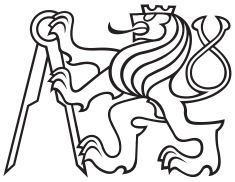
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Artificial skin calibration for a humanoid robot: comparing or combining "self-touch" and 3D reconstruction from images

Bohumila Potočná

Supervisor: Mgr. Matěj Hoffmann, Ph.D.
Field of study: Cybernetics and robotics
August 2020

Acknowledgements

Firstly I would like to thank Matěj Hoffmann, who supervised this thesis, for giving me the opportunity to participate on this project, guiding me through the work and correcting my mistakes. I would like to thank Tomáš Pajdla and Michal Polic for their professional advice in the field of 3D surface reconstruction, which contributed greatly to the quality of the results. I thank Lukáš Rustler for sharing his data and knowledge about the self-touch calibration framework I could build on, and for being always available to answer my questions and help me solve all the technical problems.

I thank my family for being a continuous support through my studies. And finally I would like to thank CTU in Prague for being a good *alma matter*.

Declaration

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

Prague, date August 14, 2020

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

V Praze dne 14. srpna 2020

Abstract

This work is focused on the calibration of tactile sensor positions on a robot body. We are working with an artificial skin consisting of triangular modules mounted on a humanoid robot Nao. This project builds on a previous work which provided a kinematic model of the robot with skin and performed several calibrations using the self-touch method. The purpose now is to improve the accuracy of estimated skin elements' position by adding a single-point end effector to one of the kinematic chains in self-touch configurations and by providing a model of the skin using 3D surface reconstruction from photographs. The single-point end effector improved the estimated positions when combined with the existing datasets consisting of configurations with two touching skin parts. The 3D reconstruction process provided very accurate mutual positions of the tactile sensors and significantly reduced the number of parameters needed to be optimized.

Keywords: artificial skin, calibration, tactile sensors, self-touch, robot kinematics, forward kinematics, 3D surface reconstruction, robot body shape, Nao, Humanoid robots

Supervisor: Mgr. Matěj Hoffmann, Ph.D.

Abstrakt

Tato práce se zabývá kalibrací pozice taktilních senzorů na těle robota. Pracuje s umělou kůží sestávající z trojúhelníkových buněk, nasazenou na humanoidního robota Nao. Tento projekt navazuje na předchozí práci, jež poskytla kinematický model robota i s kůží a v rámci níž bylo provedeno několik kalibrací pomocí sebedotykových konfigurací. Cílem nyní je zvýšit přesnost odhadu pozic prvků umělé kůže přidáním bodového koncového efektoru na jeden z kinematických řetězců a vytvořením modelu kůže pomocí 3D rekonstrukce z fotografií. V kombinaci s existujícími datasety sestávajících z doteků dvou částí kůže přidání dotekových konfigurací s bodovým koncovým efektorem zlepšilo odhad parametrů. Proces 3D rekonstrukce poskytl velmi přesný model pozic taktilních senzorů, čímž výrazně snížil počet parametrů, jež je nutné optimalizovat.

Klíčová slova: umělá kůže, kalibrace, taktilní senzory, sebedotyk, kinematika robota, přímá kinematika, 3D rekonstrukce povrchu, tvar těla robota, Nao, Humanoidní roboti

Překlad názvu: Kalibrace robotické kůže humanoidního robota: porovnání a kombinace "sebedotykových" konfigurací a 3D rekonstrukce z fotografií

Contents

Chapter 1

Introduction

1.1 Motivation

Sense of touch is one of the useful features for collaborative robots. It is typically realized using tactile sensors scattered around the robot body, forming an artificial skin. As the skin is usually added manually to the robot, there's a lot of uncertainty in the position of its elements w.r.t. the robot body. Furthermore the sensors tend to wear out often, therefore it is desirable for the artificial skin to be easily replaceable and its position to be updated in the software after the replacement. These problems can be solved by calibration.

The tactile sensors in the artificial skins are usually organized into bigger units such as modules (or cells)—typically triangular or hexagonal—and patches. In our case we are working with a skin consisting of triangular modules, which was originally used on iCub [10], imported onto a Nao robot. Given a 2D model of the skin patches and robot body dimensions provided by the manufacturer, we need to get a 3D model of the skin attached to the robot body, particularly the positions of the tactile sensors as accurate as possible.

Previous self-touch skin calibrations performed on Nao robot, where configurations of two skin parts touching were used, brought promising results. We believed the result could be improved, firstly by adding a single-point end effector to one of the chains touching, and secondly by providing an accurate layout of the individual tactile sensors by making a 3D model of the skin parts using 3D surface reconstruction.

1.2 Goals

The goals of this thesis:

1. Calibration using 3D reconstruction
   - Collect images of Nao robot with exposed skin using an external camera
   - Reconstruct 3D shapes of the robot skin and retrieve coordinates of tactile elements from the model
   - Combine the results with available information about the skin and existing calibrations
2. Calibration using self-touch configurations
   - Collect data from the artificial skin in self-touch configurations with an added end effector
   - Perform the skin spatial calibration

1.3 Related work

In the past few years, several approaches to robot kinematic calibration have been presented, using various sensors and devices. In 2011, Del Prete et al. [7] used force/torque sensors for measuring the applied force in the area of activated tactile sensors and estimating the contact point as the intersection of the force axis. Another tool commonly used in calibration is a camera, either external or mounted on the robot's body. They are typically used in combination with other sensors such as accelerometers [13] or tactile sensors from artificial skin [3], [17] with a purpose to reconstruct a 3D model of the body part and skin patch.

To remove the necessity of using external tools or additional sensors (as some robot platforms may not provide them and their installation could be expensive) a method using data from tactile sensors and robot's joints only has been introduced [15]. This method—known as self-touch—takes measured activations from the skin and joint coordinates as input and estimates the positions of given links through forward kinematics. The self-touch configurations can be produced either manually or, better, autonomously, which however requires determining the contact points by inverse kinematics and very careful planning of the configurations to avoid unwanted collisions (described in [15]).

Appropriate calibration methods may vary with the robot platform used. A very suitable subject for most of the approaches mentioned above is the iCub robot (used in [7], [15] and [17]), as it's equipped with force/torque sensors, accelerometers, artificial skin and cameras, too. The Nao robot used in [16] and in this work has been additionally equipped with the same artificial skin mounted on removable plastic parts (more details in Chapter Materials and Methods).

Chapter 2

Materials and Methods

2.1 NAO robot and artificial skin

The hardware consists of NAO humanoid robot made by Aldebaran Robotics, version H25 [4], extended by the artificial skin—a system of tactile sensors (taxels) placed on triangle-shaped flexible printed circuit boards (FPCB) developed by Maiolino et al. [10]. These triangles are assembled into patches, which are mounted on custom-made plastic holders and attached to the robot's body parts - head, torso and hands. The signals from patches are processed by a microcontroller situated on robot's back. [16] Figure 2.1 shows the proportions of the original NAO robot and figure 2.2 shows the robot with mounted skin.

The radius of an individual taxel is 2.2 mm. Within this work, some additional measurements of the triangle cell were taken: edge length, triangle height, distance between the middle taxel and edge and distance between the neighbouring taxels. All these proportions are marked in Figure 2.3.

2.2 Additional end effector

In addition for this work, a metal finger-like end effector has been attached to each of the hands. When performing the self-touch calibration in [16] , there was always a larger area of taxels activated. The aim now is to be able to activate fewer (1-3) taxels at a time in order to achieve better accuracy.

We had several parts available to assemble end effectors of different length and thickness. (See Figure 2.4.) After testing several combinations the shortest finger—made from the 10mm mount and 20mm tip—had the best reach to the skin parts and the thinnest tip had the best accuracy. The final finger shape mounted on Nao's hand is depicted in Figure 2.5.

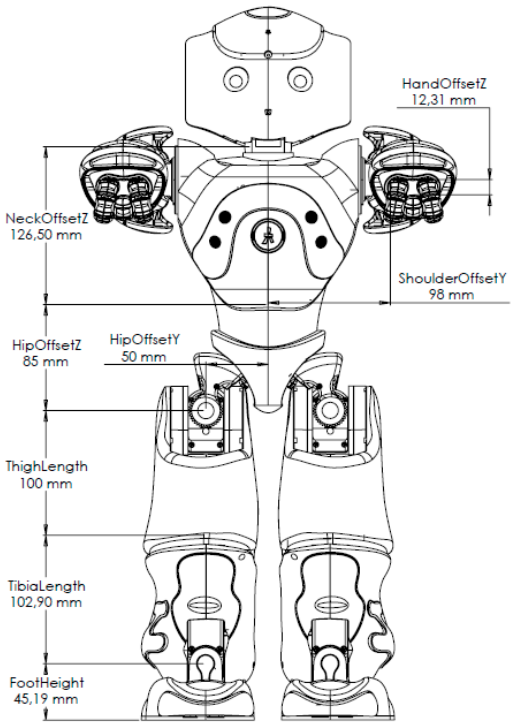


Figure 2.1: Proportions of NAO H25 from Aldebaran website [4].

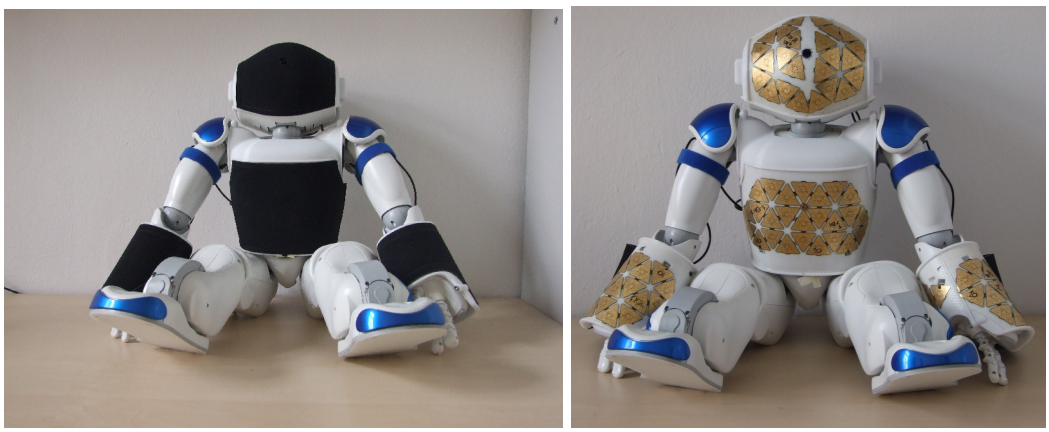


Figure 2.2: NAO with mounted skin: with and without protective layer.

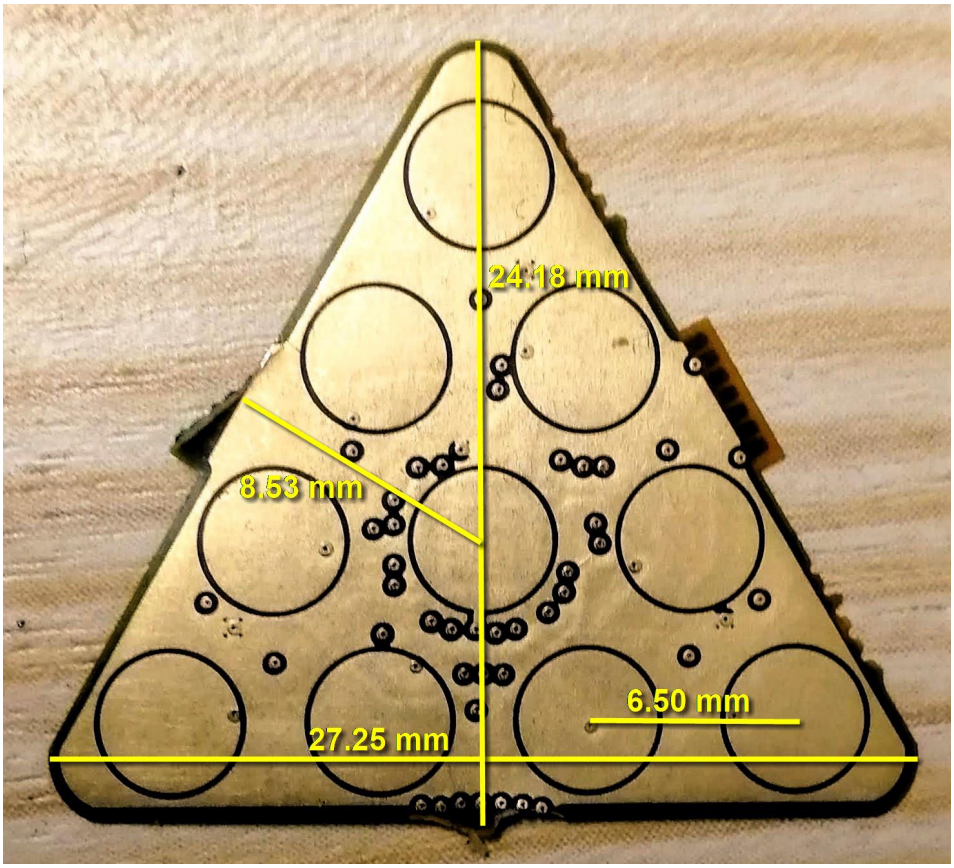


Figure 2.3: Triangle cell proportions obtained through manual measurements with a digital caliper.

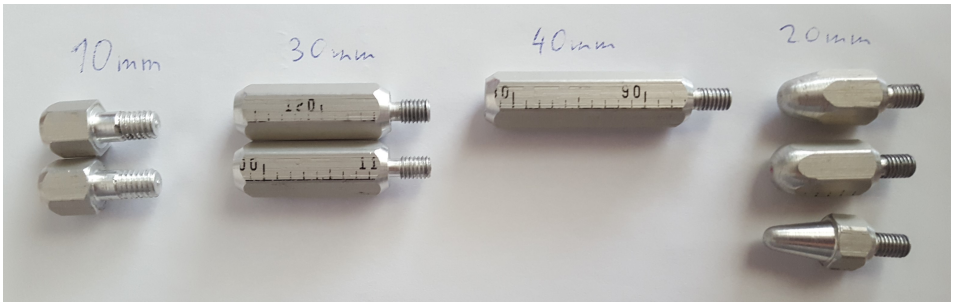


Figure 2.4: Available parts for the end effector and their lengths. In the end the mounts from left and the tip from bottom right were used.

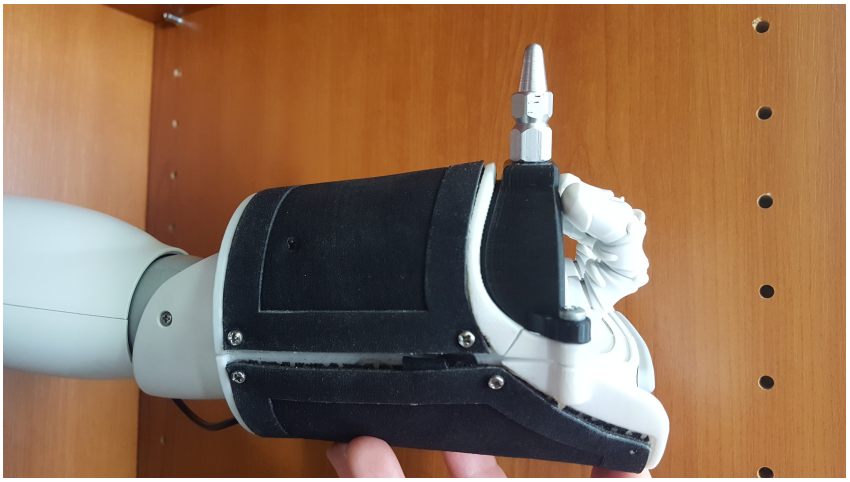


Figure 2.5: Photograph of the final shape of the finger mounted on Nao's hand.

2.3 Mathematical model of the robot

Geometry of a robot consists of links and joints. Each two neighbouring links are connected by a joint, which determines their mutual position, represented by translation vector L and rotation matrix R, which together form a transformation matrix T_i^{i-1} from link i to link *i-1*. This matrix is dependent on the joint coordinate (extension in case of sliding joint, angle in case of rotational joint).

$$L = \begin{bmatrix} x \\ y \\ x \end{bmatrix}$$

$$R \in \mathbb{R}^{3\times 3}$$

$$T_i^{i-1} = \begin{bmatrix} & R & & L \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3.1 Forward kinematics and Denavit Hartenberg notation

In forward kinematics problem we determine the position of the end effector w.r.t. the base frame P^0, with knowledge of the joint coordinates. The solution is achieved by substitution of the joint coordinates in the transformation matrices T_i^{i-1}, $i = 1, ..., n$ and multiplying the end effector position in its local frame P^n (typically the last joint) by the matrices:

$$P^0 = T_1^0 T_2^1 ... T_n^{n-1} P^n$$

Denavit-Hartenberg (DH) notation introduced by Jacques Denavit and Richard Hartenberg in 1955 [9] provides a way to determine a transformation matrix between two frames by only four parameters (compared to the other ways which use six or more parameters, e.g. translation vector to the reference point and Euler angles).

Parameters used in DH notation:

- d – offset along previous z axis to the common normal
- θ – angle about previous z, from old x to new x
- a – length of the common normal
- α – angle about common normal, from old z to new z

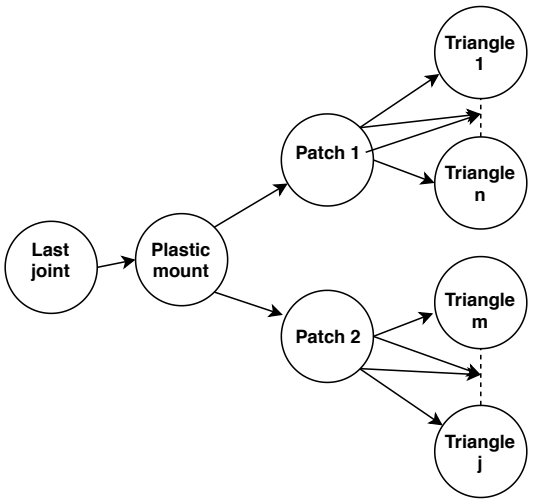
The transformation matrix from link i to link *i-1* is then represented as:

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.1}$$

NAO's DH parameters calculated from official NAO H25 dimensions have been provided by [16]. Apart from the coordinate frames assigned to each joint, several have been assigned to the skin parts:

- mount – transformation to the plastic mount from the last joint
- patches – transformation to each patch from the mount (patch is a subset of triangles with common port, each skin part has two patches)
- triangles – transformation to individual triangles from the respective patch
- taxels – transformation to individual taxels from the respective triangle

Relations of the skin frames are depicted in Figure 2.6. Another frame has been set for the mounted fingers—transformation from the last joint to the fingertip.



.

Figure 2.6: Schematics of hierarchy of links and frames related to the artificial skin. Figure from [16]

2.3.2 Optimization problem formulation

The aim is to optimize the positions of the skin elements, i.e. mounts, patches, triangles and taxels in their local frame (according to the relations stated above). The position is expressed either by DH parameters or 6D transformation (translation vector (x, y, z) and Euler angles (α, β, γ)). Therefore the vector we are estimating looks either like this:

$$\boldsymbol{\phi} = \{[a_1, \ldots, a_N], [d_1, \ldots, d_N], [\alpha_1, \ldots, \alpha_N], [\theta_1, \ldots, \theta_N]\}$$

or like this:

$$\boldsymbol{\phi} = \{[x_1, \ldots, x_N], [y_1, \ldots, y_N], [z_1, \ldots, z_N], [\alpha_1, \ldots, \alpha_N], [\beta_1, \ldots, \beta_N], [\gamma_1, \ldots, \gamma_N]\}$$

where N is the number of calibrated links (skin elements).

The objective function to be optimized is:

$$\boldsymbol{\phi}^* = argmin_{\boldsymbol{\phi}} \sum_{m=1}^{M} ||\boldsymbol{p}_m^r - \boldsymbol{p}_m^e(\boldsymbol{\phi}, \boldsymbol{\Theta}_m)||^2 \tag{2.2}$$

where M is the number of activations in the dataset, $\boldsymbol{p}_m^r$ is the real position of the contact point w.r.t. base frame and $\boldsymbol{p}_m^e$ is the estimated contact point calculated from given parameters $\boldsymbol{\phi}$ and joint coordinates $\boldsymbol{\Theta}_m$ using forward kinematics. As we do not have access to the real contact point positions, we need to make an alternative

solution. We need to take one of the touching links as reference (that means in our case either the fingertip or the touched taxel).

The first chains to be optimized in this work are the mounted fingers, i.e. position of the fingertip w.r.t. the last joint (in the calibration framework referred to as *rightEE* and *leftEE*). In this case only the translation is needed, not the orientation, so the function looks as follows:

$$\boldsymbol{p}_{\boldsymbol{f}}^{*} = argmin_{\boldsymbol{p_f}} \sum_{m=1}^{M} ||\boldsymbol{p}_{m}^{r} - \boldsymbol{p}_{m}^{e}(\boldsymbol{p_f}, \boldsymbol{\Theta}_m)||^2 \tag{2.3}$$

where M is the number of activations in the dataset, $\boldsymbol{p}_m^r$ is the activated position on the reference body part and $\boldsymbol{p}_m^e$ is the estimated end effector position calculated from given fingertip coordinates $\boldsymbol{p_f}$ and joint coordinates $\boldsymbol{\Theta}_m$.

After the fingers' positions are calibrated, they will be used as reference to calibrate the skin parts. The function then changes to:

$$\boldsymbol{\phi}^{*} = argmin_{\boldsymbol{\phi}} \sum_{m=1}^{M} ||\boldsymbol{p}_{m}^{f}(\boldsymbol{\Theta}_m) - \boldsymbol{p}_{m}^{e}(\boldsymbol{\phi}, \boldsymbol{\Theta}_m)||^2 \tag{2.4}$$

where M is the number of activations in the dataset, $\boldsymbol{p}_m^f$ is the position of the fingertip w.r.t base frame calculated from joint coordinates $\boldsymbol{\Theta}_m$ and $\boldsymbol{p}_m^e$ is the estimated contact point on the skin part from given parameters $\boldsymbol{\phi}$ and joint coordinates $\boldsymbol{\Theta}_m$.

When using 3D reconstruction we expect to get relatively accurate positions of individual taxels, which shall be used as reference.

2.3.3 Optimization algorithm used

We are solving a non-linear least squares problem. For this purpose we used Matlab's function *lsqnonlin* from *Optimization toolbox*, which performs the optimization using either *Levenberg-Marquardt* or *Trust region reflective* algorithm [11]. It provides several options to customize the optimization, such as defining the criterion function or adjusting iterations limit, function tolerance or bounds for the optimized parameters.

Chapter 3

Processing data from 3D reconstruction

This chapter describes the steps of obtaining the dataset from 3D reconstruction, starting with photographs and ending with a point cloud of tactile sensors' positions. The whole process is illustrated in Figure 3.3.

3.1 Photographs of the robot with exposed skin

The initial input for this method consists of photographs of the robot taken from different viewpoints by an external camera. The photos have been taken outside in order to get satisfactory light conditions—evenly distributed light, without glares. Behind the robot, we placed a poster so the background would be more diverse. The surface layers of the artificial skin were removed for the photographing so the individual tactile sensors are visible.

For successful 3D reconstruction, the shots needed to be very close to each other. Each point should be on at least three neighbouring pictures. We made three semicircles around the robot with the camera, having approximately 10° spacing between each two shots. The placement of the cameras can be seen in the reconstructed sparse[1] model in Figure 3.1.

Since it was impossible to capture all the skin parts in one posture of the robot (the arm skin part is placed all around the arm's circuit), three series of photos were made, each for a different posture, consisting of 125, 78 and 57 photos, respectively. The individual postures are depicted in Figure 3.2. This way we got at least one piece of position data for each taxel.

[1] Sparse model is an inter-step in the process of 3D reconstruction, consisting of calculated points and camera views.

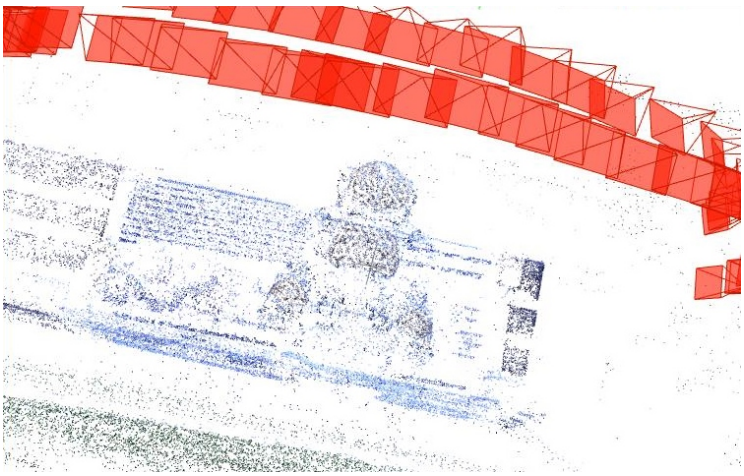


Figure 3.1: Sparse model of Nao, with camera positions used to take individual photographs marked in red.

Figure 3.2: Posture of the robot in dataset 1, 2 and 3

3.2 3D models

The photos were processed in Meshroom and turned into a 3D model (in Figure 3.3 marked as step 1). "Meshroom is a free, open-source 3D reconstruction software based on the AliceVision framework. AliceVision is a Photogrammetric Computer Vision Framework which provides 3D Reconstruction and Camera Tracking algorithms." [6]

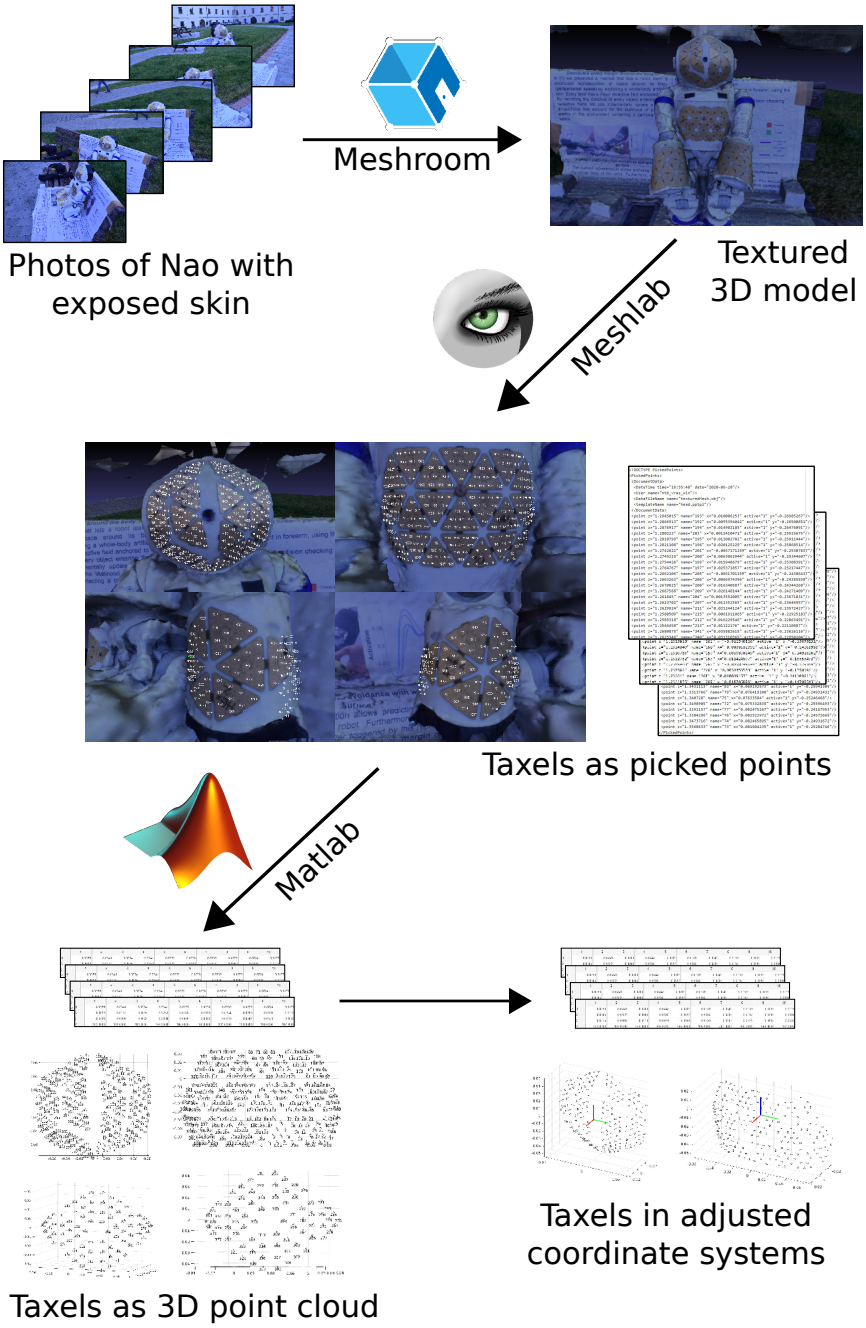


Figure 3.3: 3D reconstruction pipeline: from photographs to 3D point cloud of taxels used in calibration.

Meshroom performs the reconstruction in several steps (nodes). The reconstruction steps used in this work are listed below. For more detailed descriptions of the steps see chapter *Node Reference* in Meshroom manual [6].

1. *CameraInit* - loads images' metadata, extracts camera parameters and generates viewpoints
2. *FeatureExtraction* - extracts features from images and the features' descriptors
3. *ImageMatching* - determines which images should match to each other (i.e. which contain the same areas of the scene)
4. *FeatureMatching* - finds the correspondence between the images from feature descriptors
5. *StructureFromMotion* - reconstructs 3D points from the images
6. *PrepareDenseScene* - undistorts the images
7. *DepthMap* - retrieves the depth value for each pixel
8. *DepthMapFilter* - resolves inconsistencies in depth maps
9. *Meshing* - generates mesh—a dense geometric surface—from the depth maps
10. *MeshFiltering* - excludes unwanted elements from the mesh
11. *Texturing* - creates texture files and projects the texture on the mesh

3.3 Taxel coordinates

3.3.1 Picking points

To get the taxel positions, the textured model was imported to Meshlab, a tool for 3D modelling. This program has a feature called *pick points*, which enables to select points on the surface of a model and save their 3D coordinates into a file. The taxel centers were manually picked according to the texture (in Figure 3.3 step 2). The output of this process was a .pp file containing properties of each selected point - their 3D coordinates and their given names, corresponding to their ID within the patch.

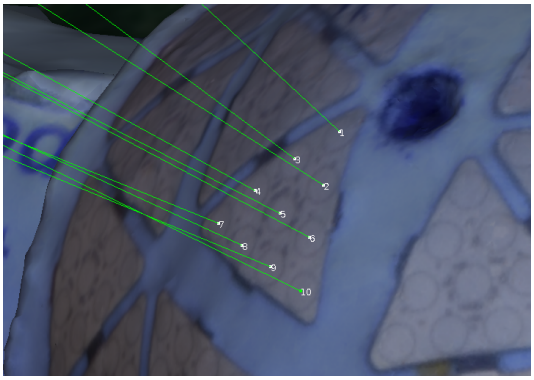


Figure 3.4: Screenshot from picking points in Meshlab

3.3.2 Point transformation

Further processing has been performed in Matlab. First, the coordinates and IDs of the points were extracted from the .pp file into a .mat array (we created the *read_picked_points.m* script for this purpose). Afterwards we needed to move the coordinate system to the correct place. The original models had their coordinate systems generated by Meshroom and so our retrieved taxel coordinates were w.r.t. these systems. However, for using them in the calibration, we needed their coordinates related to a coordinate system with known position due to the robot. Furthermore, their origin needed to be on the robot's surface in order to obtain its coordinates in the original model in Meshlab (the *pick points* feature works only for points on the reconstructed surface), so we could not choose a joint.

The coordinate systems were chosen as follows:

- head: forehead camera as origin, z axis oriented to the top
- torso: chest button as origin, z axis oriented to the top
- hands: taxel 3 (center of triangle with index 0)

The x axis is always perpendicular to the skin surface, oriented outside the robot. For better clarity see Figure 3.5.

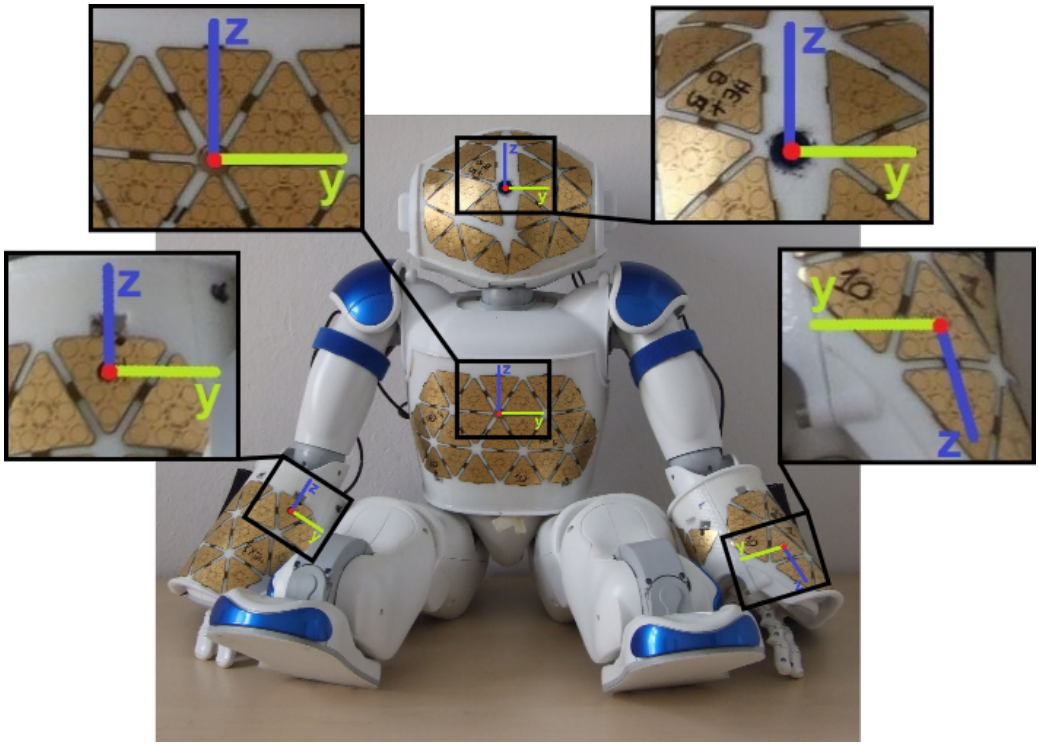


Figure 3.5: Coordinate systems chosen as relation for taxel positions: camera on the head, button on the torso and taxel 3 on the hands.

Since the camera and the button are not on the same level as the skin surface—see figure 3.6—we used the *pick points* function in Meshlab to select 8 points from their circuit and the center (i.e. the coordinate system origin) was calculated as their mean.

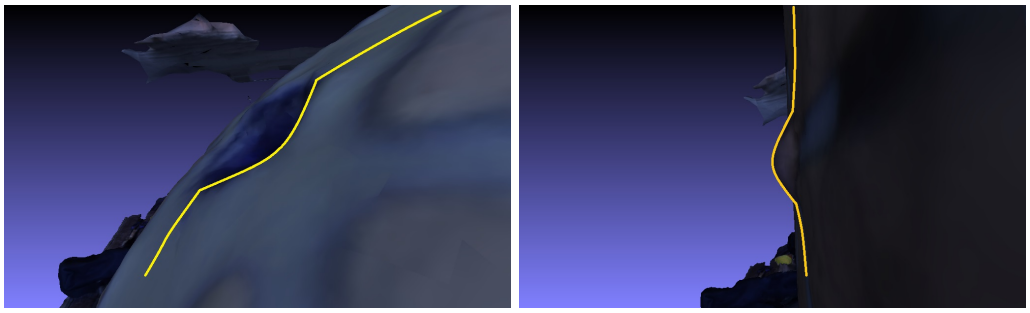


Figure 3.6: Camera (left) and button (right) 3D profiles.

By placing the newly chosen coordinate system into the 3D model of a skin part, we could determine the transformation matrix from the original coordinate system to the new coordinate system. The transformation is performed by the *transformation.m* script in following steps:

1. Load taxel IDs and coordinates of the chosen skin part from .pp file using *read_picked_points.m* function.
2. In case of head and torso load the circuit points from .pp file and compute their mean, i.e. the center of the camera/button, which will be the new origin. In case of arms the coordinates of taxel 3 are taken.
3. Move the points (taxels) into the new origin (by subtracting the new origin's coordinates from the points' coordinates).
4. Determine the new axis by following steps:
   - pick z vector (position of a chosen taxel above the origin)
   - pick y vector (position of a chosen taxel to the right from the origin)
   - calculate x as cross product of z and y
   - recalculate z as cross product of x and y (to ensure the axis are all perpendicular)
   - normalize the vectors
5. assemble the vectors into matrix

$$A = [x\ y\ z]$$

6. multiply the taxels' coordinates by the inversion of A

The skin models in new coordinate systems are shown in figure 3.7.

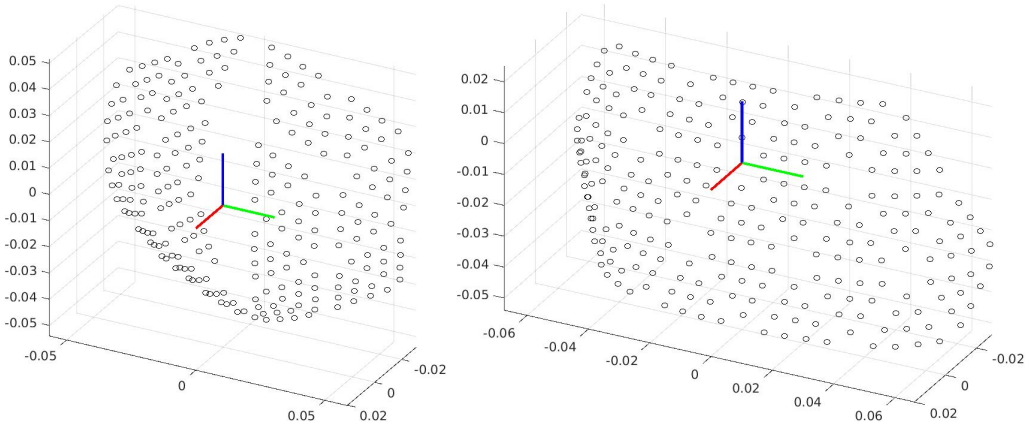


Figure 3.7: Transformed models of skin with axes: head (left) and torso (right).

3.3.3 Scaling

The last property remaining to be changed is the scale. Besides the coordinate system, the 3D model from Meshroom has its own arbitrary scale, which is different from reality. Therefore, we needed to adjust the scale so the distances in the model correspond to the real distances in metres. The scaling coefficient was determined from the mean neighbouring taxel distance: From the measurements described in Section 2.1 and depicted in Figure 2.3 we know that the distance between two neighboring taxels is $6.50mm$. For each dataset we took several pairs of neighbouring points in the model and computed their distance. The mean of these distances was then used to compute the scaling coefficient:

$$k = d_A/d_r$$

where d_A is the computed average distance in the model and d_r is the real distance.

3.3.4 Codes and output files

Photographs and 3D models

The outputs from 3D reconstruction are saved in data/3D_reconstruction folder on Google drive. That includes:

- Photographs of Nao with exposed skin – *photos_17_12_2019/session[i]*, $i = 1, 2, 3$
- Textured mesh files and picked points – */meshroom_reconstruction/dataset[i]*, $i = 1, 2, 3$

Matlab functions and scripts

All functions for processing the reconstructed points can be found in */3D_reconstruction/dense_reconstruct* folder. That includes:

- `read_picked_points.m`
- `transformation.m`
- `scale_from_taxel_dist.m`
- `hands_assemble.m`
- `parsing_for_calibration.m`
- `make_calib_points.m`

Matlab outputs

The outputs from Matlab processing are saved in the same folder in */outputs*. That includes:

- Mat arrays (4xN, where N is the number of collected points) with taxel coordinates transformed to the required frames – */scaled* and */not_scaled*
- Taxel coordinates parsed into the format required by the calibration framework (384x6 array, taxels ordered by index) – */parsed_for_calib/scaled* and */parsed_for_calib/not_scaled*
- Final datasets in */parsed_for_calib* as `[part]_3Drec.mat`

Chapter 4

Processing data from self-touch configurations

The dataset for this method consists of the skin activations and joint coordinates. The activations are performed manually by navigating the robot's arm to touch a selected taxel with the end effector (example depicted in Figure 4.1). Each time a touch is detected, the software collects data about the activated taxel and reads the joint coordinates. These together form an element of the dataset. Because the skin output comes from a different platform than the output from joints, the data has to be collected separately and united additionally by a custom-made software.

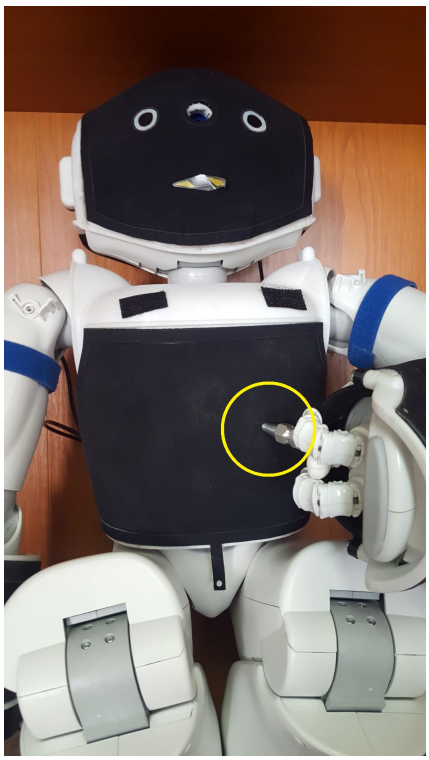


Figure 4.1: Example of a self-touch configuration where left finger touches the torso skin.

4.1 Reading skin data

As mentioned in chapter 2.1, the data from skin patches is collected by a microcontroller on Nao's back. The communication between this board and computer is provided by YARP (Yet Another Robot Platform) [12]. It provides tools for easier control of multiple external devices.

YARP tools used in this work:

- `yarpserver` - a name server, which translates the IP addresses and ports we read from to names
- `yarprobotinterface` - starts all the devices required by a robot
- `yarpmanager` - a utility for running and managing multiple programs on different devices, in this work used for connecting to the skin ports and visualisation of the skin patches

Further processing of the data is provided by iCub's `skinManager` [8]. This software takes the received raw data and performs a heat compensation, making the values independent on the temperature. Additionally it provides the option of following adjustments:

- binarization - Instead of 0-255 format, the taxel values come as either activated (100) or not activated (0). Binarization was set to 'False' in this work.
- reversing data - from range 0-255 to range 255-0
- computing centers of pressure (COP) from the positions of neighbouring activated taxels (maximum neighbour distance can be set as *maxNeighbourDist* parameter)
- include index of the skin part in each activation

To work properly the `skinManager` needs the configuration file `skinManAllNao.ini` and text files with taxel positions.

Output of the `skinManger` is *skinContactList*—a file which instead of raw data provides more detail about each activation:

- center of pressure (COP) - computed as weighted mean of activated taxels
- *taxelList* - a list of all activated taxels
- *skinPart* - index of the activated skin part (optional)
- *pressure* - average output of activated taxels

4.2 Reading data from the joints

The codes `safeMotion.py`, `yarpConnector.py` and `dataParse.py` mentioned below have been accessed in *code-nao-skin-control* gitlab repository [1].

The joints are being read through the `safeMotion.py` class which mainly serves to set the safe boundaries for robot's movement [14] and contains also a method for reading the joint coordinates. The robot's data is accessed through NAOqi framework. NAOqi is the main software running on the robot and NAOqi framework is used to program the robot.[5]

4.3 Skin and joint data unification

The data collection is performed by `yarpConnector.py` script, which communicates with `skinManager` to retrieve the data from skin—*skinContactList*—and uses `safeMotion.py` for reading the joint coordinates. As parameters it accepts the names of parts which will be touching/touched and should be read from. In [16] it is always two skin parts touching. In our case, as one of the chains touching is the fingertip, only one skin part is being read.

4.4 Parsing data for Matlab

Because the optimization framework is implemented in Matlab, the data retrieved by `yarpConnector.py` need to be parsed so Matlab script can read them. For parsing the data from self-touch with finger we used the `parseFinger` function from `dataParse.py` script. The output is a .mat file containing N cells—N is the number of activations in the dataset—where each cell contains a struct with IDs of activated taxels and an array with all the joint coordinates from the upper body.

4.5 Optimization in Matlab

For optimization, we used functions from the multirobot kinematic optimization framework from *code-calib-multirobot* gitlab repository [2]. The important functions used in this work are:

- `/Robots/Nao/loadNAO.m` - loading function which creates structure of the robot and sets the default parameters

- `/Configs/optimizationConfig.m` - sets the optimization parameters, such as algorithm used, function tolerance, maximum iterations, number of repetitions, etc.
- `/Robots/Nao/loadDatasetNao.m` - loads the datasets based on the optimization settings
- `/Utils/loadDHfromMat.m` - loads DH parameters from a .mat file, serves to set previously calibrated values as initial instead of the default parameters
- `/runCalibration.m` - the main function, prepares the robot structure and dataset, runs the calibration, evaluates and saves the results, using the functions above

4.5.1 Calibration parameters

The following parameters need to be set before running the calibration:

- robot_fcn - name of the loading function
- config_fcn - name of the function with optimization settings
- approaches - used approaches, in our case 'selftouch'
- chains - chains to be calibrated
- jointTypes - which parts should be calibrated — joint/mount/triangles/taxels (more parts at once can be chosen)
- dataset_fcn - name of the function for loading the datasets
- dataset_params - names of datasets used
- folder - where to save the results
- saveInfo - when set to 1, results are saved
- load DHfunc - name of the function to load DH
- loadDHfolder - folder with DH to load

The output of `runCalibration.m` includes updated DH parameters, performed corrections, individual and mean errors and info about the calibration settings used. The optimization framework also includes scripts for visualising the robot's configurations with activated taxels, plotting the deviations between contact points and other useful functions for presenting the calibration results.

Chapter 5

Results

This chapter presents the point clouds retrieved from 3D reconstruction, datasets collected by self-touch configurations, results of their use in the calibration and their comparison with previous calibrations.

5.1 3D Reconstruction datasets

In summary we collected three series of photos and turned them into three point clouds—these form the datasets from 3D reconstruction, they are further referred to as datasets 1, 2 and 3.

Links to all inter-step and final outputs provided in Section 3.3.4.

5.2 Outputs from 3D reconstruction

5.2.1 Results of scaling

As stated in Section 3.3.3, the scaling coefficient was calculated from distances of chosen neighbouring taxel pairs. For verification of the accuracy, we visualized the distance deviations, i.e. differences of each distance from the calculated mean distance—depicted in figure 5.1. The mean error was 0.12 mm and the maximum error was 0.5 mm.

5.2.2 Point clouds from scaled datasets

For the head and torso, all three datasets contained whole or almost whole skin part, so we had 3 measured positions for each taxel to work with, whereas for arms

there was always only part of the taxels—datasets 2 and 3 do not overlap at all and dataset 1 overlaps slightly with each of them (visible in Figure 5.4). This meant 1) we have only one measured position for most taxels, 2) some additional operations were needed for uniting the datasets into one coordinate system, including coordinate transformations to common points and afterwards to the target frame (with origin in taxel 3, as defined in Section 3.3.2). This task was performed by `hands_assemble.m` script.

After transformation (described in section 3.3.2) and scaling with the fixed coefficients the three datasets were compared. In Figures 5.2 and 5.3 one can see the distances between corresponding taxels in each dataset from their mean position. In case of head and torso most of the deviations are only tenths of millimeters. Also the visualisation of the point clouds (see figure 5.4 top) confirm that the difference between the three datasets is insignificant. Based on that we can claim this method provides very accurate mutual positions of the taxels and we can expect it will contribute greatly to the skin calibration.

In case of arms the differences are bigger, which is apparently caused by errors in transformations used for uniting the datasets (as described above). However the maximum difference—2.7 mm—is still smaller than the error achieved in previous calibration of NAO skin [16].

For the purposes of calibration one final point cloud was made by calculating the mean from the three datasets. The results have been added to *Robots/Nao/Dataset/Points* as .mat files, containing 3D translations to the taxels.

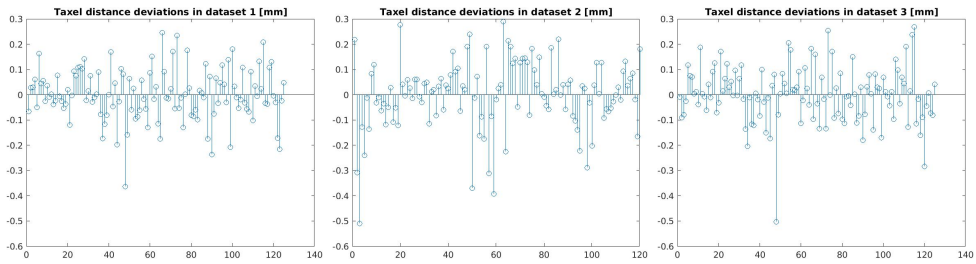


Figure 5.1: Deviations of real distances (6.50 mm) vs. reconstructed distances between neighbouring taxels in datasets 1-3.

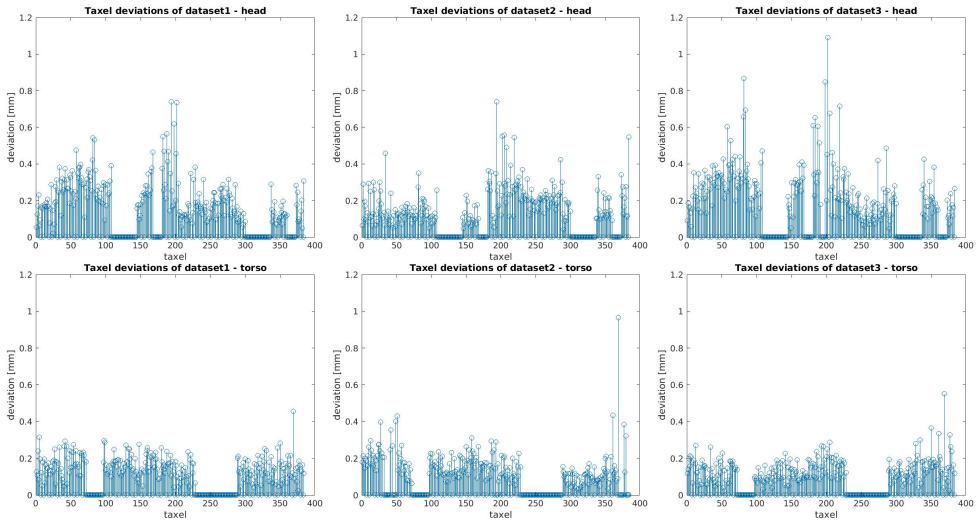


Figure 5.2: Distances of each taxel from head and torso in datasets 1-3 from their mean position (where the value is 0, the taxel is not included in the skin part).

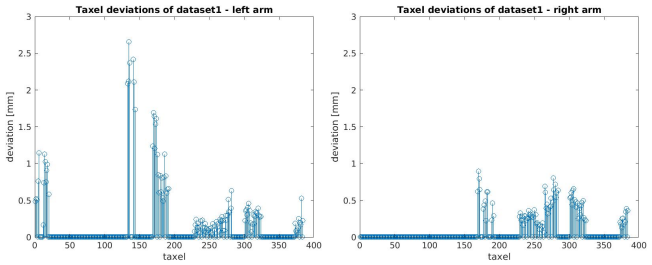
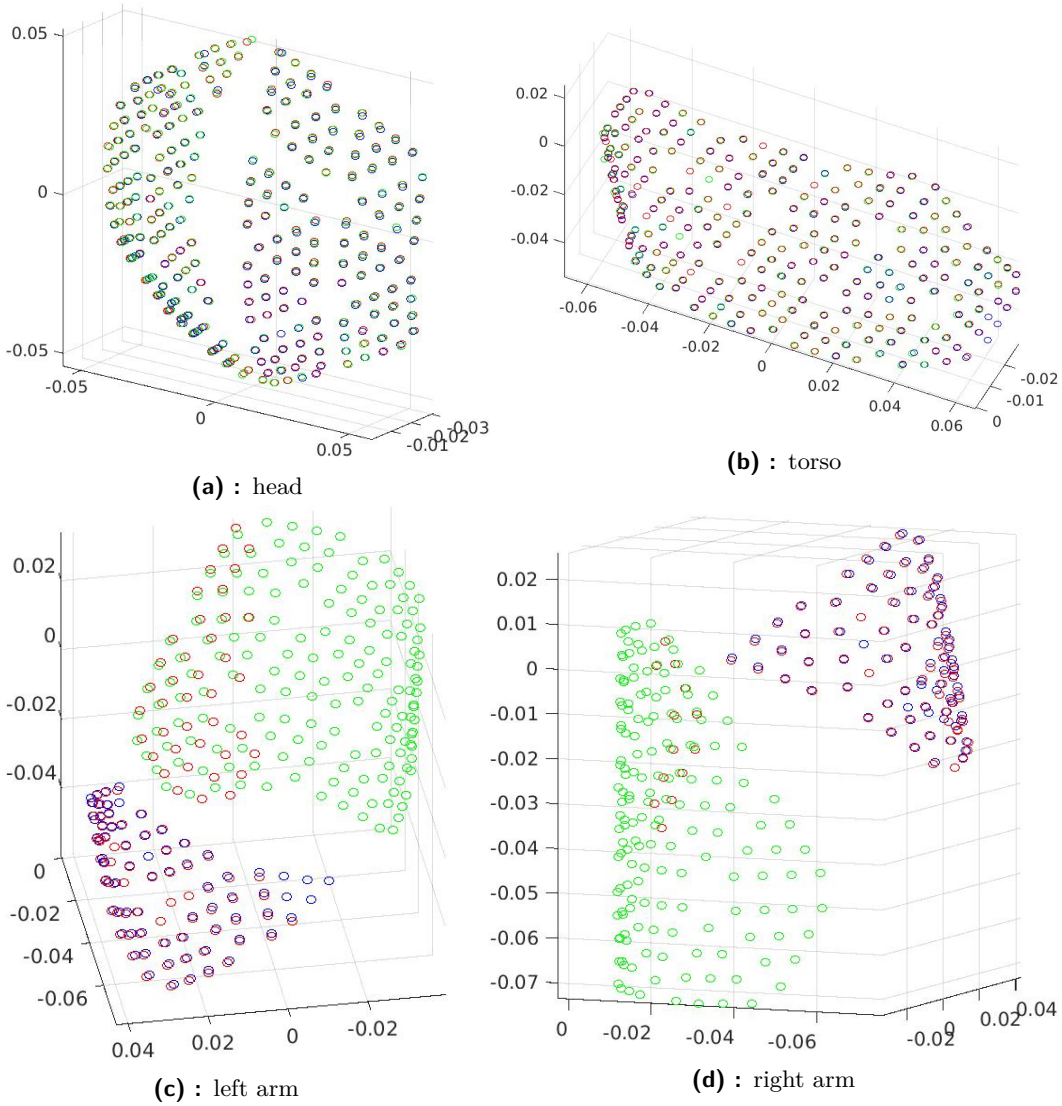


Figure 5.3: Distances of taxels in dataset 1 common with dataset 2 or 3 from the computed mean, for left arm (left) and right arm (right).

5.2.3 Possible errors and uncertainties

There was a slight inaccuracy when picking the points (described in section 3.3.1)—this operation was done by hand, therefore sometimes the picked taxel center is not exactly the center. But considering the small size of a taxel and high resolution of the textured model this inaccuracy is mostly just tenths of millimeters. There are a few isolated spaces where the error is slightly bigger (close to 1 mm) which is caused by slightly damaged structure of the model in that area.



(a) : head

(b) : torso

(c) : left arm

(d) : right arm

Figure 5.4: Visualisation of the scaled point clouds. Dataset 1 in red, dataset 2 in green, dataset 3 in blue (when too close the colors get mixed).

5.3 Self-touch datasets

Using self-touch configurations the following datasets were collected:

finger and part	dataset size
right finger - torso	1818
right finger - left arm	1814
left finger - torso	825
left finger - head	893
left finger - right arm	908

Table 5.1: Collected datasets and their sizes (i.e. number of activations).

The .mat files with the datasets are saved in */Robots/Nao/Dataset/Datasets* folder in *code-calib-multirobot* directory [2].

Apart from these datasets we used the datasets provided by Lukáš Rustler in previous work [16] with configurations of two skin parts touching:

parts touching	dataset size
right arm - torso	1072
left arm - torso	810
right arm -head	670
left arm - head	599

Table 5.2: Datasets provided by [16].

5.4 Calibration using self-touch method only

First we needed to calibrate the position of the fingertip on both hands. As its reference frame is located inside the arm, it was impossible to obtain its relative position by measurements. The estimated position was therefore very inaccurate (uncertainty could be a few centimeters).

As initial values we used the model calibrated by self-touch configurations provided by [16]. From all the parts torso's parameters showed up to be the most accurate, so it can be assumed the best dataset for calibrating the finger was the one with torso activations. Another idea was to calibrate the fingers by combined datasets from two parts. We tried combining torso and the other arm as the arms had a bigger and more reliable dataset than the head.

Afterwards the calibrated fingers were used as reference for calibrating the skin parts. The chosen process was to first calibrate the mount and patches and then calibrate the individual triangles.

The calibrations performed and their results are stated below. The errors depicted further in the graphs and mean errors stated in the tables are in accordance with Equations 2.3 and 2.4 calculated as:

$$e = |\boldsymbol{p}_m^t - \boldsymbol{p}_m^f| \tag{5.1}$$

where $\boldsymbol{p}_m^t$ is the estimated position of the touched taxel and $\boldsymbol{p}_m^f$ is the estimated position of the fingertip, both calculated from joint coordinates using forward kinematics.

The error graphs depict the error for each taxel in the dataset, distributed by triangles they belong to.

5.4.1 Finger calibration

Right finger calibration

datasets	e_t[mm]	$e_{rA}[mm]$	e_A[mm]
torso	7.7	18.7	13.5
torso and left Arm	11.0	17.0	14.0

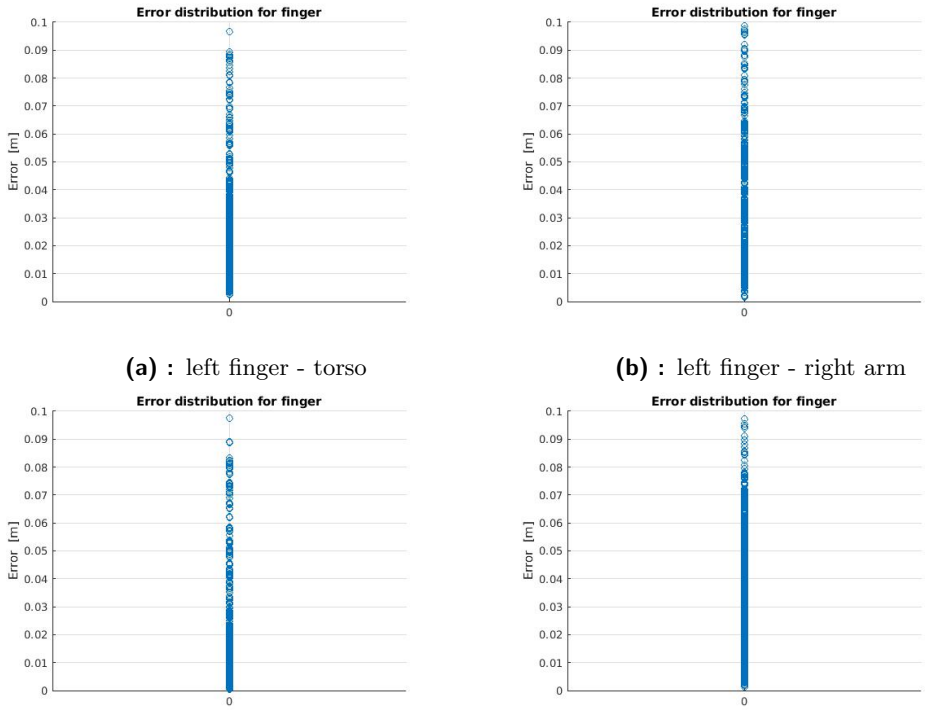
Table 5.3: Results of right finger calibration. e_t...mean error on torso dataset, e_{rA}...mean error on left arm dataset, e_A...mean error on both datasets

Left finger calibration

datasets	e_t[mm]	e_{lA}[mm]	e_A[mm]
torso	12.1	22.0	17.2
torso and left Arm	16.6	15.0	15.9

Table 5.4: Results of left finger calibration. e_t...mean error on torso dataset, e_{lA}...mean error on right arm dataset, e_A...mean error on both datasets

Error distribution for calibrated fingers



(a) : left finger - torso

(b) : left finger - right arm

(c) : right finger - torso

(d) : right finger - left arm

Figure 5.5: Distribution of distances between paired taxels for finger calibration.

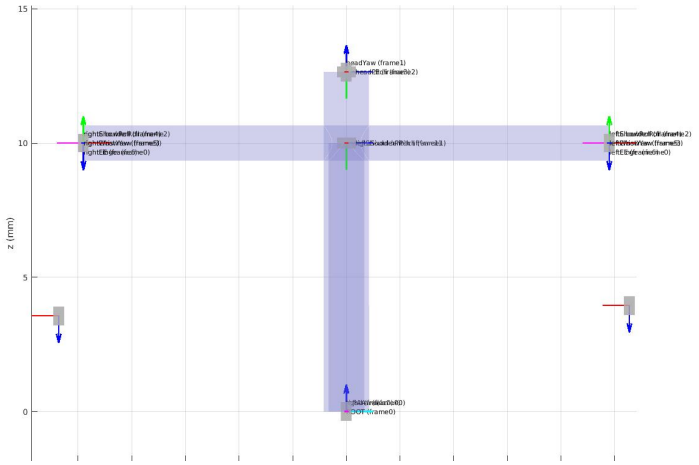


Figure 5.6: Visualization of calibrated fingers.

5.4.2 Skin parts calibration

Torso from right finger

Chains optimized	e_{rF} [mm]
mount and patches	6.7
triangles	6.3

Table 5.5: Results of torso calibration from right finger. e_{rF}...mean error on right finger dataset.

Torso from both fingers

Chains optimized	e_{rF}	e_{lF} [mm]
mount and patches	7.2	12.3
triangles	6.4	11.1

Table 5.6: Results of torso calibration from both fingers. e_{rF}...mean error on right finger dataset, e_{lF}...mean error on left finger dataset.

Left arm from right finger

Chains optimized	e_{rF} [mm]
mount and patches	11.7
triangles	10.3

...mean error on right finger dataset.

Table 5.7: Results of left arm calibration from right finger. e_{rF}

Right arm from left finger

Chains optimized	e_{lF} [mm]
mount and patches	10.5
triangles	9.4

...mean error on left finger dataset.

Table 5.8: Results of right arm calibration from left finger. e_{lF}

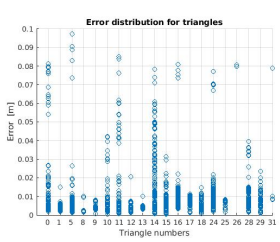
5.4.3 Summary

Resulting parameters of fingers vs. initial estimated parameters

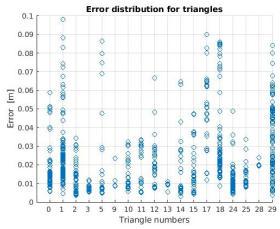
finger	x	y	z	x_0	y_0	z_0
left finger	-0.0074	-0.0860	0.0604	0	-0.06	0.06
right finger	0.0092	-0.0967	0.0643	0	-0.06	0.06

Table 5.9: Calibrated parameters of the fingers (x, y, z) in comparation with estimated parameters (x_0, y_0, z_0)

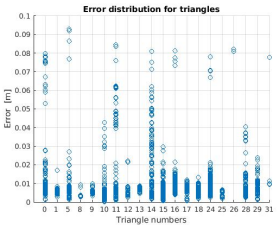
Error distribution for calibrated parts from the fingers



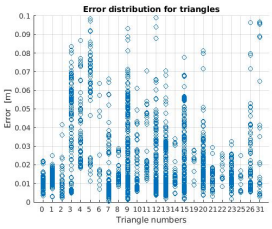
(a) : torso - right finger



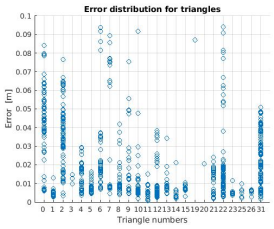
(b) : torso - both fingers, error on left finger



(c) : torso - both fingers, error on right finger



(d) : left arm - right finger



(e) : right arm - left finger

Figure 5.7: Error distribution for calibrated parts from the fingers.

5.5 Calibration using self-touch and data from 3D reconstruction

As the triangle and taxel positions are very well determined by the point clouds from 3D reconstruction (as stated in Section 5.2.2), there was no need to calibrate them, so the only calibrated chains were the mounts—more precisely we were calibrating the transformation from the the last joint's frame to the frame of the taxel point cloud. In case of left arm we added also the patches to the calibration, as the uncertainty from assembling the datasets was bigger (see Figure 5.3) and could be fixed by the calibration.

In previous calibrations performed by [16] the torso showed up to be the easiest part to calibrate and the most reliable source for the calibration of other parts, therefore we chose to calibrate torso's mount first simultanously with the right arm. As its initial position was right in the y axis and α and γ angles, we optimized only x and z position and β angle. Afterwards the fingers were calibrated from the torso. Then the arms and finally the head. To test the reliability of the calibrated fingers we chose to try calibrating the arms from the torso only and then from the torso and finger and compare the results. For the head we tried the calibration from arms only and from arms and left finger. The details of all performed calibrations and their results are stated below.

With each calibration all the settings used are stated. Meaning of the parameters explained in Section 4.5.1. The errors are depicted the same way as in Section 5.4.

Common settings for all calibrations:

- `robot_fcn: loadNAO`
- `config_fcn: optimizationConfig`
- `approaches: selftouch`
- `dataset_fcn: loadDatasetNao`
- `loadDHfunc: loadDHfromMat`
- `loadDHargs: {}`

5.5.1 Torso and right arm simultaneously

Settings:

- `chains: torso, rightArm`
- `jointTypes: mount`
- `dataset_params: rightArm_torso`
- `folder: 3Drec_torso`
- `loadDHfolder: ' '`

Results

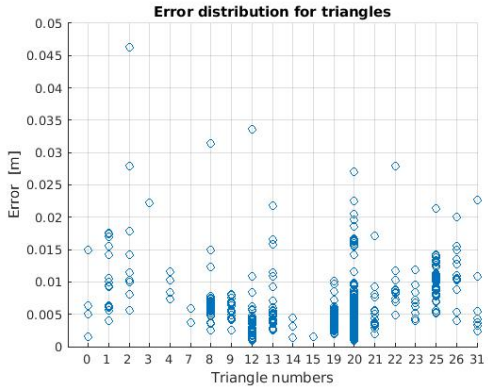


Figure 5.8: Distribution of distances between paired taxels. – **torso and right arm**

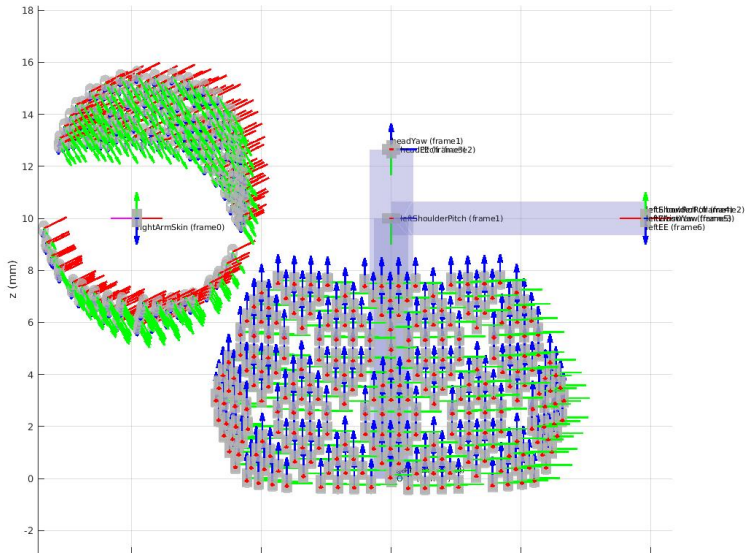


Figure 5.9: Visualization of the calibrated torso and right arm.

5.5.2 Fingers from torso

Settings:

- `chains: rightFinger`
- `jointTypes: joint`
- `dataset_params: torso_rightFinger`
- `folder: 3Drec_rF`
- `loadDHfolder: 3Drec_torso`

- `chains: leftFinger`
- `jointTypes: joint`
- `dataset_params: torso_leftFinger`
- `folder: 3Drec_lF`
- `loadDHfolder: 3Drec_rF`

Results

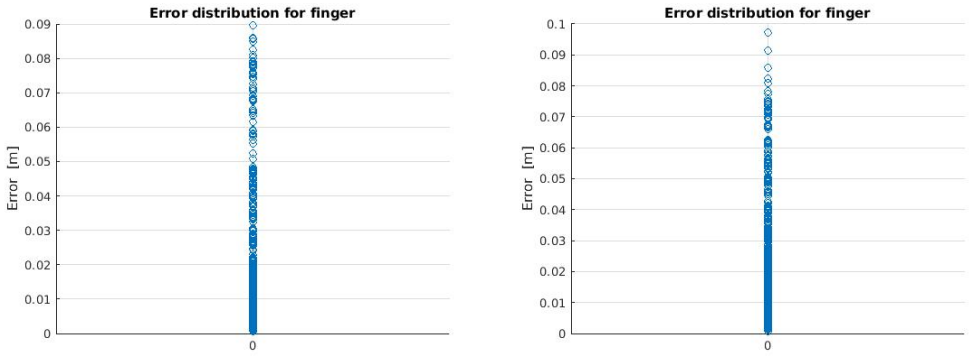


Figure 5.10: Distribution of distances between paired taxels. – **left finger and torso, right finger and torso**

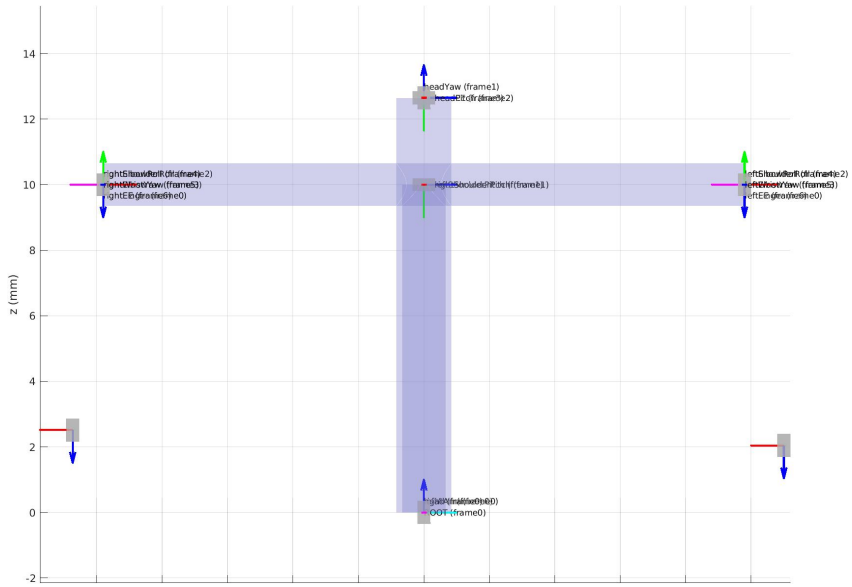


Figure 5.11: Visualization of calibrated fingers.

5.5.3 Right arm from torso

Settings:

- `chains: rightArm`
- `jointTypes: mount`
- `dataset_params: rightArm_torso`
- `folder: 3Drec_rA`
- `loadDHfolder: 3Drec_lF`

Results

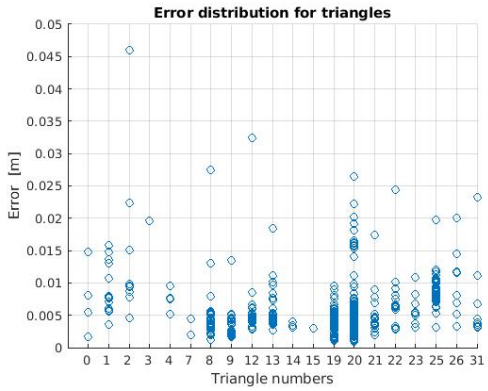


Figure 5.12: Distribution of distances between paired taxels. – **right arm and torso**

5.5.4 Right arm from torso and left finger

Settings:

- `chains: rightArm`
- `jointTypes: mount`
- `dataset_params: rightArm_torso, rightArm_leftFinger`
- `folder: 3Drec_rA_wlF`
- `loadDHfolder: 3Drec_rA`

Results

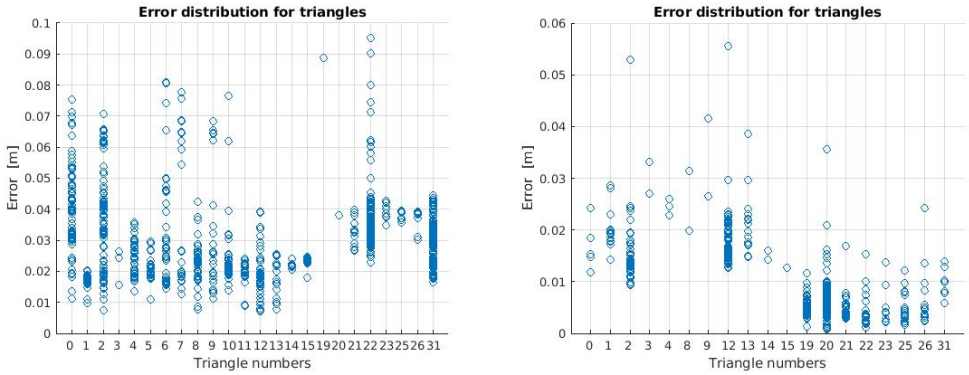


Figure 5.13: Distribution of distances between paired taxels. – **right arm and torso, right arm and left finger**

5.5.5 Left arm from torso

Settings:

- `chains: leftArm`
- `jointTypes: mount, patch`
- `dataset_params: leftArm_torso`
- `folder: 3Drec_lA`
- `loadDHfolder: 3Drec_rA`

Results

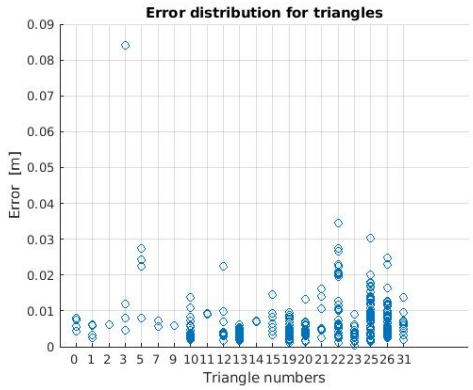


Figure 5.14: Distribution of distances between paired taxels. – **left arm and torso**

5.5.6 Left arm from torso and right finger

Settings:

- `chains: leftArm`
- `jointTypes: mount, patch`
- `dataset_params: leftArm_torso, leftArm_rightFinger`
- `folder: 3Drec_lA_wrF`
- `loadDHfolder: 3Drec_rA_wlF`

Results

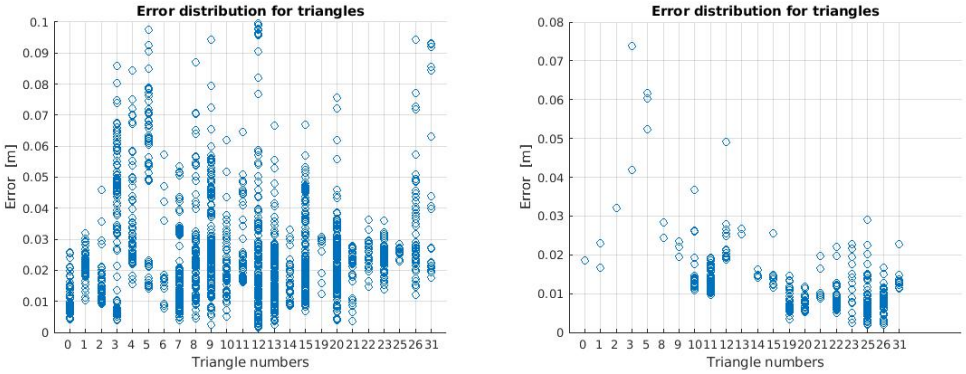


Figure 5.15: Distribution of distances between paired taxels. – **left arm and torso, left arm and right finger**

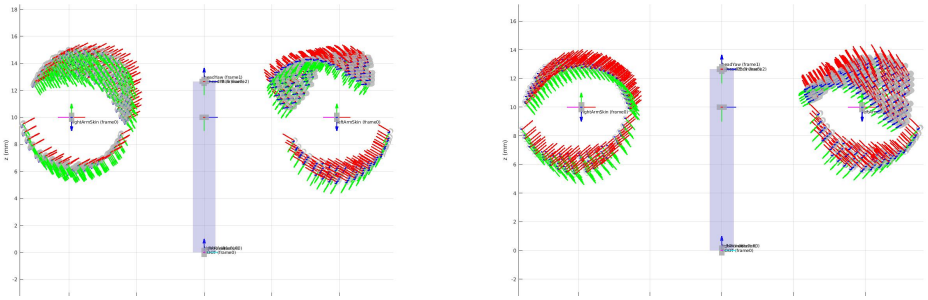


Figure 5.16: Visualization of arms calibrated from torso only (left) and from torso and finger (right).

5.5.7 Head from both arms

Settings:

- `chains: head`
- `jointTypes: mount`
- `dataset_params: rightArm_head, leftArm_head`
- `folder: 3Drec_head`
- `loadDHfolder: 3Drec_lA_wrF`

Results

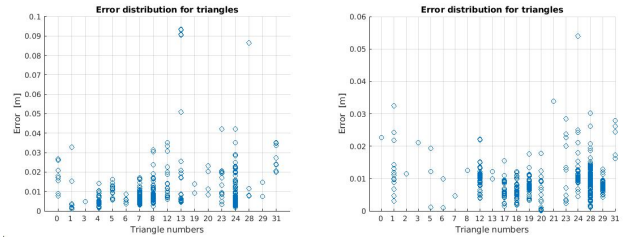


Figure 5.17: Distribution of distances between paired taxels. – **left arm and head, right arm and head**

5.5.8 Head from both arms and left finger

Settings:

- `chains:`
- `jointTypes:`
- `dataset_params:`
- `folder:`
- `loadDHfolder:`

Results

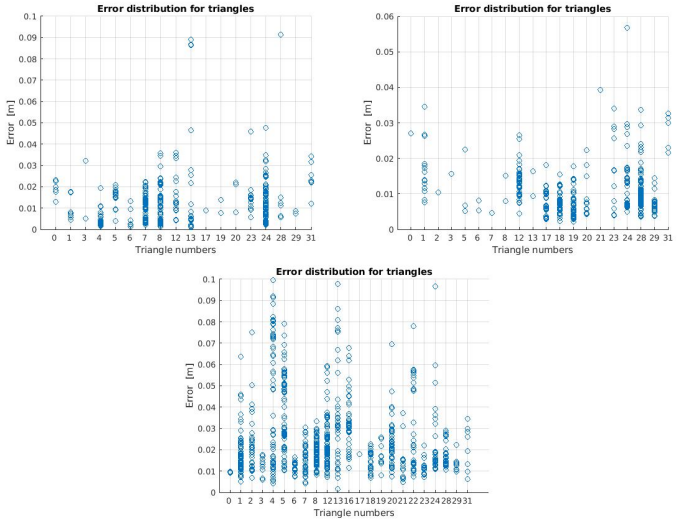


Figure 5.18: Distribution of distances between paired taxels. – **left arm and head, right arm and head, left finger and head**

Figure 5.19: Visualization of head calibrated from arms only (left) and from arms and left finger (right).

5.5.9 Summary

The following table shows the calibrated parameters of all optimized chains and their mean error in the used dataset.

calibed part	link	x [m]	y [m]	z [m]	α [rad]	β [rad]	γ [rad]	e [mm]
torso	mount	0.0532	0	0.0503	0	0.1430	0	10.0
right finger	joint	0.0093	-0.0892	0.0749	-	-	-	11.0
left finger	joint	-0.0123	-0.0901	0.0791	-	-	-	19.2
right arm 1*	mount	-0.0305	-0.0202	-0.0171	-0.3170	1.6118	2.0885	8.6
right arm 2*	mount	-0.0208	-0.0499	-0.0226	-0.6320	1.8892	1.7480	18.9
left arm 1*	mount	0.0317	-0.0569	-0.0255	1.3983	0.6037	-0.3249	7.8
	patch 1	0.0050	-0.0024	-0.0019	-0.1378	0.1723	-0.0170	-
	patch 2	-0.0755	0.0320	-0.0145	0.0276	-0.1027	0.0894	-
left arm 2*	mount	0.0221	-0.0846	-0.0235	1.1929	0.8124	-0.4777	15.5
	patch 1	-0.0009	0.0055	-0.0037	-0.1278	0.0164	-0.1089	-
	patch 2	-0.0674	0.0238	-0.0165	0.1473	-0.0598	0.1050	-
head 1*	mount	0.0692	-0.0499	0.0025	1.5708	0	0	10.5
head 2*	mount	0.0678	-0.0458	-0.0014	1.5708	0	0	14.3

Table 5.10: Final position parameters $x, y, z, \alpha, \beta, \gamma$ of calibrated parts and mean error in their used dataset e. * 1 indicates calibration without finger, 2 with finger.

Chapter 6

Discussion, conclusion and future work

We developed a system to retrieve positions of tactile sensors on a robot from a series of photographs. Previously, the positions of individual triangles and taxels in the local frames were only estimated in the beginning and needed to be calibrated as well. The 3D reconstruction provided mutual positions of individual taxels so accurately (error $< 0.5\,mm$) that their layout can be kept and only mounts need to be calibrated. This reduces the number of calibrated parameters to $6 \cdot 1$ instead of $6 \cdot 375$ (1 mount, 2 patches, 32 triangles, 320 taxels).

The uncertainty of taxel position was slightly bigger in case of the arms because of the necessity of several transformations to unite the retrieved point clouds. This could be fixed in the future by collecting more point clouds for the arms, for example a series of photos capturing the arm only all around its perimeter. Then the assembling part would be skipped and the taxel positions as accurate as for torso and head.

A small error in the scaling could occur as the scaling coefficient was calculated from the mean of taxel distances. A better approach in the future could be adding the scaling coefficient as a parameter in the calibration and have it optimized.

We also tried to improve the self-touch calibration by adding a single-point end effector—finger—on each arm to achieve better accuracy in the datasets. The number of activated taxels was much smaller than in the previous work [16] (mostly 1-3 taxels). However, the errors in calibration using the finger are considerably bigger than the errors without using the finger—in case of arms twice as big. The errors might be caused by bad activations when collecting the dataset—the arms are not easy to manipulate in a way to prevent some accidental touches and finger slipping on the skin surface. However the visualization of calibrated arms speaks in favor of using the finger—notice Figure 5.19, the right arm is clearly better placed after calibration using left finger. In case of head, the resulting position almost does not change when adding the dataset from left finger to the datasets from both arms. Apparently, despite the enormous errors on the datasets, the fingers got quite well calibrated after all.

The left arm and left finger calibration had always worse results than right arm and right finger. This could be because of the smaller datasets size. Collecting more data for left arm and left finger should be tried.

Bibliography

[1] Project repository. `https://gitlab.fel.cvut.cz/body-schema/code-nao-skin-control/tree/master`. [Online; accessed August-2020].

[2] Project repository. `https://gitlab.fel.cvut.cz/rustlluk/code-calib-multirobot`. [Online; accessed August-2020].

[3] Alessandro Albini, Simone Denei, and Giorgio Cannata. Towards autonomous robotic skin spatial calibration: A framework based on vision and self-touch. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 153–159. IEEE, 2017.

[4] Aldebaran. Nao h25 documenttation. `http://doc.aldebaran.com/2-1/family/nao_h25/index_h25.html`. [Online; accessed May-2020].

[5] Aldebaran. Naoqi description. `http://doc.aldebaran.com/1-14/dev/naoqi/index.html`. [Online; accessed August-2020].

[6] Meshroom Contributors. Meshroom manual. `https://meshroom-manual.readthedocs.io/en/latest/index.html`. [Online; accessed July-2020].

[7] A. Del Prete, S. Denei, L. Natale, Fulvio M., F. Nori, G. Cannata, and G. Metta. Skin spatial calibration using force/torque measurements. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 3694 –3700, 2011.

[8] A. Del Prete, A. Schmitz, and F. Giovannini. skinmanager description. `http://www.icub.org/doc/icub-main/group__icub__skinManager.html`. [Online; accessed August-2020].

[9] Jacques Denavit and Richard Scheunemann Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, 77(2):215–221, June 1955.

[10] P. Maiolino, M. Maggiali, G. Cannata, G. Metta, and L. Natale. A flexible and robust large scale capacitive tactile system for robots. *Sensors Journal*, 13(10):3910–3917, 2013.

[11] MathWorks. Matlab documentaion. `https://www.mathworks.com/help/optim/ug/lsqnonlin.html`. [Online; accessed August-2020].

[12] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. YARP: Yet Another Robot Platform. *International Journal of Advanced Robotic Systems*, 3(1):8, 2006.

[13] P. Mittendorfer and G. Cheng. 3D surface reconstruction for robotic body parts with artificial skins. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2012.

[14] Adam Rojik. Joint constraints for naoprague. `https://docs.google.com/document/d/14eYPeTlPOEelmroKRqpS_ajDNbR8v7G-dnKC0xnRGJ0/edit#heading=h.vldrhxxpsuhd`.

[15] A. Roncone, M. Hoffmann, U. Pattacini, and G. Metta. Automatic kinematic chain calibration using artificial skin: self-touch in the icub humanoid robot. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2305–2312, 2014.

[16] Lukáš Rustler. Artificial skin calibration for the Nao humanoid robot using "self-touch". Master's thesis, Faculty of Electrical Engineering, Czech Technical University in Prague, 2019.

[17] K. Stepanova, T. Pajdla, and M. Hoffmann. Robot self-calibration using multiple kinematic chains – a simulation study on the iCub humanoid robot. *Robotics and Automation Letters*, 4(2):1900–1907.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Potočná Bohumila** Personal ID number: **465812**

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

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Artificial Skin Calibration for a Humanoid Robot: Comparing or Combining “Self-Touch” and 3D Reconstruction from Images

Bachelor’s thesis title in Czech:

Kalibrace robotické kůže humanoidního robota: porovnání a kombinace “sebedotykových” konfigurací a 3D rekonstrukce z fotografií

Guidelines:

1. Skin calibration using 3D reconstruction:
a. Collect images of Nao robot with exposed skin in different configurations and under different light conditions, possibly with projecting patterns onto the robot.
b. Reconstruct robot 3D shapes using software for 3D reconstruction (ColMap, Capturing Reality) and explore how to retrieve coordinates of individual tactile elements.
c. Combine obtained results with other information about electronic skin (dimensions of skin in 2D) and existing calibrations (Rustler 2019). For example, parametrize the 3D point cloud as splines or meshes and use it to constrain skin spatial calibration.
2. Skin calibration using self-touch configurations:
a. Data collection on Nao robot with artificial skin in self-touch configurations (joint angles and skin activations). Follow up on Rustler (2019) but with the addition of custom end effectors to achieve point-like contacts.
b.Skin spatial calibration using hierarchical and modular optimization framework (Rozlivek and Rustler 2019) allowing to calibrate different parameters (skin part pose, skin triangle pose, individual taxel poses, DH parameters) using non-linear least squares methods (Matlab environment).
3. If time permits: evaluation of skin calibration using 3D reconstruction, self-touch, and their combination. w.r.t. different datasets, parameterizations, number of unknowns, prior knowledge about 2D skin structure and 3D robot structure.

Bibliography / sources:

[1] Albini, A., Denei, S., & Cannata, G. (2017, September). Towards autonomous robotic skin spatial calibration: A framework based on vision and self-touch. In Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on (pp. 153-159). IEEE.
[2] Del Prete, A., Denei, S., Natale, L., Mastrogiovanni, F., Nori, F., Cannata, G., & Metta, G. (2011, September). Skin spatial calibration using force/torque measurements. In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on (pp. 3694-3700). IEEE.
[3] Maiolino, P.; Maggiali, M.; Cannata, G.; Metta, G. & Natale, L. (2013), 'A flexible and robust large scale capacitive tactile system for robots', Sensors Journal, IEEE 13(10), 3910--3917.
[4] Mittendorfer, P. & Cheng, G. (2012), 3D surface reconstruction for robotic body parts with artificial skins, in 'Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)'.
[5] Roncone, A.; Hoffmann, M.; Pattacini, U. & Metta, G. (2014), Automatic kinematic chain calibration using artificial skin: self-touch in the iCub humanoid robot, in 'Robotics and Automation (ICRA), 2014 IEEE International Conference on', pp. 2305-2312.
[6] Rustler, L. (2019), 'Artificial Skin Calibration for the Nao Humanoid Robot Using “Self-touch”', Bachelor’s thesis, Faculty of Electrical Engineering, Czech Technical University in Prague.
[7] Stepanova, K.; Pajdla, T. & Hoffmann, M. (2019), 'Robot self-calibration using multiple kinematic chains – a simulation study on the iCub humanoid robot', IEEE Robotics and Automation Letters 4(2), 1900-1907.

Name and workplace of bachelor's thesis supervisor:

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

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

Date of bachelor's thesis assignment: **07.01.2020** Deadline for bachelor thesis submission: **14.08.2020**

Assignment valid until: **30.09.2021**

Mgr. Matěj Hoffmann, Ph.D. Supervisor's signature	doc. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

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

.	
Date of assignment receipt	Student's signature

