Musical NAO

Instrument Playing Humanoid Using Monocular Vision

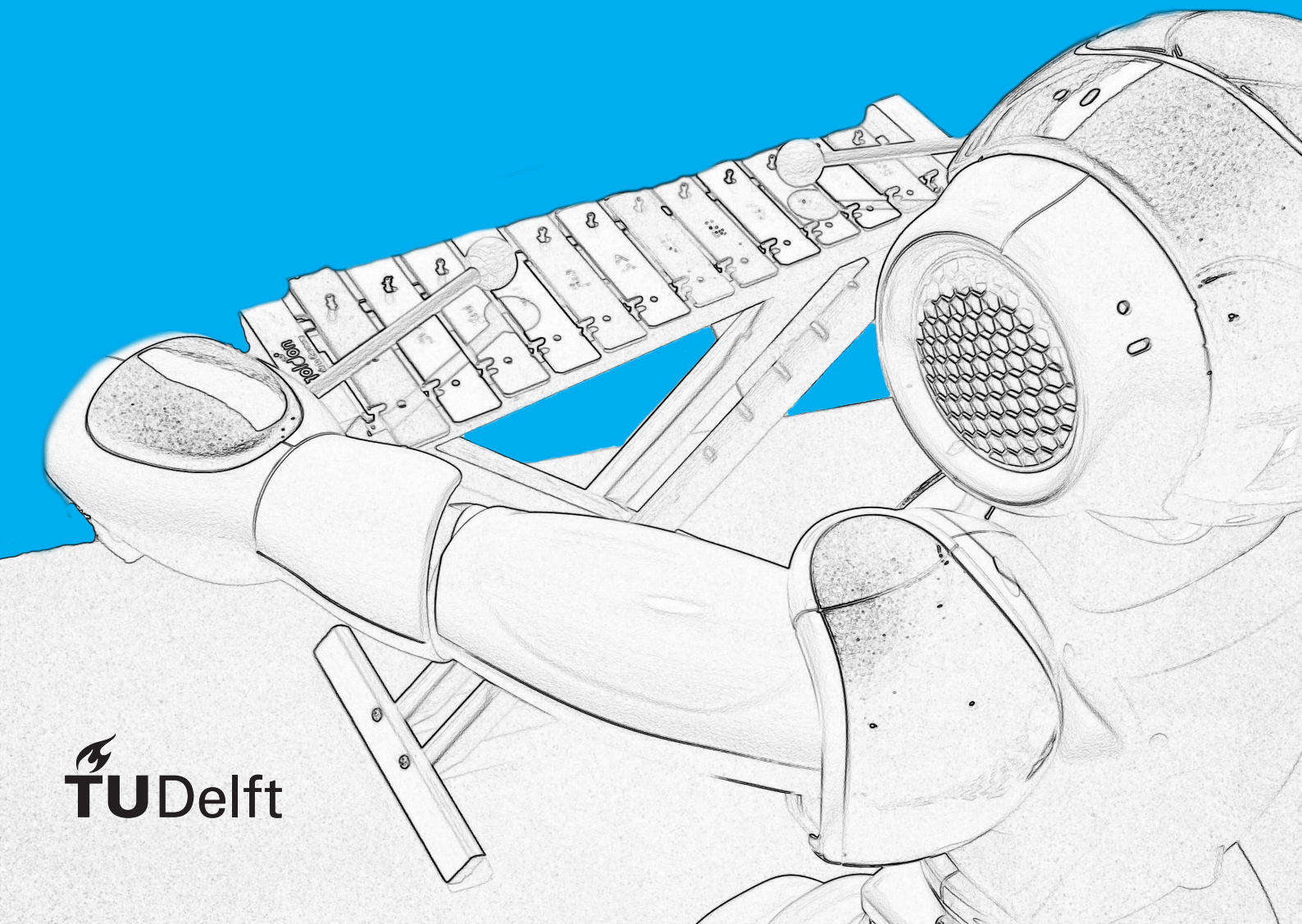
Ioannis Papakonstantinopoulos

Technische Universiteit Delft

TU Delft

Musical NAO

Ioannis Papakonstantinopoulos



TU Delft

Musical NAO

by

Ioannis
Papakonstantinopoulos

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday November 28, 2018 at 9:00 AM.

Student number:	4631838	
Project duration:	January 1, 2018 – November 28, 2018	
Thesis committee:	Dr. K. Hindriks,	TU Delft, supervisor
	Dr. J. Broekens,	TU Delft
	Dr. J. A. Mora,	TU Delft

This thesis is confidential and cannot be made public until November 30, 2018.

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

TU Delft

Preface

This thesis is a culmination of my two years as a computer science student at the Delft University of Technology. Throughout my ten-month work, in Interactive Intelligence group, I developed my knowledge about robotics. This work would not have been possible without the advice and support of many people. First and foremost, I would like to express my sincere gratitude to my supervisor Dr. Koen Hindriks, for keeping me motivated in a thesis topic related to robotics and specifically humanoids. Without his guidance and feedback, I would not have gained as much insight and my research would not have taken shape. I am deeply grateful to him also for getting me in touch with Museum Speelklok in Utrecht where my work was used. That was an extra stimulus which kept me motivated until the completion of my work. Furthermore, I would like to thank Ruud de Jong for helping me out on managing resources for the experiments and for his patience on encountering technical issues with the robots.

All gratitude is also uttered to all participants who were willing to join the experiment. Your time and feedback are really appreciated and useful to make the study happened. I thank my fellow student Maria Gatou for providing me with unfailing support and continuous encouragement throughout my two years of study and through the process of researching and writing this thesis. I would like to say thank you to my friends Alexandros Kouris and Ioannis Tsenempis for some nice brainstorming discussions about robotics, which helped me to take decisions during my research.

Lastly, I would like to thank my family: my parents and to my brothers for supporting me spiritually throughout my years of study and through the process of researching and writing this thesis.

Ioannis
Papakonstantinopoulos
Delft, January 2013

Abstract

Humanoid robots are becoming part of modern societies mainly involving the education and entertainment fields. Therefore, providing techniques that enable a humanoid to autonomously play instruments like the metallophone we enrich its usefulness.

In this work we present an approach for making humanoid robot NAO play the metallophone. Specifically, we provide an application which can be used by everyone owned a NAO robot. In our approach, no markers are necessary to be attached on the robot, making the procedure of playing music, seems natural. We propose an efficient real-time approach for recognizing and estimate the metallophone in 3D space using a single camera. Our vision approach performs well in camera displacements, illumination and aspect changes.

For accurate playing, the robot automatically validates the computed IK configurations based on a vision-based robot control approach and adapts its arm configurations if it is necessary. To achieve this, we rely on the estimated pose of the musical instrument and the detection of the beaters in 2D, based on natural features and then we apply inverse kinematics. Concerning kinematics, we use singularity-robust approach by pre-computing forward kinematics which are used to efficiently solve inverse kinematics (IK) problems.

In our approach we limited on moving only NAO's arms for playing the metallophone. Therefore, we developed a human-robot interaction phase in which the robot uses its perception about the position of the instrument in 3D space and it guides a human in order to place it in a reachable for the robot position. Moreover, in our system we provide a simple way to add songs, which NAO can immediately play.

Contents

List of Figures

List of Tables

1 Introduction

Humanoid robots are used in a range of application domains like domestic, entertainment, health care and education [6]. Humanoids are getting more and more popular in these applications because they can interact and cooperate with humans.

The last few years there has been a raise of interest in Humanoid robots in the education and entertainment field. Without any doubt, it is very interesting and fun, for people, to see a machine act like humans, but it is still a very challenging task to make a robot behave as natural as possible, since most of the time these robots are pre-programmed to perform particular tasks and interactions, as a consequence the robots will start repeat themselves.

Many humanoid robots have been used for playing musical instruments [2]. For instance, Hubo humanoid, Honda's Asimo, Toyota's musical robots have been used in playing different kind of musical instruments [32]. In addition, Chida developed a robotic flutist [8], Weinberg presented the musical playing robot Shimon [52] and Mizumoto presented the HRP-2 robot [40]. As we mentioned before, entertainment is not the only field that humanoid robots are popular, they could also be used in teaching musical instruments to children or in the treatment of people with special needs [21] [48].

In this work, we present an approach for making the humanoid robot NAO, be able to play a musical instrument like the metallophone, autonomously, after a preparation stage in which a human interacts with the robot. NAO robot is commercially available and it is designed for a wide range of multimodal expressive gestures and behaviors, therefore it is a good platform to develop our software on. NAO has plenty of sensors for perceiving the environment around of it, as well as it integrates speakers and four kinematic chains, two legs and two arms, which make it capable of walking, grasping things and talking. The metallophone is a musical instrument which produce sounds by hitting it using two beaters. Since NAO is able to hold the two beaters as well as to move its arms, it is feasible to make it play the metallophone.

In our approach, NAO uses only its arms to play the metallophone (Figure 1.1), and its positioning behind the musical instrument happens after an interaction phase with a human. More specifically, in the preparation phase, a human must give the beaters to the robot's hands and place the metallophone in the appropriate position according to the robot's instructions. We aim to develop a system which will conveniently run on any computer and on any NAO robot, so that it will be possible for anyone,who has a NAO, to use it. The work presented in this thesis, aims to build a system which will be able to work robustly in different NAO robots and environments so that it can be used by people without any expertise in robotics for education or entertaining purposes.

1.1. Problem Definition

The metallophone (Figure 1.2) is a musical instrument consisting of tuned metal bars which are struck with a mallet (beater) to make sound. For this project, we use a metallophone, consisting of 12 metal bars, and since NAO has two arms of a particular length, the size of the metallophone must meets the requirement of this length. Therefore, we chose a metallophone which the robot is able to reach all of the metal bars.

The robot must hit the appropriate metal bar, on the musical instrument, quite precisely with a mallet in order to make it produce the desired musical note. For human music players, this task seems just a simple movement of an arm, however, a NAO robot is able to achieve this, after completing a series of complex

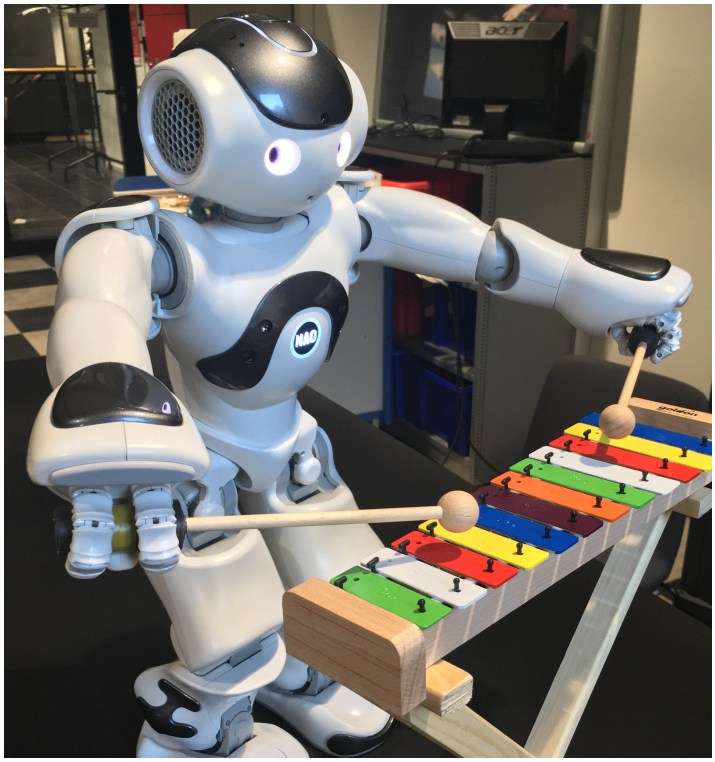


Figure 1.1: Humanoid NAO playing the metallophone

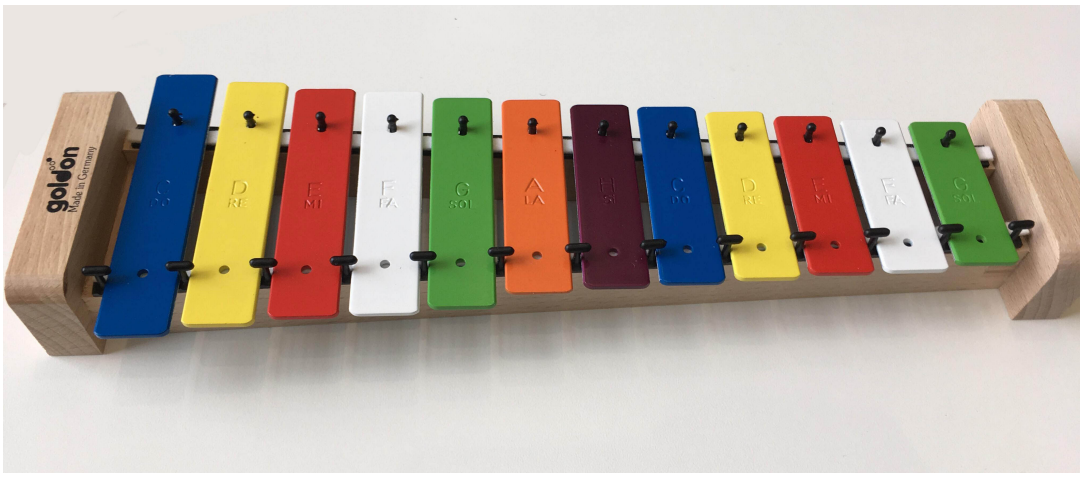


Figure 1.2: The metallophone used for this project

tasks. The first challenge for the robot is to hit accurately the metal bars in the right spot. Consequently, our kinematic implementation must have precision of millimeters. However even if we achieve that, due to variances in the actuators of the robot, the hitting procedure is not going to be completed precisely. In addition, before the robot decide about its actions, there must be a perception phase. In this phase the robot must estimate precisely the exact position of the musical instrument in 3D space and then it decides where to move its arms in order to produce the appropriate sound by hitting the metallophone.

1.1.1. Vision-Based position estimation of the musical instrument

Starting with the perception of the metallophone's position, in relation to the robot, the challenging task that we need to complete is to make the robot estimate the position and orientation of the metallophone. It is necessary for the robot to detect the position of the metallophone in 3D space in order to decide about its actions. Moreover, we want to develop a robust approach in different light conditions. The robustness

against light conditions it is always a challenging task for vision approaches, since cameras are sensitive in illumination changes.

In general, stereo vision approaches are used to estimate accurately the position of an object in the 3D space. However, we are not able to apply a stereo vision algorithm in NAO robot, because it does not provide stereoscopic camera pair. Thus, we approach the problem of position estimation of an object in the 3D space using a model-based approach, based on single camera. In this approach, which is explained in more detail in Chapter 2, some prior knowledge is necessary about the 3D dimensions of the object. After the robot estimates the position and orientation of the metallophone with respect to its body, it is able to know if it can hit the metallophone using kinematics.

1.1.2. Inverse Kinematics for playing the metallophone

Since the robot "knows" the 3D position of the instrument in relation to its body, hitting the metal bars, seems to be a simple inverse kinematic problem. However, we need to calculate the inverse kinematics for the kinematics chains of both arms holding the beaters which length is approximately 16cm.

1.1.3. Low precision in NAO's actuators

The precision of NAO's motors is not the appropriate to complete the hitting task successfully, when the robot knows where to move its arms. There is always an error in the position of the end-effector (end of beater) which sometimes makes the robot hit the wrong metal bar. This problem becomes worse when we integrate the same software in different robots. This issue is the main challenge of this work and we aim to solve it without using extra markers attached on the robot. Solving this problem without using markers makes the procedure more demanding but it seems more natural.

1.2. Research Objective

In order to make NAO play the metallophone, we use the robot's on-board camera for detecting the position of the instrument in relation with the robot's position. Then, using robot's arms we make the robot hit the musical instrument and produce the sound. At the end the robot automatically validates the computed IK configurations, using vision for detecting the beater's position.

The work presented in this thesis is inspired by Daniel et al.'s [13]. They presented an approach for making NAO play the metallophone by deploying self-calibration on the robot's kinematic model, a particle filter framework for model-based pose estimation of the instrument and the beaters and an auditory feedback method for determining the accuracy of the beating motion. During the self-calibration on the robot's kinematic model, Daniel et al. [13] use markers attached on robot's arms, which make the procedure seems less natural. Moreover, they do not refer to the robustness of their vision approach.

In our work, we divide the pose estimation problem in: i)2D image matching based on reference images (key-frames) of the object's view and ii)the position estimation of the object in 3D space. To achieve this, we use some prior knowledge of the 3D model of the object (metallophone) and some offline work for creating the key-frames. A reference-image contains the metallophone in a particular viewpoint and the coordinates of five 2D points which are used for the 3D pose estimation. In the online part, we use a feature extraction algorithm to find points, on the input frames and key-frames, with descriptive features. Then we use a feature matching algorithm to match similar points in both images and using these matches we calculate the homographies between the two images. The homographies allow us to map key-frames with input frames and according to metallophone's view in the current frame we use key-frames to extract the 3D information.

Given the pose estimation of the instrument in the 3D space, we use inverse kinematics to calculate the appropriate configurations of the arms in order to execute the beating motion. We pre-calculate the inverse kinematics by deploying a reachability analysis for both kinematic chains (arms) and we store the pre-computed forward kinematics solutions.

Finally, we use a vision-based control technique in order to precisely place the end-effector above the appropriate metal bar. In this part, the robot places the beaters above two particular metal bars. Then, the robot uses its camera and it detects the actual beaters position by using a vision approach. Finally, if the actual position of the beaters is not the same as the desired, the robot moves its arms appropriately to correct this error. In our approach, only natural features are used for the adaptation of the kinematic configurations.

1.2.1. Research Questions

Pose Estimation and Kinematics
The main goal of this project is to make NAO observe the position and orientation of the metallophone and hit the metal bars. However even if the robot holds the beaters and the metallophone is placed in front of it, the robot must find a way to hit the metal bars precisely in order to produce the appropriate sound. Hence, the first research question that arose is:

- RQ1: *How can we make the NAO autonomously and reliably hit the right note (metal bar)?*

At first we investigated methods for 3D pose estimation using a single camera, as well as we tried to find ways to use these methods on NAO. As far as the pose estimation of musical instrument is concerned, we wanted to achieve it without using any extra markers on the instrument, and this made the research more challenging considering the robustness of the system. For the kinematic part, we had to calculate the kinematics of NAO's arms from scratch, since the beaters had to be added in the kinematic chain, as well as the numerical solution provided by the manufacturer of the robot, is prone to singularities. At the end we have a vision system which returns the 3D position and orientation of each metal bar in relation to robot's body and a kinematic system with which we can move the robot's arms (end-effector) above the appropriate metal bar.

Model-based position estimation
Since there are many different methods to use for estimating the position and orientation of an object, we could complete this task in many different ways. However, as we mentioned before, we aim to develop a vision system which works robustly in different environments. Hence, we need to investigate if our approach is mainly robust against illumination changes. With the following research question, we wanted to focus on the robustness of the chosen vision approach.

- RQ2: *How can we make the pose estimation task robust to different light conditions?*

Approaching this task with an image matching approach based on key-frames we expect a robust outcome. However, the drawback of this approach is that some prior work is required in order to create the key-frames with different illumination.

Vision-Based Control (Visual-Servoing)
In theory, the approach described above seems to be enough to make the robot able to play the metallophone. However, in practice we faced some expected difficulties. The first one had to do with the kinematics part. Due to the low precision of NAO's motors, there is a variance at the movements of the actuators (Motors are precise up to a limit). Additionally, this variance becomes even worse when we integrate our software in different NAO robot. Thus, an arising question is:

- RQ3: *How can we correct the variance in robot actuators?*

To resolve this issue we investigated methods around of the vision-based robot control. In general, a vision-based control technique uses feedback information, extracted from the camera of the robot, to control the motion of a robot. This task is very challenging since we need to achieve accuracy of millimeters by cooperating a vision and a kinematic system. Moreover, we need to find a reliable way to detect the end-effector (head of the beater) using a vision approach.

Human-Robot Interaction
In our approach, there is a preparation phase in which a user has to help the robot by placing the metallophone in an appropriate position for it and then the user has to put the beaters in the robot's hands. Since the robot is able to walk, it could place itself behind the metallophone using its legs, as well as it could grasp the beaters on its own, however, in this work we do not focus on how to make NAO walk in such an accurate way that it could place itself in good position behind the metallophone. Thus, the preparation phase, it is necessary to be accomplished with human assistance.

The research question that arose from the above is:

- RQ4: *How can the robot help the user place the metallophone in an appropriate position?*

To answer this question, we created a human-robot interaction phase in which the robot guides a non-expertise person (i.e. without expert knowledge of the robot's workings) to help the robot set up the musical instrument in order to be able to play it. More specifically, the robot estimates the position of the metallophone and then it uses auditory feedback to guide the user for the appropriate placement of the instrument. Apart from the auditory feedback, we implemented also a section in which the user receives feedback from the computer's screen, which shows what the robot sees by its camera.

1.3. Contribution

Our main contribution is the development of software for making any different NAO (V5, V6) able to play the metallophone in different environments. In contrast with previous work [13] we do not use any checkerboard-markers attached in robot's arms since it makes the procedure seems less natural. Daniel et al. [13] used checkerboard-markers for calibrating the kinematic parameters and a particle filter framework for estimating the instrument's pose. In our approach, we use a visual-servoing technique, in order to adjust errors in kinematic configurations, by detecting the beaters' actual position. We achieve a robust outcome, without using any markers, making our software more accessible to simple users and the same time we achieve a more natural result. In addition, for the 3D pose estimation of the metallophone we use an approach based on multiple key-frames expecting to have a more robust outcome in different light conditions. Finally, we aim to make this software easy usable by simple users in order to use it for education or entertainment purposes.

1.4. Outline

The organization of the thesis is as follows. In Chapter 2, we summarize literature related to our study and present key ideas we adopt in our work. In Chapter 3 we describe the approach we followed in order to develop our system. In Chapter 4, we describe the experiments that we deployed for evaluating our approach and we present our results. In Chapter 5 we discuss the findings and limitations we observed in the results, as well as we propose related future work.

2
Background

In this chapter we present the background on which our research is based on and we briefly summarize some instrument playing humanoid robots. Especially, we describe the basic theory behind i)the 3D pose estimation based on monocular vision, ii)the robot kinematics and the iii)vision-based control. In addition, we present an overview of other researches in which these methods are applied on NAO.

2.1. Instrument Playing Humanoid NAO

As we mentioned in the introduction, a lot of research has been conducted for creating humanoid robots that can play a musical instrument. However, in most cases these robots are not available for commercial use. Daniel et al. [13] presented a novel approach to enable the humanoid robot NAO to play the metallophone. Specifically, they developed a technique for self-calibration of the robot kinematic model, a particle filter framework for model-based pose estimation of the instrument and the beaters. Moreover, they presented an auditory feedback method for determining the accuracy of the beating motion. Their robot was able to play the metallophone, however they did not give details about the conditions under which the robot is able to detect the position of the metallophone and play it accurately. In addition, they did not mention if their application is ready to use by users without experience with robots. We aim to create an application, for the commercially available humanoid robot NAO, which will be easy for users without experience in robotics to use it. More specifically, we aim to develop a user friendly application, by creating a human-robot interaction interface for the phases that the robot needs the human assistance as for example grasping the beaters and place the metallophone in a appropriate for the robot position.

2.2. Model-Based Object Pose Estimation

Tracking an object in a video sequence means continuously identifying its location when either the object or the camera are moving. There are a variety of approaches, depending on the camera, the type of object, the degrees of freedom of the object and the target application. The term pose refers to the position and orientation of a rigid object in space which is defined by three components of translation and three components of rotation. Because of these 6 components, a 3D pose is also called six degrees of freedom (6-DOF) pose.

People use their eyes to observe the position and orientation of the object and then they move their arms in an appropriate way which allows them to grasp or hit that object. Similarly, we want to make the robot perceive the metallophone's position through its camera in order to play it.

In our case we need to track the musical instrument in 3D space and extract information about the pose of it in relation with the robot. One of the most appropriate ways to estimate accurately the position and orientation of an object in 3D space is to use a stereo vision approach. 3D depth perception can be achieved by using two cameras, similar to how we as humans use our two eyes. Then, using triangulation we can perceive the third dimension. Stereo vision allows people to see an object as solid in three spatial dimensions x, y and z and the perception of the depth dimension makes stereo vision so rich and special. However, we are not able to develop a stereo vision algorithm in NAO robot, since it does not provide stereoscopic camera pair. Thus, we solve the problem of the 3D pose estimation by developing a model-based 3D tracking approach using a single camera [36].

2.2.1. Mathematical Model

It is not possible to extract 3D information just from a 2D image. However, if the dimensions of the depicted object in the image are known, there are methods for estimating the pose of an object just from a single image. In this section we present the basic knowledge needed for understanding our approach for observing the metallophone in 3D space.

The Pinhole Camera Model

The pinhole camera model describes the mathematical relationship between the coordinates of a point in three-dimensional space and its corresponding projection onto the 2D image plane of an ideal pinhole camera.

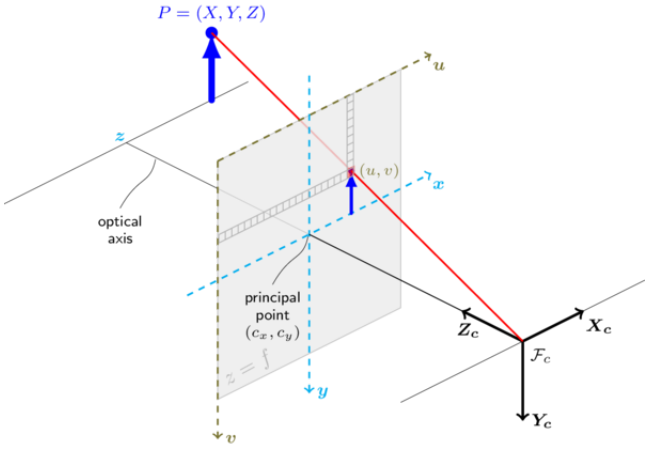


Figure 2.1: Pinhole Camera Model [4]

Perspective Projections

Mathematically, image formation can be defined as the projection from the 3D space to the image plane as it is illustrated in Figure 2.1. The coordinates of a 3D point P_W = [X,Y,Z]^T expressed in Euclidean world coordinate system and the corresponding 2D point x = [u, v]^T in the image are related by the equation:

$$\tilde{x} = P \times \tilde{P}_W \tag{2.1}$$

where = [u, v, 1]^T and P_W = [X, Y, Z, 1]^T are the homogeneous coordinates of points x and P_W, and P is a 3 × 4 projection matrix. Projection matrix is usually taken to be a perspective projection matrix since it describes the behavior of a simple camera. A perspective projection matrix can be decomposed as follows:

$$P = K \times [R|t] \tag{2.2}$$

The 3 × 3 matrix K is called the calibration matrix and describes the camera intrinsics. We also refer to this matrix as camera matrix or calibration matrix, since it is derived from the calibration of the camera. Moreover, in this project we treat this matrix as an upper triangular.

[R | t] matrix is the 3×4 external parameters matrix, and corresponds to the Euclidean transformation from a world coordinate system to the camera coordinate system. More specifically, R represents a 3 × 3 rotation matrix, and t is a 3 × 1 translation vector.

The Camera Calibration Matrix

The camera calibration matrix contains the intrinsic camera parameters, also called internal parameters. There are several ways to write the upper-triangular form of K. In this project we use the representation described below:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

which uses independent focal lengths f_x and f_y for the sensor x and y dimensions. The entry s is referred as the skew and it is non-zero only if the sensor axes are not perpendicular, which is rare in modern cameras. The parameters c_x and c_y denotes the optical center expressed in pixel coordinates also called the principal point. Thus, the center of the image is the principal point.

Estimating the Camera Calibration Matrix
The internal parameters of the NAO's camera were not available and we had to estimate them by deploying an offline camera calibration [54].

At this step, using the lower camera of NAO, we captured several images with a checkerboard and then we used the function calibrateCamera() of the OpenCV library which returns the camera matrix and the distortion coefficients which are necessary for the 3D tracking phase. Distortion coefficients indicate the distortion of the lens. The distortions that we need to eliminate are radial distortion and tangential distortion. Radial distortion makes the straight lines to appear curved and usually its effect is more as we move away from the center of image.

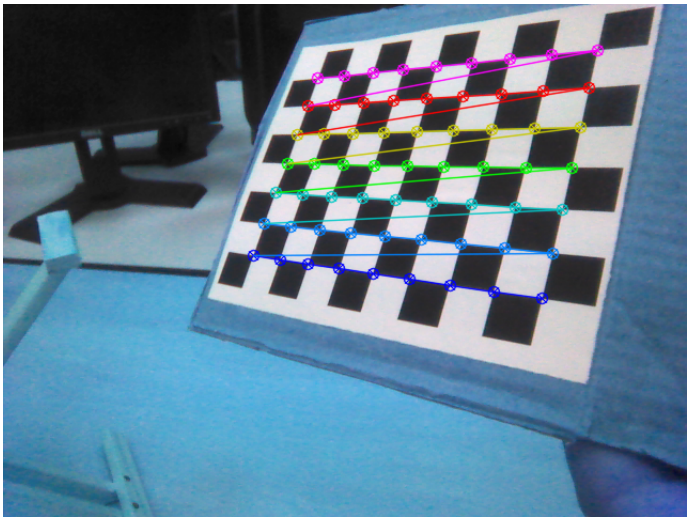


Figure 2.2: Calibration procedure of NAO's camera

The External Parameters Matrix
The 3 × 4 external parameters [R | t] matrix defines the position (x, y, z) and the orientation (α, β, γ) of the camera and we will often refer to it as the camera pose. Since we have the matrix K by the camera calibration, we just need to estimate the R and t, or, equivalently, the target object's position and orientation with respect to the camera.

More specifically, we need to define the Euclidean transformation from a world coordinate system to the camera coordinate system. Equation 2.1, shows how we can project a 3D point onto the 2D image plane. From this relation, we can easily recover the expression of the camera center, or optical center C in the world coordinate system. It must satisfy:

$$0 = [R|t] \times C \implies 0 = R \times C + t \tag{2.4}$$

which implies:

$$C = R^{-1} \times t = R^T \times t \tag{2.5}$$

Equation 2.5 defines the camera position in relation to the object.

2.2.2. Estimating the External Parameters Matrix

After retrieving the calibration matrix, the External Parameters matrix [R|t] can be calculated. The External Parameters matrix, allows us to move from the 3D world points to 2D image plane points using the equation 2.1.

There are several approaches to estimate the external parameters, from images taken by a calibrated camera. The estimation of these parameters is achieved by using some correspondences between 3D points in

the world coordinate system and their projections in the image plane. Estimating the external parameters when the internal parameters are known is also referred to as the pose estimation problem.

The pose estimation problem is solved by using n correspondences between 3D points P_W, and their projections x onto the image plane. Then the perspective projection matrix P can be used to solve the equation 2.1 up to a scale factor.

At this point it is necessary to define the number of correspondences which are required for the pose estimation. For n = 3 known correspondences, there are four possible solutions. For n = 4 or n = 5 correspondences, [20] shows there are at least two solutions, but when the points are coplanar and there is no triplets of collinear points, the solution is unique for $n \geqslant 4$. Finally, there is a unique solution for $n \geqslant 6$.

Direct Linear Transformation (DLT)

Direct Linear Transformation can be used to estimate matrix P of equation 2.1 by solving a linear system when some correspondences between 3D and 2D points are available [18] [28]. Each correspondence between P_w and x gives rise to two linearly independent equations in the entries P_{ij} of P, that is:

$$u_i = \frac{P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \tag{2.6}$$

$$v_i = \frac{P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} \tag{2.7}$$

In order to compute the 12 (or 11)unknowns in P, at least six correspondences between 3D and 2D locations must be known. Once the entries in P have been recovered, we have both the intrinsic calibration matrix K and the rigid transformation [R | t] by observing Equation 2.2. Since we have already calculated the internal parameters (Camera calibration), we can extract the orientation and position [R | t] (pose) up to a scale factor. In cases that the camera is already calibrated, we can perform pose estimation using as few as three points [20] [25].

In reality there is noise in pixels' position $\tilde{x}$ = [u, v], and it is necessary for the camera parameters estimated by the DLT algorithm to be refined by iterative optimization of the non-linear reprojection error.

Iterative methods

As shown in the equation 2.2, if we knew the right pose ([R|t]), we could predict the 2D locations of the corresponding 3D points on the image by projecting the 3D points onto the 2D image. Specifically, if matrix R and vector t were known, we could find the point p on the image for every 3D point P.

In our case we have some corresponding points, between the 2D image and 3D object, and we want to find the pose ([R|t]). Looking at the distance between the projected 3D points and 2D points, in cases where the estimated pose is perfect, the 3D points projected onto the image plane line up almost perfectly with the 2D corresponding points. When the pose estimation is incorrect, we can calculate a re-projection error measure — the sum of squared distances between the projected 3D points and 2D points.

To conclude, an approximate estimate pose [R|t] can be found using the DLT solution. A naive way to improve the DLT solution would be to randomly make small changes into the pose [R|t] and check if the reprojection error decreases. If it does, we can accept the new estimate of the pose. This method, could find better estimates, however it will be very slow. There are principled ways to iteratively change the values of R and t so that the re-projection error decreases.

The iterative approaches consider pose estimation as a nonlinear least squares problem by iteratively minimizing specific error function. This error function usually has a geometrical meaning.

Levenberg–Marquardt algorithm

In mathematics and computing, the Levenberg–Marquardt algorithm [41] (LMA or just LM), also known as the damped least-squares (DLS) method, is used to solve non-linear least squares problems. These minimization problems arise especially in least squares curve fitting.

The LMA is used in many software applications for solving generic curve-fitting problems. However, as with many fitting algorithms, the LMA finds only a local minimum, which is not necessarily the global minimum. The LMA interpolates between the Gauss–Newton algorithm (GNA) and the method of gradient descent. The LMA is more robust than the GNA, which means that in many cases it finds a solution even if it starts very far off the final minimum. For well-behaved functions and reasonable starting parameters, the LMA tends to be a bit slower than the GNA. LMA can also be viewed as Gauss–Newton using a trust region approach.

2.2.3. The Perspective-n-Point (PnP) Problem

Since we can have n correspondences between 2D and 3D points, we can track the pose of the object in 3D space by solving the Perspective-n-Point (PnP) problem. Solving the PnP problem allows us to estimate the pose of a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the image. In addition, the DLT method aims at estimating all 11 parameters of the projection matrix, thus it might also be applied to pose estimation, as we have mentioned above. However, when the internal parameters have been estimated separately, a more satisfactory approach is to explicitly use this knowledge.

The 3 point pose problem, that is the estimation of the camera pose from 3 point correspondences also known as the perspective-3-point problem (P3P) has up to 4 real solutions in most cases. For 4 more points, the solution is in general unique whether or not they are coplanar, as long as they are not aligned. However, in unfavorable cases there can be infinitely many solutions no matter how many points are supplied. In these cases, some points are hidden behind others, and then there are infinitely many solutions. These singular cases are explained in detail in Haralick et al.'s [25] work.

There are different approaches in literature, that solve the perspective-n-point problem [20] [47]. They usually attempt to first estimate the distances between the camera center C and the 3D points P_{Wi}, from constraints given by the triangles formed by the camera center and the 3D points.

Another popular method to solve the pose estimation problem for $n \geqslant 4$ non-coplanar points is POSIT [14]. This method is a combination of two algorithms. The first one, POS (Pose from Orthography and Scaling), approximates the perspective projection with a scaled orthographic projection and finds the rotation matrix and the translation vector of the object by solving a linear system and the other uses in its iteration loop the approximate pose found by the first in order to compute better scaled orthographic projections of the feature points, then applies POS to these projections instead of the original image projections. This method is quite simple to implement [5], but is relatively sensitive to noise.

In this project, we use a DLT algorithm followed by an iterative optimization of the non-linear re-projection error, for estimating the pose of the object. More specifically, we use the iterative version of the solvePnP function, provided by the openCV library. It is essentially a DLT solution followed by Levenberg-Marquardt optimization. The function finds such a pose that minimizes re-projection error, that is the sum of squared distances between the observed projections image points and the projected object points .

2.2.4. Monocular 6-DOF Pose Estimation

In the previous subsection we referred to the fundamentals of the 6-DOF pose estimation based on monocular vision. Pose estimation based on an image plane requires additional knowledge of the object and it is a problem which usually researchers in robotics and augmented reality areas have employed a number of different approaches to solve it. A simple way to retrieve the pose of an object, is by using fiducial markers. In this method, artificial markers are attached to the object or environment as camera targets and it provides an easy and robust solution for real-time pose estimation. However, in practice the markers make the procedure of 3D tracking to look less natural. Therefore, researchers focused on tracking using natural features, in order to make tracking procedure more natural. There are many methods presented on this subject and we can categorized them in: edge-based, optical flow-based, template-based, and keypoint-based [36]. We are going to analyze further the edge-based and keypoint-based methods.

The edge features are computationally efficient, and relatively easy to implement. In addition, edges are usually computed by image gradients thus it is moderately invariant to illumination and viewpoint. The keypoint features are also capable of being invariant to illumination, orientation, scale, and partially viewpoint. However, keypoint-based methods are computationally inefficient in comparison with edge features.

Edge-Based Methods

In edge-based methods, a 3D CAD model is usually employed to estimate the full pose using a monocular camera. RAPiD (Real-time Attitude and Position Determination) is one of the first marker-less 3D model-based real-time tracking system and was established by Harris [26].

In this method, the object is tracked by comparing the projected CAD model edges with the edges detected in a gray-scale image. To project the model close to the real object, the system use the previous pose estimate as a priori. Also this algorithm uses motion parameters which are subsequently estimated between frames. Drummond and Cipolla [16] enhanced the robustness by using the iterative re-weighted least squares with a M-estimator.

Keypoint-Based Methods
In keypoint-based methods, a sparse 3D metric model is used. The keypoint models are the a priori knowledge and they are built offline. In addition, a set of images containing a view of an object from different viewpoints is used and the non-linear optimization algorithm, such as Levenberg-Marquardt, return a refined 3D model of keypoints. In this way the 3D coordinates of each keypoint are maintained and the pose estimation is performed by using the correspondences between the 3D points of the model and the 2D keypoints in an input image. Using this model, Gordon and Lowe [23] proposed an augmented reality system that calculates pose with scale invariant features [37]. Vacchetti et al. [50] used standard corner features to match the current image and the reference frames, so called keyframes. Unlike the efforts using non-linear optimization, they obtained 3D coordinates of 2D corner points by back-projecting them onto the object CAD model.

Various combined approaches of edge and the keypoint methods have been presented [35], [46], since the edge and the keypoint methods are complementary to each other. Vacchetti et al. [51] presented a more robust and jitter free approach by incorporating the edge-based method with their corner point-based method. As part of the edge-based tracking, they used multiple hypotheses to handle erroneous edge correspondence, however it is equivalent to the nearest hypothesis of RAPiD-like approaches. Similarly, Rosten and Drummond [49] combined corner points with lines, but they only used corner points to estimate motion parameters between frames.

2.2.5. Overview of model-based pose estimation from natural features implementations on NAO robot

Daniel et al. [13] use a model-based technique within a particle filter framework. The main idea, which is based on Choi and Christensen's [9], Michel et al.'s [39], Gonzalez-Aguirre et al.'s [22], is, given a hypothesis about an object's pose, one can project the contour of the object's model into the camera image and compare them to actually observed contour. In the aforementioned approach, they estimate the likelihood of the pose hypothesis. They use a tracking method based on natural features. In general, edge-based methods have a fairly low computational complexity because they only work for a small fraction of the image pixels. However, they can become confused in the presence of texture or of a cluttered background.

Moreover, matching 3D models with 2D scene images is commonly used for solving the self-localization problem in mobile robots which provide single camera vision [19], [34]. Moughlbay et al. [42] used 3D visual feedback data and Model Based Tracking (MBT) techniques to execute, in real-time and closed loop, many tasks on the humanoid mobile robot NAO in a semi structured environment. More specifically, the robot solves the self-localization task by searching the corresponding set of models in its environment. After detecting a known object, the robot uses the Model Based Tracking technique to estimate the object's position and orientation.

In addition, real-time pose estimation and object tracking of rigid objects, based on image matching with natural landmarks has been presented by other researchers. Papanikolopoulos and Kanade [44] proposed algorithms for robotic real-time visual tracking using sum-of-squared differences (SSD) optical flow. Choi and Christensen [9] proposed a method that combines scale invariant features matching with optical flow based tracking, KLT tracker, for real-time 3D pose tracking. More specifically, they used a scale invariant features extraction method to estimate the initial pose (using some prior knowledge) and then the KLT tracker to track the object. KLT tracker calculate 3D pose consecutively from initially estimated pose with cheap computational cost, however, this tracker shows poor performance in significant change of scene, illumination, and occlusions.

2.2.6. NAO's Vision Hardware

NAO provides two identical video cameras which are located in the front of its head as you can see in Figure 2.3. In this project only the lower camera is used, since the metallophone will be always placed in the front and lower area of the robot. As a result, the robot is not able to capture the whole object using the upper camera.

These cameras can provide a 1280x960 resolution at 30 frames per second, however the number of frames per second are less when we request images remotely using the getImageRemote function. Ideally, we would like to get enough frames per second so that the robot could process more data, but we want to work in a wireless environment for making the instrument playing looks more natural. Hence, at this point we need to decide the resolution of the images that are best for our project, since we are using a wireless communication. After tests, we concluded that our vision approach operates well in 640x480 pixels images and 2.5 frames per seconds are enough for our purpose. Using images with lower resolution, our approach failed to detect the

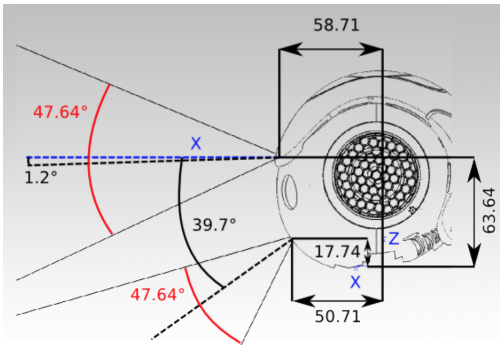


Figure 2.3: Cameras' position on NAO

object, and using higher resolution images we did not observe better performance in our vision approach. The table 2.1 illustrates the number of frames with the corresponding network communication.

Table 2.1: Frames per second for different network communication

Resolution in pixels	Gb Ethernet	100Mb Ethernet	WiFi
40x30 (QQQQVGA)	30fps	30fps	30fps
80x60 (QQQVGA)	30fps	30fps	30fps
160x120 (QQVGA)	30fps	30fps	30fps
320x240 (QVGA)	30fps	30fps	11fps
640x480 (VGA)	30fps	12fps	2.5fps
1280x960 (4VGA)	10fps	3fps	0.5fps

Image Retrieving

For retrieving an image from the camera of the robot, we create a vision module which sends a request via the broker to subscribe to ALVideoDevice with parameters such as the size of the image in pixels, and we call the getimageRemote() function. After that the image is provided by the ALVideoDevice which manages the video source. The ALVideoDevice first opens the camera device, communicating through I2C bus, then it launches the V4L2 driver in streaming mode. The V4L2 driver will create a circular buffer of n elements to grab the video stream. After that, the ALVideoDevice receives subscriptions from Vision Modules by creating for each one an ALImage to encapsulate raw data buffer from the V4L2 driver and an ALImage to store converted data. The returned image is actually an array of ASCII characters which we transform it into a .PNG file using the PIL (Library for image processing) library in python and we save it to the local computer.

2.3. Robot Kinematics

In this section we provide basic background information about generic robot kinematics, affine transformation matrices, and the Denavit-Hartenberg (DH) parameters. Using affine transformation matrices, and the DH parameters, we can describe the position and movements of NAO's arms in 3D space. In this way, we calculate the position of the beaters in 3D space, which is necessary for hitting the metallophone.

Kinematics is the science of geometry in motion [31]. More specifically kinematics or robot kinematics combine geometry with the study of the movement of multi-degree of freedom kinematic chains that form the structure of robotic systems [45] [38]. A robot kinematic chain is an assembly of rigid bodies (links) connected by joints. Each joint has one degree of freedom and it can be translational (prismatic joint) or rotational (revolute joint), as you can see in Figure 3.7. The relative pose of the connected rigid bodies (links) changes according to the motion of the joint. The one end of the chain is called the base and it is generally fixed and the other end of the chain is called the end-effector or tool and it is free to move in the space. The term degrees of freedom (DOF) refers to the number of joints in a kinematic chain and more DOF imply a more flexible chain.

Robot kinematics can be separated into forward kinematics and inverse kinematics (Figure 2.5). The problem of forward kinematics is straightforward and more simple than the inverse kinematics problem

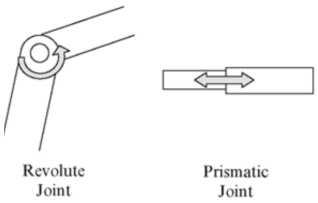


Figure 2.4: Example of revolute and prismatic joints

which is computationally expensive and singularities and non-linearities make the problem more difficult to solve.

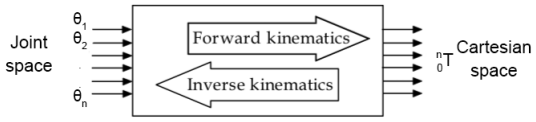


Figure 2.5: A schematic representation of forward and inverse kinematics.

2.3.1. NAO Kinematics

NAO Structure

The NAO humanoid robot has five kinematic chains (head, two arms, two legs). It is 573.2mm tall with an arm length of 290mm. The measurement between the shoulders is 273.3mm and it has about 5kg of mass (Figure 2.6). The version we are working on is the V6 and V5 with 21 DOF, however in the context of this thesis, only the two arms are needed and each arm has four DOF. In more detail, the five kinematic chains and their joints are the following:

- Head: *HeadYaw, HeadPitch*
- Left Arm: *LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll*
- Right Arm: *RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll*
- Left Leg: *LHipYawPitch, LHipRoll, LHipPitch, LKneePitch, LAnklePitch, LAnkleRoll*
- Right Leg: *RHipYawPitch, RHipRoll, RHipPitch, RKneePitch, RAnklePitch, RAnkleRoll*

In order to create the kinematic model for the NAO's arms, we need to specify the joints of the robot. According to the documentation, provided by the manufacturer of the robot, Aldebaran Robotics, we present the length of all the links of the robot and the operational range in radians and degrees of the arm joints.

Table 2.2: NAO arms joints and their operational range

Joint Name	Range in Degrees	Range in Radians
LShoulderPitch	-119.5 to 119.5	-2.0857 to 2.0857
LShoulderRoll	-18 to 76	-0.3142 to 1.3265
LElbowYaw	-119.5 to 119.5	1.5446 to 0.0349
LElbowRoll	-88.5 to -2	-0.6720 to 0.5149
RShoulderPitch	-119.5 to 119.5	-2.0857 to 2.0857
RShoulderRoll	-38.5 to 29.5	-1.3265 to 0.3142
RElbowYaw	-119.5 to 119.5	-2.0857 to 2.0857
RElbowRoll	-38.5 to 29.5	0.0349 to 1.5446
LWristYaw	-104.5 to 104.5	-1.8238 to 1.8238
RWristYaw	-104.5 to 104.5	-1.8238 to 1.8238

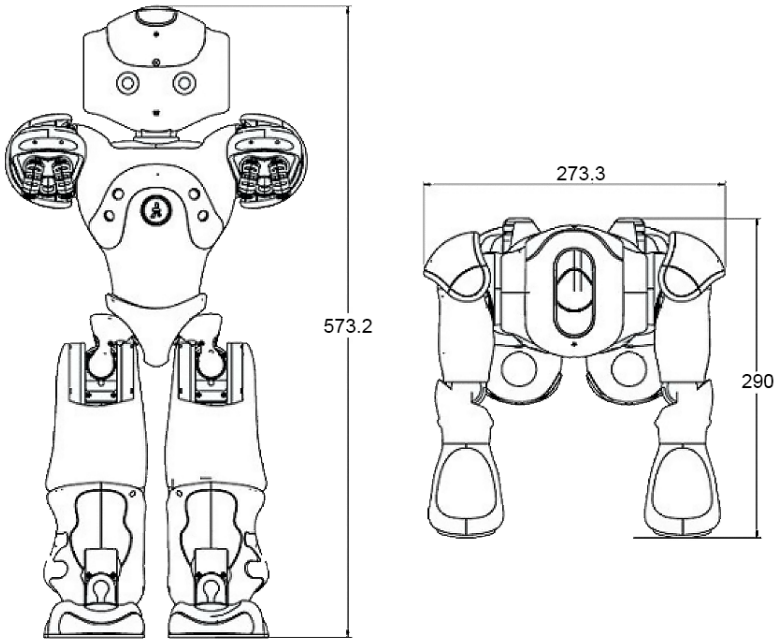


Figure 2.6: NAO's hardware lengths

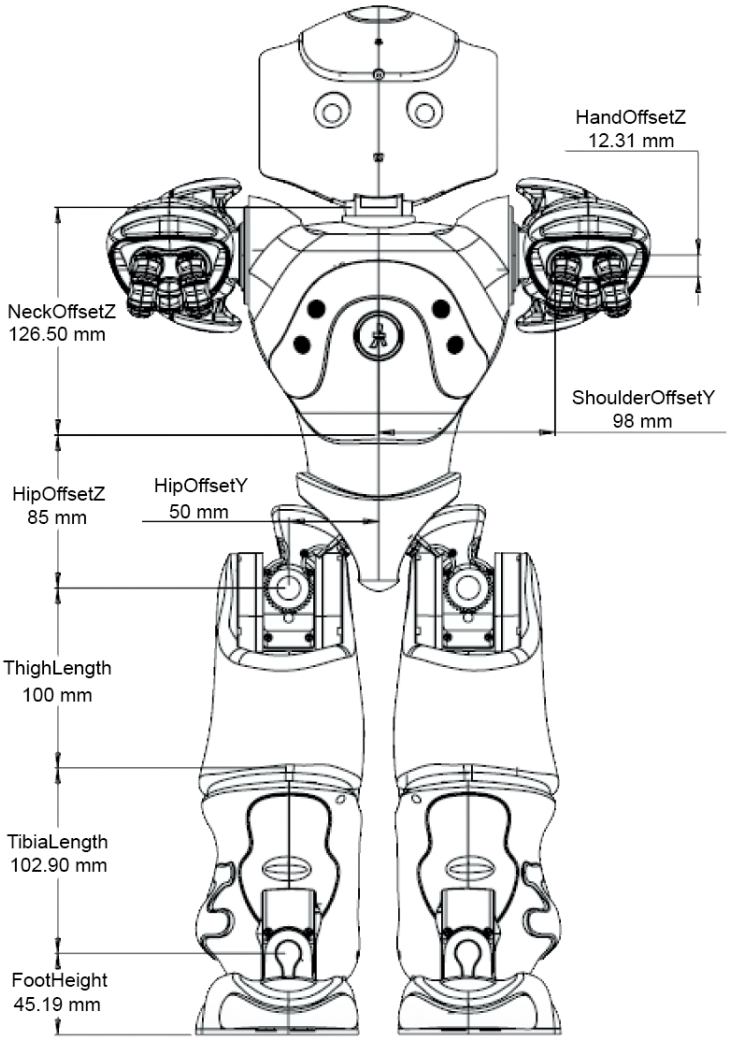


Figure 2.7: Lengths of NAO's Parts

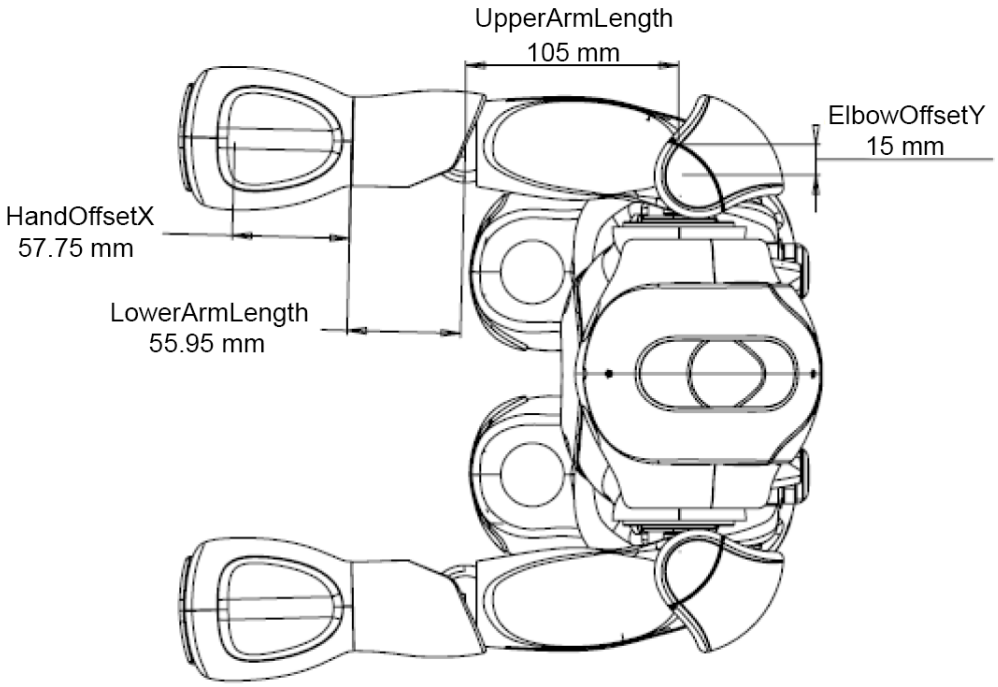


Figure 2.8: NAO's arm lengths

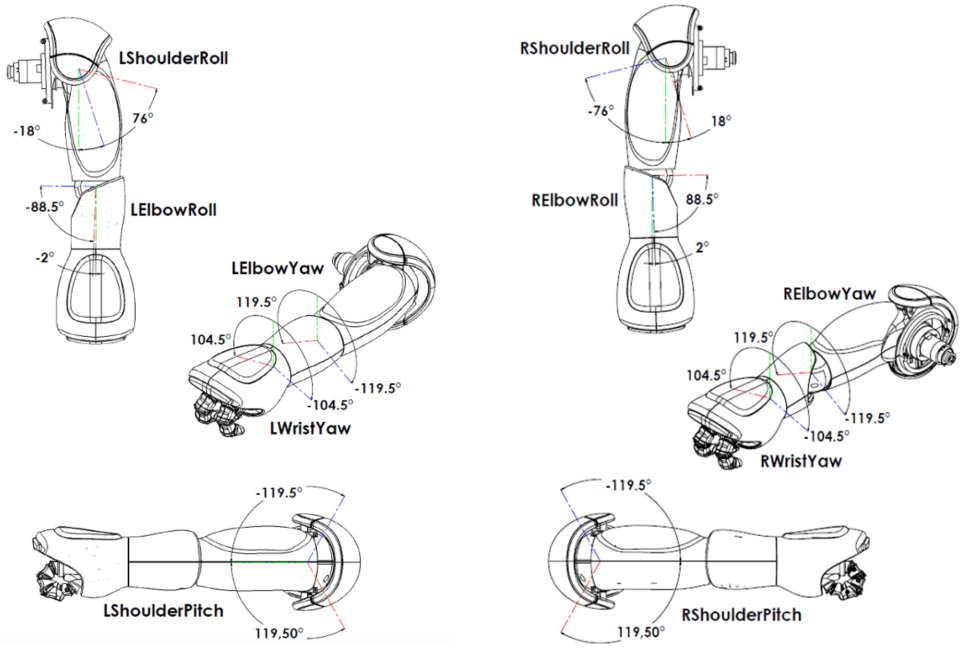


Figure 2.9: NAO's left and right arm joints

2.3.2. Forward Kinematics

Forward kinematics is the mapping from joint coordinates, or robot configuration, to end-effector pose in the three-dimensional Cartesian space. [11]. Given a kinematic chain with m joints and a set of joint values (θ_1, θ_2, θ_3, ..., θ_m), the forward kinematics can find the position (p_x, p_y, p_z) and the orientation (a_x, a_y, a_z) of the end-effector of the kinematic chain in the three-dimensional x-y-z space. Solving a forward kinematics problem is domain independent and it can be solved for any kinematic chain yielding a closed-form, analytical solution.

The Forward Kinematics Problem for NAO

The forward kinematics problem is to define a mapping from the joint space of the robot to the three-dimension space with respect to any base coordinate frame. NAO's joints contain 12-bit encoders with an update frequency of 100Hz. These encoders are able to provide the current joint values at any time.

In this project, we are interested about the kinematic problem of the robot's arms and the corresponding solution. The solution of the kinematic problem provides the position and orientation, of the end-effector, in the three-dimensional space with respect to a base coordinate frame.

2.3.3. Inverse Kinematics

We have shown how to determine the position of the end-effector in space, given the joint coordinates. However, another problem which is more practical and more interesting is the inverse problem. As in our project, we need to make the robot reach a target point in the Cartesian space and to make this happen we have to specify the appropriate values for the joints of the kinematic chain. The inverse kinematics define ways to go from the three-dimensional space to the joint space. More specifically, the inverse kinematics is a matching between points in the three-dimensional space (position (p_x, p_y, p_z) and orientation (a_x, a_y, a_z)) and joint values/angles (θ_1, θ_2, θ_3, ..., θ_m) in the joint space of a kinematic chain with m joints. The problem of inverse kinematics is domain-dependent and for each kinematic chain there is a different solution. Two approaches can be used to determine the inverse kinematics, a closed-form or analytic solution, using geometric or algebraic approaches, and an iterative numerical solution. As the number of robot joints increases, a point in the three-dimensional space may have more than one matching point in the joint space.

The Inverse Kinematics Problem for NAO

The inverse kinematics problem is to define a relation between the position and orientation of an end-effector, in the three-dimensional space, and joint values in the joint space of a kinematic chain.

Inverse kinematics represents a much more difficult problem compared to forward kinematics for at least two reasons. First, it leads to a system of non-linear equations, which may, or may not, have an analytical solution. Second, as the number of DOF increases and the kinematic chain becomes more flexible, a point in the three-dimensional space may have more than one matching points in the joint space of the chain. This multiplicity of solutions defines a complex relation, but not a mapping, between the two spaces.

The inverse kinematics problem can be solved analytically with closed-form equations or numerically with an iterative approximation method [14]. The analytical solution is in general faster than the fastest numerical solution and therefore is more appropriate for real-time execution. Numerical solutions are also subject to singularities, which result in a failure to obtain a solution, even if one exists.

2.3.4. Affine Transformations

An affine transformation is a function that transforms points, straight lines and planes from one space to another, maintaining the ratios of distances. The source and target spaces can be n-dimensional with n ≥ 2. In general, an affine transformation is a composition of rotations, translations, dilations, and shears. In this project we are going to work on the three dimensional Cartesian space thus, we are going to focus on definitions about this space. Moreover, using only rotations and translations, we can fully describe our kinematic model, therefore we will focus only on these two types of affine transformations.

Affine Transformation Matrix

An affine transformation matrix is a ((n + 1) × (n + 1)) matrix, where n is the number of dimensions in the space the transformation is defined on and is a block matrix of the form:

$$T = \begin{bmatrix} & X & & Y \\ [0 & \dots & 0] & 1 \end{bmatrix} \tag{2.8}$$

where X is a (n × n) matrix, Y is a (n × 1) vector and the last line of T contains n - 1 zeros followed by a 1. Given a point $p = (p_1, p_2, ..., p_n)$ and an affine transformation matrix T, we can apply the transformation to this point by multiplying the affine transformation matrix with the column vector $v = (p_1, p_2, ..., p_n, 1)^T$. Hence the new point p' can be found from the resulting vector v':

$$v' = \begin{bmatrix} p'_1 \\ p'_2 \\ ... \\ ... \\ p'_n \\ 1 \end{bmatrix} = T \times v = \begin{bmatrix} & X & & Y \\ [0 & ... & 0] & 1 \end{bmatrix} \times \begin{bmatrix} p_1 \\ p_2 \\ ... \\ ... \\ p_n \\ 1 \end{bmatrix} \tag{2.9}$$

In this project we work on the three dimensional Cartesian space, therefore, for a point $p = (p_x, p_y, p_z)$ in the three-dimensional space, the transformation will be:

$$v' = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = T \times v = \begin{bmatrix} X_{xx} & X_{xy} & X_{xz} & Y_x \\ X_{yx} & X_{yy} & X_{yz} & Y_y \\ X_{zx} & X_{zy} & X_{zz} & Y_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \tag{2.10}$$

Moreover, it is very common to combine two or more affine transformation matrices. In these cases we need to multiply these affine transformation matrices and the matrix that results from the multiplication is also an affine transformation:

$$T = T_{12} = \begin{bmatrix} & X_1 & & Y_1 \\ [0 & ... & 0] & 1 \end{bmatrix} \times \begin{bmatrix} & X_2 & & Y_2 \\ [0 & ... & 0] & 1 \end{bmatrix} = \begin{bmatrix} & X_1 X_2 & & X_1 Y_2 + Y_1 \\ [0 & ... & 0] & 1 \end{bmatrix} \tag{2.11}$$

Translation Matrix

In Euclidean geometry, a translation is a function that moves every point by a fixed distance in a given direction. A translation is an affine transformation with no fixed points and we can describe a translation in the three-dimensional space with a (4 × 4) matrix:

$$A = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

where d_x, d_y and d_z define the distance of translation along the x, y and z axes respectively. To sum up, translation matrix is an affine transformation with the X matrix equal to I. Thus, if we want to apply just a displacement (d_x, d_y, d_z) in a point $p = (p_x, p_y, p_z)$ in three-dimensional space, we need to multiply this point with the translation matrix:

$$v' = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = A \times v = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \tag{2.13}$$

Rotation Matrix

A rotation matrix, R, is a (3×3) orthogonal matrix and it describes the rotation of an object in the three-dimensional space. More specifically, there are three distinct rotation matrices, each for one of the three-dimensions in the three-dimensional Cartesian space x, y, z. Each of the matrices R_x, R_y, R_z represents a rotation ($\theta_x, \theta_y, \theta_z$), about the x, y, z axes respectively.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_x & -sin\theta_x \\ 0 & sin\theta_x & cos\theta_x \end{bmatrix} \quad R_y = \begin{bmatrix} cos\theta_y & 0 & sin\theta_y \\ 0 & 1 & 0 \\ -sin\theta_y & 0 & cos\theta_y \end{bmatrix} \quad R_z = \begin{bmatrix} cos\theta_z & -sin\theta_z & 0 \\ sin\theta_z & cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Similarly to the translation matrix, we can deploy a rotation on a vector, defined by a point $p = (p_x, p_y, p_z)$, by multiplying it with the appropriate rotation matrix. Thus, if we want to rotate a vector about the x axis we have:

$$p' = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = R_x \times p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_x & -sin\theta_x \\ 0 & sin\theta_x & cos\theta_x \end{bmatrix} \times \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{2.14}$$

If we want to deploy multiple rotations about the three axes, we just need to multiply the rotation matrices and the result will be again a rotation matrix. However, in this case, the order of matrices in the multiplication affects the order of the rotations. Hence, if we want to deploy a rotation on a vector,defined by a point $p = (p_x, p_y, p_z)$, about the x axis, then about the y axis and then about the z axis the order will be as follows:

$$p' = R_z \times R_y \times R_x \times p \tag{2.15}$$

In the same manner with the transformation matrix, we can represent the (3×3) rotation matrix R' as an affine transformation by padding the last line and the last column with (0, 0, 0, 1):

$$\mathrm{R} = \begin{bmatrix} & R' & & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & & & 1 \end{bmatrix}$$

In this this project we will represent all the rotation and translation matrices, as affine transformations. The X block of the affine transformation matrix, represents the rotation and the Y block represents the translation:

$$\mathrm{T} = \begin{bmatrix} & R' & & \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} & & & 1 \end{bmatrix}$$

2.3.5. Denavit-Hartenberg (DH) Parameters

Denavit and Hartenberg [15] [27], in 1955, introduced four parameters associated with a particular convention for describing points in one end of a joint to a coordinate system that is fixed to the other end of the joint, as a function of the joint state. More specifically, we can use transformation matrices to describe these changes on the coordinate frames starting from the origin joint to the end-effector. These parameters are known as Denavit-Hartenberg (DH) parameters: a, α, d, and θ but before we can explain these parameters we need to mention how the coordinate axes are formed for each joint i with respect to the coordinate axes of its previous joint. We start by defining the z_i axis which is always set to the direction of the joint axis or in other words to the rotation direction. The direction of x_i is derived using the right-hand rule from $z_i - 1$ to z_i. Moreover, the x_i axis is parallel to the common normal between z_i and $z_i - 1$. Finally, the y_i axis implies from the x_i and z_i axes forming a right-handed coordinate system.

Now we can describe the DH parameters based on the image 2.10:

- d is the offset along the z_{i-1} axis to z_i axis
- θ is the angle about the z_{i-1} axis, from x_{i-1} axis to x_i axis
- a is the displacement from z_{i-1} axis along the x_i axis
- α (alpha) is the angle about the common normal (x-axis), from z_{i-1} axis to z_i

2.3.6. Overview of Kinematics applied on NAO

Kofinas et al. [33] presented a complete, exact, analytical solution for the problems of forward and inverse kinematics of the NAO robot. The solution that they presented resulted in a high degree of accuracy, efficiency, and in the elimination of singularities. This approach, to NAO kinematics, is based on standard principled methods for studying robot kinematic chains and so far is the only published complete analytical solution for all kinematic chains of NAO robot. Graf et al. [24] had presented a solution only for the kinematic

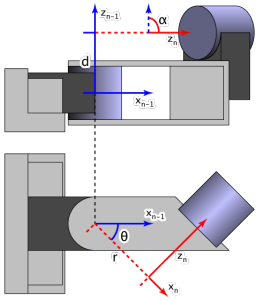


Figure 2.10: Denavit-Hartenberg parameters: d, θ, a, α (from www.tekkotsu.org)

chain of the legs of NAO which is purely geometric, and cannot be generalized to other kinematic chains. Moreover, the numerical solution [1] offered by the manufacturer of the robot, Aldebaran Robotics, is a proprietary implementation, but it is prone to singularities and, therefore, is not robust.

2.4. Vision-Based Control

The hand-eye calibration of the robot is necessary in order to achieve high precision in robot's movements. Even if the kinematic structure is known from the construction plan, errors can occur due to imperfect manufacturing or low accuracy in NAO's actuators.

Using visual feedback to control a robot is commonly termed visual-servoing [30]. Visual-servoing involves the use of one camera and a Computer Vision system, to control the position of a robotic arm relative to a part it has to manipulate, which requires detecting, tracking, servoing, and grasping. It therefore spans computer vision,robotics, kinematics, dynamics, control and real-time systems, and is used in a rich variety of applications such as lane tracking for cars,navigation for mobile platforms, and generic object manipulation.

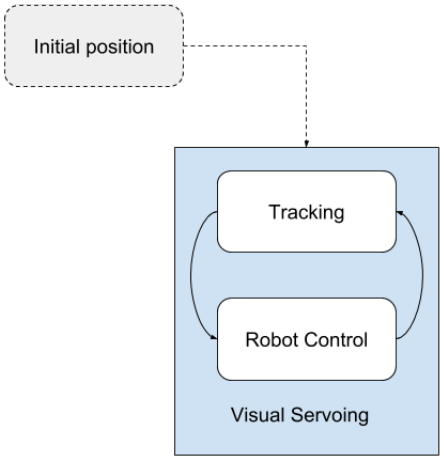


Figure 2.11: Closed–loop control

A closed–loop control of a robotic system, where vision is used as the underlying perception sensor, usually consists of two intertwined processes: tracking and control (Figure 2.11). Tracking provides a continuous estimation of the moving object. Based on this input, a control sequence is generated. Moreover, the system requires an automatic initialization which commonly includes object recognition.

The tracking information is required to measure the error between the current location of the robot and its reference or desired location from eye-in-hand cameras. As a consequence, the tracking algorithm must be robust, accurate, fast, and general. Visual features such as points, lines and regions can be used to, for example, enable the alignment of a gripping mechanism with an object. Hence, vision is a part of a control system where it provides feedback about the state of the environment.

In this project, we measure the error between the current location of the beater and the desired location. In general this error can be measured by two dimensional information, expressed by using image plane coordinates, or by three dimensional where camera/object model is employed to retrieve pose information with respect to the robot coordinate system. Hence the three main categories of visual servo are:

- *Position-based visual servo systems*
  In these systems, the pose of the goal with respect to camera is estimated by retrieving the three-dimensional information about the scene where known camera model is used to estimate the position and the orientation (pose) of the target with respect to the camera (world, robot) coordinate system.
- *Image-based visual servo systems*
  In these systems, 2D image measurements are used directly to estimate the desired movement of the robot. Typical tasks like tracking and positioning are performed by reducing the image distance error between a set of current and desired image features in the image plane.
- *2 1/2 D visual servo systems*
  This is a combination of the previous two approaches. More specifically, in these systems the error is specified and minimized both in the image and in the pose space.

2.4.1. Overview of Visual-Servoing methods applied on NAO

Daniel et al. [13] approached this issue based on [29] [3] by using self-calibration procedure based on checkerboard-markers to accurately estimate these errors. More specifically, Hubert et al. [29] presented a Bayesian approach to calibrating the hand-eye kinematics of an anthropomorphic robot. In this approach, the robot perceives the pose of its end-effector with its head-mounted camera through visual markers attached to its end-effector. Moreover Moughlbay et al. [42] presented an approach for making NAO be able to grasp objects. More precisely, 3D visual feedback data and Model Based Tracking (MBT) techniques were used to execute, in real-time and closed loop, manipulation tasks on the humanoid mobile robot NAO. In our approach, the end-effector's actual 3D position is detected through the robot's camera and it is compared with the estimated 3D position according to the kinematics configurations. In this way, the markers on the robot's arms are not necessary, making the robot looks like more natural as well as our application is more accessible to simple users.

3

Instrument Playing NAO

In this chapter we analyze in depth the methods that we chose to combine in order to make the humanoid NAO able to play the metallophone. First, we present our approach for making NAO detect the position of the metallophone in 3D space. Then, we explain how we aim to make our vision approach robust to illumination changes.

Regarding the kinematic part, we describe how we make the robot move its arms precisely by calculating the inverse kinematics in an offline phase. Finally we describe our approach for the elimination of errors in the kinematic configurations, which occur by the variances in the actuators among different robots, and by the impreciseness of the motors (they're precise up to a limit).

3.1. Pose estimation using NAO's Monocular Vision

In this section, we present our approach based on Vacchetti et al. [50], Gordon and Lowe [23] and Choi and Christensen [10] for making NAO estimate the position and orientation of the metallophone in 3D space. The robot needs this information in order to decide about its actions. The robot needs to estimate the position of the metallophone in relation to its body. Then, according to this estimate the robot moves its arms, if the metallophone is in a reachable position.

We propose a real-time approach for estimating the pose of a known object which can handle camera displacements, aspect changes and illumination changes. More specifically, we match frames from robot's current view with a limited number of reference images (key-frames). This match results in a homography matrix which is used for extracting the pose of the object in the current frame.

As discussed in Chapter 2, for estimating an object's pose, based on monocular vision, some prior knowledge of the object is necessary. In our approach, a sparse 3D metric model is used which means that it is not necessary to use the dimensions of the whole object (3D CAD model). We need the locations of a few points on the image, containing the object, with their corresponding 3D locations. In an offline stage, we create these key-point models containing this information.

We use a set of images where each image has a view of the metallophone with slightly different illumination. The non-linear optimization algorithm, returns a refined 3D model of key-points. Using this model which has the 3D coordinates of each key-point, the pose estimation is performed by using the correspondences between the 3D points of the model and the 2D marked points on an input image.

During the online stage, the robot captures the current image of the musical instrument and we use a feature detection and a feature matching algorithm to match the current image with reference images (key-frames). This matching allows us to update the key-points' position according to the current frame which results in the pose estimation of the metallophone in the current frame. In other words, the matching of input frame with key-frames results in a map of the key-frame on the current frame. This mapping change the pose of key-frames according to the current view, consequently we get the current pose of the object.

3.1.1. Key-Frames Model Acquisition

To estimate object's pose, our approach requires key-frames. Key-frames are reference images which depict the metallophone and they contain 2D and 3D information of key-points. Since key-frames will be compared with the input image, they should contain an appearance of the metallophone similar to the one in the input

image. We assume that the appearance of the metallophone in the input image will be normal (i.e the metallophone will not be upside down) in front of the robot. We do not create key-frames to cover all possible appearances of the metallophone due to variability across scale, orientation and viewpoint. Since we want to create a robust system against different light conditions, we create key-frames to cover the variability across illumination.

In an offline phase, we put the robot in the initial position and we capture the reference images using the lower camera of the robot. Lower camera is preferred because the robot has a complete view of the object without moving its head. Due to the convenience of the lower camera, no head movement is necessary in this work. As a result, the calculation of object's pose in relation to the robot is simplified.

Key-Points determination
Since key-frames are used for pose estimation, we need to generate 2D-3D correspondences. As key-points we define the 2D and 3D coordinates which we set in our key-frames. We choose five key-points on the image plane and we manually mark them by saving their 2D coordinates (Figure 3.1). We choose the four outer corners of Mi notes and the fifth point is the center of the metallophone. The center point is not need to be marked manually, it is calculated automatically using the four marked corner points. In these key-points, the center of the object is necessary for representing the position of the object in 3D world coordinates. As far as the other four points are concerned, there is not a specific rule about their choice. We could choose also different ones.

Figure 3.1 depicts an example of a key-frame. The 2D coordinates of the points are not always the same and they depend on the view of the object in the image. For this reason we need to manually extract these coordinates for each key-frame.

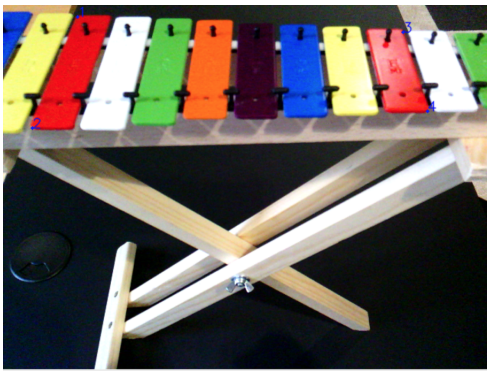


Figure 3.1: Four points used for the 2D-3D correspondences

However, regarding the relative 3D coordinates of the key-points, their values are fixed, since the metallophone is a rigid object. We set the center of the metallophone as the (0,0,0) in the world coordinate system. Then we set the rest of the points since we know the dimensions (in centimeters) of the metallophone.

- Upper left point: (-9, 4.5, 0)
- Lower left point: (-9, -4.5, 0)
- Upper right point: (9, 3.25, 0)
- Lower right point: (9, -3.25, 0)

We explain more about how we use this information in subsection 3.1.4.

3.1.2. Image Processing

In pose estimation procedure, we process the images to make the matching phase between the current frame and key-frame, more efficient and robust. We use an edge detection method to transform the image since the metallophone has a characteristic shape of multiple rectangles. With this transformation we aim to create more descriptive feature points which is going to make the image matching phase more stable and robust. In addition, one of our goals is to develop a robust vision approach against different light conditions. Converting the normal image into an edged image, we expect to achieve high accuracy in our results.

After retrieving the image from the robot (Figure 3.4), we use a Gaussian filter to reduce image noise and to preserve the edges in the image. Gaussian smoothing preserves boundaries and edges better than other, more uniform blurring filters. Then we use the Canny Edge [7] detection algorithm to extract only the edges of the image.

Canny Edge Detection Overview
The Canny edge detector, developed in 1986 by Canny [7], uses a multi-stage algorithm to detect a wide range of edges in images and it is the most used edge detector in image processing. The Canny edge detection algorithm can be analyzed into four steps (According to OpenCV library's implementation [4]):

1. Apply a Gaussian filter to the image in order to remove high frequency noise.

2. Compute the image's gradient intensity representations. Canny Edge detection algorithm uses four filters to detect edges in all directions (horizontal, vertical and diagonal) in the blurred image.

3. Apply non-maximum suppression to remove any unwanted pixels which may not constitute the edge. The image magnitude produced results in thick edges. Ideally, the final image should have thin edges, hence, non-maximum suppression thins out the edges.

4. The final step is called Hysteresis Thresholding, and at this step, the algorithm decides which edges are strong enough to keep and which are not. For this, we need to give as input to the algorithm a dual-threshold value, the minVal and the maxVal. Given these parameters, the algorithm keeps the edges with intensity gradient more than maxVal and discard edges below minVal. The edges that lie between the minVal and maxVal thresholds, are discarded if they are not connected with strong edges, otherwise they are considered to be part of edges.

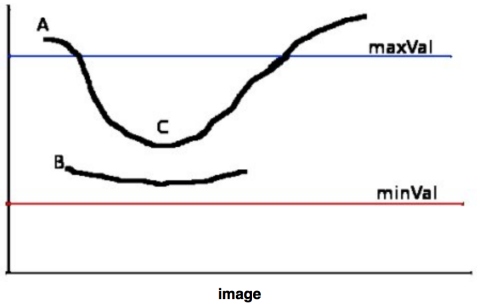
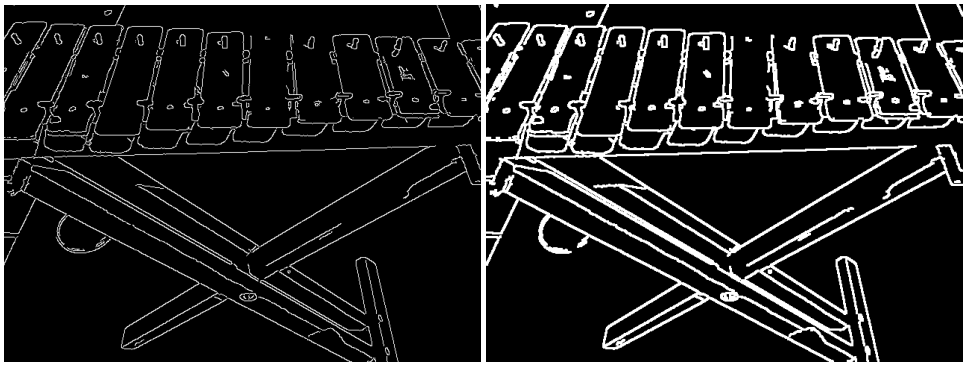


Figure 3.2: The edge A considered as a strong edge since it is above the maxVal and even if the edge C is below maxVal, it is connected to edge A, thus edge C is also considered as valid edge and we get that full curve. Edge B is above the minVal but it is not connected to any strong edge, thus, it is discarded. OpenCV documentation (https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html).

Figure 3.2 illustrates how the dual-threshold value works. It is hard to decide manually about these values, and they need small adjustments for every different image, otherwise the Canny edge algorithm's performance is poor. In our approach, we use an adaptive threshold Canny edge algorithm, based on Otsu method. Otsu method performs well in choosing threshold value automatically and is proposed in 1979 [43]. This algorithm splits the image's pixels into two classes, and confirms the best threshold value through the variance maximum value between the two classes [17]. Using adaptive thresholding on Canny edge algorithm, we succeed well shaped edges after the image processing. The most important is that we succeed well shaped edges even if the initial image was taken in darker or brighter environment, and this contributes to make our pose estimation more robust. At the final step of image processing, we dilate the edged image in order to make the lines more solid.

3.1.3. Real time 2D Image Matching based on Image Features

At the online stage of our approach, NAO captures and retrieves an image containing its current view of the bottom camera (Figure 3.4). Then, we process the input frame in the same way we processed key-frames 3.1.2, and in a while loop we match the current input frames with pre-computed key-frames. For the image matching procedure, we use the feature extraction SIFT algorithm (Scale-Invariant Feature Transform) [37]



(a) Image after Canny edge detection method (b) Image after dilation

Figure 3.3: Image processing steps

and a feature matching algorithm and we calculate the homography matrix between the two images. In this way, we update the position of the key-points which are used in the pose estimation step.

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.1}$$

where s is a scale factor and H is the homography matrix.

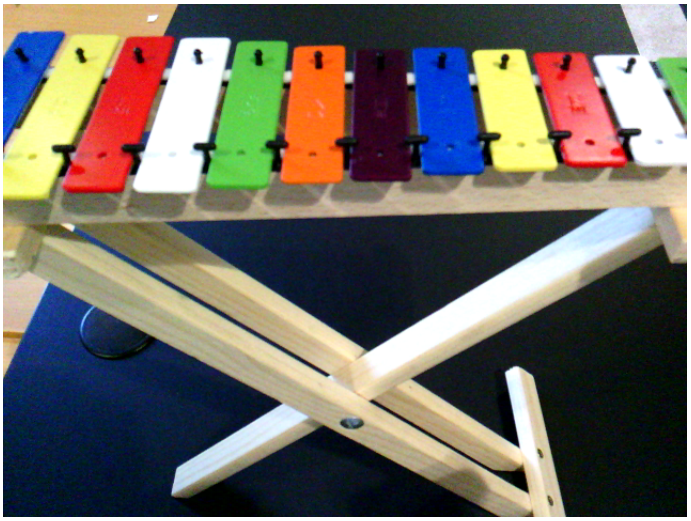


Figure 3.4: Image returned from NAO's bottom camera

SIFT (Scale Invariant Feature Transform)
Scale Invariant Feature Transform (SIFT) is an algorithm in computer vision for detecting and describing local features in images and it is commonly used for image-based matching and recognition [37]. The reason that we chose this method is because SIFT descriptor is invariant to translations, rotations and scaling transformations in the image domain. Moreover, it works robustly against perspective transformations and illumination variations. Experiments have proven that the SIFT descriptor is an effective method in practice, for image matching and object recognition under real-world conditions.

SIFT algorithm, automatically detects points on images with descriptive features and each returned point has a descriptor. More specifically a 16x16 neighbourhood around the feature point is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created and a total of 128 bin values are available. The feature points' descriptors are represented as vectors, and these vectors are compared in order to find similarities between these points.

Brute Force Matcher

After extracting 2D points, with their descriptors, in both the key-frame and the current frame, the next step is to compare the descriptors of each feature point between the two images. To do this we use the Brute force matcher which takes the descriptor of one feature in the first set and is matched with all other features in second set using some distance calculation and then the closest one is returned [4].

More specifically, there is a Boolean variable in the algorithm, the crossCheck variable which is false by default. When the crossCheck variabe it is true, the brute force Matcher returns only those matches with value (i,j) such that i-th descriptor in set A has j-th descriptor in set B as the best match and vice-versa. That is, the two features in both sets should match each other.

After finding enough matches (Figure 3.5) between the two images we calculate the homography matrix using the findHomography() function of the openCV library and then we map the key-points using the perspectiveTransform function of the openCV library.

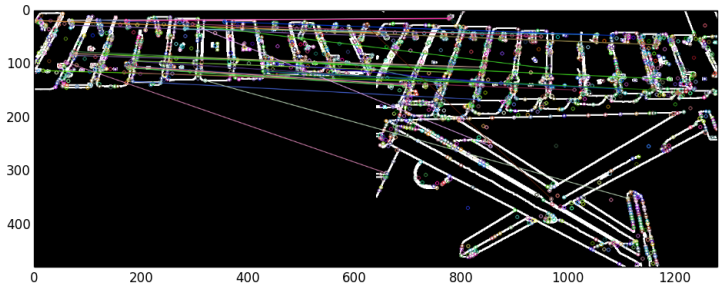


Figure 3.5: Matching key-frame (left) with current image (right)

3.1.4. 3D Pose Estimation

Estimating the External Parameters Matrix

Since we have the calibration matrix, we just need to calculate the External Parameters matrix ([R|t]) in order to be able to move from the 3D world points to 2D image plane points using the equation 2.1.

In the previous subsection, we saw how we update the 2D position of the key-points by matching the input and reference images. Now, we solve the pose estimation problem using the OpenCV function solvePnP with our key-points. The function solvePnP implements several algorithms for pose estimation. In this project, the SOLVEPNP_ITERATIVE was used which is the DLT solution followed by Levengerg-Marquardt optimization, giving stable results. More specifically, the solvePnP function takes five arguments, the 2D key-points coordinates, the corresponding 3D key-points coordinates of the object, the camera intrinsics, the distortion of the lens and the algorithm that we want to use (SOLVEPNP_ITERATIVE). The output of the solvePnP function is a rotation and a translation vector and using the Rodrigues' rotation formula, we can find the 3x3 rotation matrix from the rotation vector. Therefore, given the camera intrinsic parameters and the key-points of the object, we calculate camera extrinsic parameters (pose) between the object and the robot (Figure 3.6). Figure 3.7 illustrates the estimated pose of a current frame using the solvePnP function.

After computing R and t, using the equations 2.4 and 2.5 we retrieve the position of the object (center of metallophone) in relation to NAO's camera. Then with a simple translation we calculate the object's position in relation with robot's body. In this way the robot knows that the metallophone is placed in a x,y,z point in relation to its body. Then, the robot checks if it can reach this 3D point with its end-effector. In the next section we present our approach for calculating the kinematics which is necessary for the robot in order to move its arms.

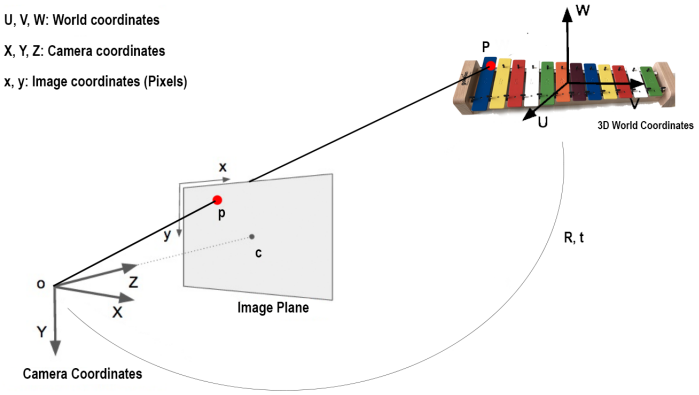


Figure 3.6: Connection between the coordinate systems

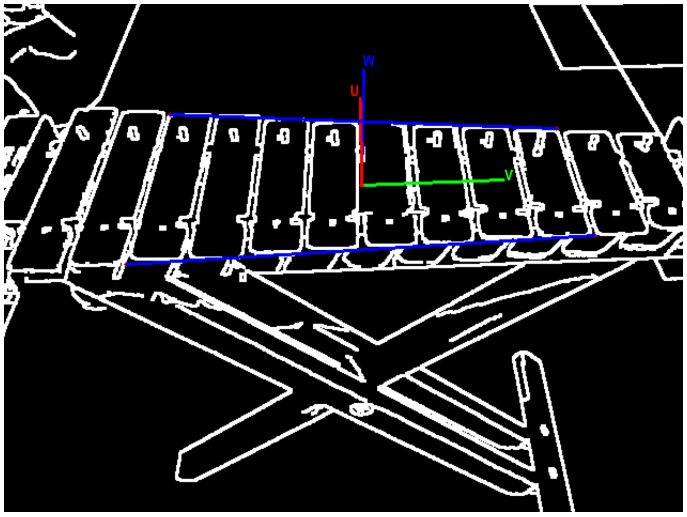


Figure 3.7: Five 3D points on the metallophone

3.2. NAO Kinematics

In Chapter 2 we presented some definitions about robot kinematics and the kinematic chains that the robot has. In this section we present our approach for calculating the inverse kinematics of both arms of the robot. Since the robot estimates the position of a note in 3D space, it must be able to move its arm in this position, and to do this we need inverse kinematics.

3.2.1. Forward Kinematics

Before we proceed with the calculation of inverse kinematics, we need to create the kinematic chains of both arms of the robot, holding the beaters. In this subsection we present how we created the kinematic chains based on the DH parameters mentioned in Chapter 2.

Forward Kinematics for the Left Arm

The left arm of NAO constitutes a kinematic chain of five joints. Therefore, we need to calculate five sets of DH parameters, one for each joint. The first step is to transfer the origin point from the torso to the base of the first joint and we can achieve that by making a translation along the y-axis and the z-axis. The second step that we need to follow, is to align the coordinate frame with the rotation axis of the first joint which is the LShoulderPitch. This can be achieved by doing a rotation of $-\frac{\pi}{2}$ about the x-axis of the coordinate frame, which means that the parameter α for the LShoulderPitch is $-\frac{\pi}{2}$, while the parameters d and a are set to 0. Now, the coordinate frame, must be rotated again to become aligned with the rotation axis of the second joint which is the LShoulderRoll and this can be achieved by doing a rotation about the x-axis by $\frac{\pi}{2}$, hence the parameter α for LShoulderRoll is $\frac{\pi}{2}$, while the parameters d and a are set to 0. Next, we need

to align the coordinate frame with the rotation axis of the third joint which is the LElbowYaw. In order to achieve this, we need to rotate the coordinate frame about the y-axis, however the DH parameters do not directly encode a rotation about the y-axis, thus, we must first rotate about the z-axis and then about the x-axis which is equivalent with a rotation on y-axis. The rotation on z-axis can be done by subtracting by -$\frac{\pi}{2}$ the angle θ_2, which is the angle of the previous joint, and then, we rotate about the x-axis by -$\frac{\pi}{2}$ therefore the parameter α for LElbowYaw is -$\frac{\pi}{2}$. Now, we need to move along the z-axis in order to reach the position of the LElbowYaw joint, so the parameter d is set to UpperArmLength, as well as the parameter a is set to 0. Subsequently, we need to rotate again the coordinate frame about the x-axis by $\frac{\pi}{2}$ in order to align the z-axis with the LElbowRoll which means that the parameter α for the LElbowRoll is $\frac{\pi}{2}$, while the parameters d and a are set to 0. Finally, for the fifth joint (LWristYaw), we make a rotation about the x-axis in order to align the z-axis with the LWristYaw joint and we move along the z-axis in order to reach the position of the LWristYaw joint, so the parameter d is set to LowerArmLength, however, we add the HandOffsetX since we want to move the coordinate frame at the point that the robot holds the stick. At the end, we need to fix the orientation of the coordinate frame by doing a rotation about the x-axis and then about the z-axis as well as we make a translation on the y-axis to reach the end-effector which is the end of the mallet (stick).

Table 3.1: DH parameters for the left arm chain of NAO robot

Frame	a	α	d	θ
Base	A(0, ShoulderOffsetY + ElbowOffsetY, ShoulderOffsetZ)			
LShoulderPitch	0	-$\frac{\pi}{2}$	0	θ_1
LShoulderRoll	0	$\frac{\pi}{2}$	0	$\theta_2 - \frac{\pi}{2}$
LElbowYaw	0	-$\frac{\pi}{2}$	UpperArmLength	-θ_3
LElbowRoll	0	$\frac{\pi}{2}$	0	θ_4
LWristYaw	0	-$\frac{\pi}{2}$	LowerArmLength + HandOffsetX	θ_5
Rotation	$R_x(\frac{\pi}{2})$			
Rotation	$R_z(\frac{\pi}{2})$			
End effector	A(0, -Stick, 0)			

Forward Kinematics for the Right Arm

The right arm of NAO constitutes a kinematic chain of five joints and it is fully symmetric with the left arm chain. Therefore, we need to calculate again five sets of DH parameters. The first step is to transfer the origin point from the torso to the base of the first joint and we can achieve that by making a translation along the y-axis and the z-axis. The second step that we need to follow, is to align the coordinate frame with the rotation axis of the first joint which is the RShoulderPitch, by doing a rotation of $-\frac{\pi}{2}$ about the x-axis of the coordinate frame. Now, the coordinate frame, must be rotated again to become aligned with the rotation axis of the second joint which is the RShoulderRoll and this can be achieved by doing a rotation about the x-axis by $\frac{\pi}{2}$. Next, we need to align the coordinate frame with the rotation axis of the third joint which is the RElbowYaw. In order to achieve this, we need to rotate the coordinate frame about the y-axis, however, the DH parameters do not directly encode a rotation about the y-axis, hence we must first rotate about the z-axis and then about the x-axis which is equivalent with a rotation on y-axis. The rotation on z-axis can be done by adding by $\frac{\pi}{2}$ the angle θ_2, which is the angle of the previous joint, and then, we rotate about the x-axis by -$\frac{\pi}{2}$ thus, the parameter α for RElbowYaw is -$\frac{\pi}{2}$. Now, we need to move reversibly along to the z-axis in order to reach the position of the RElbowYaw joint, so the parameter d is set to -UpperArmLength, as well as the parameter a is set to 0. Subsequently, we need to rotate again the coordinate frame about the x-axis by $\frac{\pi}{2}$ in order to align the z-axis with the RElbowRoll and finally, for the fifth joint (RWristYaw), we make a rotation about the x-axis in order to align the z-axis with the RWristYaw joint and we move reversibly along to the z-axis in order to reach the position of the RWristYaw joint, so the parameter d is set to -LowerArmLength, however, we add also the HandOffsetX since we want to move the coordinate frame at the point that the robot holds the stick. At the end, we need to fix the orientation of the coordinate frame by doing a rotation about the x-axis and then about the z-axis as well as we make a translation on the y-axis to reach the end-effector which is the end of the mallet (stick).

Table 3.2: DH parameters for the right arm chain of NAO robot

Frame	a	α	d	θ
Base	A(0, - ShoulderOffsetY - ElbowOffsetY, ShoulderOffsetZ)			
LShoulderPitch	0	$-\frac{\pi}{2}$	0	θ_1
LShoulderRoll	0	$\frac{\pi}{2}$	0	$\theta_2 + \frac{\pi}{2}$
LElbowYaw	0	$-\frac{\pi}{2}$	UpperArmLength	θ_3
LElbowRoll	0	$\frac{\pi}{2}$	0	θ_4
LWristYaw	0	$-\frac{\pi}{2}$	- LowerArmLength - HandOffsetX	θ_5
Rotation	$R_x(\frac{\pi}{2})$			
Rotation	$R_z(-\frac{\pi}{2})$			
End effector	A(0, Stick, 0)			

3.2.2. Inverse Kinematics

We have already shown how we can determine the position and orientation of the end-effector, given the joint coordinates, however, as we said before, we are interested in the inverse problem: given a desired pose of the end-effector ξ_E, calculate the joint coordinates [11]. The functional form of the inverse kinematics problem is:

$$q = K^{-1}(\xi) \tag{3.2}$$

and the solution of this equation is not unique. There is the possibility of different coordinate vectors q to result in the same end-effector position and orientation.

Numerical solution

We use the Robotics Toolbox in Matlab [12] in order to pre-calculate the inverse kinematics. More specifically, we used the function ikine() of Robotics Toolbox given as input all possible values of x and y coordinates without taking into account the rotation and orientation of the end-effector.

We can think of the inverse kinematics problem as one of adjusting the joint coordinates until the forward kinematics matches the desired pose. The general numerical approach does not provide explicit control over the arm's kinematic configuration as did the analytic approach as well as the general numerical approach is slower than the analytic approach. However, the numerical approach has the advantage that is able to work with manipulators at singularities and manipulators with either less or more than six joints.

3.2.3. Offline Calculation of Inverse Kinematics Based on Reachability Analysis

Our approach to solve the kinematic problem, is to use forward kinematics for calculating an initial position for both arms and then to compute the inverse kinematics, in a specific range around the initial position, using the Robotics toolbox [12] and the DH-parameters of NAO's arms (Tables: 3.1, 3.2). Figure 3.8 and 3.9 illustrate the kinematic chains of both arms as are calculated in the Robotics toolbox using the DH parameters mentioned above (Tables: 3.1, 3.2).

After creating both kinematic chains in the Robotics toolbox, we calculate the position of both end-effectors with the maximum x distance that the robot is able to reach the metallophone. We put the robot in the initial position, we attached the beaters in its hands and manually we gave the appropriate angles to the motors in order to reach 2 metal bars in the maximum possible distance in the x-axis. Then, using the function fkine() in the Robotics toolbox we found the 3D coordinates of this point and using this point as a reference point, in a double for loop, we calculate the inverse kinematics, using the function ikine(), in a range of 6cm in the x-axis and 18cm in th y-axis for both arms.

The kinematic analysis deployed only about the position on the x and y axes since we assume that the height (z-axis) is fixed and we do not care about the orientation (a_x, a_y, a_z) of the end-effector. At the end we save a matrix consisting of 6 columns and each line contains the x and y coordinates, in relation with NAO's body, and the angles in rads for the kinematic chain. Then, in the online phase of our system, NAO observe the 3D position of the musical instrument and then using these coordinates it searches for the pre-calculated kinematics and it moves its arms using the matches angles in the saved matrix.

However, even if we achieve high accuracy in the pose estimation phase, NAO's actuators are not highly accurate. Hence, even if the robot knows where to move its arms (end-effector) the accuracy of the motors is

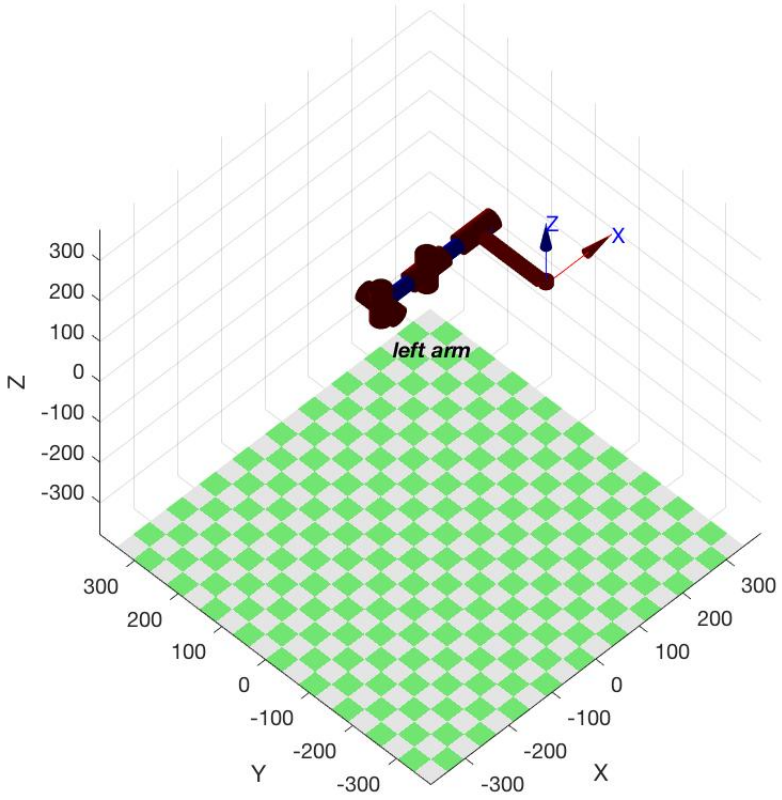


Figure 3.8: Kinematic chain of NAO's left arm

not enough to place the end-effector in the corresponding position. We analyze how we encounter with this problem in the next section.

3.3. Monocular Vision Based Control

We have already shown how we managed to get the pose of the metallophone in relation with the robot's base frame (torso). We have also calculated the inverse kinematics for our kinematic chain (arms holding the beaters), however, the robot was not always able to hit the metal bars accurately. In this section we explain how we approach this issue by implementing visual-servoing techniques.

3.3.1. Visual-servoing approach

In our approach, we exploit both the 3D information extracted from the pose estimation phase and the 2D information by detecting the head of the beater in the 2D space. Initially, the robot gets the 3D position of the center of the metallophone in relation with its base frame, from our vision approach. Then, it places the beaters above the red metal bars (Mi notes) which are -80cm and 80cm from the center point, using the pre-calculated inverse kinematics. Then, the beaters are somewhere above the red metal bars, but with a small random offset (error). For example, in Figure 3.10, the head of the beater should be inside the red circle (desired position). Assuming that the beaters are in the appropriate depth, we use the visual-servoing technique to correct the error in the z and y axes. The desired position of the beaters is 2cm above the red metal bars. More specifically, we use the projectPoints function of the openCV liblary to project on the image plane the desired position of the beaters as you can see in Figure 3.10. The projectPoints function takes as input the extrinsic parameters (R,t) and a 3D point, and it returns the 2D projection of this point. In this way, we project the points (-8, 0, 2) and (8, 0, 2).

Then we detect the actual position of the beaters and we use image-based visual-servoing to eliminate the error between the actual and the desired position of the beater. Figure 3.10 depicts the desired position

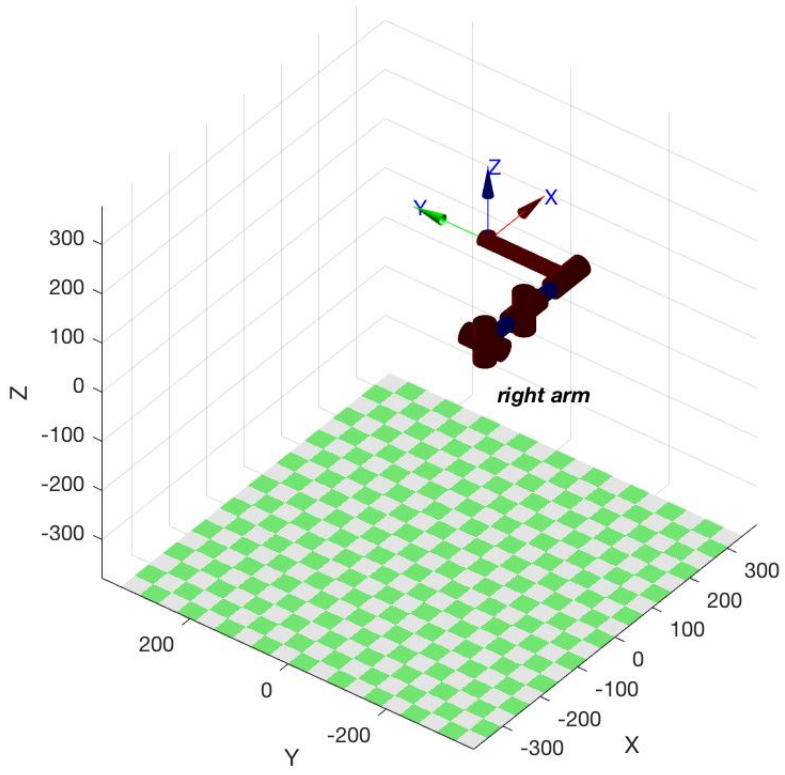


Figure 3.9: Kinematic chain of NAO's right arm

is red circle and the actual is black circle. More specifically, for each arm, we firstly fix the error in the z-axis (height) and then we fix the error in the y-axis.

Detection of the beater

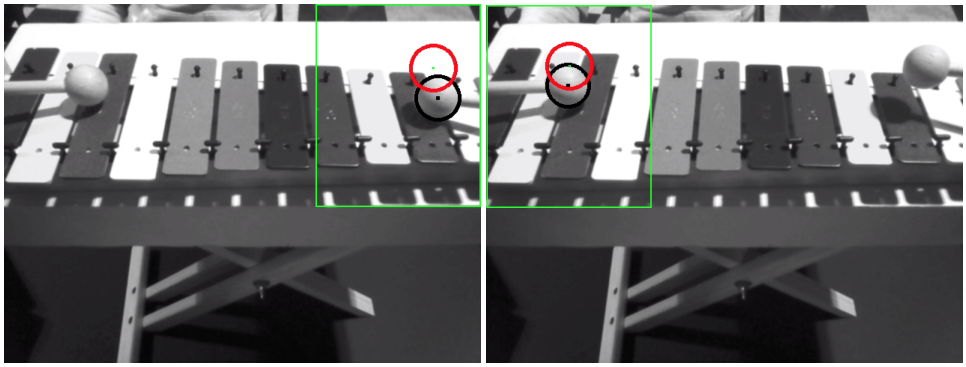
For detecting the actual position of the beater we use the circle Hough Transform which is a basic technique used in Digital Image Processing for detecting circular objects in a digital image. More specifically, we use the HoughCircles function provided by the OpenCV library, based on the Hough gradient method [53]. The aforementioned function takes as input the image that we want to detect a circle on, and other parameters which must be selected wisely. To make the detection of circle more robust, we create a search window in the 2D image, around the position that we expect to find the beaters.

Fixing the error in the z-axis

In our kinematic analysis we assume that the arms are in a specific height, consequently we cannot fix this error by adjusting the inverse kinematics parameters. In a while loop, we compare the pixel position of the desired and the actual position of the beater and the robot moves its arm up or down by adjusting the shoulder motor in pitch axis. In each loop the robot adjust its motor by one degree and it stops when the difference between the 3D observed actual position and 2D actual position of the beater is smaller than a set threshold.

Fixing the error in the y-axis

Again in a while loop, we compare the pixel position of the desired and the actual position of the beater and the robot adds or subtract 1mm in the y axis of the inverse kinematics. In each loop the robot adjust the x, y coordinates of the inverse kinematics and it moves its arm left or right until the difference of the actual and the desired position is smaller than a set threshold.



(a) Right Arm (b) Left Arm

Figure 3.10: Vision-Based Control

3.3.2. Technical implementation

This project consists of a real-time application and three offline applications (Figure 3.11). Starting with the offline applications, we developed in python a program for calibrating robot's camera using the opeCV library [4]. Moreover, our work contains a program in python for the key-frames acquisition and a Matlab code in which using the Robotics Toolbox [12] we create the kinematics chains and we calculate the inverse kinematics.

The online application consists of four components. All of these are developed in python. First, in a 2D image matching phase, the robot maps key-frames with current image. Then it estimates the pose of the object. After that it uses the pre-calculated kinematics to move its arms, and finally it automatically validates the computed IK configurations based on a vision-based robot control approach and adapts its arm configurations if it is necessary. An up-to date version of Musical NAO is available on this Github repository: https://github.com/papakonst/Musical-NAO, along with a demonstration video of our approach.

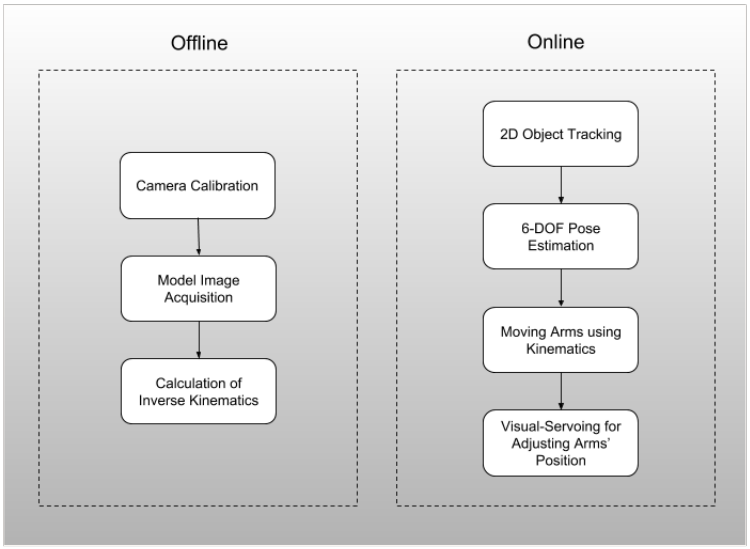


Figure 3.11: Overview of the proposed system.

3.4. Real-Time Application for playing the metallophone

Initially the robot goes to its initial position and asks the user to move the metallophone in front of it. Then the robot captures the metallophone and search for the best matching key-frame. Then, the pose estimation phase returns information to the robot about the position and orientation of the instrument in 3D space.

Especially, the robot knows how far is the metallophone (x-axis) in relation to its body and how far is the center of the metallophone in relation to robot's camera center (y-axis). In addition, the pose estimation returns the distance between the metallophone and the camera in the z-axis.

NAO is able to hit (reach) all metal bars in a specific range of the aforementioned 3D dimensions, because of its arms work space. Consequently we set some limits in which the robot is able to hit all metal bars. In order to create an area in which the robot can reach all metal bars and the robot can detect the beaters without moving its head, we set some limits. In the x-axis the we set a limit between 15cm to 17.5cm. This is a distance in x-axis in which the robot can reach all metal bars. In the y-axis the center of the metallophone cannot be more or less than 1cm away of the center of the camera. Finally, the rotation of the metallophone around the z-axis cannot be more than 5 degrees. In our approach, we assume that the metallophone is in a fixed height, thus, we do not use that dimension.

4

Evaluating the robustness of our approach

This chapter elaborates on the methodology to answer our research questions. In order to evaluate the effectiveness and the efficiency of our approach for making NAO robot play the metallophone, we evaluated our approach in our lab by creating different conditions which can affect the procedure of tracking and playing the metallophone. We will first elaborate on a detailed description of the conditions under which we are going to do our experiments. Then we present the experimental setup, the evaluated conditions, and the evaluation metrics.

4.1. Tracking the musical instrument in different light conditions

In this experiment we evaluate the robustness of our vision approach in different light conditions. We want to make the robot detect the metallophone independently of the light conditions. However, the on-board cameras of the robot have some limitations. Robot's cameras do not operate well in extreme light conditions (too dark or too bright). There are unlimited combinations of illumination changes, hence we are not able to test all possible situations. From our experience during the development of the application, we observed that one parameter that affects our approach is the position of the source of the light (light direction), as well as the source of light (natural light, artificial light). Hence, we are going to place the musical instrument in four different places inside the lab with only artificial lights on. Then we are planning to do the same with the lab windows opened in order to have the interference of the natural light.

4.1.1. Experimental setup

In this experiment we test two different scenarios of our approach. We use as a baseline our 2D matching approach with one key-frame and then we extend our 2D matching approach using 10 key-frames.

- **2D image tracking based on 1 key-frame:** In a preparation step, we create a key-frame with the (artificial) light source exactly above the metallophone.

- **2D image tracking based on 10 key-frames:** In a preparation step, we create 10 different key-frames. Four out of ten key-frames are taken for each direction of the artificial light (North, South, East, West), another four key-frames are created in the same manner but with natural light interference, one key-frame is created with the light source be exactly above the metallophone and the last one is created in darker light conditions.

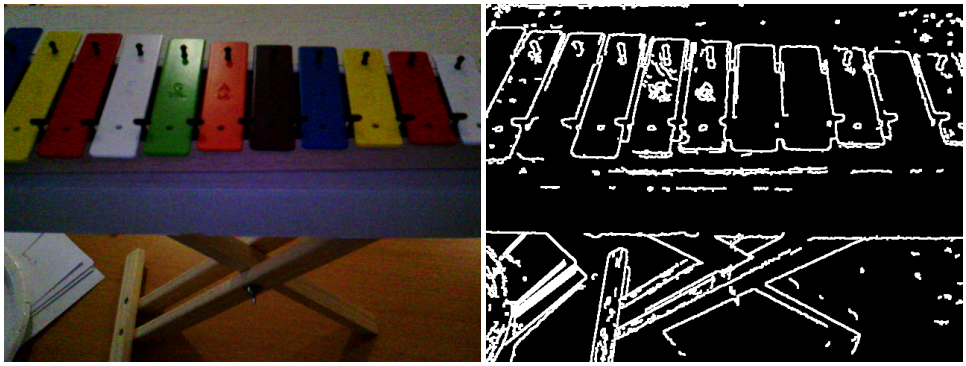
4.1.2. Measures

We evaluate these different scenarios by making the NAO track the metallophone 30 times in 15 different random positions inside the lab (15 only artificial lights, 15 with natural light). We measure how many times out of 30, the robot managed to track the metallophone for each scenario.

4.1.3. Results

- **2D image matching based on 1 key-frame:** The robot could detect and track the metallophone inside the lab, with 53.33% accuracy using only one key-frame.

- **2D image matching based on 10 key-frames:** The robot could detect and track the metallophone inside the lab, with 93.33% accuracy using ten key-frames.



(a) Input image in dark conditions (b) Processed input image in dark conditions

Figure 4.1: Input image in dark condition

Figure 4.1 depicts an example of an extremely dark, for robot's camera, environment in which the robot could not detect and track the metallophone using 10 key-frames.

4.2. Transferability of our software to different robots - differences between robots' motors

In this experiment we evaluate our Position-Based Visual Servoing implementation in order to observe if it is possible to integrate our software in different robots.

4.2.1. Experimental setup

In this experiment we work in a specific position iside the lab with good light conditions and using 10 key-frames for detecting the metallophone. Two different NAO robots are available in our lab, a version 5 (V5) and a version 6 (V6). We test our approach on them as follows:

In a preparation step we deactivate the visual-servoing algorithm from our code and adjust the kinematic configurations for the V6 robot. Then we run our code with this parameterization in the V5 robot. Then the V5 robot tries to hit all of the notes and we measure the successful hits. We do the same procedure by adjusting the kinematic configurations for the V5 robot. Then we activate our visual-servoing approach and we use the default kinematic configurations. After that, we run our code for each robot, the robot tries to hit all of the notes and we measure the successful hits. We repeat this procedure 10 times.

4.2.2. Measures

As a measure we use the success rate of the hits. A successful hit is considered a movement in which the head of the beater hits only the appropriate metal bar. The robot hits all the metal bars from left to right and the observer records the successful hits.

4.2.3. Results

Adjusted kinematic configurations for V6 Robot, deactivated visual-servoing

We observed the successful hits after deactivating the visual-servoing implementation and keeping the kinematic configurations adjusted for the V6 robot. The V5 robot hit the metal bars with 50% accuracy.

Adjusted kinematic configurations for V5 Robot, deactivated visual-servoing

We observed the successful hits after deactivating the visual-servoing implementation and keeping the kinematic configurations adjusted for the V5 robot. The V6 robot hit the metal bars with 50% accuracy.

Default kinematic configurations, activated visual-servoing

By activating the visual-servoing implementation and using the default kinematic configurations both robots hit the metal bars with 100% accuracy.

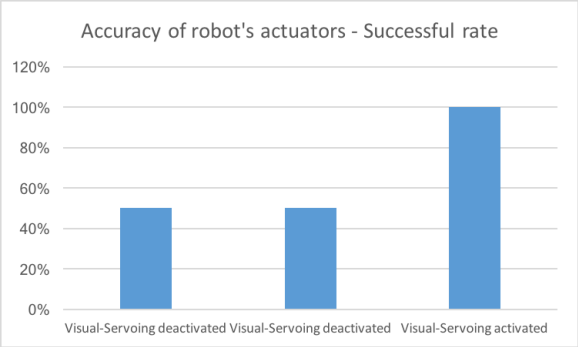


Figure 4.2: Success rate with visual-servoing activated and deactivated on V5 and V6 robots

We performed a qualitative experiment placing the robot in an environment with normal light conditions. However, the visual-servoing part contains an extra vision task which is the detection of the beater (circle). We only tested the robustness, against light conditions, of the pose estimation task, however in the final experiment the visual-servoing part will also be tested in different light conditions.

4.3. Robustness against inaccurate grasping of the beater

In this experiment we evaluate our Position-Based Visual Servoing implementation in order to evaluate whether it can handle the error of an inaccurate beater grasping. On the beaters' grips, there is a sign (yellow line) which must be aligned with the middle finger of NAO's hand (Figure 4.3). However, it is possible for the robot to grasp the beaters with a small offset. We expect that our approach can eliminate this error.

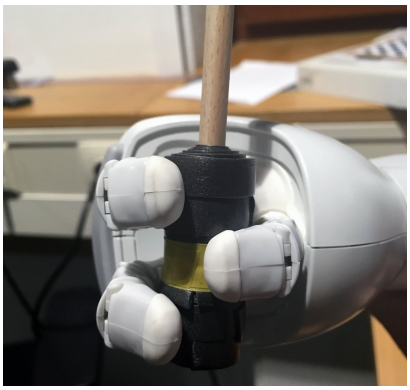


Figure 4.3: This is an example of a correct placement for the beater

4.3.1. Experimental setup

During this experiment we work in a specific position inside the lab with good light conditions and using 10 key-frames for tracking the metallophone. Working on a V6 Robot, in a preparation step we deactivate the visual-servoing algorithm from our code and adjust the kinematic configurations for this robot. Then we make a demonstration with this parameterization, and we purposely give the beaters to the robot with a random offset. The robot hits once all of the metal bars and we measure the successful hits. We repeat this procedure 10 times. Then we activate our visual-servoing approach and we use the default kinematic configurations. Then we make a demonstration, and we give in purpose the beaters to the robot with a random offset again. The robot hits once all of the metal bars and we measure the successful hits as defined in experiment 2. Then we repeat the same procedure for the V5 robot.

4.3.2. Measures

During the demonstration, the experimenter observes the robot's hits. As a measure we use the success rate of the hits.

4.3.3. Results

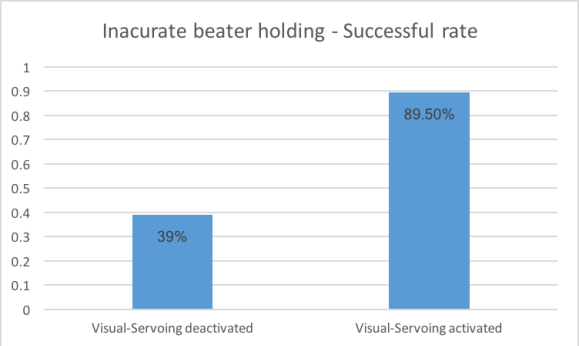


Figure 4.4: Success rate with visual-servoing activated and deactivated

We observed that the robot could correct the error in most cases, however, when the offset of the beater was very high the robot was not able to hit the appropriate metal bar. In conclusion, the robot could operate well when the offset was less than approximately $\pm 1.5cm$.

4.4. Final evaluation with Human-robot interaction

We want to evaluate whether a person without any prior experience is able to setup NAO and the metallophone such that NAO can start playing a song on it. The aim of this experiment is to evaluate the task performance of a user who is asked to (i) put the metallophone in front of the robot so it can play it and (ii) put the sticks for playing the metallophone in the NAO's hands.

4.4.1. Experimental Setup

For this experiment, we choose a random place inside the lab in which we are going to place the robot and the musical instrument. Then we power up the robot, we put it in a stand by mode and we place the metallophone in a random position in front of the robot.

In this experiment we divide the users into two different groups. The users from the first group will place the metallophone in the appropriate for the robot position by first receiving only auditory feedback and then by receiving both auditory and visual feedback. The users from the second group are provided first with both auditory and visual feedback and then with only auditory feedback. We want to test the effect of visual cues in the interaction phase. All the detailed steps of the experimental procedure are depicted in Figure 4.5.

Before the experiment starts, the participants reads a detailed instruction sheet about the experimental procedure and the actions that they need to do. When the participants are ready, the experimenter runs the code, and then the participants only interact with the robot.

4.4.2. Measures

We evaluate the two different approaches by measuring the time needed for task completion and by counting the attempts of the participants for each task. The timer starts when the application starts running. The participant will try to place the metallophone in a position that is good for the robot within a time limit of 5 minutes. If the attempt is not successful, then the participant can try again. We follow the same procedure for the placing of the beaters. After that, we investigate which is the fastest method and we check the usability (complexity) by comparing the completion time of the two approaches and evaluating the responses of the Likert scale survey.

4.4.3. Results

Each participant had 4 attempts to complete the task. In an overall of 80 demonstrations, based on 20 participants, the robot tracked the metallophone, during the interaction phase, 78 times out of 80 and the participants failed to give the beaters accurately 6 times out of 80. Regarding the successfully finished interaction phases, the completing times were always much lower than the set limit of 5 minutes. In the graph (Figure 4.7) you can see the average time spent per attempt. We observed that the preparation task was getting more easy for the participants, after a few interactions. Moreover, according to the Likert scale, most of the participants were satisfied with the frequency of the feedback provided by the robot and they found the placement of

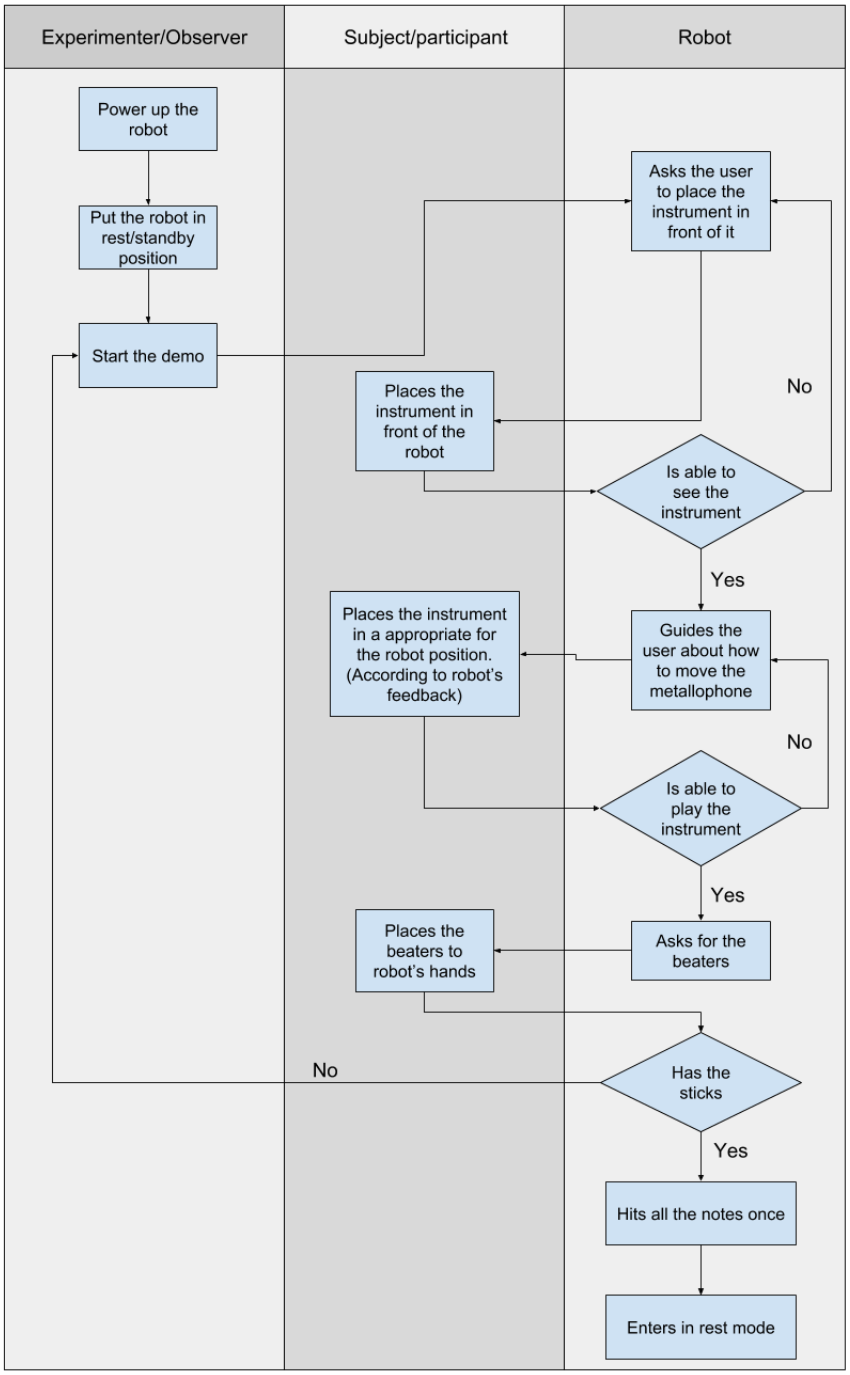


Figure 4.5: Flowchart of the experimental procedure

the metallophone and beaters simple enough. Concerning the visual feedback, the opinions differ. Approximately half of the participants said that the visual feedback helped them to place the metallophone correctly and the other half did not find it helpful.

Regarding the attempts that the participants completed the preparation task successfully, we evaluate the successful hits that the robot achieve after hitting all the notes for one time. In total the robot hit the metallophone with 91.45% accuracy and the distribution of successful hits is illustrated in Figure 4.6. The

failed hits were due to three main reasons. In some cases the visual-servoing approach failed to fix the error. Moreover, some of the participants had difficulties in placing the beaters accurately in robot's hands. The final reason has to do with extreme cases in which the metallophone was rotated slightly below the acceptable limit and the robot hit some metal bars inaccurately.

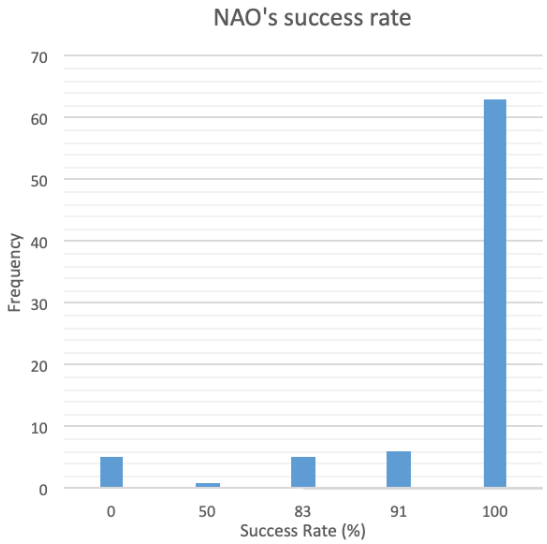


Figure 4.6: Percentages of successful hits among 80 attempts to play

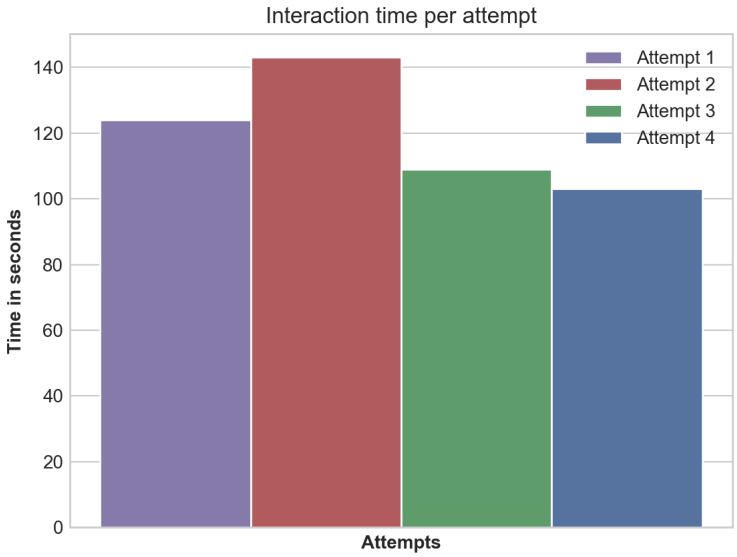


Figure 4.7: Completing time per attempt

5

Discussions and Conclusions

The primary goal of this research was the development of a software for making NAO play the metallophone. We aimed to make this software accessible and usable for everyone who owns a NAO robot, hence in our research, we focused on the robustness of our approach. More specifically, we investigated the robustness of our vision approach in different light conditions as well as the integration of our code on different NAOs. In this section, we reflect on the results and discuss our findings. Furthermore, we elaborate on the limitations of our study, and we present future research directions based on our findings.

5.1. Findings

5.1.1. Pose estimation based on key-points

For the pose estimation of the metallophone we chose a tracking by detection method based on key-frames. In this approach, although a preparation phase for creating multiple key-frames with different illumination is necessary, we observed that the pose estimation in different light conditions can be achieved with high accuracy.

5.1.2. Pre-Calculation of Inverse Kinematics

For the kinematic part of our project we tried different approaches. At first we used the kinematics provided by the manufacturer [1], however, due to singularities, the robot had many failures in its movements. Then, based on Kofinas et al.'s [33] work and using the Robotcs Toolbox in Matlab, we created the kinematic chains of NAO's arms and we calculated the DH parameters. Then, using the ikine() function of the Robotcs Toolbox we pre-calculated the inverse kinematics for a large number of possible reachable positions of the end-effector. To handle with the kinematic part, this approach has been successful, and the result has been very accurate and fast.

5.1.3. Visual-Servoing Based on 3D Pose estimation and 2D Image Detection

Even though in theory the pose estimation and the kinematics part should work, we observed that we had to adjust some configurations in the kinematics dynamically, for making the robot hit the metallophone accurately. Even if the inverse kinematics were well computed, due to lack of accuracy in NAO's motors, the end-effector was always placed some millimeters away of the desired position. Hence, for each robot, we had to adjust that offset manually which made the integration of the same code in different robots impossible.

Applying our visual-servoing approach, we made the robot detect and correct this offset automatically. We evaluated our approach in different light conditions and we observed a low failure rate. Moreover, we confirmed that we can integrate the same code in different robots and we achieved a robust hand-eye coordination without using any markers attached on the robot's arms.

5.1.4. Human Assistance for Placing the metallophone and the beaters

In our approach, the robot moves only its arms for playing the metallophone, hence we created a human robot interaction phase in which a human, guided by the robot, places the metallophone in a good position in front of it. Moreover, in our approach, the robot is not able to grasp the beaters on its own, thus, a human also needs to place the beaters into the robot's arms.

For this interaction phase, we created an instruction sheet which helps users to understand the procedure better. Then, we performed an experiment with 20 participants, and we found that it is fairly easy for anyone to complete this task. Some participants faced some difficulties at their first attempts, indicating there is a short learning phase, but in total all of the participant succeeded it.

5.2. Limitations and Future Research

In this section we discuss some of the limitations of our study. In general, in our work, the robot is limited to play only the specific metallophone that we used at a specific height. Moreover, in this project, we focused on the robot's arms movements and the robot is not able to autonomously place itself, by using its legs, behind the metallophone. In addition the head of the robot is considered to be in a fixed position and orientation. Particularly, we operate with both angles of head motors set to zero degrees. This results in more limited object tracking procedure. There are some ideas that we would like to develop. However, we focused more in making the robot estimate the metallophone robustly and play it by moving its arms with high precision. Future work concerns the enrichment of NAO's movements, according to its visual perception. The following idea could be tested:

- It could be interesting to make NAO move its head while it tries to detect the pose of the metallophone. In this way, NAO will have a wider field of view.
- This extra adjustment could be followed by a walking phase in which the robot, autonomously, could take an appropriate place behind the metallophone. However, to our knowledge, the precision of NAO's walking steps is not high enough to complete this task in reasonable time. A good idea could be to develop an walking approach, in which the robot will make quite small and precise steps.
- In addition, the robot could adjust its body position in the z-axis in order to play the metallophone placed at different heights. In our work, we adjust the motors of robot's shoulders, in order to eliminate small offsets (approximately 2cm) in z-axis. In cases which the metallophone is at a fixed height, our approach performs well. However, if someone wants to make NAO can have more options about the height of the metallophone it is important to make it adjust its body position in the z-axis.

Apart from placing the metallophone in a reachable position, the robot needs someone to place the beaters in its hands. It could be interesting if the robot could detect and grasp by itself the beaters.

Concerning our pose estimation approach, the offline manual procedure of creating the key-frames and marking the key-points could be replaced by a supervised learning procedure. In addition, an idea that we did not develop, is to make the robot create automatically additional key-frames, during the online pose estimation procedure. More specifically, when the robot detects a view of the metallophone that it has significant changes (i.e in illumination) based on its current key-frames, it could capture more frames online and use it as a key-frame in the future. For example, let as assume that the robot is placed in an environment in which its best key-frame has 20 matched points with the current frame. The robot could automatically save the current frame and transform it into a key-frame. In the next loop of the online tracking phase, the robot could replace the previous (old) key-frame with the new one. Consequently, the matches between the key-frame and the current frame will increase succeeding a more robust tracking procedure.

In our work, the pose estimation procedure is deployed before the robot starts playing. When the robot detects the instrument in a good position, it assigns the corresponding kinematics for each note. After that, the robot starts playing and it is not able to update the kinematics configurations in case of a change in the instrument's pose. In future work, the instrument's pose could be updated continuously.

Finally, as far as the adjustments on kinematic configurations are concerned, we currently rely on the detection of the end-effector in 2D. Even if we achieve satisfactory results, in our experiments, the robot is not able to perceive the depth (x-axis) of the end-effector. This also affects the estimated position in the other two dimensions y and z axes.

5.3. Conclusion

The high interest in humanoid robots in the education and entertainment field, including music, leads to a need for developing relative applications which can be easily used by everyone. Instrument-playing humanoid robots are used in both education and entertainment fields. In this work, we have engineered an efficient and robust approach to enable the humanoid robot NAO play the metallophone. NAO is a commercially available humanoid robot and thus very popular. We aimed to create an application that can be used

by everyone who owns a NAO and also by people who are not robot expert. Moreover, our work was used by the Speelklok Museum in Utrecht. In Museum Speelklok, there are some of the first self-playing musical instruments. Guides in the museum, uses our application to entertain children by playing songs. In addition, there are sessions in which the visitors (mainly children), write their own songs and then NAO plays them.

Our work had to be easily integrated in different robots and perform well in different light conditions. To achieve this task, we developed a model-based pose estimation approach for detecting the position and orientation of the instrument. The robustness of our system relies on captured key-frames which contain a view of the metallophone in different illumination conditions. We used pre-calculated forward kinematics to avoid singularities and make robot's moves more robust. Moreover, to eliminate variances in robot actuators, we developed a vision-based robot control approach in which the robot automatically validates the computed inverse kinematics configurations and adapts its arm configurations if it is necessary. In this vision-based robot control approach, we proposed a simple and efficient detection of the end-effector, based on natural features and the estimated pose of the musical instrument. In this way the robot achieved high accuracy in its movements which is necessary for playing the metallophone. Moreover, we developed a human robot interaction phase in which a human places the metallophone in an appropriate for the robot position, according to robot's auditory feedback.

We evaluated the robustness of our vision approach by creating different illumination conditions. Moreover, we integrated our software on a version 5 and a version 6 robot and we evaluated the success rate of playing the metallophone. Finally, we evaluated the simplicity of our application by deploying an experiment in which users had to place the metallophone and the beaters in the appropriate place according to robot's feedback.

Our main results reveal that, our vision approach is robust in different illumination and therefore our application can be used in different environments. We achieve high accuracy in robot's hits due to the vision-based control approach that we presented as well as our software can be easily integrated into different NAO robots. In addition, in our vision-based control approach, no markers need to be attached on the robot. As far as the interaction phase is concerned, we observed that users without experience in robotics can use this application without issues.

Concluding, we developed an application ready to be used by everyone who has a NAO and the metallophone used in this project. The outcome of this work proves that despite NAO's limitations, it is possible to make the popular humanoid NAO to play a musical instrument like the metallophone, leading to directions for future research in the field of instrument playing humanoids.

A

Instructions for preparing the metallophone (speech and visual cues)

We have developed software for a NAO robot that enables it to play the metallophone. We ask you to prepare the setup of the metallophone for the NAO. We have already powered up the robot, as well as we have set it in an initial position, so you do not have to move or touch the robot itself. Your task is to put the metallophone in a spot in front of the NAO so it can see it and it will be able to play the metallophone. The metallophone is already placed on its base and you are allowed to move the whole base.

1. The robot will ask you to place the metallophone in front of it, then it will try to find the instrument and when it is done, it will inform you.

Figure A.1: Example of metallophone's position in front of NAO

2. Then the robot will give you instructions (feedback) by speech about the (position of the metallophone and you need to follow them up in order to place the metallophone in an appropriate for the robot position. Moreover, the robot is going to give you visual feedback about the movement that you have to do. On the computer's screen, you can see what the robot sees by its camera like in the Figure below:

   The green cross is in a fixed position in the center of the window frame, and the blue cross represents the current position of the metallophone. Your task is to move the metallophone until the two crosses are as close as possible.

3. When the metallophone is in a good for the robot position, then it will inform you by saying that "The metallophone is in a good position, stop moving it". Then the robot will ask you to press and hold its head button in order to move into the next step.

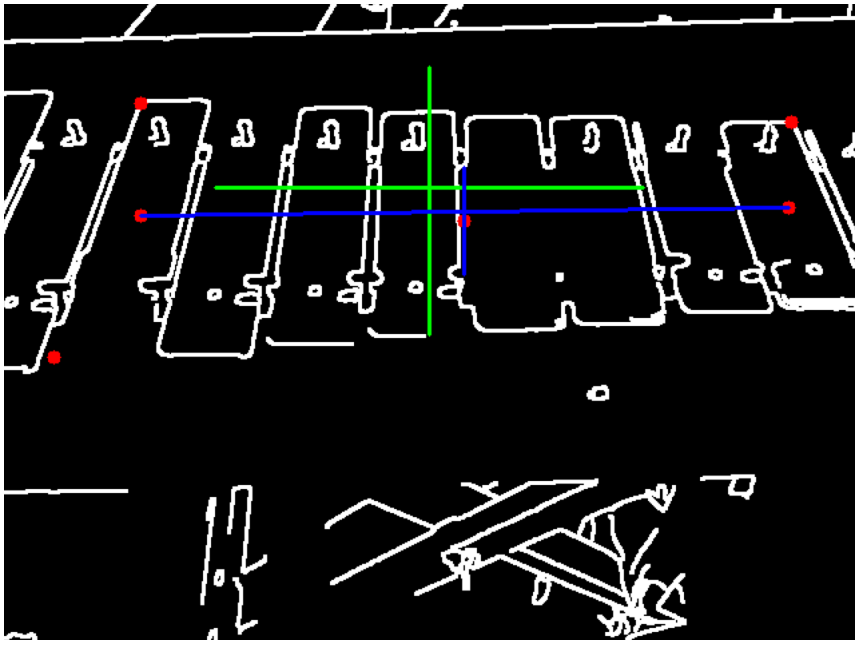


Figure A.2: Robot's view from lower camera

4. Then the robot will ask you to place the beaters in its hands. Each beater has a yellow line which must be aligned with the middle finger of NAO's robot.

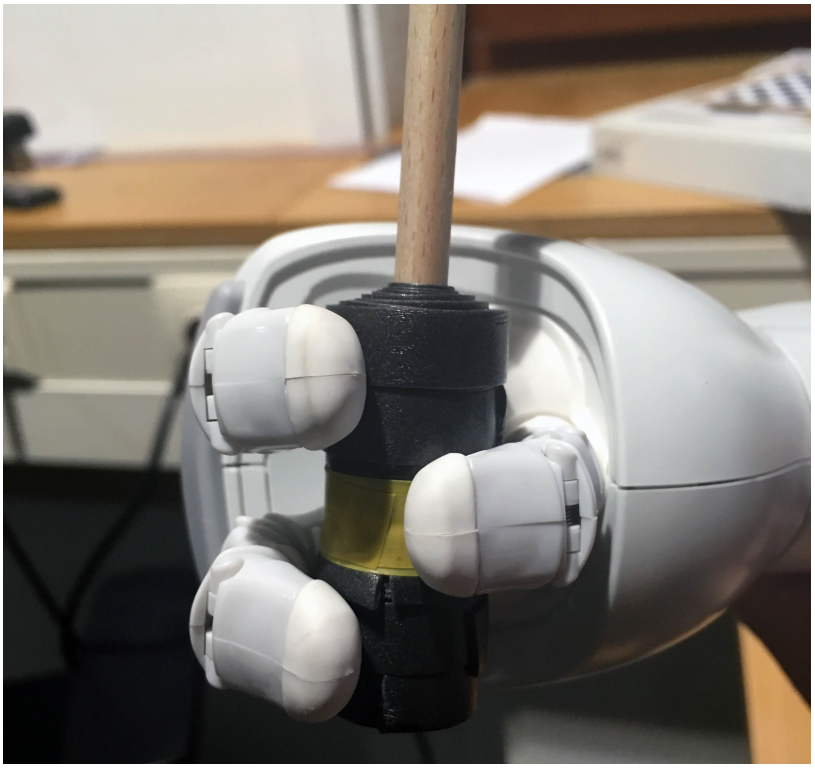


Figure A.3: This is an example of a correct placement for the beater

B

Post-experiment questionnaire

The feedback that the robot provided:

- Should be repeated more often
- Is repeated often enough
- Should not be repeated so often

It was clear to me what the robot said:

- Strongly disagree
- Disagree
- Neither agree nor disagree
- Agree
- Strongly agree

The visual feedback was clear to me:

- Strongly disagree
- Disagree
- Neither agree nor disagree
- Agree
- Strongly agree

Rate the complexity of placing the metallophone:

- Very Simple
- Simple
- Neither simple nor difficult
- Difficult
- Very difficult

Rate the complexity of placing the beaters:

- Very Simple
- Simple
- Neither simple nor difficult
- Difficult
- Very difficult

Please provide some additional comments that you have on the tasks you were asked to perform: Thank you for kindly participating in this questionnaire.

Bibliography

[1] Robotics Aldebaran. Nao documentation. *2012, only available online: www.aldebaran-robotics.com/documentation*.

[2] Alyssa M. Batula and E. Kim. Youngmoo. Development of a mini-humanoid pianist. *Humanoid Robots (Humanoids), 2010*, 10th IEEE-RAS International Conference on. IEEE, 2010.

[3] Oliver Birbach, Berthold Bäuml, and Udo Frese. Automatic and self-contained calibration of a multi-sensorial humanoid's upper body. *ICRA. 2012*.

[4] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[5] Gary Bradski, Adrian Kaehler, and Vadim Pisarevsky. Learning-based computer vision with intel's open source computer vision library. *Intel technology journal 9.2 (2005)*.

[6] Cynthia Breazeal. Emotion and sociable humanoid robots. *International Journal of Human-Computer Studies*, 59.1-2 (2003): 119-155.

[7] John. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence 6 (1986): 679-698*.

[8] K. Chida, I. Okuma, S. Isoda, Y. Saisu, K. Wakamatsu, K. Nishikawa, J. Solis, H. Takanobu, and A. Takanishi. Development of a new anthropomorphic flutist robot wf-4. *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. Vol. 1. IEEE*, (2004).

[9] Changhyun Choi and Henrik I. Christensen. Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *The International Journal of Robotics Research 31.4 (2012): 498-519*, .

[10] Changhyun Choi and Henrik I. Christensen. Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation. *Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010.*, .

[11] Peter. Corke. Robotics, vision and control: Fundamental algorithms in matlab® second, completely revised. *Vol. 118. Springer, 2017.*, .

[12] Peter I. Corke. A robotics toolbox for matlab. *IEEE Robotics Automation Magazine 3.1 (1996): 24-32.*, .

[13] Maier Daniel, Ramin Zohouri, and Maren Bennewitz. Using visual and auditory feedback for instrument-playing humanoids. *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on. IEEE*, (2014).

[14] Daniel F. Dementhon and Larry S. Davis. Model-based object pose in 25 lines of code. *International journal of computer vision 15.1-2 (1995): 123-141*.

[15] Jacques. Denavit. A kinematic notation for low pair mechanisms based on matrices. *ASME J. Appl. Mech. 22 (1955): 215-221*.

[16] Tom Drummond and Roberto Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on pattern analysis and machine intelligence 24.7 (2002): 932-946*.

[17] Mei Fang, GuangXue Yue, and QingCang Yu. The study on an application of otsu method in canny operator. *Proceedings. The 2009 International Symposium on Information Processing (ISIP 2009). Academy Publisher*, (2009).

[18] Olivier Faugeras and OLIVIER AUTOR FAUGERAS. Three-dimensional computer vision: a geometric viewpoint. *MIT press, 1993*.

[19] C. Fennema, A. Hanson, E. Riseman, J. R. Beveridge, and R. Kumar. Model-directed mobile robot navigation. *IEEE Transactions on Systems, Man, and Cybernetics 20.6 (1990): 1352-1369.*

[20] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24.6 (1981): 381-395.*

[21] I. Fujimoto, T. Matsumoto, P. R. S. De Silva, M. Kobayashi, and M. Higashi. Study on an assistive robot for improving imitation skill of children with autism. *International Conference on Social Robotics. Springer, Berlin, Heidelberg*, (2010).

[22] D. Gonzalez-Aguirre, M. Vollert, T. Asfour, and R. Dillmann. Robust real-time 6d active visual localization for humanoid robots. *Robotics and Automation (ICRA), 2014 IEEE International Conference on. IEEE*, (2014).

[23] Iryna Gordon and David G. Lowe. What and where: 3d object recognition with accurate pose. *Toward category-level object recognition. Springer, Berlin, Heidelberg, 2006. 67-82.*

[24] C. Graf, A. Härtl, T. Röfer, and T. Laue. A robust closed-loop gait for the standard platform league humanoid. *In Proceedings of the Fourth Workshop on Humanoid Soccer Robots in conjunction with the (pp. 30-37)*, (2009).

[25] B. M. Haralick, C. N. Lee, K. Ottenberg, and M. Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *International journal of computer vision 13.3 (1994): 331-356.*

[26] Chris. Harris. Tracking with rigid models. *Active vision (1992): 59-73.*

[27] Richard Scheunemann Hartenberg and Jacques Denavit. Kinematic synthesis of linkages. *McGraw-Hill, 1964.*

[28] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision. *Cambridge university press, 2003.*

[29] Uwe Hubert, Jorg Stuckler, and Sven Behnke. Monocular model-based 3d tracking of rigid objects: A survey. *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on. IEEE, 2012.*

[30] Seth Hutchinson, Gregory D. Hager, and Peter I. Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation 12.5 (1996): 651-670.*

[31] Reza N. Jazar. Theory of applied robotics: kinematics, dynamics, and control. *Springer Science Business Media, 2010.*

[32] Y. E. Kim, A. M. Batula, D. Grunberg, D. M. Lofaro, J. Oh, and P. Y. Oh. Developing humanoids for musical interaction. *International Conference on Intelligent Robots and Systems*, (2010).

[33] Nikos Kofinas, Emmanouil Orfanoudakis, and Michail G. Complete analytical inverse kinematics for nao. *Proceedings of the 13th International Conference on Autonomous Robot Systems and Competitions (ROBOTICA). Vol. 13*, (2013).

[34] Akio Kosaka and Avinash C. Kak. Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. *CVGIP: Image understanding 56.3 (1992): 271-329.*

[35] Ville Kyrki and Danica Kragic. Integration of model-based and model-free cues for visual object tracking in 3d. *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. IEEE, 2005.*

[36] Vincent Lepetit and Pascal Fua. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends® in Computer Graphics and Vision 1.1 (2005): 1-89.*

[37] David G. Lowe. Distinctive image features from scale-invariant keypoints. *nternational journal of computer vision 60.2 (2004): 91-110.*

[38] J. Michael. McCarthy. Introduction to theoretical kinematics. *MIT press, 1990.*

[39] P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade. Gpu-accelerated real-time 3d tracking for humanoid locomotion and stair climbing. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on. IEEE*, (2007).

[40] T. Mizumoto, H. Tsujino, T. Takahashi, T. Ogata, and H. G. Okuno. Thereminist robot: Development of a robot theremin player with feedforward and feedback arm control based on a theremin's pitch model. *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, (2009).

[41] Jorge J. Moré. The levenberg-marquardt algorithm: implementation and theory. *Numerical analysis. Springer, Berlin, Heidelberg, 1978. 105-116.*

[42] Amine Abou Moughlbay, Enric Cervera, and Philippe Martinet. Real-time model based visual servoing tasks on a humanoid robot. *Intelligent Autonomous Systems 12. Springer, Berlin, Heidelberg, 2013. 321-333.*

[43] Nobuyuki Otsu. The study on an application of otsu method in canny operator. *A threshold selection method from gray-level histograms.*

[44] Nikolaos P. Papanikolopoulos, Pradeep K. Khosla, and Takeo Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. *IEEE transactions on robotics and automation 9.1 (1993): 14-35.*

[45] Richard P. Paul. Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators. *Richard Paul, 1981.*

[46] Muriel Pressigout and Eric Marchand. Real-time 3d model-based tracking: Combining edge and texture information. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on. IEEE, 2006.*

[47] Long Quan and Zhongdan Lan. Linear n-point camera pose determination. *IEEE Transactions on pattern analysis and machine intelligence 21.8 (1999): 774-780.*

[48] Daniel J. Ricks and Mark B. Colton. Trends and considerations in robot-assisted autism therapy. *Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE*, (2010).

[49] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. Vol. 2. IEEE*, (2005).

[50] Luca Vacchetti, Vincent Lepetit, , and Pascal Fua. Stable real-time 3d tracking using online and offline information. *IEEE transactions on pattern analysis and machine intelligence 26.10 (2004): 1385-1391*, .

[51] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Combining edge and texture information for real-time accurate 3d camera tracking. *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality. IEEE Computer Society, 2004.*, .

[52] G. Weinberg, T. Mallikarjuna, and A. Ramen. Interactive jamming with shimon: A social robotic musician. *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction. ACM*, (2009).

[53] H. K. Yuen, J. Princen, J. Illingworth, and J. Kittler. Comparative study of hough transform methods for circle finding. *Image and vision computing 8.1 (1990): 71-77.*

[54] Zhengyou. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence 22 (2000).*