GRAZ UNIVERSITY OF TECHNOLOGY

INSTITUTE OF ELECTRICAL MEASUREMENT AND
MEASUREMENT SIGNAL PROCESSING

MASTER'S THESIS

Vision-based pick up of objects with the humanoid robot Nao

Author:
Thomas HÖLL

Supervisor:
Ao. Univ.-Prof. Dipl.-Ing.
Dr. techn. Axel PINZ

January 17, 2012

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am ……. ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

……. ……..
date (signature)

SO TEACH US TO NUMBER OUR DAYS,
THAT WE MAY APPLY OUR HEARTS UNTO WISDOM.
A prayer of Moses

Abstract

The goal of this work is to enable the humanoid robot Nao to pick up foam cubes from the floor. Based on image measurements Nao detects highly saturated, colored foam cubes and approaches them. The cubes are specified by their histograms of their hue and saturation channels. During the walking Nao receives feedback from the camera and adapts its walking direction and speed. When it is finally next to a cube, it bows down and picks up the cube. In the grasping part, the right arm of Nao is controlled, also based on image measurements. For this purpose, the thumb of Nao's right arm is projected onto the floor plane. The resulting image point is used as the input for visual servoing. After the grasping, Nao checks if the cube is in its hand. If this is not the case it stands up and tries to pick up the cube once more. In the experiments, Nao was able to pick up the cube in 88 % of the cases. The presented approach was also successfully tested on another Nao humanoid robot.

Kurzfassung

Ziel dieser Arbeit ist, dem humanoiden Roboter Nao zu ermöglichen farbige Schaumstoffwürfel vom Boden aufzuheben. Mit Hilfe von Messungen in den Kamerabildern findet der Roboter die farbigen Würfel und geht auf diese zu. Die Würfel werden durch Histogramme ihres Farbtons und ihrer Sättigung bestimmt. Während dem Gehen werden Richtung und Geschwindigkeit geändert, abhängig von den Kamerabildern. Wenn der Roboter den Würfel erreicht hat, bückt er sich und greift diesen. Dies geschieht ebenso mit Hilfe einer Regelung die auf Bildmessungen beruht. Hierzu wird der Daumen von Nao's rechter Hand auf den Boden projiziert. Der dadurch erhaltene Bildpunkt wird nun als Eingangsgröße für den Regler verwendet. Nach dem Greifen kontrolliert Nao, ob dieses erfolgreich war. Falls dies nicht der Fall ist, steht der Roboter nochmal auf und versucht es erneut. In den durchgeführten Experimenten war Nao in 88% der Fälle in der Lage, den Würfel vom Boden aufzuheben. Der präsentierte Ansatz wurde auch auf einem zweiten Nao ausprobiert und lieferte ähnlich gute Ergebnisse.

Contents

1 Introduction

For humans, picking up an item that lies on the floor is quite an easy task. Every one of us has done this many times before. We are so familiar with this task, that no one will have to concentrate much to solve it properly. But if you are confronted with the problem to teach a robot to fulfil this task, then things are getting quite difficult. Easy tasks, natural for humans, are hard problems for a robot. The goal of this work is to enable the humanoid robot Nao [1] to pick up objects from the floor. Nao perceives its environment with its camera. Based on the camera images, Nao should detect objects, walk towards them and pick them up from the floor. All these should be autonomous actions. That means, Nao should be able to fulfil this task without human support and all processing should run on Nao. A successful pick up on the Nao robot faces several problems:

- limited on-board processing power
- inaccuracy during walking
- blurred images, caused by shakes while Nao is walking
- the absence of a backbone
- pincer shaped fingers of Nao's hands
- limited motion space of Nao's arm

- high joint temperature, caused by long operations

To simplify the pick-up task and to overcome some of the limitations, the environment where Nao operates was designed in the following way:

- The environment itself is without significant color.
- The objects, which should be picked up by Nao, are highly saturated, coloured foam cubes.
- There are no obstacles in the environment.
- Nao operates in a small room ($2m \times 2m$), delimited by a white border.

To get an impression of Nao and its environment, have a look at Figure 1.1.

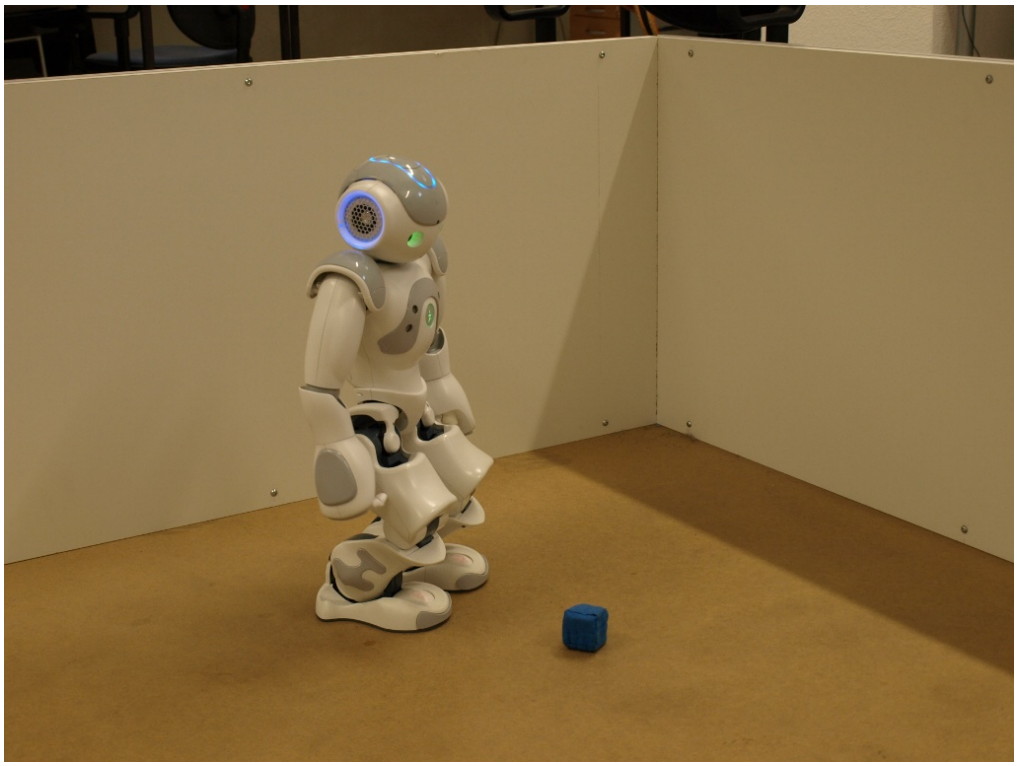


Figure 1.1: The humanoid robot Nao in its "living room". The task is to pick up such small coloured foam cubes. The environment itself is without significant colors, whereas the objects, that Nao should pick up, show highly saturated colors. There are no obstacles in the "living room".

Two main problems were solved independently, to fulfil the pick-up task:

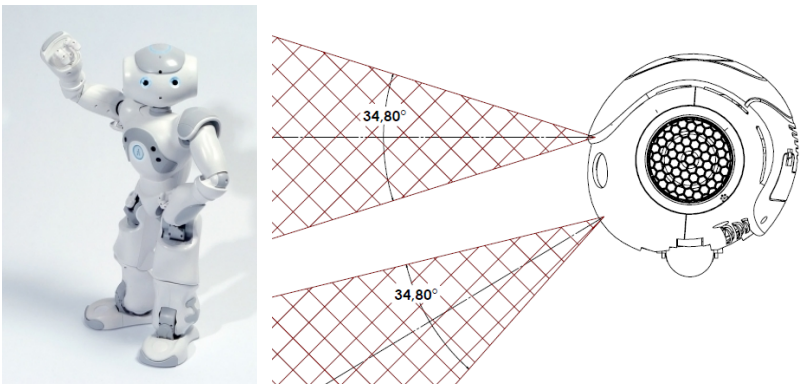
1. Grasping a foam cube
2. Approaching the foam cube

Grasping a foam cube: The process of grasping a foam cube was at first studied by placing Nao next to a small table and let it pick up a foam cube, which lies on the table (this preparatory work was done in my seminar project [2]). This approach, compared to picking up the cube from the floor, has the advantages, that Nao's joints are not getting hot so quickly, because Nao sits in an upright posture. Another advantage of this approach is, that the concept of grasping objects could be studied in a safe and fast way, because Nao does not have to bend down to pick up the foam cube from the floor. The solution of this concept by using visual servoing is flexible and can be adapted easily to grasp foam cubes which lie on the floor.

Approaching the foam cube: Because of the lack of Nao's processing power, it is actually not possible to reconstruct the whole scene and to plan Nao's motions according to this reconstruction. Therefore, other solutions must be chosen. In this work, a reactive scheme is applied. Nao may not be able to reconstruct the whole scene but it is able to respond to the scene. During the walking towards an object, Nao reacts in a proper way according to its visual perception.

1.1 The humanoid robot Nao

Nao [1] is an 85cm tall and 5kg heavy humanoid robot which is produced by a French company, named *ALDEBARAN Robotics*. See figure 1.2(a) to get an idea how Nao looks like. The robot has 25 degrees of freedom. Its onboard processing unit is a 500 Mhz AMD GEODE processor which has access to a 256 MB SDRAM memory. A special version of an embedded Linux operation system is running on it, that allows access to Nao's sensors, joints and other specific hardware. Because this work is using Nao's camera for various tasks, it is helpful that the OpenCV [3] vision library is also pre-compiled and ready to use. The vision system of Nao consists of two cameras, placed in the head of Nao. Because only one camera can be used at a time and because the cameras have no stereo overlap (see figure 1.2(b)), stereo vision is not possible on Nao. Nao's vision system can provide the captured images in various sizes and color space formats. A detailed list can be found in Nao's user guide [1]. This short introduction addresses only some parts of the vision system that are necessary for this work. One such part is the HSY [1] color space. This is an approximation of the HSV (hue, saturation and brightness) color space, as it can be found for example in [4]. The two channels, hue and saturation, are the same in the HSV and HSY color space. The only difference between these two color spaces is the third channel. Since the native color space format of Nao's camera is the YUV422 [1], there is a speed up, if not the whole HSV color space is calculated. Just the hue and



(a) Nao (b) Nao's vision system

Figure 1.2: The humanoid robot Nao and its vision system (from [1]) .

saturation parts are calculated, the Luma channel of the YUV color space is used as an approximation instead of the brightness channel of the HSV color space. This color space (HSY) is used to specify the object, that should be picked up.

The pick up part in this work depends very strongly on Nao's arms. The *inverse kinematic* of Nao's arms is provided by Nao's framework. That allows the user to specify a point in 3D space and Nao then calculates the joint angles in a way, that this point can be reached. Nao's hand consists of three pincer shaped fingers but it is not possible to move them individually.

Nao's framework also provides functions that allow a user to let Nao walk, by giving the desired goal position or goal velocity. A walking pattern generator creates the necessary joint commands and cooperates with a controller, that ensures a stable walk (see [5] for further information).

To enable Nao to bend down, the framework provides methods to record joint angle motions over time and repeat them. Similar to a video recorder, it is possible to record motions and repeat them afterwards. Furthermore it is also possible to fine tune the motions.

1.2 Related work

The humanoid robot Nao is mainly used as the standard platform in the RoboCup soccer league [6]. At the *Institute of Electrical Measurement and Measurement Signal Processing*, previous work with Nao includes: human pose detection and mimicking, and color blob detection [7] (the color blob detection algorithm is also used in this work); categorization of objects, which are handed to Nao. This project uses sophisticated vision and learning algorithms [8]; in my seminar project [2], I have done a lot of preparatory work,

to enable Nao to grasps foam cubes. In this work, Nao was placed next to a table and the goal was to grasp a foam cube, placed on the table in front of it. The presented concept of visual servoing is very flexible and can be adapted to fulfil the pick-up task from the floor.

The idea to fulfil vision based tasks with a robot is not new. For example, in [9] a mobile robot was used to collect trash. The authors proposed "coupling vision and action". Based on image data, the robot was able to pick up trash (in their case, soda cans, styrofoam cups and paper wads) and put it into a trash can. To identify the object, the authors first search for regions of interest (ROI) using a color segmentation approach. After that, they use an edge-based model to identify the different objects. Bollmann et al. [10] are also using camera input to play Domino with a mobile robot. The robot searches for Domino point cluster, and then it performs a template matching to determine which Domino part is in front of it. In both works the mobile robot was not a humanoid robot, instead it was a wheeled one. In [11] Nao also fulfilled a grasping task, where it took a soda can out of the fridge. However, it was controlled by an external observer and the fridge was constructed in a way that helped Nao grasping the can. Especially in RoboCup, robots have to fulfil vision based tasks. For example in [12], Nao hits the ball in front of it, depending on its visual input. In most cases the objects are largely color coded, to simplify the image processing.

Great work was done in the domain of object grasping. Most of these approaches, for example [13], use stereo vision to reconstruct the scene and industrial robot arms to grasp the object. None of these abilities can be found on Nao. Another very interesting approach is [14], in which a supervised learning method is used to learn the point on the object where grasping is possible from synthetic images. However, this approach also uses stereo vision and is focused on the learning task. A comprehensive survey about grasping can be found in [15]. The chapter *grasping* in this book focuses on a mathematical formulation of the grasping problem. With the help of mathematical models for the contact behaviour and rigid-body kinematics and dynamics, the problem of grasping objects is addressed. One of the first robotic hands was the Salisbury hand. This hand has three three-jointed fingers. Compared to the three fingers of Nao, this hand is able to control all six degrees of freedom of an object. Nao's hand has not the capability to do this. Another difference to our setting is, that for the grasping according to this book a model of the object is necessary to plan the grasping position.

To enable Nao to move its hand towards an object, a formalism called *visual servo control* [16], [17] or *visual servoing* [18] is used. In this work the structure of *image-based visual servo* (IBVS) is adapted and implemented. According to image measurements, the hand motion is controlled. To do this, the *interaction matrix* has to be calculated,

this is done by some exploration movements of Nao's arm. Other approaches [21] are calculating this matrix online. In section 2.3 the *visual servo control* approach is explained in more detail.

2

Essentials and their application

In this chapter, mathematical essentials of visual geometry and their applications are explained. Furthermore, the concept of *visual servo control* that is used to control Nao's arm motion is introduced and important results of the use of these essentials are presented.

2.1 The camera model

The camera model is based on the pinhole model, which maps a 3D world point $\tilde{\mathbf{X}} = (X, Y, Z)^T$ to a 2D point $\tilde{\mathbf{x}} = (x, y)^T$ that lies in an image plane. In general, the camera model is a function

$$cam : (X, Y, Z)^T \mapsto \left(\frac{fX}{Z} + p_x, \frac{fY}{Z} + p_y \right)^T \tag{2.1}$$

with f being the focal length and $\mathbf{p} = (p_x, p_y)^T$ the principal point, illustrated in figure 2.1. To achieve more generality, the camera model can be expressed in the homogeneous framework.

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} f_x X + Z p_x \\ f_y Y + Z p_y \\ Z \end{pmatrix} = \begin{bmatrix} f_x & 0 & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.2}$$

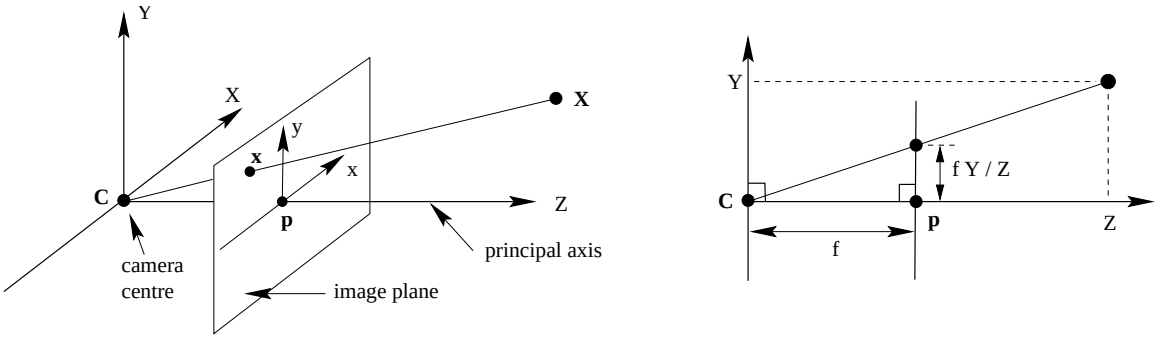


Figure 2.1: Pinhole camera geometry (from [19], p154). Here the mapping of the 3D world point $\tilde{\mathbf{X}}$ onto the image plane is illustrated.

In practice, there may be two different focal lengths: one, according to the x- axis (f_x) and the other one according to the y- axis (f_y). In equation 2.2, the vector $(X, Y, Z, 1)^T$ is the homogeneous form of the vector $(X, Y, Z)^T$.

2.1.1 Camera internal parameter

Extracting the camera internal parameters (f_x, f_y, p_x, p_y) and rewriting equation 2.2 the following result is achieved:

$$\mathbf{x} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]\mathbf{X}, \tag{2.3}$$

with $\mathbf{I}$ being the 3×3 identity matrix, $\mathbf{0}$ a 3-null vector, $\mathbf{X}$ a 3D world point, expressed in homogeneous coordinates, and $\mathbf{K}$ the *camera calibration matrix*

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

The huge advantage of this representation is, that the *external orientation* of the camera can easily be included.

2.1.2 Camera external parameters

Until now it is considered, that all 3D points are represented relative to the camera. Now the camera and the 3D point will be represented relative to a world coordinate frame. Define $\mathbf{R}$ as the 3×3 matrix which represents the camera rotation and $\mathbf{t} = -\mathbf{R}\mathbf{c}^{\mathcal{W}}$ as its translation, both relative to a world coordinate frame $\mathcal{W}$. Here $\mathbf{c}^{\mathcal{W}}$ represents the camera center in the world coordinate frame. It is now possible to compose the *camera matrix* $\mathbf{P}$:

$$\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \tag{2.5}$$

With this matrix, a 3D point $\mathbf{X}^{\mathcal{W}}$, represented in the world coordinate frame $\mathcal{W}$, can be mapped to the homogeneous image point $\mathbf{x}$ with the equation

$$\mathbf{x} = \mathbf{P}\mathbf{X}^{\mathcal{W}} \tag{2.6}$$

Figure 2.2 illustrates the whole concept. Because of the homogeneous formulation of

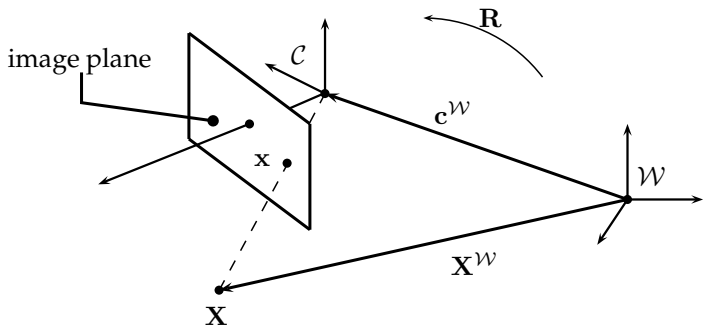


Figure 2.2: Camera and 3D point places relative to a world coordinate frame $\mathcal{W}$. $\mathbf{R}$ is the orientation of the camera in 3D space and $\mathbf{c}^{\mathcal{W}}$ is the center of the camera.

$\mathbf{x} = (x1, x2, x3)^T$ the true image coordinates have to be calculated. For $x3 \neq 0$, $x = x1/x3$ and $y = x2/x3$ can be calculated.

2.1.3 Lens distortions

Until now, we have assumed that the mapping of the world unto the image plane is a linear function. Recall

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} f_x X + Z p_x \\ f_y Y + Z p_y \\ Z \end{pmatrix}$$

is a linear function with respect to $(X, Y, Z, 1)^T$. It is easy to understand, that this function is not an accurate model of the imaging process. Lens distortion must be taken into account (The following explanations are based on [19] and [20], where additional information can be found.). Figure 2.3 shows the effect of such a lens distortion. This sort of

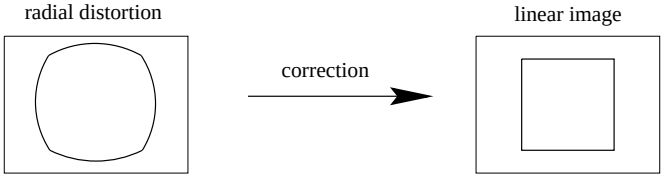


Figure 2.3: Effect of lens distortion. In this case a radial distortion is applied to a square (from [19], p190).

distortion takes place during the projection of the world onto the image plane. To take this process into account, a correction function has to be defined. $\tilde{\mathbf{x}}$ denotes the ideal image point (non-distorted) of the world point $\mathbf{X}$. To simplify it is supposed that the camera center is in the origin of the world coordinate system and its rotation matrix is the 3×3 identity matrix. Furthermore it is assumed, that the principal point $\mathbf{p} = \mathbf{0}$ and the focal length $f = 1$. The ideal undistorted point $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, 1)^T$ can be calculated, using the equation 2.3.

$$(\tilde{x}, \tilde{y}, 1)^T = [\mathbf{I} \mid \mathbf{0}]\mathbf{X} \tag{2.7}$$

The actual projected and distorted point $\mathbf{x}_d = (x_d, y_d)$ is related to the ideal undistorted point $(\tilde{x}, \tilde{y})^T$ by a radial and tangential displacement, modelled by the following equation (from [20]) :

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = (1 + \kappa_1 \cdot \tilde{r}^2 + \kappa_2 \cdot \tilde{r}^4 + \kappa_5 \cdot \tilde{r}^6) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} + \begin{pmatrix} x_t \\ y_t \end{pmatrix} \tag{2.8}$$

with $\tilde{r} = \sqrt{\tilde{x}^2 + \tilde{y}^2}$ being the distance from the center to the ideal undistorted point, $(1 + \kappa_1 \cdot \tilde{r}^2 + \kappa_2 \cdot \tilde{r}^4 + \kappa_5 \cdot \tilde{r}^6)$ being the radial distortion factor and $(x_t, y_t)^T$ being the tangential distortion vector, defined as follows (from [20]):

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} 2\kappa_3\tilde{x}\tilde{y} + \kappa_4(\tilde{r}^2 + 2\tilde{x}^2) \\ \kappa_3(\tilde{r}^2 + 2\tilde{y}^2) + 2\kappa_4\tilde{x}\tilde{y} \end{pmatrix} \tag{2.9}$$

With the help of the *radial distortion coefficients* $(\kappa_1, \kappa_2, \kappa_5)$ and the *tangential distortion coefficients* (κ_3, κ_4)it is now possible to correct the radial and tangential image distortion. These coefficients are also part of the internal camera calibration.

In table 2.1 and table 2.2 the values of the individual coefficients are given. The radial distortion coefficient κ_5 is not listed, because its value is too low and does not affect the calibration.

2.1.4 Results of the camera calibration

To solve the monocular grasping task, the camera of Nao has to be calibrated. For the calibration task the *Camera Calibration Toolbox for Matlab* [20] is used. To perform the calibration task, a set of images of a planar calibration pattern has to be taken. With the help of this set, the calibration toolbox calculates in a first step the linear intrinsic parameters (f_x, f_y, p_x, p_y). In a second step the distortion coefficients, radial as well as tangential, are estimated. The results are the internal parameters and the external parameters during the calibration process (orientation and translation of the camera). The result of the cali-

bration of Nao's bottom camera can be found in table 2.1. Figure 2.4 illustrates the impact of the distortion. Because the whole image processing on Nao is done on images with a resolution of 320×240, the images, that are used for the camera calibration, have the same resolution.

Parameter	Value	Uncertainty
f_x	379.74162	± 1.02104
f_y	380.32080	± 1.02530
p_x	151.62371	± 1.49148
p_y	114.28645	± 1.33429
radial distortion (κ_1, κ_2)	$[0.31431, -1.10508]$	$\pm[0.02009, 0.13029]$
tangential distortion (κ_3, κ_4)	$[-0.00016, -0.00092]$	$\pm[0.00179, 0.00204]$

Table 2.1: Result of the calibration of Nao's bottom camera.

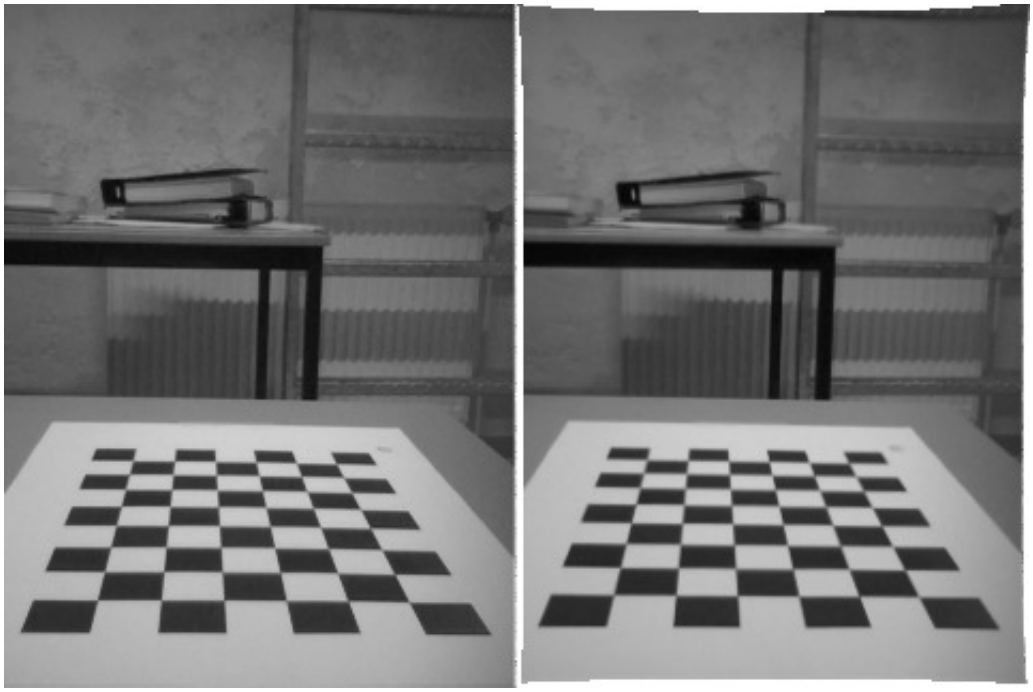


Figure 2.4: Distorted (left) and undistorted (right) image of Nao's bottom cam. The intrinsic, or internal, calibration parameters can be found in table 2.1.

To complete the explanation of camera calibration, the used images are presented in figure 2.5 and the camera position in space during the calibration process can be seen in figure 2.6. The results of Nao's top camera calibration can be found in table 2.2.

2.1.5 Calculation of an angle between two rays

With the help of a calibrated camera, the angle between two rays can be estimated [19]. Let us define $\mathbf{x}_1$ and $\mathbf{x}_2$ as two image points in homogeneous form. The corresponding

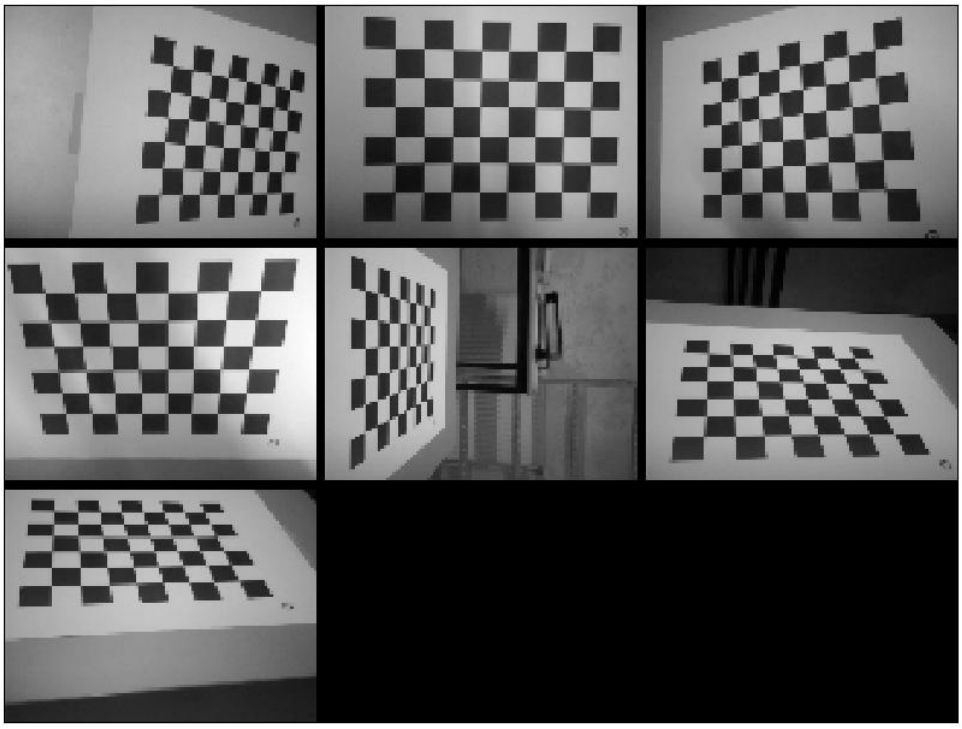


Figure 2.5: The seven images, used for the calibration.

Parameter	Value	Uncertainty
f_x	376.23709	± 0.78425
f_y	376.69494	± 0.80820
p_x	164.97117	± 1.59096
p_y	116.48445	± 1.37492
radial distortion (κ_1, κ_2)	$[0.28675, -1.02994]$	$\pm[0.01777, 0.11198]$
tangential distortion (κ_3, κ_4)	$[0.00413, -0.00097]$	$\pm[0.00177, 0.00217]$

Table 2.2: Result of the calibration of Nao's top camera.

rays are calculated according to $\mathbf{d}_1 = \mathbf{K}^{-1}\mathbf{x}_1$ and $\mathbf{d}_2 = \mathbf{K}^{-1}\mathbf{x}_2$. Here, $\mathbf{K}$ is the camera matrix (see equation 2.4). The angle between $\mathbf{d}_1$ and $\mathbf{d}_2$ is calculated according to [19] by the equation:

$$\cos\Theta = \frac{\mathbf{d}_1^T\mathbf{d}_2}{\sqrt{\mathbf{d}_1^T\mathbf{d}_1}\sqrt{\mathbf{d}_2^T\mathbf{d}_2}} \quad . \tag{2.10}$$

See figure 2.7 for a better visualization of the variables used in equation 2.10.

2.2 Iterative Estimation Methods

For this work, iterative estimation was used to enhance the measurement of the length of Nao's right thumb. This was done, because the position of Nao's right thumb fingertip is essential for the control of Naos's arm motions. Figure 2.8 illustrates the desired vector

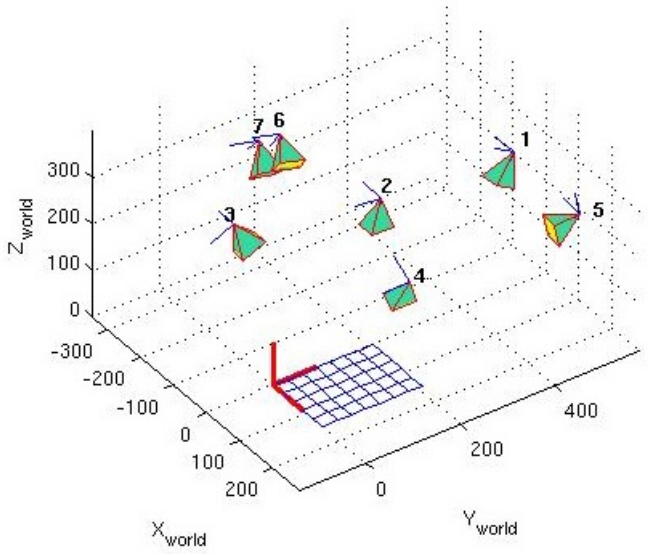


Figure 2.6: Visualisation of the external parameters during the calibration process.

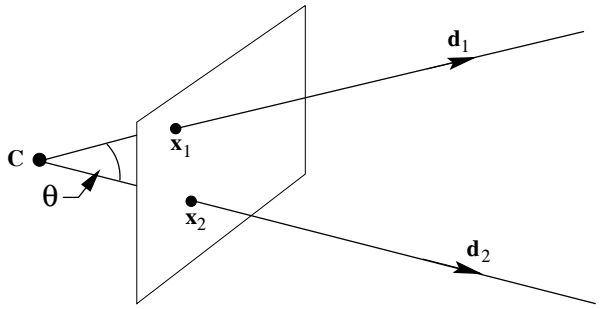


Figure 2.7: This figure (from [19], page 208) shows two image points ($\mathbf{x}_1$ and $\mathbf{x}_2$) and their corresponding rays ($\mathbf{d}_1$ and $\mathbf{d}_2$). The angle between these two rays can be calculated according to equation 2.10.

$\mathbf{D}_i^{\mathcal{H}}$. Based on image measurements the length is estimated. The mathematical essentials (Gauss- Newton) can be found in [19], p597. The next section presents how the length is estimated, using image measurements.

2.2.1 The application of iterative estimation in the grasping task

In the grasping approach, the iterative estimation method is used to estimate the vector from Nao's right arm reference frame $\mathcal{H}$ to the finger tip of its thumb, $\mathbf{D}_i^{\mathcal{H}}$ (in figure 2.8, this vector is marked as red arrow). This vector is unknown as well as it is impossible to measure the exact position of the origin of the reference frame. The idea is, to take a set of $N \geq 2$ images and calculate the desired vector. First, I set up a set of equations. For a better understanding of the particular vectors or symbols, figure 2.9 can be consulted. Let us start with the development of the measurement vector $\mathbf{X}$. In this approach, we

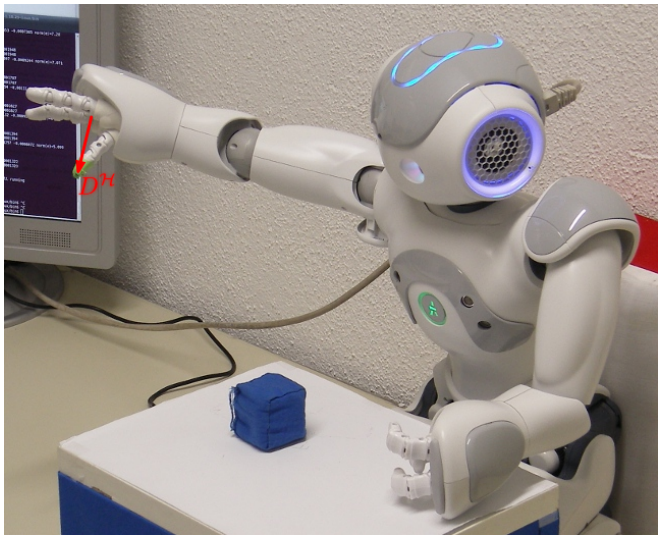


Figure 2.8: Illustration of the desired vector $\mathbf{D}_i^{\mathcal{H}}$ (red arrow). This vector plays an important role for the control of Nao's arm motion.

measure the image coordinate points $\mathbf{d}_i$ where $i = 1...N$. This is the measurement of the fingertip of the thumb in 3D space, namely $\mathbf{D}_i^{\mathcal{C}}$, in which $\mathcal{C}$ is the coordinate frame where the camera is placed. According to equation 2.6 it is possible to map the 3D point $\mathbf{D}_i^{\mathcal{C}}$ to our measurement $\mathbf{d}_i$.

$$\mathbf{d}_i = \mathbf{P}\mathbf{D}_i^{\mathcal{C}} \tag{2.11}$$

Because the 3D point is relative to the coordinate frame of the camera, the camera matrix can be expressed as

$$\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}] = \mathbf{K}[\mathbf{I} \mid \mathbf{0}] = [\mathbf{K} \mid \mathbf{0}] \tag{2.12}$$

Here $\mathbf{I}$ is the 3×3 identity matrix and $\mathbf{0}$ is a 3 null vector. I defines $\mathcal{H}$ as the hand coordinate frame and $\mathbf{D}_i^{\mathcal{H}}$ is the finger tip of the thumb related to this frame. It is to mention, that $\mathbf{D}_i^{\mathcal{H}} = \mathbf{D}_{i+1}^{\mathcal{H}} = \mathbf{D}^{\mathcal{H}}$ is constant in every measurement, because the distance between the arm coordinate frame and the finger tip of the thumb is constant. The mapping between $\mathbf{D}^{\mathcal{H}}$ and $\mathbf{D}_i^{\mathcal{C}}$ follows the equation

$$\mathbf{D}_i^{\mathcal{C}} = \mathbf{T}_{\mathcal{C},i}^{\mathcal{H}}\mathbf{D}^{\mathcal{H}} \tag{2.13}$$

where $\mathbf{T}_{\mathcal{C},i}^{\mathcal{H}}$ is the transformation from $\mathcal{C}$ to $\mathcal{H}$. Rearranging equation 2.11 and using our knowledge of the camera matrix, explained in equation 2.12, we project the measured point back into 3D space and get

$$\mathbf{D}_i^{\mathcal{C}} = \begin{pmatrix} \mathbf{K}^{-1}(\delta_i \mathbf{d}_i) \\ 1 \end{pmatrix} \tag{2.14}$$

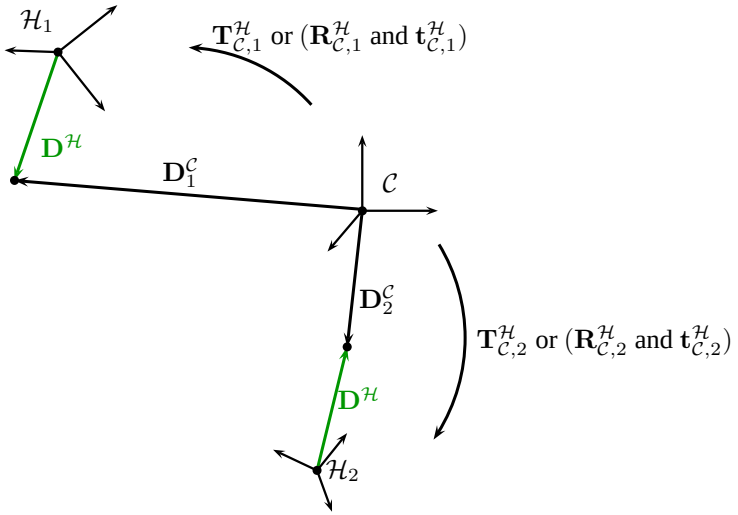


Figure 2.9: This figure illustrates the set-up for the optimization task for $N = 2$ measurements. The aim is to find the vector $\mathbf{D}^{\mathcal{H}}$. Note that this vector is constant over all measurements. Here $\mathcal{H}_1$ being the reference frame of the first hand position and $\mathcal{H}_2$ being the reference frame of the second hand position. $\mathbf{T}^{\mathcal{H}}_{\mathcal{C},1}$ and $\mathbf{T}^{\mathcal{H}}_{\mathcal{C},2}$ being the transformation matrices from the camera reference frame $\mathcal{C}$ to the first and second hand position. To get an imagination of the meaning of the vector $\mathbf{D}^{\mathcal{H}}$, have a look at figure 2.8.

where δ_i is the distance between the camera center and the 3D point. Inserting equation 2.14 into 2.13 results in

$$\begin{pmatrix} \mathbf{K}^{-1}(\delta_i \mathbf{d}_i) \\ 1 \end{pmatrix} = \mathbf{T}^{\mathcal{H}}_{\mathcal{C},i} \mathbf{D}^{\mathcal{H}} \tag{2.15}$$

Rewriting this to cancel the 1 on the left side of the equation and bringing the λ_i on the other side, we get

$$\mathbf{K}^{-1}\mathbf{d}_i = l_i \mathbf{R}^{\mathcal{H}}_{\mathcal{C},i} \widetilde{\mathbf{D}}^{\mathcal{H}} + l_i \mathbf{t}^{\mathcal{H}}_{\mathcal{C},i} \tag{2.16}$$

Here $\mathbf{R}^{\mathcal{H}}_{\mathcal{C},i}$, the rotation matrix and $\mathbf{t}^{\mathcal{H}}_{\mathcal{C},i}$, the translation vector, are the results of the decomposition of $\mathbf{T}^{\mathcal{H}}_{\mathcal{C},i}$. l_i is the substitution of $\frac{1}{\delta_i}$ and $\widetilde{\mathbf{D}}^{\mathcal{H}}$ is the inhomogeneous representation of $\mathbf{D}^{\mathcal{H}} = (\widetilde{\mathbf{D}}^{\mathcal{H}T}, 1)^T$. There is only one more step needed to solve the problem of the two

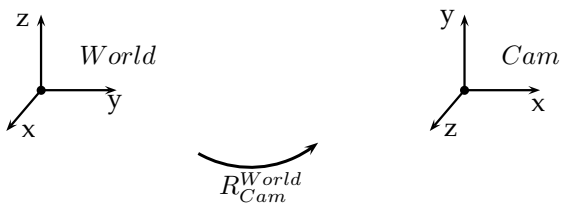


Figure 2.10: Illustration of the Nao coordinate frame rotation (left) and the usual computer vision notation (right). The rotation matrix $\mathbf{R}^{World}_{Cam}$ can be calculated easily. Note, that the origin of both coordinate frames is the same.

coordinate frames, explained in figure 2.10. The rotation matrix is

$$\mathbf{R}_{Cam}^{World} = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{pmatrix} \tag{2.17}$$

and so finally we can write our measurement equation as following:

$$\mathbf{R}_{Cam}^{World\,-1}\mathbf{K}^{-1}\mathbf{d}_i = l_i\mathbf{R}_{\mathcal{C},i}^{\mathcal{H}}\widetilde{\mathbf{D}}^{\mathcal{H}} + l_i\mathbf{t}_{\mathcal{C},i}^{\mathcal{H}} \ . \tag{2.18}$$

The desired parameter vector $\mathbf{P}$ is defined as

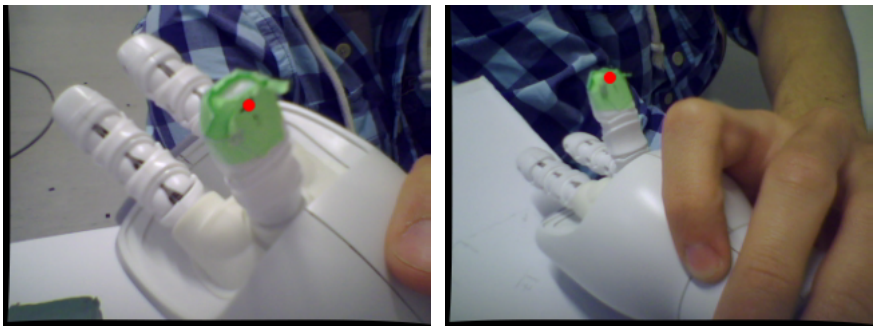
$$\mathbf{P} = \begin{pmatrix} \widetilde{\mathbf{D}}^{\mathcal{H}} \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix} \ . \tag{2.19}$$

All other vectors and matrices are known, and $\mathbf{d}_i$ is measured. We can now write the measurement equation depending on the numbers of N measurements

$$\begin{bmatrix} \mathbf{R}_{Cam}^{World\,-1}\mathbf{K}^{-1}\mathbf{d}_1 \\ \mathbf{R}_{Cam}^{World\,-1}\mathbf{K}^{-1}\mathbf{d}_2 \\ \vdots \\ \mathbf{R}_{Cam}^{World\,-1}\mathbf{K}^{-1}\mathbf{d}_N \end{bmatrix} = \begin{bmatrix} l_1\mathbf{R}_{\mathcal{C},1}^{\mathcal{H}}\widetilde{\mathbf{D}}^{\mathcal{H}} + l_1\mathbf{t}_{\mathcal{C},1}^{\mathcal{H}} \\ l_2\mathbf{R}_{\mathcal{C},2}^{\mathcal{H}}\widetilde{\mathbf{D}}^{\mathcal{H}} + l_2\mathbf{t}_{\mathcal{C},2}^{\mathcal{H}} \\ \vdots \\ l_N\mathbf{R}_{\mathcal{C},N}^{\mathcal{H}}\widetilde{\mathbf{D}}^{\mathcal{H}} + l_N\mathbf{t}_{\mathcal{C},N}^{\mathcal{H}} \end{bmatrix} \ . \tag{2.20}$$

2.2.2 Results of the iterative estimation

This section presents the results of the iterative estimation. The number of measurements is $N = 2$. That means that there are 2 images (see figure 2.11) used to estimate the length of Nao's thumb. The desired parameter vector is now $\mathbf{P} = (\widetilde{\mathbf{D}}^{\mathcal{H}\,T}, l_1, l_2)^T$. The number of experiments, in which I took 2 images was 5. After the experiment the *mean* and *standard deviation* are calculated. The results of the estimation of $\mathbf{D}^{\mathcal{H}}$ can be found in table 2.3. Figure 2.12 contains a sequence of images, that project the vector $\mathbf{D}^{\mathcal{H}}$ into the image plane (displayed by the black dot in every image).



(a) image 1 (b) image 2

Figure 2.11: Images used to calculate the desired vector $\mathbf{D}^{\mathcal{H}}$. The measured image coordinate points are marked in each image with a red dot. With this measurement and the transformation matrices obtained by Nao's framework, the desired parameter vector $\mathbf{P}$ is calculated, where $N = 2$.

2.3 Visual Servoing

In this section the concept of visual servoing, or often called *Visual Servo Control*, is introduced. A good tutorial to this topic is [16] and [17]. Here, I will use the notation from this tutorial to outline the concept. The aim of vision-based control is to minimize the error function

$$\mathbf{e}(t) = \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^{\star} \tag{2.21}$$

This error function is quite general. First this general formulation is explained and afterwards it is shown, how this error formulation is used to control Nao's hand to a desired position in space. In this formulation, $\mathbf{m}(t)$ denotes image measurements (e.g. coordinates of interest points) and the vector $\mathbf{a}$ contains parameters that represent additional knowledge about the system. The vectors $\mathbf{m}(t)$ and $\mathbf{a}$ are used to formulate the *visual feature* vector $\mathbf{s}(t) \in \mathbb{R}^k$. The *desired feature* vector is denoted as $\mathbf{s}^{\star}$. We assume, that $\mathbf{s}^{\star}$ is constant over time (the calculation of $\mathbf{s}^{\star}$ is explained in section 5.5). To accomplish the control task we develop a velocity controller. Denote the spatial velocity of Nao's hand by $\mathbf{v}_h(t) = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)^T$ where v_x, v_y, v_z are the velocities of the hand in x-, y- and z-direction, and $\omega_x, \omega_y, \omega_z$ are the angular velocities. The relationship between $\dot{\mathbf{s}}(t)$ and $\mathbf{v}_h(t)$ is given by

$$\dot{\mathbf{s}}(t) = \mathbf{L}\mathbf{v}_h(t) \tag{2.22}$$

with $\mathbf{L} \in \mathbb{R}^{(k \times 6)}$ being the *interaction matrix* or also called *feature Jacobian* or *Image Jacobian Matrix* [21]. Equation 2.21 is derivated in time and pasted into equation 2.22. Thus we receive

$$\dot{\mathbf{e}}(t) = \mathbf{L}\mathbf{v}_h(t) \tag{2.23}$$

experiment	$d_{1,x}$	$d_{1,y}$	$d_{2,x}$	$d_{2,y}$	$D_x^{\mathcal{H}}$	$D_y^{\mathcal{H}}$	$D_z^{\mathcal{H}}$
1	182	74	123	54	0.0029	-0.0041	-0.0424
2	130	89	134	123	0.0024	-0.0031	-0.0421
3	71	156	98	125	0.0037	-0.0026	-0.0409
4	66	90	152	73	0.0028	-0.0024	-0.0428
5	235	73	132	96	0.0002	0.004	-0.0502
$mean_{1-5}$					0.0024	-0.00164	-0.04368
std_{1-5}					0.001317	0.003221	0.003713
$mean_{1-4}$					0.00295	-0.00305	-0.04205
std_{1-4}					0.00054	0.000759	0.00082

Table 2.3: This table presents the results of the iterative estimation. Experiment number 5 seems to be an outlier (especially if we look at $D_y^{\mathcal{H}}$), because the arm was too far away from the camera. $mean_{1-5}$ and std_{1-5} are the *mean* and *standard deviation* over all experiments, and $mean_{1-4}$ and std_{1-4} are the *mean* and *standard deviation* from experiment number 1 to experiment number 4. The measurements $\mathbf{d}_1 = (d_{1,x}, d_{1,y})^T$ and $\mathbf{d}_2 = (d_{2,x}, d_{2,y})^T$ are measured in $[Pixel]$ and the result vector $D_y^{\mathcal{H}} = (D_x^{\mathcal{H}}, D_y^{\mathcal{H}}, D_z^{\mathcal{H}})^T$ has the unit $[m]$.

The aim is now, to ensure that the error $\mathbf{e}(t)$ decreases over time. To achieve this, we can use an exponentially decreasing error function

$$\dot{\mathbf{e}}(t) = -\lambda \mathbf{e}(t) \tag{2.24}$$

and solve equation 2.23 according to this function. This is done by pasting equation 2.24 in equation 2.23 and solving the resulting equation for $\mathbf{v}_h(t)$. As result we obtain

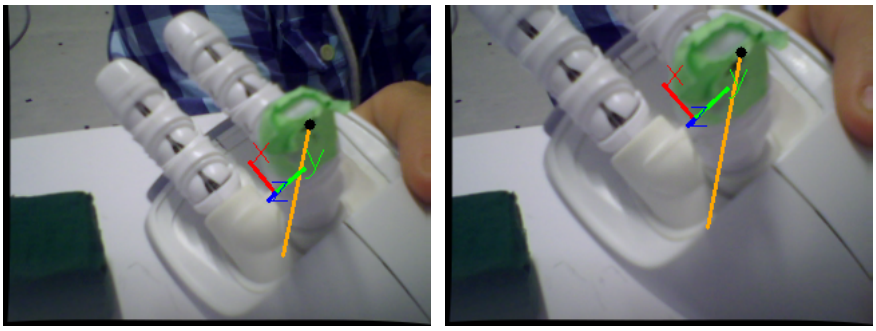
$$\mathbf{v}_h(t) = -\lambda \mathbf{L}^+ \mathbf{e}(t), \tag{2.25}$$

where $\mathbf{L}^+$ is the *Moore-Penrose pseudo-inverse* of $\mathbf{L}$. This equation gives us now the possibility to control Nao's hand according to the measured error. Here, λ is the gain of the controller and decides how fast the controller should respond to the measured error.

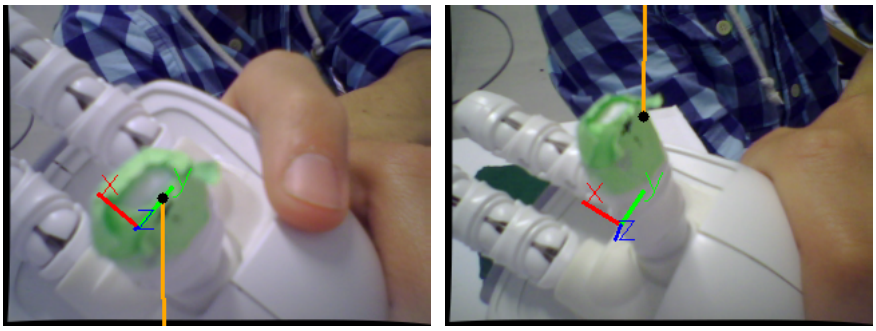
In section 3.4, the practical implementation and adaptation of this concept can be found.

2.3.1 Image based visual servo

In the visual servo domain there is a distinction between *image based visual servo* (IBVS) and *position based visual servo* (PBVS). The first approach uses the image coordinates to define the *visual feature* $\mathbf{s}(t)$. In the second approach (PBVS), the pose of the camera or hand, relative to a reference frame, is used to define $\mathbf{s}(t)$. In this work, the focus lies on the *image based visual servo* (IBVS). Thus, the *interaction matrix* maps image coordinates



(a) result image 1 (b) result image 2



(c) result image 3 (d) result image 4

Figure 2.12: A sequence of images taken during the arm movement. The black dots represent the projection of $\mathbf{D}^{\mathcal{H}}$ onto the image plane. In this sequence the hand coordinate frames are also displayed.

velocities to hand velocities according to equation 2.22.

2.3.2 The interaction matrix

In practice it is not possible to know $\mathbf{L}$ or $\mathbf{L}^{+}$ exactly, because the relations, that $\mathbf{L}$ models, can be very complex. Deguchi [22], for example, uses the main eigenvalues of the principal component analysis for his image features.

A common way to estimate the interaction matrix is to make some exploration movements and observe the change of the image feature with respect to the change of the hand position. Let us denote $\Delta\mathbf{s}$ as the variation of the image feature during the camera motion $\Delta\mathbf{v}_h$ and $\widehat{\mathbf{L}}$ as the approximation of $\mathbf{L}$. Using this notation we can rewrite equation 2.22 as

$$\widehat{\mathbf{L}}\Delta\mathbf{v}_h = \Delta\mathbf{s}. \tag{2.26}$$

We have k equations for $k \times 6$ unknowns. Performing $N \geq 6$ independent hand movements $(\Delta\mathbf{v}_{h,1}, \Delta\mathbf{v}_{h,2}, \cdots, \Delta\mathbf{v}_{h,N})$ and observing the according feature changes $(\Delta\mathbf{s}_1, \Delta\mathbf{s}_2, \cdots, \Delta\mathbf{s}_N)$, it is possible to solve $\widehat{\mathbf{L}}$ by

$$\widehat{\mathbf{L}}\mathbf{A} = \mathbf{B} \tag{2.27}$$

with

$$\mathbf{A} = (\Delta\mathbf{v}_{h,1}, \Delta\mathbf{v}_{h,2}, \cdots, \Delta\mathbf{v}_{h,N}) \in \mathbb{R}^{6\times N}$$

and

$$\mathbf{B} = (\Delta\mathbf{s}_1, \Delta\mathbf{s}_2, \cdots, \Delta\mathbf{s}_N) \in \mathbb{R}^{k\times N}$$

In practice (see [16]) , there are better results if the pseudo inverse of $\widehat{\mathbf{L}}$ is estimated directly according to

$$\widehat{\mathbf{L}^+} = \mathbf{A}\mathbf{B}^+ \ . \tag{2.28}$$

Since in equation 2.25 the pseudo inverse is used to calculate the hand velocity, it is a better choice of computation.
There also exist approaches, where the *interaction matrix* is estimated on-line during the control. Interested readers should have a look at [17].

In this work, the calculation of the interaction matrix is done the following way: the *visual feature* $\mathbf{s}(t)$ has the dimension of 2 ($\mathbf{s}(t) \in \mathbb{R}^2$), because we just measure the image point coordinates. The dimension of Nao's hand velocity is also 2, because Nao's hand is moving in a plane and the orientation of Nao's hand is not controlled ($\mathbf{v}_h(t) \in \mathbb{R}^2$). It follows that $\Delta\mathbf{v}_{h,i} \in \mathbb{R}^2$ and $\Delta\mathbf{s}_i \in \mathbb{R}^2$. Therefore, there are only two independent arm movements necessary. The matrix $\mathbf{A}$ becomes now

$$\mathbf{A} = (\Delta\mathbf{v}_{h,1}, \Delta\mathbf{v}_{h,2}) \in \mathbb{R}^{2\times 2}$$

and as a consequence of that, the matrix $\mathbf{B}$ becomes

$$\mathbf{B} = (\Delta\mathbf{s}_1, \Delta\mathbf{s}_2) \in \mathbb{R}^{2\times 2} \ .$$

As a consequence, the *interaction matrix* becomes a 2×2 matrix and can now be calculated. Because the *interaction matrix* is now a square matrix, no *pseudo inverse* is needed, therefore equation 2.28 becomes

$$\widehat{\mathbf{L}}^{-1} = \mathbf{A}\mathbf{B}^{-1} \ . \tag{2.29}$$

3

Grasping with Nao

This chapter presents how grasping with Nao was enabled. The main part of this chapter is based on my seminar project [2], and the essential equations to understand this chapter were already presented in chapter 2.

The result of an experiment, presented at the end of this chapter, proves the functionality of the grasping concept.

3.1 Overview and main idea

The practical implementation of grasping on Nao is complicated by inaccuracies of the kinematic, limited processing power, monocular vision and the pincer-shaped hand of Nao. Because of inaccuracies of the positioning ability, a control approach has to be implemented to fulfill the grasping task. Another difficulty for the grasping is, that the image of the foam cube is partially or entirely occluded by Nao's hand during the phase, where Nao's hand approaches the foam cube. The work with Nao showed, that the position and orientation of Nao's thumb is essential for successful grasping. For that reason the position of the thumb is controlled in such a way, that the grasping can be accomplished. To do this, the thumb is projected onto a plane, from now on called, the *desired plane*. This projection is used as the input of a controller, that steers Nao's arm to the right location.

To study this concept, Nao was first placed next to a table and its task was, to grasp

a color coded foam cube, that lies on that table. Just one cube was on the top of the table and the color of the table surface was unsaturated. Figure 3.1 shows this setting. The advantage of this setting is, that Nao is in an upright posture and therefore its joints

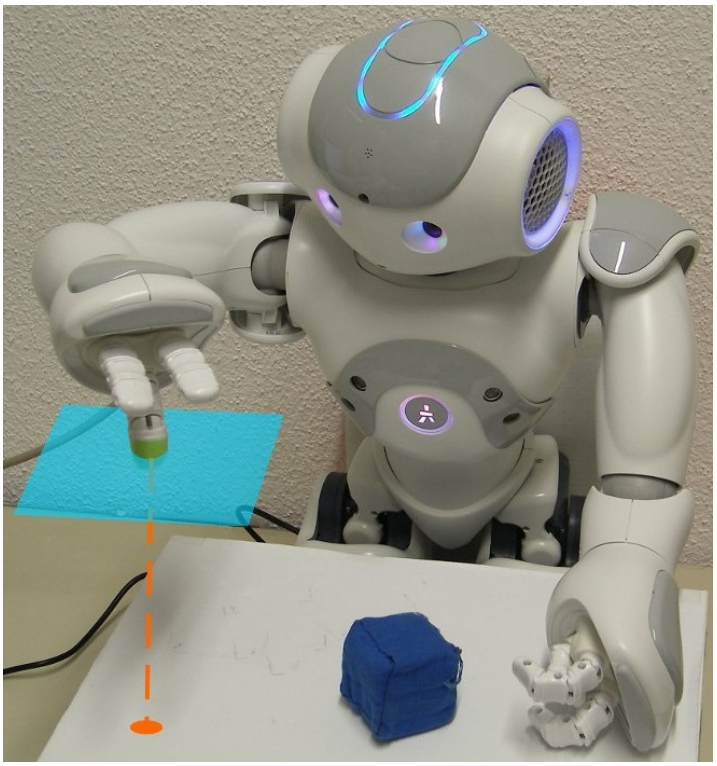


Figure 3.1: Sketch of the proposed solution for the case when Nao sits next to a table. This figure shows a cyan plane that visualizes, in which plane the positioning of Nao's arm is done. The virtual thumb (orange dot) is calculated as the projection of Nao's thumb (green) onto the *desired plane* (in this case the *desired plane* is the surface of the table). The spatial orientation of the *desired plane* is estimated relative to Nao's feet position.

are not getting hot so quickly. At the end of this chapter, the results are presented for this setting. The presented concept is very flexible and can be adapted easily to the case, where Nao has to pick up a foam cube from the floor. Major parts of the solution of the grasping task are (point 1 and 2 are illustrated in figure 3.1):

1. calculating the desired plane
2. projecting the thumb onto the desired plane
3. controlling the arm movement according to the projected thumb
4. repeating steps 1-3 until the object is grasped

3.2 Calculation of the desired plane

In order to solve the grasping task, the spatial orientation of a plane parallel to the floor plane (the *desired plane*) has to be estimated. This plane could be the floor plane itself, or the table surface. It is assumed, that the table surface has the same orientation as the floor plane, only an additional offset, the table height, is added. This section explains how this is done.

First, it is assumed, that the *desired plane* $\boldsymbol{\pi}^{\mathcal{T}}$ is parallel to the ground $\boldsymbol{\pi}^{\mathcal{T}}_{gr}$, where Nao stands. With the help of the framework, provided by Nao, the orientation and position of the ground plane can be estimated (it has the same orientation like Nao's leg). The spatial orientation of the *desired plane* can be computed by first setting the *desired plane* parallel to the ground and afterwards specifying an offset along the z-direction. In the case of picking up objects from the floor, the z-offset is 0.0 and in the case where Nao sits next to a table, the z-offset is the table height. Figure 3.2 illustrates this estimation. Mathematically the estimation of the *desired plane* can be expressed as follows: Define

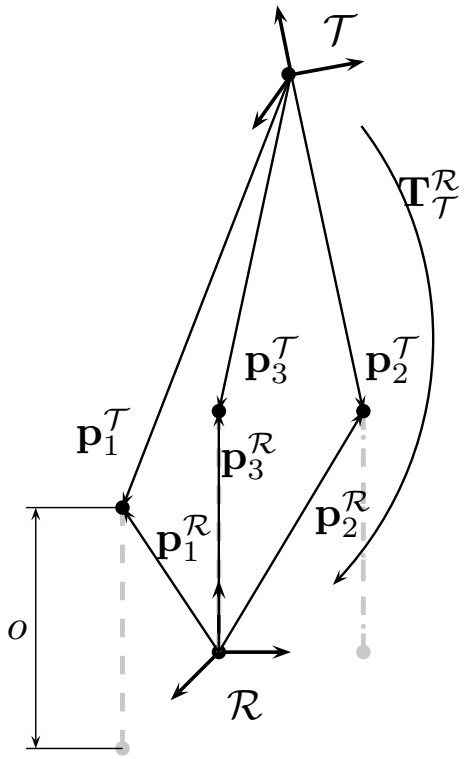


Figure 3.2: This figure illustrates the estimation of the *desired plane*. Here, o being the offset of the *desired plane* relative to the floor, $\mathbf{p}^{\mathcal{R}}_i$ with $i = 1 \ldots 3$ being three points relative to the leg reference frame $\mathcal{R}$ and $\mathbf{p}^{\mathcal{T}}_i$ with $i = 1 \ldots 3$ being three points relative to the torso reference frame $\mathcal{T}$. From $\mathbf{p}^{\mathcal{T}}_i$, the *desired plane* is calculated.

$\mathcal{R}$ as the reference frame of the right leg and $\mathcal{T}$ as the reference frame of Nao's torso. Then $\mathbf{T}^{\mathcal{R}}_{\mathcal{T}}$ is the 4×4 transformation matrix from $\mathcal{T}$ to $\mathcal{R}$. The goal is now to estimate the

desired plane relative to Nao's torso reference frame $\mathcal{T}$, denoted as $\boldsymbol{\pi}^{\mathcal{T}}$. For this purpose it is necessary to estimate three points in space, namely $\mathbf{p}_1^{\mathcal{T}}, \mathbf{p}_2^{\mathcal{T}}, \mathbf{p}_3^{\mathcal{T}}$. These three points can be used to estimate the *desired plane*. Because the *desired plane* is parallel to the ground, it is also parallel to the reference frame of Nao's right leg $\mathcal{R}$, when Nao stands on the ground. By defining three points in this reference frame ($\mathcal{R}$) and transforming them into the torso reference frame $\mathcal{T}$, the *desired plane* can be estimated. This plane is parallel to the floor and relative to the torso reference frame $\mathcal{T}$ (see figure 3.2). The estimation of the plane, relative to the reference frame of the right leg of Nao $\mathcal{R}$, is easy. The following three points are chosen relative to $\mathcal{R}$:

$$\mathbf{p}_1^{\mathcal{R}} = \begin{pmatrix} 0.1 \\ 0 \\ o \\ 1 \end{pmatrix}, \mathbf{p}_2^{\mathcal{R}} = \begin{pmatrix} 0 \\ 0.1 \\ o \\ 1 \end{pmatrix}, \mathbf{p}_3^{\mathcal{R}} = \begin{pmatrix} 0 \\ 0 \\ o \\ 1 \end{pmatrix} \tag{3.1}$$

with o being the offset between ground and *desired plane* (e.g. the table heigh). If Nao sits next to a table, this offset is given by the table height. The 0.1 in two of the three points was chosen to fit properly to Nao's scale. It has to be mentioned, that the transformation is done in the homogeneous framework. The calculation of the points relative to the torso reference frame $\mathcal{T}$ is straight forward, according to:

$$\left[\mathbf{p}_1^{\mathcal{T}}, \mathbf{p}_2^{\mathcal{T}}, \mathbf{p}_3^{\mathcal{T}}\right] = \mathbf{T}_{\mathcal{T}}^{\mathcal{R}} \left[\mathbf{p}_1^{\mathcal{R}}, \mathbf{p}_2^{\mathcal{R}}, \mathbf{p}_3^{\mathcal{R}}\right] \tag{3.2}$$

With the help of these three points ($\mathbf{p}_1^{\mathcal{T}}, \mathbf{p}_2^{\mathcal{T}}, \mathbf{p}_3^{\mathcal{T}}$), it is now possible to determine the *desired plane*. This calculation is based on the fact, that three points define a plane [19]. It follows, that all of these three points lie within the *desired plane* $\boldsymbol{\pi}^{\mathcal{T}}$ and therefore the equations $\mathbf{p}_i^{\mathcal{T}\,T}\boldsymbol{\pi}^{\mathcal{T}} = 0$ has to be satisfied for every point $i = 1 \ldots 3$. These three equations are now assembled together, to one homogeneous linear system:

$$\begin{pmatrix} \mathbf{p}_1^{\mathcal{T}\,T} \\ \mathbf{p}_2^{\mathcal{T}\,T} \\ \mathbf{p}_3^{\mathcal{T}\,T} \end{pmatrix} \boldsymbol{\pi}^{\mathcal{T}} = \mathbf{0} \tag{3.3}$$

This equation system can be solved for $\boldsymbol{\pi}^{\mathcal{T}}$ using for example the singular value decomposition (SVD) [19].

To verify the accuracy of the estimated *desired plane*, an experiment was set up. Six points in this plane, that is parallel to the ground, are defined. In the experiment, the arm was moved to each of these points and the offset between the ground plane $\boldsymbol{\pi}_{gr}^{\mathcal{T}}$ and the *desired plane* $\boldsymbol{\pi}^{\mathcal{T}}$ was measured. This procedure was repeated 15 times. The experiments

showed, that the deviation angle between the estimated *desired plane* $\boldsymbol{\pi}^{\mathcal{T}}$ and the ground $\boldsymbol{\pi}^{\mathcal{T}}_{gr}$ is $\angle(\boldsymbol{\pi}^{\mathcal{T}}, \boldsymbol{\pi}^{\mathcal{T}}_{gr}) = 4.4° \pm 1.9°$. The result of the measurements can be found in table 3.1. The measured angle is sufficiently accurate for the purpose of the grasping task. If the

	Measured heigh over the table [mm]					
Experiment	a	b	c	d	e	f
1	82	74	81	83	83	82
2	78	76	81	80	80	81
3	79	71	81	80	78	82
4	78	75	80	81	78	82
5	83	79	83	83	84	85
6	82	74	81	83	83	81
7	81	73	81	82	83	81
8	78	77	80	82	82	81
9	83	75	82	83	83	85
10	81	76	81	83	82	82
11	79	78	81	82	82	82
12	79	78	81	82	82	82
13	80	78	82	83	83	84
14	83	80	84	85	84	85
15	83	80	83	83	84	86

Table 3.1: Measurements of the Experiment. The letters 'a' to 'f' stand for the different arm positions.

worst case situation happens, that means an angle deviation of $6.3°$, the error in height, when Nao moves its arm over a distance of $15[cm]$, is about $1.65[cm]$. The work and experience with Nao show, that $15[cm]$ of Nao's arm motions is a proper value.

3.3 The virtual thumb concept

As a next step, the thumb is projected onto the *desired plane*. This leads to the *virtual thumb* concept (see figure 3.3 for details). The position of the finger tip of the right thumb relative to the torso reference frame $\mathcal{T}$ is denoted as $\mathbf{D}^{\mathcal{T}}$. The position of the thumb relative to the torso reference frame can be determined with the help of the transformation $\mathbf{T}^{\mathcal{H}}_{\mathcal{T}}$ from the torso reference frame $\mathcal{T}$ to the right hand reference frame $\mathcal{H}$ and the vector $\mathbf{D}^{\mathcal{H}}$, that represents the position of the thumb, relative to the right hand reference frame $\mathcal{H}$. The vector $\mathbf{D}^{\mathcal{H}}$ was already estimated (see subsection 2.2.1 and 2.2.2). Therefore it can be written

$$\mathbf{D}^{\mathcal{T}} = \mathbf{T}^{\mathcal{H}}_{\mathcal{T}}\mathbf{D}^{\mathcal{H}} = \begin{pmatrix} \widetilde{\mathbf{D}}^{\mathcal{T}} \\ 1 \end{pmatrix} \tag{3.4}$$

Where $\widetilde{\mathbf{D}}^{\mathcal{T}}$ is the location of $\mathbf{D}^{\mathcal{T}}$ in $\mathbb{R}^3$ after the normalization of $\mathbf{D}^{\mathcal{T}}$. The *desired plane* $\boldsymbol{\pi}^{\mathcal{T}}$ can also be represented in the following form:

$$\boldsymbol{\pi}^{\mathcal{T}} = \begin{pmatrix} n_x \\ n_y \\ n_z \\ \tilde{d} \end{pmatrix} = \begin{pmatrix} \mathbf{n}^{\mathcal{T}} \\ \tilde{d} \end{pmatrix} \tag{3.5}$$

Where $\mathbf{n}^{\mathcal{T}}$ is the normal vector of the plane relative to $\mathcal{T}$ and $\frac{\tilde{d}}{||\mathbf{n}||}$ is the *normal distance* between the plane and the origin of the torso reference frame $\mathcal{T}$. With $\widetilde{\mathbf{D}}^{\mathcal{T}}$ and $\mathbf{n}^{\mathcal{T}}$ it is

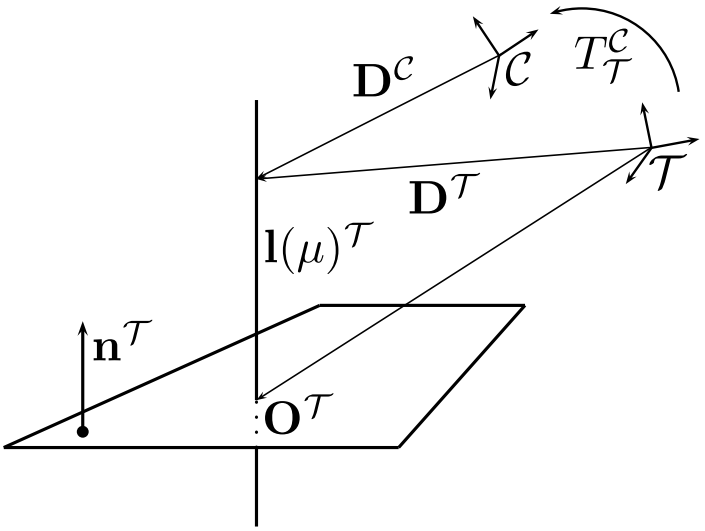


Figure 3.3: This figure illustrates the projection of the thumb onto a plane. Here, $\mathbf{n}^{\mathcal{T}}$ is the *normal vector* of the plane, $\mathbf{D}^{\mathcal{T}}$ is the vector that points from the torso reference frame $\mathcal{T}$ to the thumb of Nao's right arm and $\mathbf{O}^{\mathcal{T}}$ is the projection of Nao's thumb onto the plane (*virtual thumb*). The line $\mathbf{l}(\mu)^{\mathcal{T}}$ is the parametrized line between $\mathbf{D}^{\mathcal{T}}$ and $\mathbf{O}^{\mathcal{T}}$ and is parallel to the plane normal vector $\mathbf{n}^{\mathcal{T}}$.

now possible to write down a parametrized line in $\mathbb{R}^3$, namely $\mathbf{l}(\mu)^{\mathcal{T}}$.

$$\mathbf{l}(\mu)^{\mathcal{T}} = \widetilde{\mathbf{D}}^{\mathcal{T}} + \mu \mathbf{n}^{\mathcal{T}} \tag{3.6}$$

For each $\mu \in \mathbb{R}$ there exists a point, that lies on the line $\mathbf{l}(\mu)^{\mathcal{T}}$. The goal is now, to find the intersection point between the line $\mathbf{l}(\mu)^{\mathcal{T}}$ and the plane $\boldsymbol{\pi}^{\mathcal{T}}$. The intersection point satisfies the following equation:

$$\mathbf{l}(\mu)^{\mathcal{T}^T} \boldsymbol{\pi}^{\mathcal{T}} = 0 \tag{3.7}$$

Solving this equation with respect to μ leads to

$$\mu = \frac{-(\pi_1 d_1 + \pi_2 d_2 + \pi_3 d_3 + \pi_4)}{\pi_1 d_1 + \pi_2 d_2 + \pi_3 d_3} \ , \tag{3.8}$$

where $\widetilde{\mathbf{D}}^{\mathcal{T}} = (d_1, d_2, d_3, d_4)^T$ and $\boldsymbol{\pi}^{\mathcal{T}} = (\pi_1, \pi_2, \pi_3, \pi_4)^T$. This intersection point is the *virtual thumb* relative to the torso reference frame $\mathcal{T}$, mathematically expressed as

$$\mathbf{O}^{\mathcal{T}} = \mathbf{l}(\mu)^{\mathcal{T}}\Big|_{\mu=\frac{-(\pi_1 d_1 + \pi_2 d_2 + \pi_3 d_3 + \pi_4)}{\pi_1 d_1 + \pi_2 d_2 + \pi_3 d_3}} \quad . \tag{3.9}$$

3.3.1 Projecting the virtual thumb onto the image plane

Given the *virtual thumb* relative to the torso reference frame and the transformation from that frame to the camera frame, the *virtual thumb* can be expressed relative to the camera reference frame $\mathcal{C}$ as

$$\mathbf{O}^{\mathcal{C}} = \mathbf{T}_{\mathcal{T}}^{\mathcal{C}\,-1}\mathbf{O}^{\mathcal{T}} \quad . \tag{3.10}$$

With the help of the mathematical methods, presented in section 2.1, the *virtual thumb* is projected onto the image plane. See figure 3.4 for the result of the calculation. The orange dot in this figure is the projection of the *virtual thumb* into the image plane, the black dot is the projection of Nao's right hand thumb into the image plane and the orange line is the projection of the normal vector $\mathbf{n}^{\mathcal{T}}$ of the *desired plane*. This calculation is done in every control step.

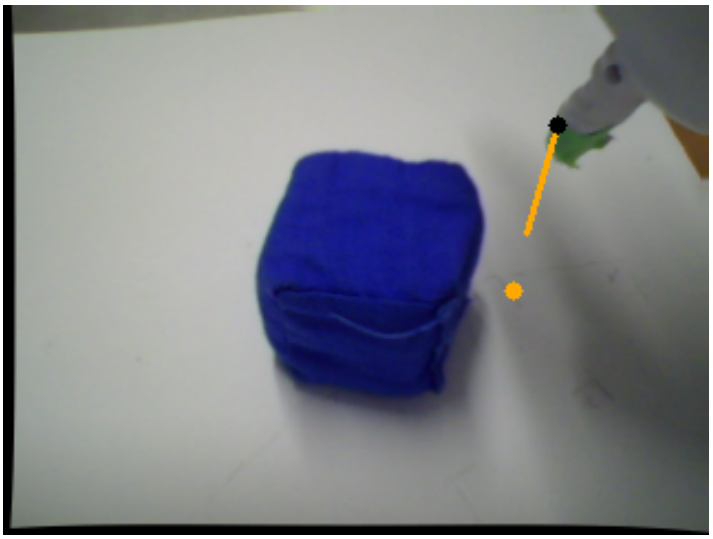


Figure 3.4: Image of Nao's bottom camera augmented by the *virtual thumb* (orange dot) , the normal vector of the *desired plane* (orange line) and the fingertip of Nao's right hand thumb (black dot).

3.3.2 Evaluation of the accuracy of the *virtual thumb*

The following experiment evaluates the accuracy of the thumb projection. A scaled paper was fixed on the table and Nao was placed next to it. A small perpendicular was fixed

on Nao's right thumb. Nao's hand was placed in several positions above the scaled paper. Once the hand was positioned, two points were marked on the paper, first the position of the perpendicular and second the position of the *virtual thumb*. Figure 3.5 illustrates the result of the experiment. It can be seen, that there is a systematic offset between the

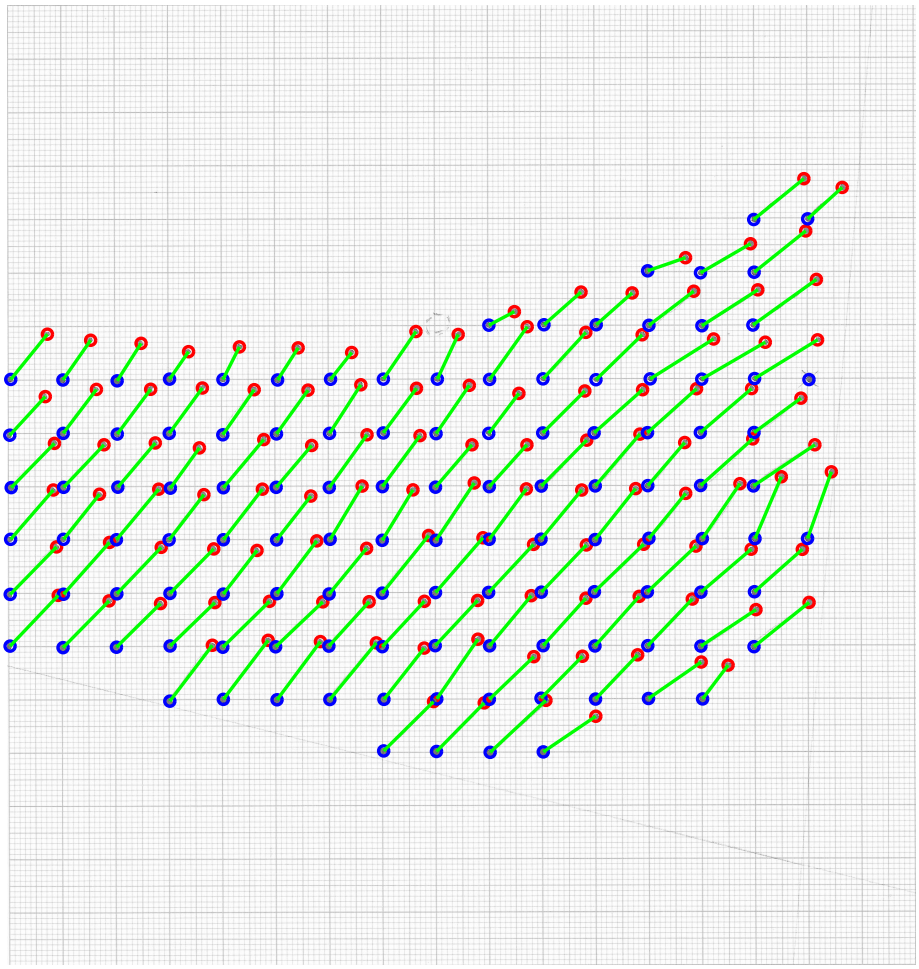


Figure 3.5: Result of the experiment to determine the accuracy of the *virtual thumb*. The blue dots represent the position of the perpendicular, that means the true position. The red dots represent the position of the *virtual thumb*. It can be seen, that there is a systematic offset between the true projection of the thumb (blue) and the calculated projection (red).

projection, obtained with the help of a perpendicular, and the calculated projection, the *virtual thumb*.

Figure 3.6 shows the error of the *virtual thumb* calculation. It should be noted, that the calculation of the *virtual thumb* depends on the accuracy of each joint angle sensor, because the used transformation matrices and arm positions are calculated with the help of the sensor readings of the joints. For example, the right hand contains six joint sensors.

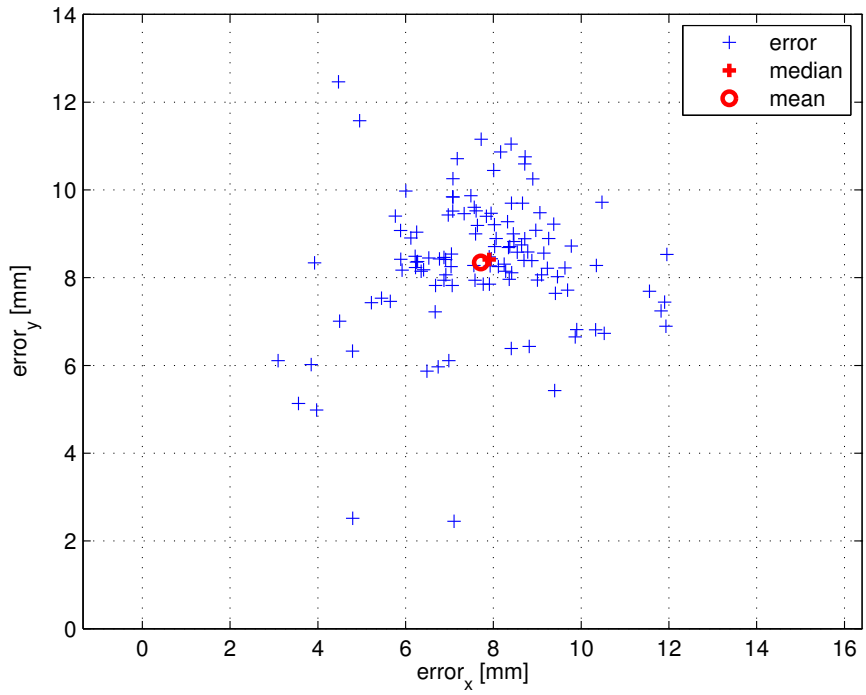


Figure 3.6: Illustration of the error obtained by the projection of the virtual thumb.

To estimate the position of the thumb relative to the torso reference frame (this is the vector $\mathbf{D}^{\mathcal{T}}$ in Figure 3.3) all of these six sensors are used, and therefore the error of each of them accumulates.

3.4 Visual servo control of the *virtual thumb*

With the help of the *virtual thumb* and the explanations in section 2.3 it is possible to formulate a control law. The goal is to control the arm movement of Nao's right arm in such a way, that the grasping can be accomplished. This section presents the control in detail. The equations of section 2.3 are adapted to fit to the grasping problem on Nao. The whole control of the arm movement is done in a plane that is parallel to the *desired plane* and is called the *motion plane*. Because of the motion that happens in that plane, two coordinates in $\mathbb{R}^3$ are free and one coordinate is constrained by the plane. In practice, that means, that only two of the three coordinates have to be controlled, the third one is calculated through the plane equation. Another difference to section 2.3 is, that only the position of the finger tip of Nao's right thumb has to be controlled, whereas the orientation is not controlled. So the controlling task is simplified to control the position of a single point in $\mathbb{R}^3$, more precisely to control the position of a point in the *motion plane* that lies in $\mathbb{R}^3$.

As a conclusion it can be said that the spatial velocity of the thumb in the plane is

$$\mathbf{v}_h(t) = \begin{pmatrix} v_x \\ v_y \end{pmatrix} \ . \tag{3.11}$$

The *visual feature* is the image point coordinate of the *virtual thumb*:

$$\mathbf{s}_h(t) = \begin{pmatrix} s_x \\ s_y \end{pmatrix} \ . \tag{3.12}$$

Therefore, the *interaction matrix* $\widehat{\mathbf{L}}$ is a 2×2 matrix:

$$\dot{\mathbf{s}}_h(t) = \widehat{\mathbf{L}}\mathbf{v}_h(t) \ . \tag{3.13}$$

The control error is

$$\mathbf{e}(t) = \mathbf{s}_h(t) - \mathbf{s}^\star, \tag{3.14}$$

where $\mathbf{s}^\star$ is the *desired feature* vector. The calculation of this vector is explained in detail in section 5.5. For the ongoing explanation we just assure that $\mathbf{s}^\star$ exists and can be calculated. According to equation 2.25 the control law is

$$\mathbf{v}_h(t) = -\lambda \widehat{\mathbf{L}}^{-1}\mathbf{e}(t), \tag{3.15}$$

in which the inverse of the *interaction matrix* ($\widehat{\mathbf{L}}^{-1}$) is pre-calculated as it is explained in subsection 2.3.2 (equation 2.29). The *gain factor* λ was set to be $\lambda = 0.3$.

Because the motion is done in $\mathbb{R}^3$ and the controller output is in $\mathbb{R}^2$, the third component has to be determined according to the plane equation. We suppose that $\boldsymbol{\pi}_m^{\mathcal{T}}$ is the *motion plane* that is parallel to the *desired plane* $\boldsymbol{\pi}^{\mathcal{T}}$. The missing coordinate can be computed as following

$$p_{z,k} = \frac{-(\pi_{m,1} * p_{x,k} + \pi_{m,2} * p_{y,k} + \pi_{m,4})}{\pi_{m,3}} \tag{3.16}$$

Where $(p_{x,k}, p_{y,k})^T$ is the position of the thumb in the *motion plane* at time k and is calculated with the help of

$$\begin{pmatrix} p_{x,k} \\ p_{y,k} \end{pmatrix} = \begin{pmatrix} p_{x,k-1} \\ p_{y,k-1} \end{pmatrix} + \begin{pmatrix} v_x \\ v_y \end{pmatrix} \Delta t \tag{3.17}$$

Here $\Delta t = 0.5$ is the sampling time and $(v_x, v_y)^T$ are the velocities given by the control law according to equation 3.15.

3.5 A comparison between grasping from a table and picking-up from the floor

Previous work [2] shows, that the area, where Nao could pick up objects is very limited. In figure 3.7 the result of the grasping experiment from the table is shown. The areas, marked with green squares are positions of the foam cube, where Nao was able to grasp it. In the yellow areas Nao was able to grasp the cube, but sometimes it failed to do so. The red areas mark positions, where Nao was unable to grasp the cube. The reason for the failure cases is the limited motion space of Nao's arm. Nao is unable to position its arm in such a way, that it could grasp the foam cubes, placed in the red area. In the case

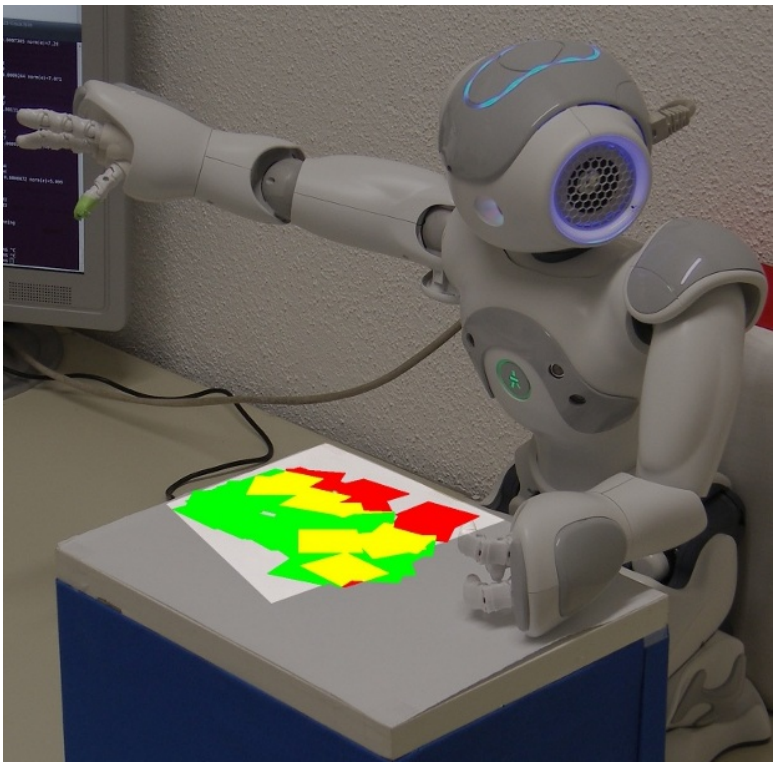


Figure 3.7: This figure shows the result of our grasping experiment. Green areas show successful grasps, red ones failure cases. Most of the failures occurred in the red area close to Nao's body and are due to the restricted freedom of motion of Nao's hand. Yellow indicates occasional failure. The highlighted area marks the field of view of Nao's bottom camera.

when Nao has to pick up foam cubes from the floor, the motion space of Nao's arm is much more restricted. For this reason, Nao has to be positioned very exactly next to the cube, before it bends down to grasp it. The next chapters present how Nao walks to an object (chapter 4) and how it is positioned in such a way, that it is able to pick up foam cubes (chapter 5). When all parts of the solution work together, Nao is able to pick up foam cubes in 88% of the cases. Compared to the pick-up task from the table, where Nao

could pick-up foam cubes in 74% of the cases, the solution presented in this work is very robust.

More detailed information about the results can be found in [2], when Nao picks up foam cubes from a table, and in section 5.7, when Nao picks up foam cubes from the floor.

4

Walking towards an object

This chapter presents the main ideas and concepts that enable Nao to walk towards an object in its environment.

It should be noted at the beginning of this chapter, that some of the displayed images have wrong colors. The reason for that is the used color space. All of the computation of color images is done in the HSY color space, but to display these images, the OpenCV computer vision library interprets them as RGB images and as a consequence, the images are wrongly coloured.

4.1 Overview and main ideas

Nao's framework delivers powerful functions that enable the user to specify a goal position that Nao will approach. The weakness behind the function is the inaccuracy of the walking. For example, it can not be guaranteed, that Nao walks one meter, if the goal position is one meter in front of it. The actual walking distance in reality is not the same as the planned or commanded one and depends very much on the friction between the floor and Nao's feet. Furthermore this error increases, if Nao has to rotate about his z axis (that means, if Nao has to turn or walk a curve). To solve this problem, a control approach was chosen. Based on image measurements, the walking velocity is adapted. More detailed information about the control loop can be found in section 4.3. The main idea behind the

control loop is, that in a first step the object is detected in the current camera image. If the object is detected, the head turns in such a way, that it looks towards the object. Now Nao steers his walking in the direction, where his head looks. When Nao is getting closer to the object, its speed is reduced until it stops in front of the object.

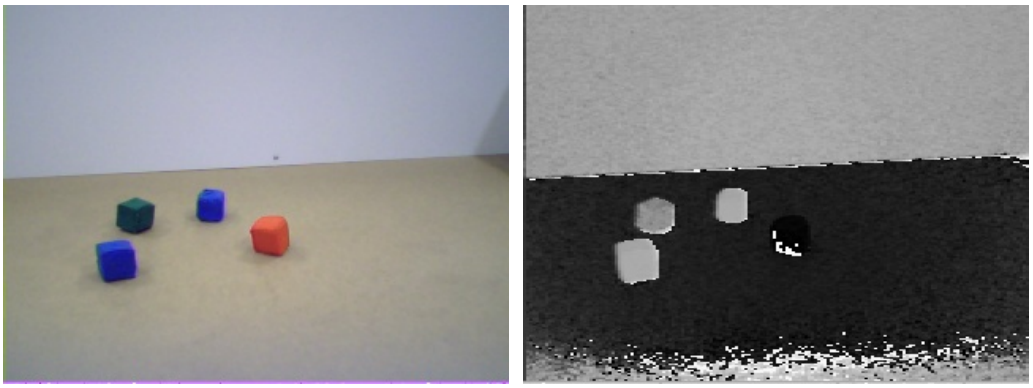
4.2 Specifying the object

As mentioned in chapter 1, Nao has a 500Mhz CPU. If computer vision algorithm should be applied on Nao, the weak CPU is a really bottleneck. It is not easy to overcome this problem, even if the images are sent to a faster external PC via WLAN. In this case the processing is much faster, but the WLAN connection is the bottleneck, because the image data has to be transferred during the walking and real-time cannot be guaranteed. To speed up the image processing, objects were chosen, that are easy to detect in the images. It was mentioned before, that Nao's framework provides access to its camera images in various colorspaces. This part is working with the HSY colorspace, an approximation of the HSV colorspace. To simplify the task, we assume a setting with uncoloured and unsaturated environment and coloured, highly saturated objects. This makes it easier to detect the specific objects in the camera image and to speed up the necessary computation. Figure 4.1 illustrates an image from Nao's camera and its separation into its three different channels in the HSY colorspace. As it can be seen in this figure (especially in figure 4.1(c)) the object detection is much simpler, if the object has a high saturation.

Based on this fact, objects are searched, by looking for highly saturated blobs in the saturation channel. Once such a highly saturated blob is found, its bounding box is used to separate it from the rest of the image. If there is more then one blob in the current image, the blob with the largest area is chosen. Based on this bounding box, histograms are calculated. One for the hue channel of the blob and one for the saturation channel of the blob. These two histograms specify the object. Figure 4.3(b) and figure 4.3(c) show an example of a cube and its histogram. In this case the histograms are calculated using 32 bins for each channel. Because a bounding box can be a very coarse approximation of the object contour, the histograms will also contain hue and saturation information from the background of the object and not from the object alone (see figure 4.4 for better illustration and discussion of such a case).

4.3 The control loop

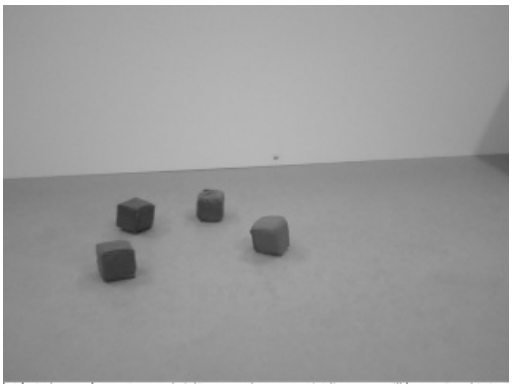
This section outlines the control loop. After specifying the object with the help of a bounding box and two histograms, the object has to be detected in every image, and



(a) cubes

(b) cubes, hue channel (H)

(c) cubes, saturation channel (S)



(d) cubes, brightness channel (Y)

Figure 4.1: This figure shows an image taken by Nao in its room. Figure 4.1(c) illustrates the high contrast between the objects (coloured cubes) and the room (brown floor and white walls).

according to the object position, Nao has to react in a proper way. Figure 4.2 illustrates the relation between *specifying the object* (section 4.2) and the *processing of the visual input* (section 4.3.1).

4.3.1 Processing the visual input

Let us define I_k as the camera image at time k. According to the two histograms, that specify the object, the *back projection* [3] B_k of the histograms is calculated for every image I_k. B_k expresses how well a pixel in the image I_k is represented by the object histograms. Pixels that have the same hue and saturation as the object, have higher values than the others. For a better illustration see figures 4.3 and 4.4. To improve the robustness of the object detection in the image, a *threshold* is applied to the *back projection*. In this case, all other pixels are suppressed, that have not the maximum value. This step ensures, that the artefacts, caused by the coarse bounding box representation of the object, are eliminated. See figure 4.5 to see the effect of the *threshold*. From now on let us define O_k

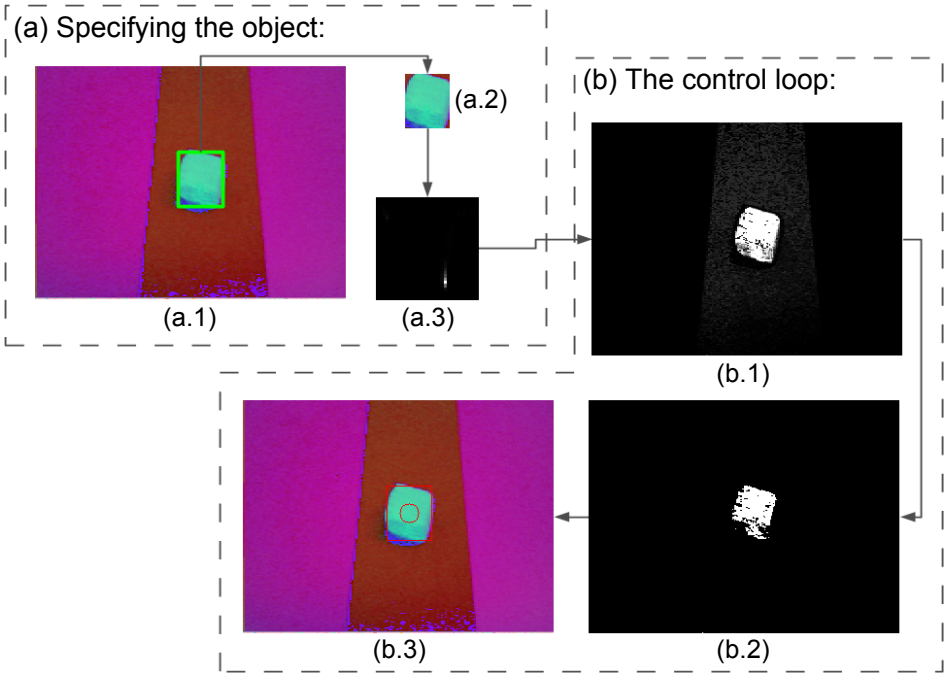
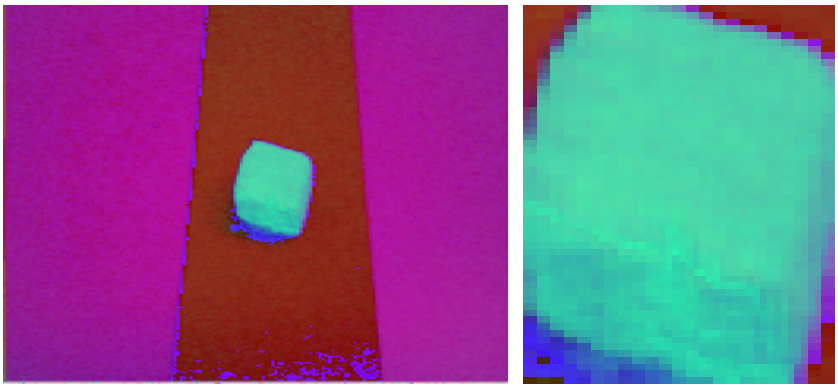
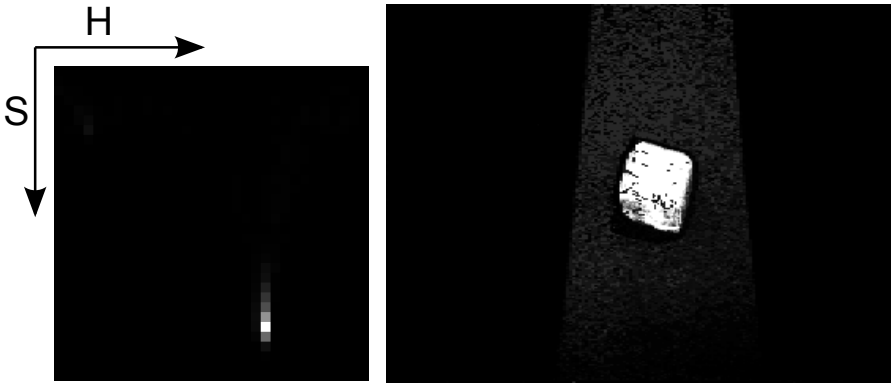


Figure 4.2: This figure illustrates the visual processing of the camera images. First, the object has to be specified (a). To do this, we search the image for highly saturated blobs (a.1). If there is more than one blob in the image, the blob with the largest area is chosen. The *bounding box* of this blob is used to extract a patch from the image (a.2). After that, the hue- and saturation- histogram of the patch is calculated (a.3). This procedure is done at the beginning to specify the object. In each time step of the control loop (b) the following operations are calculated: the *back projection* B_k from the histogram (a.3) is calculated for the current image (b.1); a threshold O_k on this *back projection* is performed (b.2); O_k is used to detect the blob of the object (b.3). If there exists more than one blob, the blob with the largest area is chosen.

as the *back projection* after the *threshold*. As it can be seen very clearly in figure 4.5(b), the object position in the image can now be extracted easily. This work uses a blob detection, implemented by [7], to do this. Blobs are extracted out of O_k and the largest one is chosen to be the object position. Figure 4.5(c) shows the result of the blob detection and extraction. From now on, the tuple $o_{bb} = (x_{ul}, y_{ul}, w, h)$ represents the bounding box of the object, where (x_{ul}, y_{ul}) is the pixel coordinate of the upper left corner of the box, w is the width and h is the height of the box. From this it follows, that the center of the bounding box is $o_c = (x_{ul} + w/2, y_{ul} + h/2) = (o_{c,x}, o_{c,y}) \in \mathbb{R}^2$. The next section describes how the head is controlled according to o_c.



(a) input image (b) the specified cube

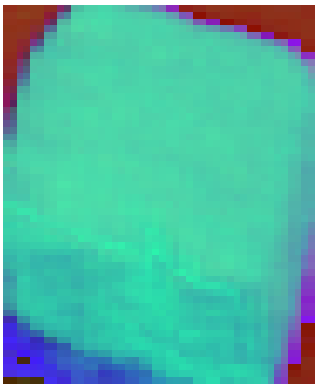


(c) H-S histogram of the cube (d) back projection

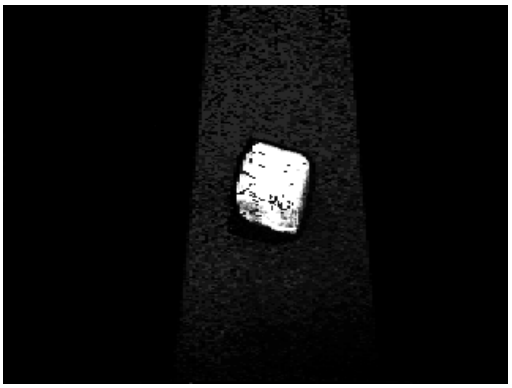
Figure 4.3: 4.3(a) shows the camera image represented in the HSY color space (this is the reason for the wrong colours). In this image the object is specified by a bounding box. The specified object is illustrated in 4.3(b) and its hue (H) and saturation (S) histogram can be found in 4.3(c). The calculated *back projection* is represented in 4.3(d). The brighter pixels are, the higher the values of the *back projection*. A closer look at 4.3(b) shows, that parts of the floor are within the bounding box. This is the reason, why in 4.3(d) also some parts of the floor get higher value. See figure 4.4 for a detailed discussion.

4.3.2 Controlling the head according to the object position

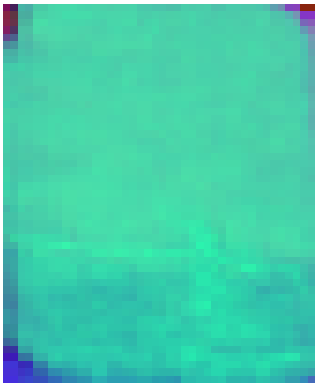
This section describes the control mechanism, that allows Nao to fixate an object by looking towards it. From the previous section the object center is known as o_c. The goal of the approach, described in this section, is, that the head should turn in such a way, that $o_c = t_c$. Where $t_c = (t_{c,x}, t_{c,y}) \in \mathbb{R}^2$ is a desired pixel coordinate position. As it is explained in chapter 2 (see equation 2.10), it is possible to estimate the angle between the principal axis to a visual ray, obtained by reprojecting an image coordinate, when the camera is calibrated. Nao's framework provides a function, that delivers angles, given pixel coordinates. Let us denote $o_{a,k} \in \mathbb{R}^2$ as the angles between the principal axis and the visual ray of o_c and $t_{a,k} \in \mathbb{R}^2$ as the angles between the principal axis and the visual ray of t_c. These angles are evaluated in every time step k. The control law, that moves the



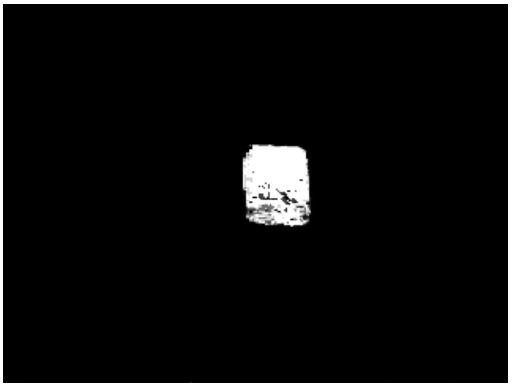
(a) slightly rotated cube



(b) back projection of 4.4(a)



(c) image of a foam cube that fits better into its bounding box



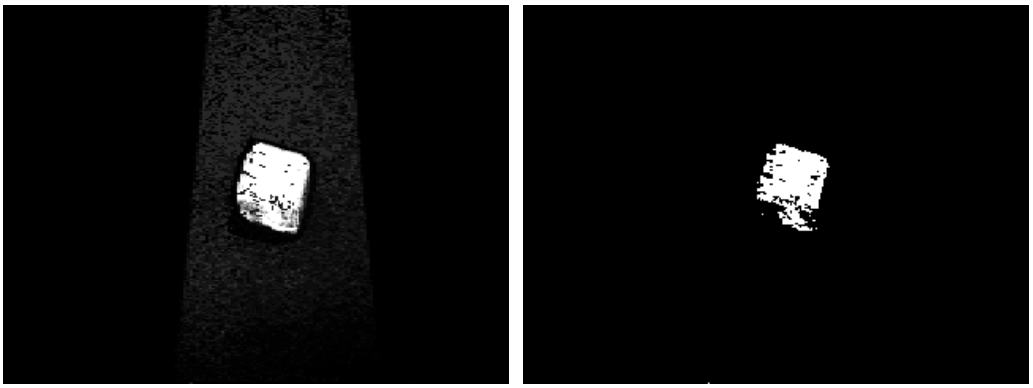
(d) back projection of 4.4(c)

Figure 4.4: This figure illustrates the effect of the bounding box approximation. The *back projection* provides better results, if the object fills the entire bounding box (as it can be seen in figure 4.4(d) and figure 4.4(c)). Because parts of the floor are within the bounding box, in figure 4.4(a) the floor appears also slightly in the *back projection*, shown in figure 4.4(b).

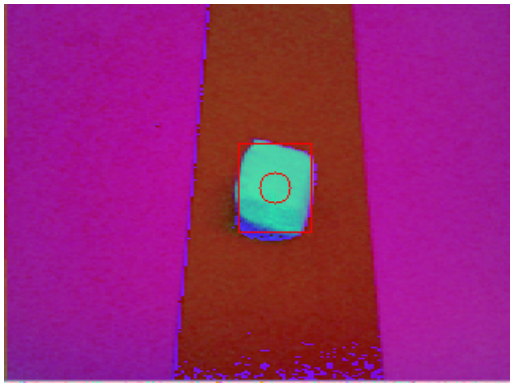
head in such a way, that over time o_c becomes t_c, is given as following:

$$u_{a,k} = (o_{a,k} - t_{a,k})h_c \ . \tag{4.1}$$

Here $u_{a,k} = (yaw_{a,k}, pitch_{a,k}) \in \mathbb{R}^2$ are the angles the head should turn around (figure 4.8(a) shows the head angles of Nao) and $h_c = (yaw_c, pitch_c) \in \mathbb{R}^2$ are controller gains, respective for the head yaw angle and the head pitch angle. h_c has to be chosen in such a way, that the head motion is fast enough to follow an object and to ensure a stable control loop. In this work the gains are chosen to be: $yaw_c = 0.9$ and $pitch_c = 0.4$. Figure 4.6 contains a sequence of images, during a head motion so that o_c becomes $t_c = (160, 80)$. In Figure 4.7 the pixel coordinate values are drawn over time. Here the blue lines represent the desired target position and the red lines are the current pixel coordinates of the object.



(a) back projection (b) after the threshold



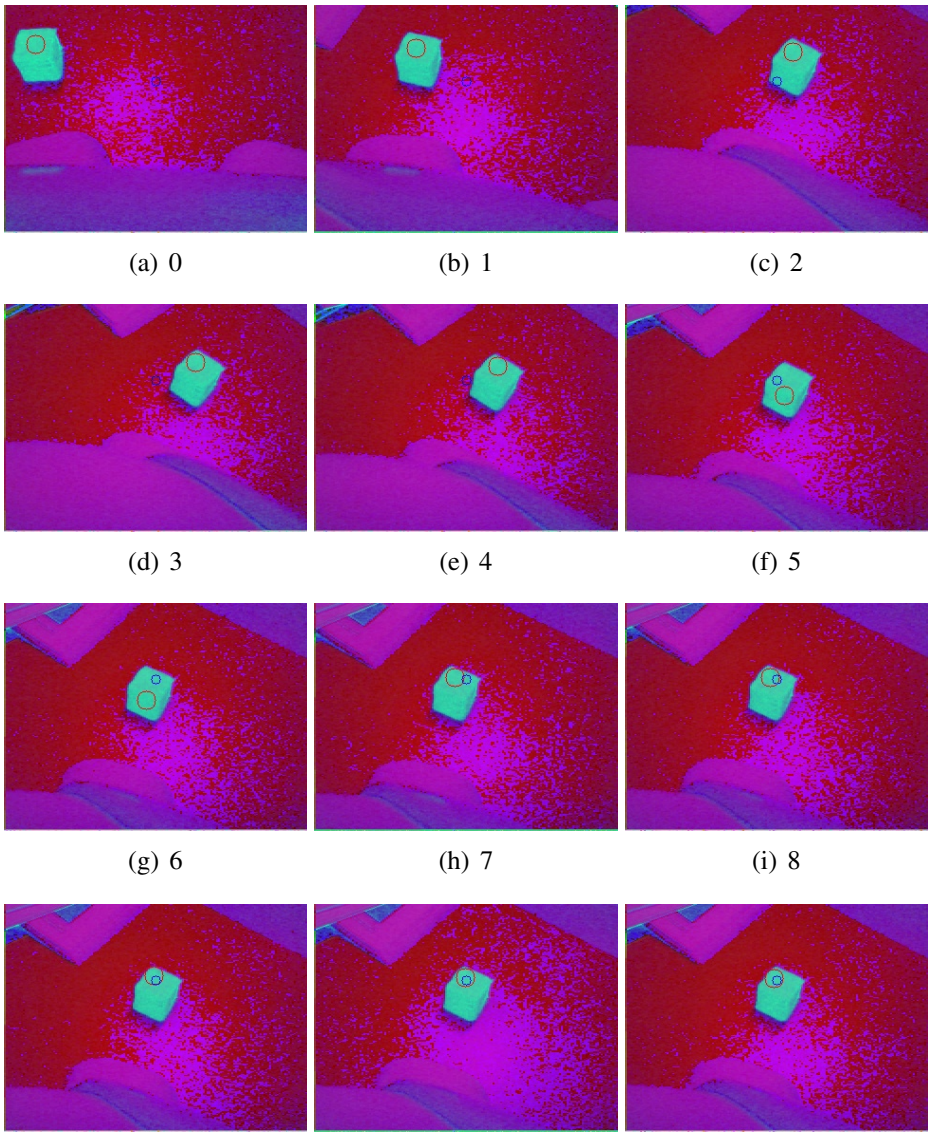
(c) tracked object

Figure 4.5: This figure illustrates the effect of the *threshold* (in 4.5(a) and 4.5(b)). Figure 4.5(c) displays the result of the image processing. The object is marked with a red square and its center is marked with a red circle.

As it was expected, the red curves approach the blue one over time. The next section describes, how the head orientation effects Nao's walking.

4.3.3 Walking velocity according to the head angles

The last sections explained, how Nao is able to track an object with its head. The idea is now, that Nao walks in that direction, where its head looks. If the head looks left, because the object lies on the left side, then Nao should turn to the left and vice versa. If the head looks far away, Nao should walk faster, to approach faster to the object. Depending on the head angles, the velocity of Nao's walking is chosen. From now on $h_{k,yaw}$ is the head yaw angle at time step k, and $h_{k,pitch}$ is the head pitch angle at time step k. Furthermore, $v_{k,x}$ is the walking velocity in x- direction, that means straight ahead, at time step k and $v_{k,\Theta}$ is the velocity with which Nao turns, also at time step k. The framework of Nao provides a function, that enables the user to specify the walking velocity of Nao. The only thing the user has to be aware of is, that the values of the velocity must lie within $-1.0 \ldots 1.0$. From



(a) 0 (b) 1 (c) 2

(d) 3 (e) 4 (f) 5

(g) 6 (h) 7 (i) 8

(j) 9 (k) 10 (l) 11

Figure 4.6: Sequence of images taken during the control process. The red circle is the calculated center of the object o_c, the blue circle is the target position t_c. Figure 4.7 contains the corresponding image positions over time.

[1] it is known that $h_{k,yaw}$ lies between $-2.0857 \ldots 2.0857[rad]$ and $h_{k,pitch}$ lies between $-0.6720 \ldots 0.5149[rad]$. For a better illustration of the different angles, see figure 4.8(a).

Let us define $v_{k,\Theta}$ as the turning velocity of Nao. This velocity should depend on Nao's *head yaw* angle $h_{k,yaw}$ and is defined as follows:

$$v_{k,\Theta} = c_\Theta \cdot h_{k,yaw}^3 \tag{4.2}$$

$$c_\Theta \geq 0$$

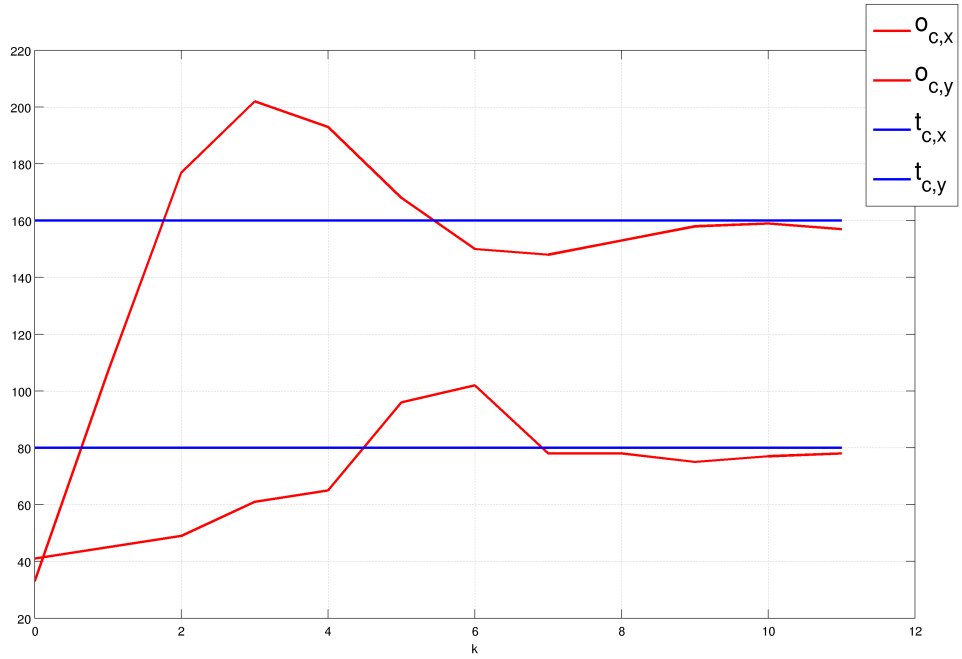


Figure 4.7: Image coordinate positions over time. The blue lines are the target position (blue circle in figure 4.6) and the red lines are the object positions (red circle in figure 4.6).

Here c_Θ is a constant that describes, how fast Nao should turn.

As a second velocity, Nao's forward velocity is defined as $v_{k,x}$. This velocity depends on the *head pitch* angle of Nao's head ($h_{k,pitch}$) and the turning velocity of Nao ($v_{k,\Theta}$) and is defined as follows:

$$v_{k,x} = (-exp(-c_x \cdot (-h_{k,pitch} + c_{MD})) + 1) \cdot (1 - |v_{k,\Theta}|) \tag{4.3}$$

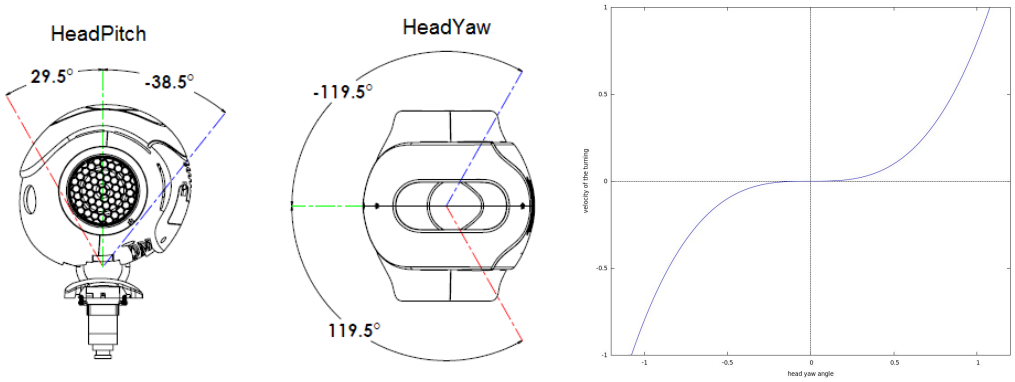
$$c_x \geq 0$$

Here c_x is a constant, that describes, how fast Nao should walk forward and $c_{MD} = 0.43$. This constant describes the angle at which Nao's head looks maximally downwards. In this work $c_x = 5.0$ and $c_\Theta = 0.8$. Figure 4.8 shows an experimental plot of both functions. As it can be seen in equation 4.3 and figure 4.8, the turning velocity also affects the walking velocity. This leads to a slower speed, when Nao has to turn very hard.

Because of Nao's software framework both velocities must lie between -1.0 and 1.0.

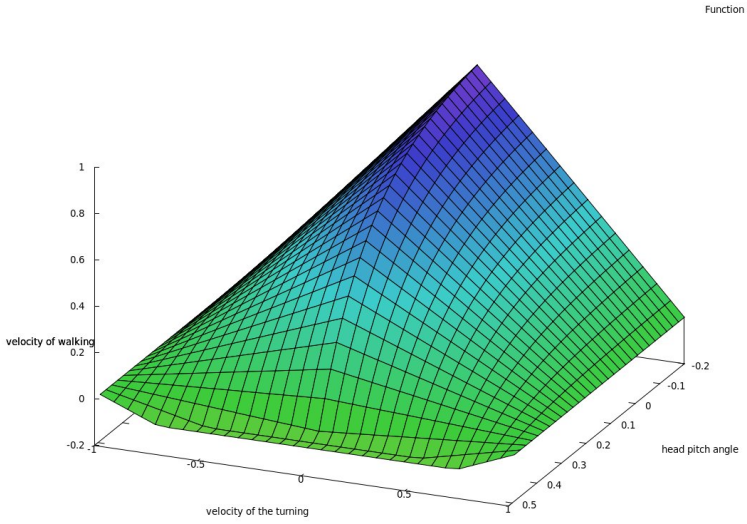
4.3.4 Stopping the walking

As stopping criteria the two velocities, $v_{k,\Theta}$ and $v_{k,x}$, are used to decide if Nao should stop its walking. According to our previous discussion (see section 4.3.3), the current



(a) Nao's head angles

(b) f_Θ



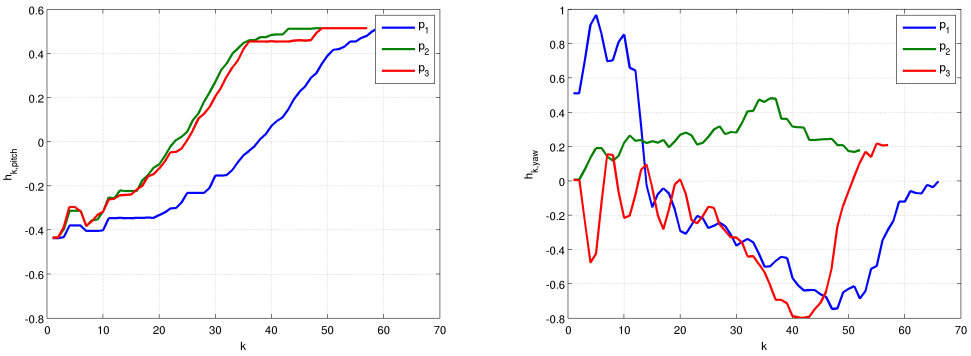
(c) f_x

Figure 4.8: Figure 4.8(a) shows the head angles of Nao (from [1]). Figure 4.8(b) is the plot of the function f_Θ with $c_\Theta = 0.8$, and figure 4.8(c) is the plot of function f_x, with $c_x = 5.0$.

head angles have impact on Nao's walking velocity. If Nao is getting closer to the object (Nao's head is tracking the object), the walking speed is reduced according to the head angles. If the values of the two velocities are below a given threshold for a given timespan, then Nao should stop the walking. In this work, the threshold for $|v_{k,\Theta}|$ is 0.02 and the threshold for $v_{k,x}$ is 0.21. If the calculated velocities are below these thresholds for $k = 10$ time steps, then Nao stops its walking.

4.4 Experimental validation

To proof this concept an experiment was set up. An object was placed on three different positions $p_1 = (1, 0.8), p_2 = (1, 0)$ and $p_3 = (1, -0.3)$ in front of Nao (all units in meter), where the x axis (first element of p_i) directs in front of Nao and the y axis (second element of p_i) directs to the left side of Nao. The task was now, that Nao should walk towards the objects. During the walking, the commanded walking velocities and the head angles were monitored. Figure 4.9 shows the result of the experiment. First, it is important to note,



(a) $h_{k,pitch}$ (b) $h_{k,yaw}$

(c) $v_{k,x}$ (d) $v_{k,\Theta}$

Figure 4.9: Plot of the head angles (figure 4.9(a) and 4.9(b)) and the resulting velocities, calculated by the equations 4.3 (figure 4.9(c)) and equation 4.2 (figure 4.9(d)).

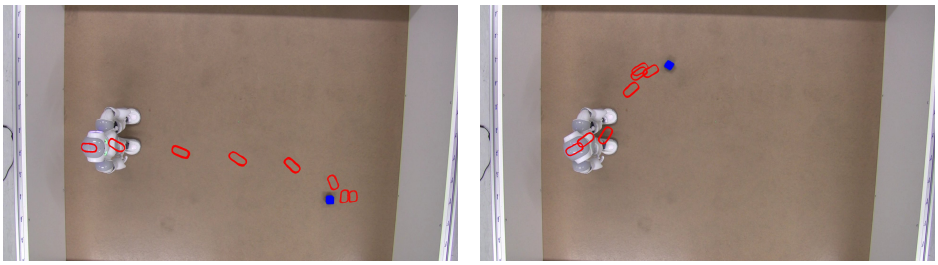
that the calculated velocities ($v_{k,x}$ and $v_{k,\Theta}$) can not be compared with the real walking speed. The framework of Nao requires values between $-1.0 \ldots 1.0$, so $v_{k,x}$ and $v_{k,\Theta}$ have to lie within this interval.

The recorded values for p_1 are shown by the blue line (figure 4.9). The object lay in front of Nao on the left side. Intuitively one would go forward and turn left, to approach the object. This is the same way, that Nao walks, as it is expressed by the blue velocity line in figure 4.9(c) and 4.9(d). If the object is "far" away (indicated by a negative head pitch angle, see figure 4.9(a) and figure 4.8(a)), Nao moves faster. Because the object

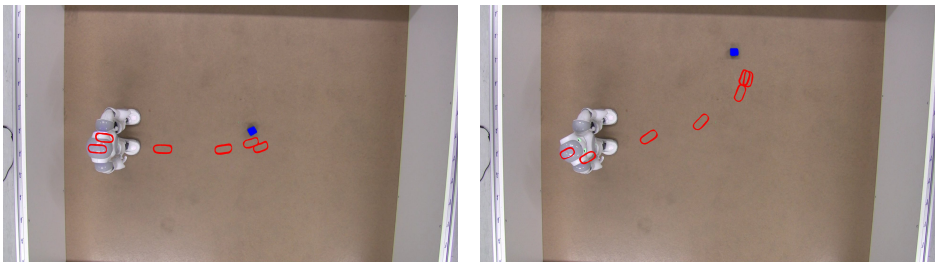
lies on the left side of Nao, it has to turn its head to the left and as a result, the speed is reduced. This explains the drop of the velocity (in figure 4.9(c)) at the beginning of the walking.

The green line (this represents p_2, the object lies straight in front of Nao) has no such velocity drop, because the object lies directly in front of Nao. It can also be observed, that there is a small velocity drop at the red line (p_3, object in front of Nao at the right side). The difference between the two velocity drops express the fact, that p_1 lies farther left, than p_3 lies right. Therefore, Nao has to turn more to the left side, than it has to turn to the right. In all three cases Nao stops its walking in front of the object. The presented approach does not ensure, that Nao stops always at the same position in front of the object. The fine positioning next to the object has to be done separately. This, and other system relevant topics, are explained in the next chapter.

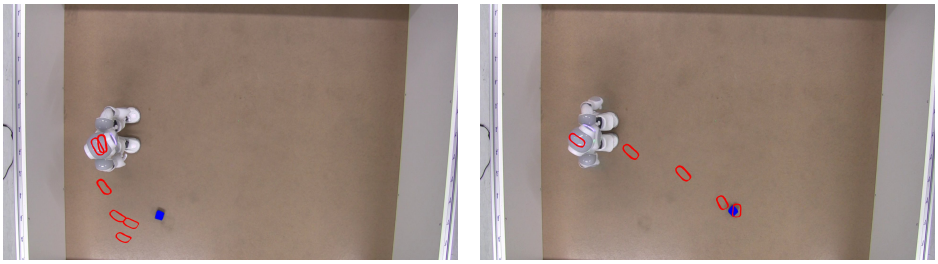
To illustrate the walking of Nao in a more meaningful way, a second experiment was set up. The cube was placed randomly inside the room and Nao had to walk towards it. During the walking, the motion of Nao was captured with a video camera. The result of this experiment is presented in figure 4.10. In this figure, the red ellipses represent the discrete states of Nao's walking trajectory.



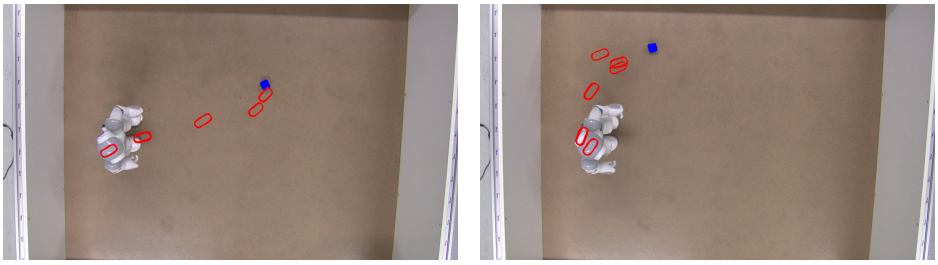
(a) Sequence 0 (b) Sequence 1



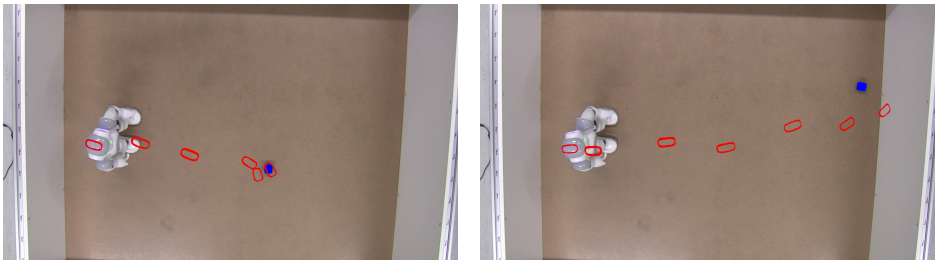
(c) Sequence 2 (d) Sequence 3



(e) Sequence 4 (f) Sequence 5



(g) Sequence 6 (h) Sequence 7



(i) Sequence 8 (j) Sequence 9

Figure 4.10: This figure illustrates the trajectory of Nao, while it walks towards a blue foam cube. The red ellipse like shapes represent the position of Nao's head tactile sensor (the discrete state of Nao's trajectory). The position of the blue foam cube is marked with a blue square.

5 Description of the whole system

Chapter 3 and chapter 4 presented the main part of this work. It was shown how the *virtual thumb* concept works, that is used for the grasping purpose, and how Nao should walk towards an object. The following chapter points out, how all things work together to enable Nao to pick up objects from the floor. For an overview of the whole data processing flow, have a look at figure 5.1. In the current chapter of the work, it has to be ensured, that Nao stands in the right place (fine positioning of Nao), bows down in a proper way and checks, if it was able to grasp the object. If Nao was not able to pick up the object, the whole pick up procedure has to be repeated.

5.1 Looking for an object

As mentioned already in chapter 4, the colors of the used objects are very saturated. Therefore, in the first step, Nao looks for very saturated objects. To do this, Nao's head looks into several directions. At each of these directions, an image is captured and analysed, if a highly saturated object is within it. This is done with the help of a blob detection algorithm, presented in [7]. If Nao is not able to find the cube, it turns and repeats the

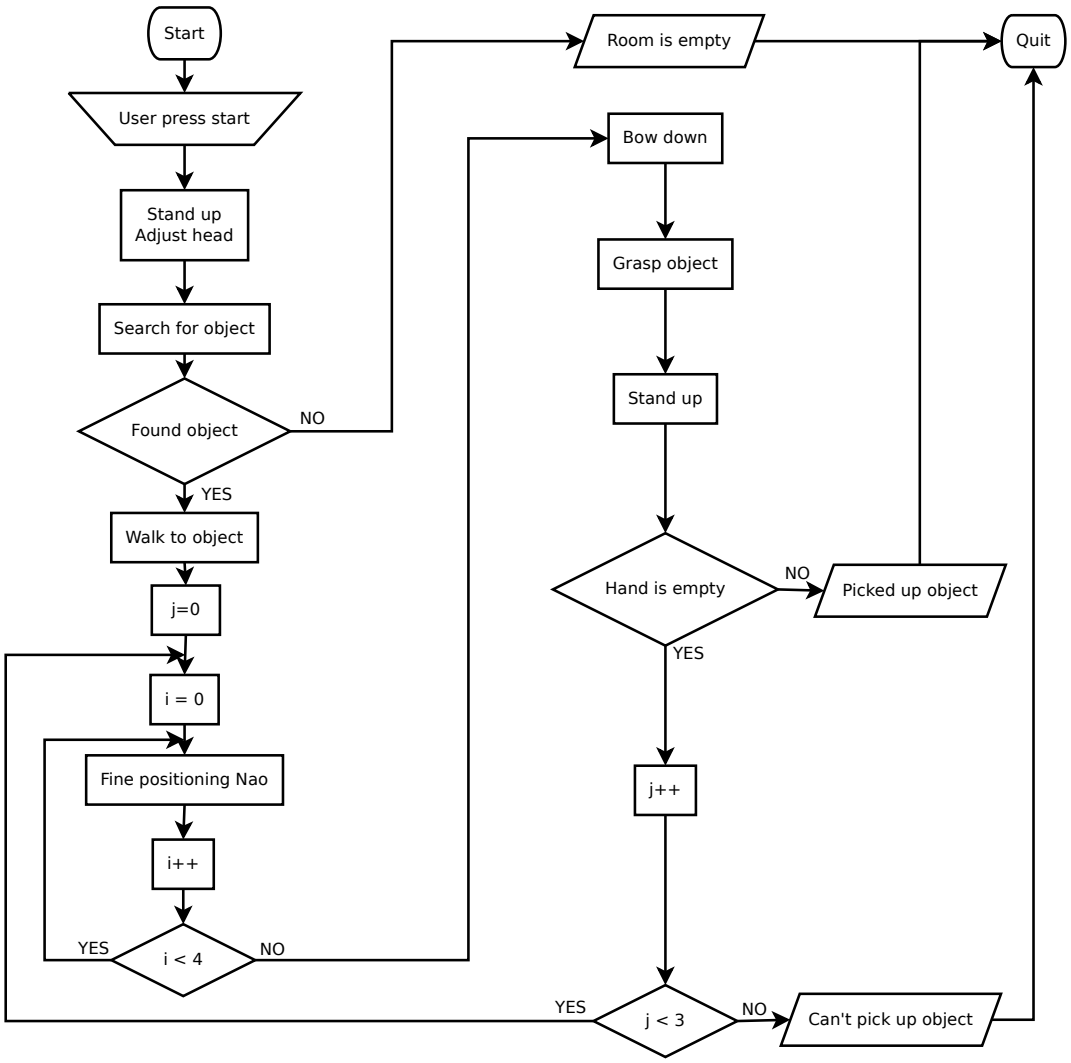


Figure 5.1: Flowchart of the whole system. For a detailed explanation of key functions, see: “Search for object” - section 5.1; “Walk to object” - section 5.2 and chapter 4; “Fine positioning Nao” - section 5.3; “Bow down” - section 5.4; “Grasp object” - section 5.5; “Stand up” - section 5.6;

search again. Here is the list of angles for the search strategy (all angles in [rad]):

$$h_{pitch} = (-0.4363, 0.0, 0.4363)$$
$$h_{yaw} = (-1.047, -0.524, 0, 0.524, 1.047)$$
$$\Theta = (2, 2, 0)$$

Here, h_{pitch} are the angles of the head pitch anlge, h_{yaw} are the angles of the head yaw angle, and Θ are the angles of the body rotation.

The size of the blob area, returned by the blob detection algorithm, has to be above a threshold ($th = 500$). This prevents the detection of small highly saturated objects in

the room. If Nao is not able to find an object, after going trough all of these angles, it is assumed, that the room is empty. Otherwise, this routine delivers the bounding box around the object.

5.2 Walking towards the object

With the bounding box, received by the previous routine, the cube is specified as it is explained in chapter 4. Then Nao walks towards object and stops next to it. The only thing that can be said about the position where Nao stops is that Nao stops "near" to the object, but of course, this is not accurate enough. The next section deals with the fine positioning of Nao, relative to the object.

5.3 Fine positioning of Nao

After getting near to the object, Nao's position is iteratively adjusted, so that it is able to grasp the object, if it bows down. This section explains first, how the object distance is calculated. Afterwards it describes an experiment, that was used to estimate the goal position of Nao, relative to the object, so that a high rate of successful grasping is achieved.

5.3.1 Estimation of the object position

As it was explained earlier, the object is represented by its bounding box. Let us denote the center of the bounding box in homogeneous pixel coordinates as $\mathbf{o}$. With the help of a calibrated camera it is possible to project this point back into the world. If $\mathbf{K}$ is the calibration matrix, this is done with $\mathbf{O}^{Cam} = \mathbf{K}^{-1}\mathbf{o}$, with $\mathbf{O}^{Cam}$ being the ray from the camera center though the pixel coordinate point $\mathbf{o}$. Because Nao uses another coordinate system than the camera framework, this ray must be converted into the Nao reference frame system. This is done by simply rotating it (In chapter 2 the same thing is done. For a better understanding look at figure 2.10 and equation 2.17) with the rotation matrix $\mathbf{R}_{Cam}^{World}$, this leads to $\mathbf{O}^{\mathcal{C}} = \mathbf{R}_{Cam}^{World\,-1}\mathbf{O}^{Cam}$, with $\mathbf{O}^{\mathcal{C}}$ being the ray relative to the camera reference frame $\mathcal{C}$. Furthermore define $\boldsymbol{\pi}_f^{\mathcal{T}}$ as the floor plane relative to the torso reference system $\mathcal{T}$, $\mathbf{T}_{\mathcal{T}}^{\mathcal{C}}$ as the homogeneous transformation from the torso reference frame $\mathcal{T}$ to the camera reference frame $\mathcal{C}$ and $\mathbf{T}_{\mathcal{N}}^{\mathcal{C}}$ as the transformation from the Nao reference frame $\mathcal{N}$ to the camera reference frame $\mathcal{C}$. The origin of the Nao reference frame $\mathcal{N}$ is the average of the two feet positions and its x-axis looks always forward [1]. This reference frame is useful for the distance measurement, and the position of Nao relative to the object is relative to this reference frame. In a first step, the plane $\boldsymbol{\pi}_f^{\mathcal{T}}$ is transformed into the

camera reference frame $\mathcal{C}$ by applying the equation $\boldsymbol{\pi}^{\mathcal{C}} = \mathbf{T}_{\mathcal{T}}^{\mathcal{C}\,T}\boldsymbol{\pi}_f^{\mathcal{T}}$. What is searched for, is the intersection point between the plane $\boldsymbol{\pi}^{\mathcal{C}} = (\pi_1, \pi_2, \pi_3, \pi_4)^T$ and the parametrized ray $\mathbf{O}(\nu)^{\mathcal{C}}$, which is $\mathbf{O}(\nu)^{\mathcal{C}} = (\nu O_x, \nu O_y, \nu O_z, 1)^T$. This is done by solving the equation $\boldsymbol{\pi}^{\mathcal{C}\,T} \cdot \mathbf{O}(\nu)^{\mathcal{C}} = 0$ for ν, which has the result

$$\nu = \frac{-\pi_4}{\pi_1 O_x + \pi_2 O_y + \pi_3 O_z} \quad . \tag{5.1}$$

After the evaluation of $\mathbf{O}(\nu)^{\mathcal{C}}$ with the calculated ν, the result has to be transformed into the Nao reference frame $\mathcal{N}$. This is done by

$$\mathbf{O}^{\mathcal{N}} = \mathbf{T}_{\mathcal{N}}^{\mathcal{C}}\mathbf{O}^{\mathcal{C}} \quad . \tag{5.2}$$

Here, $\mathbf{O}^{\mathcal{N}}$ is the position of the object, relative to the Nao reference frame $\mathcal{N}$ and is originally defined by his pixel coordinates $\mathbf{o}$. The next section describes how the position is found at which the grasping ability of Nao could be improved.

5.3.2 Finding a good distance between Nao and the object

Nao was placed next to the object. The position of the object was calculated according to the method presented in the previous section. Afterwards, Nao bows down and tries to grasp the object. In an experiment, it was noted, at which position Nao could grasp the object. Figure 5.2 shows the results of this record.

The plot of the result of the experiment shows that there is no significant cluster, at which Nao was able to pick up the object. But this experiment provides a coarse guess, where the object should lie relative to Nao. Based on these findings, a target position $p_o = (0.35, -0.06)$ (units in meter) of the object relative to Nao was selected.

The question is now, if this choice is sufficient. First, this experiment has an inherent disadvantage, because the object position is measured, while Nao is standing. During the bow down phase, Nao's position also changes. This change in position depends on the friction between Nao's feet and the floor and can not be determined, because it is a very stochastic process. To illustrate this, the error $\mathbf{e}$ between the measured distance when Nao stands, and the measured distance after the bow down phase, was calculated according to $\mathbf{e} = \mathbf{p}_{down} - \mathbf{p}_{up}$, where $\mathbf{p}_{down}$ is the distance from Nao to the object when Nao is next to the floor and $\mathbf{p}_{up}$ is the position of the object relative to Nao, when Nao stands. Figure 5.3 shows the results of this experiment.

It can be seen, that the deviation along the y axis of Nao is very high, compared to the deviation along the x axis ($\sigma_x = 0.0077$ and $\sigma_y = 0.0312$; units in meter). This means, that during the bow down phase, Nao slips to the left or to the right relative to the

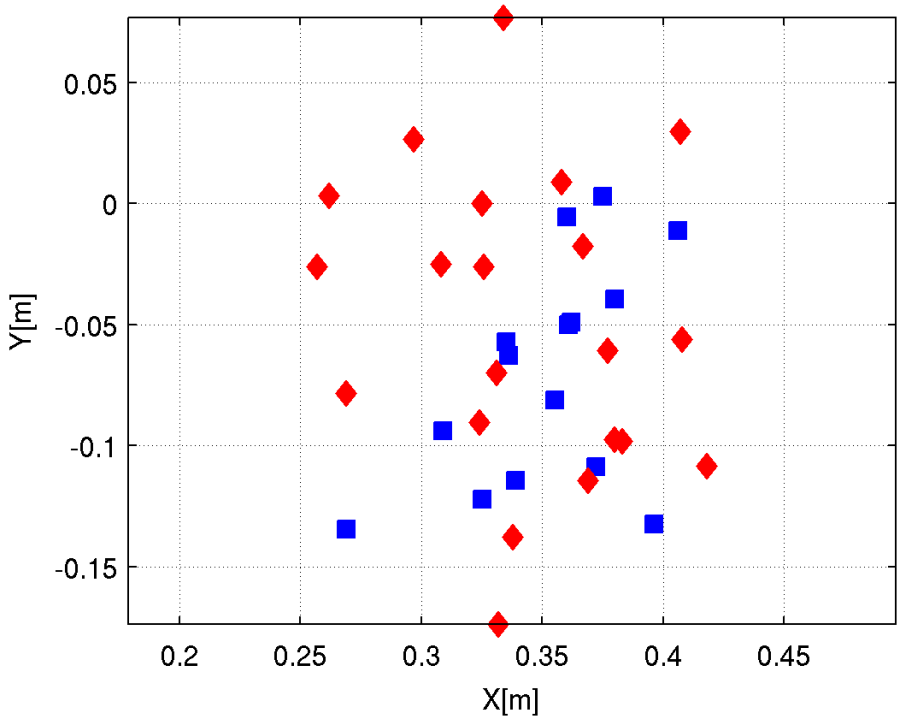


Figure 5.2: Position of the object relative to Nao during the experiment with 36 runs. In the case of the blue squares, Nao was able to grasp the object, contrary to the red diamonds, where it was not able to pick up the object. The axis with the "$Y[m]$" label is the axis, that is parallel to Nao's y axis, which means, the positive direction of this axis points to the left of Nao. The positive x axis points in front of Nao.

object. This error has to be taken into account. Therefore, an iterative pickup approach was chosen. After the fine positioning, Nao tries to pick up the cube and afterwards it checks, if the pickup was successful. If this does not happen, Nao stands up, repeats the fine positioning, and tries to pick up the cube again. Because of the stochastic process during the bow down phase, Nao may not be able to grasp to cube at the first time. If this happens, Nao tries it a second time. The experiment, presented at the end of this chapter, will proof that this concept works quite well, despite the displacement during the bow down phase.

So the fine positioning of Nao ensures, that Nao stands at a desired position relative to the object, but this position itself does not guarantee that Nao is able to grasp the object. This is caused by the displacement during the bow down phase.

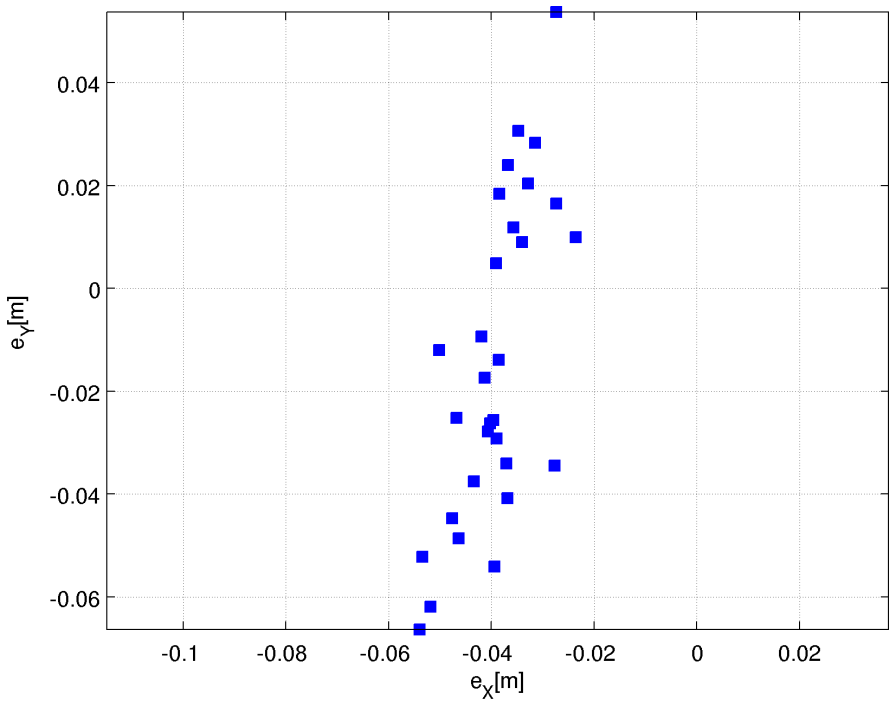


Figure 5.3: Distribution of the error $\mathbf{e} = \mathbf{p}_{down} - \mathbf{p}_{up}$ (number of experiments= 30). The axis with the "$e_Y[m]$" label is the axis, that is parallel to Nao's y axis, which means, the positive direction of this axis points to the left of Nao. The positive x axis points in front of Nao. This experiment shows a great deviation along the y axis.

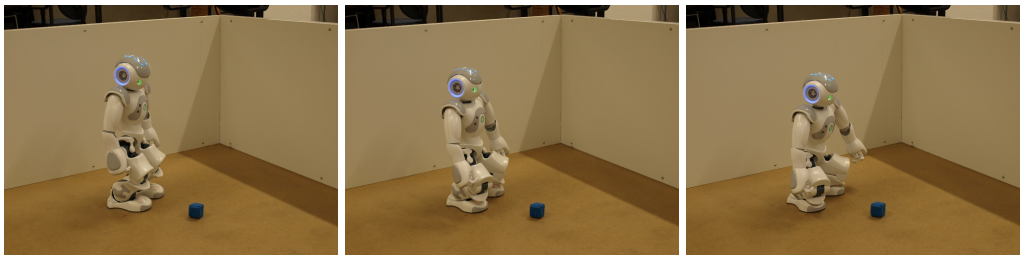
5.4 Bowing down

Nao's framework allows to record motions and to repeat them afterwards. This function was used to enable Nao to bow down to the floor. Figure 5.4 shows an image sequence of Nao while it bows down.

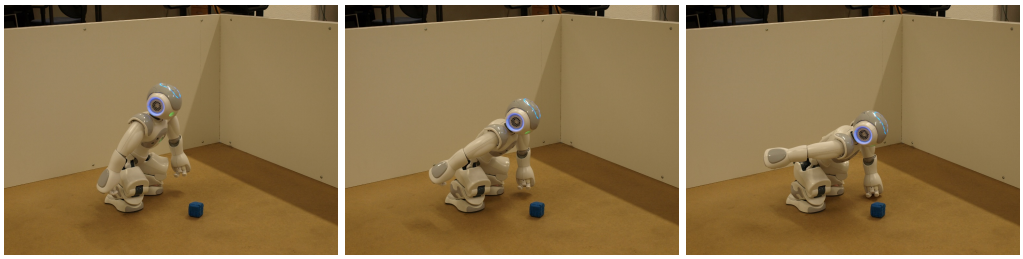
Because Nao has no backbone, it is not a trivial task to let it bow down. During the bow down phase it has to be ensured, that Nao does not fall forward. To overcome this problem, Nao supports itself with its left arm. Afterwards, the grasping is done with the right arm.

5.5 Grasping the object

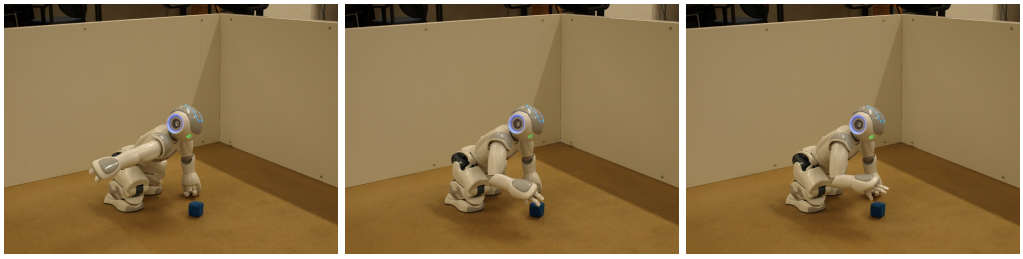
After Nao has bowed down to the floor, it has to grasp the object. This is done with the method, described in chapter 3. This section presents the details, that are necessary for the implementation, but not for the understanding of the overall process. First, the bounding box of the object $o_{bb} = (x_{ul}, y_{ul}, w, h)$ is extracted. The target position $\mathbf{s}^\star = (s_x^\star, s_y^\star)$, that the virtual thumb should approach, is the center of the bottom line of the bounding box



(a) 0 (b) 1 (c) 2



(d) 3 (e) 4 (f) 5



(g) 6 (h) 7 (i) 8

Figure 5.4: Sequence of images taken during bow down process.

plus an offset, mathematically expressed as

$$s_x^\star = \frac{x_{ul} + w}{2} + x_{offset}$$
$$s_y^\star = y_{ul} + h + y_{offset} \ ,$$

with $x_{offset} = 3$ and $y_{offset} = 40$ being the offset in pixel coordinates in x and y direction. After the target position $\mathbf{s}^\star$ is defined, the motion plane is defined. This plane is parallel to the floor with an offset of $75[mm]$ above the floor. The right arm moves now into the motion plane. As it is known from chapter 2 and chapter 3, the *interaction matrix* is required to control the arm in a proper way. This matrix is precomputed with the help of some exploration movements of the right arm. The result of this pre-computation is:

$$\widehat{\mathbf{L}}^{-1} = \begin{pmatrix} -0.001291 & 0.0004441 \\ -0.0003545 & -0.0007072 \end{pmatrix} \ .$$

After that, the controller is initialized with a controller gain of $\lambda = 0.25$ and a period time of $300[ms]$. The controlling is done, until the norm of the error $\mathbf{e}(t)$ is below a threshold $f_{Th} = 5[Pixel]$, that means, $||\mathbf{e}(t)|| \leq f_{Th}$, or the controller is unable to reach the target position with less than 40 control steps.

After the target is reached, the arm is rotated in such a way, that the y axis of the arm is parallel to the floor. This is done by converting the *desired plane* $\boldsymbol{\pi}^{\mathcal{T}}$ (this plane is relative to the torso reference system $\mathcal{T}$ and is equal to the floor plane) into the right arm reference system $\mathcal{R}$ with the equation $\boldsymbol{\pi}^{\mathcal{R}} = \mathbf{T}_{\mathcal{T}}^{\mathcal{R}\,T}\boldsymbol{\pi}^{\mathcal{T}}$. The resulting normal vector of the plane $\boldsymbol{\pi}^{\mathcal{R}}$ is $\mathbf{n}^{\mathcal{R}} = (n_x, n_y, n_z)^T$ (this vector can be constructed from the first three elements of $\boldsymbol{\pi}^{\mathcal{R}}$ [19]). Now the normal vector $\mathbf{n}^{\mathcal{R}}$ is projected onto the yz plane of the right arm reference system $\mathcal{R}$, which leads to $\mathbf{n}_{YZ} = (n_y, n_z)^T$. The angle Φ is the angle between the z axis of $\mathcal{R}$ and the projection of the plane's normal vector, called $\mathbf{n}_{YZ}$ and is calculated according to $\Phi = \sin^{-1}\frac{n_y}{\|\mathbf{n}_{YZ}\|}$. By rotating the arm about $-\Phi$ the y axis of the right arm reference system $\mathcal{R}$ is parallel to the floor.

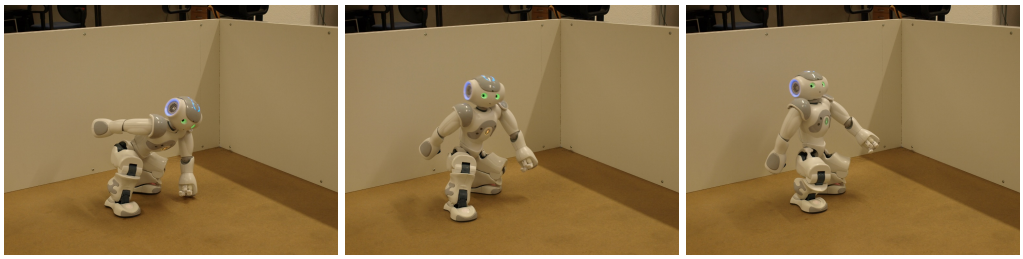
Now the left arm moves $25[mm]$ upwards. This causes, that Nao tilts slightly forward and the right arm therefore moves downwards. After that, the right arm moves $30[mm]$ along the normal vector of the *desired plane* towards the object and grasps it. After sitting up again, Nao checks (by looking for a highly saturated blob in its hand), if the grasping was successful. If that is not the case, Nao stands up, does the fine positioning, bows down and tries to grasp the object again.

5.6 Standing up

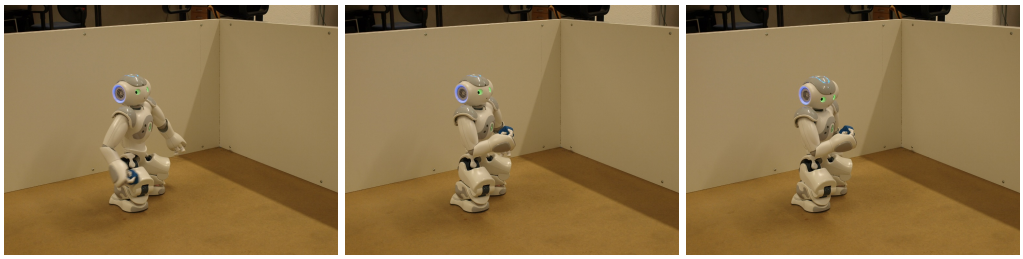
Similar to the implementation of the bow down phase, the stand up phase was implemented, based on the recorded joint angles. Figure 5.5 shows Nao while standing up. It must also be ensured again, that Nao does not fall while it stands up. Therefore it supports itself with its right leg (see figure 5.5 (a) and (b), where Nao's right leg supports itself).

5.7 Experiments and results

To verify the functionality of the presented concept, an experiment was set up. In this experiment Nao was put into the room, described in chapter 1. A coloured foam cube was placed randomly into the room (The distance between the foam cube and the room border was at least $30[cm]$). The task was now, to pick up the foam cube, no matter where the cube lies relative to Nao. This experiment was repeated 50 times and it was monitored if Nao was able to pick up the cube or not, and if it was able to pick up, how many times the fine positioning, bowing down and grasping task had to be done, until Nao had



(a) 0 (b) 1 (c) 2



(d) 3 (e) 4 (f) 5

Figure 5.5: Sequence of images taken during the standing up process.

successfully grasped the cube. Table 5.1 presents the results of this experiment. In 80

Outcome of the experiment	Number	Percent
Pick up was successful	40	80
Nao grasped the cube the first time	32	80
Nao needed two trials to grasp the cube	7	17.5
Nao needed three trials to grasp the cube	1	2.5
Pick up failed	10	20

Table 5.1: Results of the pick up experiment.

percent of the cases, Nao was able to pick up the foam cube, that was placed randomly inside the room. In 80 percent of these cases, it grasped the cube the first time. Only in 20 $(17.5 + 2.5)$ percent it had to stand up and try to grasp the cube again. The result shows, that the assumptions taken in section 5.3 are acceptable.

What happened when Nao was unable to pick up the cube? Table 5.2 gives an overview of the various reasons, why Nao was unable to pick up the foam cube. These reasons are now examined in more detail.

Hand is empty: In this case Nao picked up the cube successfully, but he was unable to recognize that the cube was in its hand.

Nao fell: During the bow down phase, Nao tilted forward and fell. This happened only once and the reason for that could be found in the temperature of Nao's joints. To

Reasons for failing	Number
Hand is empty	2
Nao fell	1
Lost cube during walking	4
Nao pushed the cube away from it	1
Room is empty	2

Table 5.2: Reasons, why Nao was unable to pick up the foam cube.

overcome this problem, it has to be ensured, that the joint temperature is not too high.

Lost cube during walking: Nao lost the cube during the walk towards it. Figure 5.6 shows a patch from a cube, that Nao lost during the walking. It can be seen, that a large

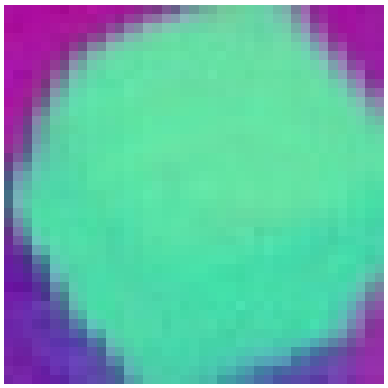


Figure 5.6: Cube patch, that was lost during the walking of Nao.

area of the patch (and therefore also of the histogram) is the background (in this case the floor). Therefore, the back projection of the histogram will also vote for the floor and the robustness of the tracking will decrease.

Nao pushed the cube away: In this case Nao pushed the cube away, which resulted in a never ending walk.

Room is empty: Nao was not able to find a cube inside the room.

Based on the outcome of this experiment, some improvements were made. The reason for the *Hand is empty* failure could be found in the parameters of the blob detection algorithm [7]. When Nao has grasped (or not grasped) the cube, it sits up and checks its right hand if a highly saturated blob is in it. The area of this blob has to be larger than a certain threshold. In the case of the *Hand is empty* failure, this threshold was too high, so the threshold level was reduced.

The case where Nao fell should be overcome by paying attention, that the temperature of Nao's joints is not too hot.

To prevent the *Nao pushed the cube away* case, the threshold, when Nao should stop, was increased to let Nao stop earlier.

The case where Nao was unable to detect the cube (*Room is empty*) was overcome by decreasing the threshold for the blob area (this is similar to the *Hand is empty* case).

In the following experiment, Nao was placed in the middle of the room. The orientation is not strictly defined. In one experimental run, Nao tried five times to pick up the cube. Before the experiment was started, the maximal joint temperature had to be less than $40°C$. After the five runs, the "naoqi" main executable was restarted. A blue coloured cube was placed randomly within the room. The distance to the room border had to be larger than $30[cm]$.

The results of this experiment are presented in table 5.3. It can be seen, that Nao's

Outcome of the experiment	Number	Perscent
Pick up was successful	44	88
Nao grasped the cube the first time	36	81.82
Nao needs two trials to grasp the cube	7	15.91
Nao needs three trials to grasp the cube	1	2.27
Pick up failed	6	12

Table 5.3: Result of the second pick up experiment.

pick up ability is higher than in the first experimental run. Table 5.4 shows the reasons, when Nao was unable to pick up the blue foam cube. It can be seen that the threshold, that

Reasons for failing	Number
Hand is empty	3
Nao fell	1
Lost cube during walking	2

Table 5.4: Reasons, why Nao was unable to pick up the foam cube.

is used to decide, if the cube is in Nao's hand, is still too high. It should be noted, that in the *hand is empty* case, Nao was able to pick up the cube, but it was unable to recognize that.

5.8 Robustness of the results

To verify the robustness of the presented approach, the approach was tested on another Nao humanoid robot. The result of the test should demonstrate, that the presented approach does not depend on the specific Nao robot, on which the program was developed.

In this experiment, the same setup was used, as in the experiments before. The difference was another Nao robot and just five pick up tasks were run, instead of fifty. In all of these five runs, Nao was able to pick up the blue foam cube. In four out of five cases, it was able to do this at the first time. In only one case Nao needed three trials to pick up the cube. This experiment shows, that the presented approach can be used also on another Nao humanoid robot.

Another interesting result is, that the exact calibration of the camera is not that important, as originally thought. As mentioned before, the only thing that was changed between the experiments was the robot, but the camera calibration remains the same. This shows, that the functionality of the robot does not depend strongly on the used camera calibration and is therefore robust against parameter variations of the camera parameters. To get an impression of the order of the variation, table 5.5 shows the parameter variation between Nao's top camera and its bottom camera. The order of magnitudes of the parameter variations could also be expected between two different robots.

Parameter	bottom camera	top camera	standard deviation
f_x	379.74162	376.23709	1.752265
f_y	380.32080	376.69494	1.81293
p_x	151.62371	164.97117	6.67373
p_y	114.28645	116.48445	1.099
radial distortion	$[0.31431, -1.10508]$	$[0.28675, -1.02994]$	$[0.01378, 0.03757]$
tangential distortion	$[-0.00016, -0.00092]$	$[0.00413, -0.00097]$	$[0.002145, 0.000025]$

Table 5.5: Typical parameter variations of the camera calibration.

It is important to notice, that the tests were made on a Nao humanoid robot *V3.3*, as other versions have shorter arms and therefore the grasping would not be possible.

6 Conclusion

The work shows that vision based picking up of objects with the humanoid robot Nao is possible. This task was accomplished with the help of algorithms, known from computer vision [4] and geometry [19], and the usage of the software frame work, provided by Nao [1]. One of the best tools to solve this task was the implementation of feedback controllers. These controllers were used to move Nao's arm in such a position that it was able to grasp the foam cube, and to enable Nao to walk towards the cube. The work with Nao shows, that a feedback is essential to solve the pick up task properly. A vision process in the feedback loop can be a very time consuming calculation. Because of the limited processing power of Nao, only simple computer vision algorithms were implemented. If an application had no time constraints (In the case when Nao walks towards a cube, there is a strong time constraint), more sophisticated algorithm could be used on Nao (this was done for example in [8]).

A strength of Nao is its software framework. The framework provides a lot of useful functions that work right away. The teach in phase to start programming with Nao does not take a long time. This is a very proper work of *ALDEBARAN Robotics*, especially when keeping in mind, what a complex system Nao is.

As mentioned before, a weakness of Nao is its low processing power, especially when image processing is required. In this work only simple image processing routines were used, to overcome this weakness. As mentioned, the grasping task is confronted with

problems, caused by the inaccuracy of Nao's positioning ability. It was necessary to overcome these problems with the help of feedback controllers.

In further work, the presented solution can be used, for example to tidy up a small room with Nao. In this case, more sophisticated algorithms have to be used to distinguish between different objects. More object properties have to be taken into account, to solve this. For example an edge model of the object could be used(this was done in [9]). In all of these works, the low processing power has to be kept in mind. To overcome this low processing power, the images could be sent to a remote PC to do the processing on a faster CPU, but our experience has shown, that online processing via WLAN involves other problems in the domain of time critical network communication.

The work with Nao shows, that this robot platform has a great benefit because it is easy to start working with. In the field of image processing creativity is needed to overcome the processing power bottleneck. It is to be hoped that *ALDEBARAN Robotics* will increase the processing power of Nao's CPU.

Bibliography

[1] Aldebaran Robotics, *NAO user guide*, 2010.

[2] T. Höll, "Vision-based grasping of objects from a table using the humanoid robot Nao." Seminar project, Inst. of El. Measurement and Measurement Signal Processing, Graz Univ. of Technology, 2011.

[3] Willow Garage, *OpenCV 2.1 C++ Reference*, 2011 (accessed August 26, 2011). available at `http://opencv.willowgarage.com/documentation/cpp/index.html`.

[4] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*. CL-Engineering, third ed., 2008.

[5] J. Strom, G. Slavov, and E. Chown, "Omnidirectional walking using zmp and preview control for the Nao humanoid robot," in *RoboCup 2009: Robot Soccer World Cup XIII* (J. Baltes, M. Lagoudakis, T. Naruse, and S. Ghidary, eds.), vol. 5949 of *Lecture Notes in Computer Science*, pp. 378–389, Springer Berlin / Heidelberg, 2010.

[6] "Robocup," 2011 (accessed August 26, 2011). `http://www.robocup.org`.

[7] T. Schlager, "Vision-based control on the Nao academic robot." Bachelor thesis, Inst. of El. Measurement and Measurement Signal Processing, Graz Univ. of Technology, 2010.

[8] V. Ramanathan and A. Pinz, "Active object categorization on a humanoid robot," in *Proc. VISAPP*, pp. 235–241, 2011.

[9] R. J. Firby, R. E. Kahn, P. N. Prokopowicz, and M. J. Swain, "Collecting trash: A test of purposive vision," in *Proc. of the Workshop on Vision for Robots*, pp. 18–27, 1995.

[10] M. Bollmann, R. Hoischen, M. Jesikiewicz, C. Justkowski, and B. Mertsching, "Playing domino: A case study for an active vision system," in *Proceedings of the First International Conference on Computer Vision Systems*, ICVS '99, (London, UK, UK), pp. 392–411, Springer-Verlag, 1999.

[11] L. Athanasia, M. Ahmed, M. Naresh, J. Pieter, and H. Victor, "Integration of the humanoid robot Nao inside a smart home: A case study," tech. rep., School of Science and Technology, Oerebro University, Sweden, 2010.

[12] H. Mellmann and Y. Xu, "Adaptive motion control with visual feedback for a humanoid robot," in *the Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Taipei), 2010.

[13] O. Kroemer, R. Detry, J. Piater, and J. Peters, "Grasping with vision descriptors and motor primitives," in *Informatics in Control, Automation and Robotics* (J. A. Cetto, J.-L. Ferrier, and J. Filipe, eds.), vol. 89 of *Lecture Notes in Electrical Engineering*, pp. 211–223, Springer Berlin Heidelberg, 2011.

[14] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *Int. J. Rob. Res.*, vol. 27, pp. 157–173, February 2008.

[15] B. Siciliano and O. Khatib, eds., *Springer Handbook of Robotics*. Springer, 2008.

[16] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *Robotics Automation Magazine, IEEE*, vol. 13, no. 4, pp. 82 –90, 2006.

[17] F. Chaumette and S. Hutchinson, "Visual servo control. ii. advanced approaches [tutorial]," *Robotics Automation Magazine, IEEE*, vol. 14, no. 1, pp. 109 –118, 2007.

[18] P. I. Corke, "Visual control of robot manipulators – a review," in *Visual Servoing*, pp. 1–31, World Scientific, 1994.

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

[20] J.-Y. Bouguet, "Camera calibration toolbox for matlab," 2011 (accessed August 26, 2011). available at `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[21] J. Su, Y. Zhang, and Z. Luo, "Online estimation of image Jacobian matrix for uncalibrated dynamic hand-eye coordination," *Int. J. Syst., Control Commun.*, vol. 1, pp. 31–52, July 2008.

[22] K. Deguchi, “A direct interpretation of dynamic images with camera and object motions for vision guided robot control,” *Int. J. Comput. Vision*, vol. 37, pp. 7–20, June 2000.