Hamburg University of Technology
Vision Systems

Prof. Dr.-Ing. R.-R. Grigat

Joint and Camera Calibration for NAO Robot

Project Thesis

Adikari Appuhamillage Darshana Sanjeewan Adikari

April 16, 2022

Statutory Declaration

I, Adikari Appuhamillage Darshana Sanjeewan Adikari, born on 1992-12-20 in Colombo, Sri Lanka, hereby declare on oath that I compiled this master thesis, submitted to Hamburg University of Technology (TUHH), on my own. I only used the declared sources and auxiliaries.

Place and Date | Signature

Eidesstattliche Erklärung

Ich, Adikari Appuhamillage Darshana Sanjeewan Adikari, geboren am 1992-12-20 in Colombo, Sri Lanka, versichere hiermit an Eides statt, dass ich diese von mir bei der Technischen Universität Hamburg (TUHH) vorgelegte Masterarbeit selbstständig verfasst habe. Ich habe ausschließlich die angegebenen Quellen und Hilfsmittel benutzt.

Ort und Datum | Unterschrift

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robotics has evolved from caged industrial machines to aesthetically appealing and "smart" robots that aim to be friendly with humans. Their proposed roles range from being learning aids for children to caring the elderly and also to play soccer!.

The RoboCup was inaugurated 21 years ago (1997) in Japan as an annual competition in robotics with the vision of robots playing against FIFA champions in the year 2050. It also features logistics and rescue categories, highlighting the fact that robots should be able to handle complex environments opposed to highly structured industrial settings in order to successfully interact with humans in day-to-day life. Since 2008, the Robocup Soccer Standard Platform League (will be referred as RoboCup SPL in this thesis) has been using NAO, a 58cm tall humanoid robot by Softbank robotics.

This thesis presents the design, implementation and application of a Camera and Joint calibration process for the NAO, in the context of "HULKs", the SPL team of Hamburg University of Technology. Not only the technical challenges, but also factors like time to calibrate, logistics are also considered due to the varying venues of the world championship.

This thesis is organized as follows; The next sections discuss problem statement, potential reasons and solutions together with goals, limitation and scope. Next chapters cover milestones, literature review, implementation, Testing and Evaluation which also discuss results and finally the conclusion.

1.1 Problem Statement

Calibrating the two cameras of the NAO has been a prerequisite for obtaining good coordination of the robots and several methods has been derived at HULKs for the purpose of Camera Calibration (Intrinsic and Extrinsic) [1].

In recent years, with the aging NAO V5 robots of the team, it was observed that NAO's joints exhibits significant amount of backlash. In certain occasions, the origin

of joint position were also found to be shifted as much as $\pm 20^{\circ}$. For a joint such as the neck, this means the error of perception with head mounted cameras is easily in the magnitude of several meters.

Therefore, it has come to attention the need for automatic joint error detection and calibration as this also affects camera extrinsic calibration process.

In the past several teams has attempted automating this with less than satisfactory results and all known processes that appears to work well are manual methods.

1.1.1 Potential Reasons for Specified Problems

Prior to presentation of a solution, it is imperative to identify potential sources of failures experienced in the past. Based on the available research, observations and practical experience with calibrating the NAO robots, following hypotheses are presented regarding causes for previous failures and difficulty in calibration.

- Used calibration poses weren't optimal, resulting in incomplete coverage of the observable subspace (considering the joint error domain) which leads to poor calibration quality including lack of convergence and local minimum exits.
- While a global minimizer could improve the results, it is still not immune from observation noise which can distort the cost function such that solved global minima is not the real one [**?**, 2].
- As noted in team B-Human's research, joint backlash plays a crucial role. Therefore, captures should also contain information about direction of backlash or limit the scope of calibration assuming the robot's usable poses always load joints in the ways resulting same backlash effects [3].
- Camera based calibration also depend on intrinsic and extrinsic parameters; The former can be calibrated without reliance of robot's positioning, but the latter depends on calibrated joints causing a cyclic dependency which can be resolved by multiple cycles of calibration for joints and extrinsic parameters. This concern is already mentioned in Team Research Report of B-Human and their manual calibration procedure appears to address it in the above mentioned method [4].

1.2 Goals

The aim of this thesis is to develop a fully or semi-automatic joint and camera calibration. This includes theoretical observer modelling, optimal pose generation and the implementation of necessary tools, modules into HULKs NAO framework and the debug tooling (which will be extending the current calibration tools).

Since item 1 and 2 of section 1.1.1 are not covered well in most literature or only in industrial context (which use laser trackers), this thesis will pay more attention for importance of picking optimal poses, observability, reduction of ambiguities and improving calibration optimizers. Finally, the evaluation phase will be used to validate the hypotheses.

Below listed is a summarized set of goals. A detailed breakdown into milestones is presented in chapter 2.

1. Derive a process to determine suitability of sensors, poses, input data, etc. This may include observation modelling.
2. Investigate the suitability of on-board sensors based on the above mentioned process. At least some joint errors should be correctable.
3. Determine the effect of backlash and offsets, level of observation and other factors.

1.3 Limitations & Scope

This research is performed with the following constraints.

1. Intrinsic Camera calibration is not considered, as it doesn't depend on joints, enabling it to function well even without the intended improvements of this research.
2. Since calibration of joints of arms are not a concern for playing soccer, they won't be considered. **??** lists the joint numbers and labels referred in this thesis.
3. Joint and limb elasticity, or similar properties are not considered.
4. Only on-board sensors are considered in this thesis, therefore Cameras will be the main source of information in addition to joint position sensors.
5. Primary testing will be with different levels simulations.
6. Optionally, perform real life testing with robots if and when simulation proves feasibility and there is enough time.

Chapter 2

Milestones

This chapter describes the goals in details in terms of milestones and the proposed workflow.

2.1 Initial Preparations

Tasks:

1. Determine which type of errors affect the accuracy of robot's perception.

   Note: Joint backlash might be observable from joint angle sensors.

   Note: Joint offsets cannot be directly observed from on-board sensors.

2. Determine which poses the robot frequently use and how the errors affect.

3. **If** backlash is not directly observed, a model is needed.

   Else; It can be assumed only joint offsets affect the kinematic chain.

4. Prepare and set-up necessary software frameworks.

2.2 Generating Candidate Poses

Given the amount of joints and movement range, virtually infinite number of poses can be reached by the NAO. Considering the requirement of capturing in similar conditions to real game play, stable, standing poses are preferred. However, captures while moving will not be considered due to concerns of temporal synchronizing joint sensors with camera capture.

Tasks:

1. Identify criterion for a "valid" pose.
2. Implement necessary software for generating poses that follow the given criterion.
3. Define and implement necessary data storage formats as well.

2.3 Observation Model

In order to identify the ability of sensors under consideration and their observation capability of joint errors, developing models is proposed.

Tasks:

1. Define and observation model/ framework for joint and camera extrinsic error space.
2. Consider abstraction of observation models in order for transparent expansion with different sensors in the future.

2.4 Extracting Optimal Poses

Given the large amount of possible candidate poses, an optimal subset of these poses must be derived in order to get the best possible results while minimizing amount of different poses in order to save time when calibrating.

Previous work has demonstrated that selecting a set of optimal calibration poses is more beneficial than randomly selected set of poses [5–8].

Tasks:

1. Apply the observation model to each possible pose and obtain information such as magnitude and direction vector of observation space.
2. Poses that do not exhibit satisfactory amount of observation capabilities should be removed.
3. Define a suitable cost function that use information from previous step to sort poses in order of minimal cost.
4. Derive a set of Optimal poses based on previously derived list. This step will consider interaction of different poses with the calibration space with similar goal as [9] which performs a thorough investigation about approaches and solutions.
5. Identify potential placement of calibration patterns based on this optimal set of poses.

2.5 Calibration Process

The following tasks aim to derive a calibration process that will use poses obtained by the method proposed in the previous section.

Tasks:

1. Prepare necessary software for performing simulated calibration, including making the robot reach each pose, capture sensor values. (Cameras, joint angles, etc).
2. Iteratively solve to determine joint errors and camera extrinsic values.
3. Obtain statistics with a suitable sample size.

2.6 Testing and Evaluation of Calibration Poses and Process

Evaluating the outcome of this research is done as follows.

1. Simulated joint calibration testing with a dense set of captured data (3D-2D correspondences in case of camera) with a suitable population size.

   The goal is to verify calibration is possible to a high e of confidence.
2. Simulated testing with relatively sparse set of captured data. This would demonstrate the ability to use a calibration pattern (a relatively sparse feature with limited amount of points) at a given place.
3. Identify the effect of noisy camera captures and noisy joint readings.
4. If time permits, real world testing with actual NAO robots will be performed in the same manner as the previous method. Obtaining reliable ground truth might be a challenge.

Chapter 3

Literature Review

3.1 NAO Robot

The Nao robot is a 58cm tall Humanoid robot introduced by Aldebaran Robotics (now part of Softbank Robotics). It is used in humanoid robotics research as a ready-to-use platform. Since 2008, this robot became the platform in RoboCup Soccer Standard Platform league. The following subsections cover information related to this robot and this thesis.

3.1.1 Joints & Coordinate Frames

NAO V5 and V6 have almost identical link lengths but the masses may be different. The joint actuators are explained in their documentation, but they don't mention typical backlash values. However, [10] has analysed and presented Nao's capabilities and general information, it provides more in depth information than some sources in considerations such as joint backlash. According to it, maximum backlash tolerated is $\pm 3^{\circ}$ for stable walking and it also mentions more about its gear train and actuators.

Nao v5 joints can be commanded at 100Hz rate, which is possible via NAOqi - a software component supplied by Softbank Robotics to actuate the joints and to read back sensor values.

3.1.2 Sensors

3.1.2.1 Camera

NAO V5 has been the primary version used in the past few years which features two identical video cameras with positioning shown as fig. 3.2.

NAO V6 will be used officially in the upcoming RoboCup, thus it is also a priority to allow the calibration framework to be used with this platform. The cameras are

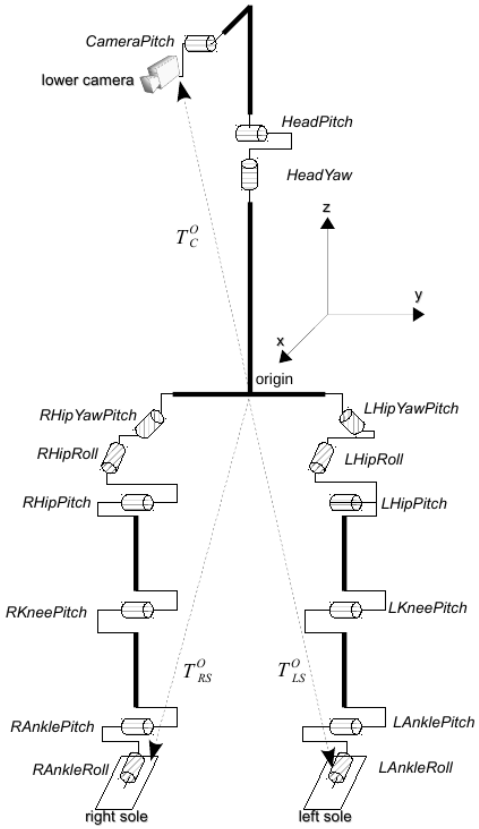


Figure 3.1: *Kinematic tree of the Nao. It depicts the joints considered for this thesis. Note that the pitch angles of legs are parallel. The CameraPitch is a fixed angle, based on values shown in documentation (fig. 3.2) of Softbank Robotics.*

upgraded to higher pixel density, and they also feature auto focus which will be disabled due to being impossible to have a static intrinsic matrix for a given camera.

3.1.2.2 Joint Encoders

They feature magnetic incremental encoders with 0.1° precision. As mentioned previously, these sensors give feedback to closed loop controllers of the joints as well as let the software framework to calculate kinematic matrices [12].

Based on experimental measurements in section 4.1.1, it could be verified that these sensors are directly connected to output of joint actuators and their resolution is 12 bits (4096 values per 360° giving 0.0878° increments which is rounded off to 0.1° in documentation. This measurement also allowed to have an idea of possible noise of these sensors.

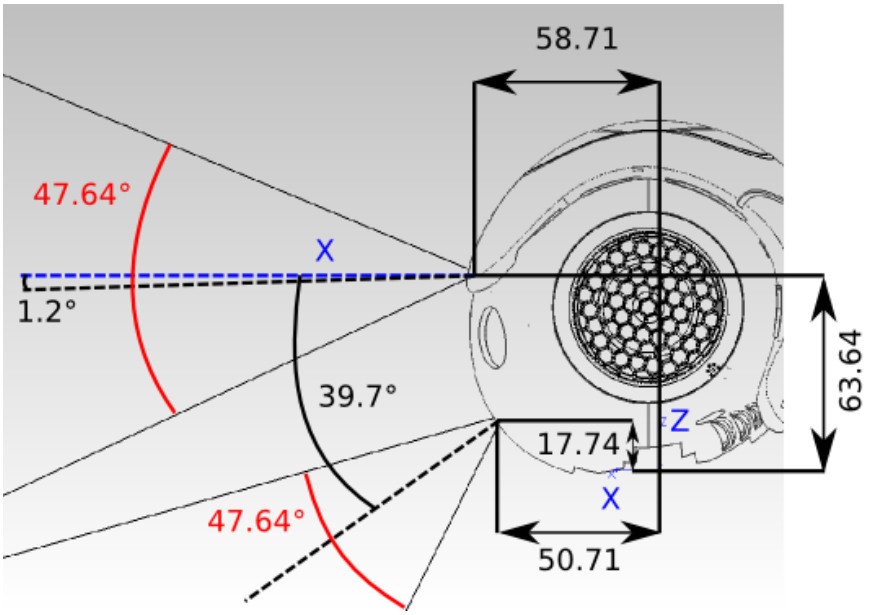


Figure 3.2: *This figure shows camera positioning of Nao v5. Source: [11]*

3.1.2.3 Other Sensors

The NAO also house other sensors such as IMU (Inertial Measurement Unit), microphones, sonar modules, FSR (Force Sensing Resistors). For this thesis, only the IMU is used in addition to Cameras and joint sensors.

3.1.3 HULKs NAO Framework

This is the software framework developed by HULKs for NAO to play soccer in RoboCup SPL. It access robot's hardware through several software components provided by SoftBank Robotics. However, it accesses cameras via a Linux kernel driver (Video For Linux - V4L2). Due to this difference and their cycle rates, two threads are employed for capturing sensor data; Vision (+ "brain" $\rightarrow$ perception & behaviour) and Motion. They cycle at 60Hz (30FPS per camera $x2$) and 100Hz (rate set by NAOqi - the interface to access hardware). Synchronization is done with a set of timestamped buffers and nearest (in temporal space) set of kinematic chain is selected for calibration purposes, therefore static poses are desirable for capturing images and joint values to get reliable readings due to limitations of this synchronizing method. The framework also supports debugging and configuration via a debug tool "MATE" - also developed by HULKs.

In addition, this framework includes a module to compensate joint offset errors. It offsets joints movement commands by the given value and it also compensates this

value from sensor output, making the robot's perception unaware of induced offsets while correcting real-world shifts. Furthermore, camera calibration is possible with a module implemented for MATE.

Further information about HULKs NAO framework and tools is present in [1].

3.2 Camera Calibration (Extrinsic)

Camera calibration is a well researched topic as it is a primary dependency for obtaining good results with camera systems. In this thesis, these standard methods & tools for intrinsic calibration is employed, while extrinsic calibration is considered as well.

However, given that head yaw and pitch joints are close to camera origins, and they are somewhat parallel to camera extrinsic parameters yaw and pitch respectively, they might cause a linear dependency or difficulty to observe together. Therefore, it was decided not to calibrate camera extrinsic parameters at the same time as joints, but do separately with multiple rounds. This approach seem to be used by B-Human in their calibration procedure [4] and appear to be converging well.

3.2.1 Method of HULKs

The calibration process is based on the debug tool used by HULKs (MATE) as well as the HULKs NAO framework. In summary, the calibration is achieved by capturing images and kinematic chain with time synchronization to solve the following system of equations with a nonlinear solver (Levenberg–Marquardt [13]). The method of operation and usage is explained in [1].

3.2.2 Other RoboCup SPL Teams

The team B-Human employs a method combined with joint calibration [4]. Nao-Team HTWK use centre circle of a SPL soccer field as the calibration feature [14], their robots gather enough feature points in several poses to solve for calibration parameters. Berlin United also employs a similar approach, but they do this with the robots in a sit down (un-stiffened) pose, and they can use all line-features in a SPL field [15].

3.2.3 Calibration Features

For typical intrinsic calibration, a chessboard or similar pattern is used. For extrinsic calibration, the same can be used. In methods such as photogrammetry, a dense feature set is used to identify the relative position of the camera.

Figure 3.3: *A NAO robot on calibration stand developed by HULKs for camera intrinsic and extrinsic calibration. It can also be used for joint calibration if necessary, but the proximity of calibration pattern seems to amplify any small amount of positioning errors of the robot.*

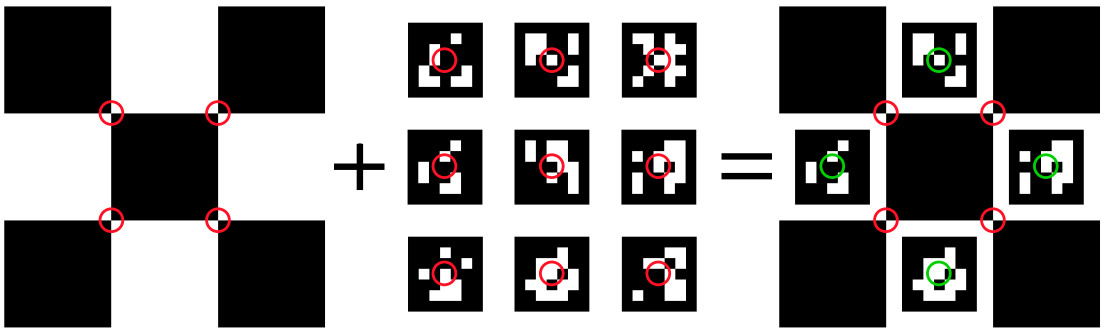
However, one disadvantage with the classic chessboard pattern is that identification of the corners is a time-consuming process and the existing methods relies on full visibility of the pattern in the image. As a solution for both limitations, ChAruco [16], [17] - a hybridization of the classic chessboard and AR markers to achieve fast detection and obtaining 3D points for partially visible patterns is employed via the implementation available in OpenCV library [18].

3.3 Joint Calibration

This section discusses the fundamentals of joint errors, reasons for calibration and possible approaches.

3.3.1 Joint Error Types

Two major causes of joint positioning errors that affect overall end effector location are backlash and shifts of sensor or actuator causing joints to position with an offset. These causes joint actuators to point at physically incorrect locations although their



(i) (ii) (iii)

Figure 3.4: *(i) Chessboard, (ii) Aruco and (iii) ChAruco patterns with calibration features circled in red. AR (Augmented Reality) markers are used to quickly locate grid positioning and corner numbering and based on it approximate location (shown with green circles) of chessboard corners are found. Then it use sub-pixel optimization if specified to further optimize exact position of each chessboard corner. This approach is much faster than scanning whole image for a chessboard pattern according to [17].*

positioning sensors precepts otherwise). Factors such as joint elasticity, looseness of mounting, worn bearings are not considered in the scope of this thesis.

3.3.1.1 Backlash

Backlash occurs due to necessary gap between gears (to avoid jamming) and wear occurred due to usage. Since output doesn't change when input is within this gap in a no load condition, it is also called "deadband" [19]. Even though the Nao appears to use gears aimed for low friction and low backlash [10], there is some amount of unavoidable backlash (Observed with Nao V5 refurbished, Nao V6 brand new) robots can be more than 1° and can be upto $\pm 5^\circ$ according to [3] (which in return cite [10] but it only mentions a maximum of 3°, yet personal observations confirm the former when Naos exhibits significant wear). Due to this unavoidable phenomena, a robot may not be in the position its sensors indicate and commanded to reach.

Figure 3.5 illustrate the effect of backlash and eq. (3.1) models it. It should be noted that this model assumes initial velocity of output to be zero whereas a complete model has to account forces/ torques applied upon the given joint's motion axis as that truly determines engagement direction or disengagement of input vs output. In addition, this model only cover simple case of backlash where gap is uniform throughout whole range of motion, no elastic deformations.

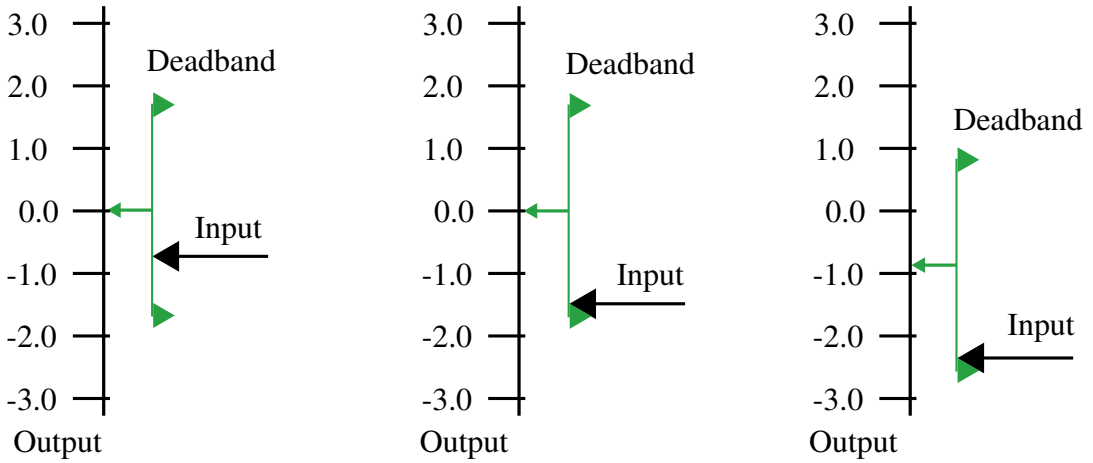


Figure 3.5: *Input vs output interaction with backlash. Deadband is the range of backlash or play. First case is disengaged, input is within the deadband resulting no output. Input is engaging the output when it reaches the end of deadband. Once the input engages with output, it changes output by same amount as long as input is moving in the same direction. This interaction is modelled in eq.* (3.1).

$$outputWithBacklash(i) = \begin{cases} o_i, & (o_i - \frac{D}{2}) < i < (o_i + \frac{D}{2}) \\ i + \frac{D}{2}, & i < (o_i - \frac{D}{2}) \\ i - \frac{D}{2}, & i > (o_i + \frac{D}{2}) \end{cases} \tag{3.1}$$

where:

D : is deadband;

i : is the input;

o_i : is the initial output.

Based on this information, it can be concluded that effects of backlash is a contributor for joint positioning errors.

3.3.1.2 Joint Offsets Due to Sensor or Actuator Mounting Errors

The secondary source of error occurs due to assembly tolerances of a robot's joints or faults of sensors or shifting of actuator; in these cases an encoder or servo motor belonging to a given joint might not align its index or "zero" position with joint's intended zero position when commanded to reach it with this type of error. Thus, any motion command given to this joint will exhibit a fixed amount of positioning error equating to

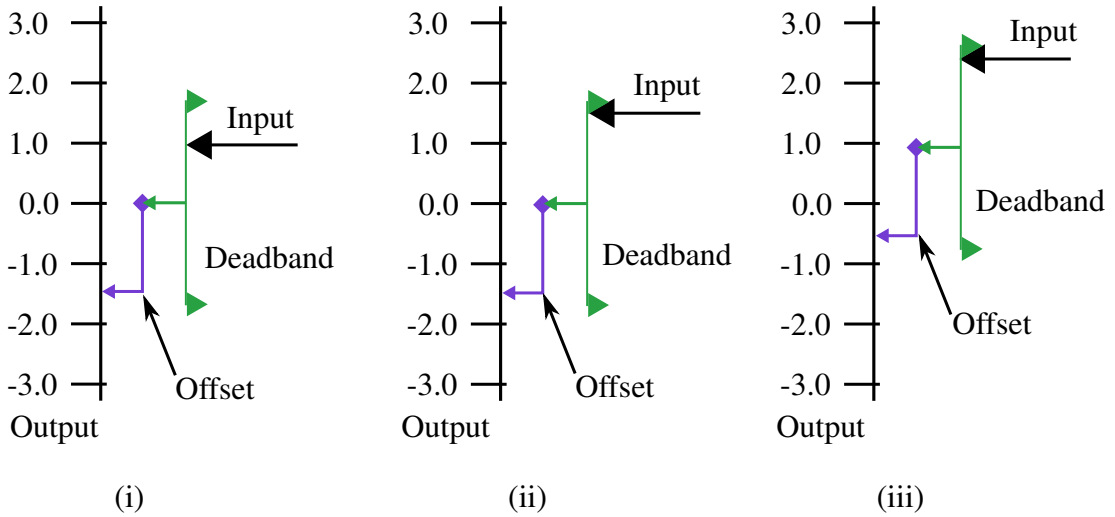


Figure 3.6: *Input vs output interaction with backlash and joint offset. Interactions are similar to case shown in 3.5, but this also shows effect of a fixed offset caused by physical movement of sensor or joint origin in addition to backlash.*

the difference of sensor zero position vs designed zero position of a joint, even if other factors are ideal such as zero backlash.

This type of shifting errors were observed with the Nao Robots during RoboCup 2018; especially with their head yaw joints, sometimes the error being as much as 20°which usually occurred after a fall in a game.

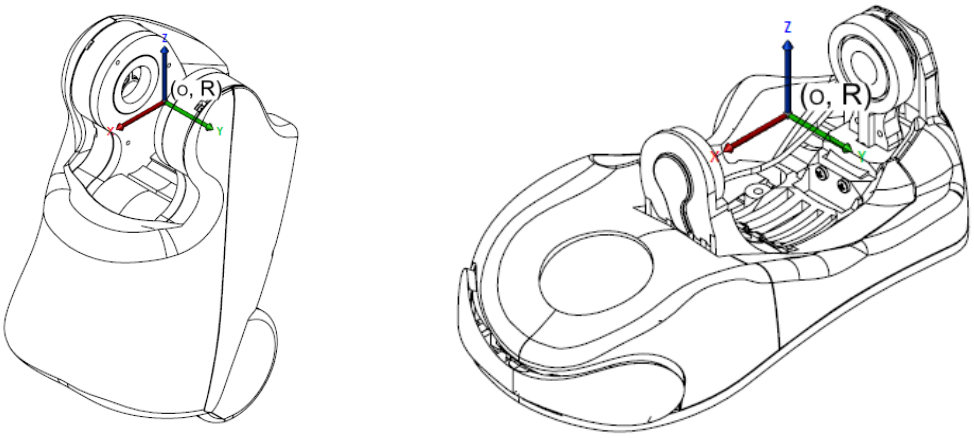
Figure 3.6 illustrates the effect of backlash with a fixed offset while eq. (3.2) models it.

$$\begin{gathered} outputWithOffsetBacklash(i) = E_o + outputWithBacklash(i) \\ \text{where:} \\ E_o : \text{is offset error;} \\ i : \text{is input;} \\ outputWithBacklash : \text{is from eq. (3.1);} \end{gathered} \tag{3.2}$$

This joint error model can be used to compensate both types of errors and also possibly applied only for a subset of all possible poses (ex: standing). This is further explored in section 4.1.1.

3.3.2 Direct Measurement

This approach is commonly used for lathes, milling machines and similar equipment with the use of dial indicators [20]. While this is possible for the NAO as well, the



(a) *Left thigh link with hip pitch joint origin.*

(b) *Left foot ankle joint origin.*

Figure 3.7: *This diagram shows the origins of left thigh and left foot which indicates the locations of the origins of left hip pitch joint and left ankle joints. As seen in the diagrams, it is difficult to locate a point of reference from outside in order to directly measure angles. This issue is further worsened due to the shapes of the surfaces of this robot. Image source: [12]*

"curvy" nature of the limbs (with almost no easily identifiable reference points, see fig. 3.7) presents a massive challenge in employing dial indicator based or similar approaches in a time efficient manner.

Another option is using an angle measuring tool or to 3D scan the joint at two extremes of backlash at a given position and obtain the angle. But this approach require more specialized equipment, processing power and possibly less portability. Yet this is a suitable method to obtain ground truth for the purpose of evaluating a particular calibration method.

3.3.3 Indirect Measurement

Indirect measurement based methods are possibly more practical in the context of NAO robot, as they involve in less specialized sensors or eliminate need to measure each sensor individually.

3.3.4 Previous Work

There are several experiments done in context of the NAO Robot for indirect joint calibration. Some involves fixing a calibration pattern in the form of a sticker [21]. One of the experiments by RoboCup SPL team B-Human involves "sandals" worn by the

NAO [3], but ultimately they concluded the results being less than satisfactory. Their experiment didn't measure nor compensate backlash, but assumed the error is a fixed offset only. In addition, the robot took measurements while lying on its back (based on images in the publication) which is not a typical pose for the use case, thus backlash affected the joints (differently) than what would be observed had the robot being standing. In addition, the authors also mention possible effect from flexing of the limbs and torque modelling would have improved their experiment based on the results by Nao Devils of TU Dortmund in modelling flexing and joint torque contributions.

Later, B-Human have been using a semi-automatic method to calibrate both joints and camera extrinsic parameters as mentioned in section 3.2.2. First, the robot is put to a specific pose (ie: standing), lifted to observe foot offsets and adjustments are done so that both feet are at same level and orientation. Next, the robot is placed on the ground and the distance to hip or another known joint origin is measured. This allows to measure the length of the kinematic chain of each leg. Any deviations are compensated by means of inverse kinematic values and finally the robot's leaning towards ground is also adjusted. While this appears to be a practical method, the disadvantage of manual measurements is time-consuming and can be erroneous.

Several publications have covered the topic of choosing optimal poses. The approach was to obtain end effector positions for each pose and stack them as columns to build a matrix. Then the condition number of its Jacobian is taken. The optimizer tries various poses and attempts to obtain the matrix with best conditioning which appears to provide the optimal set of calibration poses. Their results prove that measurements from a small amount of optimal calibration poses yields better or equal results compared to using large amount of randomly chosen measurement poses. Based on these facts, it is sensible to focus on finding for a set of optimal poses for the Nao. However, it should be noted that these publications were applied to industrial robots where the measurements were taken by accurate 3D measurement tools, whereas the preferred constraints for this thesis's use case provides further complications as the 2D cameras can only estimate the pose of a calibration target, not get a precise position. Therefore, the author's approach in this thesis has some divergence on searching the set of optimal poses [5–8].

3.3.5 Nao Kinematic and Camera Projection Model

The following equations are based on above mentioned literature for Joint and Camera extrinsic calibration [1, 22]. Since any calibration feature can be positioned relative to robot's position on ground, it is possible to express these features in camera coordinate system through robot's kinematic chain as depicted in eq. (3.3) and **??**. After using the transformations defined in that equation, it is possible to project these 3D points into camera plane using the standard pinhole projection model [18, 22].

$$\text{camera2Head}(\alpha,\beta,\gamma) = \text{camera2HeadUncalib} \times \text{R}_{\text{ext}}(\alpha,\beta,\gamma) \tag{3.3}$$

$$\begin{aligned}
\text{camera2Ground}(\mathbf{j},\alpha,\beta,\gamma) &= \text{torso2Ground}(\mathbf{j}) \\
&\quad \times \text{head2Torso}(\mathbf{j}) \\
&\quad \times \text{camera2Head}(\alpha,\beta,\gamma) \\
\text{ground2Camera}(\mathbf{j},\alpha,\beta,\gamma) &= \text{camera2Ground}(\mathbf{j},\alpha,\beta,\gamma)^{-1} \\
&\text{where:} \\
\alpha,\beta,\gamma &: \text{are extrinsic calibration;} \\
\mathbf{j} &: \text{is the list of joint angles} \\
\text{torso2Ground}(\mathbf{j}) &: \text{is from forward kinematics;} \\
\text{head2Torso}(\mathbf{j}) &: \text{is from forward kinematics;}
\end{aligned} \tag{3.4}$$

3.4 Observation Models and Ambiguities of Observations

Given that a camera can observe a given scene, it is in the interest of this thesis to determine poses of the robot where the camera can observe the difference caused by a joint error. Furthermore, as there are 14 Joints in interest (excluding arms), but only 3 or 6 dimensions to observe in a standard calibration pattern (position and orientation), for a camera based joint error observation model, there is a high possibility that the observation *strength* of multiple joints at a given pose have similar direction vector. This effect will be referred as "ambiguity" for the rest of this thesis. This issue can lead to multiple solutions or local minima situations when using a solver as these ambiguities make it difficult or impossible to clearly identify which joint caused this. Multiple research has explored this concern and methods to devise observability models for this purpose [7, 23].

This is a crucial element that can weigh into the success or failure of this experiment. Therefore, it is important to pre-determine the possible ambiguities of the observations.

It is already possible to identify problem cases by knowledge in inverse kinematics and intuition. For an example, Knee, hip and ankle pitch together may end up causing ambiguities, the reason is chance of attaining a given position with more than one possible joint configuration. This is a common case in robotics when there are more joints (that can cover one or more DOF) than degrees of freedom. In addition, the observation inaccuracies can further cloud the calibration solving [24].

The evaluation of similarity between two observation directions can be done in multiple ways, one option is cosine similarity. Another is to get difference and norm of it. Also, it is possible to simply see if each dimension is close by some amount, etc. Reduc-

ing the similarity measure into one dimension has its appeal as it is easier to understand and write programs to handle this concept.

3.5 Computing a Pose for a Robot

This section discusses approaches in deriving "poses" for the robot. In this context, generating a pose means obtaining appropriate joint angles or expressing a pose by means of feet positions relative to torso and head joint angles. (Arms are not moved, will be in 0° position.)

3.5.1 Forward Kinematics

In this approach, the pose is directly defined in terms of joint angles. The kinematic chain is generated by means of forward kinematic equations.

The drawback is that when generating in a brute-force method, it is not possible to directly identify if this pose is safe, stable, etc (these calculations is better defined via kinematic matrices). Due to this nature it is also not possible to directly define a work envelope (the region which a robot's end effector can access).

Finally, such brute force approaches can easily end up with very large number of possible iterations to exhaust entire joint angle space even with 1° step size. For example, with 16 joints, 90° range for each joint with 1° increment would result in $n_c = 1.8530202e+31 = (\frac{r}{i})^{n_j}, r = 90^\circ, i = 1^\circ, n_j = 16$. Assuming 1μs per each configuration's generation it'll take more than 5^{24} hours in total for computation, therefore this approach should be considered with care.

3.5.2 Inverse Kinematics

Inverse kinematic approach take the target pose in terms of feet and torso position relative to each other and head joint angles. Then inverse kinematics is applied for the two legs to obtain the corresponding joint angles. The arm angles are obtained by one of the predefined poses within HULKs framework (STAND pose [1]).

In conclusion, direct joint values cannot give an absolute understanding of the robot's posture to a user while the dual is that the robot cannot directly understand it's supposed to be joint positions by means of kinematic matrices of feet relative to pose.

Since both formats are required for generation, storage and interfacing, appropriate C++ classes and serialization methods are devised.

3.6 Optimizers

This section discusses the considered approaches to determine optimal (or set) of poses for a given set of conditions.

Considering the above description of Observation models, in the case of a model that can output the observability of joint errors, it is desirable to obtain the pose(s) that can give the highest observability. This maybe further constrained by applying weights such as which joint get priority, etc.

In order to solve a cost function or identification of parameters of a system, one possible way is to try all valid values for the parameters - applying the entire configuration space. This approach is a brute force method, as the name suggests it isn't "smart" or efficient. In order to cut back optimization time, various solvers exist to speed up the process. They usually employ gradient of error, sum of error squared and combinations.

Out of these optimization options, one is global optimization - they will attempt to find the best solution within the entire configuration space. For example, brute force method can do this as it explore the entire space. While the prospect of getting the best value is tempting, global solvers are computationally expensive, thus local methods are favoured when there is a good initial guess. There are also various approaches to speed up solving of global optimization problems including multi-agent, flood fill methods [25, 26].

Local optimization methods will find the closest minima (or maxima, depends on the approach) to the starting point. There is no guarantee it is the best solution. Thus the initial guess plays a major role. Levenberg Marquardt [13] is one such popular solver/ optimizer algorithm.

3.7 Conclusion

Given that previous work with regard to Nao or similar humanoid robots didn't cover aspects such as using observability indexes or models to find for optimal calibration poses compared with research concerning industrial robots and the fact that there isn't a significant amount of previous research regarding the observability of joint errors with on board sensors (esp. 2D cameras) alone, one of the key components of this thesis would be devising an observation model for joint errors and deriving most suitable poses. This should assist to explain the quality of calibration at each joint.

Chapter 4

Methodology

This chapter describes the methodology followed in this thesis. It is based on the workflow described in chapter chapter 2

4.1 Initial Preparations

4.1.1 Joint Error Model

In order to identify potential sources of joint errors, a few tests were done with a Nao robot and following observations were made.

Nao joint encoders can sense backlash, this was tested by setting the robot to a pose with high stiffness of joints and then each joint was manually moved with light force to "feel" backlash while monitoring sensor values from the debug tool (MATE). It was confirmed that the sensors respond when the joint is moved within the range of backlash, thus it is possible in theory to measure this range directly; By measuring the angular difference sensed when the above test is performed. This is the usual technique used to measure backlash of lead screws and gear trains [19, 20].

However, it is a challenge to compensate backlash of a complex kinematic chain such as the Nao Robot due to the number of joints and difficulty in identifying the direction of force acted upon a joint (as this is a real scenario including inertia and masses as opposed to simple demonstration model described in section 3.3.1.1).

Another source of error is noise found within joint encoder readings. An experiment was performed to record joint angle values over a period of 50 seconds approximately (5000 samples @ 100Hz). A Nao was on a sitting pose without any joint stiffness and joint positions weren't at zero in general. Therefore, these recordings were shifted to have zero median for better visualization and can also serve as a noise distribution for the joint sensor. Figure 4.1 depicts these recordings. It can be concluded that these values move at most by increment of sensor resolution mentioned in section 3.1.2.2

(0.087°). Based on this, a simple zero mean Gaussian distribution based noise model can be defined as depicted in eq. (4.1).

$$\alpha_{noise} = \frac{2 \cdot \pi}{s} \cdot \left\lfloor \frac{s}{2 \cdot \pi} \cdot (\alpha + \mathcal{N}(\mu = 0, \sigma^2)) \right\rceil, s = 4096, \mu = 0, \sigma \approx 0.01 \tag{4.1}$$

where:

α_{noise} = Joint angle sensor reading with noise
s = steps per revolution = 4096 according to documentation
α = input angle
μ = Gaussian distribution mean
σ = Gaussian distribution's standard deviation

Based on above facts and observations, following simplified joint error model is applied and assumed, with constraint of accuracy of this only valid for a subset of poses.

4.1.1.1 Simplified Joint Error Model

If a given joint experience force/ torque always in same direction (or same input direction for 'theoretical' massless, unforced joints), then effect of backlash can also be considered as a fixed offset as it'll not satisfy "$(o_i - \frac{D}{2}) < i < (o_i + \frac{D}{2})$" in eq. (3.1). With this assumption applied to eq. (3.2), total error can be modelled as a fixed offset. Then joint positioning error can be simplified as a sum of two fixed offsets as

$$outputSimplified(i) = E_o + i \pm \frac{D}{2}$$

, sign of $\frac{D}{2}$ depends on direction of input. According to (iii) in fig. 3.6, the scenario gives $output = E_o + i - \frac{D}{2}$; thus the sign of $\frac{D}{2}$ is the opposite direction of input (i), so this equation could be expressed as eq. (4.2). It should be stressed that this is valid if and only if input "engages" output by reaching either extreme of deadband, which means the joint must be under load, in case of transitions of load direction, this equation is invalid and eq. (3.2) should be used instead as it covers input within deadband.

$$outputSimplified(i) = E_o + i - \text{sgn}(i) \cdot \frac{D}{2} \tag{4.2}$$

By assuming the above scenario for each joint and all poses are captured attempting to enforce this, calibration process becomes significantly simplified as joint errors become fixed offsets by fixing input direction in eq. (4.2) instead of accounting for backlash directions, determining backlash values.

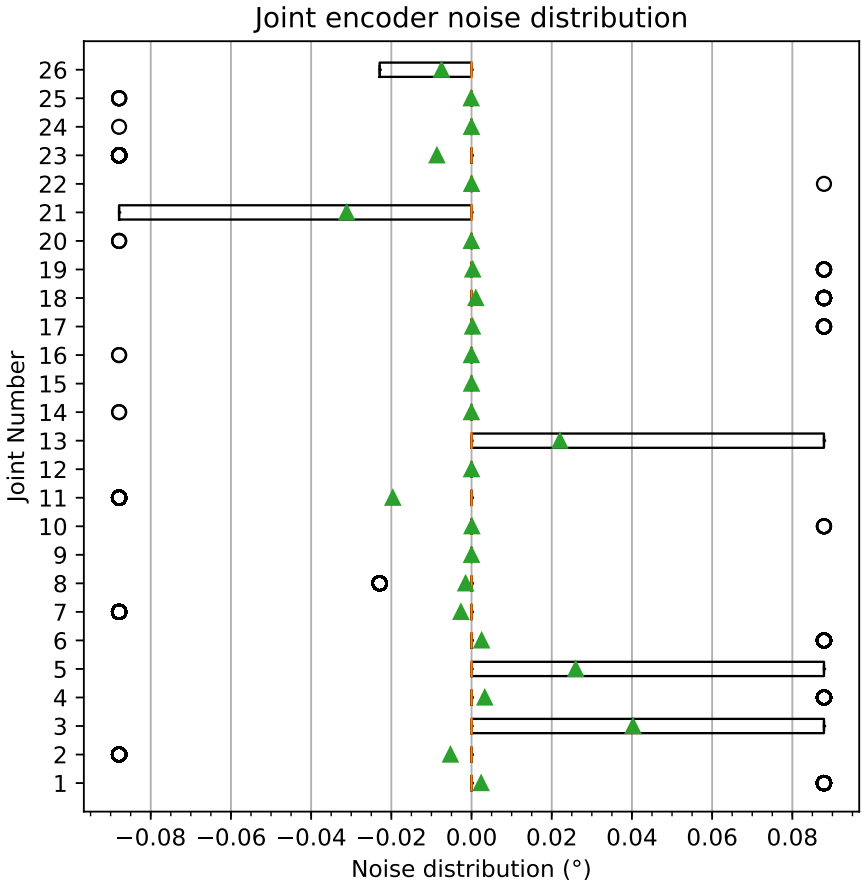


Figure 4.1: *Joint sensor value distribution when at rest. The values were shifted to zero using median for better visualization, and this is also equivalent to joint sensor noise distribution.*

4.1.2 Deciding Type of Poses for Calibration

In section 4.1.1, it was determined that a limited class of poses are to be used for calibration in order to ensure backlash is generally acting in the same direction.

In context of Robocup SPL, Nao is supposed to walk, stand up, etc similar to a human while locomotion modes such as crawling is strictly forbidden [27]. The typical cases which a robot isn't on its feet is when it has fallen down, during standing up motions or when the keeper jumps to catch a ball.

Therefore, it is sensible to restrict all calibration poses be standing, with one leg or both supporting. Restricting poses to single leg/ foot support will make the kinematic

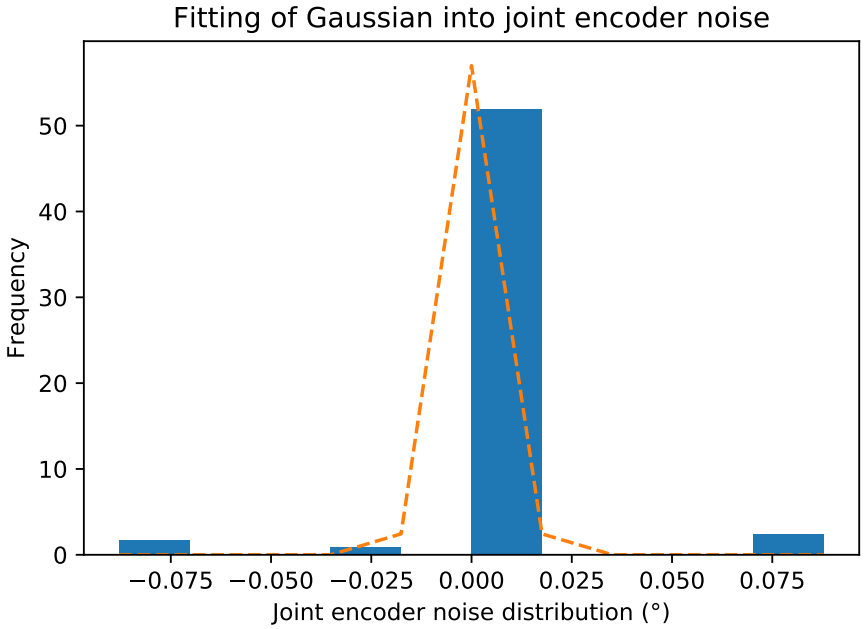


Figure 4.2: *Approximated fitting of a Gaussian distribution to joint position sensor noise (based on fig. 4.1. Standard deviation is 0.007 with zero mean.)*

chain only depend on that leg, thus able to separate joint errors of each leg.

In order to avoid time synchronization issues (as mentioned in section "Camera Calibration" of [1]), capturing is done when the robot is not moving. This also automatically implies the standing pose must be statically stable.

Based on above information, a summarized set of requirements each pose must comply is given below:

1. Be a pose that is reachable by joints and obey joint constraints.
2. Must not cause collisions between own limbs
3. Be standing with one or both feet.
4. Be statically stable.

Section 4.2 covers generation of poses based on these requirements.

4.1.3 Software Frameworks and Tools

Several sets of software were implemented for this research. One portion was within the NAO to obtain images and to get joint angles after a being stable. The next is

user interface (within debug tool) to command robots to reach a pose and the necessary calculation stage within the NAO.

The crucial set of software was in the pose generation and optimal pose finding stage. This was a set of C++ programs that depends on the HULKs NAO framework (TUHHSDK segments to be precise) to access existing kinematic, center of mass calculations as well as camera matrix and related calculations. In addition, a multitude of classes were implemented to store data in text files with space separated fields, with full serialization and de serialization capability to simplify IO operations. The following classes could be streamed into files and thus be persistently stored.

4.2 Generation of Calibration Poses

This phase became a major segment of this project due to the importance of finding out poses that could get best results as calibration data captures. About 80% of the implementation was dedicated for this matter.

4.2.1 Pose Generation and Initial Filtering

Initially, the forward kinematics approach mentioned in section 3.5.1 was employed due to near infinite number of possible joint angle configurations and the lack of "direct understanding" about a given pose (poses of feet, head and torso), this approach was not practical to be used. Pose of torso and "other foot" are defined with respect to support foot (the foot that is supporting a standing pose). This enabled to understand and visualize pose of torso and other foot relative to ground.

Inverse kinematics based approach (section 3.5.2) accepts the following parameters:

- Support foot - left, right or double foot. ([28, 29]).
- Torso pose relative to support foot (${}^{Torso}_{SF}T$ contains both position and rotations)
- Pose of the other foot relative to support foot (${}^{OF}_{SF}T$)
- Head yaw and pitch angles

This approach is somewhat complicated by the need to use inverse kinematics and sometimes it was observed that a given pose is not achieved by inverse kinematics (as opposed to generating a pose directly from joint angles). Thus further checks were needed to verify the generated pose is actually similar to the desired pose. While head angles are directly used and angles for arms are taken from a default pose, eq. (4.3) has to be used to derive leg angles.

$$
\begin{aligned}
{}^{SF}_{Torso}T &= {}^{Torso}_{SF}T^{-1} \\
{}^{OF}_{Torso}T &= {}^{Torso}_{OF}T^{-1} \\
[\alpha_{hipYawPitch}, ..., \alpha_{anklePitch}] &= legAngles_{SF} = InverseKinematics({}^{SF}_{Torso}T) \\
[\beta_{hipYawPitch}, ..., \beta_{anklePitch}] &= legAngles_{OF} = InverseKinematics({}^{OF}_{Torso}T)
\end{aligned}
$$

where: (4.3)

$$
\begin{aligned}
SF &= \text{Support foot} \\
OF &= \text{Other foot} \\
{}^{Torso}_{SF}T &= \text{Torso pose with respect to support foot} \\
{}^{Torso}_{OF}T &= \text{Other foot with respect to support foot} \\
[\alpha_n, .., \alpha_m] &= \text{Array of leg angles in support foot's side} \\
[\beta_n, .., \beta_m] &= \text{Array of leg angles in other foot's side}
\end{aligned}
$$

4.2.1.1 Pose Validation

Once angles for a pose is obtained, these angles are then subjected to multiple checks to ensure conformity of this pose to the constraints in section 4.1.2.

First, forward kinematics calculation was applied to verify if inverse kinematic calculation was performed correctly. This also enabled to verify joint constraints as forward kinematic calculations check for them, thus verifying first constraint.

To comply with second criteria of pose constraints, collision detection was introduced based on a modelling technique published by Softbank Robotics for the NAO [30].

Third constraint is checked by computing static stability of the robot while assuming a standing pose. This is done by checking if robot's COM (Centre of Mass) projection to ground plane is within the support polygon. In case of single foot (support foot on the ground, and other foot raised), the support polygon comprises portion of NAO's foot touching the ground. In case of double foot, it's the polygon containing both feet and the region between them [28, 29].

The fourth and final pose constraint is implicitly checked by the previous test by assuming the robot is standing on its own feet (or one foot if other is lifted), thus passing static stability test automatically qualify this constraint.

4.2.2 Observation Model

Considering revelations from previous research mentioned in section 3.3.4, observations and feedback gathered from discussions with other team members and teams, deriving

an observation model of joint errors with respect to various available sensors (or fused outputs) in order to find a set of optimal poses would be more effective than following a trial and error or brute force or random pose selection approach.

In context of this project, the most important criteria of selecting a pose is based on the observability of small joint movements by the available sensors. Each of them are modelled using as an observation model, thus it is possible to examine the strength and direction of observable dimensions of each sensor at a given pose by reaching that particular pose and inducing small joint movements (mimicking possible joint errors).

For the scope of this project, only the two cameras on-board the NAO robot were modelled. In addition, assumptions were made regarding the placement of calibration patterns.

4.2.2.1 Camera Observation Model

This model comprises following elements, its state is updated whenever a new pose is submitted.

1. Camera projection model object (defined in section 3.3.5).
2. Ground plane grid. This is a grid of points on the ground plane (see section 5.4).
3. Support foot, observable joints (for each camera and at each support foot).
4. The observation space of small movement by the camera's sensor (2D) is assumed to be X, Y translations and Z rotations.

It is preferred to use poses that only use one leg to stand as it constrain the chain to be using only that leg, thus any observations made will be more specific.

4.2.2.2 Extending Observation Model

Based on the technique employed for the camera model, it is possible to model other sensors as well. The inputs would be joint angles, support foot and information of observable joints by the given sensor. Output would be similar to camera observation output, in fact the same type of object can be returned albeit a possible different number of observed dimensions ($n \times m$ instead of $3 \times m$ as for return type of algorithm 1). Once the observations for a pose is returned for each sensor under consideration, they are written to a file with pose ID (generated in the previous step), observation values for each joint.

getObservability $(\mathbf{j}, sf, M, grid)$

inputs : $\mathbf{j}$: Joint angles of a given pose, sf: support foot of that pose, M: NAO kinematic and projection model (see eq. (3.4)), grid: set of points on ground as a grid.

output: A $3 \times m$ matrix containing t_x, t_y, r_z observations in each column.

$obsMat \leftarrow [0]_{3\times m}$;
M.update$(\mathbf{j}, sf)$;
`// Update projection.`
$projectedBase \leftarrow M$.project3DPoints$(grid)$;
`// Project the grid`

foreach *jointIndex in* j **do**

$\mathbf{j}_n \leftarrow j$;
$\mathbf{j}_n[jointIndex] += \delta\theta$;
M.update$(\mathbf{j}_n, sf)$;
`// Update projection.`
$projectedTemp \leftarrow M$.project3DPoints$(grid)$;
`// Project the grid`
$t_x, t_y, r_z \leftarrow getObservation(projectedBase, projectedTemp)$;
$obsMat$.append$([t_x, t_y, r_z]^T)$;

end

return $obsMat$;

Algorithm 1: Obtaining observability of a pose using the model containing elements in item 4.

4.2.3 Criteria for Evaluating Observability

When a series of measurements is organized in the form of a vector, it gives the possibility to understand if two sets of measurements are "nearby", or at same direction by using similarity metrics such as cosine similarity.

In this case, it is possible to take each column of observation matrix taken from algorithm 1 and compare with another column of it. Since each column refers to observation of a specific joint, if two (or more) observations are similar for the same original pose, there exist an ambiguity, as it isn't possible to separately identify which joint caused the particular observation. Thus avoiding poses with such similar observations for multiple joints is beneficial as the solving the equations by means of iterative solver needs as much as orthogonality between observation vectors as possible. An example structure of this vector is shown in eq. (4.4).

This will be mentioned as *joint-joint interactions* at later appearances in this thesis.

4.2.4 Extracting Optimal Poses

Considering the above mentioned need for dissimilar observation vectors for each joint observation for a given pose, and other factors, it was evident that filtering the poses with a cost function would be beneficial. The cost function had weights not only for the similarity of observation between two joints, but also possibility to bias towards a given camera and a given joint (if calibrating a certain joint without ambiguity is more important).

However, once the evaluation was performed (section 5.4), it was noticed that the rate of local minima convergence was high (70% approx). Therefore, an alternative or an improvement over the cost function method was needed to list a set of poses that complement the observation capabilities of each other. This approach is presented in section 4.2.4.1.

4.2.4.1 Pose-Pose Interaction Based Sorting

As explained in the section about ambiguities section 3.4, multiple error configurations can have the similar observation from the camera, thus there will be multiple error configurations which can lead to low ($\pm 2px$) average reprojection error, causing the solver to converge at a local minima. The aim of the above cost function was to choose poses that have minimum amount of joint observation ambiguities. However, one major problem of this approach was that it only considered the joint observation interactions within a pose. Since it did not consider if two poses have similar set of joint observation ambiguities, choosing a sample of "best" poses from this cost function did not yield good results as shown in section 5.3.3.

Therefore, a method to formulate the ability to the ambiguities of different poses was necessary. The following terminology is introduced and then the process is detailed. Complementing a pose by another pose in this context means reducing the overall *joint-joint interaction* cost.

1. *joint-joint interaction* vector: This vector contains the angle obtained from cosine similarity comparison mentioned in section 4.2.3 for each joint-joint comparison.

2. *pose-pose interaction* vector: This vector is of same size as joint-joint interaction vector. This contains values which indicate if the poses under consideration can complement joint-joint interaction. Ideally, this value will be 0, meaning the two poses don't have similar ambiguity issue for the same pair of joints.

3. Interaction count: this value indicate how many non-complementing cases are there for a pose-pose interaction or joint-joint interaction. Ideally this should be [0].

4. pose-pose interaction cost: Similar in concept to joint interaction cost, it is the sum of above mentioned pose-pose interaction vector.

The method of constructing the interaction cost or ability to reduce ambiguities between two poses was to element-wise multiply the *joint-joint interaction* vectors of these two poses. Next, In order to To connect the next pose, the previously calculated pose-pose interaction vector is similarly multiplied with the joint-joint interaction vector of the third pose. This process is shown in eq. (4.5). Based on that equation and fig. 4.3, it is evident the computation cost keep increasing and the numbers can overflow. Therefore, the joint-to-joint interaction vector (eq. (4.4)) was normalized prior to this. In addition, to obtain a mean cost or the equivalent of arithmetic mean for these, the geometric mean should be calculated. This will be considered as the pose to pose interaction cost.

$$\begin{aligned} \mathbf{j}_{P1} &= \begin{bmatrix} j_{1v2} & \cdots & j_{13v14} \end{bmatrix} \\ &\vdots \\ \mathbf{j}_{Pn} &= \begin{bmatrix} j_{1v2} & \cdots & j_{13v14} \end{bmatrix} \end{aligned} \tag{4.4}$$

$$\begin{aligned} \mathbf{k}_{P1vP2} &= \mathbf{j}_{P1} \odot \mathbf{j}_{P2}^{T} \\ \mathbf{k}_{P1vP2vP3} &= \mathbf{k}_{P1vP2} \odot \mathbf{j}_{P3}^{T} \\ &\vdots \\ \mathbf{k}_{P1vP2v\ldots vPn} &= \mathbf{k}_{P1vP2v\ldots vP(n-1)} \odot \mathbf{j}_{Pn}^{T} \end{aligned} \tag{4.5}$$

where:

$j_{P1} \ldots j_{Pn}$: are joint to joint interactions of pose 1 to n

$P1 \ldots Pn$: are poses

k_{P1vP2} :vis pose to pose interactions between P1 and P2

$k_{P1v\ldots vPn}$: is pose to pose interaction chain

Since there were over 1 million eligible poses to be tested, they were filtered out and were sorted according to the costs mentioned in section 4.2.3. Next, the best pose is selected and pose-to-pose cost is calculated for all other poses with the selection. The best out of it is selected to the next round. As it is visible, in fig. 4.3, this tree-like process with brute force testing is heavy in computation cost, therefore this was finally performed on a 24 core server with a limit of stopping when this pose-to-pose chain contains 10 poses at maximum or when the interaction cost does not improve

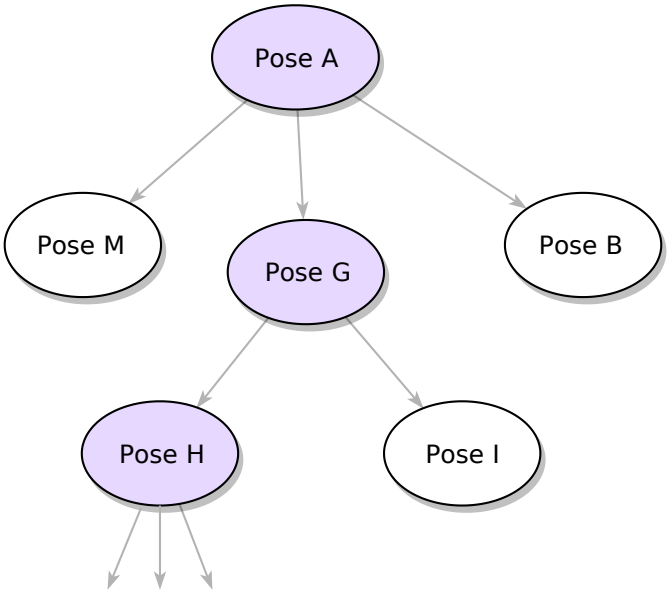
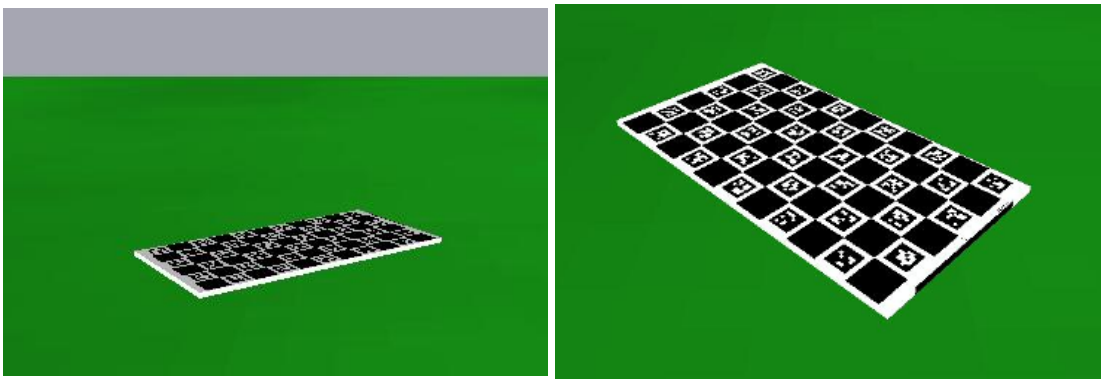


Figure 4.3: *Pose to Pose chaining graph. This shows the branching effect and the complexity. The nodes coloured as purple are the poses that were selected in this illustrated case.*

(which means there are no poses left that can improve existing joint error observation ambiguities).

4.2.4.2 Determining Calibration Pattern Position

The positioning of calibration patterns is dependent on the poses, and at the same time, these poses are constrained by the maximum distance the NAO can see the calibration features. Figure 4.4 shows an instance where this issue is noticeable. Therefore, the size of grid mentioned in section 4.2.2.1 was restricted to 4 meters assuming the NAO can see these patterns clear enough upto this distance.



(a) *Farther* **(b)** *Closer*

Figure 4.4: *View of calibration pattern at different distances (taken from SimRobot simulator). As it can be seen, the pattern is not clear at somewhat farther distances. This issue affects the poses that can be used for calibration while the final calibration pattern placement depend on these poses.*

Chapter 5

Testing and Evaluation

A sizeable amount of the work of this thesis was dedicated for testing and evaluation as this topic offered multiple avenues to obtain the calibration poses and the process itself. The next sections will introduce the method of testing each calibration attempt, the method of generating errors to calibrate against. Furthermore, in order to cover the requirements of having good accuracy with minimum amount of poses and to consider observations from previous work (chapter 3) different solving strategies as well as the number of poses, joint angle error range were also considered.

5.1 Calibration tests

Evaluation of calibration was done primarily with simulated data due to difficulty in verifying calibration done with real robots; Nao V5 robots are ageing and to be retired, not only their joints tend to be worn but their limbs exhibited looseness and elastic nature, this will adversely affect calibration and misrepresent any successes or failures. Nao V6 robots are in excellent condition due to being brand new, however certain details of hardware is not officially confirmed yet, that could also lead to false results.

Following algorithms depicts the process used to evaluate calibration of the robots. This process can be used for both real and simulation scenarios. algorithm 2 is a simulation specific algorithm which is used to generate simulated captures of a robot for a certain configuration of joint errors with a set of poses. For realistic scenario, the data structure $Fcap$ in that algorithm can be built by real captures of a robot, but in this case error configuration is induced to the robot by joint offsets (assuming robot is error free).

Once $Fcap$ is obtained, it can be given to algorithm 3 which is used to calibrate for each scenario and return an array containing solved calibration parameters, RMS error of pixels, calibration output and state of calibration attempt - success, local minima convergence, no convergence or numerical error.

getFrameCaptures $(\mathbf{e}_c, P, G, m)$

inputs : **e** : A joint error vector, containing an error value (an offset) for each joint, P: a set of poses to use in calibration, $grid$: a set of 3D points to use as calibration features, M NAO kinematic and camera projection model.

output: An array of structures $Fcap$ each containing: pose p, 3D-2D correspondence pairs observed for that pose c, camera used to observe cam and support foot sf

```
// Initialize set of 3D-2D correspondences
```

$Fcap \leftarrow \emptyset$;

M.setJointErrors($\mathbf{e}_c$);

foreach *Calibration pose p* $p \in P$ **do**

  M.setPose(p);

  $c \leftarrow M$.project3DPoints($grid$);

```
  // Returns 2D projections and corresponding 3D point
```

  $Fcap$.append(p, c, cam);

end

return $Fcap$;

Algorithm 2: Generating an array of structures each containing: pose p, 3D-2D correspondence pairs observed for that pose c, camera used to observe cam and support foot sf with projection model of NAO from eq. (3.4).

Finally, algorithm 4 is mainly used in simulated testing to run previously algorithms against numerous artificially generated error configurations by using uniform random distributions. After obtaining results from a significant amount of test runs, these results are further treated with a set of python scripts and resulting plots are displayed in the next sections along with conditions which the experiment was conducted as well.

EvaluateJointCalibration $(Fcap, M)$

inputs : Fcap: A set of captures, from algorithm 2 or from real robot captures and M: NAO projection model.

output: A set of statistics containing RMS error in pixels RMS_x, RMS_y, output from calibration Y, status of calibration $status$

```
// Intial value
```
$X \leftarrow [0] * 26;$

```
// Y is calibration offsets for joints, status_solver is
   returned by solver if it failed or not, res_px is
   residual in pixels.
```
$Y, status_{solver}, res_{px} \leftarrow calibrator(F_cap, M, X);$

```
// Calculate Root Mean Square errors of pixels in x and y
   directions
```
$RMS_x, RMS_y \leftarrow calculateRMS(res_{px});$

```
// Determine final status of calibration
```
if $solverStatus \neq SUCCESS$ **then**

```
    // Numerical error or other solver failure
```
    $status \leftarrow E_{num};$

else if $RMS_x > rmsErrorTolerance$ *or* $RMS_y > rmsErrorTolerance$ **then**

```
    // Convergence failure
```
    $status \leftarrow E_{conv};$

else if *any of* $calibratedJointOffsets > jointCalibQualityTol$ **then**

```
    // Converged to a local minima
```
    $status \leftarrow E_{local};$

else

```
    // No failures detected
```
    $status \leftarrow SUCCESS;$

return $(Y, RMS_x, RMS_y, status);$

Algorithm 3: Evaluating Joint calibration quality for a given set of error configurations, calibration poses, ground points and the NAO projection model (eq. (3.4)) to simulate kinematic chain and projections.

EvaluateCalibrationPoses (E_c, P, G, M)
inputs : A set of joint error configurations E_c; A set of poses to use in calibration P, a set of 3D points to use as calibration features G, robot's kinematic and camera projection model M
output: A list structures each containing statistics for each calibration run (output of algorithm 3)
$statsArr \leftarrow \emptyset$;
foreach *Error configuration* $e_c \in E_c$ **do**
$Fcap \leftarrow getFrameCaptures(e_c, P, G, M)$;
$\{Y, RMS_x, RMS_y, status\} \leftarrow EvaluateJointCalibration(Fcap, M)$;
// Get residual of parameters vs error configuration, ideally this should be a zero vector.
$res_{param} \leftarrow e_c - Y$;
$statsArr$.append($\{res_{param}, Y, RMS_x, RMS_y, status\}$);
end
return $statsArr$;

Algorithm 4: Process to execute algorithm 2 and algorithm 3 for many joint error configurations.

5.2 Test Configurations

Basic models for joint position sensor noise (section 4.1.1) and joint errors are defined with maximum error range known to be within $\pm 5^\circ$ [31]. Evaluations were performed to include these conditions under different levels of influence as well as limiting number of calibration poses used. The evaluation combinations are listed in table 5.1, the aim is to identify how each property affect overall effectiveness of calibration while gradually increasing difficulty/ challenge while reaching as much realism as possible.

For example, it can be assumed that ground plane-grid method yield better results than several calibration patterns as former provides dense amount of calibration points compared to latter. Another example, sensor noise modelling, results should be better when noise is not in existence.

Another important fact is that all these tests are performed one leg + head joints at a time. The reason is that including more joints increase complexity of the problem to be solved. All tests were done with both left and right legs, with similar results and values are shown for left leg only.

Following subsections will explore these parameters/ constraints which evaluations are performed.

Parameter	Options
Joint error range	$\pm 6.5^\circ$
Calibration feature type	ground plane Multiple calibration patterns
Number of calibration pose	3 5 8
Solver	Levenberg-Marquardt Lev-Mar. with random starts
Sensor noise source	No noise Pixel positioning noise (of calibration feature points) Joint sensor noise Noise from both of above
Total evaluations:	**48**

Table 5.1: *Evaluation configurations.*

Parameter	Value
Camera image size	640×480 pixels. (Floating point arithmetic used for projections to reduce errors)
Random distribution	Uniform Random Distribution
Generated error sets	10000 test cases
Solver	Levenberg-Marquardt solver

Table 5.2: *Common settings for all tests.*

5.2.1 Joint Error Generation

In order to connect calibration features to camera, only the leg(s) that support the robot at a given pose can be considered, therefore error generation may only consist of producing joint errors for a given leg if all calibration poses only comprise that leg/ foot. These errors are only limited to the joints specified in table 5.3 since the arms do not affect camera perception or other crucial tasks in the context of robot soccer.

Example: All calibration poses are from left support foot. Thus, all joint errors should only affect the resulting chain. Since the "population size" is very large (ie: 100^8 assuming 0.1° granularity, based on joint sensor resolution and $\pm 5^\circ$ error range

Joint Number	Joint Name	Short Name
1	Head Yaw	HEAD_Y
2	Head Pitch	HEAD_P
3	Left Hip Yaw Pitch	L_HIP_YP
4	Left Hip Roll	L_HIP_R
5	Left Hip Pitch	L_HIP_P
6	Left Knee Pitch	L_KNEE_P
7	Left Ankle Pitch	L_ANKL_P
8	Left Ankle Roll	L_ANKL_R
9	Right Hip YP = L_HYP (2)	R_HIP_YP
10	Right Hip Roll	R_HIP_R
11	Right Hip Pitch	R_HIP_P
12	Right Knee Pitch	R_KNEE_P
13	Right Ankle Pitch	R_ANKL_P
14	Right Ankle Roll	R_ANKL_R

Table 5.3: *Joints that will be tested for calibration. Note that both Hip Yaw Pitch joints are connected to the same actuator, so calibration for both will be same.*

per joint), a sample size of 10000 was selected. This set of joint errors was generated using random numbers obtained from a generator with uniform distribution in C++ (std :: uniform_real_distribution <**float**>). Limits for this distribution was selected as in table 5.1.

5.2.2 Calibration Feature Types

Two methods are used here, first being a grid of points lying on ground plane which are in visible range of the Nao. This is the same grid used in observation model usage (algorithm 1). While it is convenient for simulation purposes given it provides numerous points increasing observability of errors by the Nao, this is not practical in reality.

Therefore, calibration patterns placed at multiple places can be considered instead. They would be ChAruco patterns as used by HULKs already. For simplicity in hardware, logistics, and previous experience of difficulties in employing vertical calibration patterns as mentioned in fig. 3.3, it is also assumed that these patterns will be on ground plane (ex: fig. 4.4), thus the difference between these two options being the number of points available for the NAO to see.

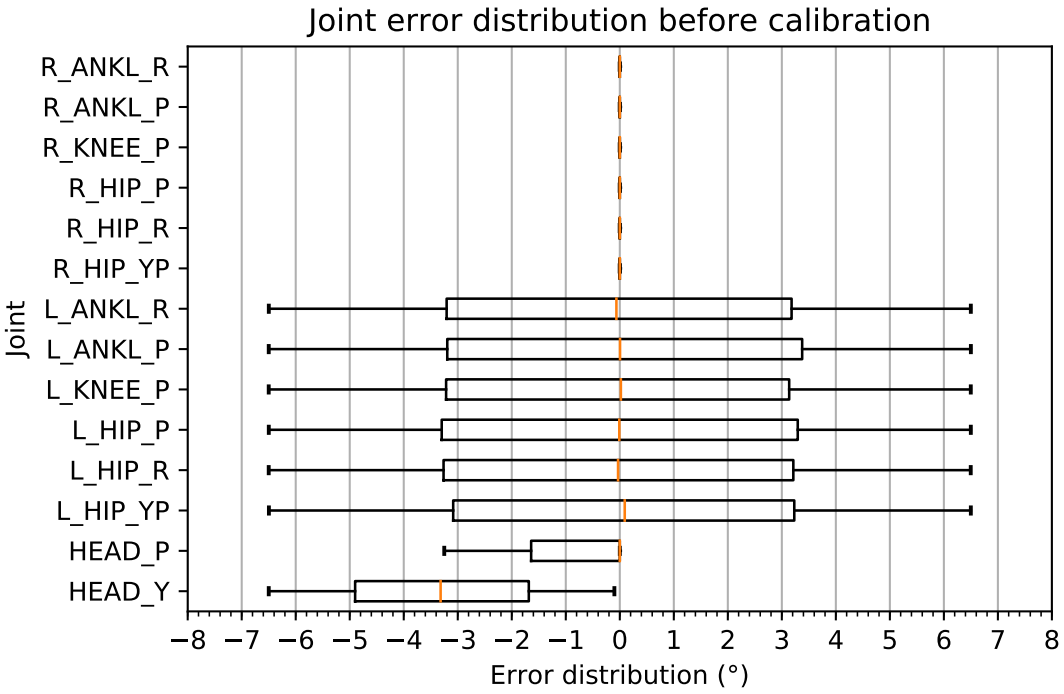


Figure 5.1: *Input joint error distribution. Calibration will be performed on an error dataset with this statistical distribution. It only shows errors induced for left leg and head joints due to reasons explained in section 5.2. The joint names are based on table 5.3.*

5.2.3 Number of calibration poses

Although an indirect aim of this thesis is to generate minimum amount of calibration poses in order to speed up the process, it is generally agreed upon that higher number of captures within reason can improve quality as explained in section 3.3.4, particularly to cancel out sensor noise (assuming they are zero mean Gaussian noise).

However, too many poses are also detrimental for quality according to previous section 3.3.4. Therefore, this factor is also tested. Prior to this finalized presentation of results, testing was done with 3, 4 and up to 8 or 10 poses and it was discovered that 3 or 4 poses usually resulted in the highest success rates without noise inputs. Another fact is that it is possible to take multiple samples at the same pose, this should average out zero mean noise to an extent. Therefore, it is also considered at a later stage of the testing.

Parameter	Options
Joint error range	$\pm 6.5^{\circ}$
Calibration feature type	ground plane
Number of calibration pose	3 8
Solver	Levenberg-Marquardt Lev-Mar. with random starts
Sensor noise source	No noise

Table 5.4: *Evaluation configurations for initial evaluation round.*

5.2.4 Sensor Noise Source

This factor could perhaps be introduced as one of the most important factors in order to achieve near-real life simulation. The recorded data and statistics depicted in fig. 4.1 and section 4.1.1 was used. Based on it, noise was assumed to be a zero mean Gaussian distribution with standard deviation of 0.007. This is depicted in fig. 4.2.

5.3 Initial Evaluations and Observations

First round of evaluations were done to verify validity of these derived poses and the methodology in general. Over the course of these tests, several observations were made and improvements were introduced.

5.3.1 Solvers

Although it was planned to use global optimization algorithms from the beginning, due to prior experience and intuition, the appeal of very fast convergence of Levenberg-Marquardt [13] (often in a few iterations - under $100ms$) and abundance of high quality implementations encouraged the author to start with this algorithm. The implementation available with Eigen library [32] was used.

5.3.2 Local Minima problems

It was noticed that while reprojection error was low ($\pm 3 pixels$ approx), the residual of joint parameters had values as large as maximum error possible. Further inspection of induced error values vs the parameters given by the solver, it was determined that the

solver got stuck at local minimums. This was considered as a failure and the rate was about 60-70%.

5.3.3 Global and Global-like Optimization Attempts

Since the failure rate with the local optimizer was unacceptable, attention for possible global optimization strategies was given. The author briefly tried the following approaches.

- dlib::find_min_global() [33].
- igl::pso (Particle Swarm Optimization) [34].
- Stochastic initial guess with Lev-Mar. [13, 32], retried up to a given amount.

The first approach sometimes converged but the results were generally inferior to Levenberg Marquardt algorithm (when it converged) and was slower (it was a non-derivative based solver). Next candidate was particle swarm optimization, it almost never had any convergence, thus a full test run was not attempted. Perhaps these results were not perfect due to the expertise required to successfully utilize these algorithms were not sufficient, or they were not suitable for the problem at hand.

The last attempt was not strictly a global optimization method, but rather using a random initial guess to act as a mutation phase in genetic algorithm terminology which is used to escape a local minima [35]. At first, calibration is run as usual with Levenberg-Marquardt and then brute-force this a given number of time (100, 50 times were tested) with fresh, randomly generated initial guess for each iteration. While this approach was computationally expensive, it yielded better results than expected (Last column of table 5.5).

5.3.4 Effect of Derivative Calculation Method

When using algorithms that require calculation of Jacobian matrix of a function for solving [13], it is highly important to ensure that the function in question is differentiable and the approach is suitable for the task. This subsection discusses impact of choosing these methods when calibration was tested with Levenberg-Marquardt algorithm which is a derivative-based method. As mentioned before, implementation from Eigen was used [32].

While Eigen library supports both automatic and numerical differentiation, only the latter was used due to need of re implementing all kinematic calculations with a different template type. However, automatic differentiation yields superior results.

The default setting for numerical Jacobian approach uses Forward differentiation which is faster but also less precise. Switching to Central Differentiation method dropped failures approximately by 10%.

Distribution (Uniform) Range	Lev.-Mar.	dlib::global	igl::PSO	stochastic
$\pm 6.5^\circ$	13%	N/A	N/A	1.9%
1° to 6.5° and -1° to -6.5°	54%	N/A	N/A	3.2%

Table 5.5: *Failure rates for solvers tested. Lev-Mar : Levenberg-Marquardt [13, 32], dlib::global [33], igl::PSO : Particle Swarm Optimizer in libigl [34], stochastic : random start positions with Lev.-Mar. solver. The reason for avoiding $\pm 1^\circ$ in second distribution is to eliminate effect of $0-0=0$ on the errors, thus artificially inflating the perceived quality of calibration.*

5.3.5 Results

Below are the results for each step of the above experiment.

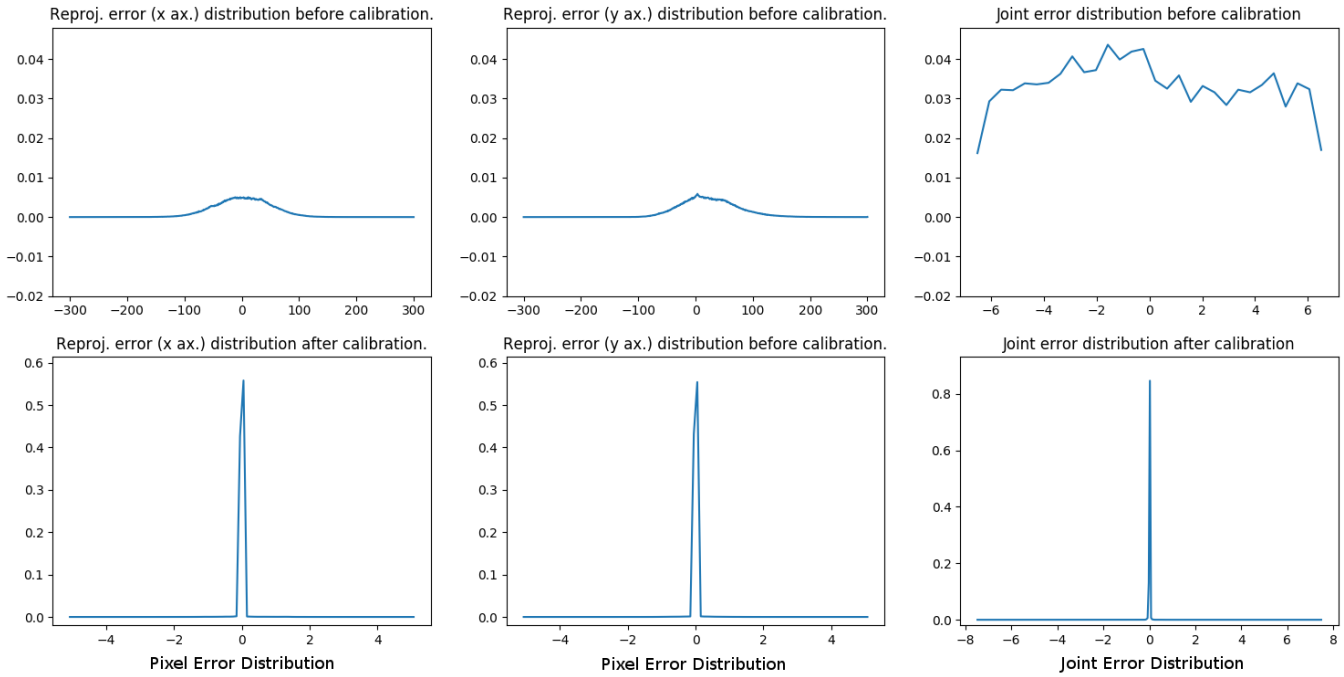


Figure 5.2: *Error distribution for $\pm 6.5^\circ$ error set. Top row is before calibration and Bottom row is after calibration.*

Based on results of this evaluation, it was evident that there is potential for further analysis so following sections cover important configurations which exhibited various results.

5.4 Ground Plane Feature with Simulated Sensor Noise

For the next phase of realism, sensor noise was introduced with other settings unchanged from initial evaluation described in table 5.4. The reason for not immediately switching to sparsely positioned calibration patterns was due to need of identifying most suitable set of poses and effect of number of poses. By determining optimal amount of poses, identifying potential positions for calibration patterns for testing was simplified.

	3 Poses			5 Poses			8 Poses		
Noise Src.	Succ.%	LocMin.%	Other%	Succ.%	LocMin.%	Other%	Succ.%	LocMin.%	Other%
noNoise	**87.02**	12.94	0.04	85.90	14.04	0.06	83.94	15.90	0.16
pixel	**86.36**	13.58	0.06	84.64	15.30	0.06	83.82	16.04	0.14
joint	47.26	52.66	0.08	53.74	46.22	0.04	**58.30**	41.38	0.32
both	46.56	53.36	0.08	53.38	46.58	0.04	**57.32**	42.52	0.16

Table 5.6: *Results from testing with ground-plane calibration feature. Abbreviations are, Succ.: Success, LocMin: Local Minima failure. The results depict several trends between calibration pose count, type of induced errors and success/ failure statistics. This is further discussed in section 5.4. The best results from each noise type is marked in boldface.*

The following conclusions can be taken based of results from this set of evaluations.

1. While minimum number of poses (3) provide the highest success rate for no-noise conditions, it doesn't perform well with added noise.
2. The Highest number of poses (8) performed 6.4% better than minimal pose setup with joint noise alone and pixel + joint noise. However, success rate was reduced by 1.1% under no-noise.
3. Pixel positioning noise affect success rate with more poses. Potentially due to effect of noise being higher than benefit of more captures.
4. However, in case of joint noise, more poses clearly improved results.
5. The probable reason is the joint noise probably affected camera positions too extremely causing to miss the calibration featured. Even if not, using pose estimation of camera could have helped.

Next phase of evaluations with calibration patterns which provides a sparse set of points might provide more insight into this matter.

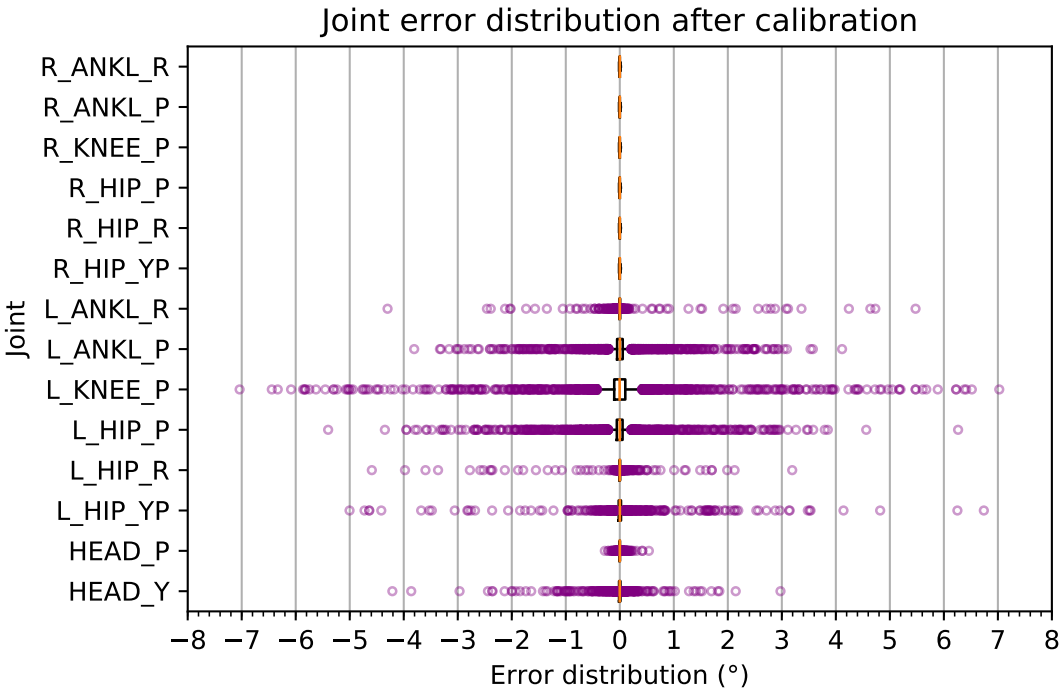


Figure 5.3: *Error distribution of ground plane based calibration with both joint and pixel noise with 8 poses and 3 samples per pose (Best scenario in table 5.6. Although there is a significant amount of outliers, majority (25%-75%) of the population is between* $\pm 0.5°$.

5.5 Multiple Calibration Patterns with Simulated Sensor Noise

Based on previous experiment's results, it was noticeable that higher amount of poses is needed in order to be robust against joint sensor noise. But at the same time, increasing the number of captured calibration features (with more poses) was detrimental. This evaluation will consider effects of reduced set of calibration features which is the more practical scenario for actually calibrating robots by placing a few calibration patterns.

Test configuration is same as section 5.4 except for calibration feature type. Therefore, this series of tests can be directly used to compare effect of calibration features.

The following conclusions can be taken based of results from this set of evaluations.

1. Unlike ground-plane scenario, there is improvement for every noise configuration with more poses. Thus, 8 poses provide best results.

	3 Poses			5 Poses			8 Poses		
Noise Src.	Succ.%	LocMin.%	Other%	Succ.%	LocMin.%	Other%	Succ.%	LocMin.%	Other%
noNoise	0.00	0.00	0.00	67.52	32.40	0.08	84.08	15.90	0.02
pixel	0.00	0.00	0.00	38.98	60.88	0.14	68.00	31.88	0.12
joint	0.00	0.00	0.00	38.52	61.38	0.10	54.32	45.62	0.06
both	0.00	0.00	0.00	27.42	72.44	0.14	45.40	54.50	0.10

Table 5.7: *Results from testing with Charuco calibration features on ground. The results depicts several trends between calibration pose count, type of induced noise and success/ failure statistics*

2. While increase of local minima failures between joint only and joint + pixel noise cases was $2 \rightarrow 4\%$ previously, now it is almost 10% except for 8 poses (6.6%).

3. There is a drastic difference between 3 poses and 5 or 8 poses in terms of no-convergence statistics. This seems to demonstrate that there is a minimum number of data points to be captured in order to converge (at least to a local minima).

4. Analysing these results together with previous experiment show that capturing a certain amount of calibration features is necessary while too much with noise reduce quality.

Next section explore application of global-like solver method experimented in section 5.3.3 which attempts usage of random generated initial positions to reduce failure rate.

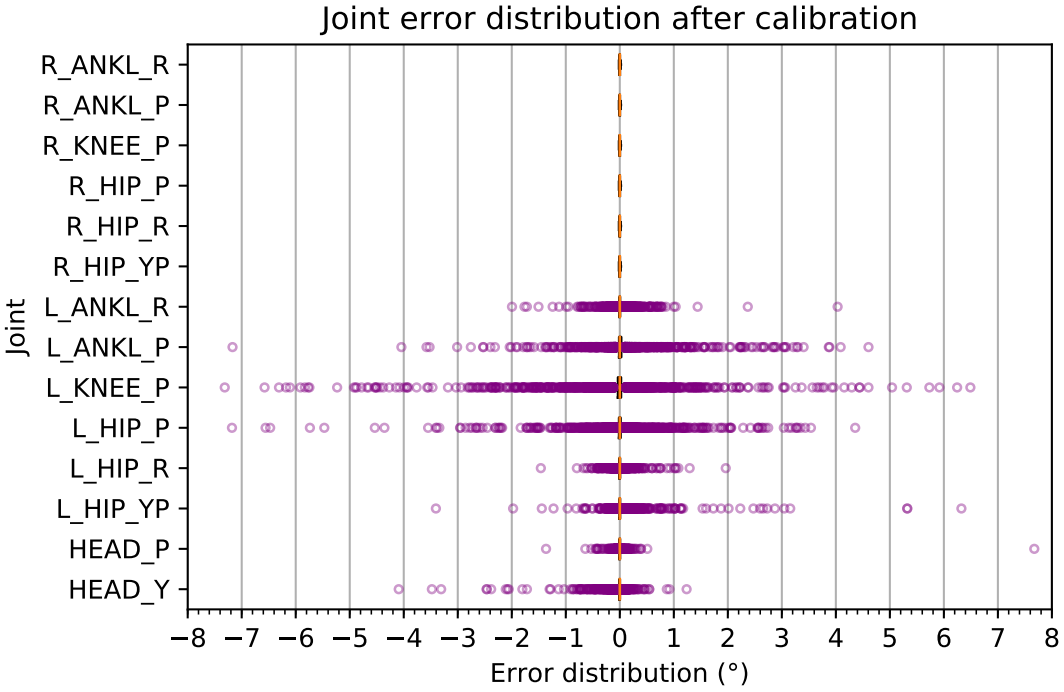
5.5.1 Using Random Initial Guesses

Since the results for section 5.3 showed good results for the method of random start points for Lev-Mar. solver and brute forcing this for 50 times or similar, the same method was tested for the case of calibration patterns on the flow under the same conditions as the previous test with exception to the solving method used.

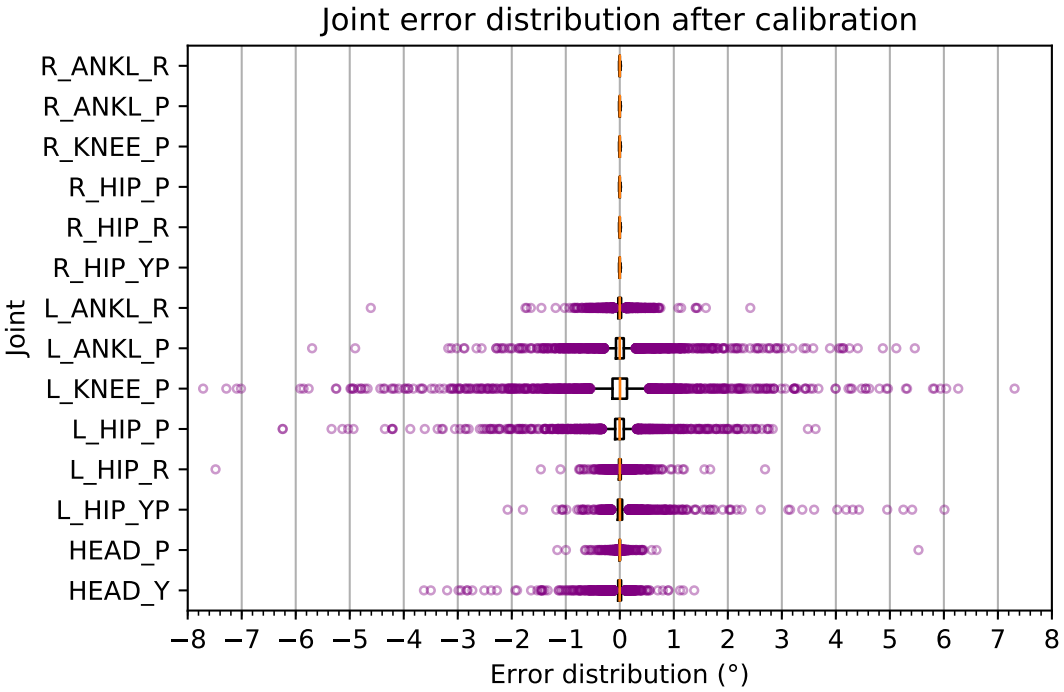
Test configuration is same as section 5.4 except for calibration feature type. Therefore, this series of tests can be directly used to compare effect of calibration features.

Based on the results in table 5.8, it is clearly evident that the local minima issue can be avoided to some extent by utilizing a random generation based method to try multiple start points for the solver, thereby behaving similar to mutation phase in genetic algorithm to attempt in escaping current local minima.

Next section will discuss these results alongside the goals of this thesis.

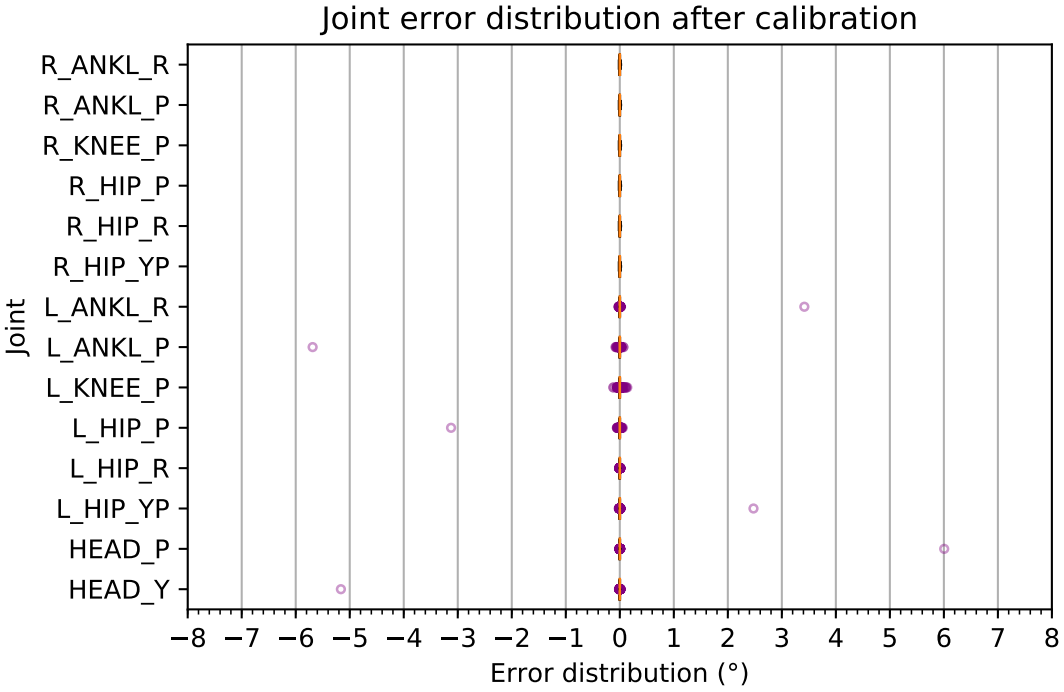


(a) *Error distribution of calibration with no noise input, 8 poses per pose and calibration patterns on ground. Although there is a significant amount of outliers, almost the entire majority of the distribution is between* $\pm 0.2°$.

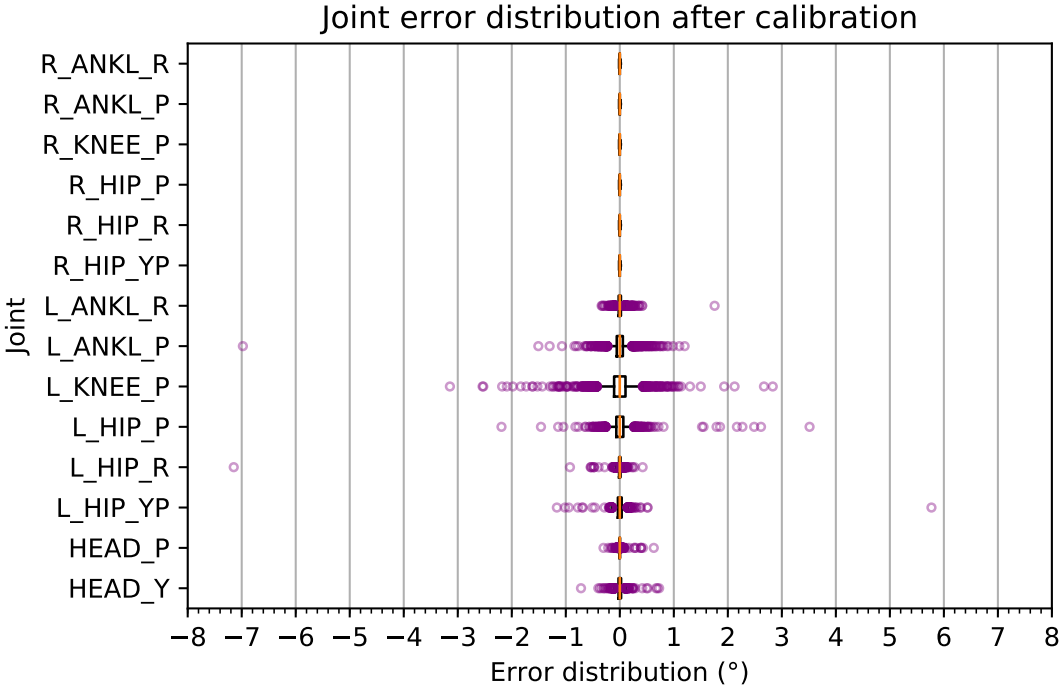


(b) *Error distribution of calibration with noise from both sources, 8 poses, 3 samples per pose and calibration patterns on ground. The amount of outliers is higher than the case of fig. 5.3, majority of the absolute error is still under* $0.8°$.

Figure 5.4: *Error distribution of tests using calibration pattern.*



(a) *Error distribution of calibration with no noise input, 8 poses per pose and calibration patterns on ground + random start method with Levenberg Marquardt.*



(b) *Error distribution of calibration with no noise input, 8 poses, 3 samples per pose and calibration patterns on ground + random start method with Levenberg Marquardt.*

Figure 5.5: *This shows the best set of results so far for the sparse calibration feature input. The outliers barely exceed* 0.4° *in the case of no noise and majority of the distribution was under* 0.5°

	3 Poses			8 Poses		
Noise Src.	Succ.%	LocMin.%	Other%	Succ.%	LocMin.%	Other%
noNoise	99.18	0.30	0.52	99.98	0.00	0.02
pixel	44.80	54.08	1.12	90.02	9.94	0.04
joint	49.30	50.06	0.64	65.38	34.60	0.02
both	25.94	72.64	1.42	56.90	43.04	0.06
joint_3	77.16	22.22	0.62	89.38	10.60	0.02
both_3	34.38	64.38	1.24	76.28	23.68	0.04

Table 5.8: *Random start method with calibration patterns. This method (initially discussed in section 5.3.3) shows highly successful results for every case better than all other methods. This is further amplified when each capture is taken 3 times for each pose enabling to average and filter some noise input.* ***Joint_3*** *and both_3 indicates the case of 3 samples per capture.*

5.6 Discussion

Joint calibration has proven to be a challenging topic in both industry and cases like the RoboCup. This is even a bigger challenge when the sensor is just a 2D camera which also has to be calibrated.

Considering the requirements and needs of this thesis, chapter 3 contains reviews of state of the art, chapter 4 containing the implementation details, especially including the observation model and optimal pose generation workflow and this chapter extensively tested multiple configurations incrementally to identify weaknesses of the system at different conditions. In addition, with this discussion, the requirements of confirming influence of sensor noise and suitability of on-board sensors will be fulfilled.

Based on the previously shown results, it is evident that the proposed methodology can indeed derive optimal poses which lead to good calibration success rate. Furthermore, these tests covered the topic of using stochastic methods to escape local minima situations to reach the best optimum.

The first set of test runs (section 5.3) validated the basic concept and the operations of the implementations. Then sensor noise was introduced to replicate the real robot as much as possible as the input has significant influence over output. At this point, it was clear that a high number of poses is highly desirable when the sensor inputs are

not perfect. Furthermore, the effect of joint sensor readings appear to affect the results most.

In order to consider the realistic deploying conditions, the case of using four calibration patterns on floor was used. This sufficiently showed that having a lesser number of calibration features to use reduce the robustness of the system. While this was a setback, the results were still promising as the method of using stochastic initial guess was not employed yet.

Finally, to somewhat replicate certain stochastic inspired global solving methods, the simplistic approach of using random initial values were used on the four calibration pattern based method. Furthermore, an additional enhancement was added by averaging 3 samples making the result a simple low pass filter with a window of 3 samples. Based on the results in table 5.8, both these additions greatly enhanced results under sensor noise while the stochastic method alone performed near perfect execution for the test without noise input.

An important factor that has to take into attention is that these tests were performed only for one leg at a time. The next iteration should consider both at the same time or a merging method and integrating this together with extrinsic calibration as performing camera extrinsic calibration together with joint calibration is not reliable as mentioned in [31]. However, this limitation in the testing method presented in this thesis does not invalidate the facts proven, instead these focussed tests revealed finer insights of the behaviour of a calibration system under various conditions.

Although all these evaluations has been performed under simulation due to practical reasons, the introduction of joint and pixel noise contributed to demonstrate the weaknesses of the system under observation noise. At the same time, it was shown that simple filtering techniques might be enough to improve overall results by a significant margin (about 20% improvement when both sensor sources supplied noise).

The next section will present the conclusions and future recommendations.

Chapter 6

Conclusion

In this project thesis, a detailed analysis of possibility to perform joint calibration with on-board sensors was performed and further tests were performed to observe resilience for noise in the inputs. In addition, these tests attempted to consider realistic factors such as using several calibration patterns instead of a dense source of calibration features.

During the literature review, it was discovered that multiple teams had attempted Joint calibration for the Nao and concluded less than successful outcome from their experiments. During this thesis, these concerns were confirmed with the reason due to adjacent joints that are always parallel in the kinematic chain under consideration (ex: knee pitch and hip pitch) where the highest failures were reported on these joints. The same observation can be seen in all the results shown in this thesis, especially under sensor noise. There is no alternative to alleviate this problem other than employing direct measurements or using enough calibration poses.

The use of observation model and specific focus to minimize the effect of having similar observations for two joints causing ambiguity made it possible to obtain good poses that generally had a high success rate of 80% (without noise) even with a local minimizing solver. But more poses and observations were needed to counter the effect of noise to an extent. This portion of the experiment reveal that joint calibration is simply difficult to achieve when sensor noise is present as it severely affects the cost function. Due to this reason, it is difficult to directly conclude if a given calibration run converged on a local minima or not.

While 8 poses are relatively a small number considering that there are 14 joints (or 8 if one leg only) to calibrate, it would be desirable to cut down the number of poses and possibly further optimize these poses such that reaching each pose would be quick in order to practically use these poses in competitions where time is limited.

Although this calibration procedure was not tested on a real Nao, it can be concluded that the use of noise models for joint and image noise should at least be close to realism. Furthermore, testing on NAO should be performed to confirm these findings since the ultimate use case is not for simulation, but to be used with real robots.

Based on the above, it can be concluded that obtaining poses by means of an observation model is more effective than arbitrarily choosing several poses. Furthermore, this selection enabled to sustain relatively acceptable success rate as well. Finally, the modelling of noise enabled to discover the main weakness of this type of calibration.

Although this thesis has proven that there is a feasibility to calibrate cameras and joints with only on-board camera and calibration patterns, it cannot be conclusively decided whether this approach will be sufficient for joint calibration as there is a minimum of 20% failure rate at high noise levels. Therefore, the author would like to recommend direct measurements if feasible, to restrict joints that are being calibrated to reduce the degrees of freedom in the system and conduct extensive tests on real robots replicating the same test conditions to compare differences of the simulated modelling to improve further research.

Bibliography

[1] Darshana Adikari, Felix Wege, Georg Felbinger, Arne Hasselbring, Yuria Konda, René Kost, Pascal Loth, Lasse Peters, Nicolas Riebesel, Thomas Schattschneider, and Felix Warmuth, *Team Research Report 2017*.

[2] E. Vazquez, J. Villemonteix, M. Sidorkiewicz, and É. Walter, "Global optimization based on noisy evaluations: An empirical study of two statistical approaches", *Journal of Physics: Conference Series*, vol. 135, p. 012100, nov 2008.

[3] T. Kastner, T. Röfer, and T. Laue, "Automatic robot calibration for the NAO", in *Applied Cryptography and Network Security* (D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, F. Bao, P. Samarati, and J. Zhou, eds.), vol. 7341, pp. 233–244, Springer Berlin Heidelberg.

[4] Thomas Röfer, Tim Laue, Arne Hasselbring, Jannik Heyen, Bernd Poppinga, Philip Reichenberg, Enno Röhrig, and Felix Thielke, "B-human team report and code release 2018".

[5] W. Khalil, M. Gautier, and C. Enguehard, "Identifiable parameters and optimum configurations for robots calibration", vol. 9, no. 1, p. 63.

[6] J. Zhou, H.-N. Nguyen, and H.-J. Kang, "Selecting optimal measurement poses for kinematic calibration of industrial robots", vol. 6, p. 291389.

[7] J.-H. Borm and C.-H. Menq, "Experimental study of observability of parameter errors in robot calibration", in *Proceedings, 1989 International Conference on Robotics and Automation*, IEEE Comput. Soc. Press.

[8] Y. Sun and J. M. Hollerbach, "Observability index selection for robot calibration", in *2008 IEEE International Conference on Robotics and Automation*, pp. 831–836, IEEE, 2008.

[9] Y. Wu, A. Klimchik, S. Caro, B. Furet, and A. Pashkevich, "Geometric calibration of industrial robots using enhanced partial pose measurements and design of experiments", *Robotics and Computer-Integrated Manufacturing*, vol. 35, pp. 151 – 168, 2015.

[10] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier, "The NAO humanoid: a combination of performance and affordability",

[11] SoftBank Robotics, *NAO Video Camera*.

[12] SoftBank Robotics, *Joint Position Sensors*.

[13] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory", in *Numerical analysis*, pp. 105–116, Springer, 1978.

[14] Rico Tilgner, Thomas Reinhardt, Tobias Kalbitz, Stefan Seering, Michael Wünsch, Jonas Mende, Hannes Hinerasky, and Jörg Schließer, *Nto-Team HTWK Team research report*. Nao-Team HTWK.

[15] H. Mellmann, B. Schlotter, S. Kaden, P. Strobel, T. Krause, C.-N. Ritter, T. Hübner, and S. Tofangchi, "Berlin united - nao team humboldt team report 2016", techreport, Adaptive Systeme, Institut für Informatik, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany, 2016.

[16] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming", vol. 51, pp. 481–491.

[17] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers", vol. 76, pp. 38–47.

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

[19] G. Ferney and S. Folkman, "Development of a method for characterizing joint stiffness, deadband, and hysteresis", *Shock and Vibration*, vol. 2, no. 4, pp. 289–295, 1995.

[20] H. Schwenke, W. Knapp, H. Haitjema, A. Weckenmann, R. Schmitt, and F. Delbressine, "Geometric error measurement and compensation of machines—an update", *CIRP Annals*, vol. 57, no. 2, pp. 660 – 675, 2008.

[21] D. Maier, S. Wrobel, and M. Bennewitz, "Whole-body self-calibration via graph-optimization and automatic configuration selection", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5662–5668, IEEE.

[22] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[23] A. Nahvi and J. M. Hollerbach, "The noise amplification index for optimal pose selection in robot calibration", in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, pp. 647–654 vol.1, April 1996.

[24] S. Tabandeh, C. Clark, and W. Melek, "A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering", in *2006 IEEE International Conference on Evolutionary Computation*, pp. 1815–1822, July 2006.

[25] J. Kennedy, "Particle swarm optimization", in *Encyclopedia of Machine Learning and Data Mining*, 2017.

[26] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization", in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 3, pp. 1945–1950, IEEE, 1999.

[27] RoboCup Technical Committee, *RoboCup Standard Platform League (NAO) Rule Book*.

[28] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The development of honda humanoid robot", in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 2, pp. 1321–1326, IEEE, 1998.

[29] J. J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-stable motion planning for humanoid robots", *Autonomous Robots*, vol. 12, no. 1, pp. 105–118, 2002.

[30] SoftBank Robotics, *NAO collision avoidance*.

[31] T. Kastner, T. Röfer, and T. Laue, "Automatic robot calibration for the NAO", in *Applied Cryptography and Network Security* (F. Bao, P. Samarati, and J. Zhou, eds.), vol. 7341, pp. 233–244, Springer Berlin Heidelberg.

[32] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3". `http://eigen.tuxfamily.org`, 2010.

[33] D. E. King, "Dlib-ml: A machine learning toolkit", *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[34] A. Jacobson, D. Panozzo, *et al.*, "libigl: A simple C++ geometry processing library", 2018. https://libigl.github.io/.

[35] M. Bouchouicha, M. B. Khelifa, and W. Puech, “A non-linear camera calibration with genetic algorithms”, in *Seventh International Symposium on Signal Processing and Its Applications, 2003. Proceedings.*, vol. 2, pp. 189–192, IEEE, 2003.