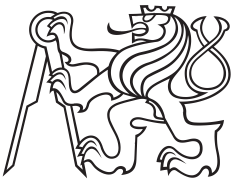
Bachelor Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Gaze Control and Stabilization for a Humanoid Robot Using Neck Joints

Zuzana Jindrová

Supervisor: doc. Mgr. Matěj Hoffmann, Ph.D.
Supervisor–specialist: Ing. Jakub Rozlivek
Field of study: Cybernetics and Robotics
May 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Jindrová Zuzana** Personal ID number: **507452**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Gaze Control and Stabilization for a Humanoid Robot Using Neck Joints

Bachelor’s thesis title in Czech:

Řízení a stabilizace pohledu humanoidního robota pomocí krčních kloubů

Guidelines:

The goal of the project is to develop a neck controller for a humanoid robot to look at a desired position and compensate for self-induced robot motion (Roncone et al., 2016).
1. Familiarize yourself with the iCub humanoid robot and its vision system and RGB-D camera mounted on its head.
2. Implement a gaze controller for the iCub using the neck joints only to control the RGB-D camera pose. Algebraic inverse kinematics is preferred over numerical solutions (Alshakhs et al., 2023).
3. Exploit the redundancy in the neck plant (3 DoF) for the gaze task (2 DoF) to maximize additional criteria (e.g., (He et al., 2022)).
4. Implement gaze stabilization using neck joints only to compensate for robot torso motions (Grotz et al., 2017).
5. Quantitatively evaluate the performance of gaze stabilization using the optical flow index (Habra & Ronsse, 2016) or similar metric.
6. Evaluate the gaze controller in a human-robot interaction scenario (e.g., https://youtu.be/gw8JB-1R3bs).

Bibliography / sources:

[1] Alshakhs, A., Mysorewala, M., Saif, A.-W., & Alshehri, K. (2023). A Novel Algebraic Inverse Kinematics Based Approach to Gaze Control in Humanoid Robots. IEEE Access.
[2] Grotz, M., Habra, T., Ronsse, R., & Asfour, T. (2017). Autonomous view selection and gaze stabilization for humanoid robots. 1427–1434.
[3] Habra, T., & Ronsse, R. (2016). Gaze stabilization of a humanoid robot based on virtual linkage. 163–169.
[4] He, K., Newbury, R., Tran, T., Haviland, J., Burgess-Limerick, B., Kulić, D., Corke, P., & Cosgun, A. (2022). Visibility maximization controller for robotic manipulation. IEEE Robotics and Automation Letters, 7(3), 8479–8486.
[5] Roncone, A., Pattacini, U., Metta, G., & Natale, L. (2016). A Cartesian 6-DoF Gaze Controller for Humanoid Robots. Robotics: Science and Systems, 2016.

Name and workplace of bachelor’s thesis supervisor:

doc. Mgr. Matěj Hoffmann, Ph.D. Vision for Robotics and Autonomous Systems FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Ing. Jakub Rozlivek Vision for Robotics and Autonomous Systems FEE

Date of bachelor’s thesis assignment: **22.01.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

doc. Mgr. Matěj Hoffmann, Ph.D. Supervisor’s signature	prof. Dr. Ing. Jan Kybic Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to express my gratitude to my supervisor, Matěj Hoffmann, for the opportunity to work on this project and for his constructive feedback on my work. I would also like to thank Jakub Rozlivek for his invaluable guidance, advice, and assistance in editing this thesis. Finally, I would like to thank my family for their patience and my boyfriend and classmates for their support and help throughout my studies, which have contributed to the creation of a positive team.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Of the software tools using AI, I used DeepL for translation of this thesis from Czech to English, Writefull for grammar correction and GitHub Copilot as a programming aid in accordance with the framework rules for the use of artificial intelligence at the CTU.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských prací.

Ze softwarových prostředků využívajících AI jsem v souladu s Rámcovými pravidly používání umělé inteligence na ČVUT využila DeepL pro překlad této práce z českého do anglického jazyka, Writefull pro korekci gramatiky a GitHub Copilot jako dopomoc při programování.

V Praze dne 04. května 2024

.....
Zuzana Jindrová

Abstract

For most tasks, humanoid robots use the view of cameras to orient themselves in space, visually control activities, track objects, or interact with humans. For these purposes, active gaze control and gaze stabilization are required. This work proposes a solution for gaze control and stabilization of the iCub humanoid robot using algebraic inverse kinematics. The strength of this work is that it uses only the robot's neck joints to accomplish these tasks, leaving space to use eye movements for secondary tasks like communicative gaze to be perceived human-like by people. The solution to gaze control and stabilization does not require all three available degrees of freedom of the neck. Therefore, we also explored the possibility of using redundancy to accomplish two secondary tasks, *i.e.*, keep two targets in the field of view and maintain a horizontal orientation of the robot head.

Keywords: humanoid robots, iCub, gaze control, gaze stabilization, redundancy, optical flow

Supervisor: doc. Mgr. Matěj Hoffmann, Ph.D.

Abstrakt

Při většinu úkolů využívají humanoidní roboti pohled kamer k orientaci v prostoru, vizuální kontrole nad činností, sledování objektů nebo interakci s člověkem. Pro tyto účely je nutné aktivní řízení a stabilizace pohledu. Tato práce navrhuje řešení pro řízení a stabilizaci pohledu humanoidního robota iCub pomocí algebraické inverzní kinematiky. Výhodou této práce je, že využívá pouze krční klouby robota a ponechává tak prostor pro využití očních pohybů pro sekundární úkoly, jako je oční mimika, aby robot působil přirozeně. Řešení řízení a stabilizace pohledu nevyžaduje všechny tři dostupné stupně volnosti krku. Proto jsme také zkoumali možnost využití redundance pro splnění dvou sekundárních úkolů: udržení dvou cílů v zorném poli a zajištění horizontální orientace hlavy robota.

Klíčová slova: humanoidní roboti, iCub, řízení pohledu, stabilizace pohledu, redundance, optický tok

Překlad názvu: Řízení a stabilizace pohledu humanoidního robota pomocí krčních kloubů

Contents

Chapter 1

Introduction

1.1 Motivation

Nowadays, vision is one of the most important senses for humanoid robots. The ability to "see" allows robots to orient themselves in the space in which they work and to move safely within it. Therefore, humanoid robots are usually equipped with cameras in the eyes and possibly even other external cameras attached to the robot's body. By analyzing images, they can also recognize objects and people, allowing them to respond to visual stimuli and interact with their environment. Humanoid robots are therefore able to perform a variety of tasks, most often replicating common human activities.

Active gaze control allows the robot to look where the activity or situation requires it. The robot is then able to track moving targets or switch view between selected locations. However, the robot does not usually use only gaze to accomplish the task; for example, gazing and grasping often take place simultaneously. The grasping movements lead in most cases to a change in the robot's gaze, *i.e.*, the robot's field of view, which can cause a loss of visual control over the activity or the robot's disorientation. Therefore, gaze stabilization is necessary to keep the robot's view focused on locations or objects relevant to the activity and not affected by the robot's movements.

1.2 Goals

The goal of this thesis is to implement a gaze controller including solutions for gaze control and gaze stabilization tasks for the humanoid robot iCub using its neck joints. Specifically, to ensure that the robot can look at a desired position and compensate for the robot's self-motion. At the same time, an algebraic solution of the inverse kinematics is preferred over a numerical one. Also, for solving gaze control and stabilization tasks, it should be sufficient to use only two degrees of freedom of the neck out of the three available, so we will also explore the possible use of this redundancy in the neck plane and incorporate it into the gaze controller. Then we will compare the gaze stabilization from our proposed gaze controller with a naive gaze stabilization method using an *optical flow* algorithm. We will also evaluate the gaze controller in a human-robot interaction scenario.

1.3 Contributions

The main contribution of this work is a neck controller for the humanoid robot iCub that allows both gaze control and gaze stabilization. The advantage of this controller is that it is completely algebraic and does not use any optimization. With this controller, the robot is able to focus its gaze on a specified location and compensate for self-induced motions using the neck joints. It also provides the use of neck redundancy for two different secondary tasks. The first of the secondary tasks allows for two gaze targets to be specified instead of one, whereby while the gaze is shifted to the first target, the redundant neck joint ensures the visibility of the second target. The other secondary task is to compensate for the rotation of the field of view and to ensure that the gaze is always parallel to the horizon of the environment. The functionality of the gaze controller was tested in simulation and on the humanoid robot iCub. The source code of the controller is available in a GitLab repository [1].

1.4 Structure of the thesis

The structure of this thesis is as follows. First, we present an overview of current state-of-the-art solutions to gaze control and gaze stabilization tasks, the possibilities of using robot redundancy in these tasks, and how gaze stabilization can be evaluated in Chapter 2. Chapter 3 introduces the humanoid robot iCub, on which this thesis was focused, along with an introduction to a naive approach for gaze stabilization and the *optical flow* algorithm for its evaluation. The algebraic solution to gaze control and gaze stabilization is described in Chapter 4. Additionally, this chapter discusses the use of redundancy for the two secondary tasks and presents a description of the gaze controller strategy that combines all the components in this chapter. Chapter 5 contains the results and presentation of experiments performed in simulation and on the real robot. Finally, Chapter 6 provides a discussion, a conclusion, and suggestions for future work.

Chapter 2

Related Work

The robot's view is dynamic and its change depends on the robot's motion and its interactions with the environment. Furthermore, the view is constrained by the specifications of the camera utilized, which requires the development of algorithms to actively control the robot's gaze so that it is able to shift its gaze to specific points in the environment or to scan the environment. Cuevas et al. [2] divide this process into three parts. First, the algorithm identifies the object, *i.e.*, it takes an image of the environment as the input, processes it, and returns the coordinates of the object of interest as the output. In the next part, the controller computes the necessary joint coordinates to keep the object in the visual field. The third part is the execution of the actual motion by the respective mechanisms that receive signals from the controller and directly manipulate the camera view.

Target position estimation. For image processing, trained neural networks are usually used, which are able to find objects of interest in the image [3, 4]. For transformations between object's pixel coordinates and 3D position in the camera frame or robot frame, we need to know the intrinsic parameters of the camera used, such as its focal length and resolution [5], and optionally extrinsic parameters such as the position and orientation of the camera relative to the robot. For tasks requiring physical interaction with the environment, depth plays an important role. The depth is most easily obtained from depth camera data [6, 7]. If the robot does not have a depth camera or its use is unsuitable [8], this information needs to be obtained in other ways. Vargas-Signoret et al. [9] describe how to obtain depth information using two stereo camera images of the robot's eyes.

Gaze control and stabilization methods. Most related to this work is the second part of the gaze control process, calculating the necessary neck joint coordinates to keep the object in the visual field. When working with the gaze of humanoid robots, attention is also paid to make their movements as natural as possible and to make them pleasant for the human interacting with the robot [10]. Many works have investigated both human and animal gaze work for natural movements and attempted to apply these findings to humanoid robots specifically [11]. According to Habra et al. [12] implementation of gaze stabilization and gaze control for robots can be classified into two approaches, (i) bio-inspired approaches based on reflexes [13, 14] and (ii) classical robotic approaches exploiting inverse kinematics [5, 15, 16]. Habra et al. [12] combine the bio-inspired and inverse kinematics approaches and show that each of these approaches is suitable for a different type of perturbation. The combination produces a fast and versatile method for gaze stabilization.

Reflexes used by bio-inspired aproaches. In humans, eye muscles are mainly

responsible for gaze control and the field of view can be controlled as needed by moving them. The most notable eye reflexes are the vestibulo-ocular reflex (VOR) and the optokinetic reflex (OKR) [17, 18]. The VOR is responsible for stabilizing gaze during body movement, using information from the vestibular system of the inner ear about head movements, which are then compensated with countermovements of the eyes [14]. The OKR is activated when tracking a moving target to keep it in the field of view [19]. Both of these reflexes ensure full gaze control and smooth gaze change.

Eye movements and reflexes can be exploited for a human-like gaze when a robot has eye-like cameras, *e.g.*, as shown by Roncone et al. [20] in their work, where they implemented the VOR reflex. However, when robots do not have or use these cameras, *e.g.*, an external camera is attached to the head, the neck is responsible for the gaze control and it is not possible to implement these reflexes.

Gaze inverse kinematics For the inverse kinematics of gaze control and stabilization, both numerical [20, 21] and algebraic solutions exist. Numerical solutions tend to be simple to implement and many solutions exist for them. Moreover, it is easy to incorporate other input data, such as measurements from an inertial measurement unit (IMU), as shown in Roncone et al. [20] for gaze stabalization. However, the main drawbacks of numerical methods include their computational requirements, uncertain number of iterations, and the risk of numerical instability [22]. On the other hand, the algebraic approach requires finding equations that correctly describe and solve the given problem [16, 22, 23], which can be difficult, for example because of the complexity of the robot's kinematic chain. However, if we find these equations, it is straightforward to solve them (if the problem is feasible). Another advantage of this approach is that we can find all existing solutions in a very short time. Algebraic solutions are usually based on knowledge of the robot kinematic chain [22].

The gaze control and stabilization problem can be reformulated using additional constraints and relations to simplify the solution of the problem. Omrčen et al. [5] for example introduced a virtual mechanism that extends the head with a virtual link. This is essentially a prismatic joint leading from the robot's eyes that adds an additional degree of freedom to the system. The task of gaze control is thus reformulated to trying to make the end of the virtual link touch the point of interest by adjusting the link length. A similar approach was applied by Habra et al. [15] where by introducing a link between the eye and the target they transformed the task into a classical serial manipulator control. Milighetti et al. [16] computed the inverse kinematics, *i.e.*, the joint coordinates of the neck, required to view the selected object with known 3D coordinates using trigonometric functions. In our work, a similar computational procedure is used.

Gaze task redundancy. Active gaze control of the robot does not usually require all available degrees of freedom (DoF) of the neck to accomplish the gaze task (looking at a point requires 2 DoF). The remaining DoF becomes redundant [24] and can be used to solve another separate task simultaneously with the main one [5, 20, 21]. The most common additional task related to gaze control is avoiding occlusions. In robotic manipulators, occlusions are most often addressed when the target is overlaid by another object [25–27] or the target object overlaps an important part of itself [28]. Occlusions that can often occur in humanoid robots are self-occlusions, *i.e.*, the target is obscured by some part of

the robot itself. Sugiura et al. [29] solve the self-occlusion of the humanoid robot ASIMO by adding a link between the camera and the target, changing the problem of self-occlusion avoidance to the problem of avoiding self-collision avoidance instead.

Gaze stabilization as the same task as gaze control. Gaze control and gaze stabilization can be thought of as similar tasks that aim to keep the point of interest in the center of the field of view. Therefore, often the solution of these tasks allows to simultaneously track a moving object (gaze control) and at the same time compensate for the movements induced by the robot (gaze stabilization) [20]. Gaze stabilization can be solved without using visual perception by purely compensating for the robot's movements. Grotz et al. [30] used a stabilization solution that uses information on the induced velocities in the robot joints. Marchand et al. [26] also use additional means such as a 2D laser scanner for self-localization, so that all robot motion in space is compensated.

Chapter 3

Materials and Methods

In this chapter, we first describe our experimental setup consisting of the humanoid robot iCub and the externally mounted RGB-D camera in Sec. 3.1.1. Then, we describe in more detail the joints of the torso and neck that are most relevant for this work and introduce the coordinate systems at important points of the robot in Sec. 3.1.2. In Sec. 3.2, we explain gaze stabilization and provide a simple approach to stabilizing the gaze of this robot using knowledge of the joint coordinates. Finally, in Sec. 3.3 we describe the *optical flow* algorithm that we employ to evaluate the quality of the gaze stabilization.

3.1 Experimental setup

For our experiments, we use the humanoid robot iCub [31]. iCub is an advanced platform developed for research in robotics and artificial intelligence. Its design and equipment enable complex interaction with the environment.

3.1.1 Hardware configuration

iCub has a humanoid anatomy with proportions comparable to the body of the human child. iCub has complex limbs, including hands with fingers, allowing it to perform object manipulations. The robot's head contains a central control unit that coordinates movements and processes inputs from various sensors to adapt to the environment.

Two stereo RGB cameras are placed in the robot's eyes for visual perception. However, the resolution of these eye cameras and the accuracy of the depth obtained from stereo are not always sufficient for the activities performed by the robot. For this reason, our robot was retrofitted with an RGB-D camera externally attached to its head, see Fig. 3.1, which provides significantly higher resolution and more accurate depth information.

3.1.2 Coordinate systems and robot joints

The knowledge of the transformations between the different parts of the robot is fundamental for the control of the robot. The origin of the reference coordinate system, *base*, is placed under the joints of the robot's torso. The x-axis of this system points behind the robot, the y-axis points perpendicular to it towards the robot's right arm, and the z-axis is parallel to gravity but with the opposite orientation, see Fig. 3.2a. This figure also shows the other important coordinate systems for this work, which are located in the neck, the eyes, and

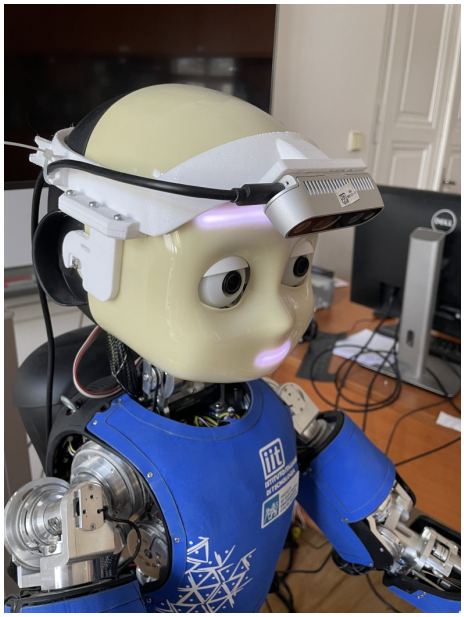


Figure 3.1: RGB-D camera mounted on the iCub's head.

the RGB-D camera attached to the robot's head. Other parts of the robot are not used in this work.

The neck coordinate system, unlike the *base* frame, is located above the joints of the neck, thus there are six joints between these frames—three neck and three torso joints (see Fig. 3.2a). Each of these joints allows for a specific type of rotation—*yaw*, *roll*, or *pitch*.

The fundamental difference between the structure of the human neck and the structure of the iCub neck lies in the relative position of the three joints. While the joints in the human neck are localized in one place, so that one joint provides all three types of rotation (*pitch*, *roll*, *yaw*), the robot has one of the joints that is offset by a certain distance from the other two joints (see Fig. 3.2b). The joint with *pitch* rotation is the lowest of the neck joints and is therefore not affected by the rotation of the other two joints. The rotation of the *pitch* joint will always be in the same direction, directly toward the center of the chest and to the center of the back on the opposite side regardless *yaw* and *roll* joints. On the other hand, the *yaw* joint, which is above the *pitch* joint, is affected by its rotation, and the direction of the head when performing the *yaw* rotation varies according to the actual *pitch* joint inclination. The rotation of the individual joints is therefore not around the axes of the neck frame, but around the z-axes of the coordinate systems of the joints themselves, which differ from each other and influence each other (see Fig. 3.3a). It can lead to complications when working with the joints and may cause minor inaccuracies.

The neck and torso rotations differ in their order and directions. Table 3.1 shows the order of the joints for the kinematic chains of the torso and neck. Figure 3.3b shows the signs for the corresponding movements of the neck and torso joints in the three rotations.

Chain	Joint index	Joint name
Neck	0	*pitch*
	1	*roll*
	2	*yaw*
Torso	0	*yaw*
	1	*roll*
	2	*pitch*

Table 3.1: Numbers of Robot Joints.

An important parameter of these revolute joints is their range of rotation, *i.e.*, their minimum and maximum safe angular rotation. Movement beyond these limits would cause damage to the robot. Table 3.2 shows the limits for both the real robot and its model in simulation.

		Hardware limit	
Chain	**Joint name**	**robot**	**simulation**
Neck	*pitch*	<-30°, 22°>	<-30°, 17°>
	roll	<-20°, 20°>	<-20°, 20°>
	yaw	<-45°, 45°>	<-45°, 45°>
Torso	*yaw*	<-50°, 50°>	<-50°, 50°>
	roll	<-30°, 30°>	<-30°, 30°>
	pitch	<-20°, 70°>	<-19.8°, 69.3°>

Table 3.2: Hardware Limits of Robot Joints.

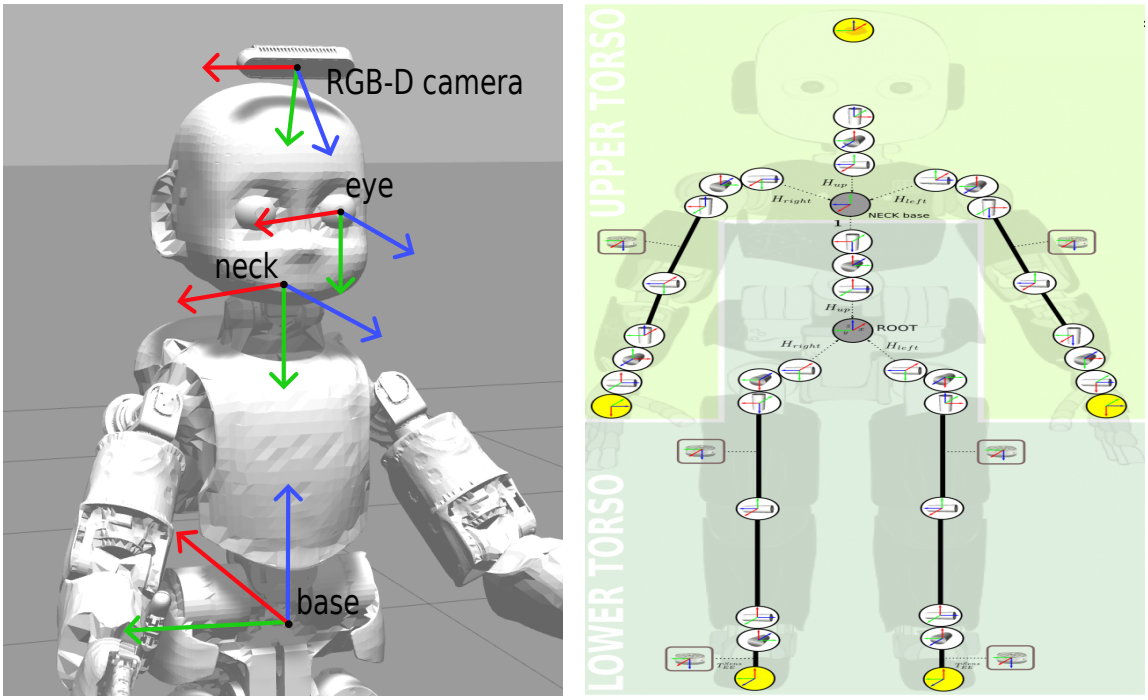
3.1.3 Robot programming

The robot programs for the iCub are usually written in C++ programming language. Yet Another Robot Platform (YARP) serves as a middleware that allows communication between different parts of the robot and external devices. Thus, it is possible to obtain real-time information about the robot's state, such as the current angular coordinates of the joints, which are important for real-time robot controllers.

3.2 Gaze stabilization

Gaze stabilization ensures that the movements of other parts of the robot do not result in any change in the field of view (FoV). This is particularly important with regard to the visual control of the robot's activity and camera image analysis. In the absence of gaze stabilization, the head is compelled to move in synchronization with the torso, as the head is connected to the torso. These movements have the potential to cause a loss of crucial points within the FoV for a given activity. Furthermore, the analysis of the FoV is complicated by the presence of large pixel movements, which also impede the robot's ability to orient itself within its environment.

A simple approach to addressing gaze stabilization is to purely compensate for the



(a) Coordinate systems in base-, neck-, eye- and RGB-D camera frame.

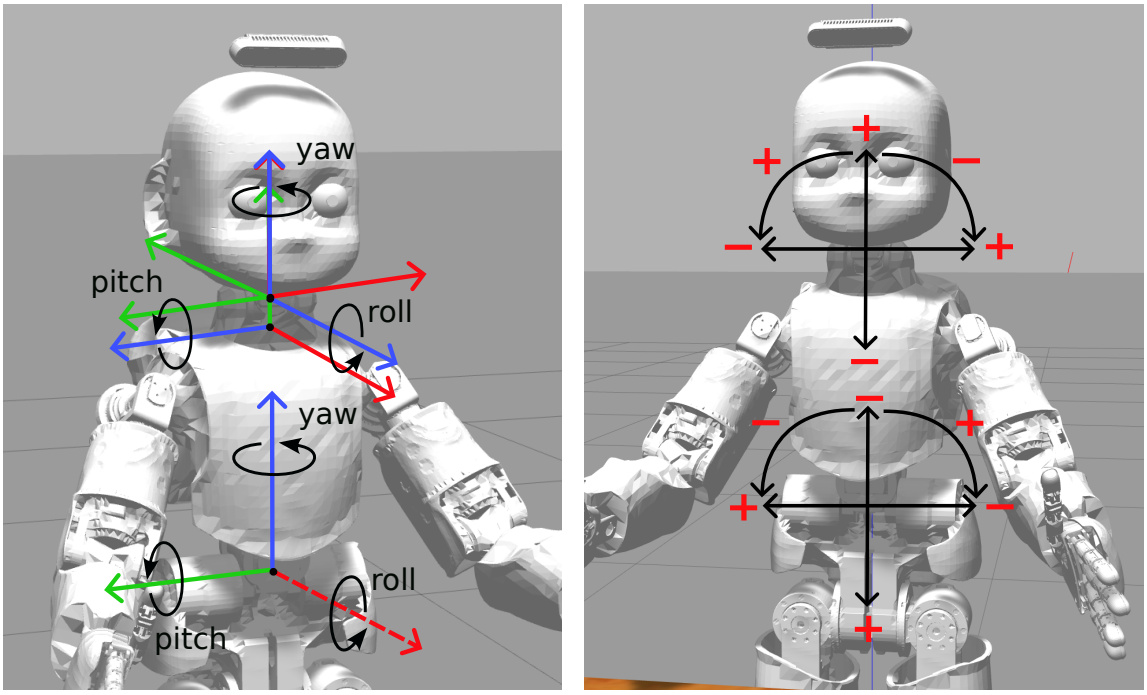
(b) Scheme of kinematic chain of the iCub robot from [32].

Figure 3.2: Signs of rotation of the joints of the torso and neck.

movement of the individual rotational joints of the torso with the corresponding neck joints. This approach uses knowledge of the robot structure and measurable joint coordinates of the torso.

Each torso joint is compensated individually by its corresponding neck joint (see Tab. 3.1). For example, movements of the torso joint allowing lateral bends are compensated by the neck joint providing lateral bends of the head. Since the rotations of the corresponding neck and torso joints have opposite signs of rotation (see Fig. 3.3), the desired values of the neck joints $(qdn_{\text{pitch}}, qdn_{\text{roll}}, qdn_{\text{yaw}})$ to provide stabilization are equal to the actual values of the torso joints $(qt_{\text{pitch}}, qt_{\text{roll}}, qt_{\text{yaw}})$.

Since the neck joints compensate for the motion of the torso joints only after they have been made, the stabilization is delayed by one time step. However, it can be assumed that the robot will not perform abrupt movements due to safety and the desire to appear calm and friendly. This assumption can be used to slightly improve the stabilization. For example, if the robot starts rotating its torso to the right, it will probably perform the motion for a longer period of time than a single loop. Thus, it is possible to predict the joint coordinates that the torso joints will reach in the next time step and improve gaze stabilization by setting more distant targets for the neck joints. For this purpose, we store the previous joint coordinates of the torso joints. We can calculate how many degrees each joint has moved between two consecutive updates (from the difference between the current and the previous joint position) and thus in which direction the robot is most likely to



(a) Different types of rotation marked in base frame and neck frame. The individual joints perform rotation about their own z-axis as indicated for the neck joints. The dashed axis indicates a negative axis orientation.

(b) Signs of rotation of individual joints of the torso and neck.

Figure 3.3: Signs of rotation of the joints of the torso and neck.

continue moving. We then use the α parameter to determine how far ahead we want to predict the movement.

$$qdn_{\text{pitch}} = qt_{\text{pitch}} - \alpha \cdot (qt_{\text{pitch}} - qt_{\text{prev, pitch}}) \tag{3.1}$$

$$qdn_{\text{roll}} = qt_{\text{roll}} - \alpha \cdot (qt_{\text{roll}} - qt_{\text{prev, roll}}) \tag{3.2}$$

$$qdn_{\text{yaw}} = qt_{\text{yaw}} - \alpha \cdot (qt_{\text{yaw}} - qt_{\text{prev, yaw}}) \tag{3.3}$$

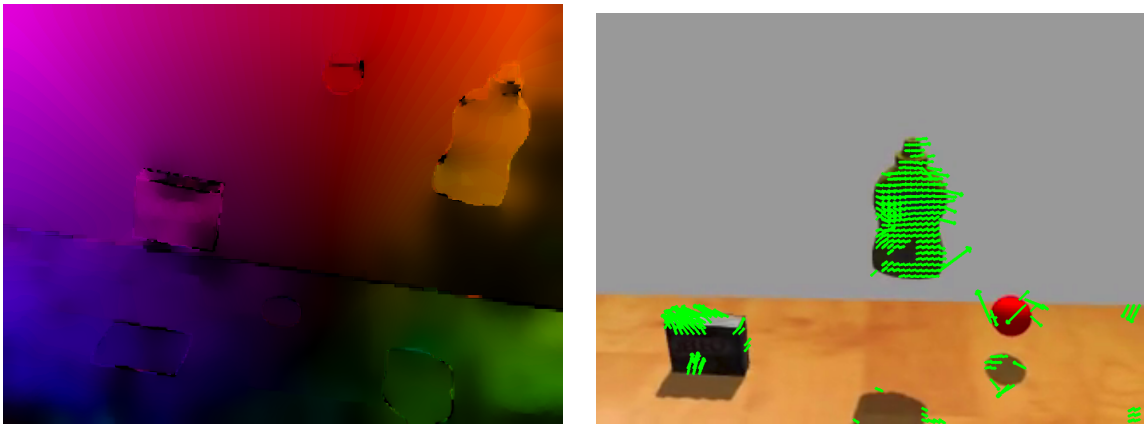
3.3 *Optical flow*

Optical flow is a technique used to describe the motion in an image. This two-dimensional image motion is the projection of the three-dimensional motion of objects, relative to a camera, onto its image plane [33]. *Optical flow* algorithm can be used to evaluate the quality of gaze stabilization [15, 34]. It is applied to videos or a series of image frames with a high frame rate and its output is a motion vector in Cartesian coordinates that represents the motion of pixels between two consecutive images. The technique uses two basic assumptions, namely that the pixel intensity of an object does not change between successive frames, and that neighboring pixels have similar direction and speed of motion.

There are two basic approaches to compute the *optical flow* depending on the volume of data they handle—sparse and dense. Sparse *optical flow* methods compute the motion vector only for defined parts of the image, for example, the corners of objects, *i.e.*, for a sparse data set. Thus, it does not detect background motion without distinct curves and displacement of round objects. It is therefore unsuitable for stabilization evaluation in simulation environment. The most well-known algorithm for computing sparse *optical flow* is the Sparse Lucas-Kanade [35] algorithm.

In contrast, the dense approach works with all pixels in the image. This slows down the computation, but also makes the results more accurate. For each point in the image, the change in position is calculated, and then the motion vector is determined. Algorithms for dense *optical flow* calculation are, for example, Farneback [36] and RLOF [37]. The difference between these methods lies mainly in the approach to *optical flow* calculation and their characteristics. The main idea of the Farnebeck algorithm is to approximate the motion in each pixel using second-degree polynomials and produce an estimate of the local derivatives from which the total motion can then be determined. RLOF (Robust Local Optical Flow) uses more robust estimation methods that are less prone to outliers, noise, and inhomogeneous background.

To illustrate the output of the algorithms, either arrows representing the motion vector of each pixel are drawn in the video, or each pixel is assigned a BGR color corresponding to the angles and magnitudes of the *optical flow*, see Fig. 3.4. In this work we use the OpenCV library and *optical flow* calculation demo from [38].



(a) Representation by BGR colors. **(b)** Representation by arrows.

Figure 3.4: *Optical flow* represent by BGR colors and arrows.

Chapter 4

Gaze Controller

This chapter introduces the proposed 3-DOF algebraic gaze controller. First, the preliminaries and assumptions are mentioned in Sec. 4.1. Then, the algebraic solution for gaze control is presented in Sec. 4.2, followed by its use for gaze stabilization in Sec. 4.3. Examples of redundancy exploitation tasks are described in Sec. 4.4 and finally, the gaze control strategy is presented in Sec. 4.5.

4.1 Preliminaries

For our method, we assume that the arrangement of the neck joints is as described in Sec. 3.1.2, specifically that the neck joint *pitch* is the first of the neck chain joints and is not located at the same point as the other two. However, our method is applicable to a configuration in which all neck joints are located at a single point.

The algebraic control solution is based on knowledge of the transformation matrices between the robot's frames and the gaze target. In Sec. 3.1.2 we have introduced coordinate systems in the **b**ase, **n**eck, **e**ye, and **r**gb-d camera. For transformation matrices the notation T is used with two indices to denote the frames to which the transformation is applied, all of them are shown in Fig. 4.1.

The transformations T_{bn} and T_{be} can be obtained from the current robot configuration. The RGB-D camera is attached to the robot's head, so its position and rotation relative to the neck are fixed. The transformation of the RGB-D camera T_{nr} is solved by calibrating it with respect to a fixed coordinate system, in our case the neck frame. This can be done using the iCub realsense-holder-calibration module[1].

The remaining transforms are obtained by the following calculations:

$$T_{br} = T_{bn} \cdot T_{nr} \tag{4.1}$$

$$T_{ne} = T_{nb} \cdot T_{be}. \tag{4.2}$$

Vectors used for calculations have a subscript indicating in which frame the vector is located. For example, o_n represents the coordinates of the object in the neck frame. We will also use the superscript H to denote vectors in homogeneous form, *i.e.*, a four-element vector with the value one as the last element. In this form, vectors can be multiplied by transformation matrices.

[1] `https://github.com/robotology/realsense-holder-calibration`

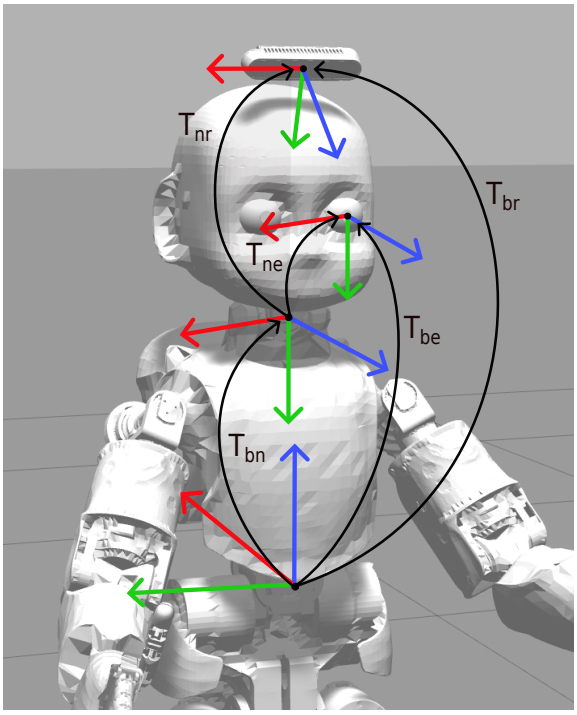


Figure 4.1: Transformations between robot **t**orso, **n**eck, **e**ye, and **r**ealsense frame.

4.2 Gaze control

As looking at a point in the scene requires only two degrees of freedom (DOF), it is possible to view almost any point in the space in front of the robot with only two joints (*pitch* and *yaw* neck joints in our case). The only limitation is their joint limits (shown in Tab. 3.2).

Figure 4.2 shows the robot in simulation along with the views through the RGB-D camera in the left window and the camera in the left eye of the robot in the right window. The visualization shows that in the robot's base position, the external camera is pointed slightly downward to obtain a better view of the manipulation area.

The following calculations do not depend on the type of camera used (one of the eyes or RGB-D), so we use general camera transformations such as T_{bc}. This matrix can then be equal to, *e.g.*, T_{be} or T_{br}.

If the robot is to maintain a view on a particular point, we need to know the 3D coordinates of that point relative to the robot. For the following calculation, we use its position in the neck coordinate system o_n. Usually, this information is not available, but can be obtained by additional calculations.

In simulation, the position of objects in the global coordinates of simulation o_g is usually known. The transformation from these global coordinates to the robot torso in our case requires a rotation by the unit rotation matrix and a shift only in the z-axis by the height of the torso, *i.e.*, 0.63 m. Therefore, it is sufficient to subtract this displacement to locate the target in the torso coordinates. Then the already mentioned T_{bn} transformation is used.

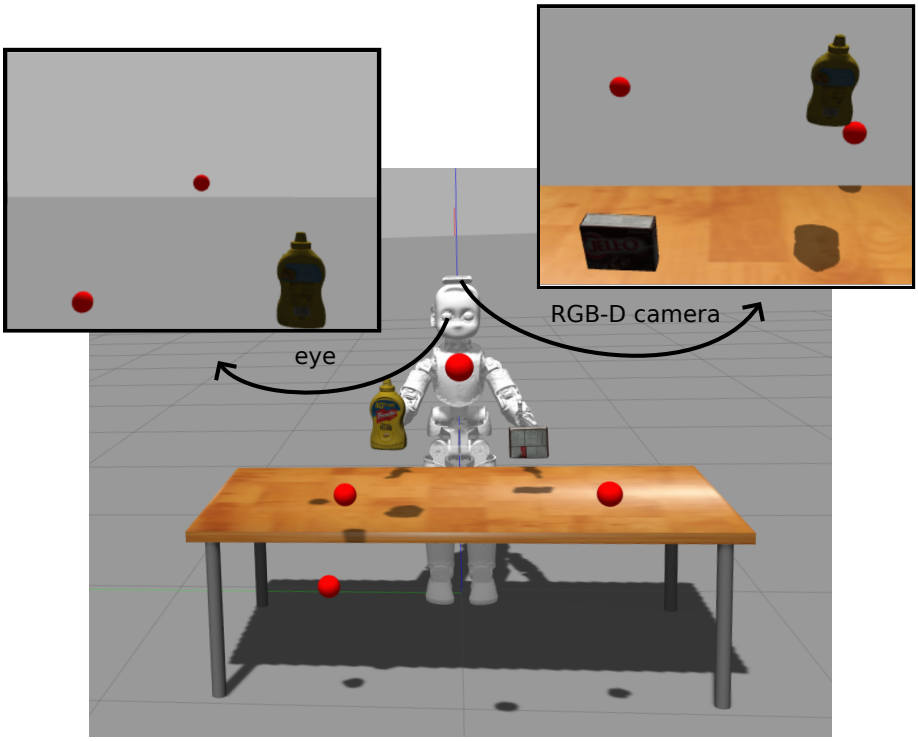


Figure 4.2: The robot in simulation and the view of the RGB-D camera mounted on its head (right) and the view of the camera in the robot's left eye (left). The figure shows the difference in orientation of these two cameras.

$$o_b^H = o_g^H - [0, 0, 0.63, 0]^T \tag{4.3}$$

$$o_n^H = T_{nb} \cdot o_b^H \tag{4.4}$$

$$\tag{4.5}$$

For gaze computation, the current rotation of the camera and the line-of-sight (LoS), *i.e.*, the line between the camera and the object must also be taken into account. We define a point on the LoS $v_c^H = [0, 0, d, 1]$, where $d = ||o_c||$, *i.e.*, is the distance between the camera and the target object. In the coordinates of the camera used, this is its z-axis. In the neck frame, we then compute the point on the LoS as follows:

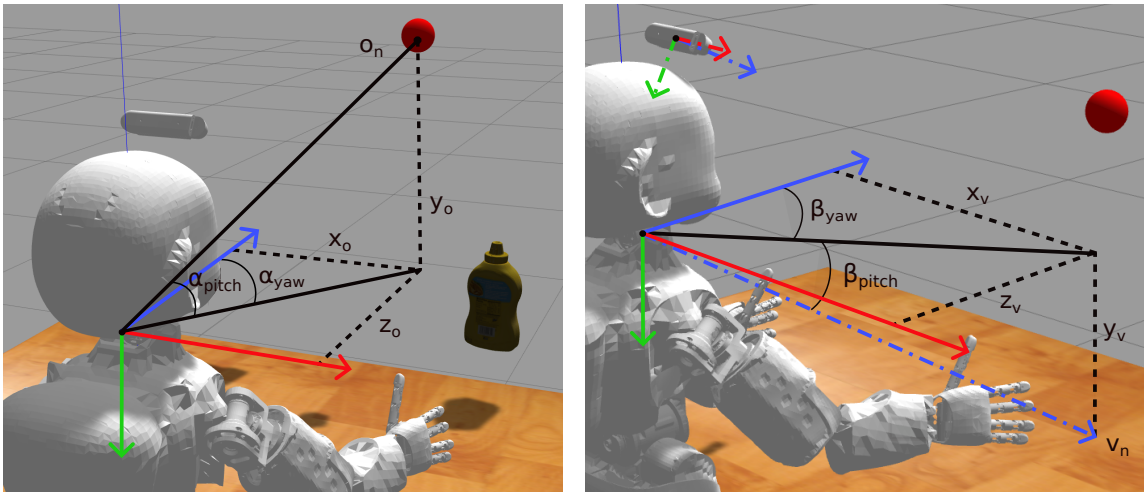
$$v_n^H = T_{nc} \cdot v_c^H \tag{4.6}$$

As mentioned in Sec. 3.1.2, the neck joints are not located at one point, but the *pitch* joint is located below the *roll* and *yaw* joints. This means that the rotation of the *pitch* joint is not around the x-axis of the neck coordinate system, but around one of its own axes. Thus, the angles related to the *pitch* joint need to be calculated in a coordinate system that is not affected by the tilt of the other two neck joints. Therefore, to calculate the *pitch*

joint values, we use the T_{pt} transformation from the simulated neck coordinate system to the torso coordinate system created for the simulated zero value of the *roll* and *yaw* joints.

$$o_p^H = T_{pt} \cdot o_b^H \tag{4.7}$$
$$v_p^H = T_{pt} \cdot T_{bc} \cdot v_c^H \tag{4.8}$$

where o_p is the coordinates of the target and v_p is the point on the LoS vector for the simulated neck frame. The components (x_o, y_o, z_o) of the vector o_p and components (x_v, y_v, z_v) of v_p are used to calculate the angles α_{pitch} and β_{pitch} between neck and the object or the point on the LoS vector. Figure 4.3a shows the situation for the calculations with the object and Figure 4.3b for calculations with the point on the LoS vector. The angles are then converted from radians to degrees.



(a) Angles α_{pitch} and α_{yaw} between the robot neck and the object of interest. The components (x_o, y_o, z_o) of the vector o_n are its coordinates in the neck frame.

(b) Angles β_{pitch} and β_{yaw} between the robot neck and the point on the line-of-sight vector (dot-and-dash blue line) in neck frame. The components (x_v, y_v, z_v) of the vector v_n are its coordinates in the neck frame.

Figure 4.3: Important angles for calculating the target joint coordinates of the *pitch* and *yaw* joints.

$$\alpha_{pitch} = \tan\left(\frac{-y_o}{\sqrt{(x_o)^2 + (z_o)^2}}\right) \cdot 180/\pi \tag{4.9}$$
$$\beta_{pitch} = \tan\left(\frac{-y_v}{\sqrt{(x_v)^2 + (z_v)^2}}\right) \cdot 180/\pi \tag{4.10}$$
$$\tag{4.11}$$

The resulting joint value for the *pitch* joint is obtained from the angles calculated above and the current rotation of this joint q_{pitch} using the following equation:

$$qn_{pitch} = \alpha_{pitch} - \beta_{pitch} + q_{pitch} \tag{4.12}$$

To calculate the angles related to the *yaw* joint, we create a transformation T^s_{bn} from the torso coordinate system to the simulated neck coordinate system, which already includes the current rotation of the *roll* and *yaw* joints, but instead of the current *pitch* joint setting, it simulates the calculated value of qn_{pitch}.

$$o^H_n = T^s_{nb} \cdot o^H_b \tag{4.13}$$
$$v^H_n = T^s_{nc} \cdot v^H_c \tag{4.14}$$

The angles α_{yaw} and β_{yaw} (the calculation shown in Fig. 4.3) and the resulting value of the yaw joint qn_{yaw} are calculated as follows:

$$\alpha_{yaw} = \tan\left(\frac{x_o}{z_o}\right) \cdot 180/\pi \tag{4.15}$$
$$\beta_{yaw} = \tan\left(\frac{x_v}{z_v}\right) \cdot 180/\pi \tag{4.16}$$

$$qn_{yaw} = -(\alpha_{yaw} - \beta_{yaw}) + q_{yaw} \tag{4.17}$$

where x_o, y_o and z_o are components of the vector o_n and x_v, y_v and z_v are components of the view vector v_n. The value of q_{yaw} is the current joint coordinate of the *yaw* joint.

Since the new target may be several tens of centimeters away from the original target, it is not possible to perform the entire movement at once. This would cause sudden and too fast movement of the neck, which could cause a technical malfunction of the robot. Therefore, we use the *minjerk*[2] function, which generates an output trajectory for the computed joint coordinates that has the properties of a quasi-minimal jerk, thus resembling human-like motions as described in [39], and also sampling the motion so that it is not too large in one step. We assume that in such small motions, the error due to rotation of the neck joints is not so noticeable at once.

It may happen that the robot receives a point of interest that it cannot look at. This means that calculating the coordinates of the neck joints needed to reach that point will produce values that are outside of their limits. In this case, it is necessary to ensure that the joints do not exceed their limits while the robot looks as close to the target as possible. Therefore, the resulting joint coordinate q_i for i-th joint is clipped as:

$$q_i = \max(q^l_i + 2.1^\circ, \min(q_i, q^u_i - 2.1^\circ)) \tag{4.18}$$

[2] `https://robotology.github.io/robotology-documentation/doc/html/group__minJerkCtrl.html`

where q_i^l and q_i^u are lower and upper safety joint limit, respectively.

The procedure shows that it can be used for any camera located on the robot head, as long as the transformation matrix from the robot neck to it is known. At the same time, matrix multiplication and computation of goniometric functions are not computationally intensive operations. Therefore, it is a fast and robust approach.

4.2.1 2D target in camera image

In practice, information about the surrounding environment is most often obtained from camera images. We can obtain the position of the object in the image plane in pixels using standard image processing methods or manually.

Without explicit knowledge of the distance of the target from the camera, we cannot accurately calculate the 3D position. However, for tracking purposes, the LoS vector is more important than the accurate 3D position. Therefore, we can assume that the distance from the camera is 1 m to estimate the 3D position approximately. For the computation, we need to know a camera calibration matrix K which contains the camera focal lengths f_x and f_y and the optical centers c_x and c_y.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.19}$$

To convert between the pixels of a camera image point and the coordinates of that point in the camera coordinate system, the following equation applies:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = K^{-1} \cdot \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{4.20}$$

where $o_c = [x, y, z]$ is the point in the camera frame, u and v are the pixel values, and w is the depth of the point in space fixed to the value 1 m. For control purposes, the resulting point o_c is transformed to the neck frame:

$$o_n^H = T_{nc} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.21}$$

4.3 Gaze stabilization

The gaze control principles described above can also be applied to gaze stabilization. Stabilization can basically be thought of as gaze control since due to the disturbing movements of the torso and other parts of the robot, the position of the target relative to its camera is constantly changing, and so the gaze needs to be controlled even though the position of the target in the environment remains the same.

Due to the nature of the gaze stabilization, we assume that the calculated joint angles will differ only slightly from the actual ones. Therefore, the calculated angles are directly applied to the joints.

4.4 Redundant joint

As mentioned and shown above, it is not necessary to use the neck joint *roll* to focus the gaze on the target. To exploit this redundancy, we have implemented two secondary tasks that complement the primary task (*i.e.*, target gazing). The first of these tasks focuses on keeping a second target visible in the field of view (FoV) while the neck moves due to the primary target's shift. The second task aims to keep the gaze parallel to the environmental horizon both during gaze shifts between targets and during stabilization.

4.4.1 Second target

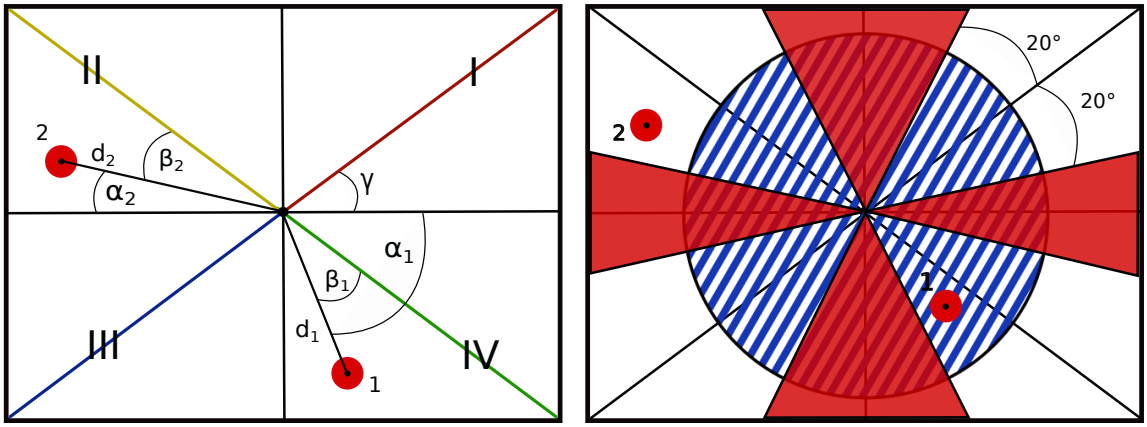
When the target is changed, the robot immediately begins to direct its gaze to the new coordinates. Depending on the distance between the original and the new target, the visible scene, *i.e.*, the part of the environment that is captured in the field of view, may also change significantly. Thus, the robot may lose visibility of most of the objects that were previously in the field of view. This loss can be avoided in some cases using lateral neck tilts.

Since the field of view is rectangular, the best way to maintain visibility of the second target is to point that target toward the diagonals of the field of view. Thus, we try to minimize the angle β that the object forms with the diagonal with respect to the quadrant of the FoV in which the object is situated. In Fig. 4.4 on the left, we can see that ball 1 is located in the fourth quadrant, so we try to minimize the angle β_1 between that object and the green part of the diagonal.

$$quadrant = \begin{cases} I & \text{if} \quad u > w/2 \ \wedge \ v < h/2 \\ II & \text{if} \quad u < w/2 \ \wedge \ v < h/2 \\ III & \text{if} \quad u < w/2 \ \wedge \ v > h/2 \\ IV & \text{if} \quad u > w/2 \ \wedge \ v > h/2 \end{cases} \tag{4.22}$$

The range of motion of the *roll* joint is only 20° to each side, so it does not give much space for maneuver. Figure 4.4 on the right shows the situation specifically for the zero value of the *roll* joint. If the head has not yet rotated in this rotation direction, then the

red space indicates where the robot is unable to direct the appropriate diagonal by tilting the head.



(a) FoV with important angles for calculating the joint coordinate of the joint *roll*.

(b) FoV with constraints and possible location of second target plotted.

Figure 4.4: The field of view for tracking the second target. The blue hatched circle indicates the area where if the second target is located inside we do not use the rotation of the *roll* joint. The red sections show where the FoV diagonals cannot be pointed by rotating the neck *roll* joint due to the limitations of this joint in the robot's default position.

All calculations are performed in the camera image plane, so the position of the second target $sec\_target = [u, v]$ must be in pixels. We show the calculations for the general camera resolution $w \times h$. Next, we introduce a vector a that will be directed from the center of the field of view to the second target.

$$a = \Big[u - w/2, v - h/2\Big] \tag{4.23}$$

If the following two conditions are fulfilled, the angle for the *roll* joint is calculated as described below:

1. the visibility of the second target can be ensured, *i.e.*, it is located in a circle circumscribed around the field of view:

$$\|a\| < \sqrt{(w/2)^2 + (h/2)^2} \tag{4.24}$$

2. the second target is not closer to the center of the field of view than a radius r:

$$\|a\| > r \tag{4.25}$$

If the second condition is not satisfied, then the second target is near the center of the field of view. In that case, it is not advisable to change the *roll* rotation in any way because the quadrant in which it lies, and thus the sign of the rotation may often change, which

would lead to an oscillating movement of the head from side to side. In addition, the target at that moment is in a clearly visible part of the field of view, so no *roll* movement is needed. In Fig. 4.4 on the right, this area is marked with a blue hatch.

We use the pixel values of the second target to determine in which quadrant it is located and compute the angle α, the angle between the vector a and the vector s, which lies on the horizontal axis of the field of view with the origin at its center and the orientation dependent on the quadrant.

$$s = \begin{cases} [w/2, 0] & \text{if} \quad quadrant = I \ \vee \ quadrant = IV \\ [-w/2, 0] & \text{if} \quad quadrant = II \ \vee \ quadrant = III \end{cases} \tag{4.26}$$

$$\alpha = \arccos \frac{a \cdot s}{\|a\| \cdot \|s\|} \cdot 180/\pi \tag{4.27}$$

The horizontal axis of the field of view and its diagonal meet at the angle γ, its value depends on the resolution of the particular camera and is computed as follows:

$$\gamma = \arccos \frac{[w/2, h/2] \cdot s}{\|[w/2, h/2]\| \cdot \|s\|} \cdot 180/\pi \tag{4.28}$$

The angle β is equal to the absolute value of the difference of the angles α and γ:

$$\beta = |\alpha - \gamma| \tag{4.29}$$

Since the orientation of the head is constantly changing as the *pitch* and *yaw* joints control gaze toward the primary target, the position of the second target in FoV is also constantly changing. Therefore, the *roll* joint is moved only by an angle k of at most one tenth of a degree at each step.

$$k = \begin{cases} 0.1 & \text{if} \quad \beta \geq 0.1 \\ \beta & \text{if} \quad \beta < 0.1 \end{cases} \tag{4.30}$$

To find out whether the rotation by the obtained angle β should be made in the positive or negative direction, *i.e.*whether this angle should be added or subtracted from the current *roll* joint coordinate q_{roll}, we use the magnitude of the angle α and the quadrant number as follows:

$$roll = \begin{cases} q_{roll} + k & \text{if} \quad (quadrant = I \ \vee \ quadrant = III) \ \wedge \ \alpha < \gamma \\ q_{roll} + k & \text{if} \quad (quadrant = II \ \vee \ quadrant = IV) \ \wedge \ \alpha \geq \gamma \\ q_{roll} - k & \text{if} \quad (quadrant = I \ \vee \ quadrant = III) \ \wedge \ \alpha > \gamma \\ q_{roll} - k & \text{if} \quad (quadrant = II \ \vee \ quadrant = IV) \ \wedge \ \alpha \leq \gamma \end{cases} \tag{4.31}$$

4.4.2 Horizontal view

If the robot uses only neck joints *pitch* and *yaw* for stabilization, the field of view rotates around the center while the robot's torso is laterally inclined. This causes a large pixel movement and can lead to blurring, thus a more difficult image analysis. Therefore, for some tasks, it may be advantageous to compensate for this rotation with the redundant neck joint to keep the view parallel to the environmental horizon.

The following procedure describes how to find the angle ϕ, the angle between the horizontal axis of the view and the horizon. Figure 4.5 shows this procedure schematically.

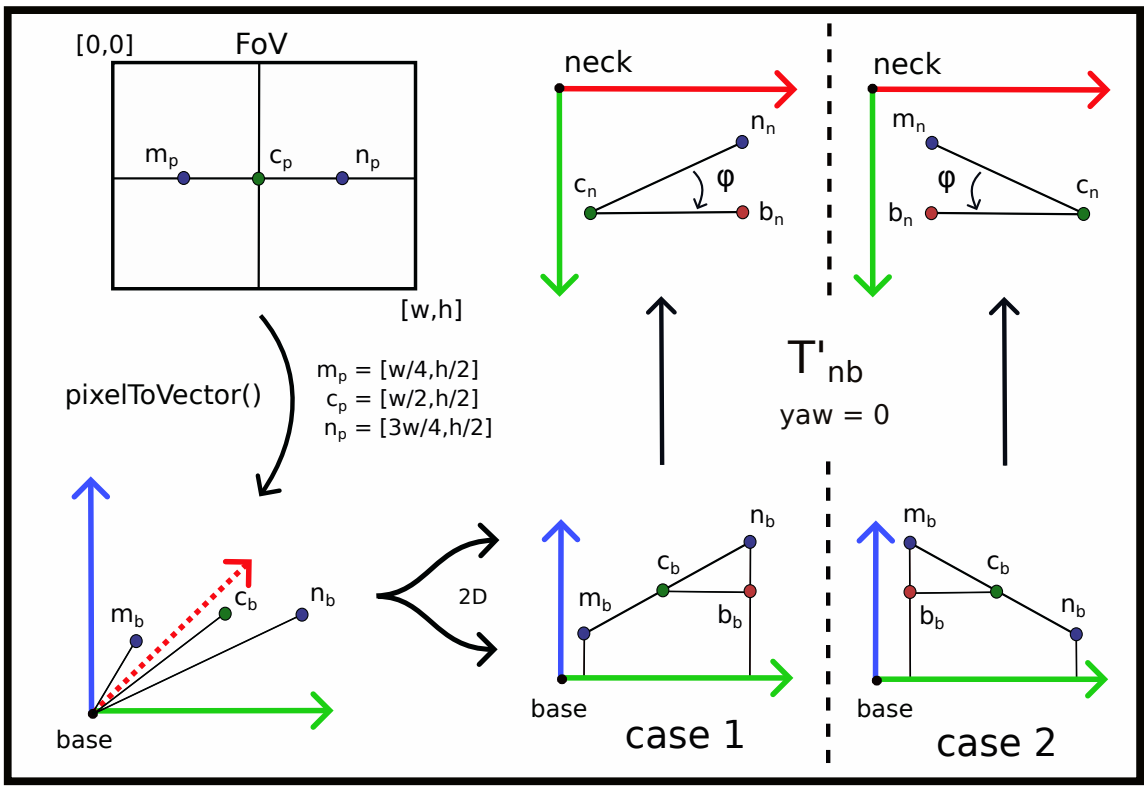


Figure 4.5: The diagram shows the process of calculating the angle ϕ by which the robot's FoV is rotated relative to the environment horizon.

We select three important points in the field of view that lie on its horizontal axis. The point c_p is the center of the image, and the points m_p and n_p are on the right and left halves of the axis, to be able to tell to which side the view is tilted. In our case, we have determined their positions at a quarter and three-quarters of the width w.

$$m_p = [w/4, h/2] \tag{4.32}$$
$$c_p = [w/2, h/2] \tag{4.33}$$
$$n_p = [3w/4, h/2] \tag{4.34}$$

These three points are converted into the torso frame using the camera matrix (the detailed procedure is described in Sec. 4.2.1). The dashed red arrow of the torso frame in the diagram indicates the negative x-axis.

To find out how the view is rotated, we only need to use a 2D space projection, since the distance of the points from the robot is not relevant. The projection is done in the torso and neck frames. Figure 4.5 shows that the distribution of points in a 2D space can take two forms. If the robot's field of view is tilted to the left, the points corresponding to the right half of the horizontal axis of the field of view are higher in the torso frame than those from the left half, so their z coordinate has a larger value. In the diagram, this case is denoted by the number one. Otherwise, the view is tilted to the right side, denoted by number two. In both cases, we create a new point b_b, which has the same values of the x and y coordinates as the n_b and m_b points above and is shifted to the level of the point c_b on the z-axis. The line between points c_b and b_b marks the horizon and therefore where the horizontal axis of the field of view should ideally be projected if the view were horizontal with the environment.

$$b_b = \begin{cases} [n_b[0], n_b[1], c_b[2]] & \text{if} \quad m_b[2] \leq n_b[2] \\ [m_b[0], m_b[1], c_b[2]] & \text{if} \quad m_b[2] > n_b[2] \end{cases} \tag{4.35}$$

The goal is therefore to minimize the angle between the points b_b, c_b, and the higher point from the points m_b and n_b. To calculate the appropriate angle more accurately, we convert these points to the neck frame, specifically the *roll* joint frame. This joint is placed below the *yaw* joint in the robot structure as the second neck joint in the sequence. To transform the points, we use the matrix T^s_{nb} with the simulated zero value of the *yaw* joint.

$$c_n = T^s_{nb} \cdot c_b \tag{4.36}$$
$$b_n = T^s_{nb} \cdot b_b \tag{4.37}$$
$$m_n = T^s_{nb} \cdot m_b \tag{4.38}$$
$$c_n = T^s_{nb} \cdot c_b \tag{4.39}$$

$$\phi = \begin{cases} \arccos \frac{(n_n - c_n)\cdot(b_n - c_n)}{\|n_n - c_n\| \cdot \|b_n - c_n\|} \cdot 180/\pi & \text{if} \quad m_b[2] \leq n_b[2] \\ \arccos \frac{(m_n - c_n)\cdot(b_n - c_n)}{\|m_n - c_n\| \cdot \|b_n - c_n\|} \cdot 180/\pi & \text{if} \quad m_b[2] > n_b[2] \end{cases} \tag{4.40}$$

As in the calculation for the second target, we do not want the robot to rotate the *roll* joint by a large angle in one step. Therefore, the *roll* joint moves straight to the desired joint coordinate *roll* (see Eq. (4.41)) only in the stabilization loop, where we assume that the angle ϕ will not reach large values. In gaze control, the actual joint displacement in the current step is computed by the *minjerk* function, which tries to approximate *roll* by a small step.

$$roll = \begin{cases} q_{roll} + \phi & \text{if} \quad m_b[2] \leq n_b[2] \\ q_{roll} - \phi & \text{if} \quad m_b[2] > n_b[2] \end{cases} \tag{4.41}$$

4.5 Gaze control strategy

The gaze controller uses the proposed methods for gaze control and stabilization described above and connects them to a fully functional unit. Thus, it is possible to control the robot's gaze on the selected target, stabilize its gaze, and use redundancy in an adjustable way.

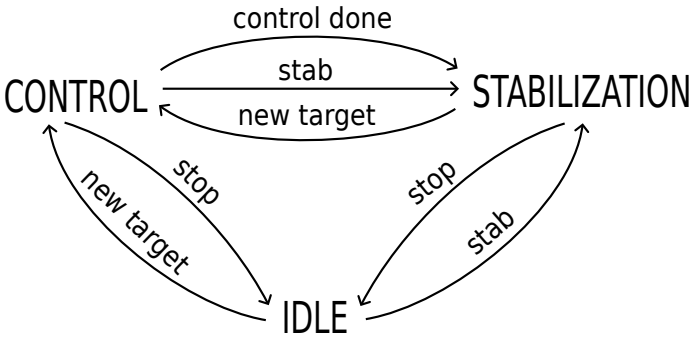


Figure 4.6: Gaze controller as state machine with tree states—IDLE, CONTROL and STABILIZATION. Switch between states by using the STAB and STOP commands or by specifying a gaze control target.

The Gaze controller works on the principle of a state machine with a total of three states—CONTROL, STABILIZATION and IDLE. Figure 4.6 shows schematically how these states are connected and what events trigger a switch between them. All activation commands are entered externally via a YARP port. The initialization state of the gaze controller is the IDLE state, in which the robot's neck is not controlled in any way; a return to this state can be invoked later with the STOP command. In the following subsections, we describe the two other states—CONTROL and STABILIZATION—in more detail. The pseudocode of the controller is presented in Algorithm 1 and the detailed scheme is shown in Fig. 4.7.

4.5.1 CONTROL state

Entering a new target puts the gaze controller into the CONTROL state and starts a loop of the gaze controller, which starts to smoothly change the orientation of the head, and

Algorithm 1 Gaze controller

```
while True do
    state ← IDLE
    if command entered then
        if command ∈ STOP then
            state ← IDLE
        else if command ∈ STAB then
            target ← LineOfSight
            state ← STABILIZATION
        else if command ∈ NewTarget then
            target ← NewTarget
            state ← CONTROL
            if command ∈ NewSecondTarget then
                secondTarget ← NewSecondTarget
            end if
        end if
    end if
    switch state do
        case IDLE
            wait()
        case CONTROL
            qdNeck_goal[0] ← computePitch()
            qdNeck_goal[2] ← computeYaw()
            if horizontal view then
                qdNeck_goal[1] ← computeHorizontalWiew()
            else if not second target then
                qdNeck_goal[1] ← 0
            end if
            checkLimits()
            qdNeck ← minjerk(qdNeck_goal)
            if secondTarget then
                qdNeck[1] ← computeSecondTarget()
            end if
            move(qdNeck)
            if abs(qdNeck_goal - qdNeck) < thresholds then
                state ← STABILIZATION
            end if
        case STABILIZATION
            if unsuccessful control then /* Only once */
                target ← LineOfSightVector
            end if
            if invisible secondTarget then
                qdNeck[1] step to zero
            end if
            qdNeck_goal[0] ← computePitch()
            qdNeck_goal[2] ← computeYaw()
```

```
            if horizontal view then
                qdNeck_goal[1] ← computeHorizontalWiew()
            end if
            checkLimits()
            qdNeck ← qdNeck_goal
            move(qdNeck)
    end while
```

thus controls the gaze. The new target can be sent from an external script as a pixel position in the image plane, *i.e.*, two values input, or as a 3D point in the torso frame or world frame for simulation, *i.e.*, three values input. To keep the target format uniform, we always convert it to the torso frame, from the world coordinates using the Eq. (4.3) and from the pixels according to the procedure described in Sec. 4.2.1. A second target can also be specified, in case we want to take advantage of the robot's redundancy and keep the second target in the field of view while shifting the view to the first target. The second target should have the same format as the first one.

The control loop calculates the target joint coordinates $qdNeck_{goal}$ for the *pitch* and *yaw* joints as detailed in Sec. 4.2. If we want to use redundancy to provide horizontal view, the *roll* joint value is calculated, see Sec. 4.4.2. However, if we do not want to use redundancy at all, the target *roll* joint value is set to zero. The computed joint values $qdNeck_{goal}$ are bounded into the joint ranges. We then pass them to the *minjerk* function, which returns the new $qdNeck$ values. Once we know exactly what motion the robot will perform in this step, we calculate the value of the joint *roll* $qdNeck$ if a secondary goal was also specified. The exact procedure is described in Sec. 4.4.1.

Finally, the robot moves the neck joints to the computed coordinates $qdNeck$. We check if the movement to the new target has been successfully completed. We consider the movement completed successfully if the difference between the values of $qdNeck_{goal}$ and $qdNeck$ is less than the set threshold of 0.1°. If a second target has been entered, the threshold for the pitch and yaw joints is 0.5° and the roll joint is not checked. Once the movement is completed, the robot is ready to stabilize its gaze on the newly focused target, and the stabilization loop is activated.

Currently, the controller and the inverse kinematics (IK) computation are closely connected. If the original gaze target is not reachable due to neck joint limits, the controller moves the gaze as close to it as possible. If the torso during the gaze control moves in a way that makes the target reachable, the gaze controller moves the gaze to the target pose. As the reachability of the target can change during the motion, it is currently not possible to retrieve the output of the IK solution itself without performing the movement.

4.5.2 STABILIZATION state

When the gaze control is complete, the gaze controller is automatically switched to the STABILIZATION state and stabilizes the gaze on the specified target. If the robot was unable to get the gaze on the target during the control loop due to joint limits or if

stabilization was externally induced by the STAB command, a new virtual gaze target is created. The virtual target is defined as the point on the LoS that is 1 m away from the camera converted to the torso frame. The view thus remains stable and within the range of the neck joints until a new target is specified. This recalculation is done at most once, when the stabilization loop starts.

$$o_c^H = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}^T \tag{4.42}$$

$$target = T_{bc} \cdot o_c^H \tag{4.43}$$

As we described in Sec. 4.3, gaze stabilization uses principles similar to gaze control; therefore, the following steps of the stabilization loop are similar to the control loop. The target value of the *pitch* and *yaw* joints is calculated, and in the case of using the horizontal view, the value of the *roll* joint is also calculated. The only difference occurs if a second target was also set for the control, but it could not be kept in the field of view; in this case we do not want to leave the robot head tilted unnecessarily and we set the *roll* joint value by 0.1° closer to zero, so it slowly returns to the default state.

After checking the limits of the robot, the movement of the joints to the $gdNeck$ values is executed again. Here, the *minjerk* function is no longer used, as the stabilization loop is fast enough to compensate for the small target motion with small neck movements that are not dangerous to execute directly without any sampling.

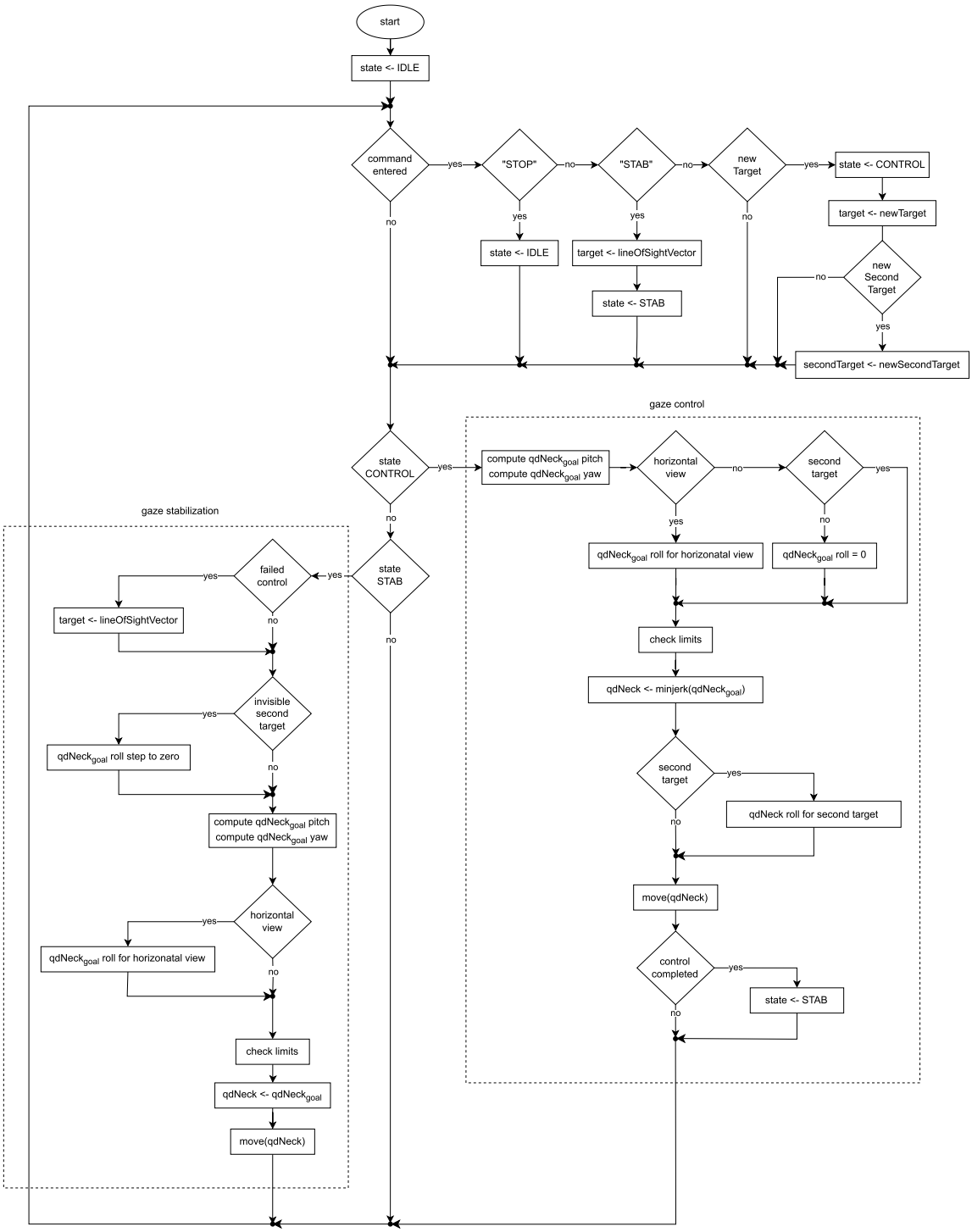


Figure 4.7: Gaze controller diagram.

Chapter 5

Experiments and Results

In this section, we describe the experiments performed to demonstrate the functionality of the proposed gaze controller described in Chapter 4. First, the results of the gaze control experiment are shown in Sec. 5.2. The evaluation of the gaze stabilization is described in Sec. 5.3. Next, the redundant joint is exploited in two experiments presented in Sec. 5.4. Finally, the gaze controller is used during a children's game against a human opponent. This experiment is described in Sec. 5.5.

5.1 Setup for experiments

Experiments were performed in both simulation and on the real iCub robot. An Intel RealSense D435 RGB-D camera mounted on the head of iCub was used to acquire visual data. Table 5.1 shows the resolution, optical center values (c_x, c_x) and focal lengths (f_x, f_y) for the RGB-D camera on the real robot and for its model in simulation.

Parameters	Real robot	Simulation
resolution	1280 × 720	320 × 240
c_x	639.18	160
c_y	342.85	120
f_x	918.48	343.12
f_y	916.39	343.12

Table 5.1: Parameters of the RGB-D camera on the real robot and in simulation.

As described in Sec. 3.1.2, the transformation matrix from the RGB-D camera to the neck T_{nr} is obtained by calibration using a calibration checkerboard. For the real robot, it has the following values:

$$T_{nr} = \begin{bmatrix} 0.9987059879 & -0.04040485152 & 0.03088361407 & -0.03887093391 \\ 0.004155840177 & 0.6700826921 & 0.7422748243 & -0.02957368113 \\ -0.05068607932 & -0.7411859644 & 0.6693835131 & 0.03798804518 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

The pose of the RGB-D camera model on the iCub in simulation does not match its position on the real robot due to different kinematic chain representation (iCub kinematics

vs. URDF model). Since camera calibration cannot be performed in simulation, this transformation matrix T_{nr} is approximated as:

$$T_{nr}^{s} = \begin{bmatrix} 0.9926778672 & -0.08964789262 & 0.08095620594 & -0.0372932849 \\ 0.01472872958 & 0.7550455074 & 0.6555069384 & -0.09669132672 \\ -0.1198904352 & -0.6495148475 & 0.7508373635 & 0.09589647841 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

5.2 Gaze control

In this experiment, we focused on demonstrating the functionality of the gaze control described in Sec. 4.2. The experiment was carried out in both simulation (see Sec. 5.2.1), and on the real robot (see Sec. 5.2.2). In both cases, targets were defined by pixel coordinates. These targets were sent to the gaze controller and the task was to move the head to have the target in the center of the FoV. This starting position of the target in the image is not relative to the FoV in the robot's default configuration, but is determined from the camera image taken after reaching the previous target.

5.2.1 Gaze control in simulation

The robot was given 12 gaze targets in the simulation experiment. If we had specified the location of the target in global or torso coordinates, the robot would not move to have the target in the center of FoV due to the inaccurate transformation matrix T_{nr}^{s} (see Sec. 5.1). Therefore, we always specified the position of the targets in pixels.

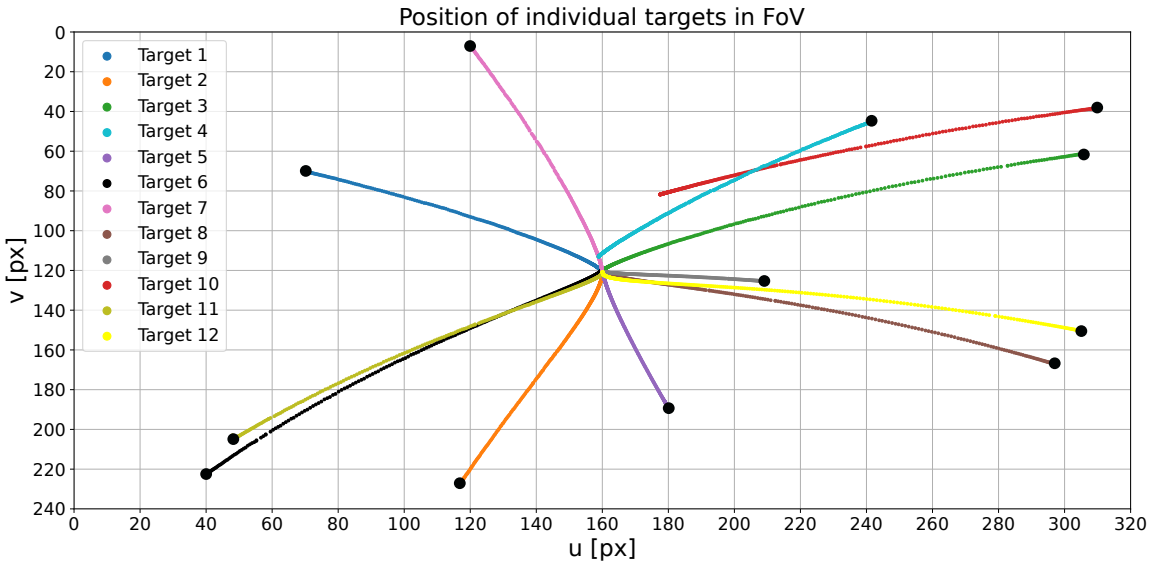


Figure 5.1: Gaze control experiment in simulation. Trajectories of individual targets in FoV as the gaze is moving towards the target. The black point on each trajectory shows the starting position of a target.

The trajectories of all specified gaze targets in the FoV during gaze control are marked in Fig. 5.1. The black point on each trajectory shows the starting position of a target. It can be seen that all targets, except Target 4 and Target 10, ended up in the center of the field of view $[160, 120]$.

In Fig. 5.2, we can see the profile of the joint coordinates of the neck joints *pitch* (blue curve) and *yaw* (green curve) during the experiment while moving the gaze to the new target. The data between moving gaze to targets are not shown, as they are not relevant for this experiment. It can be seen that the *pitch* joint reached its safety limit (dashed blue line) during the gaze control on Target 4 and therefore the gaze controller was not able to align the gaze center exactly on Target 4. In the case of Target 10, both joints reached their safety limits (dashed blue and green line).

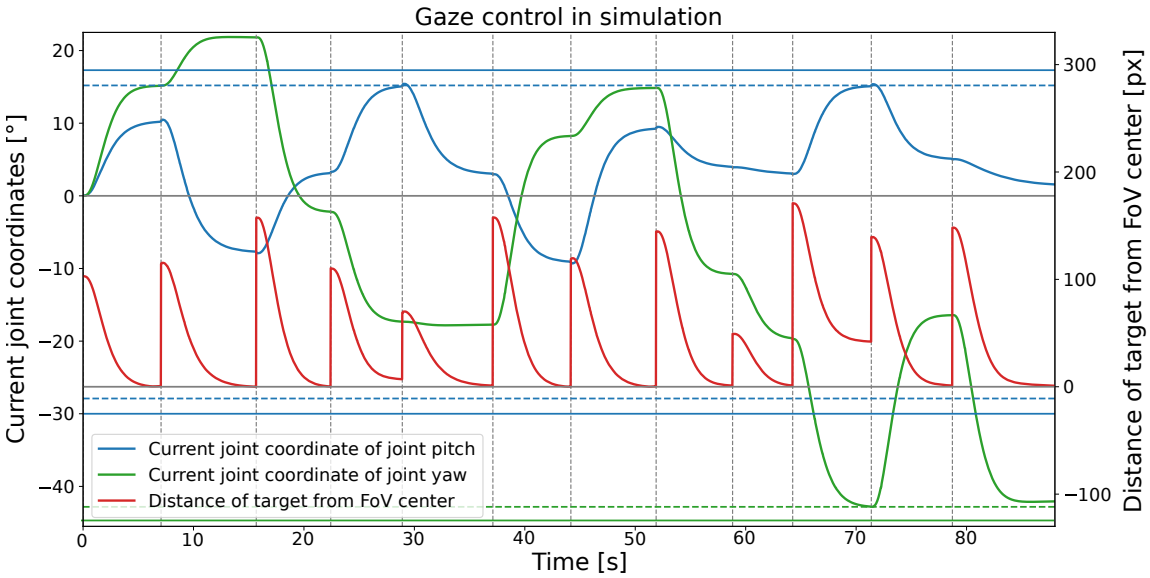


Figure 5.2: Gaze control experiment in simulation. Joint coordinates of neck joints *pitch* (blue curve) and *yaw* (green curve) and distance of gaze targets from FoV center (red curve). The vertical dashed lines indicate when the new target was entered and the horizontal lines with the corresponding colors mark the hardware limits (solid line) and the safety limits (dashed line) of *pitch* and *yaw* joints.

Figure 5.3 shows the view of the RGB-D camera when Target 1 was entered (on the left) and when the control was completed (on the right). In this case, the target is in the center of the FoV. On the other hand, Figure 5.4 shows the same situation for Target 10, which was not reached by gaze due to the limits of the neck joints.

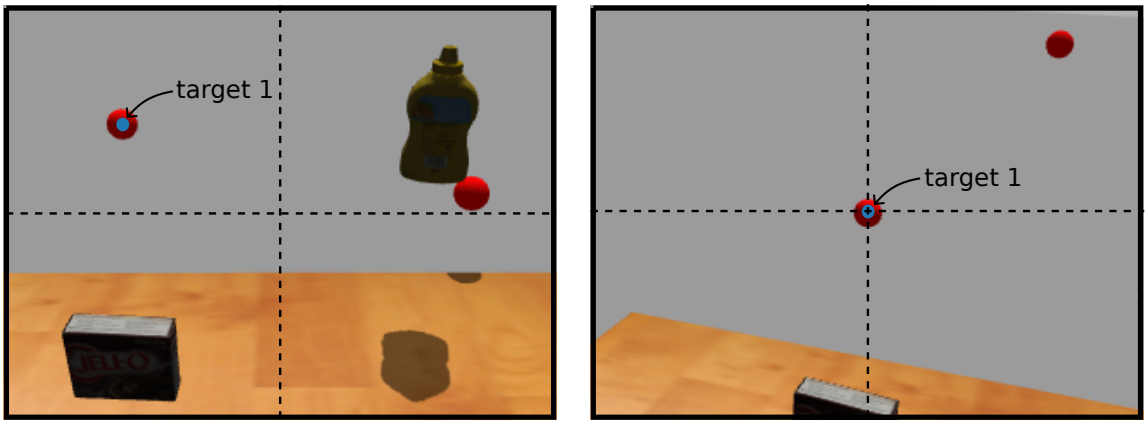


Figure 5.3: Gaze control experiment in simulation. Position of Target 1 before (left) and after (right) gaze control. The dashed lines indicate the centre of the FoV.

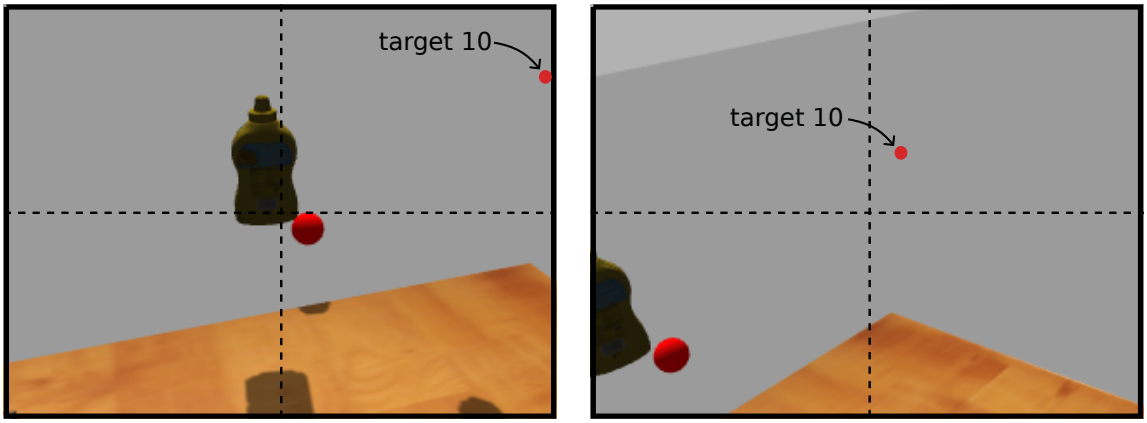


Figure 5.4: Gaze control experiment in simulation. Position of Target 10 before (left) and after (right) gaze control. The dashed lines indicate the centre of the FoV.

5.2.2 Gaze control on the real robot

For the experiment with the real robot, we placed 12 items (gaze targets) on the table in front of the robot, as seen in Fig. 5.5. Figure 5.6 shows the trajectories of 12 specified targets. The black point on each trajectory shows the starting position of a target. The trajectory of target 6 marked in black and the trajectory of target 8 marked in brown do not end in the center of the field of view (point $[640, 360]$). For the other targets, the trajectories end at the point center of FoV. The trajectories of the targets in this experiment are more jerky than the trajectories of the targets in a similar experiment in simulation (see Fig. 5.2) because here the head motion is affected by the real robot parameters.

The movement of gaze towards the target is provided by the *pitch* and *yaw* joints. The value of their joint coordinates during the experiment is shown in Fig. 5.7. The figure shows how the distance of the current gaze target from the center of the FoV (red curve) decreased with changing joint coordinates of the *pitch* (blue curve) and *yaw* joints (green curve). The plot also explains why the gaze control was not completed successfully for Target 4 and Target 8; in the case of Target 4, the *pitch* joint reached its safety limit (dashed blue line).

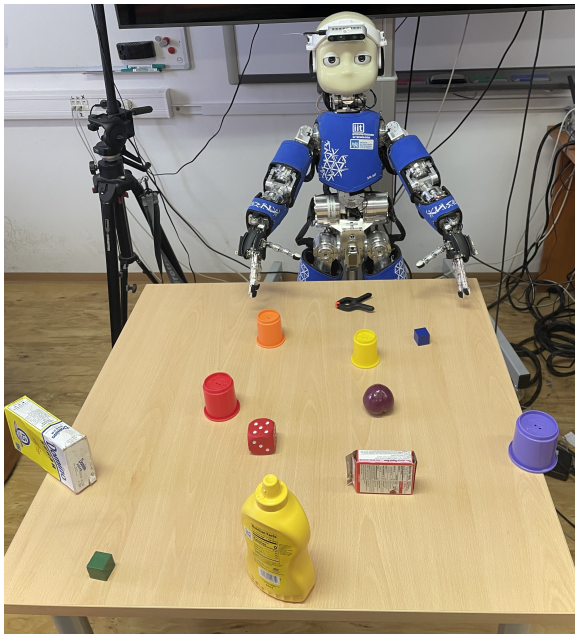


Figure 5.5: Gaze control experiment on the real robot. Scene during the experiment.

In gaze control on Target 8, neck joint *yaw* reached its safety limit (dashed green line).

Figure 5.8 shows the FoV of the robot before (left) and after (right) gaze control on Target 4. The image shows that the gaze control was successful as the target is located in the center of the FoV. On the other hand, Figure 5.9 shows the FoV before (left) and after (right) gaze control for Target 8, where the robot failed to get the target to the center of the FoV due to the limits of the neck joints.

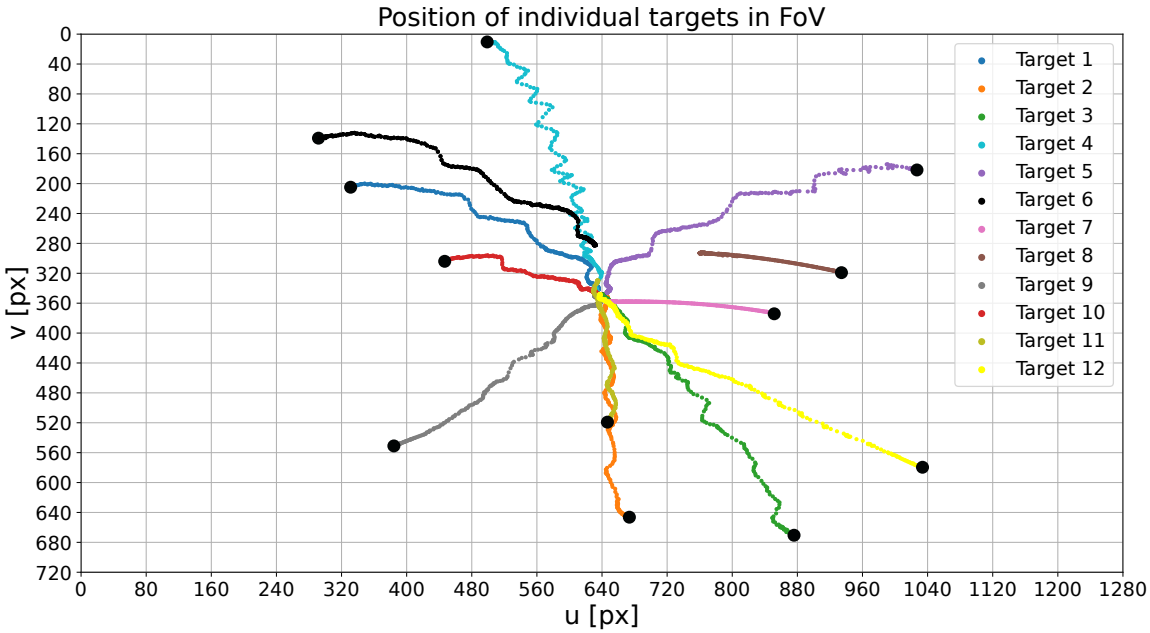


Figure 5.6: Gaze control experiment on the real robot. Trajectories of individual targets in FoV. The black point on each trajectory shows the starting position of a target.

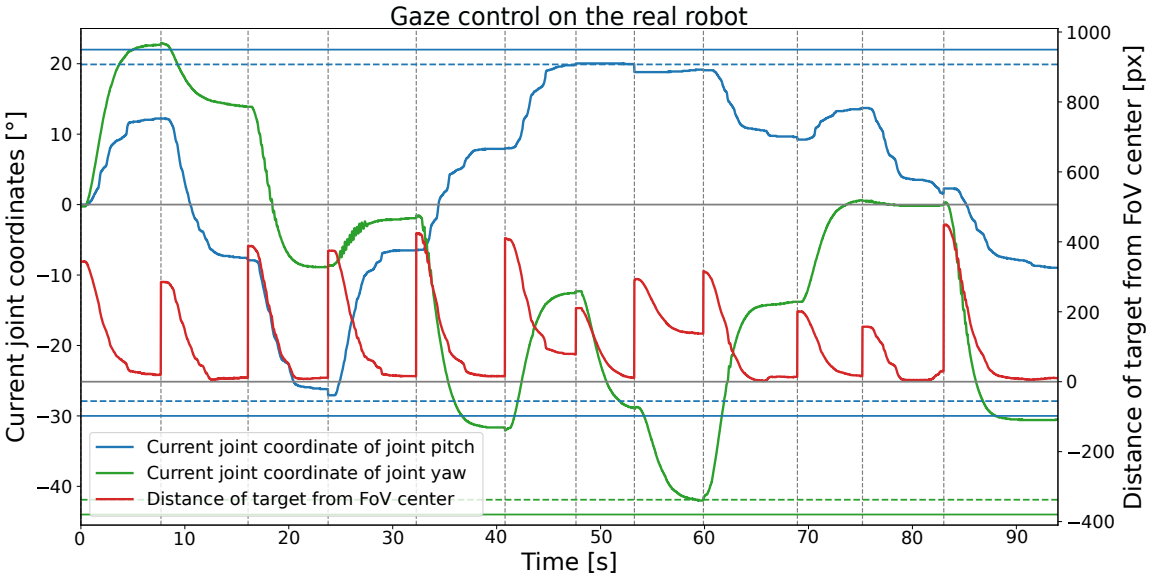


Figure 5.7: Gaze control experiment on the real robot. Joint coordinates of joints *pitch* (blue curve) and *yaw* (green curve) and distance of gaze targets from FoV center (red curve). The vertical dashed lines indicate when the new target was entered and the horizontal lines with the corresponding colors mark the hardware limits (solid line) and the safety limits (dashed line) of *pitch* and *yaw* joints.

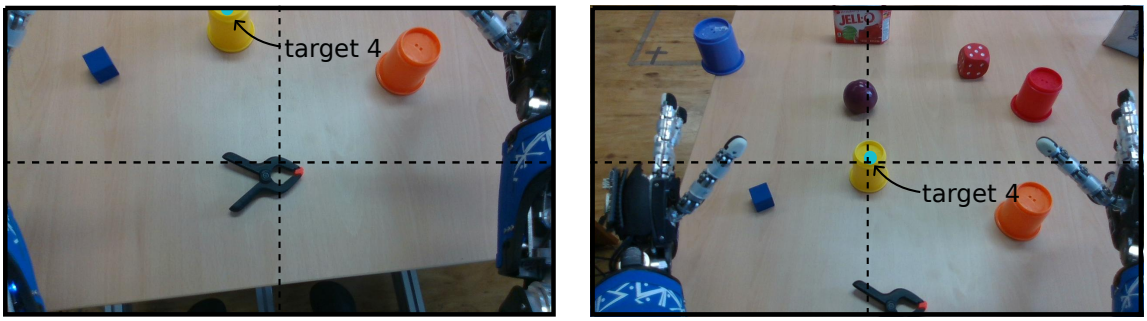


Figure 5.8: Gaze control experiment on the real robot. Position of Target 4 before (left) and after (right) gaze control. The dashed lines indicate the centre of the FoV.

Figure 5.9: Gaze control experiment on the real robot. Position of Target 8 before (left) and after (right) gaze control. The dashed lines indicate the centre of the FoV.

5.3 Gaze stabilization

In this section we present experiments evaluating the quality of gaze stabilization in simulation (see Sec. 5.3.1) and on the real robot (see Sec. 5.3.2).

The evaluation of the gaze stabilization quality was performed using the *optical flow* algorithm RLOF [37] described in Sec. 3.3. The output of this algorithm is the Cartesian motion vectors of individual pixels between two consecutive images. For comparison of the approaches, we calculated the average amplitude of the motion vectors per pixel per image.

The robot's surroundings were fixed during the experiment and the robot's arms were moved to a position such that they did not interfere with the realsense field of view (FoV). Thus, any pixel movements in the camera image during gaze stabilization were induced only by the torso motions.

5.3.1 Gaze stabilization in simulation

The aim of the experiment in simulation was to evaluate and compare the quality of the algebraic solution for gaze stabilization (see Sec. 4.3) and naive gaze stabilization (see Sec. 3.2). The torso performed a total of seven different movements during the experiment, first moving each of the torso joints individually, then combining the joint movements in pairs, and finally moving all the torso joints simultaneously. Table 5.2 shows the results for each type of gaze stabilization and for all combinations of torso joint movements.

Torso motion	1. Stab.	2. Stab. + horiz.	3. N. stab. (α=0)	4. N. stab. (α=1)
pitch	0.05 px	**0.04** px	0.22 px	0.11 px
roll	0.33 px	**0.04** px	0.71 px	0.23 px
yaw	**0.05** px	**0.05** px	0.10 px	0.06 px
pitch, *roll*	0.94 px	**0.05** px	0.37 px	0.50 px
pitch, *yaw*	0.23 px	**0.16** px	0.36 px	0.28 px
roll, *yaw*	0.66 px	**0.10** px	0.59 px	0.54 px
pitch, *roll*, *yaw*	1.46 px	**0.11** px	0.44 px	0.80 px

Table 5.2: Gaze stabilization experiment in simulation. Average amplitude of motion vector of one pixel in one frame for four type of gaze stabilization (1. gaze stabilization, 2. gaze stabilization with horizontal view, 3. naive gaze stabilization with α=0, 4. naive gaze stabilization with α=1) and seven types of torso motion. Lower values are better.

For algebraic gaze stabilization (1.), it is noticeable that it is the worst at stabilizing torso motions with *roll* rotation. This problem is due to the fact that the goal of algebraic gaze stabilization is to keep the target at the center of the FoV using the *pitch* and *yaw* neck joints, not to ensure minimal pixel motion, and thus the view rotates around the center. On the other hand, naive gaze stabilization (3.) uses all three neck joints and compensates for the *roll* joint of the torso, which gives better results than algebraic gaze stabilization (1.) in this case. However, the results show that by using the redundant neck joint *roll* in the algebraic stabilization (2.), these rotations can also be compensated. Therefore, algebraic stabilization with horizontal view is the best type of stabilization if we want to

keep the target in the center of the FoV and minimize the movement around the target. As mentioned in Sec. 3.2, the results of naive gaze stabilization can be improved by changing the parameter α. The results for $\alpha = 1$ (4.), are better than for $\alpha = 0$ (3.) in most cases.

5.3.2 Gaze stabilization on the real robot

The task of this experiment was to evaluate the gaze stabilization on the real robot while its torso was moving in random directions with the three joints simultaneously. The movement of the torso was provided by the iCubBreather module [1], which generates small random body movements as a baseline naturalistic behavior for social human-robot interaction. The speed of the torso joints is adjustable and is given in degrees. We evaluated the quality of stabilization using the *optical flow* algorithm for speeds of $5.0\,^\circ\,\mathrm{s}^{-1}$ and $10.0\,^\circ\,\mathrm{s}^{-1}$.

Figure 5.10 shows average amplitude of motion vectors in each frame. The torso movements during the experiment were random and therefore the curves in the graph should not be compared. As in simulation experiment, we computed the average motion vector amplitudes of one pixel in one frame, for torso speed $5\,^\circ\,\mathrm{s}^{-1}$ it was 3.95 px and for torso speed $10\,^\circ\,\mathrm{s}^{-1}$ 5.49 px. Videos from the experiment for both torso speeds can be found in the folder [40].

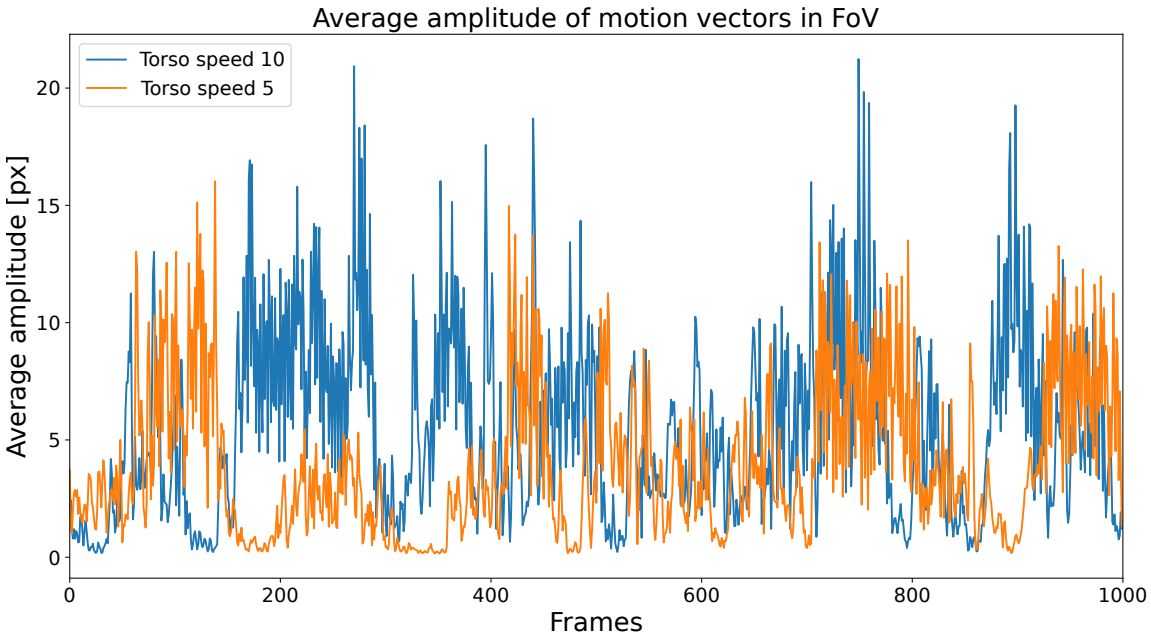


Figure 5.10: The graph shows the average amplitude of motion vectors in each frame at torso velocities of $5\,^\circ\,\mathrm{s}^{-1}$ (orange curve) and $10\,^\circ\,\mathrm{s}^{-1}$ (blue curve). Note that the torso movements during the experiment were random and therefore the curves should not be compared.

[1] `https://robotology.github.io/funny-things/module/iCubBreather.html`

5.4 Secondary task experiments

This section presents experiments that illustrate the utilization of the redundant roll joint in the two secondary tasks defined in Sec. 4.4. In Section 5.4.1, the experiments demonstrated the behavior of the redundant roll joint with a second view target also specified. An experiment to illustrate the use of redundancy for the horizontal view is described in Sec. 5.4.2.

5.4.1 Second target

This experiment is intended to illustrate how the redundant joint *roll* behaves when a second view target is also specified in simulation environment. When two gaze targets are specified, the gaze controller controls the neck joints so that the main target is located at the center of the FoV after the gaze movement, while the second target remains visible during the movement. The visibility of the second target is provided by the redundant neck joint *roll*. This is done by tilting the FoV so that the second target approaches its diagonals; a more detailed explanation can be found in Sec. 4.4.1.

We performed five runs for different main targets, and in the first, second, and fourth runs, we also specified a second target, see Tab. 5.3. The position of the targets is determined from the camera image taken after reaching the previous target. Figure 5.11 shows the camera image before the control started (on the left) and after the control finished (on the right) for Run 2. The main target is marked with a red point and the second with an orange point. The image shows that after the end of the control loop, the second target was on the FoV diagonal.

Run	Position of main target [px]	Position of sec. target [px]	loss of visibility
Run 1	[253, 75]	[175, 7]	no
Run 2	[117, 226]	[302, 169]	partly
Run 3	[89, 223]	-	-
Run 4	[250, 23]	[187, 147]	no
Run 5	[140, 172]	-	-

Table 5.3: Second target experiment. Position of the main and second targets in FoV and whether visibility of the second target was lost during gaze control.

Figure 5.12 shows the trajectories of the movements in the FoV of all three second targets. The area marked in blue shows the central circle with radius 100 px. If the second target is in this space, the value of qdNeck for the *roll* joint is not changed; see Sec. 4.4.1 for an explanation. This figure also shows that the second target from Run 3 was outside the FoV for a while. The *roll* joint was unable to rotate the view to ensure uninterrupted visibility due to its limits. The other two targets remained fully visible during the individual gaze movements.

Figure 5.13 shows how the value of the *roll* joint coordinate changed depending on the position of the second targets in FoV if they were specified. The data between moving gaze to targets are not shown, as they are not relevant for this experiment.

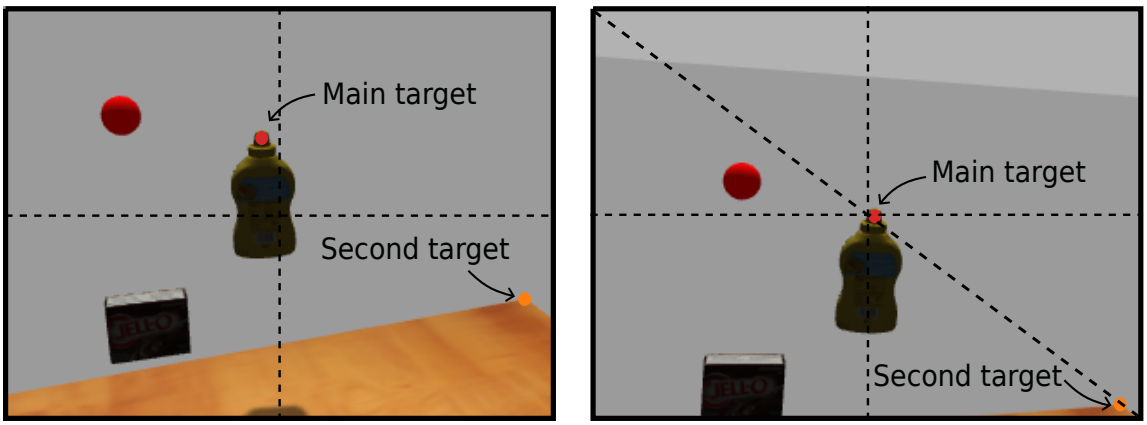


Figure 5.11: Second target experiment. Position of main and second targets in Run 2 before (left) and after (right) gaze control. The dashed lines indicate the centre of the FoV, in the right picture, in addition, a diagonal is marked.

Since the second target specified in Run 1 was initially located in quadrant I, the *roll* joint began to rotate in the negative direction, to point the diagonal passing through quadrant I towards this target. However, due to the rotation of the *pitch* and *yaw* joints, which were simultaneously trying to control the view to the main target, the second target moved to the center circle, as indicated by the blue curve that changed to False (also visible in Fig. 5.12). Thus, it can be seen that the angle β between the second target and the nearest diagonal increased before it reached the vertical axis of FoV. Then β decreased as the object moved to another quadrant, and hence the nearest diagonal was different. The value of β becomes constant when the second target hit the central circle and its position ceased to play a role. The gaze was focused only on the main target before the second target left the center circle.

The second target entered in Run 2 was successfully controlled to the diagonal by the *roll* joint, so the angle value β fell to zero. The second target did not deviate from the diagonal until the end of the gaze-stabilization loop. In Run 3, the second target was not specified, so the *roll* joint returned to its starting position during the view shift to the main target, so that the head did not rotate unnecessarily. The second target in Run 4 could not be brought close enough to the diagonal due to joint limits, and therefore its visibility was partially lost during gaze movement. Run 5, like Run 3, did not contain a second target and the *roll* joint gradually returned to the zero position.

5.4.2 Horizontal view

The purpose of this experiment was to show how the rotation of the neck joint *roll* depends on the rotation of the torso joint *roll* and how the ϕ representing the angle of rotation of the FoV relative to the environmental horizon changes during motion. This angle should ideally be zero, which would mean that the view is horizontal. To eliminate this rotation, redundancy can be exploited, and the *roll* neck joint can compensate for the movement of the *roll* torso joint to provide a horizontal view. The experiment was carried out in simulation, the robot was in STABILIZATION state and the joint *roll* of the torso performed

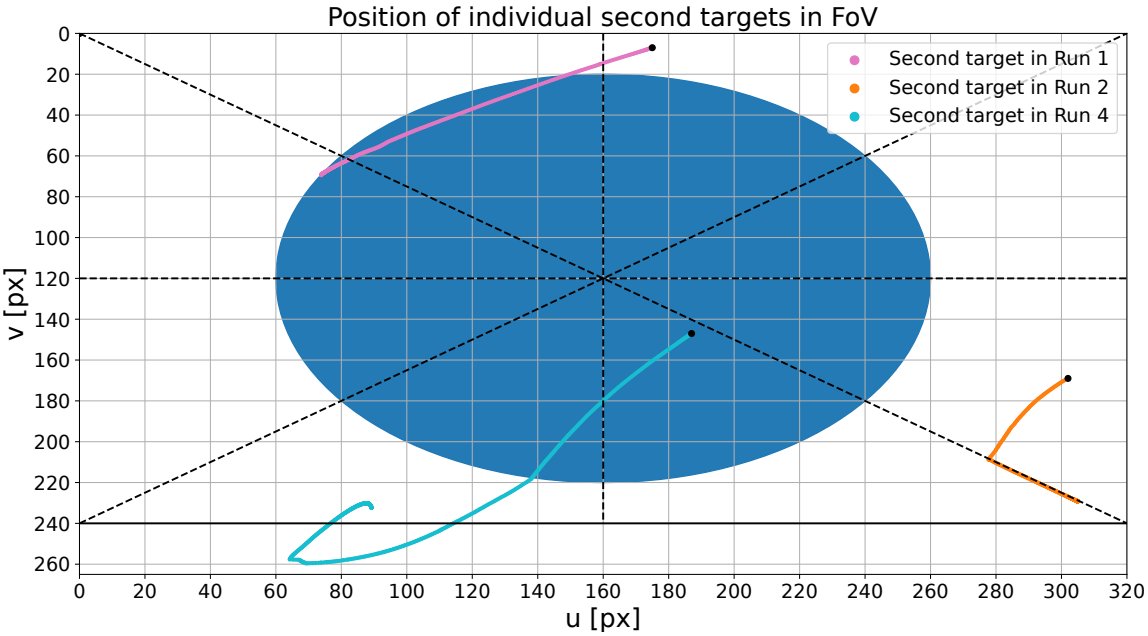


Figure 5.12: Second target experiment. Position of individual second targets in FoV with 320 pixels wide and 240 pixels high. The black points mark the beginnings of the trajectories of the movements of the second targets. The dashed lines mark the centre and diagonals of the FoV.

the motion.

The values of the joint coordinates and the angle ϕ during the experiment are visualized in Fig. 5.14. The neck roll joint has the opposite sign of rotation to the torso roll joint, so if the joints take the same values, the ϕ angle is zero and the view is horizontal. In cases where the neck joint *roll* reached its limit (marked by the dashed brown line in the figure) and was thus unable to compensate for the torso movement, the angle ϕ increased and the view was no longer horizontal.

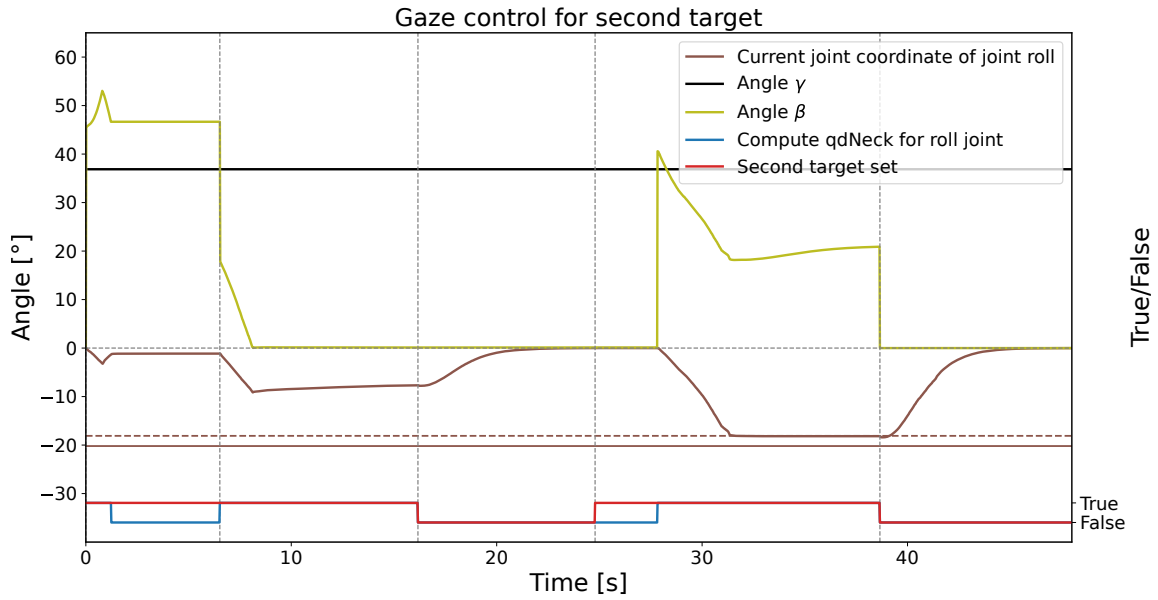


Figure 5.13: Second target experiment. Joint coordinates of the joint roll (brown curve) and his hardware limit (solid brown line) and safety limit (dashed brown line). The vertical dashed lines indicate when the new target was entered, and the red curve becomes True if a second target has been entered. The blue curve takes on the True value when the qdNeck value for the *roll* joint is updated, so that the second target is not in the center circle and is reachable by gaze. The value of the angle γ, which aligns the second target with the diagonal FoV, is plotted in olive color.

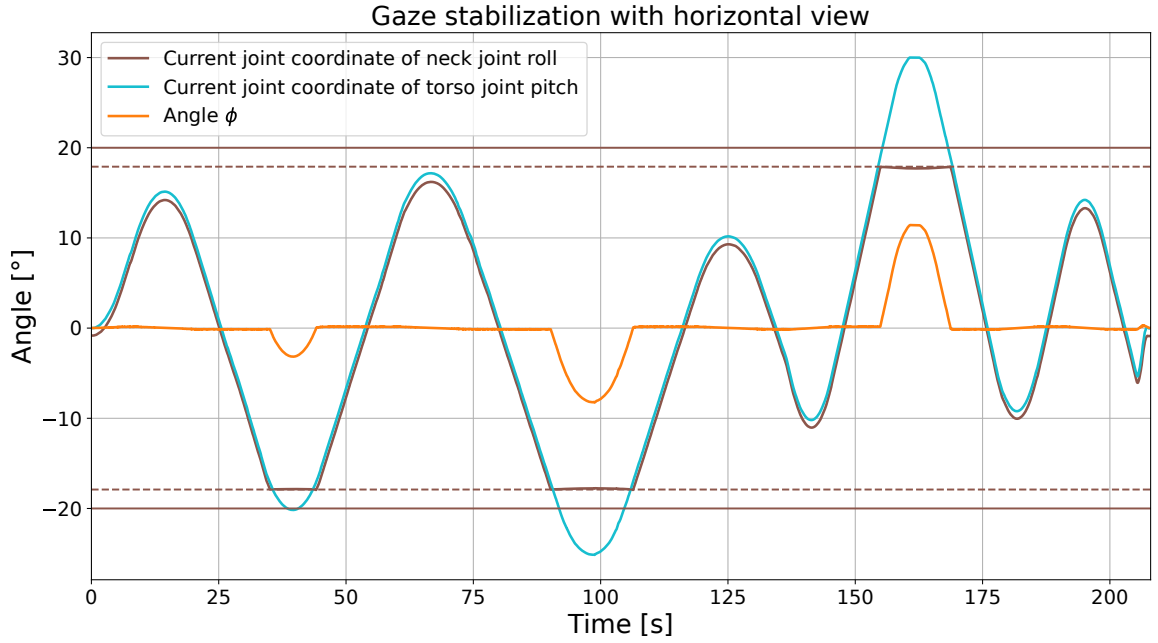


Figure 5.14: Horizontal view experiment. Dependence of the neck *roll* joint coordinate and the ϕ angle on the torso *roll* joint coordinate during gaze stabilization with horizontal view. The angle ϕ represents the rotated of the robot's FoV relative to the environmental horizon. The brown horizontal line represents the hardware limit of the *roll* neck joint, the dashed line represents its safety limits.

5.5 Bubbles game

FInal experiment demonstrates the use of the Gaze Controller described in Sec. 4.5 in a human-robot interaction scenario. In particular, the gaze controller is used during a children's game called Bubbles against a human opponent. Bubbles is a children's game by Piatnik [2] that requires a smart eye and a quick reaction. The game contains 4 colored dice (red, blue, green, and yellow) with numbers 1-24, so each dice has different numbers. The game also contains cards with pictures of four differently sized bubbles whose colors correspond to the colors of the dice. There are 24 cards in total, one for each combination of sizes and colors. The goal of the game is to evaluate as quickly as possible after the dice are rolled which card matches the order of the sizes of the colored bubbles with the order of the sizes of the numbers on the colored dice and to mark that card.

For our experiment, we used only six cards (those where the yellow bubble is always the smallest one), so that there are not too many cards on the table and the robot is able to reach all of them. The game is therefore simplified to only three dice (red, blue, and green). Figure 5.15 shows what the game scene looked like and Fig. 5.16 shows the cards and dice.

The Gaze controller is employed for visual control of the game. When the robot reaches over a card to mark it, the RGB-D camera view is controlled to look at the center of that card (see Fig. 5.18). On the other hand, when the robot returns to the starting position and prepares for the next round of play, the camera view is directed to the table so that all cards are in FoV and the robot can again detect and distinguish individual cards (see Fig. 5.17). This allows the robot to see most of the cards throughout the game, allowing it to be more robust and responsive to any changes in card positions. For arm and torso movement during the game we employed HARMONIOUS controller [41]. A video from experiment can be found in [40].

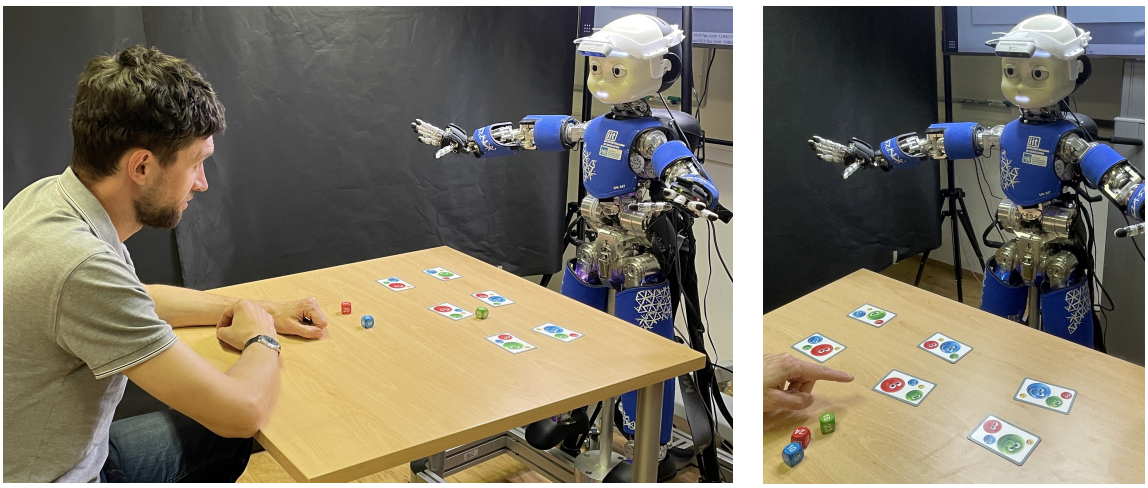


Figure 5.15: Bubbles game experiment. Scene of experiment.

[2] https://www.piatnik.com/en/games/board-games/children-games/bubbles.html

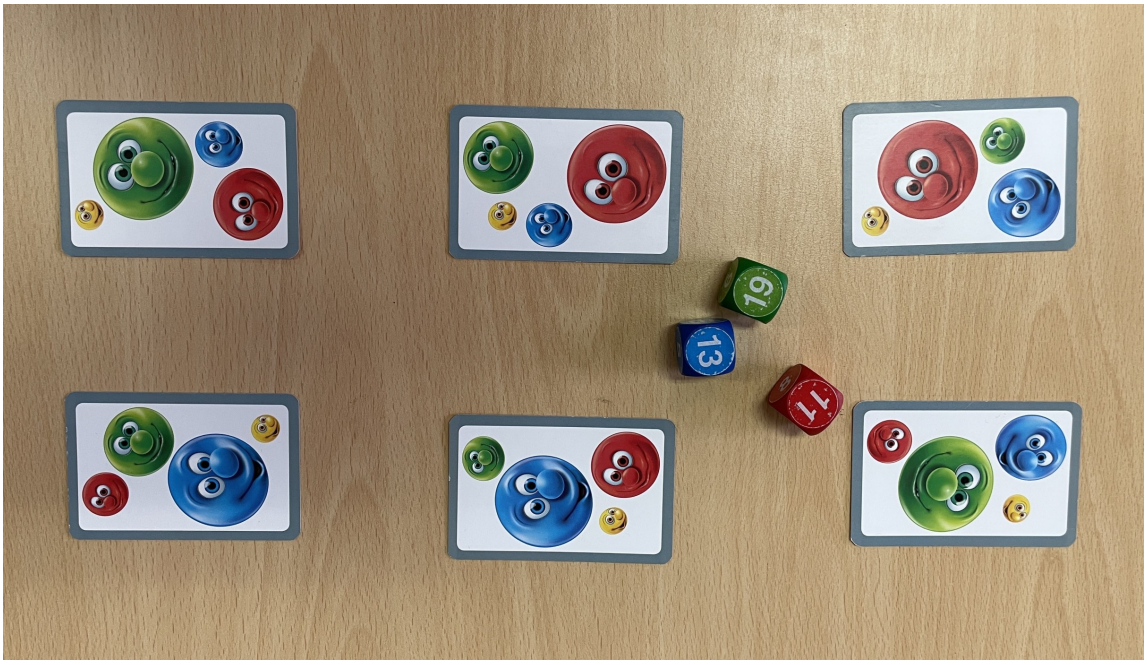


Figure 5.16: Bubbles game experiment. The subset of cards and dice that were used for the experiment.

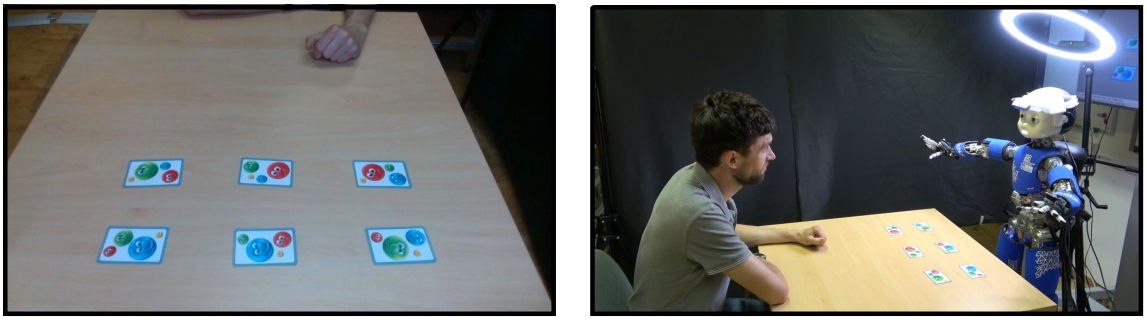


Figure 5.17: Bubbles game experiment. Camera view (left) and scene (right) before target entry.

Figure 5.18: Bubbles game experiment. Camera view (left) and scene (right) when marking the card.

Chapter 6

Discussion, Conclusion and Future Work

6.1 Conclusion

In this thesis, we presented a gaze controller for the humanoid robot iCub, which allows to control and stabilize the gaze using the neck joints. The controller allows for focusing the robot's gaze on a given target and maintaining the gaze even when the robot's torso starts to move. The gaze target can be specified as a 3D point in the robot's torso frame or as a point in its FoV. The neck joints *pitch* and *yaw* are used to focus the view on the specified target. The computation of the desired joint coordinates of these joints is based on algebraic inverse kinematics and knowledge of the robot structure. The source code of the controller is available in a GitLab repository [1].

We showed how the redundant neck joint *roll* can be exploited for other tasks by implementing two secondary tasks—tracking two targets and maintaining a horizontal view. When the robot is given two gaze targets, the gaze controller controls all three neck joints to point the first target to the center of the FoV while not losing visibility of the second target. The visibility of the second target during motion is maintained (if feasible) by rotating the head with the joint *roll*. Maintaining a horizontal gaze helps to minimize pixel movement in the robot's FoV during stabilization, since the movement of the torso joint *roll* is compensated by the neck joint *roll*. This can be useful if the image is being processed for other tasks and a large motion would make analysis impossible. The functionality of all the implemented methods—gaze control, gaze stabilization, and use of redundancy—has been experimentally tested in simulation and some of them on the real robot.

An experiment demonstrating the functionality of gaze control showed that the gaze controller successfully focuses the camera's gaze so that the target is at its center in all cases where the limits of the neck joints allow it to do so. As seen in Fig. 5.2 for the simulation experiment and in Fig. 5.7 for the real robot experiment. Then we evaluated our gaze stabilization using an *optical flow* metric and compared it to a baseline naive gaze stabilization method. The comparison of the results in Tab. 5.2 shows that our algebraic stabilization with the horizontal view minimizes the pixel motion in the image best.

The movement of the second targets during gaze control can be seen in the Fig. 5.12, the data plotted in Fig. 5.13 and Tab. 5.3 show that the gaze controller will maintain visibility of the second target if the limits of the roll joint allow it. Similarly, within its limits, the neck *roll* joint compensates for the movement of the torso *roll* joint to provide a horizontal camera view as seen in Fig. 5.14.

6.2 Discussion

The gaze controller presented in this thesis is designed for the iCub robot and robots with the same neck joint arrangement. Thus, the controller cannot be used for an arbitrary robot without additional modifications.

The evaluation of the stabilization quality using *optical flow* was affected by the fact that the iCub robot used for the experiments does not have a perfectly working PID controller for the yaw joint. This problem causes an occasional oscillating motion of this joint, causing the image to move, but not because of the inaccuracy of the stabilization. This oscillating motion was also observed in the experiment for gaze control in Fig. 5.7.

Another factor that can affect the gaze stabilization evaluation is the number of camera frames from which the quality is evaluated. With fewer images, there is more pixel movement between two consecutive images than when the same length of movement is recorded in more images. It was not possible to ensure exactly the same number of images during the experiment, but the time differences between the images were similar enough; thus it should not have a significant effect on the results.

The joint coordinate profiles from the real robot during reaching experiment (see Fig. 5.7) are much more jerky than those of simulation (see Fig. 5.2). This behavior is caused not only by the problematic PID controller of the neck yaw joint, but also, in general, by the parameters and properties of the real robot, which are not ideal and affected, for example, by friction. Therefore, the simultaneous movement of two neck joints causes jerky motion. The motion is smooth when only one joint is moving, as can be seen in Fig. 5.7.

6.3 Future work

The solution for gaze control and stabilization proposed in this thesis depends on the arrangement of the neck joints of the humanoid robot iCub, which is described in Sec. 3.1.2. A straightforward extension of the controller is to generalize the calculations to other robots that have a different arrangement of at least two independently controllable neck joints. Thus, it would be necessary to modify the computation of the joint coordinates so that either they are applicable to all types of neck structures or their computation is able to adapt to a specific joint arrangement. Moreover, we could implement functions for retrieving the inverse kinematics solutions to test the feasibility of the problem without performing the actual movements.

In addition to the already implemented secondary tasks, others could be added, such as self-occlusion avoidance. *Roll* neck joint rotation could ensure that the object of interest is not only in the center of the FoV but it is visible as well, *i.e.*, not covered by, for example, the robot's hands. For this purpose, it would be necessary to determine where the robot's body is to evaluate whether self-occlusion has occurred or not and find joint coordinates of the redundant joint to ensure the visibility of the target. This algorithm would increase the level of robustness of our solution. Finally, the iCub robot has an inertial measurement unit (IMU) placed in its head. We could take inspiration from [34] and explore whether the IMU data could be used in our gaze stabilization solution.

Bibliography

[1] Z. Jindrova, "iCub neck controller," https://gitlab.fel.cvut.cz/body-schema/icub/icub-neck-controller, accessed: May 23, 2024.

[2] E. Cuevas, D. Zaldivar, and R. Rojas, "Neurofuzzy prediction for gaze control," *Canadian Journal of Electrical and Computer Engineering*, vol. 34, no. 1/2, pp. 15–20, 2009.

[3] T. Groueix, G. Ponimatkin, V. Lepetit, T. Hodan *et al.*, "CNOS: A Strong Baseline for CAD-based Novel Object Segmentation," in *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. IEEE, 2023, pp. 2126–2132.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[5] D. Omrčen and A. Ude, "Redundant control of a humanoid robot head with foveated vision for object tracking," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 4151–4156.

[6] D. Maier, A. Hornung, and M. Bennewitz, "Real-time navigation in 3D environments based on depth camera data," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 2012, pp. 692–697.

[7] J. Biswas and M. Veloso, "Depth camera based indoor mobile robot localization and navigation," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 1697–1702.

[8] S. R. Fanello, U. Pattacini, I. Gori, V. Tikhanoff, M. Randazzo, A. Roncone, F. Odone, and G. Metta, "3D stereo estimation and fully automated learning of eye-hand coordination in humanoid robots," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 1028–1035.

[9] M. Vargas-Signoret, M. Rojas-Romero, I. Trejo-Ávila, J. Velasco-Avella, E. Robles-Martínez, M. Santoyo-Mora, K. Camarillo-Gómez, G. Pérez-Soto, and L. Morales-Hernández, "Depth map construction with stereo vision for humanoid robot navigation," in *2016 XVIII Congreso Mexicano de Robotica*. IEEE, 2016, pp. 1–6.

[10] S. Schulz, F. Lier, A. Kipp, and S. Wachsmuth, "Humotion: A Human Inspired Gaze Control Framework for Anthropomorphic Robot Heads," in *Proceedings of the Fourth International Conference on Human Agent Interaction*, ser. HAI '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 207–214.

[11] I. Farkhatdinov, H. Michalska, A. Berthoz, and V. Hayward, "Review of anthropomorphic head stabilisation and verticality estimation in robots," *Biomechanics of anthropomorphic systems*, pp. 185–209, 2019.

[12] T. Habra, M. Grotz, D. Sippel, T. Asfour, and R. Ronsse, "Multimodal gaze stabilization of a humanoid robot based on reafferences," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 47–54.

[13] T. Shibata and S. Schaal, "Biomimetic gaze stabilization based on feedback-error-learning with nonparametric regression networks," *Neural Networks*, vol. 14, no. 2, pp. 201–216, 2001.

[14] A. Lenz, T. Balakrishnan, A. Pipe, and C. Melhuish, "An adaptive gaze stabilization controller inspired by the vestibulo-ocular reflex," *Bioinspiration & biomimetics*, vol. 3, no. 3, p. 035001, 2008.

[15] T. Habra and R. Ronsse, "Gaze stabilization of a humanoid robot based on virtual linkage," in *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2016, pp. 163–169.

[16] G. Milighetti, L. Vallone, and A. De Luca, "Adaptive predictive gaze control of a redundant humanoid robot head," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3192–3198.

[17] G. Schweigart, T. Mergner, I. Evdokimidis, S. Morand, and W. Becker, "Gaze Stabilization by Optokinetic Reflex (OKR) and Vestibulo-ocular Reflex (VOR) During Active Head Rotation in Man," *Vision Research*, vol. 37, no. 12, pp. 1643–1652, 1997.

[18] C. Distler and K. Hoffmann, "The optokinetic reflex," *The Oxford handbook of eye movements*, pp. 65–83, 2011.

[19] O. A. Masseck and K.-P. Hoffmann, "Comparative neurobiology of the optokinetic reflex," *Annals of the New York Academy of Sciences*, vol. 1164, no. 1, pp. 430–439, 2009.

[20] A. Roncone, U. Pattacini, G. Metta, and L. Natale, "A Cartesian 6-DoF Gaze Controller for Humanoid Robots." in *Robotics: science and systems*, vol. 2016, 2016.

[21] K. He, R. Newbury, T. Tran, J. Haviland, B. Burgess-Limerick, D. Kulić, P. Corke, and A. Cosgun, "Visibility Maximization Controller for Robotic Manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8479–8486, 2022.

[22] A. N. Alshakhs, M. F. Mysorewala, A.-W. A. Saif, and K. Alshehri, "A Novel Algebraic Inverse Kinematics Based Approach to Gaze Control in Humanoid Robots," *IEEE Access*, vol. 11, pp. 50 350–50 363, 2023.

[23] S. Stifter, "Algebraic methods for computing inverse kinematics," *Journal of Intelligent and Robotic Systems*, vol. 11, pp. 79–89, 1994.

[24] L. Sciavicco and B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 403–410, 1988.

[25] M. Baumann, S. Léonard, E. A. Croft, and J. J. Little, "Path planning for improved visibility using a probabilistic road map," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 195–200, 2010.

[26] E. Marchand and G. D. Hager, "Dynamic sensor planning in visual servoing," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 3. IEEE, 1998, pp. 1988–1993.

[27] D. Nicolis, M. Palumbo, A. M. Zanchettin, and P. Rocco, "Occlusion-free visual servoing for the shared autonomy teleoperation of dual-arm robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 796–803, 2018.

[28] K. Tarabanis, R. Y. Tsai, and A. Kaul, "Computing occlusion-free viewpoints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 3, pp. 279–292, 1996.

[29] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick, "Real-time collision avoidance with whole body motion control for humanoid robots," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 2053–2058.

[30] M. Grotz, T. Habra, R. Ronsse, and T. Asfour, "Autonomous view selection and gaze stabilization for humanoid robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1427–1434.

[31] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori, "The iCub humanoid robot: an open platform for research in embodied cognition," in *Proceedings of the 8th workshop on performance metrics for intelligent systems*, 2008, pp. 50–56.

[32] M. Randazzo and L. Taverna, "iCub Reference Frames," https://icub-tech-iit.github.io/documentation/icub_kinematics/icub-forward-kinematics/icub-forward-kinematics/, accessed: May 23, 2024.

[33] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Comput. Surv.*, vol. 27, no. 3, p. 433–466, sep 1995.

[34] A. Roncone, U. Pattacini, G. Metta, and L. Natale, "Gaze stabilization for humanoid robots: A comprehensive framework," in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 259–264.

[35] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, 1981, pp. 674–679.

[36] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*. Springer, 2003, pp. 363–370.

[37] T. Senst, V. Eiselein, and T. Sikora, "Robust Local Optical Flow for Feature Tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 9, pp. 1377–1387, 2012.

[38] "Optical Flow in OpenCV," https://github.com/spmallick/learnopencv/tree/master/Optical-Flow-in-OpenCV, accessed: May 23, 2024.

[39] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *Journal of Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[40] Z. Jindrová, 2024. [Online]. Available: https://drive.google.com/drive/folders/1iV60Wm0-1vS12fcLPpwGpbKLujax6WW4?usp=drive_link

[41] J. Rozlivek, A. Roncone, U. Pattacini, and M. Hoffmann, "HARMONIOUS–Human-like reactive motion control and multimodal perception for humanoid robots," *arXiv preprint arXiv:2312.02711*, 2023.