"Look at this"

Multimodal Joint Visual Attention Model for Natural Human-Robot Interaction in Domestic Environments

J. F. M. Domhof

Delft University of Technology

"LOOK AT THIS"

MULTIMODAL JOINT VISUAL ATTENTION MODEL FOR NATURAL HUMAN-ROBOT INTERACTION IN DOMESTIC ENVIRONMENTS

by

J. F. M. Domhof

in partial fulfillment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Delft University of Technology,
to be defended publicly on Friday March 13, 2015 at 9:30 AM.

Thesis committee:	Prof. Dr. ir. Pieter Jonker,	TU Delft
	Dr. ing. Maja Rudinac,	TU Delft
	Ir. Aswin Chandarr,	TU Delft
	Dr. David Tax,	TU Delft

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

PREFACE

This study was performed as a part of Master program Mechanical Engineering at the department Vision Based Robotics at the Delft University of Technology. I followed the track Biomechanical Design with specialization Biorobotics.

J. F. M. Domhof
Delft, February 2015

ABSTRACT

Due to population ageing, the cost of health care will raise in the coming years. One way to help humans, and especially elderly people, is the introduction of domestic robots that can assist people in daily life such that they are less dependent on home care. Joint visual attention models can be used for natural robot-human interaction. Joint visual attention is that two humans or a robot and a human have a shared attention to the same object. This can be accomplished by pointing, eye-gaze or by using speech. The goal of this thesis is to develop a non verbal joint visual attention model for object detection that integrates gestures, gaze, saliency and depth. The question that will be answered in this report is: how can the information from gestures, gaze, saliency and depth be integrated in the most efficient way to determine the object of interest?

Existing joint visual attention models only work when the human is in front of the robot, so that the human is in view of the camera. Our model should be more flexible than existing models, so it needs to work in different configurations of human, robot and object. Furthermore, the joint visual attention model should be able to determine the object of interest when the pointing direction or the gaze location is not available.

The saliency algorithm of Itti *et al.* [1] has been used to create a bottom up saliency map. The second bottom-up cue, depth, is determined by means of segmenting the environment to extract the objects. Apart from the bottom-up cues, top-down cues can be used as well. The pointing finger is identified and based on the eigenvalues and eigenvectors of the finger the pointing direction will be retrieved. A pointing map is created by means of the angle between the 3D pointing direction vector and the 3D vector from the pointing finger to the object. A hybrid model, which computes a gaze map, has been developed that switches depending on texture of the object between texture based approach and color based approach.

Depending on the configuration of the human, robot and object, three or four maps are available to determine the object of interest. In some configurations, the pointing map or gaze map is not available. In that case the combined saliency map is obtained by point wise multiplication of these three maps. If all four maps are at our disposal, all maps are added and multiplied by the pointing mask.

When the human and robot are opposite of each other and pointing, bottom up saliency and depth are combined, 93.3% of the objects are detected correctly. If the human is standing next to the robot, the gaze map, bottom up saliency map and depth map are combined, then the detection rate is 67.8%. If robot, human and object are standing in a triangular shape, the detection rate is equal to 96.3%.

The main contribution is that the joint visual attention model is able to detect objects of interest in different configurations of human, robot and object and it also works when one of the four cues is not available. Furthermore, a hybrid model has been developed that is able to create a probability gaze map. Depending on the texture of the object, the model chooses texture based approach or color based approach to generate this gaze map. The probability pointing map is generated using the 3D point cloud data instead of 2D information, which results in an accurate pointing map.

CONTENTS

1	Introduction	1
2	Existing Models	5
2.1	Joint Visual Attention Models	5
2.2	Bottom-Up Saliency Models	7
2.2.1	Saliency Models	7
2.2.2	Depth Segmentation	7
2.3	Top-Down Saliency Models	8
2.3.1	Gesture recognition	8
2.3.2	Gaze	9
2.4	Conclusion	10
3	General Setup	11
3.1	Test Setup	11
3.2	Hardware Specifications	12
4	Bottom-Up Saliency	13
4.1	Saliency Estimation	13
4.2	Depth Segmentation	14
4.3	Conclusion	16
5	Top-Down Saliency	17
5.1	Pointing Direction Estimation	17
5.1.1	General Outline	17
5.1.2	Estimation of Pointing Direction	17
5.1.3	Mathematical Description of a Pointing Gesture	20
5.2	Gaze Map Estimation	21
5.2.1	General Outline	21
5.2.2	Creating a Gaze Map	21
5.2.3	Converting from HD to kinect	23
5.3	Conclusion	24
6	Integration	25
6.1	Software Architecture	25
6.2	Integration of Bottom-Up and Top-Down Cues	25
6.3	Object Segmentation	27
6.4	Conclusion	27
7	Performance	29
7.1	Case: Opposite	30
7.2	Case: Next to	31
7.3	Case: Triangle	32
7.4	Case: Distance	33
7.5	Case: Bad Lightning Conditions	35
7.6	Case: Influence Different Saliency Maps	35
7.7	Case: Textured Background	36
7.8	Conclusion	37
8	Conclusion and Recommendations	39
	Bibliography	41

1

INTRODUCTION

Due to population ageing, the cost of health care will raise in the coming years. The first reason of the population ageing is that the people from the baby boom in the period after the second world war will become 65+. The population growth is visualised in figure 1.1, where the birth rate is shown by the yellow line. The peak around 1946 corresponds to the baby boom in the years after the second world war.

Figure 1.1: Population growth broken in birth rate (yellow) and migration (purple), 1900-2012 (CBS Bevolkingsstatistiek).

In addition to the baby boom, the life expectancy has increased significantly as well in the last 50 years because of better hygiene, health care, living conditions, food and peace. In figure 1.2, a graph is shown with the amount of 65+ and 80+ people. It is clearly visible that the amount of 65+ people will significantly increase to 4.7 million people in 2040.

Due to the composition of the population, a shrinking percentage of young people should take care of a growing percentage of elderly people. Figure shows 'de grijze druk', which is ratio between the amount of people of 65 years and older and the amount of people between 20 and 65 years old. It shows the ratio of older people with respect to the labour force.

Figure 1.2: Amount of 65+ and 80+ people in the Netherlands between 1950-2012 and a forecast 2013-2060. (CBS Bevolkingsstatistiek; CBS Bevolkingsprognose voor 2013-2060)

Figure 1.3: De grijze druk. (CBS Bevolkingsstatistiek; CBS Bevolkingsprognose voor 2013-2060))

One way to help humans, and especially elderly people, in daily life is the introduction of domestic robots as robots become more and more advanced. These robots are able to help the human with tasks in the household. Therefore it might help elderly people to take care of themselves longer and they are less dependent on home care. For instance, a robot could assist an elderly person by handing ingredients while the person is cooking a meal. Furthermore, the robot could hand a cup of tea to the person during breakfast. Another option, robots folds the laundry and put it in the wardrobe.

These service robots have to collaborate with humans to perform a task, for instance the human needs assistance from a robot to prepare a sandwich by handing certain objects. Furthermore, the robots should learn from humans to do certain task. A human is showing a robot how to prepare a sandwich and the robot is observing the performed actions and imitates these actions. These service robots are dealing with a diverse environment with a lot of different objects. So, there is a need for a system that enables the human to communicate naturally with robots. The robot and human should interact just like two humans would interact with each other. How to create a joint visual attention model for object detection?

Joint visual attention is that two persons or a person and a robot have the focus on the same object. How do humans communicate with each other to obtain a shared attention to the same object? There are three ways to determine the object of interest. One person can tell the other human or robot where to look, in this case speech is used to achieve joint visual attention. Furthermore, the person can show with gestures that a certain object is of interest, for example the person can point at it. Then, in some social situations, eye-gaze can be used to indicate a certain direction. In figure 1.4 an example can be found. Here the human is pointing and looking at the cup of coffee.

Furthermore, Humans are able to detect the most dominant part in a certain scene. For instance, if you look at figure 4.3a, the road sign is the most dominant region in the image. These dominant regions are called salient regions, because these regions discriminate against their surroundings. The visual attention system of the human consist of two different cues: bottom up (stimulus driven) and top down (task/goal driven). So,

Figure 1.4: An example of a joint visual attention model. The human is pointing and looking at the cup of coffee.

the bottom up salient regions have distinctive local features resulting in that it is classified as salient whereas the top down salient regions have a certain features related to the goal at a certain moment (e.g. you are looking for a blue t-shirt in your laundry).

Figure 1.5: The road sign is the most dominant region in this image.

I propose a multimodal joint visual attention model that imitates human-human interaction. This joint visual attention model can be used to communicate in a more natural way with robots such that the robot is able to detect the object of interest. The model combines bottom-up cues, depth and saliency, with the top-down cues, pointing and gaze. However it should also work when one of the four cues is not available. For example, if the robot and human are standing opposite of each other, gaze cannot be used because the viewing angle differs too much. In this case pointing, depth and saliency are combined to detect the object of interest. If the human and robot are standing next to each other and the object is in front of them, the robot cannot see the human because the human is not in view of the camera. Then, gaze, saliency and depth are combined to find the object of interest. The question that will be answered in this report is: how can the information from gestures, gaze, saliency and depth be integrated in the most efficient way to determine the object of interest?

The robot is equipped with a RGB-D camera. This camera has a RGB camera and a sensor that is based on the structured light principle to perceive the depth. Saliency detection models consider an object salient if it has distinctive local features that are different from their surroundings. Using features that are present in the colour images, such as colour, intensity and orientations (of edges), an saliency map can be estimated. In this map, the bright pixels correspond to salient regions and dark pixels correspond to inconspicuous regions. Another bottom up cue is depth. The point cloud, containing 3D points, can be segmented to find the objects in the scene.

In addition to bottom up cues, there are also top down cues such as the pointing gesture and the gaze direction. The pointing gesture can be estimated using the point cloud with 3D points, since the pointing

hand is present in the point cloud. Humans sometimes use eye-gaze to indicate a certain direction. The novelty of this work is that an eye-tracker is used for a joint visual attention model. An eye-tracker, which is a device for measuring eye positions and eye movement, is used to estimate the gaze direction.

Since this system is intended to work in a domestic environment, the system should work indoors. In this indoor environment, the objects are standing on the table. The human is wearing the eye-tracker and the robot is equipped with a Microsoft Kinect camera. Both the robot and the human are able to move, so the model should be able to deal with moving cameras. Since the robot should be suitable for real world applications, it is required that the algorithm is fast and accurate. Otherwise, the robot application is not viable. The human is pointing approximately 2 seconds to a certain object, so it should not take more than 2 seconds to determine the object of interest.

The structure of the report is as follows. In chapter 2 existing joint visual attention models are presented in order to determine where contribution can be made. This is followed by a chapter stating the specifications of the sensors and elaborates on the three different setups. Chapter 4 shows how the bottom up saliency is estimated. Furthermore, it is explained how the objects are segmented out of the scene. In chapter 5 elaborates on determining the pointing direction and how a pointing map can be created. In the second part of this chapter, the other top down map is made by using the gaze direction of the eye-tracker. In chapter 7 the performance of the system is determined by testing the system in the three different setups. The conclusions and recommendations can be found in chapter 8.

2

EXISTING MODELS

This chapter elaborates on three groups of existing non verbal joint visual attention models. For every model, it is briefly explained how the object of interest is obtained. Based on the existing models, it is determined where contributions can be made in order to improve their shortcomings. Existing models depend on bottom-up and top-down cues, so section 2.2 and 2.3 elaborates on these cues, respectively.

2.1. JOINT VISUAL ATTENTION MODELS

In the work of Yucel *et al.* [2], saliency and gaze are integrated to identify the object of interest. The gaze of the human is approximated based on an estimation of the head pose, which is corrected by Gaussian Process Regression. Then this target location is combined with a saliency map to find the object of interest.

The robot and human are sitting opposite of each other, meaning that the robot and human are facing each other. Between them a table with objects is located. First, a Viola Jones algorithm (Viola and Jones [3]) detects the face area and elliptical cylinder (Xiao *et al.* [4]) is fitted at the position of the face by means of Adaboost classifier with Haar wavelet features (Freund and Schapire [5]). In that way, the head pose of the human is estimated.

However, the head pose does not directly match the gaze direction, because possible eye movements are not taken into account. Therefore a Gaussian Process Regression method (Rasmussen and Williams [6]) has been trained to interpolate the gaze direction and depth based on the 3D head pose.

The final step consists of locating the object of interest. This is accomplished by narrowing the search window in the saliency map by the gaze direction and depth. Then, locally the maximum value in the saliency map corresponds to the most salient region. A result is shown in figure 2.1.

Figure 2.1: Estimated gaze direction and object depth via GP regression (line) together with the center of the manually annotated object of interest (solid dot).

Schauerte *et al.* [7] determines the pointing direction of the human and uses that to find the object of interest in the saliency map by locally searching for the most salient point.

First, a head-shoulder detector based on Histogram of Oriented Gradients (based on [8]) is used to find the position of the head and the skin color. Based on skin-color and detected motion a motion saliency map is created, which indicates the location of the hands. The pointing directions is defined as the vector from the center of the head to the hand locations. Using this vector and taking into account the uncertainty, a top-down saliency map is created.

Furthermore, a bottom up approach is used to use intensity, colors and orientation (of edges) to indicate salient regions. The spectral residual approach from Hou and Zhang [9] is adapted to efficiently run on multi-core CPUs and modern GPUs. Finally S_b a bottom up saliency map is obtained.

Figure 2.2 shows typical results for a bottom-up and top-down saliency map. The focus of attention point corresponds to the maximum value in the joint saliency map, being equal to $S = S_b(x)S_t(x)$. The image is segmented using the maximal stable extremal regions algorithm (Matas *et al.* [10]) in order to find the object of interest. At the right hand side in figure 2.2 the attended object is shown.

Figure 2.2: Left to right: the input image, bottom-up saliency map, top-down pointing saliency map, and attended object.

Schauerte and Fink [11] join contextual information from language with a stimulus driven saliency model to create a top down saliency map. The attention in this map is directed towards a certain region by means of a pointing gesture. Then the object of interest is found at the location of the most salient region in the joint saliency map.

A combination of a modulatable neuron-based saliency model (Navalpakkam and Itti [12]) with a phase-based saliency model (Guo *et al.* [13]) is computed as top down modulated visual saliency map. The features in the top-down modulated visual saliency map, being hue, lightness and orientation are encoded by a populations of neurons with overlapping Gaussian curves. The stimulus driven model that uses the phase spectrum of the Fourier transform from Guo *et al.* [13] is used for the stimulus driven cues.

The pointing direction is defined as the vector from the eyes to the hand location. This method originates from the fact that people align their pointing finger with their dominant eye. To estimate the position of the eyes, a head-shoulder detector is used. The hand is located based on movement and skin color.

An example can be found in figure 2.3, where the input image, visual saliency model, top down pointing map and combined saliency map are visualized.

Figure 2.3: Saliency example, left-to-right: the original image I , the visual saliency map SB (modulated with the background distribution as distractor model and a uniform target model), the top-down pointing map ST , and the combination result $S = (SB \cdot ST)$. In this example is the referred-to blue cup directly selected in the initial focus of attention (FoA).

2.2. BOTTOM-UP SALIENCY MODELS

The existing joint visual attention models, which are explained in the previous section, consists of bottom-up and top-down cues. These cues are integrated to find the object of interest. In this section, existing bottom up saliency models are elaborated on in order to discover state of the art models. The first part is devoted to saliency models and the second part to segmentation algorithms.

2.2.1. SALIENCY MODELS

The biologically plausible model of Itti *et al.* [1]. Since it is one of the first saliency algorithm it is widely used as a comparison when developing new saliency algorithms. Furthermore, according to [14] it is both fast and effective therefore it is very suitable for real world applications. An example saliency map can be found in figure 2.4.

Figure 2.4: Image and saliency map as computed with model of Itti *et al.* [1]

The model of Walther and Koch [15] is based on the visual attention model of Itti *et al.* [1], but it is focused on detecting salient proto-objects (Rensink [16]). The model of Itti *et al.* [1] is extended in this paper by a function to select the proto-object in the saliency map. This is accomplished by selecting the dominant conspicuity map and use this "winning" feature map to threshold the neighbourhood of the "winning" point. Every pixel that is larger than the threshold belongs to the object and in this way the shape of the proto-object is retrieved to extract the proto-object.

Furthermore, the model of Hou and Zhang [17] is very fast and effective according to Borji *et al.* [14]. The authors present an alternative approach for human visual attention that is motivated by the sparse coding strategy (Olshausen *et al.* [18]) in the visual cortex. It uses a combination of different linear filters, called features, to code each image patch. Each feature is evaluated with respect to its Incremental Coding Length (ICL). Finally, the saliency map is created by summing up the activity of all features.

The algorithm of Achanta *et al.* [19] is based on frequency tuning, which is done by using a Difference Of Gaussians (DoG) filter to select the desired passband width. Actually, the output of several edge detectors is added up at different scales resulting in that salient regions are filled with the output of the edge detectors.

Due to a high performance, the Adaptive Whitening Saliency (AWS) is also very interesting to use (Garcia-Diaz *et al.* [20]). It is a top performing model in static and dynamic scenes and second best in synthetic images according to Borji *et al.* [14].

Rudinac and Jonker [21] uses the spectral residual approach of Hou and Zhang [9] to three different channels. In that way, an inexpensive computational model for saliency detection is created.

2.2.2. DEPTH SEGMENTATION

Parvizi and Wu [22] present a segmentation method that is based on adaptive range segmentation. The first step in the method of Parvizi and Wu [22] is to create a depth histogram:

$$h_z(n) = \sum_{(x,y) \in \hat{I}} [\delta(\hat{I}(x,y) - n)] \quad (2.1)$$

This means that the depth map $\hat{I}$ is mapped from R^2 to $0, 1, 2, \dots, N - 1$ bins. Then, a probability density function of the depth map is created using a Kernel density estimation method. The depth segmentors S_i are based on the local extrema from the probability density function, so these are denoted as: $L_{max} \in \mathbb{R}^{M \times 1}$ and $L_{min} \in \mathbb{R}^{M-1 \times 1}$. Then, the boundaries of the segmentors are defined as defined as the minimal values in the probability density function. Based on these segmentors, individual depth divisions can be formed. The results of implementing this method can be found in figure 2.5.

Figure 2.5: Segmentation process of the proposed algorithm.

Figure 2.6: Segmentation result of the Enhanced Mean Shift Segmentation (Bleiweiss and Werman [23])

The segmentation algorithm of Bleiweiss and Werman [23] is based on the general mean shift segmentation algorithm of Comaniciu and Meer [24]. To improve the general mean shift segmentation, depth can be added into the mean shift vector as an extra dimension. However, depth data is noisy and color data is dependent on lightning conditions, therefore fusing the two should be done in the most efficient way. The noise in the depth data is estimated. First, the color data is converted to $L^*u^*v^*$. Then the original depth image D is filtered with a bilateral filter, which results in a smooth image S . The residual is the approximation of the noise matrix:

$$W = |S - D| \quad (2.2)$$

Then the original weights are scaled by W :

$$w_6(x) = \frac{1}{W(x) + 1} w_6(x) \sigma \quad (2.3)$$

where σ is an additional scale factor to give more overall weight to depth data when it is not noisy.

2.3. TOP-DOWN SALIENCY MODELS

Apart from bottom-up cues, there are top-down cues as well. Top-down cues can specify a certain region by means of, for example, gestures and gaze. In this way, the search region is narrowed down by these top down cues. The first section is dedicated to gesture recognition, since a pointing gesture can be used as a top down cue. The second part elaborates on gaze estimation methods, because humans use often eye-gaze to indicate a certain direction.

2.3.1. GESTURE RECOGNITION

The OpenNI tracker estimates the location of all body parts, including the location of the hand. In the work of van den Bergh *et al.* [25], the authors use the hand location to determine the pointing direction. The

pointing direction in 2D is determined in three steps, which are illustrated in 2.7. First, the hand is segmented based on the depth information. Then, a rectangle is placed over the hand/arm and the point where the rectangle intersects with the wrist is found. The vector from the intersection with the wrist to the center of the hand is the pointing direction. The input of the Average Neighbourhood Margin Maximization classifier is a combination of a cropped grey scale hand image, skin color segmentation and depth image. It is able to classify four different gestures, open hand (all fingers extended), fist, pointing (index finger extended), and a second pointing gestures (index finger and thumb extended).

Figure 2.7: Estimating the hand orientation: (a) hand center and segmentation; (b) rectangle placed over hand; (c) intersection with the rectangle and estimated wrist position; (d) estimated hand orientation.

Using the 3D coordinates of joints of the human body determined by the Kinect software, a Gaussian Mixture Hidden Markov Model (GMHMM) is trained for recognizing full body gestures in the paper of Song *et al.* [26]. Then it is able to extract skeleton information based on the location of the joints (see figure 2.8). The body parts are used as features for the GMHMM: forearm, upper arm, thigh, crus and the spine. The GMHMM is trained to be able to recognize gestures and after that the model is able to classify human gestures.

Figure 2.8: Wave the right hand gesture

2.3.2. GAZE

Eye-tracking devices fall into three different classes. Eye-attached tracking is a technique where a special lens is measuring the gaze direction of the subject. Another method is called Electrical potential measurements that makes use of electrodes placed around the eye to measure electrical potential. Then, the third class of devices are optical tracking devices where cameras are used to estimate the gaze direction. This can be done by either a head mounted optical tracker (see figure 2.9) or a camera in front. Furthermore, there are also optical tracking techniques to estimate the gaze based on the head pose (Yucel *et al.* [2]).

The working principle of an eye-tracker is explained in Guestrin and Eizenman [27]. The gaze direction is determined based on the relative position of the pupil with respect to the the corneal reflection point (or glint). The glint is a virtual point in the eye that originates from reflection of light of the LEDs. Consider a ray of light that falls perpendicular onto the corneal surface. This ray of light is passing trough the corneal surface and reaches the lens. The second ray of light is the reflection of the light from the LED to the camera. In that case, the incidence angle and the reflection angle are equal; meaning that the corneal surface acts like a mirror. These two rays together form the corneal reflection point and it is used as a reference point to measure the position of the pupil.

Figure 2.9: Example of an eye-tracker¹

2.4. CONCLUSION

The problem with existing joint visual attention models is that they only work when the robot and human are facing each other. This means that these models are not very flexible, because these only work in one configuration of human, robot and object. The idea is to create a joint visual attention model that is more flexible, because it is both working when the human is standing next to the robot and it is working when the human is facing the robot. Furthermore, our model should combine the pointing direction and gaze together to find the object of interest, but it should also work when gaze or gestures are not available. An eye-tracker will be used to estimate the gaze for this joint visual attention model.

The goal of this thesis is to create a flexible joint visual attention model that works in the different scenarios. A flexible system is obtained by integrating bottom up cues with top down cues. The bottom up cues are bottom up saliency and depth and the top-down cues are pointing and eye-gaze. The object of interest is determined by integrating all four available cues.

¹Pupil Labs eye-tracker, pupil-labs.com

3

GENERAL SETUP

In this chapter general information is given about the system and the setups. In the first section of this chapter, the setups are explained and in the second section the specifications of the hardware are stated.

3.1. TEST SETUP

The Joint Visual Attention (JVA) model should be very flexible and work in an indoor environment. The condition is that the objects are standing on a table. Furthermore, the human is wearing the eye-tracker and the robot is equipped with a Kinect camera. On top of the Kinect sensor a HD camera is mounted.

When humans are interacting with each other there are three ways how the human, robot and object are situated. In the first case, the human and robot are standing opposite of each other and the objects are located in between them. This setup will be called 'opposite' and it is visualised in figure 7.1a. In this case the human is in view of the camera of the robot, so the robot is able to determine where the human is pointing at. However, the difference in viewing angle between the robot and the human is 180° , so the human and the robot have a totally different view. For instance, the human is seeing the front side of a object and the robot the back side.

This setup will be referred to as 'next to', since the robot and human are standing next to each other. The object is in front of them as can be seen in figure 7.1b. In this case the human is not in view of the camera, it is not possible to determine the pointing direction, so the robot should be able to find the object of interest by combining the gaze with bottom up saliency and depth.

In order to combine all four cues, the human should be in view of the camera and the human and the robot should have approximately the same viewing angle. This means that the human, robot and object are positioned in a triangular shape, therefore it will be referred to as 'triangle'. This setup can be found in figure 7.1c. The pointing direction can be estimated because the robot sees the human and the difference in viewing angle is small enough to create a gaze map.

Figure 3.1: The setups 'opposite', 'next to', and 'triangle' are shown here from left to right.

3.2. HARDWARE SPECIFICATIONS

Below, the specification of the Kinect, eyetracker, HD camera and the laptop are listed.

SPECIFICATIONS KINECT

Below, the specification of the Kinect sensor are summarised. The main advantages of this sensor compared to RGB camera are: working in low light conditions, color and texture invariant, calibrated scale estimate, resolving silhouette ambiguities in pose (Shotton *et al.* [28]). The sensor is based on the structured light principle; projecting the scene with IR light and the depth sensor is determining the 3D locations of the scene. Furthermore the resolution of the Kinect sensor is 640×480 pixels at a frame rate of 30 fps. According to Khoshelham and Elberink [29] the depth resolution decreases quadratically with increasing distance. At a distance of 5 m it is approximately equal to 7 cm. Both the IR camera and the RGB camera are calibrated using a chessboard pattern. Furthermore, both cameras are stereo calibrated to have an aligned 2D images with the point cloud.

SPECIFICATIONS OF THE PUPIL LABS EYE-TRACKER

An eye-tracker, which is a device for measuring eye positions and eye movement, is used to estimate the gaze direction. The device that is used for the joint visual attention model is the bare version from Pupil Labs. It consist of two cameras and a light source. The eye camera has a resolution of 640×360 pixels, while the world camera has a resolution of 1280×720 . Both cameras record images at a frame rate of 30 fps. The viewing angle of the world camera is 68° .

Figure 3.2: The pupil labs eye-tracker.

SPECIFICATIONS HD CAMERA

Since the resolution of the color camera of the Kinect was not sufficient, because the objects did not have enough detail to find key point matches in the eye-tracker image, a Logitech HD camera has been attached to the Kinect. This HD camera has a resolution of 1920×1080 pixels. The camera is positioned on top of the Kinect camera and both cameras are stereo calibrated, so the rotation matrix and translation vector are known.

COMPUTER

The C++ software is running on a laptop with a Intel Core i7-2670QM CPU @ 2.20GHz $\times$ 8 processor and 8GB RAM memory. Furthermore, Robot Operating System (ROS), which is running on Ubuntu 12.04 LTS (Precise Pangolin), is used as a set of software libraries and serves as building blocks for this model. A ROS node is a process that performs a certain computations. There are communicating with each other using ROS topics. A node subscribes to a certain topic if it wants to receive ROS message. It can also publish messages on a certain ROS topic. In addition to ROS, the PointCloud Library (PCL) will be used for 3D processing and OpenCV for 2D processing.

4

BOTTOM-UP SALIENCY

This chapter presents the bottom-up cues for the joint visual attention model. Bottom-up models consider an object salient if it has distinctive local features that are different from their surroundings. Based on the RGB images a bottom up saliency detection model can be used to estimate the saliency for every pixel. This is elaborated on in section 4.1. Based on the pointcloud from the Kinect camera, all the objects in the scene can be segmented. This means that the location and the shape of all objects is known. Section 4.2 explains how the scene is segmented and how the objects are extracted.

4.1. SALIENCY ESTIMATION

The saliency detection algorithm of Itti *et al.* [1] is used to compute the saliency map, because it is a widely used and biologically inspired. Furthermore, this method is suitable for real time applications because it is fast.

This biologically plausible saliency model is based on the model of Koch and Ullman [30] and the general overview is shown in figure 4.1. An intensity image is created using the RGB channels: $I = \frac{r+g+b}{3}$. Furthermore, the hue component is decoupled from the intensity by normalizing it by the intensity. Four color channels are created: $R = r - (g + b)/2$ for red, $G = g - (r + b)/2$ for green, $B = b - (r + g)/2$ for blue, and $Y = (r + g)/2 - |r - g|/2 - b$.

The Gaussian pyramid of nine spatial scales is created for the intensity, colors and orientations in order to sub-sample and filter the input RGB image. Using this Gaussian pyramid, 42 feature maps are generated as explained in the following paragraphs. Then, these feature maps are created by means of the center surround

Figure 4.1: General Architecture.

difference with $c \in \{2, 3, 4\}$ and $s = c + \delta$, $\delta \in \{3, 4\}$.

$$I(c, s) = |I(c) \ominus I(s)| \quad (4.1)$$

Where c is a centre fine scale and s is a surround coarse scale. In total six different feature maps are generated for the intensity channel. Another twelve feature maps are created based on "color double opponent" system. This principle is visualized in figure 4.2, where in the center of the receptive field one color is excited and the surround is inhibited by the opponent color. This results in six feature maps for the red-green opponency and another six feature maps for the yellow-blue opponency.

Figure 4.2: Visual representation of color double opponency of color feature maps.

$$RG(c, s) = |(R(c) - G(c)) \ominus (G(s) - R(s))| \quad (4.2)$$

$$BY(c, s) = |(B(c) - Y(c)) \ominus (Y(s) - B(s))| \quad (4.3)$$

In addition to the feature maps for intensity and color, there are maps created for local orientation as well. These are constructed using oriented Gabor pyramids $O(\sigma, \theta)$, where $\sigma \in [0..8]$ and $\theta \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$. In total, 24 different feature maps are obtained:

$$O(c, s, \theta) = |O(c, \theta) \ominus O(s, \theta)| \quad (4.4)$$

Now, conspicuity maps are computed by combining all feature maps from each of the three features into one map:

$$\bar{I} = \bigoplus_{c=2}^4 \bigoplus_{s=c+3}^{c+4} N(I(c, s)) \quad (4.5)$$

$$\bar{C} = \bigoplus_{c=2}^4 \bigoplus_{s=c+3}^{c+4} [N(RG(c, s)) + N(BY(c, s)))] \quad (4.6)$$

$$\bar{O} = \sum_{\theta \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}} N \left(\bigoplus_{c=2}^4 \bigoplus_{s=c+3}^{c+4} N(O(c, s, \theta)) \right) \quad (4.7)$$

Where $N(\cdot)$ is a normalization operator and $\oplus$ is a point-by-point addition. Finally, the saliency map is obtained by averaging all normalized conspicuity maps:

$$S = \frac{1}{3} (N(\bar{I}) + N(\bar{C}) + N(\bar{O})) \quad (4.8)$$

A result of this method can be found in figure 4.3. The ROS implementation of van der Weem and Akman [31] has been used, because it provided the correct outputs and speed.

4.2. DEPTH SEGMENTATION

Due to the usage of a RGBD camera, a point cloud with 3D points of the environment is available. This can be used to find the shape and location of every object in the scene, therefore object clusters are extracted. These objects cluster consists of the 3D points that belong to the object. In this section it is explained how the object clusters are segmented out of the scene. This implementation is based on the Organized Multiplane Segmentation algorithm of Trevor *et al.* [32].

Figure 4.3: Image and saliency map as computed with model of Itti *et al.* [1]

The first step consists of filtering the point cloud in order to reduce its size by removing the background out of the cloud. A working area of 2 m is considered. In most cases these points belong to the wall, so these points can safely be deleted since the objects are located on top of the table.

Then, the surface normals are estimated. A surface normal is a vector that is perpendicular to the surface. Therefore, it can be used to detect the planes in the point cloud, because all normal vectors of a plane point in the same direction. The Organized Multi-plane Segmentation algorithm of Trevor *et al.* [32] is used to detect planes based on the surface normals. If a region consists of more than 1000 points, than the region is considered as a plane. This value can be adapted depending on the situation.

Figure 4.4: Result image with the coloured objects and planes are visualised by means of the red normal vector.

After that, the object clusters are extracted by clustering the remaining points after planar removal. The default depth threshold is 5 cm, but it can be adapted if a lower threshold is required. In figure 4.4 a segmentation result can be found where the coloured areas corresponds to the objects and the bordered planes with normal vector are the planes. The remaining clusters belong to the objects located on the table. The final step is transform the clusters to a 2D image using equation 4.9. The intrinsic parameters are obtained by calibrating the RGBD camera. The mask denotes all objects in the scene, which can be seen in figure 4.5b.

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = A \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} \quad (4.9)$$

Figure 4.5: At the left hand side the image of the scene is shown and at the right hand side the image containing all objects.

4.3. CONCLUSION

In this chapter, the biologically inspired and widely used saliency detection algorithm of Itti *et al.* [1] is explained. It computes conspicuity maps by means of center-surround differences for each of the three features: color, intensity and orientations. The normalized conspicuity maps are averaged to obtain the bottom up saliency map.

The objects are segmented out of the scene by computing the surface normals for all points in the point cloud. Since the surface normals belonging to a plane are aligned, the planes can be detected in the scene. The objects are extracted by clustering the remaining points after planar removal. All objects are transformed from 3D to 2D to create a mask containing all objects.

5

TOP-DOWN SALIENCY

Apart from bottom up cues, there are also top down cues such as pointing and eye-gaze. This chapter shows how the pointing direction is estimated and how it can be used to create a pointing map. This map encodes the probability if a pixel belongs to a point in the pointing direction or not. The higher the value in the pointing map is, the brighter the pixel. In second part of this chapter, the hybrid model that creates a probability gaze map is explained.

5.1. POINTING DIRECTION ESTIMATION

Since the goal is to develop a model for shared visual attention, both the pose and gesture of the human should be estimated. The main idea is that the human is pointing at a certain object and the robot knows which object is of importance. This section elaborates how the pointing finger is detected. First, the general outline is stated in section 5.1.1, after that the whole procedure is explained more thoroughly in section 5.1.2. Finally, in section 5.1.3 the pointing direction will be visualised by means of a pointing probability map that is created to be able to integrate multiple cues.

5.1.1. GENERAL OUTLINE

A cascade of filters is used to extract the hand point cloud. When the hand cloud is available, the cloud will be used to identify the pointing finger. Then, the pointing direction of the finger should be estimated. The general outline for this procedure can be found in figure 5.1.

Figure 5.1: General outline of the pointing direction estimation algorithm.

5.1.2. ESTIMATION OF POINTING DIRECTION

The OpenNI-tracker tracks all the limbs of the human body based on a Randomized Decision Forest Classifier (Shotton *et al.* [28]). This means that an approximate position of the hand is available. The first step is to filter the point cloud to remove all irrelevant points and to keep all points that represent the hand. For that purpose,

the point cloud is filtered such that all points inside a cube around the hand location are kept. A value of 0.20 m is used because it takes into account the inaccuracy of the OPEN-NI body tracker that estimates the position of the hand.

The second step is to determine a more accurate location of the hand, because the location from the tracker is not very accurate. A conditional filter is a filter that removes all points that does not satisfy a certain condition. This condition (or conditions) can be chosen by the user. In this case, a conditional filter with spherical conditions is used to remove all points outside a sphere which is located at the hand location. The spherical condition is equal to:

$$x^2 + y^2 + z^2 = R^2 \quad (5.1)$$

This equation can be rewritten in a more general quadratic form, where T represents the 3D location of the sphere and R represents the radius of the sphere. The default value for R is 0.15 m, because normally the point cloud of the hand should fit in the sphere of 15 cm. The general form is:

$$(X - T)^T A (X - T) - R^2 = 0 \quad (5.2)$$

Where X represent the 3D coordinates of a point in euclidean space and A equals:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (5.3)$$

It should be rewritten in the general quadratic form:

$$X^T A X + 2v^T X + C \leq 0 \quad (5.4)$$

$$v^T = -TA \quad (5.5)$$

$$C = -R^2 + T^T A T \quad (5.6)$$

The large circle in figure 5.2 visualises the spherical conditions. After filtering, the remaining point cloud consists for the most part of points belonging to the hand. Furthermore, some points from the arm are present and in some cases some points belonging to objects are present as well if there are objects in the neighbourhood. The remaining point cloud is used to estimate a more accurate and stable location of the arm, since the hand location of the OpenNI tracker sometimes turned out to be inaccurate. The new hand location equals the centroid of the filtered point cloud. In figure 5.3, the location of the centroid of this filtered point cloud is shown with a blue dot at the edge of the hand.

Figure 5.2: In this image, the conditions are visualised in 2D.

As explained in the previous paragraph, the point cloud still contains points that belong to the arm of the human. Therefore, another filter is used to remove these points as well. For that a conditional filter with cylindrical conditions is used (the white box in figure 5.2). The cylinder is located at the centroid of the cloud,

which was computed in the previous step and it is pointing in the direction of the vector from elbow to hand that were obtained using the OpenNI tracker. The cylindrical conditions look like:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (5.7)$$

It should be rewritten in the general quadratic form:

$$X^T A X + 2v^T X + C \leq 0 \quad (5.8)$$

$$v^T = -T A \quad (5.9)$$

$$C = -R^2 + T^T A T \quad (5.10)$$

where R is the radius of the cylinder. The default for this ROS parameter is 0.25 m. And a second condition is needed that the cylinder is only present in the local positive z direction. This means that it is not a infinite cylinder but a finite cylinder. This can be described by the equation $z > 0$ and in a general form it equals:

$$X^T A X + 2w^T X + C \leq 0 \quad (5.11)$$

$$A = 0 \quad (5.12)$$

$$v = \begin{pmatrix} 0 \\ 0 \\ 0.5 \end{pmatrix} \quad (5.13)$$

$$w^T = v^T R^T \quad (5.14)$$

$$C = -2v^T R^T T \quad (5.15)$$

In order to remove objects close to the hand, the cluster of the hand is extracted in step four. All clusters are extracted from the remaining point cloud and the cluster with the shortest distance to the centroid of the hand is the hand cluster. So, now the point cloud of the hand is available.

The current point cloud only consists of points that belong to the hand. However, the pointing finger point cloud is needed to estimate the pointing direction therefore the next step is to remove the hand palm. Again a conditional filter with spherical conditions (small circle in figure 5.2), which is located at the centroid of the hand pointcloud (see figure 5.3) is used, but this time all points inside the sphere are removed. This means that points that belong to the fingers are still present. Clusters in the remaining point cloud are extracted and the cluster with the largest distance from the elbow represents the pointing finger cluster.

The position of the finger is determined by computing the centroid of the pointing finger. Furthermore, the pointing direction should be estimated. The eigenvectors from a cluster depends on the shape of the cluster. In the pointing direction the shape of the cluster is most spread, so the eigenvalue is highest in that direction (Gallagher [33]). So the eigenvalues of the finger cloud are computed and the eigenvector belonging to the largest eigenvalue represents the pointing direction.

Figure 5.3: In this figure, the point cloud of the hand and the point cloud of the pointing finger are visualised in purple and red, respectively. The blue dot at the hand represents the location of the hand as computed in step 2 and the green dot is the centroid of the hand cloud. Furthermore the two blue dots on the finger represent the centroid and the pointing direction of the finger.

Figure 5.4: In this image, the pointing gesture is shown. The centroid of the hand is visualised with the green dot and the blue dot represents the pointing finger. Furthermore the line determines the projected from 3D to 2D pointing direction.

5.1.3. MATHEMATICAL DESCRIPTION OF A POINTING GESTURE

In the previous section the procedure to determine the pointing direction is explained. The question is how can this pointing gesture be used in a joint visual attention model. In order to integrate the different cues, a pointing probability map is created that shows if an object is in the pointing direction or not. So, objects in the pointing direction should have a high value and objects that are not should have a low value. In this section it is explained how this can be computed.

In section 4.2, it is explained how the objects are extracted in the scene. The object clusters are used here to create a pointing map. For every point that belongs to an object, the vector from the position of the finger to this point is computed. The angle between this vector and the pointing direction vector is computed using equation 5.16.

$$\cos\beta = \frac{x \cdot d}{|x| \cdot |d|} \quad (5.16)$$

where d is the pointing direction vector and x is the vector from the position of the finger to the object point.

The value in the probability map has an inverse relation with the angle, so if the angle is small than the value is high. If the angle is large, than the value is low. In this case, a normal distribution with a zero mean is used to create a saliency map (Schauerte and Fink [11]). Equation 5.17 shows the normal distribution, where σ is the standard deviation. The value depends on the configuration and application, but in general a standard deviation of 15.0° works well. An example probability pointing map can be found in figure 5.5. Note that the probabilistic value is only computed if the angle is smaller than 2σ in order to limit the number of points in the pointing map.

$$P_{i,j} = \mathcal{N}(\beta, 0, \sigma^2) \quad (5.17)$$

Figure 5.5: An example probability pointing map with a standard deviation of 15.0°

5.2. GAZE MAP ESTIMATION

A eye-tracker is a device that is estimating the gaze direction of the human. However in this case the eye tracker 'knows' where the human is looking at, however the robot needs to know where the human is looking at. This section explains how the eye tracker can be used to create a gaze map, which indicates the probability that a certain pixel belongs to the object of interest.

5.2.1. GENERAL OUTLINE

There are several options how the information of the eye-tracker can be used to detection objects of interest. This section explains why and which approach is chosen to create a gaze map.

The first option is that the gaze direction can be transformed from eye tracker image to the Kinect image by matching key points, determining the fundamental matrix and locating the gaze direction in the Kinect image. However, this method is not stable enough, because the key point matches are not stable and therefore the fundamental matrix is not stable either. This means that this method will not be reliable. Moreover, taking into account that both cameras can move this is not a good approach.

Another option is to place markers in the scene and use those markers to compute a more stable fundamental matrix. The advantage is that a marker has a unique ID, so these can be matched in both image. The location of the markers in both images can be used to determine the fundamental matrix. In this case, a least eight matches are required to be able to compute a fundamental matrix. These markers need to be visible in both images and this is in practice pretty hard because of the different viewing angles of both cameras. All the markers should be large or close to the camera and this limits the view significantly, because the human and objects are less visible in that case.

Furthermore, instead of transforming the gaze direction, it is also possible to create a probability map. Based on the object where the human is looking at in the eye tracker image, a probability map is created in which the most likely location of that object is shown. The advantage of this map is that it is easy to combine it with the other cues, for instance with the saliency map. Thus, the most efficient approach is to create a probability map that shows the most likely location of the object of interest. In the next section this approach will be explained in more detail.

5.2.2. CREATING A GAZE MAP

This section explains how to create a probability map based on gaze which is estimated by an eye-tracker. The hybrid model, which creates a gaze map, automatically switches between a texture based approach and color based approach depending on the texture of the object. The first approach is very suitable for textured objects, but not for uniform coloured objects. So, for uniform coloured objects, the second approach is used.

TEXTURE BASED APPROACH

First, a sub image around the gaze point is obtained which contains the object of interest in the eye-tracker image. In general the size of the sub image is approximately 150×150 pixels, because in that case the object fits in the window taking into account the accuracy of the eye-tracker. However in some cases, a smaller window might be desired because for example the objects are located very close together. This can be done

by changing the ROS parameter window size.

Then key points are detected in the sub image and in the target image using a Speeded-Up Robust Features (SURF) Detector ([34]) and for every key point the descriptors are calculated. After that, the key points are matched using a Fast Library for Approximate Nearest Neighbors (FLANN) based descriptor matcher ([35]). The worst matches in terms of distance between the descriptor values are removed. An example image with good matches can be found in figure 5.6.

Figure 5.6: At the left hand side the eye tracker sub image is shown and at the right hand side the Kinect image is shown. The matches are visualised with the coloured lines.

So the location with a lot of matches is most likely the location of the object of interest. This will be visualised by a probability map. At every location of a key point a Gaussian function with a standard deviation of 101 pixels is drawn, which works well for the given size of objects in the scene. So, the gaze map is computed by summing all Gaussian functions that are located at every location of a key point:

$$G = \sum_{i=1}^n N(\mu_i, \sigma^2) \quad (5.18)$$

If key points are close, they are overlapping so locally a very high value appears. This results in locations with a high value where a lot of key points are present. The last step is to normalize this probability map and an example can be found in figure 5.7. For textured objects a lot of key points can be found, but not for uniform coloured objects, therefore a second approach will be used for these objects.

Figure 5.7: This is the probability map of the scene, which is computed based on the matched key points. If you compare this image with 5.6, then the most bright location correspond with the coffee cup.

COLOR BASED APPROACH

The object of interest in the eye-tracker image is segmented first. Using the segmented object, a color model for the object is created. Then for every pixel in the robot camera, the probability is determined that the color belongs to the color model.

First, the images are converted to HSV. Note that only Hue and Saturation are used. Value is not used, because in that case the model is illumination invariant.

Then a segmentation of the object of interest is obtained to serve as a color model for the object. So, a Grabcut algorithm (Rother *et al.* [36]) is used to find the segmented object (figure 5.8).

Figure 5.8: The segmented apple. This is the result of the Grabcut algorithm when the human is looking at the apple.

Note, that foreground locations are available because the human will be looking at the object, however the algorithm is also able to segment the object without having these foreground points. Using the segmented object a model histogram with the color of the object can be created. For every pixel in the Kinect image, the color is compared to the model histogram. If the value of a pixel corresponds to the model color histogram, then the probability is high in back-projection map. This method is called histogram backprojection (Swain and Ballard [37]). An example back-projection map is shown in figure 5.9.

Figure 5.9: The back projection probability map of the apple. Clearly visible that the apple has the most bright value in this map.

5.2.3. CONVERTING FROM HD TO KINECT

The resolution of the Kinect was not sufficient when pointing and gaze are combined ('triangle' configuration). Because in that case both the human and the objects should be in view of the camera and that resulted in that the objects are too small in view of the camera. This makes it very hard to match key points because the eye-tracker has a higher resolution than the Kinect RGB camera. The object in the eye-tracker image has much more detail than the object in the Kinect RGB image. Therefore it was decided to place a HD camera on top of the Kinect to solve the resolution problem. In order to be able to combine the different saliency map, a transformation from HD camera to Kinect camera is required.

The first step is to obtain the intrinsic parameters of the HD camera, which is done by the standard calibration procedure using a chessboard pattern. The calibration board is located at different positions and orientations in the view of the camera. The intrinsic parameters are obtained by using the camera calibration toolbox for Matlab (Bouguet [38]). Having the intrinsic parameters at our disposal, stereo calibration can be performed to get the extrinsic parameters. The extrinsic parameters are a translation and a rotation between the two camera reference frames.

Then, the HD pixel coordinates should be projected to HD camera coordinates. The following equation projects the camera coordinates to image coordinates:

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = A \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} \quad (5.19)$$

where A is a matrix containing all intrinsic parameters. These parameters are f_x and f_y , which are the focal lengths of the camera. And u_0 and v_0 are the values for the principal points. Since the pixels should be transformed from HD camera to Kinect camera, the equation should be written in the following form:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}_{HD} = A^{-1} \cdot s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}_{HD} \quad (5.20)$$

In order to obtain the camera coordinates the depth s should be known. This value is estimated by averaging the distance from all the objects in the scene as is computed in section 4.2. Then, the camera coordinates in the Kinect reference can be computed using the extrinsic parameters R , T , which are the rotation matrix and the translation vector respectively.

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}_{kinect} = R \cdot \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}_{HD} + T \quad (5.21)$$

Then the last step is to project the world camera coordinates back to pixel values using equation 5.19.

5.3. CONCLUSION

The pointing direction is estimated based on the hand location estimated by the OpenNI tracker. After that, the point cloud is filtered to obtain the hand cloud of the pointing finger. The next step is to remove the hand palm and to locate the pointing finger. The eigenvector belonging to the largest eigenvalue corresponds to the pointing direction.

Using the pointing direction, a pointing map should be created that has high for objects in the pointing direction and low values for objects outside the pointing direction. Since the point cloud containing all objects has been computed in chapter 4, so it can be used to efficiently create the pointing map. The vector from the finger to the every point belonging to an object is calculated. The angle between this vector and the pointing direction is computed. This can be represented by a normal distribution with an average of 0° .

An eye-tracker is a device that is able to determine eye movements and gaze location. This means that it is able to pinpoint the gaze location in the image of the eye-tracker. However, the robot should now where the human is looking at therefore the point should be transformed. For that a hybrid system has been developed that is able to create a probability map based on key point matches for textured objects and based on color for uniform coloured objects. Note that for this purpose a HD camera has been attached on top of the Kinect, because the resolution of the Kinect was not sufficient. This means that both cameras are stereo calibrated such that the extrinsic parameters are known.

6

INTEGRATION

In previous chapters, the four cues of the Joint Visual Attention Model are elaborated on. For each cue, a map is created and in this chapter it is explained how these maps are merged to create a model that combines both the bottom up cues with the top down cues. Section 6.1 gives an overview of the software architecture and section 6.2 explains how the bottom up and top down cues are integrated to obtain the combined saliency map. Finally, in section 6.3 the object of interest is determined based on the most bright point in the combined saliency map.

6.1. SOFTWARE ARCHITECTURE

Figure 6.1 gives a schematic of the software. Clearly visible are the three sensors at the top of the schematic and at the bottom the object of interest can be found. In between are several nodes that compute probability maps for every cue. A ROS node is a process that performs a certain computations. They are communicating with each other using ROS topics. A node subscribes to a certain topic if it wants to receive ROS message. It can also publish messages on a certain ROS topic. All ROS nodes are running at a rate of 5 hz except for the segmentation node. This is done to reduce computing power, because the objects will not be moving in this case so it is not interesting to segment at higher rate.

The eye-tracker software, which estimates the gaze direction of the human wearing the device, is adapted to be used in ROS. Now, the eye-tracker publishes the computed gaze location in ROS. This location is used in the 'create gaze map' node to compute the gaze map. Also the HD camera and Kinect are used as an input. Key points are matched between the eye-tracker image and the HD camera image. After that the key points are converted to the Kinect image, because all saliency maps have a size of 640×480 pixels.

Then the pointing map is computed in the following way. First the 'estimate pointing direction node' determines the pointing direction using the point cloud from the Kinect and the coordinates of the OpenNI tracker. The pointing direction and the point cloud containing all objects are used to create the pointing map.

The depth map is obtained by the segmentation node. It uses the point cloud and computes a mask containing all objects. Furthermore, a point cloud with all objects is calculated which will be used by the 'create pointing map node' and by the integration node.

The saliency detection node computes a bottom up saliency map based on the RGB images from the Kinect.

All four maps are integrated in the integration ROS node. The combined saliency map is calculated as will be explained in section 6.2 and after that the object of interested in segmented which will be covered in section 6.3.

6.2. INTEGRATION OF BOTTOM-UP AND TOP-DOWN CUES

In figure 6.2, examples of the four cue maps can be found. These cues need to be integrated into one saliency map. For that purpose, a Hadamard product is used (Schauerte and Fink [11]). For the two matrices A and B with the same size, the Hadamard product is equal to:

$$(A \circ B)_{i,j} = (A)_{i,j} \cdot (B)_{i,j} \quad (6.1)$$

In this case the matrices are the saliency maps that have the dimension 640×480 . In all cases, the bottom up

Figure 6.1: Schematic overview of software. The three sensors can be found on the top with rounded boxes. Each ROS node is represented by a rectangular box. The diamond shape boxes represent the four maps.

Figure 6.2: In this figure the four saliency maps are showed. From left to right the images represent the Gaze map (G), Pointing map (P), Depth map (D) and bottom-up Saliency map (S).

saliency map (S) and the depth map (D) are available, but the gaze (G) and the pointing (P) probability map depend on the configuration of the human, robot and object. If the human and robot are standing opposite of each other, gaze is not available since the viewing angle differs too much resulting in that the robot and human are looking at a different side of the object. In this case, the combined saliency map is obtained by:

$$(C)_{i,j} = (P)_{i,j} \cdot (S)_{i,j} \cdot (D)_{i,j} \quad (6.2)$$

In other words, all the available saliency maps contribute to the value in the combined saliency map by means of entry wise multiplication. The advantage of using the Hadamard product is that objects outside the pointing direction will have a zero value in the combined saliency map. Furthermore, entry-wise multiplication makes the algorithm invariant to background properties because these have a zero value in the combined saliency map.

If the human and robot are standing next to each other, the robot cannot analyse the pointing gesture of the human, since it is outside its view. This means that the pointing map is not available. Then, the combined saliency map is a multiplication of the Gaze map (G), bottom-up Saliency map (S) and the Depth map (D):

$$(C)_{i,j} = (G)_{i,j} \cdot (S)_{i,j} \cdot (D)_{i,j} \quad (6.3)$$

When the bottom up saliency model (S), the depth map (D), the pointing probability map (P) and the gaze probability map (G) are available, it is not recommended to point wise multiply all four cues because the gaze map is not always reliable. This is caused by two reasons. First, the gaze position is not always located at the

object of interest sometimes it has a small offset. The second reason is that the system should deal with two different RGB cameras. This means that the colour perception differ between the cameras. Sometimes the gaze map is empty due to lack of pixels in the robot view with the colour of the object of interest according to the eye-tracker image. Therefore it is better to add up all available maps. However, the pointing map is already accurate so all maps are only added for pixels that have a non-zero value in the pointing map. This can be accomplished by adding all maps and multiply it by the mask $(P)_{i,j} > 0$:

$$(C)_{i,j} = ((S)_{i,j} + (D)_{i,j} + (P)_{i,j} + (G)_{i,j}) \cdot ((P)_{i,j} > 0) \quad (6.4)$$

6.3. OBJECT SEGMENTATION

In the chapter 4.2, it is explained how the object clusters are obtained by detecting the planes in the scene and remove them. The remaining points belong to the objects are present in the scene. For every object, the point cloud is obtained.

For every object, the centroid is calculated and transformed to pixel coordinates. The object cluster with the smallest euclidean distance from centroid in 2D to the location of the maximum value in the combined saliency map is the object of interest. Using this cluster, the object can be segmented and a result can be found in figure 6.3.

Figure 6.3: A drinking bottle is segmented by means of the result of euclidean clustering. The green dot represents the most salient point in the saliency map.

6.4. CONCLUSION

This chapter explains how the pointing map, gaze map, bottom up saliency map and depth map can efficiently be integrated into a combined saliency map. Depending on the setup, three or four maps are available to be integrated. If the pointing map or gaze map is not available, it is most efficient to point wise multiply all four maps. If all four cues are available the approach is different, because the gaze map is less reliable. Therefore all maps are point wise added and multiplied by the pointing map mask.

7

PERFORMANCE

In this chapter, the performance of the system is evaluated. The joint visual attention model should work in all three set-ups; the human and robot are standing opposite of each other, the human and robot are standing next to each other and the the human is looking and pointing to the side. The performance is tested by computing the detection rate of the experiment (Schauerte *et al.* [7]). Since the system is detecting an object of interest every frame, the only two options are that the object is detected correctly or incorrectly.

Table 7.1 summarises all variables for the experiments. The first column correspond to the number of the experiment and to the section of chapter 7 where the results can be found.

Table 7.1: This table summarises all the different experiments. From left to right the setup, available cues, number of objects, number of subjects and illumination conditions are listed. The available cues are Pointing (P), Gaze (G), Saliency (S) and Depth (D).

	Setup	Cues	# Object	# Subjects	Type of objects	Illumination conditions
1	'opposite'	P,S,D	90	10	textured & coloured	good
2	'next to'	G,S,D	90	4	textured & coloured	good
3	'triangle'	P,G,S,D	90	10	textured & coloured	good
4	'triangle'	P,G,S,D	20	1	textured & coloured	good
5	'next to'	G,S,D	54	6	textured & coloured	challenging
6	all	all	9	1	textured & coloured	good
7	'triangle'	P,S,D	25	1	textured	good

In the first setup (see figure 7.1), the human is in view of the robot, so the pointing direction can be estimated. Since the viewing direction of the robot and the human are opposite, it is hard to use gaze for object detection. This setup is tested in section 7.1. When the robot and the human are standing next to each other, the robot is not in view of the camera of the robot so only depth, saliency and gaze can be used to determine the object of interest. In section 7.2, the results of these test can be found. In the last setup, both gestures and gaze are available so both are combine to find the object of interest in section 7.3.

Figure 7.1: The setups 'opposite', 'next to', and 'triangle' are shown here from left to right.

In section 7.4, the influence of the distance between the objects is evaluated. So, the detection rate is computed as a function of distance between the objects. Furthermore, in section 7.5 it is investigated how the joint visual attention model is working when the light conditions are challenging. In section 7.6, the detection rate is computed for different combinations of the pointing map, gaze map, bottom-up saliency

map and depth map. Finally, the performance of the joint visual attention model is evaluated when the background is textured in section 7.7.

7.1. CASE: OPPOSITE

Opposite means that the robot and the human are standing opposite of each other with the object between them. In this experiment, it is evaluated if the system works for different subjects because the geometry of the hand differs between humans. Each of the ten subjects has to point at nine different objects. The order of pointing is the same for all subjects, because in that way the results can be easily compared. In table 7.2, it can be seen that 93.3% of the objects are detected. This means that 84 objects out of the 90 objects are detected correctly.

In some cases, the OpenNI-tracker was not tracking the location of the hand accurate enough and therefore the algorithm is not able to estimate the pointing direction correctly. In some other cases the most salient point was located at the edge of an object. It sometimes chooses the wrong object as most salient object because the 2D center of that object is closer to the most salient point.

Typical results are presented in figure 7.2 and 7.3. Figure 7.3 shows that the tennis ball is the most salient object in the scene, because the tire repair box and the blue coffee cup has lower values in the combined saliency map. Both images also show that the system works in two different pointing strategies, figure 7.2 shows that it is working when the human is leaning forward to indicate the object and figure 7.3 is showing when the human is pointing from a larger distance.

Table 7.2: Detection rate of the system in the case where the human and robot are facing each other. In this case Pointing (P), Saliency (S) and Depth (D) are combined to find the object of interest.

Setup	Cues	# Objects	Detection rate
Opposite	P, S, D	90	93.3 %

Figure 7.2: The subjects is pointing at the blue coffee cup as indicated in the left image. The combined saliency map is located in the middle and at the right hand side the segmented object is found. The standard deviation of the pointing probability map is equal to 15° . The green dot in the saliency map and the RGB image correspond to the most salient location in the combined saliency map.

Figure 7.3: The subjects is pointing at the tennis ball as indicated in the left image. The combined saliency map is located in the middle and at the right hand side the segmented object is found. The standard deviation of the pointing probability map is equal to 15° . The green dot in the saliency map and the RGB image correspond to the most salient location in the combined saliency map.

7.2. CASE: NEXT TO

The pointing direction is not available when the human and the robot are standing next to each other. In this case the information from the eye-tracker is combined with the depth information and bottom up saliency. In these experiments, ten recordings are performed by four different subjects. Each subject is looking for two seconds to each of the nine objects.

The hybrid model switches between the color based approach and the texture based approach depending on the texture of the object. If more than 15 key point matches found the model chooses the texture based approach otherwise the color based approach is chosen. In figures 7.5 a result can be found when the subject is looking at the tennis ball (see figure 7.4). In this case color based approach is used to compute the gaze map. When the subject is looking at the pack of sugar as can be seen in figure 7.6, the gaze map is created by means of the texture based approach. An example gaze map can be found in figure 7.7.

Table 7.3 shows the results. The detection rate for textured objects is equal to 96.0% and for the uniform coloured objects it is equal to 32.5%. The overall detection rate equals 67.8%. This means that the system works very well for textured objects, but for uniform coloured objects the performance depend on the lighting conditions. Because the robot and the eye-tracker have different RGB cameras, the colour perception deviates slightly between the two cameras. Furthermore, based on colour it is harder to find a object because there are multiple objects with the same colour, so in that case bottom up saliency determines the objects of interest.

Note that it is important that the eye-tracker does not move on the subject's head after calibration because it affects the accuracy of the eye-tracker. Furthermore if the subject is not looking at the object the system is not able to pinpoint the object of interest.

Table 7.3: Detection rate of the system in the case where the human and robot are next to each other. In this case Gaze (G), Saliency (S) and Depth (D) are combined to find the object of interest.

Setup	Cues	# Objects	Detection rate
Next to	G, S, D	50 textured objects	96.0 %
Next to	G, S, D	40 uniform coloured objects	32.5 %
Next to	G, S, D	90	67.8 %

Figure 7.4: Subject is looking at the tennisball as is indicated by the red dot.

Figure 7.5: The view of the robot is presented at the left hand side. The green dot represent the most salient point in the combined saliency map. This map, which is a combination of the Gaze map (G), Bottom up Saliency (S) and Depth (D), is located in the middle and at the right the segmented object can be found. In this case color based approach is used to compute the Gaze map.

Figure 7.6: Subject is looking at the pack of sugar as is indicated by the red dot.

Figure 7.7: The view of the robot is presented at the left hand side. The green dot represent the most salient point in the combined saliency map. This map, which is a combination of the Gaze map(G), Bottom up Saliency(S) and Depth(D), is located in the middle and at the right the segmented object can be found. In this case texture based approach is used to compute the Gaze map.

7.3. CASE: TRIANGLE

In this case, Pointing (P), Gaze (G), Saliency (S) and Depth (D) are integrated to pinpoint the object of interest. In order to combine all four cues, the human should be in view of the camera and the human and the robot should have approximately the same viewing angle. This means that the human, robot and object are positioned in a triangular shape. The human is standing in view of the robot and is pointing and looking sideways. Here the system is tested for ten subjects and each of those subjects are pointing and looking for two seconds to each object.

In figure 7.8 and 7.9, it can be seen that the subject is pointing and looking at the pack of sugar. The combined saliency map is an addition of all four maps (see figure 7.10). But it is only added for points in the pointing map, since the pointing map is more reliable than the gaze map. In figure 7.9 it is clear that the maximum point (green dot) is located at the pack of sugar, resulting in that the pack of sugar is detected correctly. Without using the gaze map, the system finds sometimes the edge of the table as object of interest. The gaze map pulls the most salient point to the pack of sugar.

Figure 7.8: The red dot indicates where the subjects is looking at.

In figure 7.11 and 7.12 the subject is showing that the tire repair box is the object of interest. At the right image in figure 7.12 the result is clearly visible. In the pointing map, which can be found in figure 7.13, the tire repair box, pack of sugar, red coffee cup and the edge of the table are present as possible objects of interest. The tire repair box is already the most salient object but by adding the gaze map this object is even more salient. So, by adding the gaze map, the system is more convinced that the tire repair box is the most interesting object.

Table 7.4 shows the detection rates of the total system. The detection rate of the system without using the gaze map is equal to 91.1% and if the gaze map is added the detection rate increases to 96.7%.

Figure 7.9: From left to right: the view from the robot, combined saliency map and the found object of interest. The green dot indicates the most salient point in the saliency map.

Figure 7.10: All four saliency maps from left to right: Gaze map (G), Pointing map (P), Depth map (D) and bottom up Saliency map (S).

Figure 7.11: The red dot indicates where the subjects is looking at.

Figure 7.12: From left to right: the view from the robot, combined saliency map and the found object of interest. The green dot indicates the most salient point in the saliency map.

Figure 7.13: All four saliency maps from left to right: Gaze map (G), Pointing map (P), Depth map (D) and bottom up Saliency map (S).

7.4. CASE: DISTANCE

The next experiment intends to look at the accuracy of the system as function of the distance between the objects. In this experiment five objects are located at 1.2 m from the robot and the distance between the

Table 7.4: Detection rate of the system in the case where the human and robot are facing each other. In this case Pointing (P), Gaze (G), Saliency (S) and Depth (D) are combined to find the object of interest.

Setup	Cues	# Objects	Detection rate
Triangle	P, S, D	90	91.1 %
Triangle	P, G, S, D	90	96.7 %

object is 6 cm. The experiment is repeated for 4 cm and 2 cm. For each distance, the subject indicates four times at the five objects so in total twenty objects should be detected. Figures 7.14 and 7.15 show that the pack of sugar is detected in both cases correctly.

Clearly visible is that the pack of sugar and the drinking bottle are both visible in the pointing map, but due to the gaze map the attention is attracted to the pack of sugar. In this case, the pointing map is created with a standard deviation of 10° and the gaze map is created with a window size of 100 pixels and a minimal amount of key points matches is equal to 10.

Table 7.5 shows the results. As you can see, the system is working very well, because the detection rate only lowers 5% when the distance between the objects decreases to 2 cm. With a distance of 2 cm however, the system once in a while merge two neighbouring objects because that is the limit of the segmentation algorithm.

Figure 7.14: The distance between the objects is equal to 6 cm and the subject is indicating the pack of sugar.

Figure 7.15: The distance between the objects is equal to 2 cm and the subject is indicating the pack of sugar.

Table 7.5: Detection rates at different distances between the objects.

Setup	Cues	# Objects	Detection rate
2 cm	P, G, S, D	20	95.0 %
4 cm	P, G, S, D	20	100.0 %
6 cm	P, G, S, D	20	100.0 %

7.5. CASE: BAD LIGHTNING CONDITIONS

Since the system uses two RGB cameras to create the gaze map, it is important to check if what is happening when the light conditions are bad. In that case, the illumination is not equally distributed over the scene and shadows are present. In total six different subjects have performed a test in daylight.

Table 7.6 shows the performance of the system in the 'next to' setup. So due to bad lightning conditions the detection rate decreases from 67.8% to 48.2%. Figure 7.17 shows the result when the subject is looking at the red coffee cup (see figure 7.16).

Figure 7.16: The red dot indicates where the subjects is looking at.

Figure 7.17: From left to right: the view from the robot, combined saliency map and the found object of interest. The green dot indicates the most salient point in the saliency map.

Table 7.6: Detection rate of the system in the case where the human and robot are next to each other. In this case Gaze (G), Saliency (S) and Depth (D) are combined to find the object of interest.

Setup	Cues	# Objects	Detection rate
Next to	G, S, D	30 textured objects	90.0 %
Next to	G, S, D	24 uniform coloured objects	33.0 %
Next to	G, S, D	54	48.2 %

7.6. CASE: INFLUENCE DIFFERENT SALIENCY MAPS

In this experiment the influence of each map is investigated. For each of the three setups, four different combinations of saliency maps are used to detect the object of interest. These results are obtained by looking at one experiment for every setup.

The minimal combination of cues is the combination of the bottom up cues, being the depth map and bottom up saliency. If only depth is used, all objects are equally salient because the depth map is a mask. If only the saliency map is used, we might find a pixel salient that does not belong to an object. Furthermore, the object cannot be segmented because the object clusters are not available. Based on the local distinctive features (bottom-up saliency and depth) one object of the nine objects is the most salient object at the first focus of attention. Table 7.7 shows that in all cases the detection rate is 11.1% for the combination of S and D.

In the triangle setup, it can be seen that if the G, S and D are combined that the detection rate is 33.3%. This is not very high, because the viewing angles differ a lot therefore it is difficult to match key points. If P, S, and D are combined the result is 77.8%. However if you combine P, G, S and D the detection rate is 100.0%. This proves that integrating all four cues is a very powerful way of detecting objects of interest.

Table 7.7: Detection rates for the three different setups for four different combinations of saliency maps. The detection rate is shown in different combination of Pointing (P), Gaze (G), Saliency (S) and Depth (D).

	S,D	G, S, D	P, S, D	P, G, S, D
Opposite	11.1 %	x	100.0 %	x
Next to	11.1 %	77.8 %	x	x
Triangle	11.1 %	33.3 %	77.8 %	100.0 %

7.7. CASE: TEXTURED BACKGROUND

In this experiment, five objects are located at the table as can be seen in figures 7.18 and 7.19. The background and the table are covered by a coloured blanket in order to test to what extend the background is influencing the results. The subject is pointing and looking at the five objects. This experiment has been repeated five times, so in total 25 objects should be detected.

It resulted in a detection rate of 100%. The background will influence the gaze map slightly because more key points will be found due to the textured background. However the pointing map is also available therefore the object will be detected correctly. In figure 7.19 a result can be found when the subject is pointing at the drinking bottle. The gaze map shows a high peak at the location of drinking bottle and some smaller peaks. Combining this with the pointing map, bottom up saliency map and depth results in a correct segmentation of the object of interest.

Figure 7.18: The gaze location is indicated by the red dot, that is located at the yellow drinking bottle.

Figure 7.19: From left to right: the view from the robot, combined saliency map and the found object of interest. The green dot indicates the most salient point in the saliency map.

Figure 7.20: All four saliency maps from left to right: Gaze map (G), Pointing map (P), Depth map (D) and bottom up Saliency map (S).

7.8. CONCLUSION

Experiments show that the Joint Visual Attention model is a robust and flexible system, because it performs well in the three different setup. Except for the 'next to' setup the detection rate is higher than 90%. Table 7.8 shows the detection rate percentages. When the human and robot are opposite of each other and pointing, bottom up saliency and depth are combined, the detection rate is 93.3%. In the setup 'next to', pointing cannot be used since the robot cannot see the human however gaze can be used. In this case the detection rate is 67.8%. When the robot, human and object are standing in a triangular shape, the performance of the model is the highest of all setups because all cues are combined in this case. Table 7.8 shows that the detection rate is 96.3%.

Table 7.8: Detection rate of the system in different configurations. The first column shows the test configuration. The second column shows the available cues for object detection, where the Pointing map, Gaze Map, bottom-up Saliency and Depth Map are indicated by P, G, S and D, respectively. In the third column, the total amount of tested objects is shown and in the last column the detection rate is presented.

Setup	Cues	# Objects	Detection rate
Opposite	P, S, D	90	93.3 %
Next to	G, S, D	90	67.8 %
Triangle	P, S, D	90	91.1 %
Triangle	P, G, S, D	90	96.7 %

Due to combining the different cues, the system is accurate when objects are located closer together. The detection rate drops only 5.0% when the objects are moved from 6 cm to 2 cm. Moving them even further together would result that it would be recognized as one object.

In section 7.5, the model has been tested in bad lightning conditions. The performance in the 'next to' setup dropped due to bad lightning conditions from 67.8% to 48.2%. However, if this is combined with a pointing map the decrease in performance will be significantly lower, because it will highly rely on the pointing map which is not influenced by bad lightning conditions.

Furthermore, in section 7.6, it is clear that integrating all maps is the most efficient way of creating a joint visual attention model. In section 7.6 it can be seen that if the G, S and D are combined that the detection rate is 33.3%. If P, S, and D are combined the result is 77.8%. The performance of the model is 100% when combining all available cues.

Then, the system has been tested to check the influence of the background on the performance. The gaze map is slightly influenced because more key points and key point matches are found. Due to the reliable pointing map the correct object is found in all cases.

8

CONCLUSION AND RECOMMENDATIONS

In chapter 2, existing joint visual attention models are elaborated on. These existing models determine the object of interest when the human is in view of the camera. Furthermore, an eye-tracker has not been used in existing joint visual attention models.

The model should work in three different setups: human and robot are facing each other, the human and robot are next to each other and the robot, human and the object are standing in a triangular shape. If the human and the object are standing in a triangular shape, the object and human are both in view of the camera and the human is pointing sideways to indicate the object of interest. In this case all four cues can be used, but it should also work when the gaze map or pointing map is not available. The assumption is made that the objects are standing on a table. Furthermore, the human is wearing the eye-tracker and the robot is equipped with a Kinect camera. On top of the Kinect a HD camera is attached.

Then chapter 4 elaborates on bottom up cues. The model of Itti *et al.* [1] is used for bottom up saliency detection. It computes conspicuity maps by means of center-surround differences for each of the three features: color, intensity and orientations. Combining these three conspicuity maps by averaging gives the bottom up saliency map. Furthermore, a mask can be created with objects by first detecting all the planes in the scene and remove them out the point cloud. After that the remaining points are clustered to find the objects. Finally, all objects are transformed from 3D to 2D to create a mask containing all objects.

In chapter 5, a pointing map is created by means of the angle between the pointing direction vector and the vector from the pointing finger to the object. The angle is computed for every point belonging to an object and a normal distribution determines the value in the pointing map. A hybrid system has been developed to create a probability gaze map. Based on the texture of the object, the hybrid model switches between texture based approach and color based approach.

The four maps are integrated to obtain a combined saliency map. If the pointing map or gaze map is not available, it is most efficient to point wise multiply all four maps. When all four maps are available the combined saliency map is obtained by adding all four maps and multiply it with the pointing map mask.

The performance has been tested in the three setups in chapter 7. When the human and robot are opposite of each other and pointing, bottom up saliency and depth are combined, the detection rate is 93.3%. If the human is standing next to the robot, the gaze map, bottom up saliency map and depth map are combined, then the detection rate is 67.8%. If robot, human and object are standing in a triangular shape, the detection rate is equal to 96.3% because all four maps can be combined.

Furthermore, the influence of the distance between objects is investigated. The detection rate drops only 5.0% when the objects are moved from 6 cm to 2 cm apart. In bad light conditions, the model shows a drop from 67.8% to 48.2% in the setup where the human and robot are standing next to each other. Finally, it has been proven in chapter 7 that combining all four maps is the most efficient way of determining the object of interest.

The main contribution of this work is the development of a flexible joint visual attention. In comparison to existing models this model can work in different arrangements of human, robot and object. It is able to detection object of interest if the robot and human are facing each other, standing next to each other and the configuration where the human is in the view of the camera and looking sideways.

Furthermore, another novelty of this model is the usage of an eye-tracker for a joint visual attention

model. A hybrid model has been developed that is able to create a probabilistic gaze map. Depending on the amount of key point matches found, the model chooses key points or back projection to generate this gaze map. Moreover instead of using 2D information, in this work 3D information is used to create a pointing probability map.

The current non-verbally joint visual attention model can be improved, therefore some recommendations are made below. Verbal components can be added to improve the model even more. For example, the person can describe the object and the speech model creates a saliency map based on features such as color and shape. A better option to use speech is to add inhibition of return. In that way, the human verifies if the object of interest is correct. For example the robot is asking: "Do you mean the yellow drinking bottle?" and the human can answer "Yes" or "No". If the robots has indicated the wrong object of interest, the robot will look at the second most salient object.

Instead of attaching a HD camera on top of the Kinect, it is convenient to use the Kinect 2.0. The resolution of this Kinect camera is 1280×960 pixels which is much better than the 640×480 pixels of the Kinect 1.0. Furthermore, the system consists of an eye-tracker world camera and the HD camera, but both cameras have a different color perception. Therefore it is advisable to color calibrate both cameras because the color differences between both cameras will be much smaller. This can be done by a colorimetric calibration procedure.

If all four maps are available it will be efficient to reduce the area for matching key points and back projection by means of the pointing map mask. This will speed up the process of creating a gaze map and it will result in a more reliable map because objects outside the pointing direction will not be considered. The gaze map can be determined accurately if the difference in viewing angle is small. If the viewing angle becomes larger, it is recommended to use the uniform color based approach. In this case back projection is used to create a gaze map and it depends less on differences in viewing angle. Apart from the descriptors texture and color, shape can also be used to create a gaze map. Instead of a hybrid model that based on texture of the object determines which approach is used to create the gaze map, a model can be created that merges three feature maps based on texture, color and shape.

In order to improve the pointing map, a faster pointing direction estimation algorithm can be used that has been optimized. Another option is to optimize the current code by programming it more efficiently. Since the OpenNI tracker is not always accurate, a hand tracker model can be developed that uses the hand location of the OpenNI tracker and centroid of the previous frame. Furthermore, a faster implementation can be obtained by splitting the current node into two nodes. The first node filters the cloud and publishes the hand cloud and the second node determines the pointing direction of the hand cloud. In this way both nodes can run parallel. Since the model is used for object detection, the pointing direction does not need to be computed continuously. Since the human is pointing for 2 seconds at an object, the pointing direction could only be estimated if the hand is not moving for 1 second for example. The last recommendation to improve the pointing map is to adapt to algorithm to create the pointing map that it also takes into account the distance from finger to the object. So if a object is closer to the finger it should have a higher value than if an object is further away.

If objects are partly standing behind each other, the system sometimes finds the most salient point at the edge of both objects. In that case, the algorithm designates the wrong object as most salient object in a number of cases, because the distance to the centroid in 2D is smaller. This can be solved by changing the depth map such that it focusses more on the centroid of the objects. It is recommended to use, for instance, a Gaussian function at the location of the centroid and multiply it by the depth mask. In this case, the system tends to shift its attention more to the centroids of the objects, so the most salient point be guided away from the edge of the objects.

If all these recommendations are implemented, the system will be a viable solution for naturally robot-human interaction in domestic environments. Especially when a wireless eye-tracker will be used, because that makes the system more user-friendly. In that case, the human can move more freely instead of being connected with cables to the robot.

BIBLIOGRAPHY

- [1] L. Itti, C. Koch, and E. Niebur, *A model of saliency-based visual attention for rapid scene analysis*, (1998).
- [2] Z. Yucel, A. A. Salah, C. Mericli, T. Mericli, R. Valenti, and T. Gevers, *Joint attention by gaze interpolation and saliency*, IEEE Transactions on cybernetics **43**, 3 (2013).
- [3] P. Viola and M. Jones, *Robust real-time object detection*, in *International Journal of Computer Vision* (2001).
- [4] J. Xiao, T. Moriyama, T. Kanade, and J. Cohn, *Robust full-motion recovery of head by dynamic templates and re-registration techniques*, International Journal of Imaging Systems and Technology **13**, 85 (2003).
- [5] Y. Freund and R. E. Schapire, *A short introduction to boosting*, in *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (Morgan Kaufmann, 1999) pp. 1401–1406.
- [6] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)* (The MIT Press, 2005).
- [7] B. Schauerte, J. Richarz, and G. Fink, *Saliency-based identification and recognition of pointed-at objects*, IEEE/RSJ International Conference Robots and Systems, October 18-22, Taipei, Taiwan (2010).
- [8] Y. Jia, S. Li, and Y. Liu, *Tracking pointing gesture in 3d space for wearable visual interfaces*, in *Proceedings of the International Workshop on Human-centered Multimedia* (2007) pp. 23–30.
- [9] X. Hou and L. Zhang, *Saliency detection: A spectral residual approach*, IEEE Conference on Computer Vision and Pattern Recognition **23**, 1 (2007).
- [10] J. Matas, O. Chum, M. Urban, and T. Pajdla, *Robust wide baseline stereo from maximally stable extremal regions*, in *Proc. BMVC* (2002) pp. 36.1–36.10, doi:10.5244/C.16.36.
- [11] B. Schauerte and G. Fink, *Focusing computational visual attention in multi-modal human-robot interaction*, (2010).
- [12] V. Navalpakkam and L. Itti, *Attentional modulation of tuning width, preferred features and gains during visual search*, in *Proc. Vision Science Society Annual Meeting (VSS07)* (2007).
- [13] C. Guo, Q. Ma, and L. Zhang, *Spatio-temporal saliency detection using phase spectrum of quaternion fourier transform*, (2008).
- [14] A. Borji, D. Sihite, and L. Itti, *Quantitative analysis of human-model agreement in visual saliency modeling: A comparative study*, IEEE Transactions on image Processing **22**, 55 (2013).
- [15] D. Walther and C. Koch, *Modeling attention to salient proto-objects*, (2006).
- [16] R. A. Rensink, *The dynamic representation of scenes*, Visual Cognition **7**, 17 (2000).
- [17] X. Hou and L. Zhang, *Dynamic visual attention: Searching for coding length increments*, (2008).
- [18] B. Olshausen *et al.*, *Emergence of simple-cell receptive field properties by learning a sparse code for natural images*, Nature **4**, 607 (1996).
- [19] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, *Frequency-tuned salient region detection*, (2009).
- [20] A. Garcia-Diaz, X. Fdez-Vidal, X. Pardo, and R. Dosi, *Saliency from hierarchical adaption through decorrelation and variance normalization*, Image and Vision Computing (2012).
- [21] M. Rudinac and P. P. Jonker, *Saliency detection and object localization in indoor environments*, (2010).

- [22] E. Parvizi and Q. Wu, *Multiple object tracking based on adaptive depth segmentation*, IEEE computer society , 273 (2008).
- [23] A. Bleiweiss and M. Werman, *Fusing time-of-flight depth and color for real-time segmentation and tracking*, Lecture Notes in Computer Science , 58 (2009).
- [24] D. Comaniciu and P. Meer, *Mean shift: A robust approach toward feature space analysis*, IEEE Transactions on Pattern Analysis and Machine Intelligence **24**, 603 (2002).
- [25] M. van den Bergh, D. Carton, R. de Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlentz, D. Wolherr, L. van Gool, and M. Buss, *Real-time 3d hand gesture interaction with a robot for understanding directions from humans*, (2011).
- [26] Y. Song, Y. Gu, P. Wang, Y. Liu, and A. Li, *A kinect based gesture recognition algorithm using gmm and hmm*, 2013th 6th International Conference on Biomedical Engineering and Informatics (2013).
- [27] E. D. Guestrin and M. Eizenman, *General theory of remote gaze estimation using the pupil center and corneal reflections*, IEEE Transactions on Biomedical Engineering **53**, 1124 (2006).
- [28] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchin, R. Moore, A. Kipman, and A. Blake, *Real-time human pose recognition in parts from single depth images*, Communication of the ACM **56**, 116 (2013).
- [29] K. Khoshelham and S. O. Elberink, *Accuracy and resolution of kinect depth data for indoor mapping applications*, Sensors 2012 **12**, 1437 (2012).
- [30] C. Koch and S. Ullman, *Shifts in selective visual attention: towards underlying neural circuitry*, Human Neurobiology **4**, 4700 (1985).
- [31] J. van der Weem and O. Akman, *Ros c++ implementation of saliency detection algorithm of itti et al(1998)*, .
- [32] A. J. B. Trevor, S. Gedikli, R. B. Rusu, and H. I. Christensen, *Efficient organized point cloud segmentation with connected components*, (2013).
- [33] G. Gallagher, *Ros electric package finger detection*, .
- [34] H. Bay, T. Tuytelaars, and L. V. Gool, *Surf: Speeded up robust features*, (2006).
- [35] M. Muja and D. G. Lowe, *Fast approximate nearest neighbors with automatic algorithm configuration*, in *International Conference on Computer Vision Theory and Application (VISSAPP'09)* (INSTICC Press, 2009) pp. 331–340.
- [36] C. Rother, V. Kolmogorov, and A. Blake, *Grabcut: Interactive foreground extraction using iterated graph cuts*, ACM Trans. Graph. **23**, 309 (2004).
- [37] M. J. Swain and D. H. Ballard, *Indexing via color histograms*, Third international conference on computer vision (1990).
- [38] J.-Y. Bouguet, *Camera calibration toolbox for matlab*, .