. Figure 2.1 (top) shows one example of matching using the present descriptor. The bottom graph shows a comparison between our descriptor (128 bits), BRISK (512 bits) and BRIEF (128 bits) and LDAHash (128 bits) over 200,000 patches of the Yosemite dataset.

2.2.2 Stereo vision for robot navigation and obstacle detection

Stereo vision is a fundamental process for robot navigation: it allows the robot to detect open spaces, obstacles on its path and to estimate the distance to those obstacles. It can also be useful for computing visual saliency. The algorithm we used to generate the disparity maps is the same as previously used by Saleiro et al. [2014]: (a) resize the left and right images to a small size (160×120); (b) extract complex cell responses on circles around each pixel; (c) compare each pixel P in the left image to the next K pixels in the right image on the same line and starting from the same position P as in the left image (we used $K = 35$); (d) use the Hamming distance to find the best-matching pixels; and (e) apply median filtering (5×5 kernel) to reduce noise due to wrong matches. The computed disparity maps are then thresholded using a threshold value, t_d, to find nearby obstacles (we used $t_d = 70$). Whenever the robot detects an obstacle by using the thresholded disparity maps, it looks up and evaluates if it is a person by trying to find a human face using a linear SVM classifier, as described in Section 2.2.4. Figure 2.2 (top) row shows an example of stereo images acquired by the robot and the respective disparity map.

2.2.3 Visual saliency

Visual saliency is also an important component of the real-time vision system, since by using it the robot can select important regions to process instead of processing entire images. The generated saliency maps are also useful to segregate hands from the background, to reduce clutter and to improve gesture recognition rates. The visual saliency algorithm we developed is an improvement of the one described in Saleiro et al. [2014] and combines three different features: color, disparity and motion. The three features are processed separately and then merged into a single saliency map with equal weights. The top row of Fig. 2.2 shows an image of a person in front of the robot (left and right frames) and the resulting disparity map. The middle row shows the motion (left), color (middle) and disparity(right) saliency maps. The bottom row shows the combined saliency map, the thresholded map and the selected regions. Details are explained below.

Color Saliency: We build a stack of 6 retinotopic maps representing different channels in CIE L*A*B color-opponent space, which is based on retinal cones and provides a standard way to model color opponency. The first three channels code white, green and blue. The remaining three channels complement the first ones and represent black, red and yellow. After computing the maps, a stack of bandpass Difference-of-Gaussians (DoG) filter kernels with $\sigma_+ \in \{5, 10, 20\}$ and $\sigma_- = 2\sigma_+$ are used for blob detection. Since a saliency map does not need to be detailed, we compute them using subsampled color images for faster processing.

Disparity Saliency: To generate the disparity-based saliency map, we use the disparity map as described in Section 2.2.2, and extract a single disparity layer where pixels with disparity $d = 100$ get the maximum value $M = 20$ and pixels with disparities bigger or smaller than d get smaller values according to the difference from d. After this step we apply the same filtering used for the color-based saliency maps.

Motion Saliency: To compute the motion-based saliency map, we first

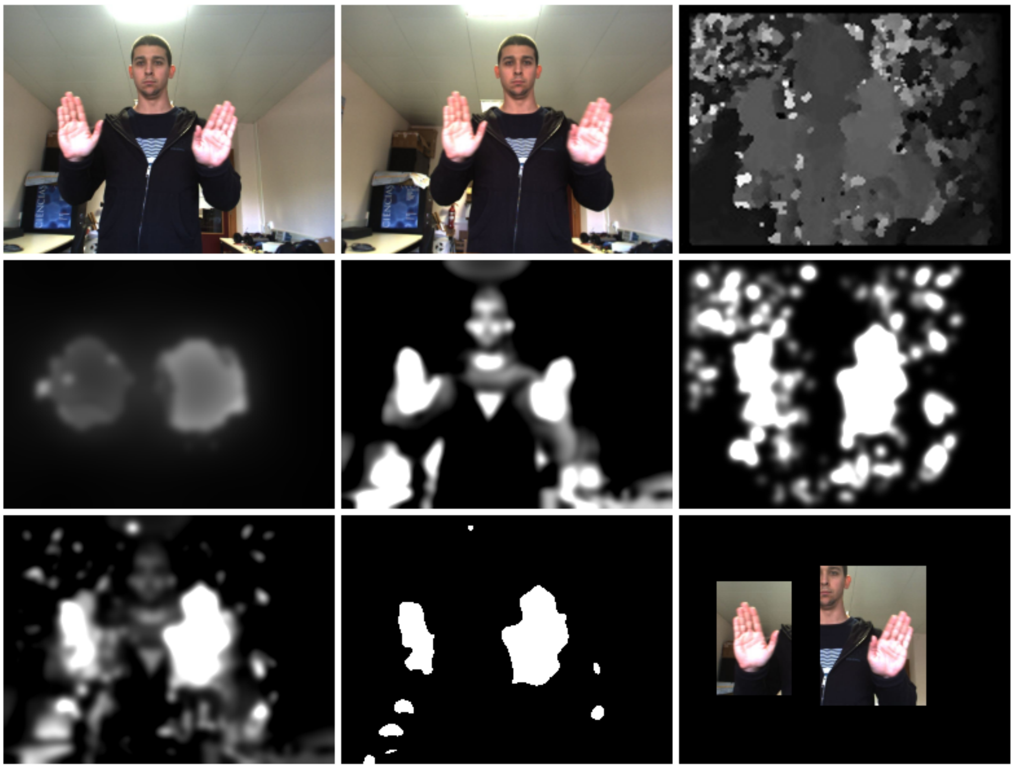


Figure 2.2: Top row: stereo images acquired by the robot camera and the disparity map. Middle row, from left to right: motion saliency, color saliency and disparity saliency. Bottom row, from left to right: resulting saliency map, thresholded saliency map (using threshold value $t_f = 200$) and selected regions after blob detection and region growing.

calculate the optical flow using Farneback's method for every pixel [Farneback, 2003]. Then we process the flow's magnitude and direction separately, creating a feature stack of magnitudes and orientations of motion. The feature stack has 3 maps representing the magnitude (speed) of motion and it also has 8 maps representing 8 different directions of motion with either 0 or the magnitude of movement at that orientation. In practice, 0 occurs when a pixel is not moving in the preferred direction or with preferred speed. When a pixel is moving in the preferred direction or with preferred speed we consider the value to be the value of the pixel at that location. Objects moving in a certain direction with certain speed will thus cause large coherent regions in one of the maps of the stack.

Final Saliency Map: The final saliency is the equally-weighted sum of the three normalized saliency maps. After computing this map we threshold it using threshold value $t_f = 200$ to get only the nearest regions and then apply the fast blob detection algorithm from Saleiro et al. [2009]. After blob detection, only the two biggest blobs are kept. Each blob is converted into a square region that afterwards is grown by 15 pixels in all directions, so that there is enough margin to apply the Gabor filters to extract the simple cell responses and keypoints and to build keypoint descriptors.

2.2.4 Face and hand detection

Detection of faces and hands is achieved by coding responses of cortical complex cells within a region into a feature vector and by using a classifier (linear SVM) to predict whether a face/hand is inside the detection region or not. The process is similar to that of Dalal and Triggs [2005], but in our case the detection process employs a single Gabor filter scale ($\lambda = 6$) and 8 orientations ($N_\theta = 8$) for the complex cells (see Section 2.2), but also in combination with several scales of the HOG-like features (several sizes) over the entire image, and then a sliding window (6×6 blocks) to scan all blocks inside the sliding window's region.

At each layer, the pooling cell size is increased, but the detection window size and the block's size remain the same. To this purpose, we use between 6×6 and 10×10 pixels per cell with a stride of 1 pixel. Finally, non-maximum suppresion is applied to eliminate multiple detections of the same face/hand (see below). We used the FaceScrub dataset [Ng and Winkler, 2014] and the Oxford Hand dataset [Mittal et al., 2011] to train and evaluate our face and hand detectors. For each detector we trained an initial classifier using the positive and a random set of negative examples, then we used it to scan over images not containing faces or hands in order to collect false positives, and then we did a second round of training by including the false positives into the negative training set. The FaceScrub dataset consists of 107,818 face images of 530 celebrities (265 male and 265 female), although only 80,659 faces could be downloaded successfully and a few samples were unusable for training. For negative samples we took 100,000 42×42 pixel patches from random images of the SUN database [Xiao et al., 2010]. For training our hand classifier we

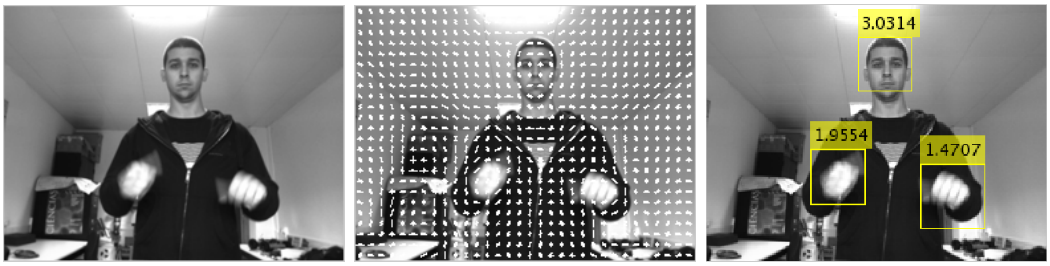


Figure 2.3: (Left to right) a person in front of the robot in gray scale (left) coded by HOG-like features (middle) with detected face and hands (right).

used the Oxford hand dataset which contains 13,050 annotated hand instances (26,100 mirrored samples). Again, we took 30,000 random 42×42 pixel patches from the SUN database as negative samples.

HOG-like features: A modified version of the Dalal and Triggs HOG features is used for person detection. In their implementation, Dalal and Triggs [2005] used the RGB color space with no gamma correction, 1D gradient filters, linear gradient voting into 9 orientation bins, 16×16 pixel blocks of four 8×8 pixel cells, a Gaussian spatial window with $\sigma = 8$ pixels, L2-Hys block normalization, a block spacing stride of 8 pixels, and a 64×128 detection window for training the linear SVM classifier for pedestrian detection. Here we use an adapted and slightly modified version of the previous procedure for face and hand detection: (a) we use only grayscale information for speed purposes, (b) complex cell responses are used as gradient information, (c) no Gaussian window is applied, (d) the L2-norm is used instead of L2-Hys and (e) a 42×42 detection window is used. Complex cell responses provide a good alternative for gradient information since they are also robust to noise. In addition, linear gradient voting can be skipped because of the cells' oriented responses. We use 12×12 pixel blocks of four 6×6 pixel cells with 50% block overlap; see below for parameter assessment and evaluation. Our selection ensures optimal performance while also complying with Dalal and Triggs' recommendations [Dalal and Triggs, 2005] of having many orientation bins in combination with moderately sized, strongly normalized and overlapping descriptor blocks. See Fig. 2.3 (middle) for the bio-inspired HOG-like features.

Classification: To detect a face or a hand, features in a detection window are classified using a linear SVM. The face and hand classifiers are trained using the FaceScrub dataset and the Oxford hand dataset, respectively, with a soft linear SVM (C=0.01) using LIBLINEAR [Fan et al., 2008]. As mentioned above, we used a 42×42 pixel detection window for training both classifiers, since the sizes of hands and faces in the image relative to the robot are similar. This results in 6×6 blocks to be used by the classifiers across the image at all scales. We found that a detection window of 42×42 pixels for training constitutes a good trade-off between performance and running speed. Increasing the detection window's size beyond 42×42, although increasing detection performance, has a significantly higher computational cost due to more features being used for classification.

Non-maximum suppression: The detection window is applied to salient image regions using a sliding window approach where multiple detections of the same head or hands often occur. To remove multiple detections due to the sliding window, a non-maximum suppression technique is applied to discard overlapping windows: when two windows overlap at least 50%, the window with the weakest classification response is discarded. To this purpose, we use the unsigned SVM classification output in order to determine a window's classification response. Figure 2.3 (right) shows detected regions of face and hands after non-maximum suppression.

Concerning performance evaluation and optimization, several factors have been evaluated in the classifier training stage, namely Gabor filter scale (λ), number of orientations (N_θ), cell size, block size and overlap. Smaller sets of 2000 positive and 2000 negative random samples for training, and 1000 positive and 1000 negative random samples for testing were used for cross-validation and parameter optimization for both classifiers. We used detection error trade-off (DET) and miss rate (better: 1.0 - Recall) measures to quantify the performance.

First, the scale of the cortical cells was analyzed in order to determine the optimal λ. Figure 2.4 (top-left) shows the overall performance of seven different scales $\lambda = [4, 10]$. Smaller scales yield better performance than bigger scales, mainly due to lines and edges being better encoded by smaller filters. Moreover, by choosing a smaller λ the processing time decreases. Here, $\lambda = 4$ performed best. Figure 2.4 (top-middle) shows the performance impact of the number of orientations used ($N_\theta \in [4, 12]$) in the HOG-like feature bins. Increasing the number of orientations beyond 8 does not improve performance significantly. Therefore, for the final classifier we chose $N_\theta = 8$ orientations which gave a 1.4% and 8.03% miss rates for face and hand detection, respectively. Three other key factors taken into account were the block size vs. cell size vs. block overlap. Figure 2.4 (middle and bottom) shows three graphs with 0% (left), 25% (middle) and 50% (right) block overlap, with block sizes ranging from 1×1 to 4×4 and pooling cell sizes from 4×4 to 10×10 for faces (middle) and hands (bottom). From all tested combinations, block sizes of 2×2 with 6×6 pooling cells and 50% overlap, gave the best performance with a 1.1% miss rate for face and 8.6% for hand detection. Figure 2.4 (top-right) shows the overall performance of the two detectors.

2.2.5 Gesture recognition

In order to be able to recognize gestures, the robot keeps in its memory a small set of templates of different hand gestures. These templates have been prepared prior to robot operation, and by applying exactly the same processing as done during real-time robot operation. At the moment we use a set of 7 different gestures for both hands, with several samples for each gesture with different backgrounds and sizes. After finding the hand regions, the robot then processes those regions for keypoint extraction and their descriptors. Descriptor matching is based on the 128-bit Hamming distance: when the distance between two keypoint descriptors is smaller than 30, a match is accepted. When at least 4 matches between the acquired hand region and a template are found, the robot assumes that the gesture that corresponds to the matched template has been detected. For faster computation time and higher reliability of the system, we extract keypoints at scale $\lambda = 12$ and use simple cell responses at scale $\lambda = 6$ for the descriptors. By extracting keypoints at a coarser scale, they become more stable. By using scale $\lambda = 6$ on extracted keypoint locations, more detail is available to describe the keypoints.

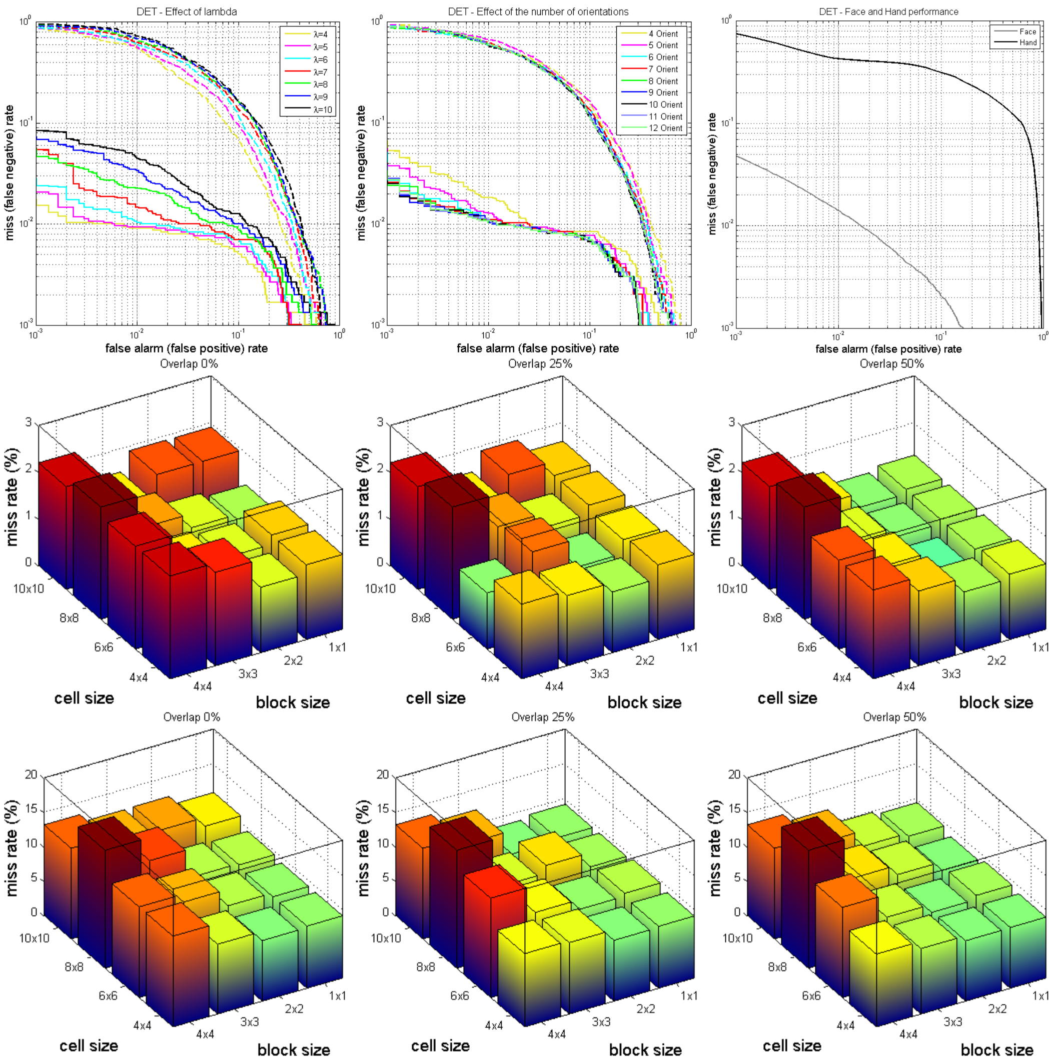


Figure 2.4: HOG-like feature parameter performance. Top row: effects of λ (left) and the number of orientations (middle) in face (solid lines) and hand (dotted lines) detection. The right plot shows the overall performance of our face and hand detectors. Middle and bottom rows, left to right: block vs. cell size combinations with 0%, 25% and 50% block overlap, and with block size from 1×1 to 4×4 and cell size from 4×4 to 10×10 (bluer is better) for face (middle) and hand (bottom) detectors. The best result is given by 2×2 blocks with 6×6 cell size and 50% overlap with 1.1% miss rate for face and 8.6% for hand detection

2.3 Tests and results

To test the developed system we used a child-sized Pioneer 3DX robot, equipped with a Bumblebee-2 stereo camera, a PhantomX robot turret for pan and tilt movement of the camera, and ultrasonic and laser rangefinder sensors (see Fig. 2.5 left). The Bumblebee-2 camera captures images at a resolution of 1024 × 768 pixels. The range sensors are only used for emergency collision avoidance, not for navigation. A structure has been mounted on the robot in order to make it taller, providing the point of view of a child with a height of 115 cm and eyes at 110 cm. The robot has been set up with ROS (Robot Operating System). Although the robot is of mecanoid type, its pan and tilt system combined with the stereo camera convey the idea that it has a neck and a head with two eyes. Since it is programmed to focus on a person's head in the center of the image, it seems like it is looking the person in the eyes. This makes it much more engaging than a robot with static cameras and no neck movements.

Figure 2.5 right shows some results: on the left we can see the mobile robot platform used for testing; on the middle left column we show two examples of gesture recognition using the extracted keypoints and their descriptors; on the middle right column we show some of the gesture templates; and on the right we show some examples of head and hand detection using the SVM classifiers.

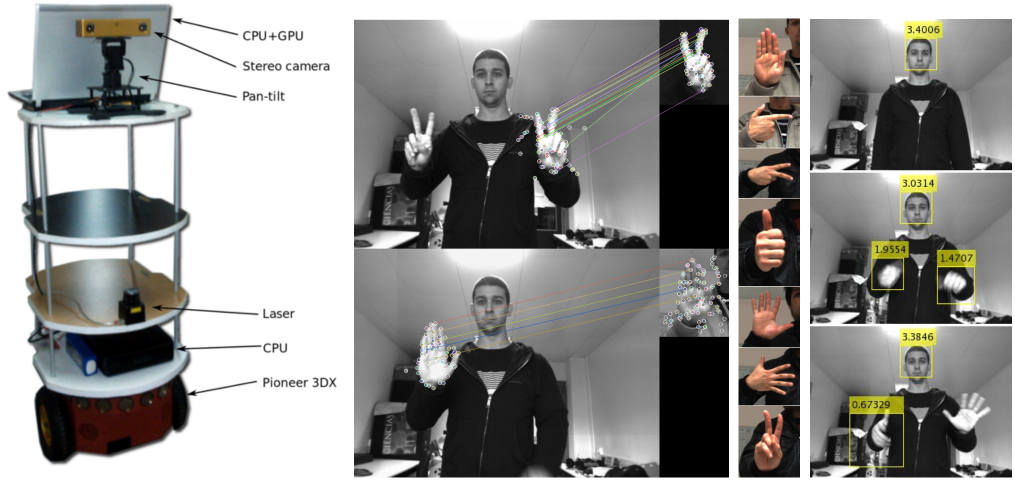


Figure 2.5: Left: robotic platform used for testing. Middle left: examples of recognition of two different gestures. Middle Right: some of the gesture templates. Right: detection of heads and hands using the SVM classifiers.

The system proved to work quite well, being able to recognize the 7 different gestures for both hands in most situations. At the start it failed to recognize some of them but as we added more gesture templates with different sizes and backgrounds the gesture recognition was improved.

As usual with most vision systems, we noticed that under bad illumination the system can sometimes fail to detect hands or head and thus may not be able to recognize gestures. Failure in detection of hands and recognizing gestures could also happen during fast hand movements, since in these cases the image of the hands is blurred.

2.4 Discussion

In this chapter we presented a biologically inspired vision system for human-robot interaction. It allows the robot to navigate and to evaluate if an obstacle is, in fact, a person by trying to find a human face whenever it encounters an obstacle. The system also allows the robot to direct its attention to important visual areas using an attention model which is based on color, disparity and motion. The presented methods allow the robot to identify human hands in salient regions and then recognize the gestures being made by using keypoints and keypoint descriptors based on V1 cell responses. Although biologically inspired methods usually require a high computational power and long computation time, due to the many filter kernels at several orientations and scales, by choosing only a few scales the system is able to run in real time. The SVM classifiers showed to be fast in detecting faces and hands in an image using a sliding window. Faces had the best performance overall with few false negatives, while hands had less performance in detection mainly due to the large intra-class variability. The keypoints and their descriptors proved to be quite robust and work quite well for recognizing the gestures previously stored as templates. Although sometimes few keypoints were matched to wrong keypoints, most of them were matched to the correct ones. These wrong matches can easily be eliminated by using some geometry between groups of keypoints to validate or invalidate matches. Experiments showed that the system can now be programmed to execute different actions according to specific hand gestures.

As future work we plan to add more gestures to the system, increase the precision such that individual fingers can be detected, and to integrate facial expression recognition in order to allow the robot to act according to a person's mood. We also plan to use the ability of recognizing gestures and facial expressions as a way to teach the robot simple tasks through reinforcement learning: by using different gestures or facial expressions it will be possible to tell the robot that it is doing the right or the wrong thing. Another part of future work consists of integrating the binary keypoint descriptor into the GPU implementation of the keypoint extractor in order to free the CPU for other future developments. Future work will also address motion prediction, a process that occurs in cortical area MST.

Chapter 3

Learning biological binary keypoint descriptors

Abstract: Learning robust keypoint descriptors has become an active research area in the past decade. Matching local features is not only important for computational applications, but may also play an important role in early biological vision for disparity and motion processing. Although there were already some floating-point descriptors like SIFT and SURF that can yield high matching rates, the need for better and faster descriptors for real-time applications and embedded devices with low computational power led to the development of binary descriptors, which are usually much faster to compute and to match. Most of these descriptors are based on purely computational methods. The few descriptors that take some inspiration from biological systems are still lagging behind in terms of performance. In this chapter, we propose a new biologically inspired binary descriptor. Built on responses of cortical V1 cells, it significantly outperforms the other biologically inspired descriptors. The new descriptor can be easily integrated with a V1-based keypoint detector that we previously developed for real-time applications.

3.1 Introduction

During the last decades, the modeling of processes in vision is a research field that has been attracting more and more attention. Models of simple, complex and end-stopped cells in visual area V1 have been developed and these models have been used for line, edge and keypoint detection [Rodrigues and du Buf, 2006, 2009]. Lines and edges have been successfully used for multiple applications like object segregation, scale selection, saliency maps and disparity maps [Rodrigues et al., 2012], optical flow [Farrajota et al., 2011b], face detection and recognition [Rodrigues and du Buf, 2006], facial expression recognition [Sousa et al., 2010], etc. The model for keypoint detection was computationally too expensive to be used in real-time applications at the time it was developed, but recent advances in computer hardware and code optimizations led to a much faster model that can now be used in real time [Terzic et al., 2015]. However, although keypoints indicate the location of specific events in an image, they do not contain information about the regions where they are located: the local image structure. For keypoint matching across images it is necessary to take the local image information around a keypoint and to code this, building a local image descriptor for every single keypoint. These descriptors must be robust to image variations, like changes in illumination, rotation, translation, etc.

Descriptors characterize local image regions by means of a compact numerical representation which should be consistent under a wide range of image transformations. A metric such as Euclidean distance in descriptor space can then provide a measure of similarity between two image patches. Robustness and reliability of modern descriptors have made them indispensable in countless applications of Computer Vision, such as image stitching, optical flow and object recognition and tracking. Although there are already some very robust and reliable histogram-based descriptors such as SIFT [Lowe, 2004] and SURF [Bay et al., 2008], which are floating-point descriptors that can be used with the Euclidean distance metric, there has been a more recent trend towards binary descriptors such as ORB [Rublee et al., 2011] and BRISK [Leutenegger et al., 2011]. These encode a region by a bit vector and typically use the

Hamming distance in the matching process[1]. These descriptors have therefore many advantages over the floating-point ones: reduced memory and bandwidth requirements, and faster extraction and matching. This makes them ideal for a growing number of real-time applications and embedded devices with low computational power. Some recent binary descriptors have been shown to be on par with histogram-based methods, or even excelled them in some test scenarios [Trzcinski et al., 2013].

It is generally accepted that the brain actively constructs explanations for its sensory inputs [Bastos et al., 2012], and several predictive coding models have appeared in the literature [Larkum, 2013; Mumford, 1992; Mumford and Lamme, 1998]. There has been much research into hierarchical models and intermediate representations for fast object recognition [Fukushima, 2003; Serre et al., 2007], but feature matching may occur already at the earliest stages of vision, especially in the case of stereo disparity and motion processing. We note that binary descriptors can neatly represent excitation patterns of neural populations, and that their dissimilarity in terms of Hamming distance can be easily computed by networks of integrate-and-fire neurons [Vogels and Abbott, 2005]. Pairwise comparison of patches at the early stages of vision can thus be seen as a comparison between simple and efficient codes built directly from the responses of V1 cells. This could explain the speed and accuracy of feature matching at the earliest stages of vision. The remaining challenge then is to create a neuronally plausible method for efficiently extracting binary codes from V1 responses, which can compare in performance to the best descriptors in Computer Vision. Some recent binary descriptors such as FREAK [Alahi et al., 2012] and BRISK [Leutenegger et al., 2011] have been partly motivated by biological vision, but their performance is still poor compared to the state of the art in descriptor matching [Trzcinski et al., 2013]. Our work addresses this problem by learning receptive fields needed to construct good binary codes based on the responses of cortical V1 cells.

Two things are often considered to be important when constructing a binary

[1] When using a processor with SSE4.2 instructions, matching can be done by using only two instructions: XOR and POPCNT.

descriptor: (i) compact representation, thus reducing storage and bandwidth costs, and (ii) low redundancy between bits, since the Hamming metric implicitly assumes their independence. In this chapter, we present a compact binary biological local descriptor built from responses of cortical V1 cells. The coding of the responses is trained on 200K pairs of image patches using LDAHash [Strecha et al., 2012]. This allows to create a projection matrix that minimizes the in-class covariance and maximizes the covariance across classes. Each bit of the descriptor is then computed through a linear combination of cell responses. The result of the linear combination is thresholded to binarize the output. From a biological point of view, we can consider each bit as a cell that takes input from multiple V1 cells, each one with a different weight. Depending on the linear combination of the inputs, the cell may fire (output one), or not (zero). The resulting binary vectors can then be matched by using the Hamming distance. We will show that our descriptor outperforms all biologically inspired descriptors proposed in the literature, and that it approaches the matching performance of the best binary descriptors, even outperforming the SIFT-based LDAHash descriptor. During the development of the proposed descriptor we performed extensive tests, trying cell responses with several combinations of scales, with different maximum pooling steps and different pooling sizes. We also experimented with applying the same approach to biological HMAX and HMIN features instead of V1 cell responses. Since the developed descriptor is compact and takes V1 cell responses as input, it is fast to compute and it can be used for real-time applications. By integrating the new descriptor with the previously developed V1 keypoint extractor Terzić et al. [2013a], a biologically motivated keypoint extraction, description and matching pipeline is achieved.

In the next section related work concerning keypoint descriptors will be described. Section 3.3 details the methods that were used to build binary descriptors, and Section 3.4 deals with the optimization experiments in order to improve the performance. This ends with performance comparisons with other biological and non-biological descriptors and some considerations concerning the processing time (Section 3.5). In the last section (4.9) conclusions and suggestions for future work will be presented.

3.2 Related work

Image regions are often described in terms of gradient histograms collected in predefined sampling regions [Bay et al., 2008; Dalal and Triggs, 2005; Lowe, 2004]. Although such histograms have been considered the absolute state of the art for a long time, they have a large dimensionality. They require a lot of storage capacity and this results in comparably slow matching. Subsequent work has reduced dimensionality by using subspace methods [Ke and Sukthankar, 2004] or compression [Chandrasekhar et al., 2009], while keeping most of the benefits of the original methods. An appealing alternative approach is based on binary descriptors which can be matched by using the Hamming distance. Early binary descriptors were based on the BRIEF descriptor, which employed pairwise comparisons of amplitudes, i.e., gradients or partial derivatives [Calonder et al., 2012]. BRIEF was later modified to take advantage of discriminative projections [Trzcinski and Lepetit, 2012] and keypoint dominant orientation [Rublee et al., 2011]. Improved sampling patterns were shown to boost performance while still using gradients [Alahi et al., 2012; Leutenegger et al., 2011]. Descriptors such as BRIGHT [Iwamoto et al., 2013] introduced a variable bitstring length for mobile applications. An alternative, but very popular way to obtain binary descriptors, is by quantizing more complex descriptors, as in [Strecha et al., 2012; Gong et al., 2013a,b].

Recently, focus has shifted towards learning efficient descriptors, which was aided by the availability of large datasets of registered image patches [Brown et al., 2011]. Approaches include learning the Mahalanobis distance [Jain et al., 2012], supervised hashing [Liu et al., 2012; Wang et al., 2010] and LDA [Strecha et al., 2012; Gong et al., 2013b]. More recent research has moved into learning not only feature weights, but also the best pooling strategy [Brown et al., 2011; Simonyan et al., 2014]. Most impressive results so far have been achieved by a boosting approach [Trzcinski et al., 2013], which jointly optimizes both feature weights and the pooling strategy. Like in our approach, each subsequent bit is optimized to correct the mismatches which are left after learning the previous bits.

While there is some recent work on biologically plausible keypoint detection [Terzić et al., 2013a,b], we are not aware of a completely biologically plausible approach for extracting low-level image descriptors. Biological features are usually formulated with object recognition in mind, creating a hierarchy with intermediate features of increasing invariance, such as alternations of pooling and maximum operations which simulate stacked layers of simple and complex cells, the HMAX model [Fukushima, 2003; Serre et al., 2007], or intermediate-level convolutional kernels in deep neural networks [Schmidhuber, 2012]. However, these features were optimized for object classification, not feature matching. While BRISK [Leutenegger et al., 2011] and FREAK [Alahi et al., 2012] use biologically-motivated sampling patterns, they make no use of cortical simple and complex cells. Descriptor learning of Brown et al. [2011] comes closest, by using cortically-inspired filters similar to HMAX [Serre et al., 2007]. Binary trees were used for efficient histogram compression by Chandrasekhar et al. [2009], but the resulting descriptors require a special matching procedure and they are based on histograms instead of biological features.

The novelty of the present work lies in two aspects: (i) only biologically plausible V1 features will be used, so the descriptor can be implemented using a neural architecture; and (ii) to our knowledge, we present the first system that performs keypoint detection, feature extraction and matching using only biologically plausible features and processes. We note, like Brown et al. [2011], that cortical cells perform many operations which can be used for descriptor coding: they are orientation and scale selective, they pool over spatial regions of varying sizes and shapes, and they respond strongly to discontinuities (gradients). Unlike most descriptor learning approaches, here the goal is not to create a short descriptor; we are more interested in creating a simple coding mechanism that can be used for disparity and optical flow in the earliest stages of vision and that can also be applied in some object recognition applications. In this chapter, we show that the coding of V1 features can summarize the local image content by relatively short binary codes that allow for robust patch matching, outperforming any other biologically inspired image descriptors and closely approaching the non-biologicals but state-of-the-art descriptors.

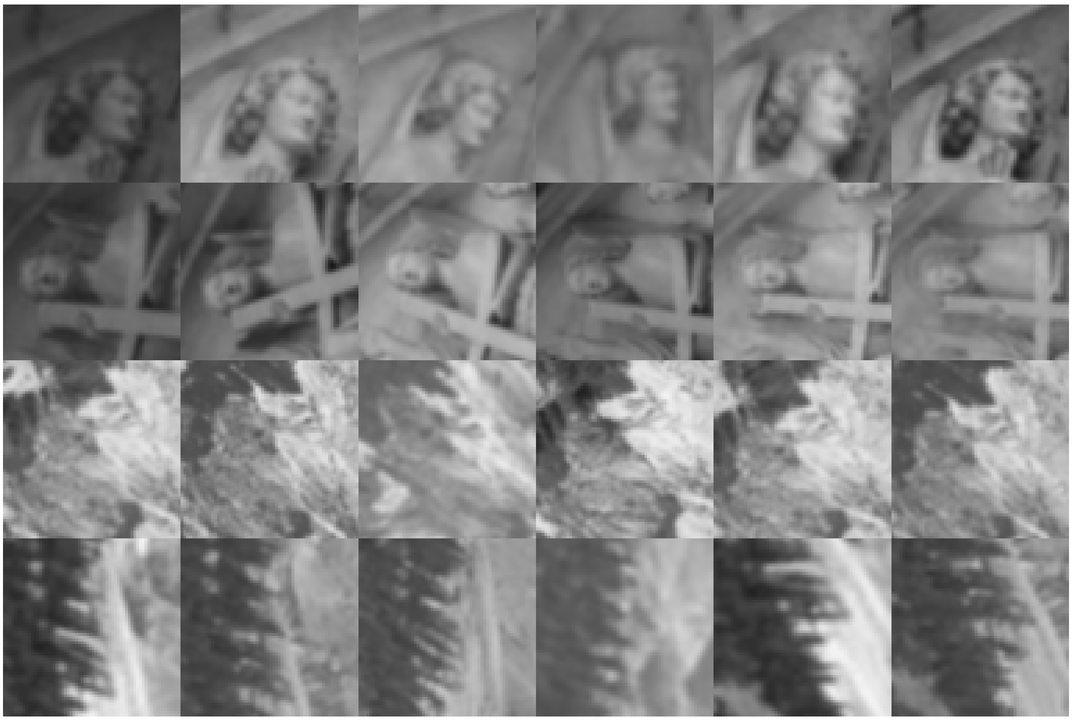


Figure 3.1: Sample patches from the Notre Dame (top two rows) and Yosemite (bottom) datasets that show small differences in terms of scale, displacement, rotation, contrast and brightness.

3.3 Method

In order to build and test the descriptors we used two datasets: Notre Dame (the church in Paris, France) and Yosemite (the Natural Park in California, USA) [Brown, 2011]. Both sets contain image patches sampled from 3D reconstructions of interest points, i.e., 64 × 64 pixels in grayscale, with variations in viewpoint, distance, contrast and brightness; see Fig. 3.1. *Matching* patches are from the same interest points with a maximum of 5 pixels difference in position, 0.25 octave in scale and $\pi/4$ radians in angle. *Non-matching* patches have the same variations but are from different interest points. For training we used 200K patch pairs from the Notre Dame dataset (100K matches and 100K non-matches) and for evaluation we used 100K patch pairs from the Yosemite dataset (50K matches and 50K non-matches), so that we could compare our results with those obtained by Trzcinski et al. [2013].

All patches were resized to 32 × 32 pixels in order to speed up processing, after which responses of even and odd simple cells and of complex cells were computed (see below). Since we applied up to 8 cell orientations and 7 cell scales, in principle it is possible to compute $3 \times 8 \times 7 \times 32^2$ features for each patch. The total number of 172,032 is excessive for any real-time application. Therefore, the basic idea is to construct an optimal projection matrix which also reduces the number of features, and to experiment with different feature selections and sampling or pooling strategies.

We extracted cell responses from all the image patches of the Notre Dame set at multiple scales and orientations and then applied the LDAHash [Strecha et al., 2012] method to construct a projection matrix that minimizes the in-class covariance (matching patch pairs) and maximizes the covariance across classes (the non-matching patch pairs). Then we applied this projection matrix to the cell responses of the training data to determine the optimal threshold vector to binarize each projected feature value. To evaluate the results we used the Yosemite dataset. In the following subsections will be described each step in detail.

3.3.1 Training and evaluation

The training set serves to determine an optimal projection (rotation) matrix together with an optimal threshold vector which must be applied to all image patches and their features of a real application, like stereo disparity or optical flow. Once such a matrix and vector have been determined by using the training data, exactly the same procedure must be applied to other data of the real application: the same cell selection, filter orientations, scales and pooling scheme. In other words, the Notre Dame training data are used *once* for constructing a matrix and corresponding vector, and these are then applied in robot vision, for example. The Yosemite test data are completely irrelevant.

Here, in this chapter, the goal is to experiment with different parameter selections: responses of simple and/or complex cells, the number of cell orientations and their scales, etc. This means that *for each parameter selection* the Notre Dame data are used to determine *one matrix* and *one vector*. Then, exactly the same parameters and matrix plus binarization vector are applied to the Yosemite test data for evaluation purposes: now matching thresholds are systematically varied in order to count true and false positive matches and to construct ROC curves. These curves only serve to gain insight into the performance of the different parameter selections. At the end, once we can draw firm conclusions about which parameter selections are best, one selection must be applied to the training data in order to determine a final projection matrix and its corresponding binarization vector.

3.3.2 V1 cortical cells

In cortical area V1 there are different types of cells: simple, complex and end-stopped . These cells are thought to play an important role in coding the visual input: they allow to extract multiscale lines and edges [Rodrigues and du Buf, 2009] and keypoints, which are line/edge vertices or junctions, but also blobs [Rodrigues and du Buf, 2006].

Responses of even and odd simple cells, which correspond to the real and imaginary part of a Gabor quadrature filter [Rodrigues and du Buf, 2009], are denoted by $R^E_{s,i}(x,y)$ and $R^O_{s,i}(x,y)$, i indicating the orientation (we used $N_\theta = 8$). The scale s is given by λ, the wavelength of the Gabor filters, in pixels (we used $\lambda = \{4, 6, 8, 12, 16, 24, 32\}$.) Responses of complex cells are modeled by the modulus

$$C_{s,i}(x,y) = [\{R^E_{s,i}(x,y)\}^2 + R^O_{s,i}(x,y)^2]^{1/2}. \tag{3.1}$$

For more details see Rodrigues and du Buf [2006]. What is important here is that receptive fields of even and odd simple cells can be seen as derivatives of Gaussians. Hence, like SIFT and derived methods, our method is based on spatial derivatives of the local image structure.

In summary, from each patch we can extract responses of even and odd simple cells at 8 orientations and 7 different scales. The responses at each scale are normalized to unit vector, and then we perform maximum pooling on circular receptive fields with a diameter of 4, 6 or 8 pixels and with strides (shifts) of 2 or 4 pixels.

3.3.3 Learning the projection matrix

For learning optimal data projections we apply the LDAHash method [Strecha et al., 2012]. As already mentioned, we use two sets of the Notre Dame training data, matching patch pairs and non-matching patch pairs. Let us call these P (positive) pairs in which patches must be matched and N (negative) in which patches must *not* be matched. The goal is to obtain a projection that (a) reduces the number of features, (b) maximizes the probability that P patches are matched, and (c) minimizes the probability that N patches are matched.

Suppose we have one feature (column) vector x of length n and we want to have a projection $y = \mathcal{P}x + t$ where y has dimension $m \ll n$, $\mathcal{P}$ is a matrix with dimensions $m \times n$ and t is a translation vector with dimension m. The translation vector serves binarization and will be dealt with later.

The training data contain matching (positive, P) and non-matching (negative, N) patch pairs. Let us call their feature vectors x and x' in case of positive, and x and x'' in case of negative pairs. The pairs of feature vectors of positive patches are "stacked" into two matrices with n features (rows) and N samples (columns): X and X', here with $N = 100,000$. Likewise, for the negative pairs we have two matrices X and X''.

Now, concerning the LDAHash method, the two matrix pairs are subtracted: $P = X - X'$ and $N = X - X''$. Then, the two covariance matrices of P and N are computed: Σ_P and Σ_N; and the ratio matrix after computing the inverse of Σ_N: $\Sigma_R = \Sigma_P \Sigma_N^{-1}$.

Because Σ_R is positive semidefinite, SVD can be applied for the eigendecomposition $\Sigma_R = USU^T$ where S holds the (positive) eigenvalues and U the corresponding set of orthogonal eigenvectors. The orthogonal $m \times n$ matrix $\mathcal{P}$ that minimizes the trace of $\mathcal{P}\Sigma_R\mathcal{P}^T$ is the projection of x onto the space spanned by the m eigenvectors with the smallest eigenvalues of Σ_R, and

$$\mathcal{P}\Sigma_N^{-1/2} = \tilde{S}_m^{-1/2}\tilde{U}^T\Sigma_N^{-1/2}, \tag{3.2}$$

where $\tilde{S}$ is the $m \times m$ matrix with the smallest eigenvalues and $\tilde{U}$ is the $n \times m$ matrix with the m eigenvectors. We note that the eigenvectors are divided by the square roots of their eigenvalues, and Strecha et al. [2012] keep the normalization by $\Sigma_N^{-1/2}$ because this "... makes the projected differences $\mathcal{P}(x - x')$ normal and white."

3.3.4 Learning the optimal binarization thresholds

To calculate the optimal binarization thresholds we evaluate each dimension independently. We start by finding the minimum, m_T, and maximum value, M_T, of all projected feature values in all Notre Dame patches. We created a vector of K values (we used $K = 3000$), equidistantly spaced between m_T and M_T. We then tested each of these values as a binarization threshold and evaluated which of them provided the largest sum of correct matches and correct non-matches for the current feature dimension being evaluated. After evaluating all the dimensions we have a threshold vector, t, containing as many elements as projected feature dimensions.

After having the projection matrix $\mathcal{P}$ and the optimal binarization threshold vector, t, we can finally compute binary descriptors by thresholding each of the feature dimensions. Each dimension is coded with a 1 if it is bigger than the corresponding threshold value, or a 0 if it is smaller.

3.4 Evaluation

As previously mentioned, for evaluation we used 100K patches from the Yosemite dataset (50K matches and 50K non-matches). This means that the projection matrix $\mathcal{P}$, as trained on the Notre Dame dataset, is applied to all patches of the Yosemite set. Then, for each selection of cell responses and pooling step and size, the projected data are binarized by varying the threshold values. For each threshold, true and false positive detection rates are determined in order to construct ROC curves. We ran multiple tests in the attempt of finding the best selection of input data: we experimented with different cell scales $\lambda = \{4, 6, 8, 12, 16, 24, 32\}$, different pooling steps $p = \{2, 4\}$, and different pooling sizes $s = \{4, 6, 8\}$. All patches were resized to 32×32 pixels before feature extraction to speed up evaluation. Pooling means response selection by searching for the maximum response in circular windows with a diameter of e.g. 4 pixel positions (the pooling size) and this window is shifted e.g. 2 pixels in x and in y (the pooling step). Each scale is evaluated independently in order to determine the scale's optimal pooling step and pooling size.

3.4.1 Cell selection: even simple, odd simple or complex?

The first test was about which cells or combination of cells contained the most discriminative information: even simple cells, odd simple cells or complex cells. Throughout all our tests we verified that, in most cases, complex cells provided the least discriminative information: since they are computed from combinations of simple cell responses, there is loss of discriminative information. On the other hand, even and odd simple cells performed better at most scales, and very similarly. At the finest scale ($\lambda = 4$) best results are obtained when using combinations of odd and even simple cell responses (95% error rate of 50.2%, as in [Trzcinski et al., 2013]: this means that at a true positive rate of 95%, the false positive rate is 50.2%). This result is followed by using only even or only odd cell responses. At all other scales, best results were obtained by using a combination of even, odd and complex cells. Worst results of each scale were obtained by using only complex cells.

By combining even and odd simple cell responses, the use of more features leads to the best performance. When combining all cell responses there is also an improvement, except for scale $\lambda = 4$. At coarser scales ($\lambda = \{8, 12, 16, 24\}$), the combination of even, odd and complex cell responses yields the best performance: since the number of features per cell type becomes smaller, combining multiple cell types leads to better results. However, in most cases the difference between combining only even and odd simple cells and also using complex cells responses is not very significant, which means that complex cells do not add much more discriminative information. The increase in performance by combining simple cell responses with complex cell responses gets smaller at coarser scales. At scale $\lambda = 32$ the result of using even, odd and complex cells is worse than if using only even and odd simple cells.

In most cases, best results were obtained by using even and odd simple cells. To illustrate the differences in performance, Fig. 3.2 shows results obtained with descriptors built from features at three different scales, $\lambda = \{4, 12, 24\}$, and with a pooling step of 2 pixels. It can be seen that for the three scales shown in Fig. 3.2 complex cells have a poorer performance than even or odd simple cells at the same scale. Similar differences in performance were obtained when testing other scales.

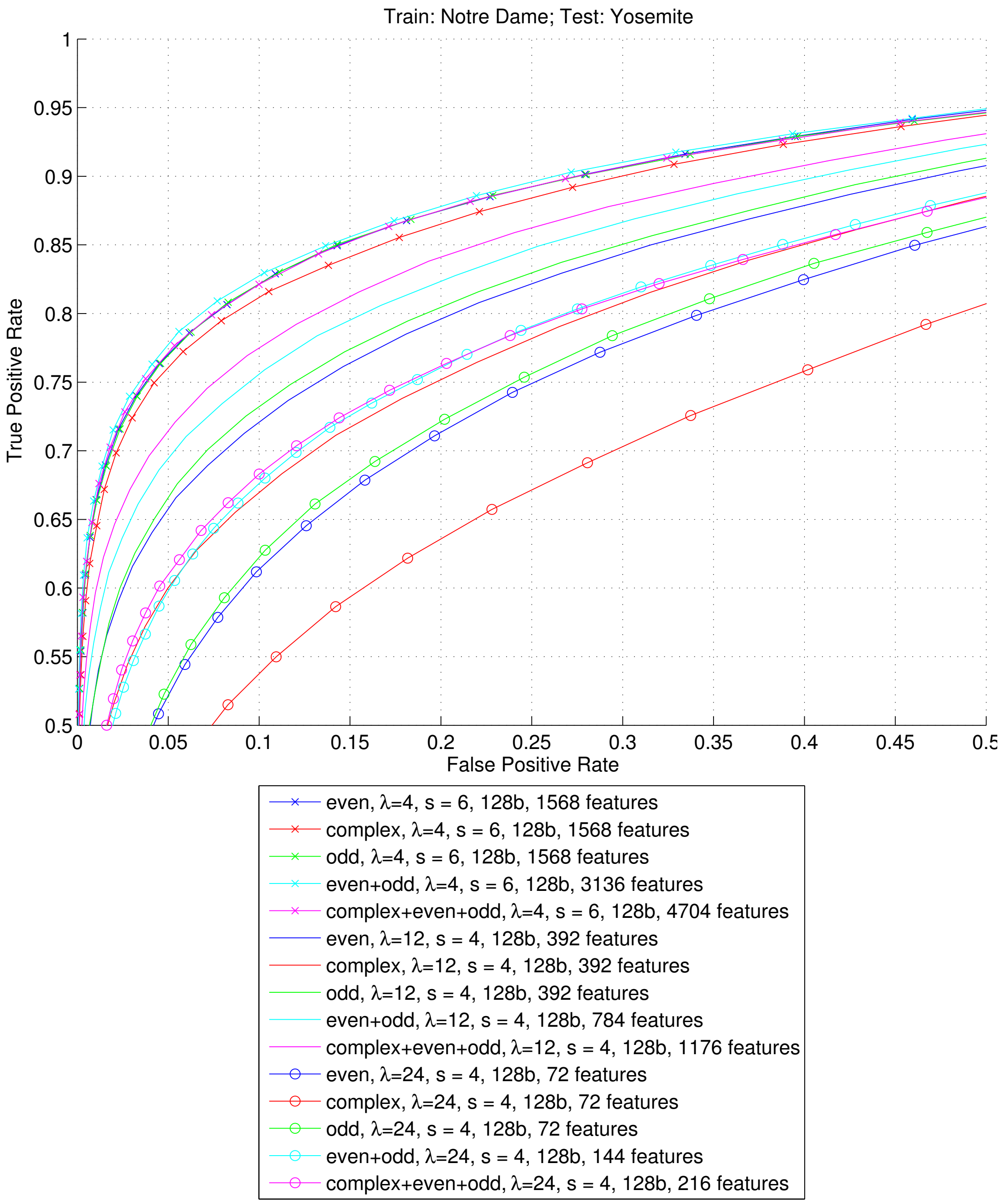


Figure 3.2: Comparison of results using different types of cells. In most cases even or odd simple cell responses result in better performance than when using responses of complex cells. Results obtained with even and odd simple cells are very similar. Combining different cell responses may or may not result in better performance, depending on the scale and the cells used. All the plots were generated using a pooling step $p = 2$ pixels. At scale $\lambda = 24$ only 64 bits were used for each individual type of response because the number of features was insufficient to generate a 128 bit descriptor.

3.4.2 Scale comparison

As previously mentioned, we tested seven different cell scales: $\lambda = \{4, 6, 8, 12, 16, 24, 32\}$. From our tests we can say that, in general, the more features are used, the better the performance will be. Since finer scales have more features, they performed much better than features from coarser scales. When comparing scales with similar numbers of features (e.g. $\lambda = 4$ and $\lambda = 6$), the finer scales perform slightly better. Figure 3.3 shows the best results obtained for each of the scales, and it also mentions the number of features used for each of the results.

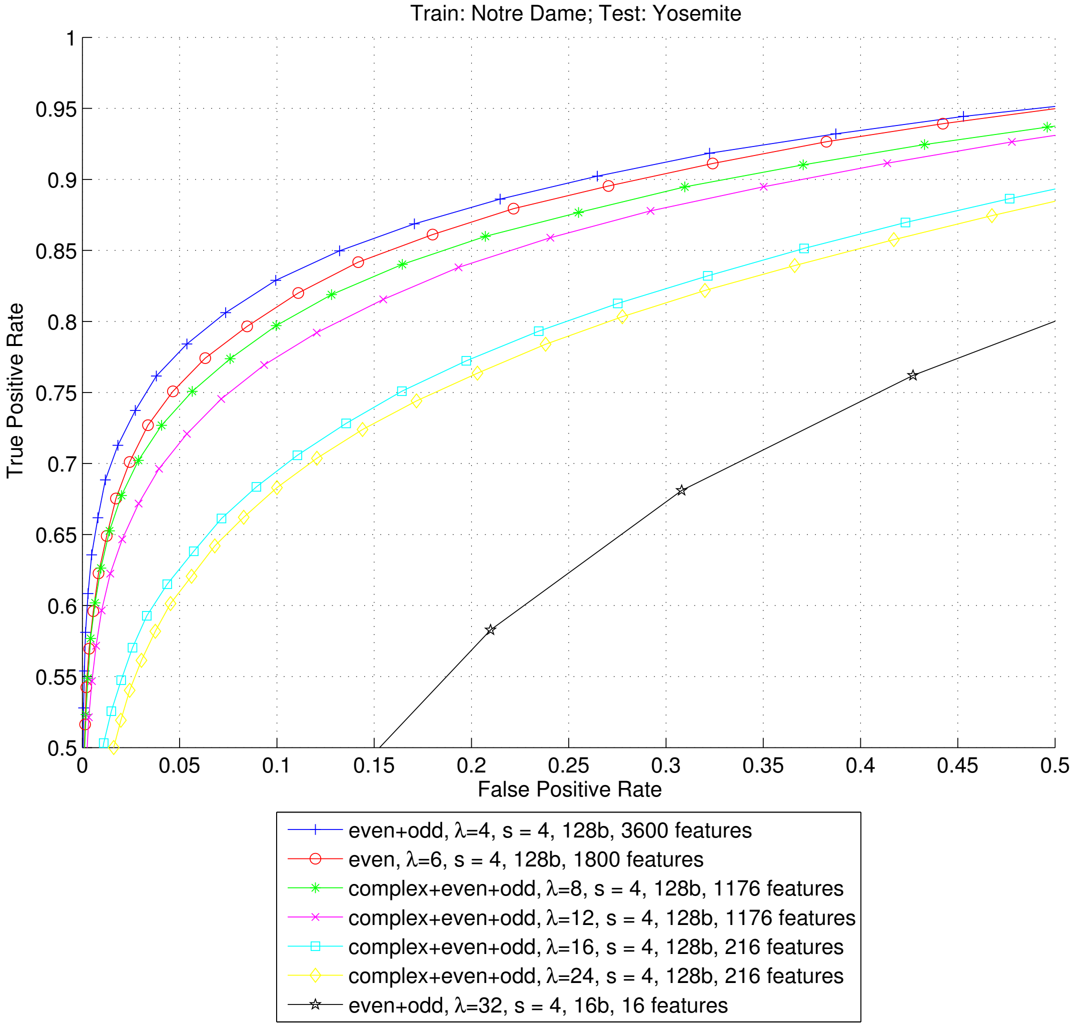


Figure 3.3: Comparison of the best results for each of the scales tested. The best result is obtained using $\lambda = 4$ (95% error rate of 49.1%), closely followed by $\lambda = 6$. As the scale gets coarser, the number of features is reduced and performance also decreases.

In addition, Fig. 3.3 shows that scale $\lambda = 4$ yields the best result when combining responses of even and odd simple cells, followed by scale $\lambda = 6$, using only even simple cell responses.

3.4.3 Pooling region size

The size of the pooling region also influences the performance of the descriptor. By using a larger pooling region we get less features, but the features may be more robust and have a higher invariance to small transformations. We tested all the scales with pooling regions with diameters of 4, 6 and 8 pixels to determine which pooling size would lead to a better performance. Some results are shown in Fig. 3.4.

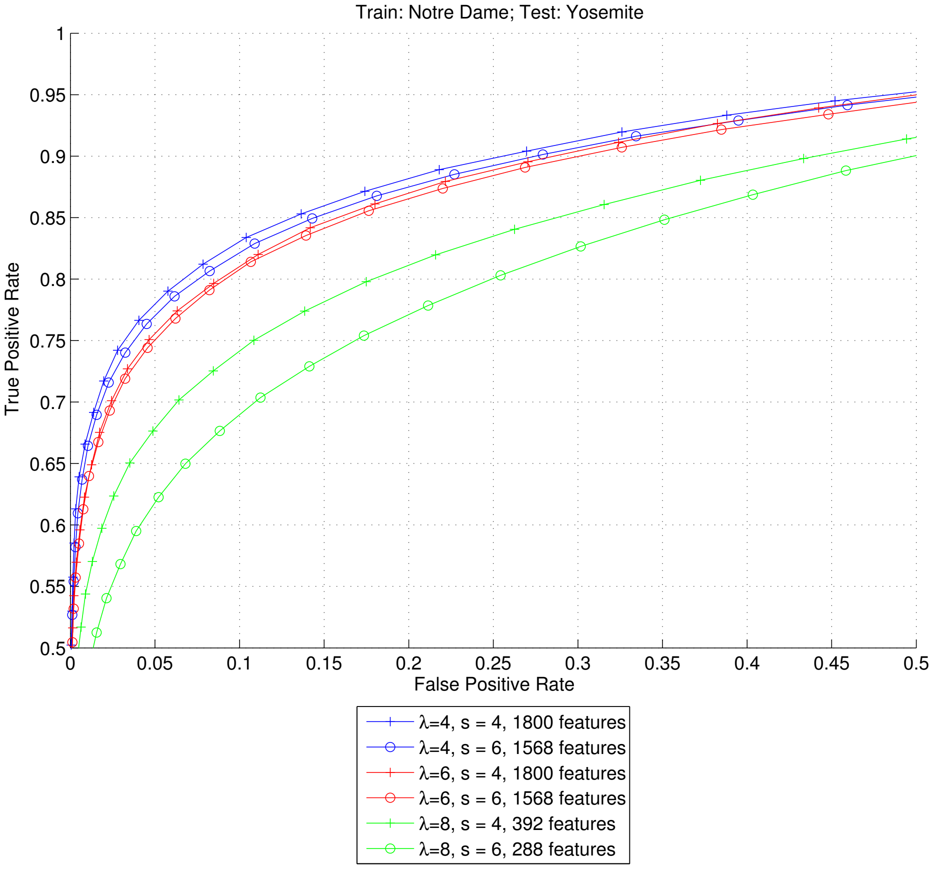


Figure 3.4: Evaluation of performance for pooling regions with different sizes. For all scales a pooling size of 4 pixels leads to the best results. Performance decreases with increasing pooling size. Results shown are for 128 bit descriptors built from even simple cell responses with a pooling step of 2 pixels. Results are similar for other cell responses.

From this figure it can be seen that for the three scales a pooling size of $s = 4$ provides the best results (95% error rate of 48.5%). As the pooling size increases, the number of features decreases, as does performance. Results for a pooling size of 8 pixels are not shown in the figure, but they were worse than those with sizes of 4 and 6 pixels.

At scale $\lambda = 8$ we can see that the performance clearly drops as the pooling size increases. The same happens at other coarse scales. A bigger pooling size drastically reduces the number of features at coarser scales.

3.4.4 Pooling step selection

As shown in the previous sections, although a larger number of features usually leads to better results, that is not always true. We therefore tested the influence of increasing the pooling step to evaluate if by cutting down the number of features, especially at finer scales, we could obtain a similar performance.

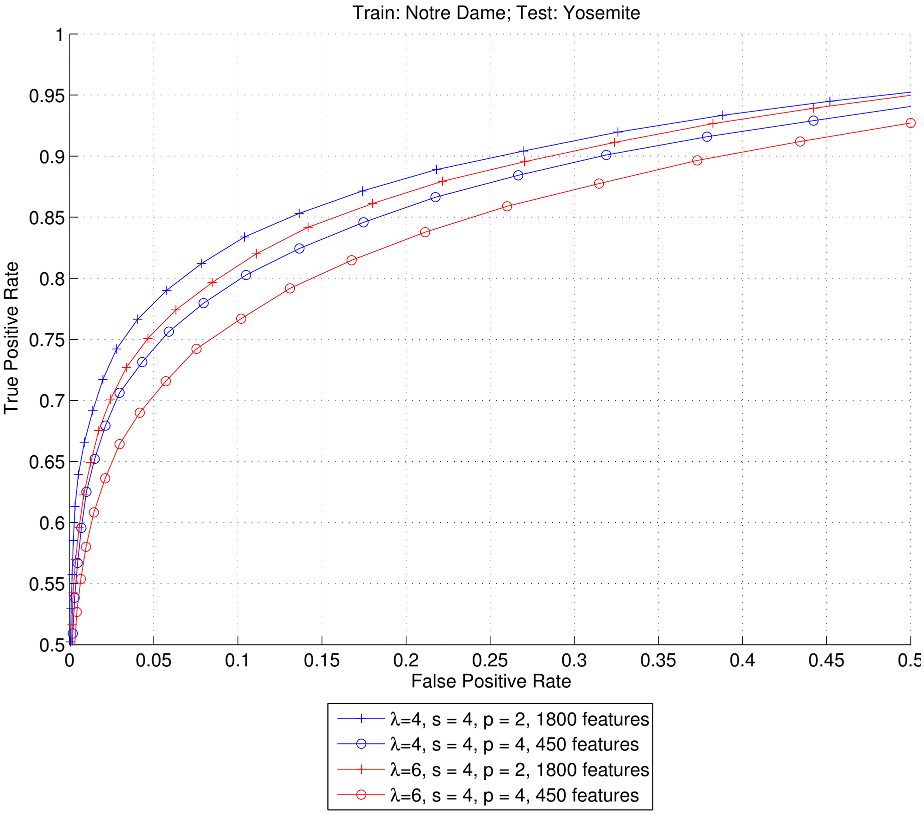


Figure 3.5: Evaluation of performance of 128-bit descriptors generated from even simple cell responses with pooling steps of 2 and 4 pixels and a pooling size of 4 pixels. There is a clear decrease in performance for $p = 4$.

This could be useful to improve the descriptor computation speed. The pooling step was increased to 4 pixels, which effectively reduces the number of features to 1/4 of the features extracted with a pooling step of 2 pixels. Some results are shown in Fig. 3.5. It can be seen that by reducing the number of features to 1/4 of the total number of features, the performance significantly decreases. This means that most of the features contain useful discriminative information and cannot be discarded without affecting performance.

3.4.5 Dimensionality of the descriptor

In this step we tested the influence of the dimensionality, or number of bits, $nbits$, on the performance of the descriptor. We evaluated the best results from the two scales with best results ($\lambda = 4$ and $\lambda = 6$ with a pooling size of 4 pixels and a pooling step of 2 pixels) for $nbits = \{64, 128, 256, 512\}$.

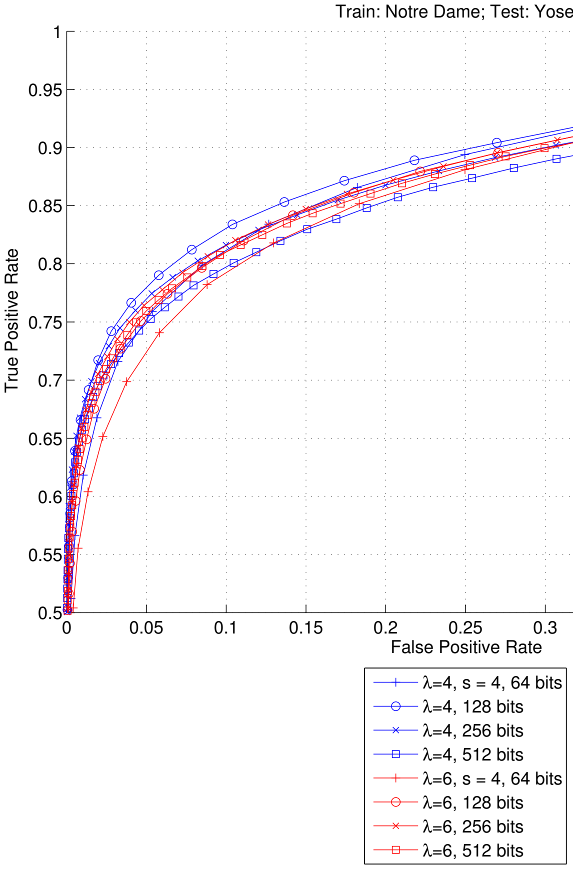


Figure 3.6: Comparison of results using 4 different numbers of bits for scales $\lambda = 4$ (blue) and $\lambda = 6$ (red). Best results at both scales are obtained using 128 bits, but for scale $\lambda = 6$ results with 256 bits are very similar.

Figure 3.6 shows that the performance peaks at 128 bits and it decreases as the number of bits increases. Using 256 bits results in a drop in performance and a bigger drop occurs when using 512 bits. Results using 64 bits are worse than those with 256 bits for small false positive (FP) rates, but they are better at larger FP rates. These results are consistent with the dimensionality evaluation done for LDAHash; see Fig. 3.6 from Trzcinski et al. [2013].

3.4.6 Combining different scales

Up to this point we evaluated the influence of different parameters on descriptor performance for each scale individually. However, in cortical area V1 cell responses at multiple scales are combined to create higher level representations.

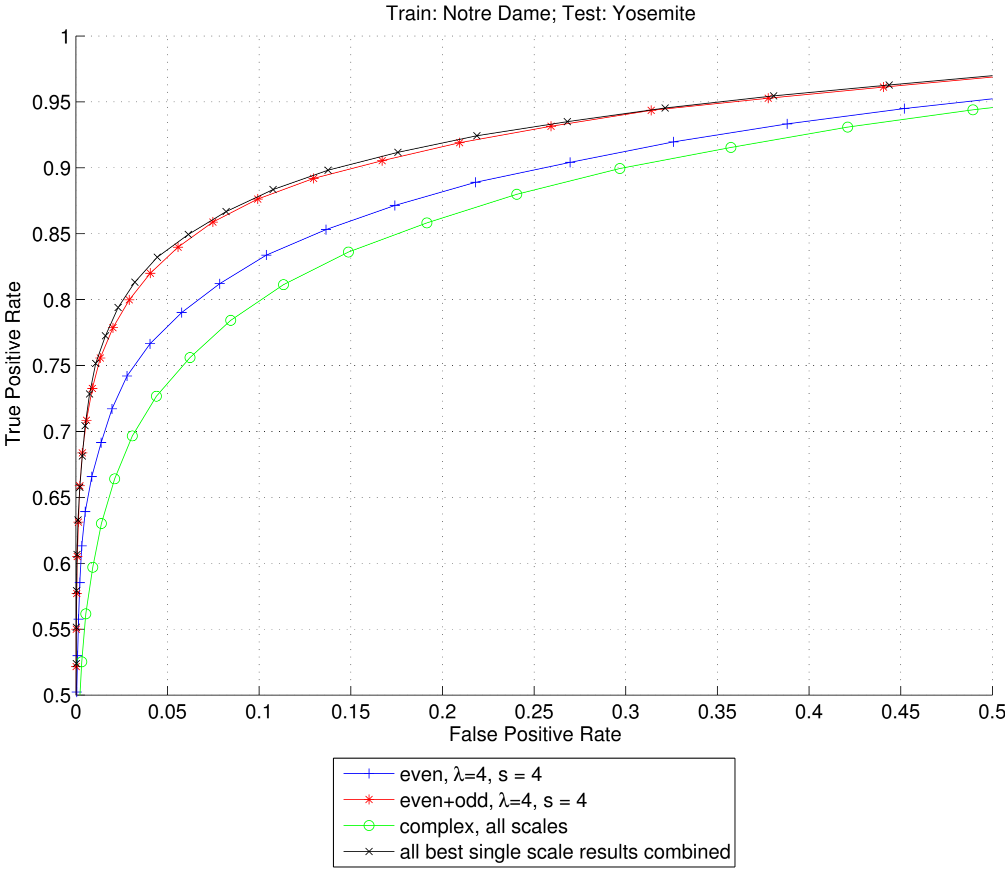


Figure 3.7: Comparison between the best 128-bit single-scale result previously obtained ($\lambda = 4$ with a pooling size of 4 pixels and a pooling step of 2 pixels) and three 128-bit descriptors built from different combinations of cell responses. Only the combination of complex cells at all scales performed worse than the best single scale result.

Taking this into account, it can perhaps be expected that by combining cell responses at several scales a better result may be achieved. Therefore we experimented with a few combinations of responses at all scales. We also tested what happens when we combine the cells that provided the best result at each individual scale. Figure 3.7 illustrates the results obtained.

From Fig. 3.7 it can be seen that two of the three different cell combinations perform much better than the best single-scale, even simple cell 128-bit descriptor: (1) the combination of even and odd simple cells at scale $\lambda = 4$; and (2) the combination of all best single scale results. It can also be seen that the descriptor built only from even simple cells provides a better result than that built only from complex cells at all scales. The best result is obtained with the combination of all the best single-scale results (95% error rate of 35.1%) but it is very similar to the one obtained by combining even and odd simple cells at scale $\lambda = 4$ (95% error rate of 35.8%). From the plot it seems once more that responses of complex cells are less discriminative than those of even or odd simple cells.

3.4.7 Testing HMAX features

Since the goal of the work described in this chapter was to develop a biologically-plausible local descriptor, we also included tests with biologically inspired HMAX features [Serre et al., 2007]. We employed the HMIN implementation [Mutch, 2011], which extracts 8150 features from each patch, and we also experimented with combining HMAX features with our even simple cell responses at scale $\lambda = 4$. The results can be seen in Fig. 3.8. Using only HMAX features results in a poor performance. Using HMAX features together with even simple cell responses leads to a result which is only slightly better. Relative to the best result that we had up to this point (all even and odd simple cells at all scales with a pooling step of 2 pixels), we can see that performance is much worse. Even using only even simple cell responses at scale $\lambda = 4$ yields a better result. Apart from the lower performance, HMAX features also have the disadvantage of taking much more time to compute than our simple cell responses. This makes them unsuitable for any real-time applications with the

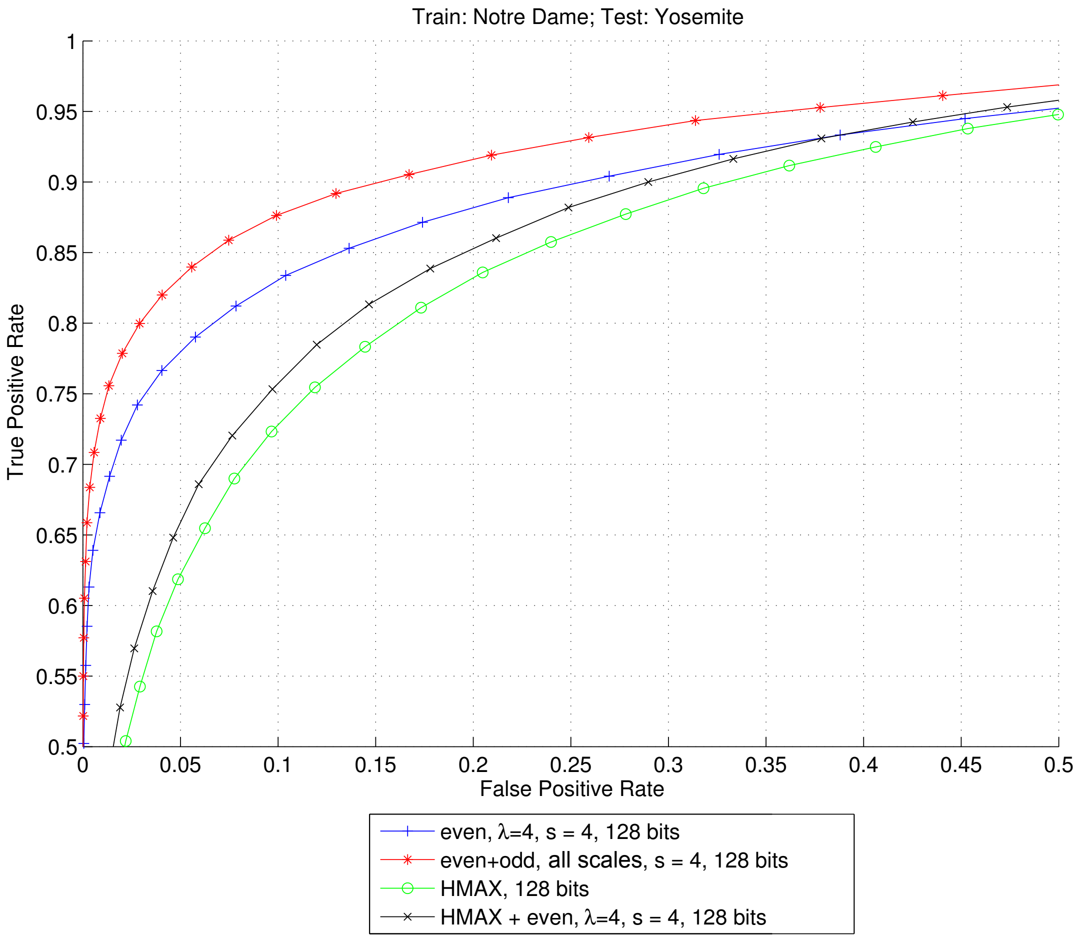


Figure 3.8: HMAX features are much weaker than our even simple cell responses at scale $\lambda = 4$, even considering that we could use 8150 HMAX features and only 1800 even simple cell features. When combining them with our even simple cell responses, the performance becomes slightly better but it is still worse than the result when using only even simple cell responses.

computational power currently available.

3.4.8 Comparison with other biologically inspired local descriptors

In this section we compare the descriptor that we developed and the biologically inspired descriptors BRISK (512 bits) and BRIEF (256 bits). Figure 3.9 shows that the descriptor that we developed significantly outperforms both of them. The 64-bit version of our descriptor, either using only scale $\lambda = 4$ or using the combination of even and odd simple cells at all tested scales, is already good enough to clearly outperform them. They have respectively false positive rates of 48.5% (single-scale) and 35.8% (multi-scale) for a 95% error rate, versus 55.0% of BRIEF and 73.2% of BRISK, which use 4 and 8 times more bits.

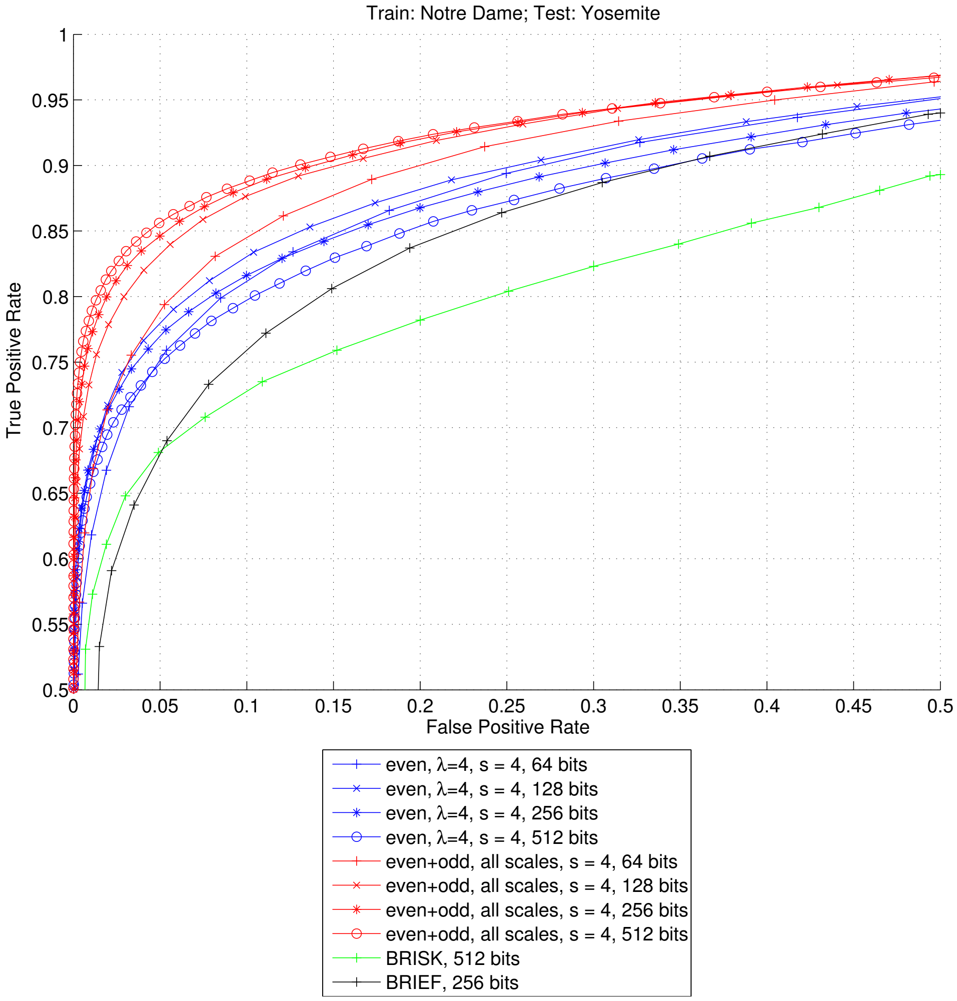


Figure 3.9: Comparison of our descriptor performances with other biologically inspired descriptors BRIEF and BRISK. Our descriptor, either using only even simple cells at scale $\lambda = 4$ or using a combination of even and odd simple cells at multiple scales, outperforms both BRIEF and BRISK, even when using four or eight times less bits.

If we consider a direct comparison in terms of bits, the 256 bit version of our descriptor at scale $\lambda = 4$ has a 95% error rate of 54.3%, which is only 0.7% better than BRIEF but it is much better at low false positive rates. Comparing the present descriptor built from multiple scales, the 95% error rate difference to BRIEF is bigger: 19.8%. Making the same comparison with BRISK, we have 95% error rate differences of 37.7% and 14.8%. The present descriptor clearly outperforms the other biologically inspired descriptors BRIEF and BRISK, even when using four or eight times less bits.

3.4.9 Comparison with the SIFT-based LDAHash local descriptor

Figure 3.10 shows that both our single-scale and multi-scale 128-bit descriptors outperform the SIFT-based 128-bit descriptor LDAHash, a state of the art, non-biological descriptor. The present descriptors have 95% error rates of 48.5% (single-scale) and 35.8% (multi-scale), which are both better than LDAHash (52.9%). When considering low false positive rates, our descriptors are still better than LDAHash, and even when using a 64-bit version of the multi-scale descriptor we still get better results (a 95% error rate of 40%).

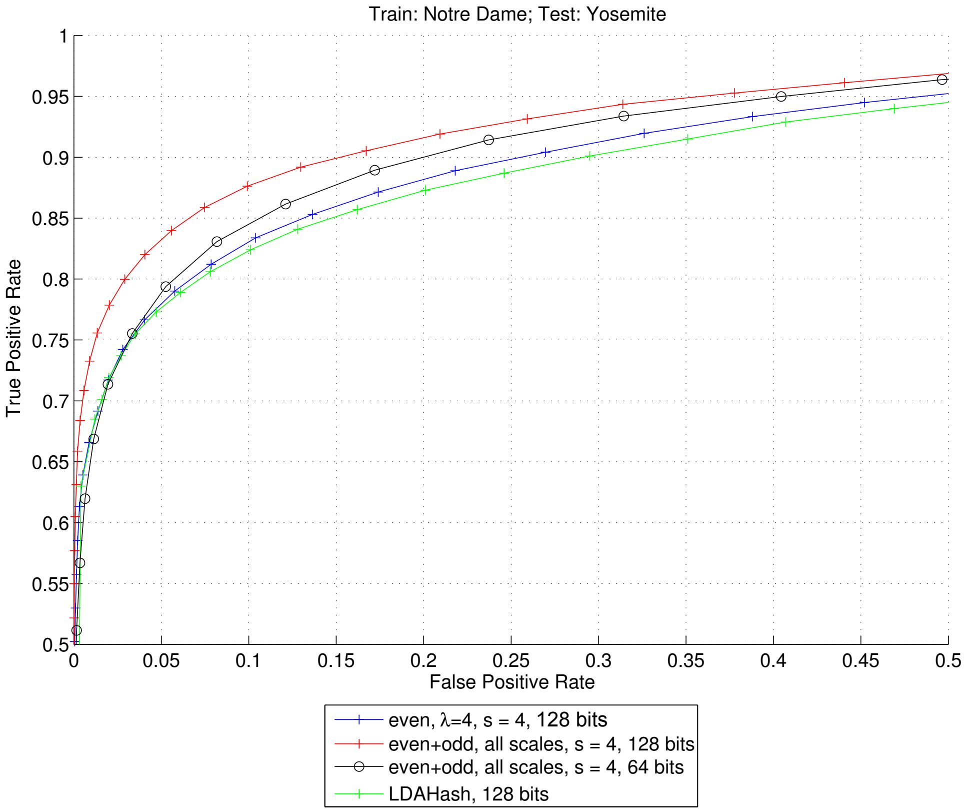


Figure 3.10: Comparison of our descriptors with the non-biologically inspired descriptor LDAHash.

3.4.10 Computation time

After showing that the new descriptor outperforms the other biologically-inspired descriptors BRIEF and BRISK and also the non-biological SIFT-based descriptor LDAHash comes the question: can it be used as alternative to other descriptors currently used in real-time applications? To evaluate the computation time in a proper way the processing time was split into two steps: feature extraction and keypoint computation.

We measured the average time for *feature extraction* over 1000 patches of 32×32 pixels, thus implicitly assuming one keypoint at the center of each patch. For the single scale features ($\lambda = 4$) features the average time was 2.15 ms and for the multi-scale features ($\lambda = \{4, 6, 8, 12, 16, 24, 32\}$) the average time was 2.35 ms. Times were measured on a computer with a quad-core processor running at 2.4 GHz and with OpenMP for parallel processing. Regarding the *construction* of the descriptor, it is a much faster process and takes only an average time of 0.45 ms for only scale $\lambda = 4$ and 0.52 ms for the combination of several scales. However, in this step no parallel processing was applied and there is still room for optimization. Since feature extraction and descriptor computation take a few milliseconds, processing times are good enough for some real-time applications. However, the processing time can be greatly reduced using a more modern processor. Another option is to use the GPU implementation of feature extraction and keypoint detection Terzic et al. [2015]. This can reduce computation time very significantly, while leaving the CPU processor free for other processes.

3.5 Applications

In this section some applications will be shown of the descriptor that we developed. As referred to in previous sections, descriptors can be used for early vision processes, such as optical flow and stereo vision, but also for object recognition.

3.5.1 Optical flow

Optical flow is the motion pattern caused by moving objects in a visual scene. It can be described by the motion or displacement vectors of entire objects or parts of them between consecutive time frames. From a biological point-of-view, there are strong arguments indicating that neurons in a specialized region of the cerebral cortex play a very important role in flow analysis [Wurtz, 1998]. According to William and Charles [2008], neuronal responses to flow are shaped by visual strategies for steering, and according to Warren and Rushton [2009], the flow processing has a very important role in the detection and estimation of scene-relative object movement during egomotion.

Using our previous developed keypoint detector [Terzic et al., 2015], we extracted keypoints in sucessive camera frames. Then we applied our new descriptor to match keypoints from one frame to the other, thus obtaining each keypoint's displacement vector. Since keypoint displacement is expected to be relatively small from one frame to the next one, depending on frame rate, we can limit keypoint matching to a specific range, and not waste computation time in matching very distant keypoints. Since our descriptors are binary and we only have to match each descriptor against a few other descriptors, optical flow can be determined in a very simple, fast and effective way.

Figure 3.11 shows two examples of optical flow estimated by using our cortical keypoint detector and descriptor. Keypoints were extracted at scale $\lambda = 4$ and annotated by the 128-bit single-scale descriptor, also using only $\lambda = 4$. The matching was done between each keypoint in the first frame and all keypoints within a 30 pixel distance in the second frame. The top video in Fig. 3.11 has a resolution of 584×388 pixels and an average of 1932 keypoints per frame. The bottom video has a size of 640×480 pixels with an average of 1594 keypoints.

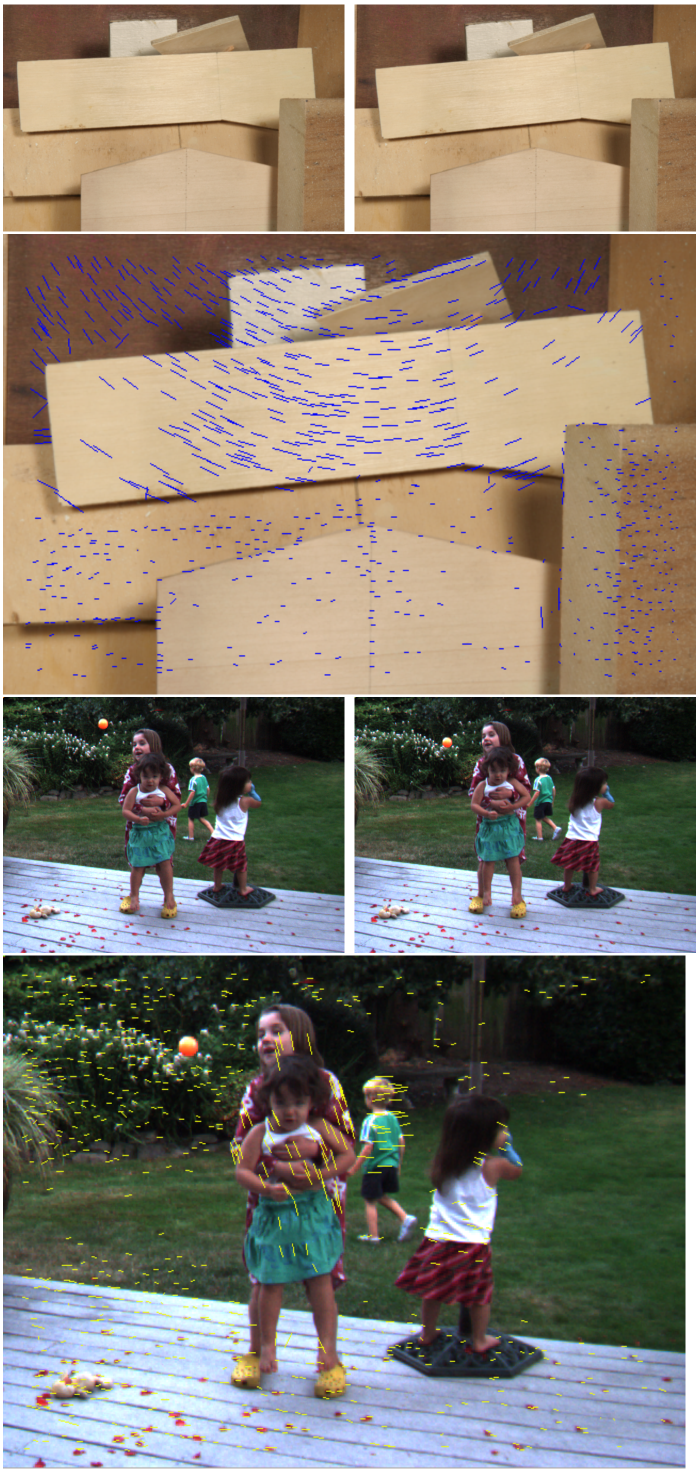


Figure 3.11: Example of optical flow determination using our keypoints and keypoint descriptors. Rows 1 and 3 show two sequential frames from two different sets. Rows 2 and 4 show the optical flow results obtained from matching keypoints between the two frames. The images belong to the Middlebury Optical Flow dataset [Scharstein, 2009a].

3.5.2 Stereo vision

Stereo vision is another important process in our brain. It allows us to estimate distances to obstacles and objects and is especially important for navigation. In our primary visual cortex we have binocular neurons which take input from both left and right eyes and integrate the signals together to create a perception of depth. These cells are selective, usually tuned to specific disparity ranges. We assume that each binocular neuron takes as input higher level representations from points on the same epipolar line from left and right images, compares them, and fires if the representations are similar. These higher level representations could be our keypoints and their descriptors.

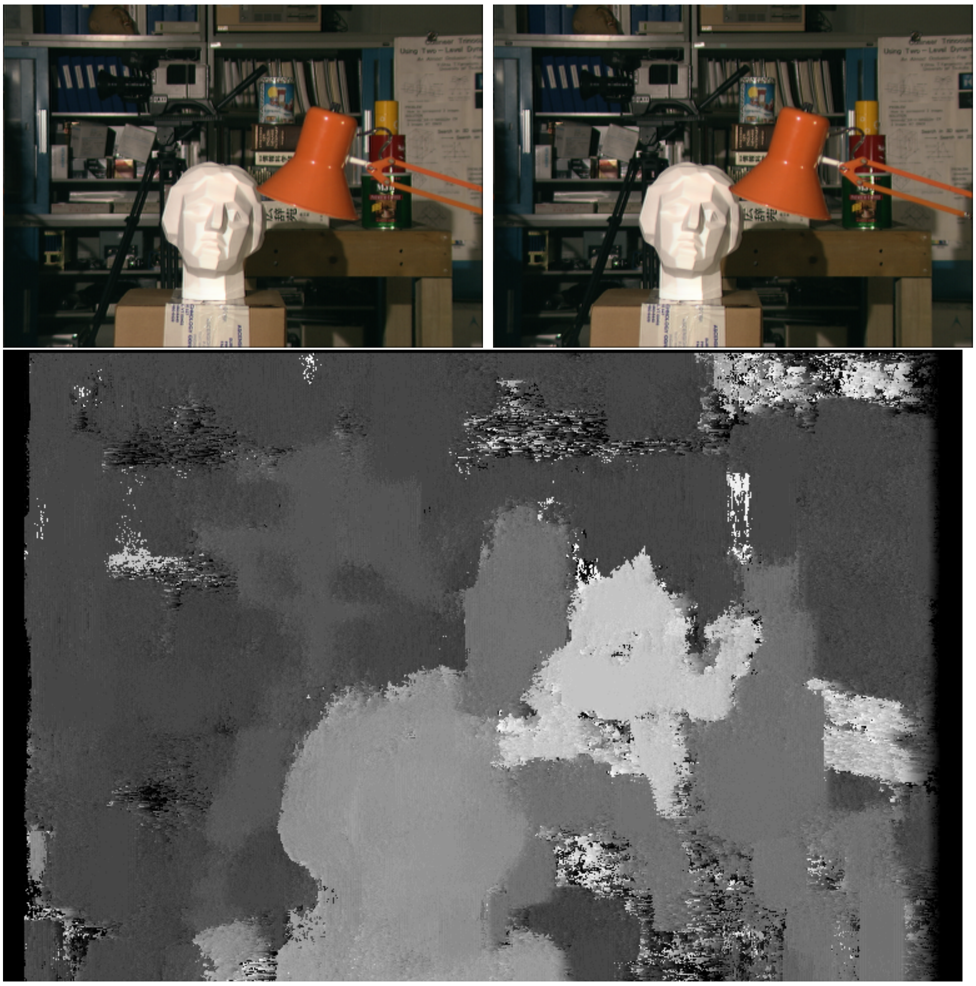


Figure 3.12: Example of stereo matching using our descriptor to match pixels from the left image to the right image. The image is part of the Middlebury dataset for stereo vision [Scharstein, 2009b].

Looking at the problem from a computational point of view, we can compute disparity maps through the following process: instead of applying keypoint detection we can code each pixel from the left and right images by using our keypoint descriptor. Then we match each coded pixel of the left image to the next K pixels on the same epipolar line of the right image. The horizontal displacement between the two best matching pixels then corresponds to the disparity value of that pixel.

Figure 3.12 shows an example of a disparity map, with $K = 30$. Since the goal of this section is only to show the possibility of applying the descriptors for stereo matching, no extra pre- or post-processing was applied. If the goal was to achieve a state of the art stereo algorithm, further processing is necessary. However, rough stereo maps are sufficient for robot navigation and obstacle avoidance.

3.5.3 Object recognition

Object recognition can also be achieved by using our descriptor. We applied keypoint extraction and description to a library of labeled object templates so that each object is represented by groups of keypoint descriptors. Then we can take a query image, apply the same keypoint detection, build the keypoint descriptors, and use the Hamming distance to match them against the ones of the labeled templates. Once we have at least 30% of matching keypoints between the query image and a template, we can assume that a match has been detected. Figure 3.13 shows an example of matching a cup. We are able to perform object recognition in real time and by using several scales we can match objects that have different sizes.

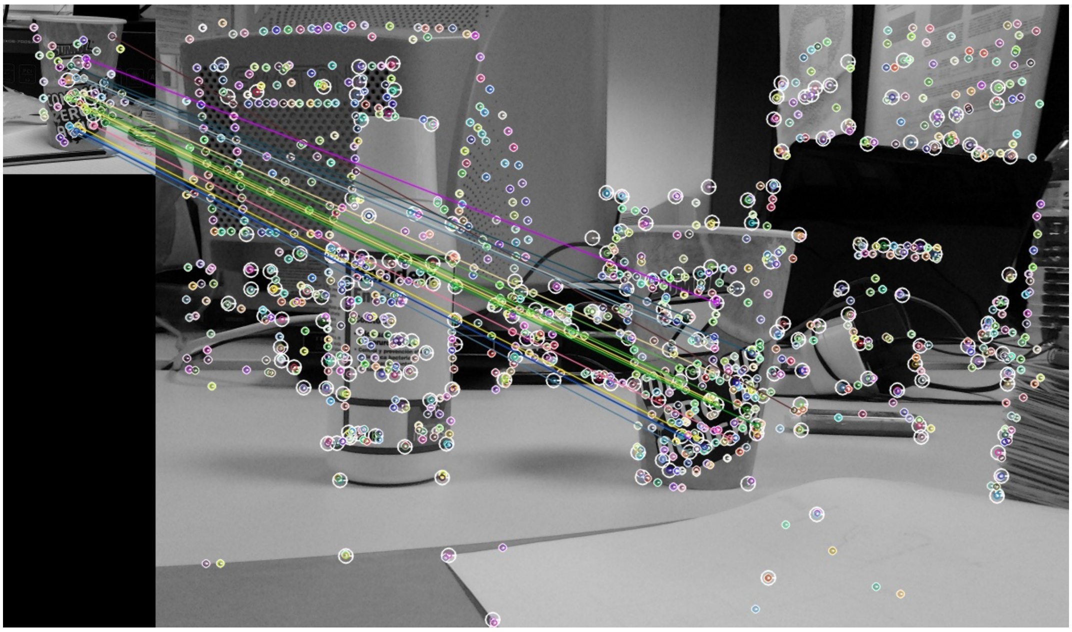


Figure 3.13: Example of object recognition using our keypoint descriptors, extracted at scales $\lambda = 8$ and $\lambda = 16$. Since the cup in the left image is at a smaller scale than the one in the right image, the descriptors from scale $\lambda = 8$ in the left image correctly match those from scale $\lambda = 16$ in the right image.

3.6 Conclusions

In this chapter we presented a process to learn and optimize biologically inspired binary keypoint descriptors. We showed that they clearly outperform other biologically inspired binary descriptors and also the non-biological SIFT-based LDAHash descriptor. Both the single-scale and the multi-scale descriptors that combine even and odd simple cell responses are fast enough for real-time applications, and we have shown that they can be used to model some early vision processes such as optical flow, stereo disparity and also object recognition. The best descriptor we were able to build combines even and odd simple cell responses at multiple scales. According to the experiments it was verified that by using more scales and more even and odd simple cell responses, the result is usually better. Complex cells, on the other hand, do not add much useful information.

As further work we intend to create a GPU implementation of the descriptor construction process which can be integrated with the keypoint detection implementation. Since keypoint detection is also based on responses of simple and complex cells, the responses of these cells are readily available for the descriptor process. The result will be a complete keypoint detection and description framework that is fast enough for several real-time applications, such as real-time vision for biologically inspired robotics. Another step for the future is to repeat the work described in this chapter for building descriptors with a smaller number of features and with a smaller number of bits. Although these descriptors will perform worse than the ones tested in this chapter, they may still be useful to speed up processes such as optical flow and stereo vision, which, as shown, can be implemented by matching descriptors against a small set of other descriptors. Our brain performs these processes very efficiently.

Chapter 4

Biologically Inspired Active Vision for Robot Head

Abstract: Active vision has been an interesting research topic for about four decades. The possibility of having a camera system that can acquire information, process it, and move the camera towards regions that can provide more information, has a wide range of applications. Robotics, either fixed or mobile, stands as one of the areas where active vision can be particularly useful. In order to create such systems it is necessary to have a visual perception system that is able to analyze images captured by the robot's cameras and extract interesting regions to where the cameras should point at next. Depending on the application, multiple different features can be used to evaluate the level of importance of certain image regions. In this chapter we propose a real-time biologically inspired active vision system to control a robot head with two degrees of freedom. Acquired images are processed for color, disparity and motion information which are used to create a saliency map that will generate motor commands. Motor control is performed using 2D attractor dynamics. In order to achieve real-time performance, we developed a new biologically-inspired binary descriptor, a new stereo-vision model that outperforms any other biologically-inspired model. In addition, a new object recognition model was also developed.

4.1 Introduction

The first general framework for active vision was proposed by Aloimonos and Weiss [1988] with the objective of improving the perceptual quality of tracking results. The term “active vision” implies the existence of a continuous perception-action cycle where the acquisition of information directly affects the movements to be made, which will then also affect the information that will be captured next. In robotics, active vision can be particularly useful since it helps in focusing computational resources on the important elements of a scene [Tatler et al., 2011].

In nature, active vision is very common and essential for survival, since it allows animals to track their preys or predators. We, humans, have a very sophisticated active vision system that processes the information acquired by our eyes, extracts several cues like color, movement, disparity and texture, and combines them to create a saliency map that determines where to look next. Although the (computational) interest in active vision has emerged almost four decades ago, most computer vision systems nowadays still use a static camera. Some of these systems have the camera mounted on a vehicle or on a moving arm, but usually there is little or no control of the camera movements.

On the other hand, in biology, both the eye and the head are extremely active: the eyes perform rapid and frequent saccades and tracking movements, and the head pans and tilts to change the viewpoint so that the eyes can acquire more information relative to the current position. In our retinae the distribution of receptor cells is extremely non-uniform, having the greatest concentration at the fovea. Only the center region is seen sharp. Saccades are then used to complete the perception of the scene by joining multiple sharp regions. Also, since our eyes have a limited field of view (FoV), head movements are used to improve the amount of information that we can acquire. In these systems the acquisition of information is an active process, not passive. The foveal structure of the human eye is related to the selective aspect of human active vision [Findlay and Gilchrist, 2003].

Naturally, this selectivity is also related to overt and covert visual attention.

Visual attention is a biological process that processes the information acquired by the eyes to select which regions must be analyzed in more detail. Since the brain cannot process all sensory stimuli in the physical world it must focus resources on the stimuli that can maximize the information gain. Saliency models for visual attention can be built from several cues (see Section 4.2). One cue, which is surely important for systems that have some sort of real world interaction, is disparity. A mobile robot can benefit from getting it's attention drawn to nearby obstacles in order to avoid them and a robot arm can also benefit from being able to select objects on top of a table to grasp them.

Proof of those benefits is the growing trend of using depth sensors like Microsoft Kinect in robotics applications [Oliver et al., 2012; Biswas and Veloso, 2012]. Traditionally, depth information is calculated using stereo vision, by taking images from two horizontally displaced cameras and matching the pixels from the left image to the right one or vice-versa. Many stereo vision models have been proposed with outstanding results, see e.g. Zhang et al. [2015] (see also Section 4.2).

In direct relation to disparity, the process of visual saliency is useful for selecting interesting regions in a scene. These interesting regions can then be processed for object or landmark recognition. One of the used approaches to detect objects or landmarks is by using keypoints and keypoint descriptors [Terzic et al., 2015]. Keypoints indicate the location of specific events in an image but they do not contain any information about the regions where they are located, hence they cannot be matched to each other. To perform keypoint matching across images it is necessary to take the local image information around them and to code this, building a local image descriptor for every keypoint. These descriptors must be robust to image variations, like illumination changes, rotation, etc. Descriptors characterize image regions by means of a numerical representation which can be floating point numbers or a binary strings (see e.g. [Rublee et al., 2011; Leutenegger et al., 2011]).

An active vision system implies a continuous perception-action cycle where the pan and tilt motor are jointly actuated according to the acquired information. Most robotics systems commonly use PID controllers to drive the robot motors, including the pan and tilt camera motors (if they exist) [Wu et al., 2010; Zhang et al., 2010]. An alternative to PID controllers is to use Attractor Dynamics controllers, which take inspiration from biology and have been successfully used in several robotics applications such as sensor data fusion, autonomous vehicle control and obstacle avoidance [Bicho et al., 1998] and formation control [Monteiro et al., 2004]. Dynamic Neural Fields, an extension of attractor dynamics, have also been used for learning sensorimotor transformations [Sandamirskaya and Conradt, 2013].

All models presented in this chapter are biologically plausible processes, mostly based on features extracted in V1 cortical region. The main contributions are: (i) a novel biologically inspired stereo model that is able to run in real-time for robotics applications; (ii) the integration of this stereo model with color and motion for saliency extraction; (iii) the application of a state-of-the-art biologically inspired binary descriptor for object recognition in salient regions (the same as the one presented in Chapter 3); (iv) the use of attractor dynamics for pan control; and (v) the test of the whole system in real-time using a robot head with real objects and object recognition in a real environment.

In the next section we will be briefly describe some of the work that has been done in active vision and related fields. In Section 4.3 the biologically inspired active vision system, including the system architecture, is described, and in Section 4.4 we will present the bio-inspired multiscale binocular cell stereo model. Section 4.5 presents the saliency model, and in Section 4.6 the object recognition model. Section 4.7 deals with attractor dynamics for pan control, followed in Section 4.8 by all the evaluations and tests. In the last section we will draw some conclusions and refer to some future work.

4.2 Related work

The use of biologically inspired models in vision has been getting more attention and some interesting implementations have already been realized. Serre et al. [2005] proposed a set of features for object recognition inspired by the visual cortex that outperformed some more complex computer vision systems. Holzbach and Cheng [2014a] also proposed a concurrent real-time biologically-inspired visual object recognition system. Their approach was based on a hierarchical model of the visual cortex for feature extraction and rapid scene categorization. To achieve real-time performance, they applied optimization methods from signal detection theory, signal processing and linear algebra. They made improvements on the computation speed of HMAX features, which are composed of several layers which model the simple and complex cells from the human visual cortex. They used maximum response occurrences in the image to perform object localization.

Directly associated with object recognition is visual attention, which can be categorized into two distinct functions: bottom-up and top-down attention. Bottom-up attention refers to attentional guidance performed by external factors that are salient due to their inherent properties (color, texture, orientation, disparity, motion, etc.) relative to the background of the image. On the other hand, top-down attention refers to attention guidance based on prior knowledge and task objectives [Katsuki and Constantinidis, 2014].

Visual attention can be materialized by saliency maps. Several computational saliency models have been developed so far. One of the first models, which is still a reference today, was proposed by Itti et al. [1998], and uses cues like color, intensity and orientation. These cues are processed by a bank of center-surround filters to create several saliency maps that finally are combined to generate a single, final saliency map. Other approaches use only color information [Martins et al., 2008] or keypoint density, since keypoints code local image complexity [Rodrigues and du Buf, 2006]. More recently, a texture based model was proposed by Terzić et al. [2015] and results have shown it to be competitive with the original model by Itti et al. [1998]. Moreover, when combined with color it outperformed the original model. This model also has biological inspiration since it uses responses of complex cells from V1 cortical area. Several saliency-based object recognition approaches were proposed by different authors [Walther et al., 2002; Holzbach and Cheng, 2014b; Frintrop et al., 2014].

It is important to stress at this point that, just like our brain, current computer processors also have their limitations. It is known that our brain is much more powerful and efficient than current processors and yet it still cannot process all the available stimuli, therefore using saliency for a more efficient usage of resources. Hence, using saliency for robotics applications could also bring some benefits, reducing processing time while increasing perception efficiency. Associated with the concept of visual saliency, and also a tool to create more efficient systems, is the concept of Inhibition of Return (IoR): the relative suppression of FoA stimuli that have been recently/before processed [Posner et al., 1995]. The goal of this stimulus suppression is to drive attention to novelty [Posner and Cohen, 1984], serving as a search or exploration facilitator [Itti and Koch, 2001].

Nevertheless, other solutions exist. Gould et al. [2007] developed an interesting approach for peripheral-foveal vision for real-time object recognition and tracking in video. Their method uses a learned, attentive low-resolution interest map to direct a high-resolution fovea towards the interest points. Objects recognized in the fovea can then be tracked using peripheral vision. In their tests they used a low-resolution wide angle camera for peripheral vision and a high-resolution pan-tilt-zoom camera for foveal vision. With this setup they were able to achieve real-time performance, since object recognition only had to be performed on a small foveal area.

Siagian and Itti [2007] developed a robot localization system using biologically-inspired robotics vision. The system models two extensively studied human visual capabilities: extracting the gist of a scene to produce a coarse localization hypothesis and refining it by using saliency to locate landmark regions in the scene. Gist was computed as a holistic statistical signature of the image, allowing for abstract scene classification and spatial layout. Saliency was computed as a measure of interest at all image locations in order to direct the landmark identification process towards the most likely candidate locations. Both gist and salient landmark features are then processed using a Monte-Carlo localization algorithm. Their system was tested in three different outdoor environments and the robot was able to localize itself in most situations.

One of the most advanced works on biologically inspired active vision was the model proposed by Leitner et al. [2013] for learning visual object detection and localization using the icVision framework. They used an iCub robot to detect and recognize objects in a table-top scenario. They used several filters, in RGB and HSV color space, to detect objects and the FAST features [Rosten et al., 2010] to perform pre-segmentation. They also used genetic programming to automatically learn filters to detect objects and experimented using the saliency model developed by Itti et al. [1998] coupled with stereo vision provided by the iCub stereo camera for scene exploration and object segmentation.

Related to stereo, MeshStereo proposed by Zhang et al. [2015] is one of the top-performing models and, unlike other stereo models, it does not output a simple disparity map but is able to output a 3D triangular mesh which can be directly used for view interpolation. NTDE [Kim and Kim, 2016] is another of the top-performing stereo algorithms. It applies adaptive smoothness constraints using texture and edge information. The method penalizes depth discontinuities in non-textured regions and it complements the primary CNN-based matching cost with a color-based cost. Then, by combining two edge maps from the input image and a pre-estimated disparity map, denoised edges are extracted which correspond to depth discontinuities with high probabilities. Small differences of neighboring disparities near the denoised edges are penalized.

It is surprising that although stereo vision is commonly present in biological systems, there are not many biologically inspired stereo vision models. Most of the available methods rely on purely mathematical or computational models. One biologically inspired method is BioDEM, [Martins et al., 2011] which consists of a disparity energy model using a trained neuronal population. Actually, two neuronal populations are used: one for encoding, which consists of a set of neurons tuned to a wide range of parameters such as horizontal disparities, spatial frequencies and orientations; and another for decoding which estimates the disparity. The same authors also proposed an improvement over this method, LCVB-DEM [Martins et al., 2015] which combines luminance, color, viewpoint and line and border detection with the disparity energy model. This model is currently the best bio-inspired stereo model in the Middlebury dataset. However, although these bio-inspired models can generate disparity maps that could be good enough for disparity-based saliency or disparity-based robot navigation, they are computationally too expensive for real-time applications.

Also related to stereo and to the Martins et al. work, in our visual system there are binocular neurons, which are neurons that assist in the creation of stereopsis from binocular disparity. These neurons are present in the primary visual cortex, where the first steps of binocular convergence begin [Qian and Zhu, 1997; Scholl et al., 2013]. Binocular neurons are considered to have receptive fields in both left and right eyes and integrate the signals together to create a perception of depth. This type of cells can be tuned to a wide range of spatial frequencies [DeValois et al., 1982; Shapley and Lennie, 1985] and phase differences [Qian and Zhu, 1997], responding differently to stimuli with different disparities. According to some authors, these cells can be classified to 3 types: far cells, near cells and zero-tuned cells. Far cells respond to disparities in planes more distant than the plane of fixation, near cells respond to stimuli in planes closer than the plane of fixation, and zero-tuned cells respond to disparities in the plane of fixation [Purves et al., 2012]. In some work this classification is extended to 6 types: near, far, near tuned, zero tuned and inhibitory tuned [Poggio and Fischer, 1977; Poggio et al., 1988; Poggio, 1990].

Going back to disparity maps, one of the keys for fast disparity maps that can be used by a robot in real time, is a fast (keypoint) descriptor. The most known keypoint descriptors are SIFT [Lowe, 2004] followed by SURF [Bay et al., 2008]. Both yield a floating point representation and for matching they use the Euclidean distance. However, binary descriptors such as ORB [Rublee et al., 2011] and BRISK [Leutenegger et al., 2011] have been getting more and more attention because they need less memory and are much faster to extract and match. This makes them suitable for real-time applications, even on embedded devices with low computational power.

There are also biologically inspired keypoint detectors, such as the ones based on simple, complex and end-stopped cells from cortical area V1 [Rodrigues and du Buf, 2006, 2009] and a binary biologically-inspired descriptor, which encodes the cell responses into a binary vector using LDAHash [Strecha et al., 2012]. This descriptor was shown to clearly outperform any other descriptors with biological inspiration and it also slightly outperforms the SIFT-based LDAHash (see also Chapter 3).

Finally in our previous work, Saleiro et al. [2012] developed a minimalistic vision-based cognitive SLAM system comprising visual saliency, object segregation and object recognition. The vision processes were integrated with a cognitive memory structure composed of short- and long-term memories, the first one having a small capacity and storing only information necessary for immediate navigation. The latter stores important information, selected from short-term memory, for longer periods of time so that it can be used for global navigation. The system was tested on a small robot in a small sandbox. The vision components of this system used standard computer vision methods for saliency and object recognition. In more recent work, Saleiro et al. [2014] developed a preliminary version of the biologically-inspired binary keypoint descriptors and stereo algorithm. The authors also developed saliency based on color, motion and disparity, and integrated the three vision models in a child-sized mobile robot and performed navigation tests in a real environment, with both static and mobile obstacles. In this work the robot-head was static and no active vision was used.

4.3 Biologically inspired architecture of the active vision system

The biologically inspired robot head architecture is composed of (a) an acquisition module, (b) vision modules and (c) a motor control module. When connected together they create an action-perception loop for active vision.

The acquisition module is based on a stereo camera, which captures a pair of images. These images are used as input for the vision modules, i.e., to create the disparity and saliency maps. The saliency module takes as input (besides other features) the disparity, and the output of the saliency module is fed into the Focus-of-Attention (FoA) module, that provides salient regions of the image to the object recognition module (by the order of saliency). The output of the saliency module is also used as input for the Attractor Dynamics Motor Control module. Fig. 4.1 illustrates the generic system architecture.

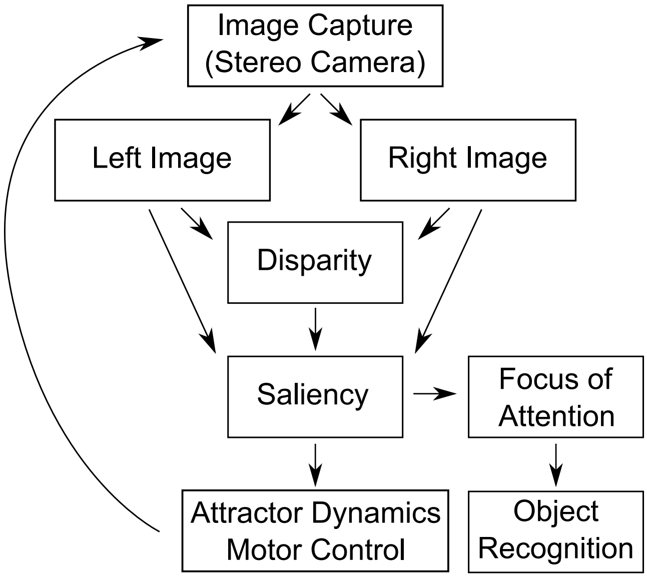


Figure 4.1: System architecture.

For a better comprehension of the following sections, we briefly describe the vision process (see Fig. 4.2). It starts with image capture by the left and right cameras from the stereo camera. V1 cortical cell responses and CIE LAB color components [Martins et al., 2008] are computed for each image. V1 cortical cell responses [Rodrigues et al., 2012] are also used for line/edge and keypoint extraction. Disparity maps are computed using the cell responses, line/edges and LAB color channels. Disparity maps and LAB color components are used as input for the saliency module that generates regions of interest (RoI). In each RoI the keypoints are computed and the keypoint descriptors are created from the left image. These are matched against keypoint descriptors from previously stored templates in memory and the system is able to perform object recognition. The visual process is illustrated in Fig. 4.2. The following sections describe the whole process in detail.

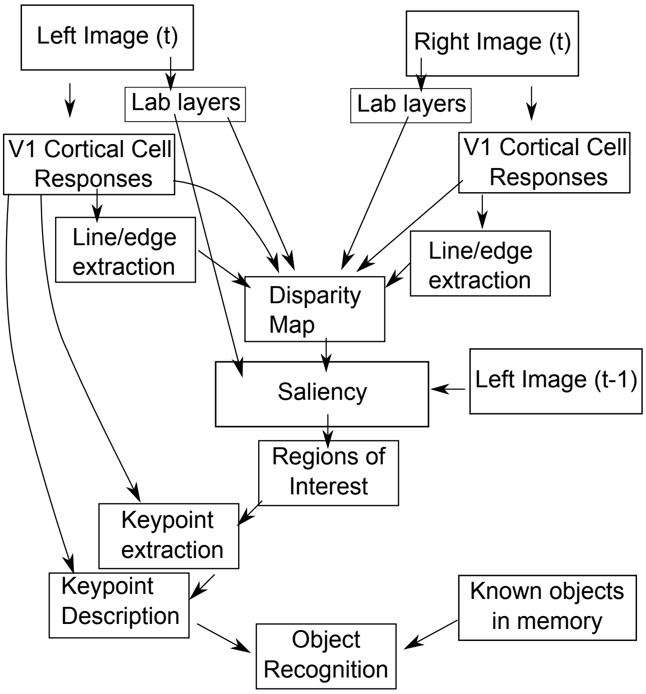


Figure 4.2: Vision component overview.

4.4 Multiscale binocular stereo cells

From a biological point of view, our stereo model takes inspiration from the concept that there are binocular cells (see e.g. Qian and Zhu [1997]; Scholl et al. [2013] and Chap. 4.2) tuned to different disparities and that, like many other neurons in the visual cortex, can receive excitatory or inhibitory input from other neurons. From an implementation point of view, in order to achieve real-time performance, the stereo vision model we propose takes inspiration from the Binary Stereo Matching algorithm [Zhang et al., 2012], which applies the BRIEF binary keypoint descriptor [Calonder et al., 2012] to every single pixel of both left and right images to perform matching between pixels of both images.

This matching is combined with a binary mask built by comparing the absolute color difference in the LAB color space between pixels from left and right images. However, to meet our speed requirements and to improve the results we use a different descriptor and apply the binary mask in a different way, as well as add some extra steps to the algorithm that we will explain in the following sections.

As mentioned before, the stereo model has also input from the V1 cells. Let's briefly define the multi-scale information retrieved by the cells in visual cortical area V1. In this area we find simple, complex and end-stopped cells [Rodrigues and du Buf, 2006], which are thought to play an important role in coding the visual input: they allow to extract multiscale lines and edges [Rodrigues and du Buf, 2009] and keypoint information (keypoints are line/edge vertices or junctions, and also "blobs"). Responses of even and odd simple cells, which correspond to the real and imaginary part of a Gabor quadrature filter [Rodrigues and du Buf, 2009], are denoted by $R^E_{s,i}(x, y)$ and $R^O_{s,i}(x, y)$, being i the orientation. The scale s is given by λ, which is the wavelength of the Gabor filters, in pixels. Responses of complex cells are modelled by the modulus, $C_{s,i}(x, y) = [\{R^E_{s,i}(x, y)\}^2 + R^O_{s,i}(x, y)^2]^{1/2}$ (for more details see Rodrigues and du Buf [2006]).

The basic scheme for line and edge detection is based on responses of simple cells: a positive or negative line is detected where R^E shows a local maximum or minimum, respectively, and R^O shows a zero crossing. In the case of edges the even and odd responses are swapped. This gives four possibilities for positive and negative events. An improved scheme proposed by Rodrigues and du Buf [2009] consists of combining responses of simple and complex cells, i.e., simple cells serve to detect positions and event types, whereas complex cells are used to increase the confidence. Lateral and cross-orientation inhibition are used to suppress spurious cell responses beyond line and edge terminations, and assemblies of grouping cells serve to improve event continuity in the case of curved events. We denote the line and edge map by $LE_s(x, y)$.

If two edges, e.g., one vertical and one horizontal, meet in a point, complex cells tuned to the edge orientations will produce a maximum response at that position. Cells tuned to other orientations will also respond at the edges, but with weaker responses. However, if we compute the first and second derivatives of the responses of the complex cells we obtain single and double end-stopped cells, in the same orientations as those of the complex cells. This implies that single-stopped cells will produce responses that are maximum on both sides but zero in the middle. On the other hand, double-stopped cells will produce responses that are maximum at the edge positions and decrease on both sides. By summing all responses over all orientations we get a peak at the corner, where all cells respond. At these peak locations we extract keypoints; $K_s(x, y)$, [Terzic et al., 2015].

Our algorithm follows 7 major steps, which are repeated for each scale (s): (a) LAB image resizing; (b) binary mask computation; (c) building the binary pixel descriptors; (d) pixel descriptor matching between left and right images (pixel correspondence); (e) DoG filtering; (f) final disparity map, built from matching costs contribution, spatial filtering and a winner-take-all (WTA) approach.

As mentioned, our approach is multi-scale, taking a coarse-to-fine approach. We start with the coarser scales, which are much faster to compute and result in a coarser disparity map, and use progressively finer scales to get more detailed disparity maps. Disparity information from coarser scales is weighted according to their level of certainty and is provided as input to finer scales where certainty is low. This progressive multi-scale approach not only yields good results but also permits to find a good compromise between computation time and detail of disparity maps for the pretended application. It is important to stress that in most of the applications small and coarse disparity maps are good enough to compute the saliency for object detection.

4.4.1 LAB image resizing

If $I^{l,r}$ are the original left (l) and right (r) images acquired by the RGB stereo camera, we start by computing a stack of N_s differently-sized RGB images (with N_s the number of scales), so that they have smaller sizes as a function of the scale s, i.e., for instance for three scales $s = \{1, 2, 3\}$ with $\lambda = \{4, 8, 16\}$ we compute a stack of 3 RGB images, having the first one the same size of the original images, the second one half of that, and the third one a quarter of the original size. As the scale doubles, the image size gets divided by a scaling factor (sf) of 2. After computing the stack of differently scaled RGB images we convert them to LAB color space, obtaining $I^{l,r}_{s,c}$, being s the scale of the image and $c = \{L, A, B\}$ the different components of the LAB color space.

These images will be used in the next step to compute the binary masks for each scale.

4.4.2 Binary masks

Like BSM [Zhang et al., 2012], our algorithm uses binary masks as part of the matching cost computation in order to minimize the problem known as edge-flattening.

For the three components of the CIE LAB color space (c), and for each pixel (p_c^l) from the left image ($I_{s,c}^l$) we compute the sum of the absolute differences (SAD) between that pixel and next N_d pixels (p_c^r) from the right image ($I_{s,c}^r$) on the epipolar line, i.e., $\mathrm{SAD}(m,k) = \sum_{c=\{L,A,B\}} |p_c^l(m) - p_c^r(m+k)|$, with m the pixel coordinate (x, y), $k = \{0, ..., N_d\}$, being $N_d = \frac{m}{sf}$, md the maximum disparity and sf the scale division factor for the current scale.

The next step consists of computing the binary mask ϕ, with size N_d, for each pixel position (m), by using a binary threshold BT which corresponds to the N_{th} smallest SAD value (e.g. fifth smallest value, for $N_{th} = 5$), with $N_{th} = log_2(md/sf)$. The binary mask is given by

$$\phi(m,k) = \begin{cases} 1 : SAD(m,k) \leq BT \\ 0 : SAD(m,k) > BT. \end{cases}$$

The binary masks will afterwards be used to calculate the matching cost between pixels (see Section 4.4.4).

4.4.3 Binary pixel descriptors

For the pixel correspondence between left and right images to create the disparity map, instead of using the BRIEF descriptor as used in BSM [Zhang et al., 2012], we created a simpler and faster descriptor to suit our purpose. Apart from being slower, BRIEF uses a 32 × 32 window to sample the information to build the descriptor. This causes a 16-pixel wide "waste" of information near the borders of the images, which is especially problematic when dealing with small images. The same arguments apply to the descriptor explained in

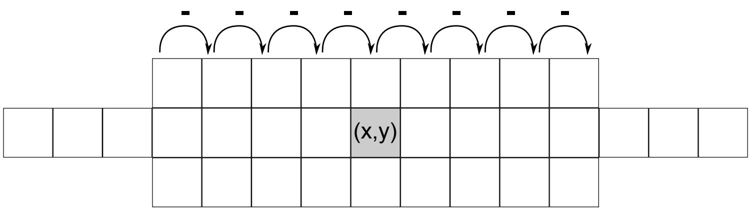


Figure 4.3: Descriptor sampling pattern.

Section 3.5.3, which was built for object recognition.

Our descriptor for stereo matching is a 256-bit binary vector, built using a sampling region of 33 cell responses (see Fig. 4.3) around and including the pixel position (x, y). It uses information from odd simple cells ($R^O_{s,i}$) at 8 orientations (sampling region of 33 × 8); similar results can be achieved by using the complex cells or the even simple cells. For each pixel position x we take the 8 cell responses, before and after the pixel position. The same is done on the lines below and above, but now for the 4 pixels before and after the pixel position.

In the sampling region defined above, each value of $R^O_{s,i}$ is compared with its right neighbor and if it is bigger the bit will be 1, otherwise it will be 0 The same is done on the above and below lines, returning 32-bit responses for each of the 8 orientation (32 × 8 = 256-bit descriptor),

$$\hat{\gamma}_{\{s,i\}}(x+i,z) = \begin{cases} 0 : R^O_{s,i}(x+i,z) \leq R^O_{s,i}(x+i+1,z) \\ 1 : R^O_{s,i}(x+i,z) > R^O_{s,i}(x+i+1,z). \end{cases}$$

with $i = \{-8, ..., 8\}$ for $z = y$ and $i = \{-4, ..., 4\}$ for $z = \{y-1, y+1\}$.

To create the 256-bit descriptor we simply concatenate all $\hat{\gamma}$, i.e.,

$$\gamma_s(x,y) = \underset{i=\{0,...,7\};z=\{y-1,y,y+1\}}{CONCAT}(\hat{\gamma}_{\{s,i\}}(x,z))$$

The overall computation speed comes from the fact that to compute the descriptor for position $(x+1, y)$ we just have to rotate the bits of all the $\hat{\gamma}_{\{s,i\}}$ segments by one position to the left, and perform the same comparison described above with the new cell responses from the right, placing the new bits at the free position on the right of each of the $\hat{\gamma}_{\{s,i\}}$. Using this strategy we get a fast binary descriptor with information sampled from the region around each pixel.

4.4.4 Pixel correspondence

In this step we calculate and store the matching cost (M) between each descriptor $\gamma_s(x,y)^l$ from the left image and the next N_d descriptors on the same epipolar line starting from the same position (x,y) in the right image ($\gamma_s(x,y)^r$). The matching cost is computed using the Hamming distance between descriptors. At the end of this matching process we have for each scale s an array of N_d images with the matching costs between each descriptor and the descriptor at position m at all the possible horizontal displacements $m+k$ (see also Section 4.4.2).

From a biological point of view, we can consider each of these images as populations of binocular cells tuned to a certain horizontal disparity and each matching cost is the response of a binocular cell, however the response amplitude is inverted: the lower the matching cost, the bigger the cell response.

After calculating all the matching costs (M) we combine them with the binary mask previously computed in Section 4.4.2, $\phi(m,k)$, and obtain the combined matching costs, M_ϕ. The binary mask based on color information is used to boost or decrease the matching costs. Each combined matching cost, M_ϕ, is computed by

$$M_\phi(m,k) = \begin{cases} M(m,k) \times (1+\kappa) : \phi(m,k) = 0 \\ M(m,k) \times (1-\kappa) : \phi(m,k) = 1 \end{cases}, \kappa \in [0,1],$$

with m the (x,y) position, and $k = \{0,...,N_d\}$ (see Section 4.4.2). All empirical tests showed that the best results were obtained by using $\kappa = 0.2$.

4.4.5 DoG filtering

In this step we apply a 1D Difference-of-Gaussians (DoG) function to the matching costs, M_ϕ, along the displacement dimension (k) for each pixel position. The objective is to make adjacent low matching costs boost each other and erroneous low matching costs get penalized by adjacent high matching

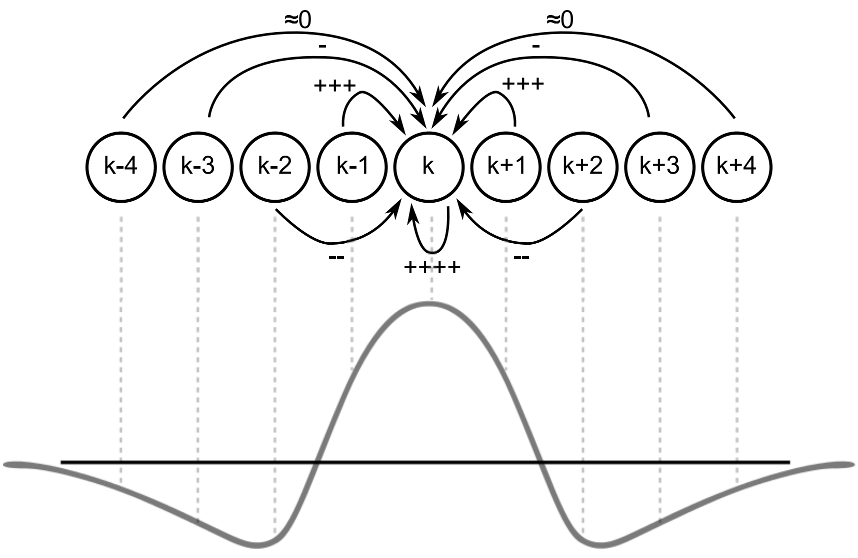


Figure 4.4: Example showing the use of the DoG to model the inhibition and excitation interactions between cells tuned to different disparities.

costs. From a biological point of view we could see this filtering process as a series of excitation and inhibition interactions between binocular neurons tuned to different disparities.

Being $M_\phi(m,k)$ the 3D matrix ($m = (x,y)$) with the k elements, the vector containing the matching costs of all possible displacements at position m, the filtered matching costs, $M'_\phi(m,k)$, are computed as follows:

$$M'_\phi(m,k) = \tau \times M_\phi(m,k) * \left(\alpha \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{k^2}{2\sigma_1^2}} - \beta \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{k^2}{2\sigma_2^2}} \right),$$

with $\tau = 0.4$, $\sigma_1 = 1/sf$, $\sigma_2 = 8/sf$, $\alpha = 1$ and $\beta = 0.5$ (obtained empirically).

4.4.6 Disparity map

Since we are using multiple scales, at this point we can combine the filtered matching costs of the current scale ($M'_{\phi,s}(m,k)$) with the final matching costs (M_F) from the previous, coarser scale.

In case the image of the previous scale is smaller than the one of the current scale, the matching costs matrix is upsampled to the same size (by multiplying all dimension (m,k) by sf) of the matching costs matrix of the current scale.

While matching costs below a value L remain unchanged (we used $L = 10\% \times$ MAX($M'_{\phi,s}$); typically the value of L is 20), all the other matching costs are changed through an average with matching costs for the same displacement from the previous coarser scale, $M_{F,s-1}$:

$$M_{f,s}(m,k) = (M'_{\phi,s}(m,k) + M_{F,s-1}(m,k))/2,$$

with,

$$M_{F,s}(m,k) = \begin{cases} M_{f,s}(m,k) + |M_{f,s}(m,k) - w_s| \times r : w_s > M_{f,s}(m,k) \\ M_{f,s}(m,k) - |M_{f,s}(m,k) - w_s| \times r : w_s \leq M_{f,s}(m,k), \end{cases}$$

where w_s is the weighted sum of the matching costs at $M_{f,s}(m,k)$, in a 3×3 window centered at m, ignoring the values at positions that contain lines or edges, i.e., LE_s (lines and edges appear due to discontinuities in images, where they are used as barriers for spatial filtering). The value of $r = 1.6$ was determined empirically.

The above process is repeated while the number of "stable" values between two iterations is smaller than $w\%$ (we used $w = 10\%$) of the total number of elements in the matching costs matrix. A value is considered stable if both $M_F(m,k)$ and $M_f(m,k)$ are smaller than a threshold value Th_D (we used $Th_D = 25$), and it will not change in remaining iterations. The value $w = 10\%$ can be changed to a higher/lower value if we pretend a faster or more precise final disparity map.

This step helps in clearing outliers, especially in flat regions. From a biological point of view we can consider this step as excitation and inhibition between binocular neurons tuned to the same disparities at different scales.

The final disparity map (for the left image, $D^l(x,y)$) is generated by using a winner-take-all approach and taking only the displacements with the smaller matching cost for each position m.

Once more, this final step can be seen as a series of interactions between binocular neurons that keep occurring until stability of the disparity map has been achieved.

4.5 Saliency

To detect salient regions in the captured images a saliency module that combines (a) color with (b) disparity and (c) motion to create a saliency map, $S^l(x, y)$, was developed.

For the color component (a) we build a stack of 6 retinotopic maps representing different channels in the CIE LAB color-opponent space. The CIE LAB model is based on retinal cones and provides a standard way to model color opponency (see Saleiro et al. [2015] or Chapter 2). The first three channels code the image in LAB color space, thus represent white, green and blue color components. The other three channels are the complements of the first three channels, thus represent black, red and yellow components. After computing the retinotopic maps we apply a blob-detection kernel based on a Difference of Gaussians operation.

The same process is applied to (b) the disparity map, $D^l(x, y)$, after we threshold (15% of the maximum value of D^l) it to get only the nearest regions.

For the motion component (c) we calculate the optical flow using Farneback's method [Farneback, 2003] with images I^l_t, I^l_{t-1} (see Fig. 4.2). The flow's magnitude and direction are processed separately, creating a feature stack of magnitudes and orientations of motion. Once more, the same blob-detection kernel used for the disparity and color is used. We stress that t reflects the present image and $t - 1$ the previous image.

The saliency map (S') is obtained by summing the individual color maps, the individual motion maps and the disparity map, giving equal weights to each of the three different components. Since a saliency map does not need to be detailed, we compute it using the subsampled images for faster processing. After computing the above visual saliency map we can threshold it to 66% (2/3) of the maximum value of S' obtaining the final saliency map S. Interesting regions can then be processed for object recognition.

Figure 4.5 shows the examples of the resulting saliency map. On the left, 1st and 3rd rows, we can see the image captured by the left camera and on the right the respective disparity map is shown. In the 2nd and 4th rows, left

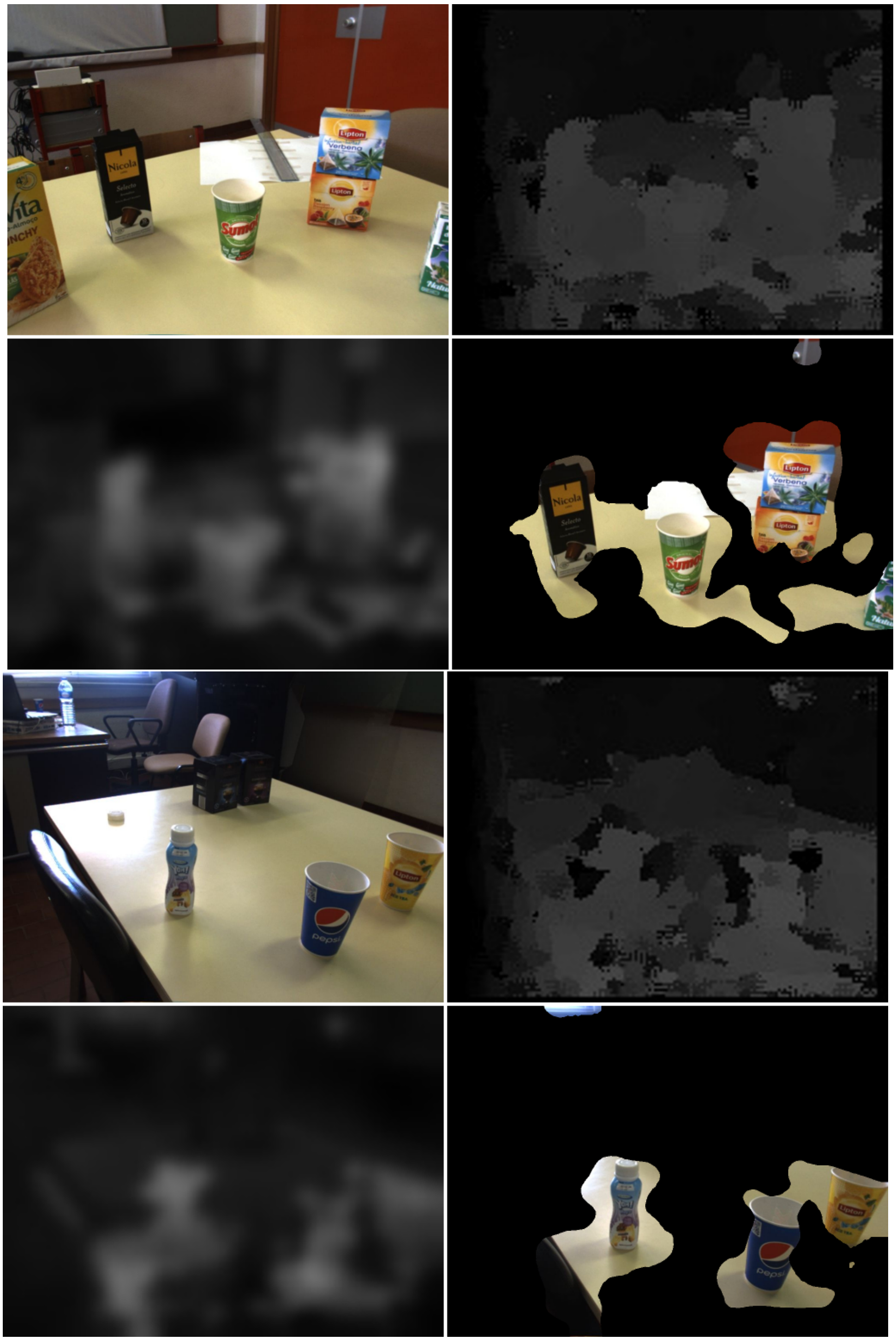


Figure 4.5: Two examples (1st and 3rd row) showing the resulting saliency map (2nd and 4th row). On the top we can see the image captured by the left camera (left) and the respective disparity map (right). On the bottom we can see the saliency map (left) and the highlighted regions (right) that will be used for object recognition (see Fig. 4.6).

and right, we can see the resulting saliency S^l, and the representation of the objects cropped by the saliency. It is important to stress that in the saliency map on the left of 4th row the two black boxes (see image in the 3rd row left) do not appear as salient, as expected, once they are "not important" for the near field robot vision or for robot navigation.

4.6 Object Recognition

For object recognition we selected a set of objects to which we apply the biologically inspired keypoint extraction of Terzic et al. [2015] and the binary keypoint descriptor algorithm that was presented in Chapter 3.

The resulting keypoint descriptor arrays are then stored in the robot's memory for faster processing during runtime. During robot activity, these descriptor arrays are matched to keypoint descriptors extracted from the salient regions (S^l; see Fig. 4.5) of the images captured by the robot's left camera. Since we use binary descriptors, matching is quickly done by calculating the Hamming distances.

We consider that when at least 5 of the descriptors of a certain object are detected, i.e. by having a Hamming distance smaller than 20% (i.e., 25 in 128 bits), a match is confirmed. Figure 4.6 shows the recognition of the objects from the image presented in Fig. 4.5, 1st row on the left, using the saliency map presented on the left of the 2nd row of the same figure. All objects were recognized, and all the matches were "true matches" with the exception of one. However, although it matches to a keypoint in a wrong object, it's actually a correct match in terms of keypoint matching because it matches to the "Lipton" logo that exists in both tea boxes (Fig. 4.6 bottom row).

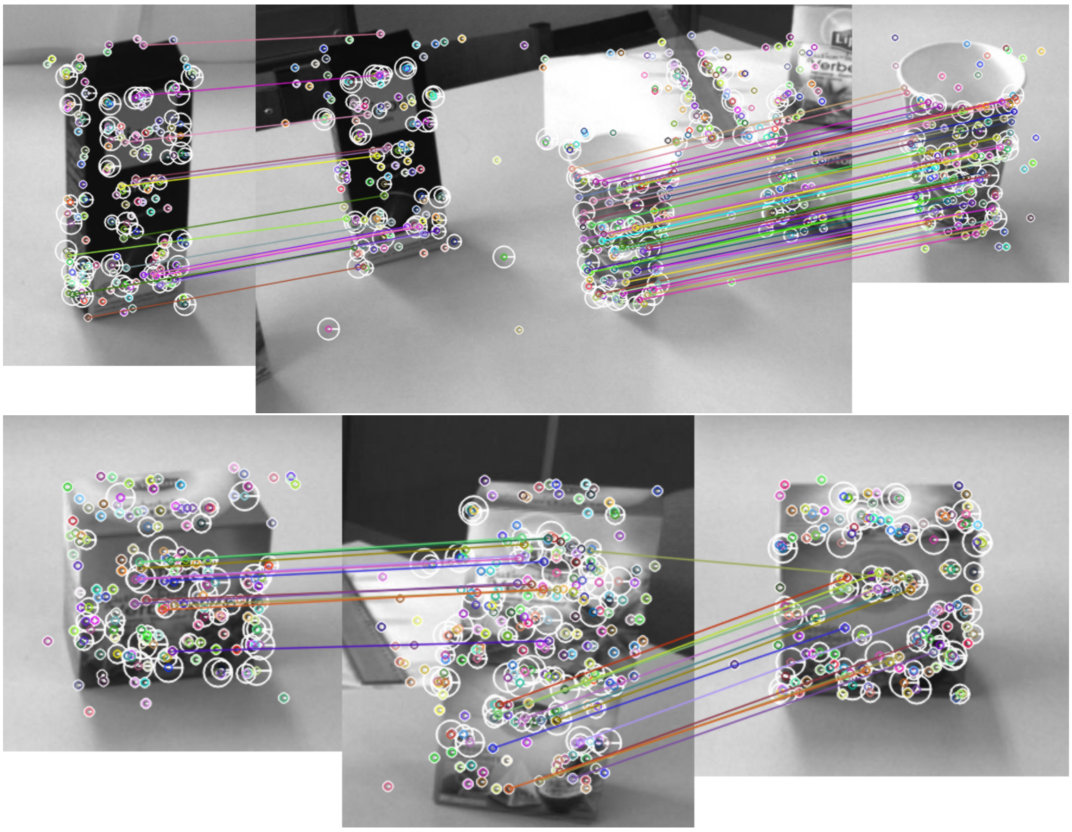


Figure 4.6: Object recognition from the image presented in Fig. 4.5, 1st row at left. All objects were recognized.

4.7 Attractor dynamics for robot head pan control

To control the robot head, we tried to model typical human behavior, generically characterized by how humans (and many animals) explore their field of view using eye saccades, and only moving the head when they need to change the field of view to a “new” attraction point.

Our test setup was a table top, following the setup proposed by Leitner et al. [2013]. We used a fixed tilt angle, with the robot cameras facing a little downward to look at the objects on the table top (see also Section 4.8). To control the horizontal rotation of t he head (pan) we used an attractor dynamics approach, and the “center” of the salient regions (S^l) of the images were used as attraction points.

Salient regions in the current camera field of view are analyzed for object recognition by order of saliency (FoA) without requiring any head movement. However, groups or individual salient points can create attractors which make the head of the robot rotate (camera pan; φ) to better fit those salient regions into the camera's field of view. In order to create an exploring behavior we also use a time-variant repulsive forcelet having its main repulsive point at the current pan (φ) orientation. The strength of the repulsive forcelet for a given orientation increases with time, which forces the robot to switch to a different attraction point.

To implement the above behavior we split the 180^o robot's head (not camera) field of view, ranging from -90^o to 90^o, into 9 sections, each with a 20^o range, and stored the orientation values of the 9 sections in an array $Ao = \{-80^o, -60^o, ..., 0, ..., 60^o, 80^o\}$. In the present setup, the horizontal FoV of the robot camera was only 43^o.

Considering the saliency map S^l of size $M \times N$, we split it into B regions (taking into account the small horizontal field of view of our robot camera we used $B = 3$) corresponding to B of 9 sections that correspond to the orientations of Ao, depending on the robot head orientation (φ) With vertical size N and horizontal size M/B, each region has a size of $(M/B) \times N$. Then, for each of the B regions, we compute the maximum saliency value inside salient regions with a size of at least 200 pixels. These saliency values are stored in a vector As_t, with the same size as Ao.

In order to create an exploration behavior and to make the robot change it's field of view after some time, we applied a repulsive forcelet, which is built in a similar way as the attractor, but taking a repulsion vector, Ar_t (same size as Ao). Ar_t starts with all values set to a maximum, which corresponds to minimum repulsion, except in the first and last position of the array. In these positions the values are set to a minimum. This creates maximum repulsion which prevents the robot from rotating its head more than 90^o in each direction. Every time (t) that a new frame is captured at an orientation φ, the Ar_t that corresponds to the φ section in Ao is decremented by a constant (5% of the minimum repulsion value), which increases repulsion at that orientation. The values of the remaining sections of Ar_t are incremented with the same constant, decreasing the repulsion forces.

As mentioned, there are two vectors (with the same dimension as Ao) to create the attractor (Φa) and the repulsion (Φr) forcelets, namely:

$$\Phi a_t = -\alpha a \times Ao \times e^{\frac{As_t}{\beta a}} \times e^{\frac{Ao^2}{\epsilon_a}},$$

and

$$\Phi r_t = \alpha r \times Ao \times e^{\frac{Ar_t}{\beta r}} \times e^{\frac{Ao^2}{\epsilon_r}},$$

with $\alpha a = \alpha r = 50$, $\beta a = \beta r = 10$, $\epsilon_a = 10$, and $\epsilon_b = 15$. All values were empirically estimated. By changing these values we can change the behavior of the robot head.

By summing all the values of vector Φa_t at each frame aquisition (t) we get the attraction rate towards the strongest attraction point, and by summing all the values of Φr_t we get the repulsion rate away from the strongest repulsion point.

The final pan movement rate of the camera/robot head, $\hat{\varphi}_t$, at each instance t, is then calculated by summing all Φa_t and Φr_t, i.e.,

$$\hat{\varphi}_t = \Sigma_9(\Phi a_t) + \Sigma_9(\Phi r_t),$$

.

The value $\hat{\varphi}_t$ is converted to the pan motor control of the robot head at instance t.

To get the angular position which must be applied to the pan motor, it is only necessary to multiply the attraction rate with a constant value, which depends on the specifications of the motor driver (in our case the value is 1 since the motor driver accepts angle commands in radians).

Since the attraction rate decreases as the pan angle gets closer to the attraction point, the robot head will eventually reach a stable state which will only be affected if a much stronger attraction point appears in the camera's field of view, or if a strong repulsion point appears at the current robot head orientation. Otherwise the robot head pan angle will tend to remain unchanged.

To create the exploration behavior we simply increase the repulsion at the section that corresponds to the current robot pan angle every time that a new image is captured. On the other hand, all the other sections get their repulsion force decreased. This leads to a big repulsive effect at the current robot orientation after some time, forcing it to move to a new attraction state.

This is a very versatile strategy, since different behaviors can be created by simply manipulating the repulsive and attraction forces. As an example, for robot navigation an attraction point could emerge periodically at orientation zero (looking in front) to force the robot to detect obstacles frequently. The same could be done for other orientations in order to make the robot look to the sides every now and then, since the horizontal FoV of the camera is very small if compared to the FoV of our eyes. To create a tracking behavior repulsion can be increased at all orientations except at the orientation where the object is detected.

Figure 4.7 shows one example of vision steering using attractor dynamics. At the top we can see the captured image which is processed for saliency and split into $B = 3$ parts. The middle and right parts contain salient regions and the maximum saliency value of these regions are placed in the positions of the attractor forcelet array that correspond to the current head orientation. The orange arrows show where the two attractor points are in the plot. Attractor points are represented by decreasing pink curves which cross the horizontal axis of the plot. In the second plot, on the right, we see that a big repeller appears at the current orientation (ascending yellow curve crossing the zero axis) forcing the robot orientation (black dot) to move to the attractor point on the right. Finally we can see the captured image after the head rotation. In the following sections a summary of the evaluation tests will be presented.

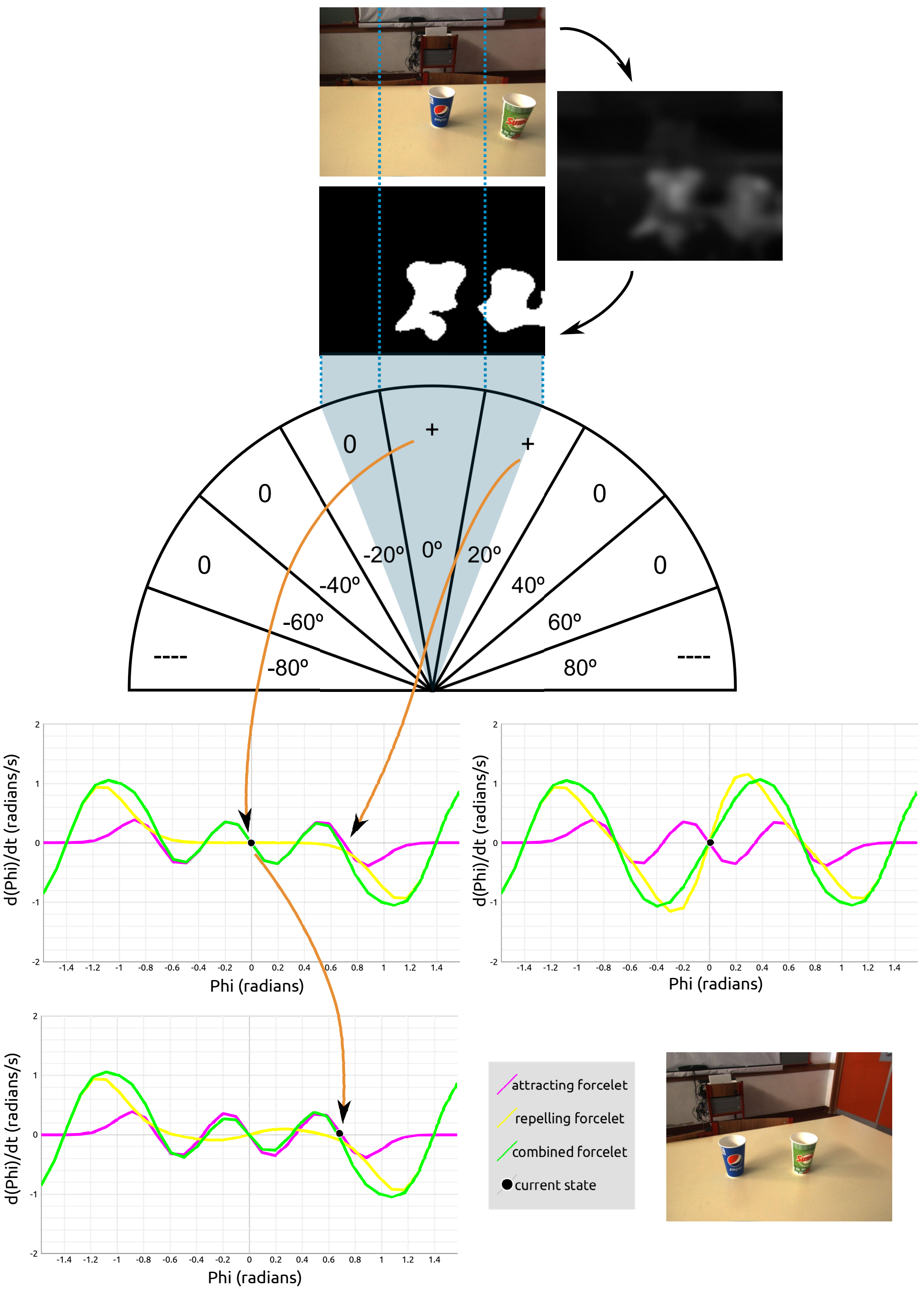


Figure 4.7: Example of vision steering using attractor dynamics (see text).

4.8 Evaluation and tests

In the following sections we will present: (a) the robot setup developed to test the active vision, (b) the tests and evaluation of the stereo model, a key element of the active vision model, and (c) the robot active vision tests realized in real environments. The last addressed scene exploration, object recognition and tracking.

4.8.1 The robotic platform

As briefly already mentioned along the text, to test the biologically inspired active vision system we used a child-sized Pioneer 3DX robot, equipped with a Bumblebee-2 stereo camera and a PhantomX turret for pan and tilt camera movement.

The robot structure (see Fig. 4.8) has several platforms mounted in order to give it more height, and provide the point-of-view of a child with a height of 1.20 m. The robot has been set up with ROS (Robot Operating System). Although the robot is also equipped with a laser rangefinder and 6 ultrasonic sensors, they were not used.

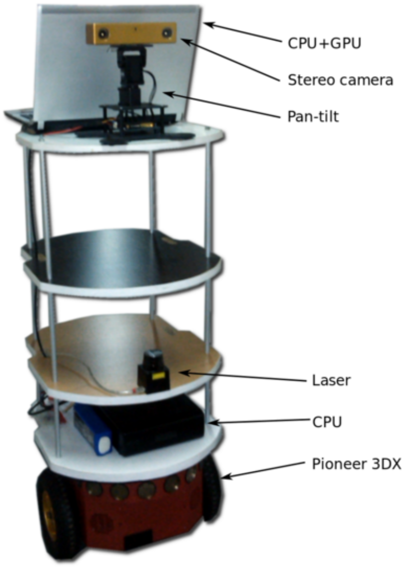


Figure 4.8: Child-sized robot used for the active vision tests.

For processing we used a notebook with an Intel I7-4700HQ quad-core processor and 16GB of DDR3 RAM. The average achieved processing frame rate was 2.4 frames per second.

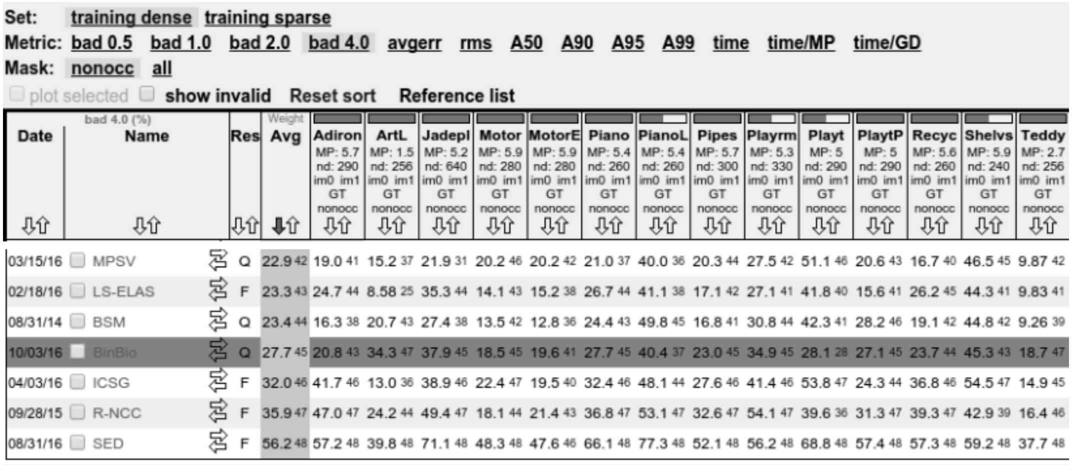
To achieve real time processing we used only scale $\lambda = 16$ for stereo computation during the tests except for the Middlebury stereo evaluation.

4.8.2 Middlebury dataset for stereo evaluation

The goal of developing the new stereo model was to get fast, reliable and biologically-inspired stereo vision that could be used in real-time robotics applications. It was tested using the Middlebury evaluation dataset [Scharstein and Szeliski, 2002].

The dataset (version 3) is composed of training and testing datasets, with 15 image pairs each, having each a different image size and a different maximum disparity value. Ground-truth images are provided for the training set but not for the testing set to prevent training with the testing images, as occurred in version 2 of the evaluation set. Moreover, submission of results is only once possible. First we proceeded to optimize our method's parameters using the training set, and the best results we achieved were using 7 scales: $\lambda = \{6, 8, 10, 12, 14, 16, 24\}$. Since our method was designed with a real-time goal (speed), we have as "inputs" small images. However, when using bigger images our method requires too much memory, due to the storage of matching costs at all disparities and at several scales. Hence, we were only able to perform evaluation using the quarter size images, which is a big handicap when comparing to other methods that can use half- or full- size images. However, as shown at the top in Fig. 4.9 on the Middlebury training set our model, BioBin, performed almost as good as the previously mentioned BSM (Binary Stereo Matching) of Zhang et al. [2012] from which we got some inspiration. However, in terms of computing speed, our method was 5.3 times faster; see Fig. 4.9 at bottom (we stress that BSM was also evaluated using the quarter-sized images).

Since in the Middlebury evaluation version 2 BSM has a much higher score than BioDEM and in version 3 our method ranks very close to BSM, we could say that our method is probably the best biologically inspired stereo algorithm. To truly validate this affirmation we are still waiting for the Middlebury acceptance of the submitted test results and their final ranking. Figure 4.10 shows



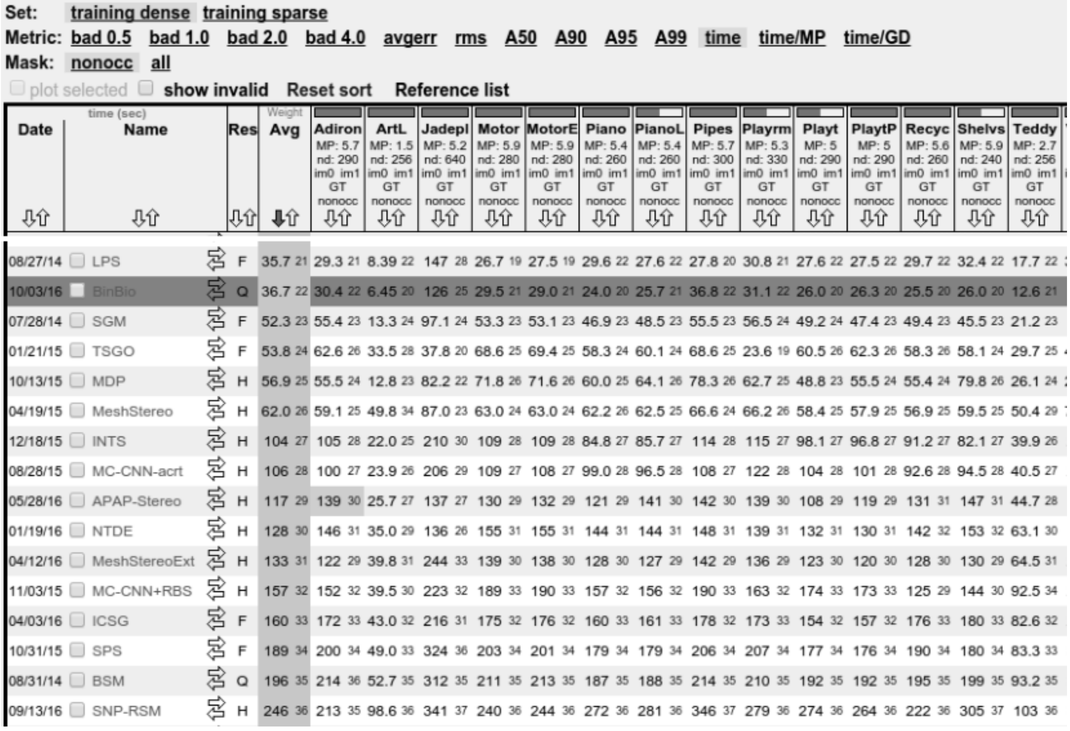
Set: training dense training sparse

Metric: bad 0.5 bad 1.0 bad 2.0 bad 4.0 avgerr rms A50 A90 A95 A99 time time/MP time/GD

Mask: nonocc all

□ plot selected □ show invalid Reset sort Reference list

Date	bad 4.0 (%) Name	Res	Weight Avg	Adiron MP: 5.7 nd: 290 im0 im1 GT nonocc	ArtL MP: 1.5 nd: 256 im0 im1 GT nonocc	Jadepl MP: 5.2 nd: 640 im0 im1 GT nonocc	Motor MP: 5.9 nd: 280 im0 im1 GT nonocc	MotorE MP: 5.9 nd: 280 im0 im1 GT nonocc	Piano MP: 5.4 nd: 260 im0 im1 GT nonocc	PianoL MP: 5.4 nd: 260 im0 im1 GT nonocc	Pipes MP: 5.7 nd: 300 im0 im1 GT nonocc	Playrm MP: 5.3 nd: 330 im0 im1 GT nonocc	Playt MP: 5 nd: 290 im0 im1 GT nonocc	PlaytP MP: 5 nd: 290 im0 im1 GT nonocc	Recyc MP: 5.6 nd: 260 im0 im1 GT nonocc	Shelvs MP: 5.9 nd: 240 im0 im1 GT nonocc	Teddy MP: 2.7 nd: 256 im0 im1 GT nonocc
03/15/16	□ MPSV	Q	22.9 42	19.0 41	15.2 37	21.9 31	20.2 46	20.2 42	21.0 37	40.0 36	20.3 44	27.5 42	51.1 46	20.6 43	16.7 40	46.5 45	9.87 42
02/18/16	□ LS-ELAS	F	23.3 43	24.7 44	8.58 25	35.3 44	14.1 43	15.2 38	26.7 44	41.1 38	17.1 42	27.1 41	41.8 40	15.6 41	26.2 45	44.3 41	9.83 41
08/31/14	□ BSM	Q	23.4 44	16.3 38	20.7 43	27.4 38	13.5 42	12.8 36	24.4 43	49.8 45	16.8 41	30.8 44	42.3 41	28.2 46	19.1 42	44.8 42	9.26 39
10/03/16	□ BinBio	Q	27.7 45	20.8 43	34.3 47	37.9 45	18.5 45	19.6 41	27.7 45	40.4 37	23.0 45	34.9 45	28.1 28	27.1 45	23.7 44	45.3 43	18.7 47
04/03/16	□ ICSG	F	32.0 46	41.7 46	13.0 36	38.9 46	22.4 47	19.5 40	32.4 46	48.1 44	27.6 46	41.4 46	53.8 47	24.3 44	36.8 46	54.5 47	14.9 45
09/28/15	□ R-NCC	F	35.9 47	47.0 47	24.2 44	49.4 47	18.1 44	21.4 43	36.8 47	53.1 47	32.6 47	54.1 47	39.6 36	31.3 47	39.3 47	42.9 39	16.4 46
08/31/16	□ SED	F	56.2 48	57.2 48	39.8 48	71.1 48	48.3 48	47.6 46	66.1 48	77.3 48	52.1 48	56.2 48	68.8 48	57.4 48	57.3 48	59.2 48	37.7 48



Set: training dense training sparse

Metric: bad 0.5 bad 1.0 bad 2.0 bad 4.0 avgerr rms A50 A90 A95 A99 time time/MP time/GD

Mask: nonocc all

□ plot selected □ show invalid Reset sort Reference list

Date	time (sec) Name	Res	Weight Avg	Adiron MP: 5.7 nd: 290 im0 im1 GT nonocc	ArtL MP: 1.5 nd: 256 im0 im1 GT nonocc	Jadepl MP: 5.2 nd: 640 im0 im1 GT nonocc	Motor MP: 5.9 nd: 280 im0 im1 GT nonocc	MotorE MP: 5.9 nd: 280 im0 im1 GT nonocc	Piano MP: 5.4 nd: 260 im0 im1 GT nonocc	PianoL MP: 5.4 nd: 260 im0 im1 GT nonocc	Pipes MP: 5.7 nd: 300 im0 im1 GT nonocc	Playrm MP: 5.3 nd: 330 im0 im1 GT nonocc	Playt MP: 5 nd: 290 im0 im1 GT nonocc	PlaytP MP: 5 nd: 290 im0 im1 GT nonocc	Recyc MP: 5.6 nd: 260 im0 im1 GT nonocc	Shelvs MP: 5.9 nd: 240 im0 im1 GT nonocc	Teddy MP: 2.7 nd: 256 im0 im1 GT nonocc
08/27/14	□ LPS	F	35.7 21	29.3 21	8.39 22	147 28	26.7 19	27.5 19	29.6 22	27.6 22	27.8 20	30.8 21	27.6 22	27.5 22	29.7 22	32.4 22	17.7 22
10/03/16	□ BinBio	Q	36.7 22	30.4 22	6.45 20	126 25	29.5 21	29.0 21	24.0 20	25.7 21	36.8 22	31.1 22	26.0 20	26.3 20	25.5 20	26.0 20	12.6 21
07/28/14	□ SGM	F	52.3 23	55.4 23	13.3 24	97.1 24	53.3 23	53.1 23	46.9 23	48.5 23	55.5 23	56.5 24	49.2 24	47.4 23	49.4 23	45.5 23	21.2 23
01/21/15	□ TSGO	F	53.8 24	62.6 26	33.5 28	37.8 20	68.6 25	69.4 25	58.3 24	60.1 24	68.6 25	23.6 19	60.5 26	62.3 26	58.3 26	58.1 24	29.7 25
10/13/15	□ MDP	H	56.9 25	55.5 24	12.8 23	82.2 22	71.8 26	71.6 26	60.0 25	64.1 26	78.3 26	62.7 25	48.8 23	55.5 24	55.4 24	79.8 26	26.1 24
04/19/15	□ MeshStereo	H	62.0 26	59.1 25	49.8 34	87.0 23	63.0 24	63.0 24	62.2 26	62.5 25	66.6 24	66.2 26	58.4 25	57.9 25	56.9 25	59.5 25	50.4 29
12/18/15	□ INTS	H	104 27	105 28	22.0 25	210 30	109 28	109 28	84.8 27	85.7 27	114 28	115 27	98.1 27	96.8 27	91.2 27	82.1 27	39.9 26
08/28/15	□ MC-CNN-acrt	H	106 28	100 27	23.9 26	206 29	109 27	108 27	99.0 28	96.5 28	108 27	122 28	104 28	101 28	92.6 28	94.5 28	40.5 27
05/28/16	□ APAP-Stereo	H	117 29	139 30	25.7 27	137 27	130 29	132 29	121 29	141 30	142 30	139 30	108 29	119 29	131 31	147 31	44.7 28
01/19/16	□ NTDE	H	128 30	146 31	35.0 29	136 26	155 31	155 31	144 31	144 31	148 31	139 31	132 31	130 31	142 32	153 32	63.1 30
04/12/16	□ MeshStereoExt	H	133 31	122 29	39.8 31	244 33	139 30	138 30	128 30	127 29	142 29	136 29	123 30	120 30	128 30	130 29	64.5 31
11/03/15	□ MC-CNN+RBS	H	157 32	152 32	39.5 30	223 32	189 33	190 33	157 32	156 32	190 33	163 32	174 33	173 33	125 29	144 30	92.5 34
04/03/16	□ ICSG	F	160 33	172 33	43.0 32	216 31	175 32	176 32	160 33	161 33	178 32	173 33	154 32	157 32	176 33	180 33	82.6 32
10/31/15	□ SPS	F	189 34	200 34	49.0 33	324 36	203 34	201 34	179 34	179 34	206 34	207 34	177 34	176 34	190 34	180 34	83.3 33
08/31/14	□ BSM	Q	196 35	214 36	52.7 35	312 35	211 35	213 35	187 35	188 35	214 35	210 35	192 35	192 35	195 35	199 35	93.2 35
09/13/16	□ SNP-RSM	H	246 36	213 35	98.6 36	341 37	240 36	244 36	272 36	281 36	346 37	279 36	274 36	264 36	222 36	305 37	103 36

Figure 4.9: At the top the Middlebury Stereo Evaluation table for the training dataset. Our model is referenced as BinBio. Bottom row: the same but now for execution time.

two examples of our model applied to the Middlebury dataset (version 3). On the top row we show the Playtable image from the training dataset and at the bottom Djembe from the testing dataset.

Finally, it is important to mention that in Fig. 4.9, top, we selected the bad=4 metric because we used the quarter-sized images and when the results are up-scaled for evaluation it is natural to get disparity errors of up to 4 pixels.

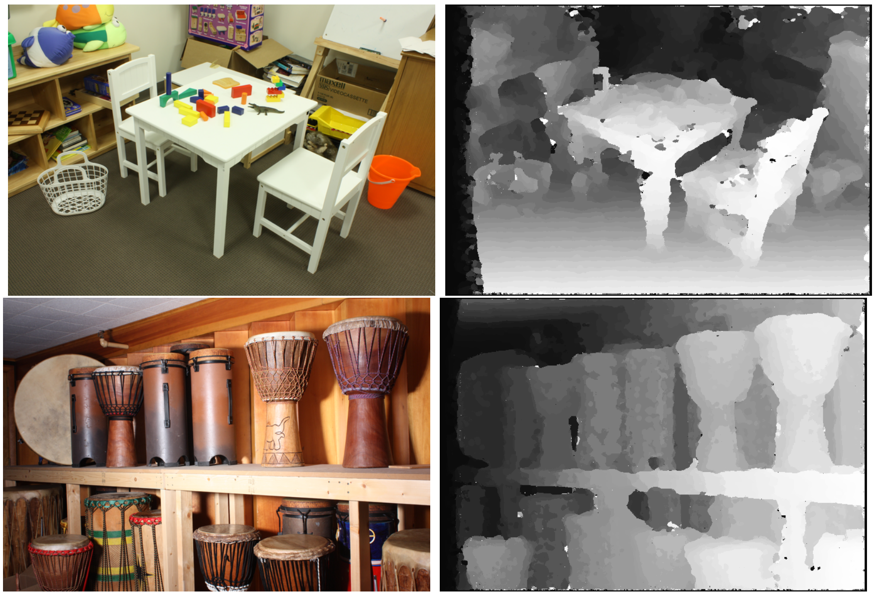


Figure 4.10: Playtable image from Middlebury training dataset (version 3) and Djembe image from Middlebury the testing dataset.

4.8.3 Test results of the active vision system

To test our system we performed three types of tests: (a) presenting several objects in the robot's field of view to see if it is able to direct its attention towards them and recognize them; (b) placing the robot in different static and dynamic scenarios and observe which elements of the scene are focused by the robot's cameras, and (c) test if the system is able to track objects. Again, as previously mentioned, we followed the test setup proposed by Leitner et al. [2013]: a table top scenario with multiple objects on it.

Scene exploration

For the scene exploration tests, in which the robot had to detect and focus peaks of the saliency map by doing head/camera pan movements, we tried multiple table top setups with different numbers of objects, different positions, etc. Regardless of any changes we made to the setup, the robot was able to successfully explore the table top scenario in a very natural way, looking successively at salient regions in its FoV caused by nearby objects. During runtime we removed objects, added new objects and changed object positions, and the robot head would move accordingly to the available stimuli. The two big repulsion values on the sides of the repulsive forcelet array also successfully prevented the robot from rotating its head more than 90°. Figure 4.11 shows a sequence of images with movements of the head to focus the salient regions.

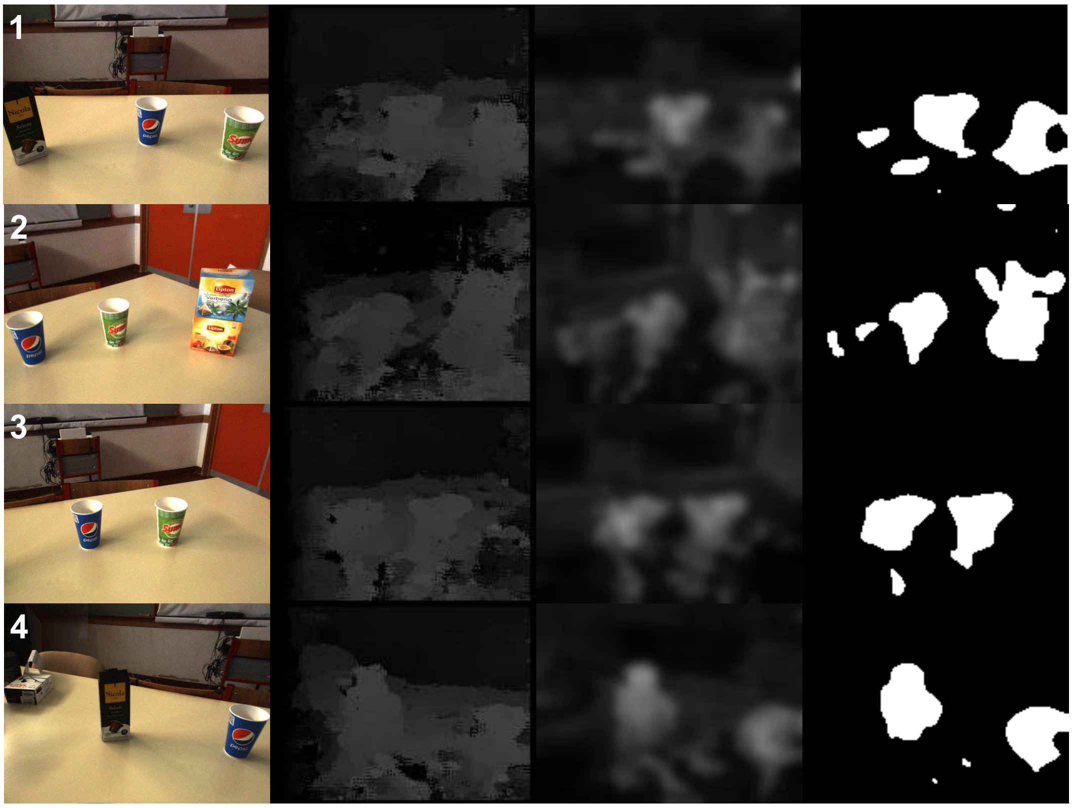


Figure 4.11: Sequence of images showing the movement of the head to focus the salient regions on the table top setup. From left to right: captured image, disparity map, saliency map and thresholded saliency map. Top 4 rows: table top setup; bottom rows: corridor setup.

We also explored non-table top scenarios which also worked as expected. The robot was able to explore different, cluttered and dynamic scenarios and change its head orientation according to the salient information in its FoV. Figure 4.12 shows the corridor setup.

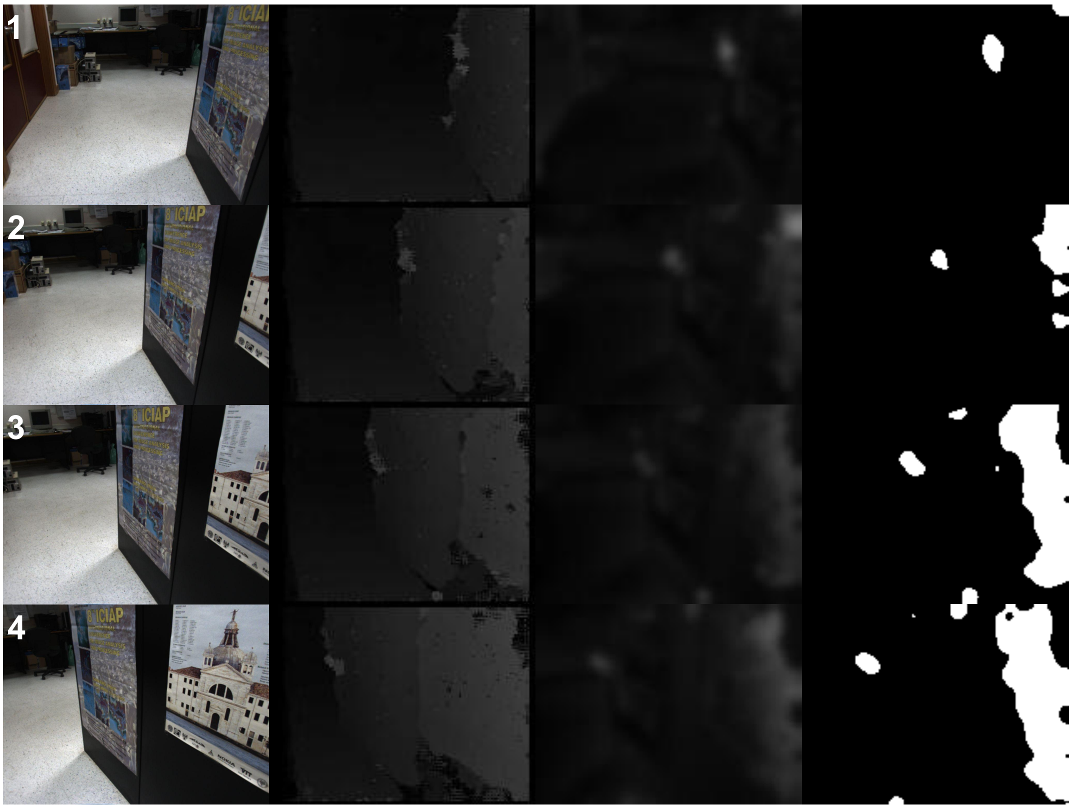


Figure 4.12: Sequence of images showing the movement of the head to focus the salient regions on the corridor setup. From left to right: captured image, disparity map, saliency map and thresholded saliency map. Top 4 rows: table top setup; bottom rows: corridor setup.

Object Recognition

In these tests we used a set of 50 objects: tea boxes, cups, bottles, coffee boxes, etc. Some of the objects were very different (e.g., yogurt bottle vs. tea box) but others were very similar (two Lipton tea boxes, three paper cups). Keypoints scales were $\lambda = \{6, 12, 24\}$.

The tests showed that the number of false positives was quite low, even when the keypoints that were matched were common among similar objects (successful recognition rate was typically around 70%, with the table top scenario even 85%). Examples of the above mentioned erroneous matches were cup borders and the "Lipton" logo on the tea boxes. Figure 4.6 shows some examples of object recognition and in one of them it is possible to see the matching of a

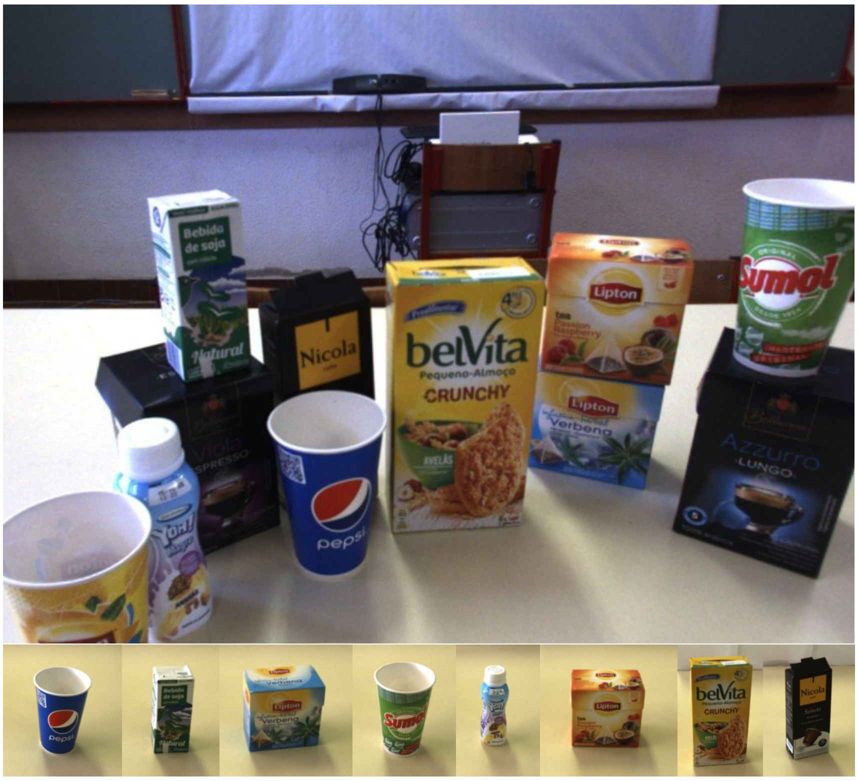


Figure 4.13: More examples of the robot recognizing objects. The recognized objects are shown below the image. Since the yellow cup is only partially visible in the bottom left corner of the image, it could not be recognized. The two black boxes were also not recognized because they generate only a very small number of keypoints, making it hard to recognize them.

keypoint in the "Lipton" logo of the wrong tea box. However, we stress that although the two tea boxes are very similar, the object recognition model was able to distinguish between them (see Fig. 4.6, bottom row). As can also be seen the descriptors were quite robust, allowing for matches even when the objects were slightly rotated, with different sizes and under different perspectives. However, according to our tests, to achieve good invariance for 3D objects, a template image for every 20° of the object is necessary. Figure 4.13 shows a few more examples of objects recognized during the tests.

Object tracking

In this test we used the same table top scenario and used several different objects. To illustrate the tracking behavior we modified the repulsive forcelet

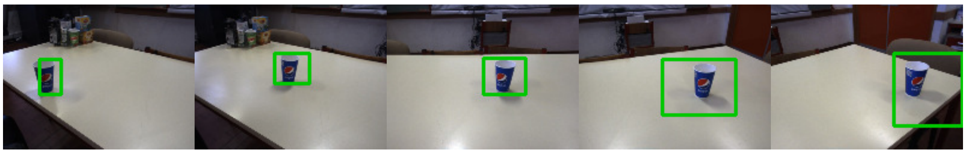


Figure 4.14: Sequence example of a tracked object.

to be active at every orientation except at the current orientation of the tracked object.

The robot was able to successfully track an object in its field of view. A few problems emerged due to the narrow FoV of the robot's camera: if we moved the object out of the camera's FoV it would lose the object and stop tracking it. However, this is a technical limitation of the hardware we used. Humans have a much wider FoV and use peripheral vision to aid in scene exploration and object tracking. Figure 4.14 shows a sequence of frames acquired by the robot when tracking a "predefined" object.

4.9 Discussion

In this chapter we presented a real-time biologically inspired active vision system to control a robot head. The vision part comprises a new model for stereo vision, a saliency module based on color, disparity and motion, and also object recognition using a state-of-the-art biological binary keypoint descriptor that we developed (see Chap. 3). The motor control component makes use of attractor dynamics to implement a behavior based on input from the vision component.

Throughout our tests we were able to verify that the developed stereo model proved to work fast and good enough for real-time robot operation. The single-scale version we used provided very rough disparity maps but they worked as desired when embedded in the active vision framework that we developed. Moreover, even though it was not our goal to make a state-of-the-art biologically-inspired stereo model, and although (so far) we could not make a direct comparison, the present model achieved results close to BSM, which was directly compared and clearly outperformed other state-of-the-art biological stereo models in Middlebury stereo evaluation version 2. Although this is a good indicator that our method might outperform other biologically-inspired stereo models, we can only claim to have a state-of-the-art biologically-inspired stereo model once we have the final testing results on the Middlebury dataset - version 3.

Regarding the saliency model, it also proved to work within our active vision framework, highlighting the salient objects at nearby positions, with distinct colors and moving objects. The use of motion as a saliency feature caused some minor problems in the generation of saliency maps, but only when the robot camera moved. These problems could be minimized by computing the global optical flow of the image and only considering motions in different directions than that. This would be fundamental for using this active vision system in a mobile robot.

The binary descriptor which we developed also worked very well. It was possible to accurately match keypoints between templates and salient regions in the captured images, leading to robust object recognition in our tests. We verified that the number of false positives was quite low, and many times those false positives were actually due to matching similar keypoints that ocurred in other areas of the image or in similar objects. Taking the results of our tests into account we consider that we have a good basis for the development of a more complex, higher level, object recognition system, for example by using the relative keypoint positions to validate the recognition of objects. One more option for future work would be to develop an automatic object learning system, so that the robot could learn new objects during runtime.

In terms of motor control the attractor dynamics approach also worked as desired both for scene exploration and for object tracking, with some minor issues in the latter case due to technical limitations of the used hardware (the small horizontal FoV of the camera). The approach proved to be very versatile since the behavior of the robot can be modified in a very easy way by manipulating the attraction and repulsion vectors. As future work in terms of motor control we could extend this approach to two dimensions, using it also for tilt control. Another option for future work would be to explore the use of 2D neural dynamic fields for motor control.

Although we were able to successfully test the active vision system, we realize that the processing time for each perception-action cycle was near the acceptable limit. Although the achieved processing frame rate may seem low, to the best of our knowledge it was the first time that a fully biologically-inspired active vision system was implemented and tested in real time on a real robot and in a real environment, which is not an easy task if we consider the heavy processing usually necessary for biological vision models. To use this active vision framework on a mobile robot, it would have to navigate very slowly.

As future work we could improve the stereo model by reducing the number of constants, having them computed during runtime in an optimized way. In addition, we could take advantage of the distributed computing capabilities of ROS and low-cost and low-power single-board computers that are getting cheaper and more powerful. Multiple independent modules could process every component of the vision module in parallel. Another possibility would be to develop GPU implementations of the vision components.

Chapter 5

A cognitive robot with biological vision for SLAM

Abstract: A major goal of cognitive robotics is to create a system which can adapt to dynamic environments and which can learn from its own experiences. We present a cognitive SLAM architecture, which is minimalistic in terms of sensors and memory. It employs a stereo camera with pan control, three memories, and odometry. Short-term memory is an egocentric map which holds information at close range at the actual robot position. Long-term memory is used for mapping the environment and the registration of encountered objects. Object memory holds features of learned objects which are used as navigation landmarks and task targets. Saliency maps are used to sequentially focus important areas for object and obstacle detection, but also for selecting directions of movements. Reinforcement learning is used to consolidate or to weaken environmental information in long-term memory. The system is able to achieve complex tasks by executing sequences of visuomotor actions, decisions being taken by goal-detection and goal-completion tasks. Experimental results show that the system is capable of executing tasks like localizing specific objects while building a map, after which it manages to return to the start position even when new obstacles have appeared.

5.1 Introduction

Autonomous mobile robots must be able to learn and interact with dynamic environments in which they navigate. Many explored approaches are based on traditional algorithms from artificial intelligence. They employ perceptions of the surrounding environment which are based on precise data from distance sensors (infrared, ultrasonic, laser) in combination with precise odometry for navigation. They may allow robots to perform simple tasks in controlled environments, but are less appropriate for highly dynamic and complex environments [Ratanaswasd et al., 2005]. The latter require flexible and adaptive systems, i.e., cognitive ones like our brain. However, also many lower animals like mice and crows are experts in mastering complex tasks in dynamic environments.

Egocentric navigation using a predefined map but without precise metrics can be achieved by a cognitive robot [Kawamura et al., 2002]. This robot featured a memory system split into short- and long-term ones, and it employed a simple vision system to detect colored tags which were used as references. However, it was not used for SLAM (simultaneous localization and mapping) because the map was predefined. Another robot [Meger et al., 2008] employed a more sophisticated vision system, combining saliency, object recognition and stereo vision along with a laser sensor for navigation and mapping. RatSLAM [Milford and Wyeth, 2010] is a navigation and mapping system which relies on goal-oriented navigation and on biologically inspired SLAM. Another biologically inspired navigation system consists of an implementation of a particular path-integration model, i.e., modified continuous-time recurrent neural networks [Papauschek and Zillich, 2010]. A cognitive architecture can also be based on biologically inspired multiple memory systems, involving episodic, semantic, procedural and working memory units [Kleinmann and Mertsching, 2011].

In this chapter we present a cognitive framework for robots composed of four interactive systems: vision, memory, SLAM and task management. The major contributions are: (a) the implemented SLAM system is directly integrated with the short- and long-term memories and it is affected by time, which allows the robot to adapt to dynamic environments by means of reinforcement learning; (b) the implemented task management and task building system is based on a hierarchy of only three basic actions; (c) we employ a stereo camera for vision, and no other sensors apart from odometry are used; (d) the robot features a complex vision system with disparity, saliency, Focus-of-Attention and object recognition; and (e) the robot head is steered by the complex vision system (active vision).

The rest of this chapter is organized as follows. In Section 2 we present the cognitive robot framework, including the vision, task manager and localization and mapping modules. In Section 3 we present results, and in Section 4 a final discussion.

5.2 Cognitive robot framework

Cognition generally refers to the faculty of mental activities of humans and certain animals dealing with abstractions of information from the real world, their representations in memory, as well as automatic recall [Patnaik, 2007]. Cognition may provide a solution to overcome the limitations of Artificial Intelligence (AI), allowing us to create cognitive robots that can execute new tasks by combining knowledge gathered in previous experiences and emotions [Ratanaswasd et al., 2005].

Through cognition, the human brain is able to acquire and process perceptions of the surrounding environment in a very fast and efficient way. Such perceptions consist of complex information captured using our sensory systems, notably our visual system which collects the most important information for navigation. In contrast to precise sensor systems in traditional robotics, our visual system allows us to recognize objects and to know whether they are near or far, but it does not provide us with very precise distances.

Another important feature of the human brain is the close connection between sensory systems and memory. This close connection also works as a filter: some of the information acquired remains in memory for a long time, whereas other information may be instantly discarded or kept for a brief time. This duality in information storage has been studied in cognitive psychology for many decades, which resulted in the assumption of two major types of memory: short-term memory (STM) and long-term memory (LTM) [Patnaik, 2007]. In this memory structure, only the most important information flows from STM to LTM, the importance of the information being affected by attention and concentration. Although most information may be discarded, the human brain can store a higher level of detail if necessory [Brady et al., 2008].

In the same way that the brain can select which information should be stored, the visual system can select which areas of the surrounding environment deserve most attention based on the saliency of those areas [Itti et al., 1998]. This information-filtering process is also very important because otherwise the brain would always be busy processing all the information gathered by our senses [Rensink, 2000]. After selecting which areas deserve most attention, the brain processes those areas for object recognition, comparing the objects seen with normalized templates previously stored in memory [Rodrigues and du Buf, 2009].

Another feature of human cognition is to plan and execute sequences of actions for accomplishing a goal, even sequences which have never been done before. This property is closely related to learning, because we can deduce new sequences by combining actions which we have done or seen before [Meinert, 2008].

The implementation of a cognitive system may lead to a new generation of robots which can adapt to dynamic environments and can interact with humans. There have already been some developments. The most relevant ones use vision as the main source of information, a memory system composed of STM and LTM, and an egocentric navigation system [Kawamura et al., 2002]. Stereo vision is also possible [Meger et al., 2008]. In addition to memory and visual systems which differ from those used in traditional robotics, a cognitive robot also needs SLAM capability, but one which, unlike traditional SLAM, is not based on large amounts of data from precise range sensors and precise odometry systems [Montemerlo et al., 2002].

In our previous work [Saleiro et al., 2012] we developed a minimalistic vision-based cognitive SLAM system that comprises a complex visual system and cognitive memory structures: short- and long-term memories. The visual system models processes like saliency, Focus-of-Attention (FoA), Inhibition-of-Return (IoR), and object recognition (all vision modules were based on computer vision algorithms, not bio-inspired). The system also featured a task management system. The system was successfully tested on a small robot platform (20 centimeter height).

In the following sections we describe a new version of the developed architecture. The complex visual system still features the mentioned visual processes, but all algorithms have been replaced by biologically inspired models. Motor control of the robot head and wheels has also been improved by using attractor dynamics. As before, the system features STM and LTM, which are closely integrated with the SLAM system.

5.2.1 Robotic platform

The cognitive system was implemented and tested using a child-sized Pioneer 3DX robot, equipped with a Bumblebee-2 stereo camera and a PhantomX turret for pan and tilt camera movement. The robot also features differential steering. All the processing is performed on a notebook computer mounted on the robot, and the robot was a completely independent system. The notebook features an Intel I7-4700HQ quad-core processor and 16GB of DDR3 RAM.

Regarding the movement of the head, we only used one tilt position which, due to the height of the cameras, allows for a good vertical field of view. Panning of the camera is done using an attractor dynamics approach that takes input from saliency maps (see Chapter 4).

The robot structure (see Fig. 5.1, top) has several platforms mounted in order to give it more height, and provide the point-of-view of a child with a height of 1.20 m. The robot has been set up with ROS (Robot Operating System). Although the robot is also equipped with a laser rangefinder and 6 ultrasonic sensors, they were not used.

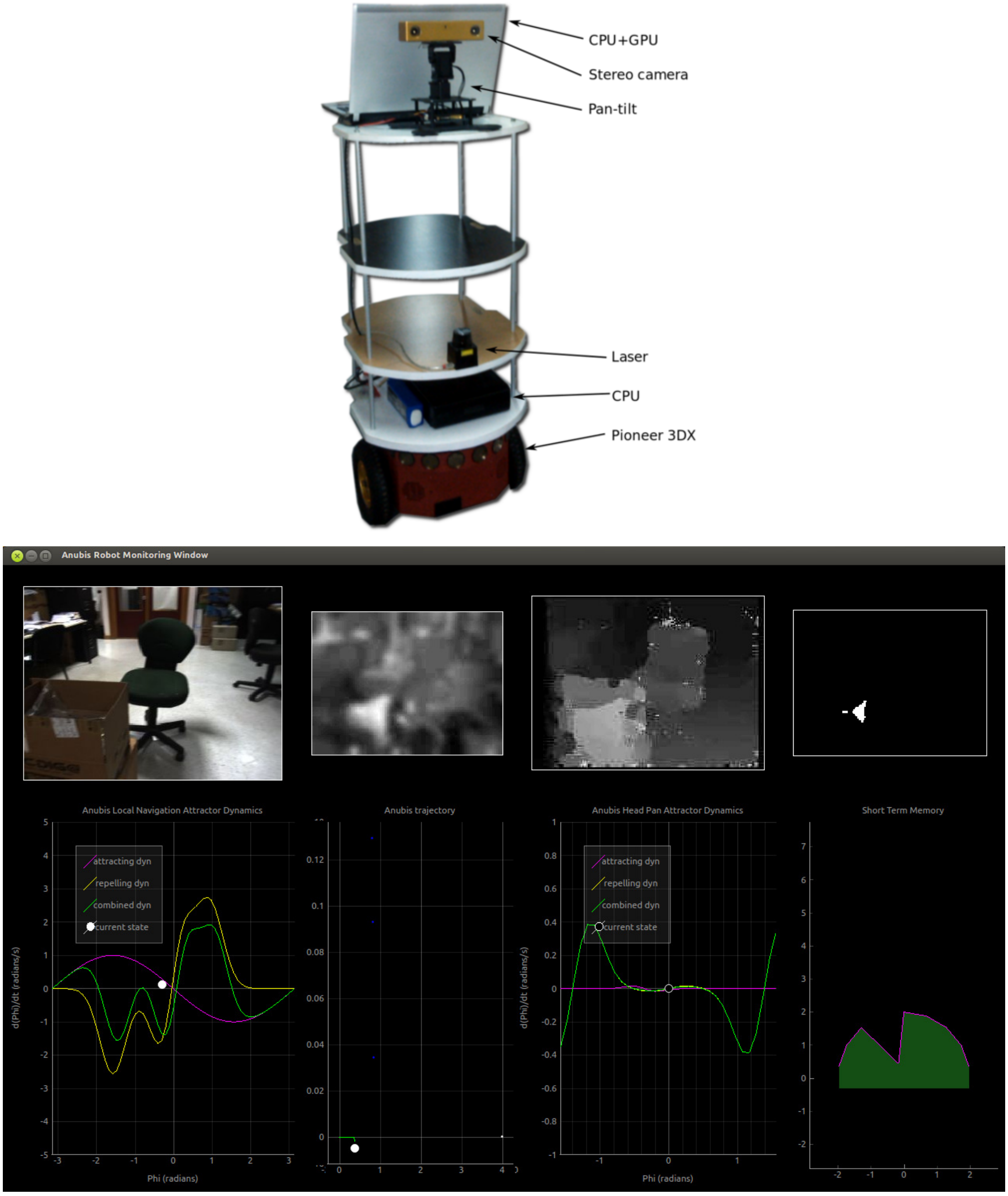


Figure 5.1: Child-sized robot used for the active vision tests (top) and image showing the robot monitoring panel (bottom). On the robot monitoring panel we can see, from top to bottom, left to right: captured left image, saliency map, disparity map, thresholded saliency map, navigation attractor dynamics, long-term memory, head pan attractor dynamics and short-term memory.

In terms of software the robot system is composed of several independent nodes, either for the vision processes or for the motor control modules which are interconnected using the ROS framework. For saliency we used the method described in Section 2.2.3 and for object recognition we used the keypoint descriptors and keypoint matching that we described in Chap. 3. For obstacle and limit detection we used the stereo algorithm described in Section 4.4. For motor control we used an attractor dynamics approach for differential steering that will be described in Section 5.4. In order to monitor all robot activity we also developed a robot monitoring tool, which shows the robot images, saliency map, disparity map, robot pan attractor dynamics, robot movement attractor dynamics, long-term memory map and short-term memory in real-time (see Fig. 5.1, bottom).

On average the robot achieved a processing rate of 2.4 frames per second. The complete system is going to be detailed in the following sections.

5.2.2 Active vision system

As previously mentioned, we used a complex biologically inspired visual system that we developed (see Chapter 4) for active vision that comprises: (a) stereo vision, (b) saliency, (c) object recognition and (d) attractor dynamics motor control. We will briefly describe each component of the active vision system.

The biologically-inspired stereo model (see Section 4.4) takes inspiration from the concept of binocular cells [Qian and Zhu, 1997; Scholl et al., 2013] tuned to different disparities and that, like other neurons in the visual cortex, can receive excitatory and inhibitory input from other neurons.

For stereo we start by computing responses from V1 cells at multiple scales and orientations and create a binary descriptor for every pixel position. Then we use these descriptors to perform matching on the same epipolar lines from left and right images.

A binary mask is created by calculating the absolute maximum difference of the three color components between every pixel of the left image and every pixel at every possible disparity from the right image in the same epipolar line. The maximum absolute difference is then binarized by using a threshold BT. The binary masks are then used to boost or decline the descriptor matching scores.

The result is a three-dimensional matrix with x, y and disparity dimensions. Afterwards we apply a DoG filter to the disparity dimension. When more than one scale is used, this is followed by a weighted sum with disparity costs from the previous scale.

Finally we apply an iterative spatial filtering processes, which consists in performing weighted sums of the matching costs in 3×3 windows until the number of changed pixel disparities is a smaller than $w\%$ (we used $w = 10\%$) of the total number of elements in the matching costs matrix. A WTA (Winner-Take-All) approach is used to select the best matching results in the 3D matrix.

For fastest computation of disparity maps during robot navigation, we used only one scale ($\lambda = 16$), which results in a small disparity map of only 160×120 pixels. Despite the small size it is big enough to be used as input for the saliency algorithm and also for obstacle detection. Figure 5.2 (top row) shows an example of a disparity map computed during robot navigation.

Instead of processing the entire images for object recognition we applied a saliency algorithm to select the image regions which may contain the most important information. Each selected region is then processed for object recognition. To this purpose the regions are delimited and sorted in decreasing order of saliency, which mimics sequential processing by Focus-of-Attention (FoA) with Inhibition-of-Return (IoR) to already analyzed regions.

The detection of salient regions is done using a modified version of the saliency algorithm that we developed (see Chap. 4.5). The original algorithm combined color, disparity and motion to create a saliency map, but in this application we removed the motion component since it would cause salient regions to be detected everywhere during robot navigation. To use the motion component, a modification would have to be made to the algorithm: the global optical flow vector would have to be subtracted from the individual local optical flow vectors (removing the egomotion).

For the color component we build stacks of 6 retinotopic maps representing different channels in the CIE LAB color opponent space. This model is based on retinal cones and provides a standard way to model color opponency. The first three channels code the image in LAB color space, thus representing white, green and blue color components. The remaining three channels are the complements of the first ones, representing black, red and yellow. Afterwards we apply a blob-detection kernel based on a Difference-of-Gaussians (DoG) operation.

For the disparity component we threshold the disparity map at 15% of the maximum disparity value to get only the nearest regions, and then apply the same blob-detection kernel as applied for the color component. For fast processing we apply the saliency algorithm to captured images downscaled to 160 × 120 pixels. Figure 5.2 (2nd row) shows an example of a saliency map captured during robot navigation.

As mentioned before, we use the descriptor that we developed (Chap. 3) to recognize objects. Like we humans store normalized images of many objects in memory [Rodrigues and du Buf, 2009], the robot also needs to memorize one or more views (templates) of the objects which will serve as navigation landmarks and task targets. Before the robot starts navigating, all stored templates are processed by our keypoint extraction algorithm and the keypoints are coded using our binary keypoint descriptor, creating the keypoint arrays for fast object matching during robot navigation. During navigation, when a region with high saliency captured by the left camera must be analyzed, the region's keypoint descriptor array is built by using the same process as used for the object/landmark templates. This array is matched against all stored object arrays using the Hamming distance, and if there are at least 5 matching keypoints with a score of at least 20% (25 out of 128 bits) an object is recognized. For an example of object recognition see Fig. 5.2 (top row on the right, with a rectangle around the object).

For the control of the robot head we tried to model typical human behavior, usually characterized by how we (and many animals) explore the field of view using eye saccades and only moving the head for changing the field of view to a "new" attraction point. To achieve this behavior we used an attractor dynamics approach (see Section 4.7) and used the "center" of salient regions as attraction points. Salient regions in the current camera field of view are analyzed for object recognition by order of maximum saliency (FoA) and do

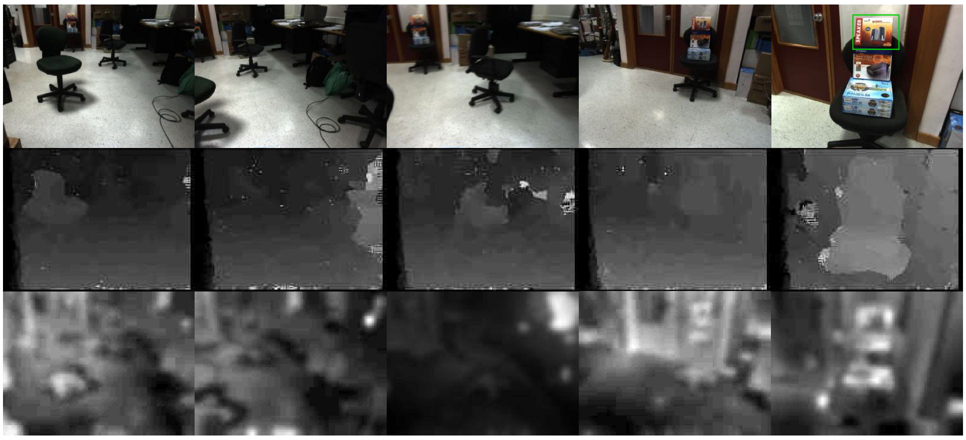


Figure 5.2: From top to bottom, captured images during robot navigation, disparity map and saliency map. Object recognition is shown in the first row on the right, being the recognized object marked with a rectangle around it.

not require any head movement. However, groups or individual salient points create attractors which make the head of the robot rotate to better fit (center) those salient regions into the camera's field of view. Multiple salient points are used to create an attractor forcelet.

To generate an exploring behavior we employ a time-variant repulsive forcelet, with its main repulsive point at the current pan orientation. As the strength of this repulsive forcelet increases with time, it forces the robot to switch to a different attraction point.

By summing both attractive and repulsive forcelets we obtain the resulting attractor forcelet from which we can extract the rotation angle to be applied to the panning motor. Every time the head rotates to a new attraction point, it gets into a stable position which will only be affected when the repulsive forcelet increases its strength. An example of attractor dynamics working during robot navigation can be seen in Fig. 5.6 (bottom row).

5.3 Memory and mapping

Building a map of the environment is fundamental for navigation and localization. As mentioned before, the cognitive robot architecture we developed uses three memories: short-term memory (STM), long-term memory (LTM) and object/landmark memory (ObjM). STM consists of a small map containing only information needed for immediate navigation. On the other hand, LTM consists of a global map containing information for global navigation. ObjM can be considered as part of LTM that contains object and landmark location information. We will now describe STM and LTM in a little more detail.

5.3.1 Short- and long-term memory

Since short-term memory is essentially used for immediate navigation, it only contains obstacle and wall information at the current robot position. This is used for obstacle avoidance. STM is an egocentric map that comprises the 180^o field of view and accumulates any obstacle information in that range. Any detected obstacles remain in STM if the robot moves a small distance, but they are removed as soon as they are beyond the 180^o range. Similarly to what was done in Section 4.7, the 180^o range is split into 9 radial sections.

To build the STM we start by thresholding the disparity map to get only the closest obstacles (20% of maximum disparity). We then split the thresholded disparity map into three vertical sections. In each of the sections we evaluate the maximum disparity and attribute that value to one of the 9 sections of the 180^o field of view, depending on the current head pan angle. The values attributed to the 9 sections are stored in an array that will be afterwards used by the attractor dynamics for differential motor control and also by the LTM to build a global map. See Fig. 5.3 (bottom right) for one example map of the STM.

Long-term memory (LTM) is a memory that stores all the information acquired by the robot during navigation. It can be seen as a global map, with lower resolution than STM. Detected obstacles at a current location are translated and rotated, according to the robot's global position and orientation, and stored in the global LTM map. Detected objects or landmarks are also stored in LTM. Whenever an obstacle, object or landmark is stored, an associated time-tag is also stored. Figure 5.3 (bottom left) shows one example map of the LTM.

LTM map is afffected by positive and negative reinforcements: (a) if an object is repeatedly detected at the same position, it is positively reinforced

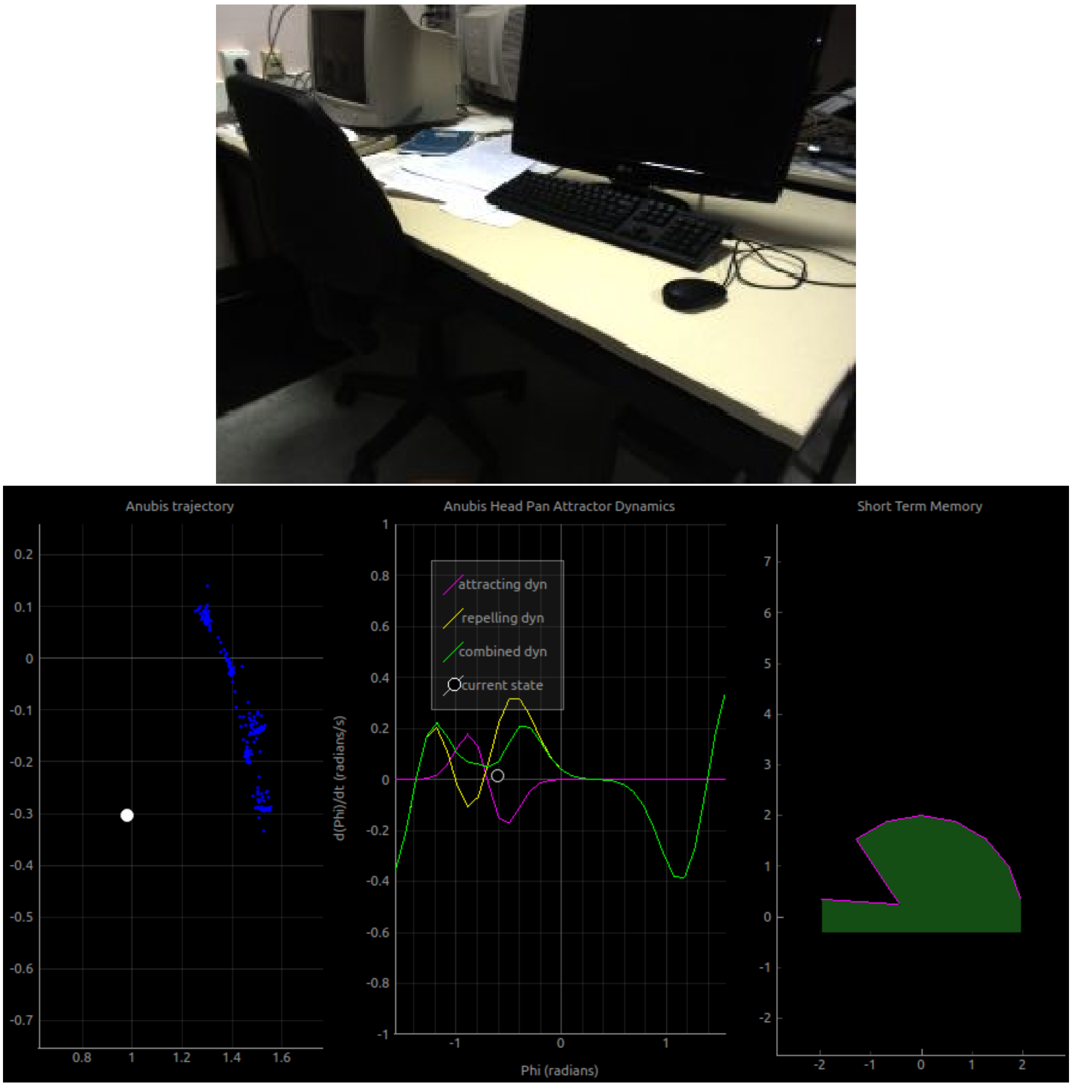


Figure 5.3: Top: captured image. Bottom, from left to right: LTM map, head pan attractor dynamics, STM. LTM shows the accumulated information that transitioned from STM. The black dot in the attractor dynamics plot shows that the robot is looking to the left, and explain why the STM shows a black "slice"(obstacle) on the left.

with a value P_R, until a maximum value M_R is reached; (b) if a previously detected object is not found at the same position, it is negatively reinforced with N_R; (c) there is also a negative reinforcement over time. Objects which have reached M_R are assumed to be fixed and stable, but all other ones are "feebled" by N_R after every time interval of t_{LTM} seconds. The latter process allows the robot to "forget" inconsistent information. In the actual setup we used $M_R = 200$, $P_R = 40$ $M_R = 200$, $N_R = -40$ and $t_{LTM} = 300$. Figure 5.4 shows an LTM map acquired while the robot was completing a SLAM task. Some of the blue dots (obstacles) are false obstacles which were registered due to noise caused by motion blur. However, since it is unlikely that they will be positively reinforced, after some time they end up disappearing.

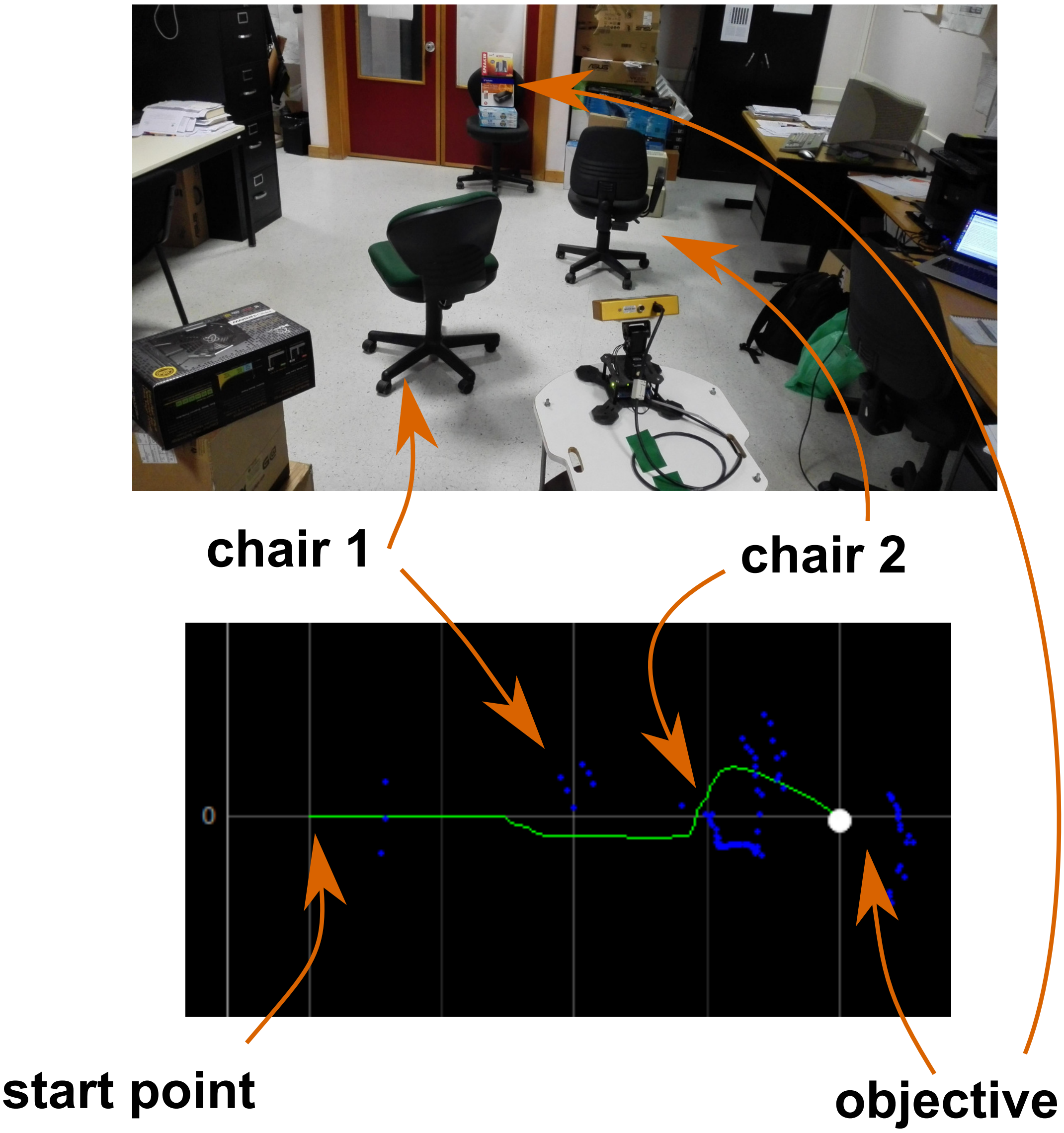


Figure 5.4: LTM map acquired while the robot was completing a SLAM task. The green line represents the robot's trajectory. The objective was to find the box shown in Fig.5.2.

5.3.2 Task management

As mentioned before, in our previous work [Saleiro et al., 2012] we designed a task management system that allows the robot to build and execute complex tasks from combining simpler ones. A central agent is responsible for building and organizing the tasks in order to accomplish a requested goal [Ratanaswasd et al., 2005; Alami et al., 2006]. Only two levels of tasks are considered to exist: micro-tasks, which are atomic actions like “go forward” and “turn right”, and macro-tasks, the latter are more complex. They are built from the aggregation of micro-tasks and simple macro-tasks. The simple macro-tasks were pre-programmed, but always associated with simple actions like “search” and “return home.” When the robot was given more than one general (top) task, it placed them in a task buffer for their sequential – but also concurrent – execution.

Each macro-task consisted of three blocks: (a) a visuomotor action cycle which consisted of movements to be made according to newly captured visual information; (b) the objective detection function that served to detect the actual, requested goal; and (c) the condition for task conclusion function to track whether a task had been completed or not.

The block structure is hierarchical: the three above blocks can each be split into sub-blocks, using the same block structure. This hierarchical and dynamic structure enables concurrent execution of multiple tasks. However, since there is always one main task, only one visuomotor cycle can be active at any time, during which all other visuomotor cycles remain inactive. Nevertheless, while one visuomotor cycle is being executed, the two detection functions related to other tasks are still evaluated. When a task is completed, its blocks are deactivated and the next task’s visuomotor cycle is activated. This process is repeated until all pending tasks are completed. Concurrent task processing is implemented by using three separate block buffers: one for visuomotor blocks, one for objective detection blocks, and one for objective conclusion condition blocks. The blocks in the last two buffers are executed sequentially within each visuomotor block which controls the robot. The visuomotor blocks are always the most complex ones, since the other two types only consist of simple detection functions. Figure 5.5 illustrates the task building process.

The main difference between our previous task management system and the current one lies in the abstraction level of micro tasks. Before, the sequence of motor control actions, either for robot movement or for head movement, were previously pre-programmed, one by one. Instead, the current robot system makes use of active vision with attractor dynamics in order to deal with this level of actions in a behavioral way, making it much more robust and versatile. This approach makes it unnecessary to specify every visuomotor action, and it also allows the robot to handle static and mobile obstacle avoidance effortlessly while maintaining the route to the target destination.

For the robot's navigation, we considered only two fundamental modes: (a) the exploration mode is used to navigate in an unknown environment, and (b) in the excursion mode the robot uses gathered information to move itself in a more efficient way.

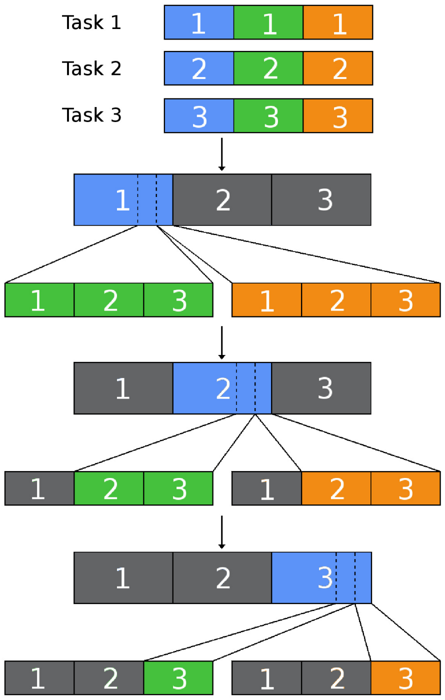


Figure 5.5: Three macro-tasks are concurrently executed by sequences of simple macro-tasks and micro-tasks. Blue: visuomotor blocks. Green: objective detection blocks. Orange: objective conclusion condition blocks. The grey blocks are inactive.

5.4 Attractor dynamics for differential motor control

For motor control we also use an attractor dynamics approach as done before by Bicho et al. [1998]. However, instead of using IR sensors or sonars we use STM information obtained from the disparity map.

As in Chapter 4, the approach requires attractor and repeller forcelets that are combined to create the resulting forcelet. However, in this case the attractor forcelet is computed differently. The rate of change of the attractor forcelet (Φa_t) towards the target destination is computed as follows:

$$\Phi a_t = -\alpha_a \times \sin(\phi_{robot} - \phi_{target}).$$

On the other hand the repelling forcelet is computed the same way as in Chapter 4, i.e., we split the 180^o egocentric field of view, ranging from -90^o to 90^o, into 20^o ranges, and store the orientation values of the 9 sections in an array $Ao = \{-80^o, -60^o, ..., 0, ..., 60^o, 80^o\}$. The maximum disparity values detected in each section, already stored in STM, are used to build another array Ar_t (repulsion vector). The repelling forcelet can then be calculated as follows:

$$\Phi r_t = \alpha r \times Ao \times e^{\frac{Ar_t}{\beta r}} \times e^{\frac{Ao^2}{\epsilon_r}},$$

with $\alpha a = 1, \alpha r = 25$, $\beta r = 40$ and $\epsilon_r = 0.34$ (all empirically determined).

By summing all the values of vector Φa_t at each frame acquisition (t) we get the attraction rate towards the strongest attraction point, and by summing all the values of Φr_t we get the repulsion rate away from the strongest repulsion point.

The final angular velocity of the robot, $\hat{\varphi}_t$, at each instance t, is then calculated by summing all Φa_t and Φr_t, i.e.,

$$\hat{\varphi}_t = \Sigma_9(\Phi a_t) + \Sigma_9(\Phi r_t),$$

with $\hat{\varphi}_t$ the value that is converted to the angular velocity to be applied to the robot base at instance t.

To get this angular velocity value to be applied to the robot, it is only necessary to multiply the attraction rate with a constant value, which depends on the specifications of the motor driver (in our case the value is 1 since the motor driver accepts angle commands in radians).

This approach will result in the robot navigating towards the goal. It will smoothly avoid any obstacles that may appear, since the rate of change away from the obstacles changes depending on the distance at which they are detected. Figure 5.6 shows a sequence of images of the attractor dynamics being used for robot navigation. In the bottom line we can see the evolution of the repulsive forcelet as the robot approaches the obstacle: as the robot gets closer to the obstacle the repulsive force increases (yellow curve), forcing the white dot (robot orientation) to move to the left in the plot, which corresponds to a rotation to the right.

5.5 Conclusions

We presented a cognitive robot architecture for SLAM which integrates biologically inspired active vision, dual memory and hierarchical task management. The main contributions are: (a) visual saliency is used for FoA and object recognition; (b) SLAM is directly integrated with short- and long-term memory and affected by time, allowing the robot to filter important information and adapt to changes in the environment; and (c) the task management system can build complex tasks from simpler ones.

Regarding vision, we verified that the use of saliency with attractor dynamics allows the robot to explore its surroundings in most situations. Since the robot moves, the attraction points are always changing, which means that the repulsive forcelet is always repelling different orientations, never getting strong enough to make the robot change its field of view. This causes the robot to keep looking at obstacles until it passes around them, sometimes missing other obstacles outside the field of view at that orientation. The stereo also proved to work well for wall and obstacle detection. Noise would occasionally appear in captured images that were blurred due to the robot's movement and panning

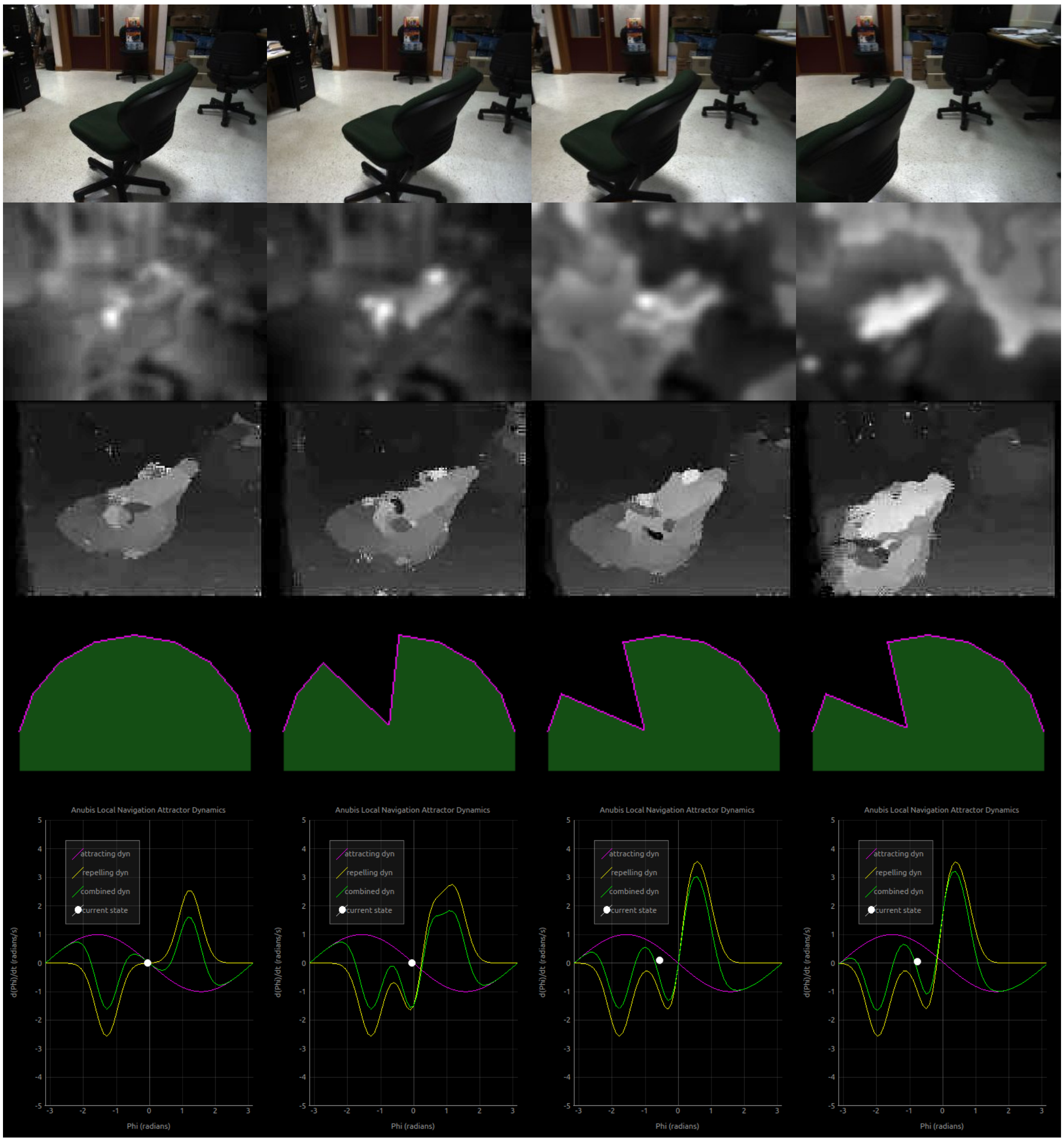


Figure 5.6: Sequence of images of the attractor dynamics being used for robot navigation. From top to bottom: captured left image, saliency map, disparity map, short-term memory, navigation attractor dynamics. In the bottom row it is possible to see the changing repulsive forcelet due to the obstacle. Is is also possible to see the white dot (current orientation) changing to the left, indicating that the robot is turning right.

movements. The keypoint descriptor also worked quite well for object recognition, showing only some problems in case of untextured objects. The use of these biological models in combination with motor control based on attractor dynamics results in a vision model which resembles the human visual system.

The use of attractor dynamics in navigation also proved to work as expected, allowing the robot to navigate from it's current location to a target destination.

Although we were able to verify that the system can work, we must say that much further testing is required with different scenarios. We can also say that there is much room for improvement in multiple fields. Regarding the hardware field, a camera with a wider field of view would be very useful, allowing the robot to capture more information without needing to be constantly panning the head, which in conjunction with the robot's movements caused blurring of the images. In addition, we could take advantage of the distributed computing capabilities of ROS and low-cost and low-power single-board computers that are getting cheaper and more powerful. Multiple independent modules could process every component of the vision module in parallel. Another possibility would be to develop GPU implementations of the vision components. By speeding up the processing we could also increase frame rate and decrease the blurring of the images caused by the robot's movements. By increasing frame rate we could also increase the speed of the robot during navigation.

Regarding the system itself, there is still much room for tuning the vision modules and the attractor dynamics motor controllers. There is also room for studying ways of manipulating the attractor and repulsive forcelets to create different behaviors.

Chapter 6

Concluding Remarks

Abstract: This chapter summarizes the previous chapters, main conclusions and achievements and presents directions for further research.

6.1 Summary

Chapter 1 introduced the scope of the thesis along with some brief explanations on the differences between Computer Vision (CV) and Human Vision (HV). It briefly described the main research topics, offering a brief overview of each one and of the cortical aspects of HV and their own very different purposes.

In Chapter 2 we showed an application of the biologically inspired vision processes to human-robot interaction. We used a child-sized robot platform with an initial version of our biological stereo algorithm presented in Chapter 4 to detect obstacles, and when an obstacle is detected the robot lifts its head in order to see if the obstacle is a human. To detect if an obstacle is a human it uses a linear SVM. Once a human is detected, it uses a saliency algorithm based on color, motion and disparity to detect important image regions, and it applies another SVM for hand detection in those regions. When the hands are detected, the robot then uses cortical keypoints and an initial version of the keypoint descriptors we presented in Chapter 3 to match gestures with previously stored gesture templates, allowing the human to interact with the robot, giving it orders or feedback.

Chapter 3 presented the process of learning a robust and fast binary keypoint descriptor using cell responses from cortical area V1. We described the extensive tests that we performed to optimize the descriptor, a step-by-step analysis of the results showing the influence of the type of cell response, scale, pooling step, pooling region size and the number of features on the performance. After describing the whole evaluation and training process, we compare it to other biologically inspired binary descriptors and also with a non-biological keypoint descriptor. We showed some applications, arguing that it can be used in some early vision processes such as stereo vision, optical flow and object recognition. We showed that by simply doing basic keypoint matching, without any pre- or post-processing, we can obtain very acceptable results.

Chapter 4 presented a biologically-inspired active vision system which integrates processes like disparity, saliency, Focus-of-Attention, Inhibition-of-Return and object recognition with motor control based on attractor dynamics. The disparity model was specifically designed to be fast to allow for real-time robot operation. Although the generated disparity maps are not very detailed, they are good enough for robot navigation and to be used as input to the saliency algorithm that is also based on color and motion saliency. We also described some tests that we have done using the same robot platform used as used in Chapter 2, in front of a table-top scenario where it successfully explored salient regions in its field of view, performed object tracking and recognized most of the objects using the descriptor presented in Chapter 3.

Chapter 5 focused on a cognitive robot architecture that integrates the active vision model from Chapter 4 with a human memory model comprising STM and LTM for SLAM. The cognitive system also features a task management unit, which allows the robot to build complex tasks from simpler ones. Attractor dynamics are used to control the motors of the robot base.

6.2 Integrated architecture

This thesis dealt with several modules that can be integrated into a single architecture for active vision in robot cognition. In terms of vision, a robot can use the saliency algorithm to generate saliency maps which are then used to select important regions of the visual field that must be processed for object recognition. Using saliency in combination with attractor dynamics, motor control can direct the camera towards interesting regions, acting according to the visual *stimuli* that it receives. Once it focuses on important regions, the robot can proceed to object recognition in order to attempt to identify what it is looking at. The recognition of objects depends on the dataset of objects that is previously given to the robot. The stereo algorithm is also part of the global architecture and is used as input for the saliency algorithm and for obstacle detection. The detected obstacles are then stored for a short period in STM, which is used for local navigation. Afterwards the detected obstacles and objects transition from STM to LTM, where they will remain or fade with time if their existence is not often confirmed. Items available in long-term memory can be used for global navigation and path and task planning. This architecture is very different from the usual non-biological robotics architectures, which make use of many types of different sensors and which store huge amounts of data.

6.3 Achievements

There are four major achievement categories. These are summarized in this section and they are the basis for future research that will be presented in the next section. Namely:

- **Biologically inspired vision for human robot interaction**. We managed to create a robotic system for human robot interaction that makes use of vision processes based on cortical cells. This achievement is important because it validated the use of biological processes in real-time robotics applications.

- **Learning biologically inspired binary descriptors**. Throughout the process of learning biologically-inspired binary descriptors we were able to create keypoint descriptors that are able to outperform any other biologically-inspired local descriptors, even when using 4 times less bits than the top binary descriptor. Moreover, some of the tested descriptors are also able to outperform the non-biological SIFT-based LDAHash descriptor. By combining our descriptor with our keypoint detector, we have the first biologically inspired integrated keypoint detection and description system, completely based on cortical cell responses from area V1. Moreover, we were able to successfully use the integrated system in real time for robotic vision.

- **Biologically inspired stereo vision for robot navigation**. We were able to create a fast, real-time stereo vision algorithm based on cortical cell responses, and used it for robot navigation and as input to a visual saliency model.

- **Active vision framework for robotics**. The integration of several visual processes with attractor dynamics motor control allowed us to create an active vision framework that enables a stimulus-guided exploration of the robot's field of view.

- **Cognitive robot architecture with integrated active vision for SLAM**. We managed to integrate the active vision framework with a cognitive framework that comprises a human memory model for SLAM and task realization.

6.4 Final considerations for future research

The work of this thesis took about 5 years. It yielded significant achievements in evolving models of human vision processes and combining them into an integrated architecture that is fast enough to run in real time and be used in robotics applications. To the best of our knowledge, this was done for the first time. In general, although we were able to successfully integrate the system for robot vision and robot navigation, we feel that we have just scratched the surface of what can be done using biological vision models and how much they can be improved. Regarding the binary keypoint descriptors, we would like to perform even more extensive tests using many more scales. Since the brain is a massively parallel machine, analyzing visual *stimuli* at many different scales simultaneously, and by looking at the results of the descriptor learning process, we suspect that by using cell responses extracted at many more different scales it may be possible to achieve even better results and maybe compete with other non-biological state-of-the-art descriptors. Another goal we would like to achieve in the future would be to integrate the keypoint descriptor code with the GPU implementation of the keypoint detector, creating a fast unified system for cortical keypoint detection and description.

Regarding stereo, we believe that there is still room for improvement by experimenting with combinations of different cells at different scales and using both top-down and bottom-up propagation of disparity information between different scales. Concerning saliency, there is also a lot of room for improvements: new features like texture, shape or symmetry could lead to better saliency maps; to create saliency models that can be tuned to a certain goal could also be very interesting.

On the topic of object recognition there is also space for improvement. The development of a hierarchical coarse-to-fine approach for object recognition could be very interesting for robotics applications, speeding up the process when using large object libraries: each object in the library would be described by a tree of keypoints described at multiple scales and the robot, by using a coarse-to-fine approach would be able to progressively reduce the number of possible matches, using finer keypoints as the number of possible matches is reduced. Regarding the whole robotic architecture, it would be interesting to integrate the entire visual system with more complex neural-dynamics approaches for head pan and tilt, for robot steering and task learning.

Since our knowledge of how the human brain works is still so limited, each process that we try to model and understand leads to a new set of interesting possibilities that must be experimented with and evaluated.

Bibliography

Al-Absi, H., Abdullah, A., 2009. A proposed biologically inspired model for object recognition. Proc. 1st Int. Visual Informatics Conf.: Bridging research and practice 5857, 213–222.

Alahi, A., Ortiz, R., Vandergheynst, P., 2012. FREAK: Fast retina keypoint. Proc. Int. Conf. Computer Vision and Pattern Recognition (CVPR), 510–517.

Alami, R., Clodic, A., Montreuil, V., Sisbot, E. A., Chatila, R., 2006. Toward human-aware robot task planning. Association for the Advancement of Artificial Intelligence Spring Symposia (AAAI), 8pp.

Aloimonos, J., Weiss, I., 1988. Active vision. Int. J. of Computer Vision 1 (4), 333–356.

Astua, C., Barber, R., Crespo, J., Jardon, A., 2014. Object detection techniques applied on mobile robot semantic navigation. Sensors 14 (4), 6734–6757.

Ball, D., Heath, S., Wiles, S., Wyeth, G., Corke, P., Milford, M., 2013. Openratslam: an open source brain-based slam system. Autonomous Robots 34 (3), 149–176.

Bar, M., Kassam, K., Ghuman, A., Boshyan, J., Schmid, A., Dale, A., Hämäläinen, M. S., Marinkovic, K., Schacter, D., Rosen, B., Halgren, E., 2006. Top-down facilitation of visual recognition. Proc. 1st Int. Visual Informatics Conf.: Bridging Research and Practice 103 (2), 449–454.

Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., Friston, K. J., 2012. Canonical microcircuits for predictive coding. Neuron 76 (4), 695–711.

Bay, H., Tuytelaars, T., van Gool, L., 2008. SURF: Speeded-up robust features. Computer Vision and Image Understanding 110 (3), 346–359.

Bicho, E., Mallet, P., Schoner, G., 1998. Using attractor dynamics to control autonomous vehicle motion. Proc. 24th Annual Conf. IEEE Electronics Society, 1176–1182.

Biswas, J., Veloso, M., 2012. Depth camera based indoor mobile robot localization and navigation. In: 2012 IEEE Int. Conf. Robotics and Automation (ICRA). pp. 1697–1702.

Brady, T., Konkle, T., Alvarez, G., Oliva, A., 2008. Visual long-term memory has a massive storage capacity for object details. Proc. Nat. Acad. Scie. 105 (38), 14325–14329.

Brown, M., 2011. Learning local image descriptors data. [Online; accessed 17-Feb-2015].
URL `http://www.cs.ubc.ca/~mbrown/patchdata/patchdata.html`

Brown, M., Hua, G., Winder, S., 2011. Discriminative learning of local image descriptors. IEEE Trans. on Pattern Analysis and Machine Intelligence 33 (1), 43–57.

Buschka, P., Saffioti, A., 1998. Some notes on the use of hybrid maps for mobile robots. Proc. 8th Int. Conf. Intelligent Autonomous Systems, 547–556.

Butko, N., Zhang, L., Cottrell, G., Movellan, J., 2008. Visual salience model for robot cameras. Proc. 2008 IEEE. Int. Conf. on Rob. and Automation, 2398–2403.

Calonder, M., Lepetit, V., Özuysal, M., Trzcinski, T., Strecha, C., Fua, P., 2012. BRIEF: Computing a local binary descriptor very fast. IEEE Trans. on Pattern Analysis and Machine Intelligence 34 (7), 1281–1298.

Chandrasekhar, V., Takacs, G., Chen, D., Tsai, S., Grzeszczuk, R., Girod, B., 2009. CHoG: Compressed histogram of gradients a low bit-rate feature descriptor. Proc. Int. Conf. Computer Vision and Pattern Recognition (CVPR), 2504–2511.

Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection. Proc. Int. Conf. Computer Vision and Pattern Recognition (CVPR), 886–893.

Davison, A., Reid, I., Molton, N., Stasse, O., 2007. MonoSLAM: Real-time single camera SLAM. Proc. IEEE Int. Conf. on Robotics and Automation, 688–694.

Deco, G., Rolls, E. T., 2004. A neurodynamical cortical model of visual attention and invariant object recognition. Vision Res. 44 (6), 621–642.

Deco, G., Rolls, E. T., 2005. Attention, short-term memory, and action selection: A unifying theory. Prog. in Neurobiol. 76 (4), 236–256.

Dellaert, F., Stroupe, A. W., 2007. Linear 2D localization and mapping for single and multiple robot scenarios. Proc. IEEE Int. Conf. on Robotics and Automation 1 (6), 1052–1067.

DeValois, R. L., Albrecht, D. G., Thorell, I. G., 1982. Spatial frequency selectivity of cells in macaque visual cortex. Vision Research 22, 545–559.

Dmitriy, Y., Postolsky, A., Grigoriy, P., Gennadievich, G., 2013. Mobile robots navigation based on artificial landmarks with machine vision system. World Applied Sciences Journal, 1467–1472.

du Buf, J., 1993. Responses of simple cells: events, interferences, and ambiguities. Biol. Cybern. 68, 321–333.

du Buf, J., Barroso, J., Rodrigues, J., Paredes, H., Farrajota, M., Fernandes, H., José, J., Teixiera, V., Saleiro, M., 2010. The smartvision navigation prototype for the blind. Proc. Int. Conf. on Software Development for Enhancing Accessibility and Fighting Info-exclusion, 167–174.

Evans, C., 2009. Notes on the OpenSURF Library. Tech. Rep. CSTR-09-001. University of Bristol. URL http://www.chrisevansdev.com.

Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C., 2008. LIBLINEAR: A library for large linear classification. J. of Machine Learning Research 9, 1871–1874.

Farneback, G., 2003. Two-frame motion estimation based on polynomial expansion. Proc. 13th. Scandinavian Conf. on Image Analysis 2749, 363–370.

Farrajota, M., Martins, J., Rodrigues, J., du Buf, J., 2011a. Disparity energy model with keypoint disparity validation. Proc. 17th Portuguese Conf. on Pattern Recognition, 2pp.

Farrajota, M., Rodrigues, J. M. F., du Buf, J. M. H., 2011b. Optical flow by multiscale annotated keypoints: a biological approach. Proc. Int. Conf. on Bio-inspired Systems and Signal Processing (BIOSIGNALS), 307–315.

Findlay, J. M., Gilchrist, I. D., 2003. Active Vision, The Psychology of Looking and seeing. Oxford University Press.

Forssén, P., Meger, D., Lai, K., Helmer, S., Little, J., Lowe, D., 2008. Informed visual search: Combinning attention and object recognition. IEEE Proc. Int. Conf. Robotics Automation, 935–942.

Frintrop, S., Nuchter, A., Surmann, H., Hertzberg, J., 2014. Saliency-based object recognition in 3D data. Proc. IEEE Int. Conf. on Intelligent Robots and Systems 3, 2167–2172.

Fukushima, K., 2003. Neocognitron for handwritten digit recognition. Neurocomputing 51, 161–180.

Ghaleb, F., Youness, E., Elmezain, M., Dewdar, F., 2014. Hand gesture spotting and recognition in stereo color image sequences based on generative models. Int. J. of Engineering Science and Innovative Technology 3 (1), 78–88.

Gong, Y., Kumar, S., Rowley, H. A., Lazebnik, S., 2013a. Learning binary codes for high-dimensional data using bilinear projections. Proc. Int. Conf. Computer Vision and Pattern Recognition (CVPR), 484–491.

Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F., 2013b. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. IEEE Trans. on Pattern Analysis and Machine Intelligence 35 (12), 2916–2929.

Gould, S., Arfvidsson, J., Kaehler, A., Sapp, B., Messner, M., 2007. Peripheral-foveal vision for real-time object recognition and tracking in video. Proc. 20th Int. Joint Conf. on Artificial Intelligence, 2115–2121.

Han, J., Gold, N., 2014. Lessons learned in exploring the leap-motion (TM) sensor for gesture-based instrument design. Proc. New Interfaces for Musical Expression, 371–374.

Holzbach, A., Cheng, G., 2014a. A concurrent real-time biologically-inspired visual object recognition system. Proc. IEEE Int. Conf. on Robotics and Automation, 3201–3206.

Holzbach, A., Cheng, G., 2014b. A fast and scalable system for visual attention, object based attention and object recognition for humanoid robots. Proc. IEEE Int. Conf. on Humanoid Robots, 316–321.

Hossain, M., Rashid, M., Bhuiyan, M., Ahmed, S., Akhtaruzzaman, 2013. A qualitative approach to mobile robot navigation using RFID. IOP Conf. Series: Materials Science and Engineering 53 (1), 012064.

Hubel, D. H., 1995. Eye, brain, and vision, 2nd Edition. Scientific America Library.

Ionescu, B., Coquin, D., Lambert, P., Buzuloiu, V., 2005. Dynamic hand gesture recognition using the skeleton of the hand. EURASIP J. of Applied Signal Processing (13), 2101–2109.

Itti, L., Koch, C., 2001. Computational modeling of visual attention. Nature Reviews: Neuroscience 2 (3), 194–203.

Itti, L., Koch, C., Niebur, E., 1998. A model of saliency-based visual attention for rapid scene analysis. IEEE Trans. on Patt. Recogn. and Mach. Intell. 20 (11), 1254–1259.

Iwamoto, K., Mase, R., Nomura, T., 2013. Bright: A scalable and compcact binary descriptor for low-latency and high-accuracy object identification. Int. Conf on Image Processing, 2915–2919.

Jain, P., Kulis, B., Davis, J. V., Dhillon, I. S., 2012. Metric and kernel learning using a linear transformation. J. of Machine Learning Research 13, 519–547.

José, J., Farrajota, M., Rodrigues, J., du Buf, J., 2010. A vision system for detecting paths and moving ostacles for the blind. Proc. Int. Conf. on Software Development for Enhancing Accessibility and Fighting Info-exclusion, 175–182.

Jung, Y., Choi, Y., Park, H., Shin, W., Myaeng, S.-H., 2007. Integrating robot task scripts with a cognitive architecture for cognitive human-robot interactions. Proc. IEEE Int. Conf. on Information Reuse and Integration, 152–157.

Katsuki, F., Constantinidis, C., 2014. Bottom-up and top-down attention: different processes and overlapping neural systems. Neuroscientist 20 (5), 509–521.

Kawamura, K., Koku, A., Wilkes, D., Peters II, R., Sekmen, A., 2002. Toward egocentric navigation. Int. J. Robotics and Automation 17 (4), 135–145.

Ke, Y., Sukthankar, R., 2004. PCA-SIFT: a more distinctive representation for local image descriptors. Proc. Int. Conf. Computer Vision and Pattern Recognition (CVPR), 506–513.

Kim, K.-R., Kim, C.-S., 2016. Adaptive smoothness constraints for efficient stereo matching using texture and edge information. Proc. Int. Conf. Image Proc., 3429–3433.

Kleinmann, L., Mertsching, B., 2011. Learning to adapt: Cognitive architecture based on biologically inspired memory. In: 6th IEEE Conf. on Industrial Electronics and Applications (ICIEA). pp. 936 –941.

Kronander, K., Billard, A., 2014. Learning compliant manipulation through kinesthetic and tactile human-robot interaction. IEEE Tr. on Haptics 7 (3), 367–380.

Kumar, P., Verma, J., Prasad, S., 2012. Hand data glove: a wearable real-time device for human-computer interaction. Int. J. of Advanced Science and Technology 43, 15–25.

Kurzweil, R., 2006. The Singularity Is Near: When Humans Transcend Biology. Penguin (Non-Classics).

Larkum, M., 2013. A cellular mechanism for cortical associations: an organizing principle for the cerebral cortex. Trends in Neurosciences 36 (3), 141–151.

Leitner, J., Harding, S., Chandrashekhariah, P., Frank, M., Frster, A., Triesch, J., Schmidhuber, J., 2013. Learning visual object detection and localisation using icVision. Biologically Inspired Cognitive Architectures 5, 29 – 41.

Leutenegger, S., Chli, M., Siegwart, R. Y., 2011. BRISK: Binary robust invariant scalable keypoints. Proc. Int. Conf. Computer Vision (ICCV), 2548–2555.

Liu, W., Wang, J., Ji, R., Jiang, Y. G., Chang, S. F., 2012. Supervised hashing with kernels. Proc. Int. Conf. Computer Vision and Pattern Recognition (CVPR), 2074–2081.

Lowe, D., 1999. Object recognition from local scale-invariant features. Int. Conf. Comp. Vis. 1150–1157.

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. Int. J. of Computer Vision 60, 91–110.

Martins, J., Rodrigues, J., du Buf, J., 2012. Local object gist: meaningful shapes and spatial layout at a very early stage of visual processing. Gestalt Theory 34 (3/4), 349–380.

Martins, J. A., Rodrigues, J., du Buf, J. M. H., 2008. Region segregation and saliency using colour information. Proc. 14th Portuguese Conf. Pattern Recognition, 2pp.

Martins, J. A., Rodrigues, J., du Buf, J. M. H., 2009. Focus of attention and region segregation by low-level geometry. Proc. Int. Conf. on Computer Vision Theory and Applications (VISAPP) 2, 267–272.

Martins, J. A., Rodrigues, J., du Buf, J. M. H., 2011. Disparity energy model using a trained neuronal population. Proc. IEEE Int. Symp. on Signal Proc. and Info. Technology, 287–292.

Martins, J. A., Rodrigues, J., du Buf, J. M. H., 2015. Luminance, colour, viewpoint and border enhanced disparity energy model. PLos One 10 (6), 24pp.

Masciocchi, C. M., Mihalas, S., Parkhurst, D., Niebur, E., 2009. Everyone knows what is interesting: Salient locations which should be fixated. J. of Vision. 9 (11), 1–22.

Meger, D., Forssén, P., Lai, K., Helmer, S., McCann, S., Southey, T., Baumann, M., Little, J. J., Lowe, D. G., 2008. Curious George: An attentive semantic robot. Rob. Aut. Sys. 56 (6), 503–511.

Meinert, P., 2008. The impact of previous life experience on cognitive structure changes and knowledge acquisition of nursing theory and clinical skills in nontraditional nursing students. PhD Thesis, Kent State Univ., College of Education, Health and Human Services, USA, 173.

Messias, J., Ventura, R., Lima, P., Sequeira, J., Alvito, P., Marques, C., Carriço, P., 2014. A robotic platform for edutainment activities in a pediatric hospital. In Proc. IEEE Int. Conf. Autonomous Robot Systems and Competitions, 193–198.

Milford, M., Wyeth, G., August 2010. Persistent navigation and mapping using a biologically inspired slam system. Int. J. Robotics Res. 29 (9), 1131–1153.

Mittal, A., Zisserman, A., Torr, P. H., 2011. Hand detection using multiple proposals. In: BMVC. Citeseer, pp. 1–11.

Monteiro, S., Vaz, M., Bicho, E., 2004. Attractor dynamics generates robot formations: from theory to implementation. Proc. IEEE Int Conf. on Robotics and Automation, 2582–2587.

Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., 2002. FastSLAM: A factored solution to the simultaneous localization and mapping problem. Proc. AAAI. Nat. Conf. Art. Int., 593–598.

Morton, R., Olson, E., 2013. Robust sensor characterization via max-mixture models: GPS sensors. Proc. Int. Conf. Intelligent Robots and Systems, 528–533.

Mumford, D., 1992. On the computational architecture of the neocortex. Biol. Cybern. 66 (3), 241–251.

Mumford, D., Lamme, V. A. F., 1998. The role of primary visual cortex in higher level vision. Vision Research 38 (15-16), 2429–2454.

Mutch, J., 2011. HMIN: A minimal HMAX implementation. Online; accessed 14-Jul-2015.
URL `http://cbcl.mit.edu/jmutch/hmin/`

Mutch, J., 2012. HMAX package for CNS. Online; accessed 14-Jul-2015.
URL `http://cbcl.mit.edu/jmutch/cns/hmax/doc/`

Ng, H., Winkler, S., 2014. A data-driven approach to cleaning large face datasets. Proc. IEEE Int. Conf. on Image Processing (ICIP), 343–347.

Nistér, D., Stewénius, H., 2006. Scalable recognition with a vocabulary tree. Proc. IEEE. Comp. Soc. Conf. on Comp. Vision and Patt. Rec. 2, 2161–2168.

Ohali, Y., 2011. Computer vision based date fruit grading system: design and implementation. J. of King Saud Univ. - Comp. and Inf. Sciences (23), 29–36.

Oliva, A., Torralba, A., 2006. Building the gist of a scene: the role of global image features in recognition. Progress in Brain Res.: Visual Perception 155, 23–26.

Oliver, A., Kang, S., Wünsche, B. C., MacDonald, B., 2012. Using the Kinect as a navigation sensor for mobile robotics. In: Proc. 27th Conf. on Image and Vision Computing New Zealand. pp. 509–514.

Orban, G., Lagae, L., Verri, A., Raiguel, S., Xiao, D., Maes, H., Torre, V., 1992. First-order analysis of optical flow in monkey brain. PNAS 89 (7), 2595–2599.

Papauschek, C., Zillich, M., 2010. Biologically inspired navigation on a mobile robot. In: IEEE Int. Conf. Robotics and Biomimetics. pp. 519 –524.

Patnaik, S., 2007. Robot Cognition and Navigation: An Experiment with Mobile Robots, 1st Edition. Springer.

Peng, S., Dong, W., 2012. Robot navigation system with RFID and sensors. Int. Conf. on Computer Distributed Control and Intelligent Environmental Monitoring, 610–612.

Poggio, G. F., 1990. Cortical mechanisms of stereopsis studied with dynamic random-dot stereograms. Cold Spring Harb. Symp. Quant. Biology 55, 749–758.

Poggio, G. F., Fischer, B., 1977. Binocular interaction and depth sensitivity of striate and prestriate cortex of behaving rhesus monkey. Journal of Neurophysiology 40, 1392–1405.

Poggio, G. F., Gonzalez, F., Krause, F., 1988. Stereoscopic mechanisms in monkey visual cortex: binocular correlation and disparity selectivity. Journal of Neuroscience 8, 4531–4550.

Posner, M. I., Cohen, Y., 1984. Components of visual orienting. Attention and performance X: Control of language processes 32, 531–556.

Posner, M. I., Rafal, R. D., Choate, L., Vaughn, J., 1995. Inhibition of return: Neural basis and function. Cognitive Neuropsychology 2 (3), 211–228.

Purves, D., Augustine, G. J., Fitzpatrick, D., Hall, W. C., LaMantia, A., White, L. E. (Eds.), 2012. Neuroscience. Sinauer Associates.

Pyo, Y., Hasegawa, T., Tsuji, T., Kurazume, R., Morooka, K., 2014. Floor sensing system using laser reflectivity for localizing everyday objects and robot. Sensors 14 (4), 7524–7540.

Qian, K., Niu, J., Yang, H., 2013. Developing a gesture based remote human-robot interaction system using Kinect. Int. J. of Smart Home 7 (4), 203–208.

Qian, N., Zhu, Y., 1997. Physiological computation of binocular disparity. Vision Research 37 (13), 1811–1827.

Ramisa, A., Vasudevan, S., Scharamuzza, D., de Mántaras, R. L., 2008. A tale of two object recognition methods for mobile robots. Proc. 6th Int. Conf. Comp. Vision Systems, Springer LNCS 5008, 353–362.

Ratanaswasd, P., Gordon, S., Dodd, W., 2005. Cognitive control for robot task execution. Proc. IEEE Int. Worksh. on Robot and Human Interactive Communication (RO-MAN) (5), 440–445.

Rensink, R., 2000. The dynamic representation of scenes. Visual Cogn. 7 (1-3), 17–42.

Rodrigues, J., 2008. Integrated multi-scale architecture of the cortex with application to computer vision. Doctoral Thesis, University of the Algarve, Portugal, 156pp.

Rodrigues, J., du Buf, J. M. H., 2006. Multi-scale keypoints in V1 and beyond: object segregation, scale selection, saliency maps and face detection. BioSystems 2, 75–90.

Rodrigues, J., du Buf, J. M. H., 2009. Multi-scale lines and edges in V1 and beyond: brightness, object categorization and recognition, and consciousness. BioSystems 95, 206–226.

Rodrigues, J., du Buf, J. M. H., 2009a. A cortical framework for invariant obect categorization and recognition. Cogn. Proc. 10 (3), 243–261.

Rodrigues, J. M. F., Martins, J. A., Lam, R., du Buf, J. M. H., 2012. Cortical multiscale line-edge disparity model. In: Campilho, A., Kamel, M. (Eds.), Image Analysis and Recognition. Vol. 7324 of LNCS. pp. 296–303.

Rosten, E., Porter, R., Drummond, T., 2010. Faster and better: A machine learning approach to corner detection. IEEE Trans. on Patt. Anal. and Mach. Intell. 32 (1), 105–119.

Rublee, E., Rabaud, V., Konolige, K., Bradski, G. R., 2011. Orb: An efficient alternative to SIFT or SURF. Proc. Int. Conf. Computer Vision (ICCV), 2564–2571.

Ruesch, J., Lopes, M., Bernardino, A., Hörnstein, J., Santos-Victor, J., Pfeifer, R., 2008. Multimodal saliency-based bottom-up attention. a framework for the humanoid robot iCub. Proc. IEEE. Int. Conf. Robotics Automation, 962–967.

Saeedi, P., Lowe, D., Lawrence, P., 2003. 3D localisation and tracking in unknown environments. Proc. IEEE Int. Conf. on Robotics and Automation 1, 135–145.

Saleiro, M., Carmo, B., Rodrigues, J., du Buf, J., 2013a. A low-cost classroom-oriented educational robotics system. Int. Conf. on Social Robotics, Springer LNCS 8239 (8), 74–83.

Saleiro, M., Farrajota, M., Terzić, K., Krishna, S., Rodrigues, J., du Buf, J., 2015. Biologically inspired vision for human-robot interaction. Universal Access in Human-Computer Interaction, Springer LNCS 9176, 505–517.

Saleiro, M., Farrajota, M., Terzić, K., Rodrigues, J., du Buf, J., 2013b. A biological and realtime framework for hand gestures and head poses. Universal Access in Human-Computer Interaction. Design Methods, Tools and Interaction Techniques for eInclusion 8009 (60), 556–565.

Saleiro, M., Rodrigues, J., du Buf, J., 2009. Automatic hand or head gesture interface for individuals with motor impairments, senior citizens and young children. Proc. Int. Conf. Softw. Dev. for Enhancing Accessibility and Fighting Info-Exclusion, 165–171.

Saleiro, M., Rodrigues, J. M. F., du Buf, J. M. H., 2012. Minimalistic vision based cognitive slam. Proc. 4th Int. conf. on Agents and Artificial Intelligence, Special Session on Intelligent Robotics, 614–623.

Saleiro, M., Terzić, K., Lobato, D., Rodrigues, J., Buf, J., 2014. Biologically inspired vision for indoor robot navigation. Vol. LNCS 8815. pp. 469–477.

Sandamirskaya, Y., Conradt, J., 2013. Learning Sensorimotor Transformations with Dynamic Neural Fields. Springer Berlin Heidelberg.

Scharstein, D., 2009a. Middlebury datasets for optical flow. Online; accessed 15-Sept-2015.
URL http://vision.middlebury.edu/flow/data/

Scharstein, D., 2009b. Middlebury datasets for stereo vision. Online; accessed 15-Sept-2015.
URL http://vision.middlebury.edu/stereo/

Scharstein, D., Szeliski, R., 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Computer Vision 47 (1/2/3), 7–42.

Schmidhuber, J., 2012. Multi-column deep neural networks for image classification. Proc. Int. Conf. Computer Vision and Pattern Recognition (CVPR), 3642–3649.

Scholl, B., Burge, J., Priebe, N., 2013. Binocular integration and disparity selectivity in mouse primary visual cortex. Journal of Neurophysiology 109 (12), 3013–3024.

Se, S., Lowe, D., Little, J., 2001. Vision-based mobile robot localization and mapping using scale-invariant features. Proc. IEEE. Int. Conf. Robotics Automation 2, 2051–58.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., Poggio, T., 2007. Object recognition with cortex-like mechanisms. IEEE Trans. on Pattern Analysis and Machine Intelligence 29 (3), 411–426.

Serre, T., Wolf, L., Poggio, T., 2005. Object recognition with features inspired by visual cortex. Proc. IEEE Comp. Soc. on Comp. Vision and Patt. Recogn. 2, 994–1000.

Shapley, R., Lennie, P., 1985. Spatial frequency analysis in the visual system. Annual Review of Neuroscience 8, 547–583.

Shotton, J., Blake, A., Cipolla, R., 2008. Efficiently combining contour and texture cues for object recognition. Proc. British Machine Vision Conf., 10pp.

Siagian, C., Itti, L., 2007. Biologically-inspired robotics vision monte carlo localization in the outdoor environment. Proc. IEEE Int. Conf. on Intelligent Robots and Systems, 1723–1730.

Simonyan, K., Vedaldi, A., Zisserman, A., 2014. Learning local feature descriptors using convex optimisation. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1573–1585.

Smith, R., Lane, P. C. R., Gobet, F., 2008. Modelling the relationship between visual short-term memory capacity and recall ability. Proc. Euro. Symp. Comp. Mode. Sim. 99–104.

Sousa, R., Rodrigues, J. M. F., du Buf, J. M. H., 2010. Recognition of facial expressions by cortical multi-scale line and edge coding. Proc. Int. Conf. on Image Analysis and Recognition (ICIAR) 1, 415–424.

Strecha, C., Bronstein, A. M., Bronstein, M. M., Fua, P., 2012. LDAHash: Improved matching with smaller descriptors. IEEE Tr. on Pattern Analysis and Machine Intelligence 34 (1), 66–78.

Suk, H., Sin, B., Lee, S., 2010. Hand gesture recognition based on dynamic Bayesian network framework. Pattern Recogn. 43 (9), 3059–3072.

Tao, Z., Bonnifait, P., Frémont, V., Ibañez-Guzman, J., 2013. Mapping and localization using GPS, lane markings and proprioceptive sensors. Proc. Int. Conf. Intelligent Robots and Systems, 406–412.

Tatler, B. W., Hayhoe, M. M., Land, M. F., Ballard, D. H., 2011. Eye guidance in natural vision: reinterpreting salience. Journal of Vision 11 (5), 1–23.

Terzić, K., Krishna, S., du Buf, J. M. H., 2015. A parametric spectral model for texture-based salience. Proc. 37th German Conf. on Pattern Recognition, 331–342.

Terzić, K., Lobato, D., Saleiro, M., Martins, J., Farrajota, M., Rodrigues, J. M. F., Lam, R., du Buf, J. M. H., 2013a. Biological models for active vision: Towards a unified architecture. Proc. Int. Conf. on Computer Vision Systems 7963, 113–122.

Terzić, K., Rodrigues, J., du Buf, J., 2013b. Fast cortical keypoints for real-time object recognition. Int. Conf. Image Processing, 3372–3376.

Terzic, K., Rodrigues, J. M. F., Lam, R., du Buf, J. M. H., 2015. BIMP: A real-time biological model of multi-scale keypoint detection in V1. Neurocomputing (150), 227–237.

Thrun, S., Burgard, W., Fox, D., 2000. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In: Proc. IEEE Int. Conf. on Robotics and Automation. Vol. 1. pp. 321–328.

Thrun, S., Gutmann, J., Fox, D., Burgard, W., Kuipers, B., 1998. Integrating topological and metric maps for mobile robot navigation: A statistical approach. Proc. 15th Conf. on Artificial Intelligence 1, 989–995.

Tomatis, N., Nourkbakhsh, I. Siegwart, R., 2001. Simultaneous localization and map building: A global topological model with local metric maps. Proc. IEEE/RSJ. Int. Conf. Int. Rob. Sys. 1, 421–426.

Trzcinski, T., Christoudias, C. M., Fua, P., Lepetit, V., 2013. Boosting binary keypoint descriptors. Proc. Int. Conf. Computer Vision and Pattern Recognition (CVPR), 2874–2881.

Trzcinski, T., Lepetit, V., 2012. Efficient discriminative projections for compact binary descriptors. Proc. European Conf. on Computer Vision, 228–242.

Vasudevan, S., Gächter, S., Siegwart, R., 2007. Cognitive Spatial Representations for Mobile Robots: Perspectives from a User Study. Eidgenössische Technische Hochschule Zürich, Autonomous Systems Lab.

Vogels, T. P., Abbott, L. F., 2005. Signal propagation and logic gating in networks of integrate-and-fire neurons. J. of Neuroscience 25 (46), 10786–10795.

Walther, D., Itti, L., Riesenhuber, M., Poggio, T., Koch, C., 2002. Attentional selection for object recognition - a gente way. Proc. of the Second International Workshop on Biologically Motivated Computer Vision, 472–479.

Wang, J., Kumar, S., Chang, S. F., 2010. Sequential projection learning for hashing with compact codes. Proc. Int. Conf. Machine Learning, 1127–1134.

Warren, P., Rushton, S., 2009. Optic flow processing for the assessment of object movement during ego movement. Current Biology 19 (19), 1555–1560.

William, K., Charles, J., 2008. Cortical neuronal responses to optic flow are shaped by visual strategies for steering. Cerebral Cortex 18 (4), 727–739.

Wu, Y., chuan Geng, L., Zhang, Q., li Chen, S., 2010. Self-adaptive Fuzzy PID controller for airborne three-axis pan tilt. Quantitative Logic and Soft Computing 82, 553–559.

Wurtz, R., 1998. Optic flow: A brain region devoted to optic flow analysis? Current Biology 8 (16), 554–556.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., Torralba, A., 2010. Sun database: Large-scale scene recognition from abbey to zoo. Proc. IEEE Conf. on Computer vision and pattern recognition (CVPR), 3485–3492.

Zetzsche, C., Wolter, J., Schill, K., 2008. Sensorimotor representation and knowledge-based reasoning for spatial exploration and localisation. Cognitive Processing 9 (4), 283–297.

Zhang, B., Huang, J., Lin, J., 2010. A novel control algorithm for object tracking by controlling pan/tilt automatically. Proc. 2nd Int. Conf. on Education Technology and Computer, 596–602.

Zhang, C., Li, Z., Cheng, Y., Cai, R., Chao, H., Rui, Y., 2015. Meshstereo: A global stereo model with mesh alignment regulariazation for view interpolation. Proc. Int. Conf. Computer Vision, 2057–2065.

Zhang, K., Li, J., Li, Y., Hu, W., Sun, L., Yang, S., 2012. Binary stereo matching. Proc. Int. Conf. Pattern Recognition, 356–359.