Pavan Vishwanath

Simultaneous localization and mapping for camera-based EEG electrode digitalization

Institute of Medical Technology and Intelligent Systems
Building E | 21073 Hamburg
www.tuhh.de/mtec

A master thesis written at the Institute of Medical Technology and Intelligent Systems and submitted in partial fulfillment of the requirements for the degree Master of Science.

Author: Pavan Vishwanath

Title: Simultaneous localization and mapping for camera-based EEG electrode digitalization

Date: May 26, 2021

Supervisors: Martin Gromniak (MSc.)
Alexander Schlaefer (Dr.-Ing.)

Referees: Prof. Dr.-Ing. Alexander Schlaefer
Prof. Dr.-Ing. Rolf-Rainer Grigat

Contents

List of Figures

List of Tables

Abstract

Electroencephalography (EEG) technology is widely used in clinical medicine to monitor the electrical activity of the brain. EEG captures the brain signals with the help of electrodes attached to the scalp. A crucial step in determining the exact source of the brain signal is the accurate estimation of the electrodes position on the scalp known as electrode digitalization. In this thesis work, an inexpensive RGB-D (color+TOF) camera such as Azure Kinect by Microsoft based electrode digitalization system is developed and evaluated on 3 different caps, one with 63 and two with 23 electrodes on them. This system employs a machine learning based electrode detection framework, pose-graph based simultaneous localization and mapping, and various cluster processing algorithms for electrode digitalization. These methods are first applied to the data-set created by guiding the camera around the phantom head using a robot where the camera motion is precisely controlled then on the free hand guided trajectories. For the robot-guided case, average digitalization errors (post registration) of 3.7 mm $\pm$ 0.3 for the cap with 63 electrodes, 6.8 mm $\pm$ 2.0 mm and 5.6 mm $\pm$ 1.2 mm for two caps with 23 electrodes were achieved respectively. For the hand-guided case, 10 mm $\pm$ 2.7 mm for the cap with 63 electrodes, 14.6 mm $\pm$ 3.4 mm and 19.3 mm$\pm$ 2.3 mm, for caps with 23 electrodes respectively. These methods resulted in lower digitalization accuracy in comparison with widely used electromagnetic based method.

1 Introduction

Electroencephalography(EEG) is an electrophysiological monitoring method to record the electrical activity of the brain. EEG measures voltage fluctuations resulting from ionic current within the neurons of the brain with the help of electrodes attached to the scalp [1]. Due to its non-invasive nature, economy, and ease of use, EEG technology is now widely used in clinical medicine to diagnose epilepsy, coma, brain death, etc.[2]. In order to determine the exact source of the brain signal, the accurate 3D position of the electrodes has to be known. Currently available EEG electrode digitalization methods are broadly classified as (1) manual methods [3] [4], (2) electromagnetic digitalization [5] [6], (3) ultrasound digitalization [7], (4) magnetic resonance imaging (MRI) assisted methods [8], (5) photogrammetric methods [9] [10] [11] [12]. The manual methods are time, labor intensive and prone to errors. The electromagnetic, Ultrasound, and MRI assisted methods are prone to single point measurement error, which demands multiple repetitions of the process. These methods also suffer from electrical signal interference with EEG signals and often require precise control of the measurement environment. Photogrammetric methods on the other hand encounter multiple camera calibration challenges, manual electrode detection errors, etc. Recent developments in the areas of high precision time of flight (TOF) depth cameras have led to the possibility of combining a high resolution industrial camera and a TOF depth camera for the EEG electrode digitalization. One such system uses an expensive CCD and depth (MESA-SR-4000) cameras for digitalization [13]. This system captures images from five different perspectives (five fixed camera positions) then RGB and depth information is fused together to localize the EEG electrodes. The tab. 1.1 gives an overview of typical digitalization methods.

A variety of odometry based positioning techniques are available in the field of robotics. The techniques which rely on the 3D point clouds employ scan registration algorithms that find the best transformation that aligns 2 subsequent point clouds (source and target). Especially in the era of autonomous driving, these algorithms serve as the basis for localizing the car and simultaneously build the map of the environment, and also to estimating the car's trajectory by fusing the data from a variety of sensor suites (sensor fusion). The most commonly used scan registration technique is iterative closest point (ICP) algorithm [14]. While using ICP, similar to any odometry system, the inherent accumulation of error at each scan registration leads to the drift in the trajectory estimation. The simultaneous localization and mapping (SLAM) technique is

Tab. 1.1: Comparison of typical methods from [13]

Method	Principal	Equip size	Time	Accuracy	Reliability	Typical Ref.
Manual measurement	Coordinate measuring, calipers	Small	very slow	0.4mm	Mid	De Munck et al. (1991)
Camera matrix	Stereo vision	Large	real-time(<0.1s)	1.27mm	Bad	De Koessler et al. (2007)
Positioning tool	Electro magnetic digitizer	Small	5 min	2-8 mm	Mid	Datal et al. (2014)
Photogrammetry	Structure from motion	Small	Slow (5-10) min	0.8 mm	Mid	Clausner et al. (2017)
Laser scanner	Laser	Small	Slow	0.05-0.2 mm	Good	Jeon et al. (2018)
Color + depth	color+TOF	Small	Real time	0.3-3.3 mm	Good	Chen S et al. (2019)

usually the go to solution in order to minimize the drift over time and to obtain the best estimate of the trajectory. SLAM in its probabilistic form is solved using pose-graph based formulation by constructing a graph whose nodes represent the robot pose and the edge between 2 nodes act as noisy odometry constraints and loop closure constraints can be added if the robot revisits previously visited places. A solution to such a graph based SLAM involves solving a large non-linear optimization problem [15].

In this thesis work, an inexpensive RGBD (color+TOF) camera such as the Azure Kinect by Microsoft based electrode digitalization system is developed and evaluated. This system employs a machine learning based electrode detection framework, pose-graph based simultaneous localization and mapping, various cluster processing algorithms for electrode digitalization. These methods are first applied to the data-set created by guiding the camera around the phantom head using a robot where the camera motion is precisely controlled. The second part focuses on free hand-guided trajectories. The objective of this thesis work is to develop algorithms to accurately estimate the 3D position of the EEG electrodes by simultaneously localizing the camera relative to the head and building an EEG electrode map based on scan registration and the pose-graph formulation.

This thesis is structured in the following way. In chapter 2, the required background is provided for various important topics that are necessary to the thesis. For example, a brief introduction to the camera fundamentals including calibration, pose estimation, etc. is first presented. We then present the core algorithms used in this thesis such as scan registration, fundamentals of SLAM, especially in pose-graph formulation, electrode detection, and cluster processing algorithms. In chapter 3, we discuss the algorithmic workflow for electrode digitalization along with various open source libraries used in this thesis. We then outline the experimental setup and different apparatus used. Furthermore, we will elaborate on robot and hand-guided data acquisition systems and various calibration processes involved along with the challenges encountered. A detailed summary of pose-graph construction, optimization, and loop closure are included. We close the section by discussing performance metrics used to evaluate the algorithms. Chapter 4 is dedicated to the results of various calibrations and electrode digitalization. In chapter 5, we discuss few important aspects of the results. The final conclusion and future scope are presented in chapter 6.

2 Background

2.1 Camera fundamentals

A gentle introduction to the pinhole camera model, projection matrix, and calibration (internal and external) will be presented in this section. The underlying mathematics and equations are referenced from [16], [17].

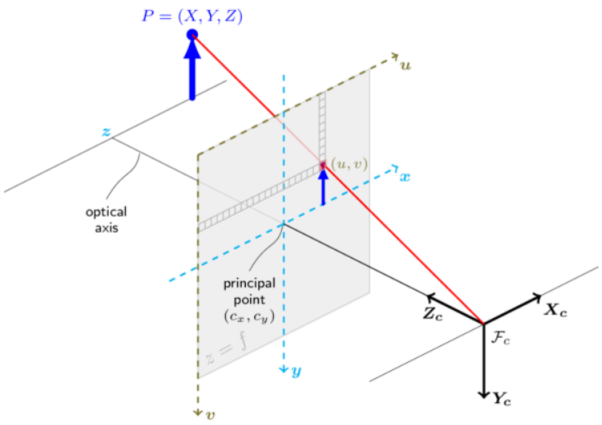

Fig. 2.1: Pinhole camera geometry. $\boldsymbol{F_c}$ *is the camera center,* (C_x, C_y) *is the principal point,* $\boldsymbol{P}$ $(X, Y, Z)^T$ *is world point,* $\boldsymbol{P_c}$ $(X_c, Y_c, Z_c)^T$ *is world point measured in camera coordinate system.[16]*

A simplified perspective projection is shown in fig. 2.1. A world point $(X_c, Y_c, Z_c)^T$ is mapped to $(f\frac{X_c}{Z_c}, f\frac{Y_c}{Z_c}, f)^T$ on the image plane placed at a focal length distance f from the camera center (F_c). Two key facts can be derived from the above projection equation that farther the object is (larger the Z_c) from the camera smaller the size in image plane (shrinking operation), these shrunk points are magnified by the focal length f to be placed on the image plane. World points are first shrunk $(x, y)^T = (\frac{X_c}{Z_c}, \frac{Y_c}{Z_c})^T$ and then magnified $(f\frac{X_c}{Z_c}, f\frac{Y_c}{Z_c}, f)^T$.

Often, the world points are represented in homogeneous coordinates due to several advantages, the ability to express infinite quantities is one of them. The linear mapping between world and image points is evident in the homogeneous representation as shown below.

$$Z_c \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{2.1}$$

The linear mapping can be expressed in a compact form $\boldsymbol{x} = \mathtt{P}\boldsymbol{X_c}$ where $\boldsymbol{x}$ is a vector

of image points, $\boldsymbol{X}$ is a vector of world points and P is a 3x4 homogeneous matrix called camera projection matrix. In general, the origin of the image plane may not coincide with the principal point, therefore it is necessary to map the projected points to pixels before using the image for further use as show in fig. 2.2.

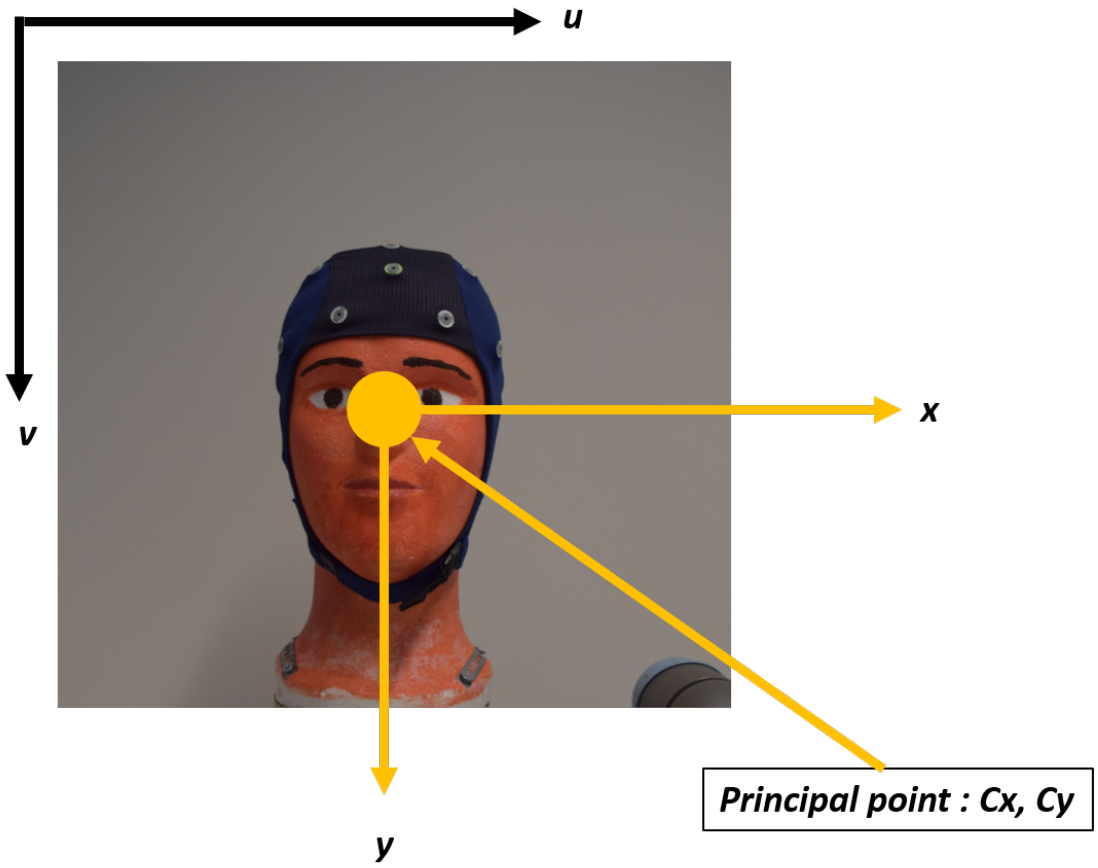

Fig. 2.2: Pixel coordinates (u, v) and the camera coordinates (x, y).

The equation 2.2 depicts mapping image to pixel coordinates. where (S_x) is size of pixel width and (S_y) is size of pixel height.

$$\begin{aligned}(u - C_x) &= \frac{x}{S_x} \\ (v - C_y) &= \frac{y}{S_y}\end{aligned} \tag{2.2}$$

Therefore a 3D world point $(X_c, Y_c, Z_C)^T$ is mapped to $(f\frac{X}{Z} + C_x, f\frac{Y}{Z} + C_y, f)^T$ to the pixel coordinates (u, v, z) and can be expressed as a matrix multiplication. Where α_x is $(\frac{f}{S_x})$, α_y is $(\frac{f}{S_y})$ and S' is slant factor, when the image plane is not normal to the optical axis.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{S_x} & S' & C_x \\ 0 & \frac{1}{S_y} & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.3}$$

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & S & C_x \\ 0 & \alpha_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{2.4}$$

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} K_{3\times3} \end{bmatrix} \begin{bmatrix} I_{3\times3}|0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{2.5}$$

In summary, the equation 2.1 maps the world point measured in camera coordinates to image coordinates while equation 2.3 converts these image coordinates to pixel coordinates. Overall mapping from camera coordinates to the pixel coordinates is depicted in the equation eq. (2.4). K in equation 2.5 is known as the camera intrinsic matrix and depends only on the internals of the camera. In general, a 3D world point $(X, Y, Z)^T$ may not be known in the camera coordinate system however it can be mapped using 4×4 homogeneous transformation matrix. equation 2.6 is a mapping from a 3D world point to pixel coordinates. The 4×4 homogeneous transformation matrix is known as an extrinsic matrix and describes the position and orientation of the 3D world point in the camera coordinate system.

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & S & C_x \\ 0 & \alpha_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.6}$$

The equation 2.6 can be compactly written by combining both intrinsic and extrinsic matrix known as the projection matrix P_{3X4}. Where P = K[R|t], and λ is a arbitrary scaling factor for which equation 2.7 is satisfied.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & r_{22} & r_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.7}$$

We extract the camera center from the projection matrix as the camera center C = $-R^{-1}t$ or in other words, $t = -RC$. Therefore the projection matrix can be written as,

$$\begin{aligned} \mathtt{P} &= \mathtt{K}\ [\mathtt{R} \mid \boldsymbol{t}] \\ &= \mathtt{KR}\ [\mathtt{I} \mid -\boldsymbol{C}] \\ &= \mathtt{M}\ [\mathtt{I} \mid \mathtt{M}^{-1}\ p_4] \end{aligned} \tag{2.8}$$

where M = KR, K is a 3x3 upper triangular camera matrix, R is a 3x3 rotation matrix and p_4 is the last column of the projection matrix.

2.2 Camera calibration

Often in practice, estimating intrinsic and extrinsic parameters are important. Two approaches will be presented here, one using projection matrix from equation 2.7 (2D/3D correspondence) and using homography (2D/2D correspondence).

2.2.1 2D/3D correspondence

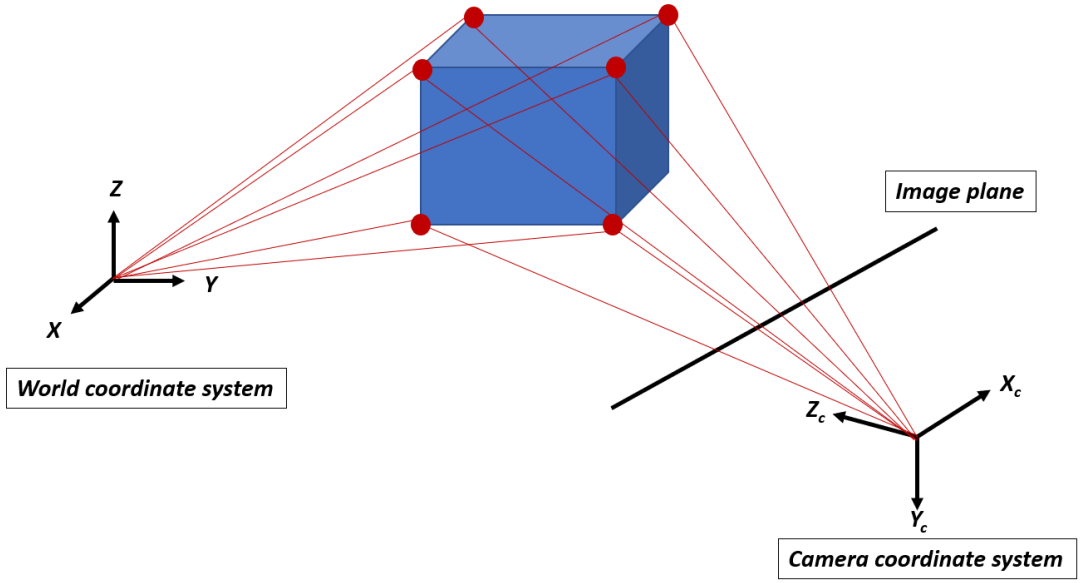

Fig. 2.3: Mapping 3D world points on to a 2D image plane.

The fig. 2.3 depicts a 2D image of a 3D object with known 6 unique points. Recall the equation 2.7 projection matrix P maps the world point to pixel coordinates and the equation is valid for the arbitrary scaling factor λ. First, the projection matrix P will be estimated using a given set of 3D world points and 2D image points and then it will be decomposed to intrinsic and extrinsic matrices as P = K[R|t]. The first step in estimating projection matrix is to convert the equation 2.7 into least square problem of type II (see appendix) Ap = 0 subjected to $\|p\| = 1$ and solve for vector $\boldsymbol{p}$ which is reshaped version of non-trivial elements of the projection matrix P.

Given a set of N corresponding 2D/3D points (u_i,X_i), a projection matrix P is needed such that

$$\lambda u_i = PX_i \ , where \ i = 1.....N$$

since the scaling factor λ is unknown and has to be estimated while estimating P. we will make use of DLT (direct linear transformation) to convert the above equation to the form Ap = 0.

$$\begin{bmatrix} \lambda u_i \\ \lambda v_i \\ \lambda \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & r_{22} & r_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

extracting the bottom row, equation for λ and substituting λ in other 2 rows yields

$$\lambda = p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}$$

$$u_i\lambda = u_i(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}) = p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}$$
$$v_i\lambda = v_i(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}) = p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24}$$

Rearrangement of the above equations leads to a linear system of equations with non-trivial elements of the projection matrix being reshaped into a vector $\boldsymbol{p}$.

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \end{bmatrix} \mathbf{p} = \mathbf{0} \tag{2.9}$$

$$with\ \mathbf{p} = \Big(p_{11}, p_{12}.....p_{33}, p_{34}\Big)^T \in \mathcal{R}^{12}$$

A projection matrix has 12 non-trivial elements thus 11 degrees of freedom (ignoring scaling) therefore it is necessary to have 11 equations to solve for P. Each pair of 2D/3D point correspondences leads to 2 equations thus a minimum of 6 2D/3D point correspondences are required. Stacking these 6 equations along rows leads to a linear system of equations Ap=0 as shown in eq. (2.10). This process of rearranging the equations is known as direct liner transformation (DLT).

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_6 & Y_6 & Z_6 & 1 & 0 & 0 & 0 & 0 & -u_6X_6 & -u_6Y_6 & -u_6Z_6 & -u_6 \\ 0 & 0 & 0 & 0 & X_6 & Y_6 & Z_6 & 1 & -v_6X_i & -v_6Y_6 & -v_6Z_6 & -v_6 \end{bmatrix} \mathbf{p} = \mathbf{0} \tag{2.10}$$

Exact solution. With a minimum of 6 correspondences, the solution to 2.10 will be exact which means 3D world points will be exactly gets projected to their measured image points correspondingly. The exact solution p, to $\mathrm{Ap} = 0$ is the right null-space of matrix A.

Over-determined solution. Practical measurements will be noisy due to various reasons and therefore we may require more than 6 2D/3D correspondences. In this case, the solution to $\mathrm{Ap} = 0$ will be obtained by minimizing the algebraic or geometric error of projection, subjected to some valid constraints.

Minimizing algebraic error. In this case the approach is,

$$\min \quad \|Ap\| \quad \text{s.t.} \quad \|p\| = 1 \tag{2.11}$$

The solution to eq. (2.11) is obtained from unit singular value of A corresponding to the smallest singular value (the least square problem of type II, see appendix).

Minimizing geometric error. First let us define what is geometric error. Let us recall eq. (2.7), $u_i = \mathrm{P}X_i$, suppose we know 3D world points X_i far more accurately than the measured image points then the geometric error in the image is

$$\sum_{i=1}^{n} d(u_i, \hat{u}_i)^2$$

where u_i is measured point in the image and $\hat{u}_i$ is P X_i which is the exact projection of X_i on to the image under P. If the measurement errors are Gaussian then the solution to

$$\min_P \sum_{i=1}^{n} d(u_i, PX_i)^2 \tag{2.12}$$

is the maximum likelihood of P as per [17]. Minimizing geometric error requires non-linear iterative methods such as Levenberg-Marquardt (LM) algorithms. Local minima can be found via LM, in order to find the global minima, the initial starting point can be linear solution obtained from eq. (2.11).

Having estimated projection matrix task at the hand is to decompose $P = K[R \mid t] = M[I \mid M^{-1} p_4]$ where $M = KR$. Decomposing M to K and R can be achieved using RQ decomposition with the constraint that diagonal entries of the K matrix has to be positive.

2.2.2 2D/2D correspondence

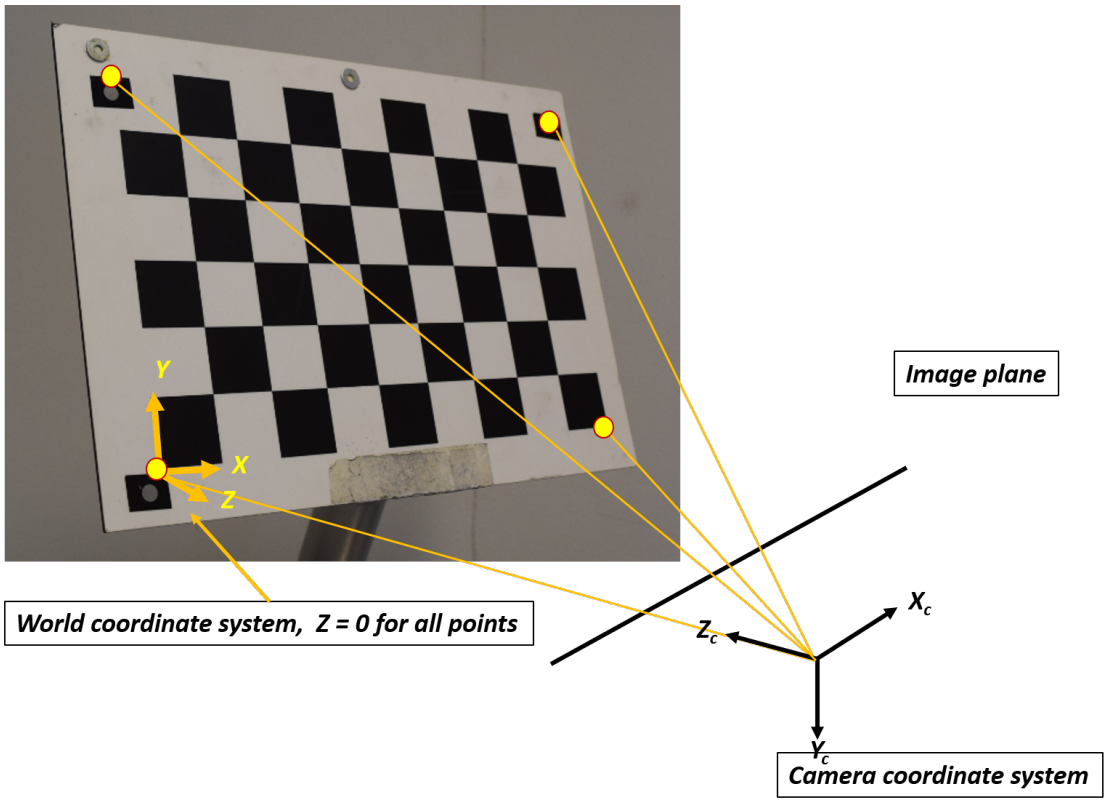

Fig. 2.4: Mapping 2D planar world points (Z = 0) on to a 2D image plane.

The disadvantage of 2D/3D correspondence way of estimating the projection matrix is that complete knowledge of the 3D location has to be known and it has to be precise as well. There is a flexible and computationally inexpensive method to estimate the projection matrix developed by Zhang [18] using a 2D planar object in 3D space as shown in fig. 2.4 along with its 2D image. Let us recall the equation 2.6

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & S & C_x \\ 0 & \alpha_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Using 2D planar object we eliminate Z coordinate thereby eliminating 3^{rd} column of the extrinsic matrix.

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & S & C_x \\ 0 & \alpha_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & S & C_x \\ 0 & \alpha_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2.13}$$

As in the case of 2D/3D problem setup, the equation 2.13 can be compactly written by combining both intrinsic and extrinsic matrices known an Homography matrix $H_{3\times3}$. And λ is a arbitrary scaling factor for which equation 2.14 is satisfied. In other words, a projection matrix $P_{3\times4}$ in planar case reduces to homography matrix $H_{3\times3}$.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2.14}$$

As in the case of 2D/3D problem, the homography matrix H will be estimated using a given set of 2D planar world points and 2D image points and then it will be decomposed to intrinsic and extrinsic matrices.

Given a set of N corresponding 2D/2D points (u_i,X_i), a homography matrix H is needed such that

$$\lambda u_i = HX_i \ , where \ i = 1.....N$$

Problem set up in 2D/2D case is very similar to 2D/3D problem except that the homography matrix has 9 non-trivial elements thus 8 degrees of freedom (ignoring scaling) therefore 8 independent equations are necessary to estimate H, hence 4 2D/2D point correspondences are required. At this point, the same estimating procedure used previously can be employed.

Exact solution. With a minimum of 4 correspondences, the solution to $Ah = 0$ will be exact which means 2D world points will be exactly gets projected to their measured image points correspondingly. The exact solution h, to Ah = 0 is the right null space of matrix A.

Over-determined solution. In this case solution to Ah = 0 will be obtained by minimizing the algebraic or geometric error of projection, subjected to some valid constraints.

Minimizing algebraic error. In this case the approach is,

$$\min \quad ||Ah|| \quad \text{s.t.} \quad ||h|| = 1 \tag{2.15}$$

The solution to eq. (2.15) is obtained from unit singular value of A corresponding to the smallest singular value (the least square problem of type II, see appendix).

Minimizing geometric error The geometric error in the image in 2D/2D case is

$$\sum_{i=1}^{n} d(u_i, \hat{u}_i)^2$$

where u_i is measured point in image and $\hat{u}_i$ is H X_i which is the exact projection of X_i on to the image under H. If the measurement errors are Gaussian then the solution to

$$\min_{P} \sum_{i=1}^{n} d(u_i, HX_i)^2 \tag{2.16}$$

is the maximum likelihood of H. Minimizing geometric error requires non-linear iterative methods such as Levenberg-Marquardt (LM) algorithms. Local minima can be found via LM, in order to find the global minima, initial starting point can be linear solution obtained from 2.15.

The disadvantage of 2D/2D homography is that with a single homography, not all the parameters of a projection matrix can be determined as Z coordinate of the world point is eliminated.

$$\mathbf{H} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_4 \end{bmatrix}$$

The decomposition of H to obtain extrinsic and intrinsic matrices is not possible, therefore, follows a different approach. We will first see how to get the intrinsic matrix K and as soon we have K, obtaining extrinsic is trivial.

Intrinsics Recall that $H = K\ [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}] = [\mathbf{p}_1\ \mathbf{p}_2\ \mathbf{p}_3]$ and the fact that R is a rotational matrix and can be expressed as,

$$\begin{aligned} \mathbf{r}_1^\intercal \mathbf{r}_2 &= 0 \\ \mathbf{r}_1^\intercal \mathbf{r}_1 = \mathbf{r}_2^\intercal \mathbf{r}_2 &= 1 \end{aligned}$$

rewriting with K gives,

$$\begin{aligned} \mathbf{p}_1^T K^{-T} K^{-1} \mathbf{p}_2 &= 0 \\ \mathbf{p}_1^T K^{-T} K^{-1} \mathbf{p}_1 = \mathbf{p}_2^T K^{-T} K^{-1} \mathbf{p}_2 &= 1 \end{aligned}$$

defining $\omega = K^{-T} K^{-1}$ as a symmetric matrix, above equations can be written as,

$$\omega = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{12} & \omega_{22} & \omega_{23} \\ \omega_{13} & \omega_{23} & \omega_{33} \end{bmatrix}$$

$$\begin{aligned} \mathbf{p}_1^T \omega \mathbf{p}_2 &= 0 \\ \mathbf{p}_1^T \omega \mathbf{p}_1 - \mathbf{p}_2^T \omega \mathbf{p}_2 &= 0 \end{aligned} \tag{2.17}$$

$\mathbf{p}_1$ and $\mathbf{p}_2$ are known from homography matrix and $\boldsymbol{\omega}$ has to be calculated. $\boldsymbol{\omega}$ can be estimated with techniques that we employed earlier with DLT () and SVD [23] by defining $\mathbf{b} = (\omega_{11}\ \omega_{12}\ \omega_{13}\ \omega_{22}\ \omega_{23}\ \omega_{33})^T$ and solving $\mathbf{Ab} = 0$. Each homography provides 2 independent rows therefore 3 such homographies are required to estimate $\boldsymbol{\omega}$. Once

the $\boldsymbol{\omega}$ is computed it can be decomposed in to $K^{-T}K^{-1}$ using Cholesky decomposition.

Extrinsics. Computing extrinsics post intrinsics is very straight forward as given by [18] we will first calculate rotation matrix followed by translation vector. $H = K\ [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]$ having known K we can rearrange as $K^{-1}\ H = H^{'} = [\mathbf{h}_1^{'}\ \mathbf{h}_2^{'}\ \mathbf{h}_3^{'}]$ which is equal to $[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]$. Since the rotational matrix is a orthogonal matrix, the third column is orthogonal to first 2 and therefore can be calculated by vector cross product $[\mathbf{h}_1^{'} \times \mathbf{h}_2^{'}]$. Hence, the complete rotational matrix is $Q = [\mathbf{h}_1^{'}\ \mathbf{h}_2^{'}\ \mathbf{h}_1^{'} \times\ \mathbf{h}_2^{'}]$ again which is equal to $R = [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_1\ \times\ \mathbf{r}_2\ ]$ which may not be true due to the noisy data. Therefore, it is necessary to find a best 3x3 rotation matrix R from a given 3x3 matrix Q. The "best" in the sense of smallest Frobenius norm of the difference $[R - Q]$ (see appendix). If the SVD of given matrix is $Q = UDV^T$, then the best rotation matrix is given by UV^T. The translation vector can be calculated as $\mathbf{t} = \mathbf{h}_3^{'}$ but the vector $\mathbf{t}$ has to be normalized therefore $\mathbf{t} = \mathbf{h}_3^{'}/\|\mathbf{h}_1^{'}\|$.

2.2.3 Lens distortions

A world point is mapped to image plane as $(x, y, z)^T = (f\frac{X_c}{Z_c}, f\frac{Y_c}{Z_c}, f)^T$ then image points are mapped to pixel coordinates as $(u, v, z)^T = (x + C_x, y + C_y, f)^T$ however, usually, there are radial and tangential distortions in the lens as depicted in fig. 2.5 therefore, image coordinates have to be corrected before mapping to pixel coordinates. The corrected image coordinates are

$$\begin{aligned} x^{'} &= x \times \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 xy + p_2(r^2 + 2x^2) \\ y^{'} &= y \times \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + p_1(r^2 + 2y^2) + 2p_2 xy \end{aligned} \tag{2.18}$$

where $r^2 = (x^2 + y^2)$, $k_1 ..... k_6$ are the radial distortion coefficients and p_1, p_2 are the tangential distortion coefficients. It is these corrected image coordinates are mapped to pixel coordinates as $(u, v, z)^T = (x^{'} + C_x, y^{'} + C_y, f)^T$.

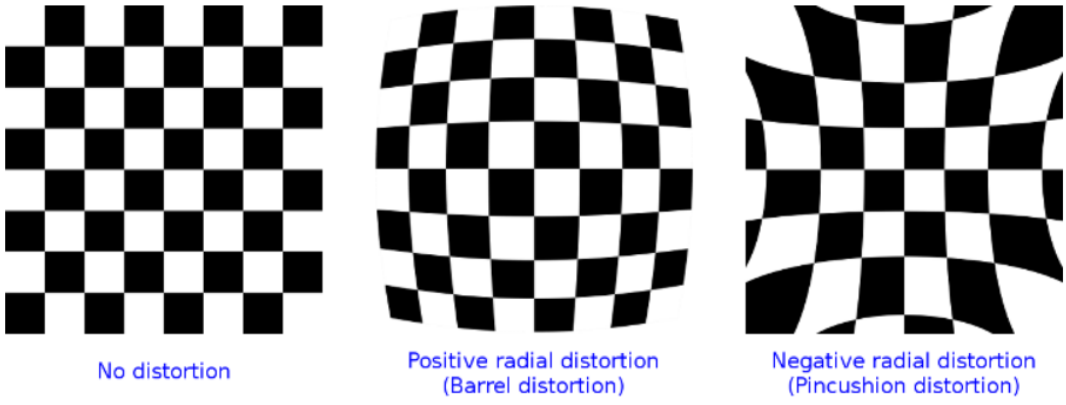

Fig. 2.5: Barrel distortion ($typically\ k_1 > 0$) and pincushion distortion ($typicallyk_1 < 0$) [16]

Camera calibration is the process of determining, intrinsic, extrinsic parameters along with distortion coefficients. The tab. 2.1 summarizes the calibration process discussed above.

Tab. 2.1: Calibration summary table

correspondences	Min. points	No.of images	[R | t]	K	world points
2D im./3D world	≥ 6	1	✓	✓	given
2D im./2D world	≥ 4	3	✓	✓	given
2D im./2D world	≥ 4	1	✓	given	given

2.3 Pixel to 3D points

Having obtained RGB and corresponding depth images of the scene (assuming depth to RGB calibrated), any specific pixel coordinates can be out-projected to obtain 3D locations expressed in the camera coordinates. Recall eq. (2.4) a 3D world point $(X_c, Y_c, Z_C)^T$ is mapped to the pixel coordinates and can be expressed as matrix multiplication. λ is the scaling factor.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & S & C_x \\ 0 & \alpha_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

By rearranging the above equation $(X_c, Y_c, Z_C)^T$ can be expressed as below.

$$\begin{aligned} X_c &= \frac{u - C_x}{\alpha_x} * \lambda \\ Y_c &= \frac{v - C_y}{\alpha_y} * \lambda \\ Z_c &= \lambda \end{aligned} \tag{2.19}$$

2.4 Hand-eye calibration

In general, robotic systems make use of many sensors and often require knowledge of static transformation between them. The fig. 2.6 depicts one such situation. These transformations can be estimated using a process known as hand-eye calibration [20].

There are four transformations, namely end-effector to base ${}^{\text{Base}}\mathtt{T}_{\text{EE}}$, chessboard to end-effector ${}^{\text{EE}}\mathtt{T}_{\text{CB}}$, chessboard to camera ${}^{\text{Cam}}\mathtt{T}_{\text{CB}}$ and camera to base ${}^{\text{Base}}\mathtt{T}_{\text{Cam}}$. Defining new variables ${}^{\text{Base}}\mathtt{T}_{\text{EE}}$ as M, ${}^{\text{EE}}\mathtt{T}_{\text{CB}}$ as X, ${}^{\text{Base}}\mathtt{T}_{\text{Cam}}$ as Y and ${}^{\text{Cam}}\mathtt{T}_{\text{CB}}$ as N we can write the transformation equation as

$$\mathtt{M}_i\mathtt{X} - \mathtt{Y}\mathtt{N}_i = 0 \tag{2.20}$$

where `M` is obtained through forward kinematics, `N` is obtained by calculating the pose of the calibration pattern from the camera image. Hand-eye calibration aims to solve

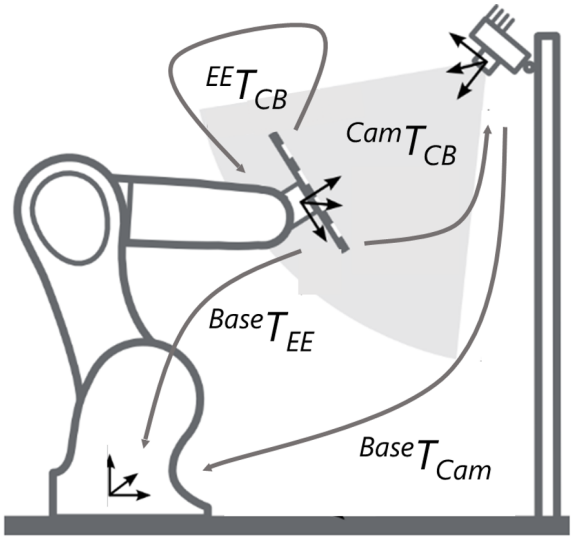

Fig. 2.6: Generic hand-eye calibration setup [19] (Modified). setup consists of a robot, a chessboard mounted to robot end-effector, and a RGB-D camera

the two unknowns in eq. (2.20) i.e the chessboard position with respect to the robot's end-effector $\mathtt{X}$ and the camera position with respect to the robot base $\mathtt{Y}$.

The equation 2.20 is transformed to a system of linear equations as

$$\mathtt{A}\boldsymbol{w} = \boldsymbol{b} \tag{2.21}$$

where, $\boldsymbol{w} = [x_{11}, x_{21}, ..., x_{34}, y_{11}, y_{21}, ..., y_{34}]$ is a vector of non-trivial elements of matrices $\mathtt{X}$ and $\mathtt{Y}$.

$$\mathtt{A} = \begin{bmatrix} A_1 \\ A_2 \\ . \\ . \\ . \\ A_n \end{bmatrix} \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_n \end{bmatrix}$$

where, $\mathtt{A} \in \mathbb{R}^{12n \times 24}$, $\boldsymbol{b} \in \mathbb{R}^{12n}$ and defined as,

$$\mathtt{A}_i = \begin{bmatrix} \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{11} & \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{21} & \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{31} & \mathtt{Z}_{3\times3} & \\ \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{12} & \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{22} & \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{32} & \mathtt{Z}_{3\times3} & -\mathtt{E}_{12} \\ \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{13} & \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{23} & \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{33} & \mathtt{Z}_{3\times3} & \\ \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{14} & \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{24} & \mathtt{R}[\mathtt{M}]_i(\mathtt{N}_i)_{34} & \mathtt{Z}_{3\times3} & \end{bmatrix}$$

and

$$\boldsymbol{b}_i = \begin{bmatrix} \mathtt{Z}_{9\times1} \\ -\mathtt{T}[\mathtt{M}_i] \end{bmatrix}$$

Here, $\mathtt{R}[\mathtt{M}]_i \in \mathbb{R}^{3\times3}$ is rotational part of $\mathtt{M}_i$ and $\mathtt{T}[\mathtt{M}_i] \in \mathbb{R}^3$ is translational part of $\mathtt{M}_i$. $\mathtt{Z}_{m\times n}$ is a m $\times$ n zero matrix and $\mathtt{E}_k$ is a k $\times$ k identity matrix.

Over-determined solution The system of equation 2.20 is a form of least square of type I (see appendix) and can be solved using QR factorization [21] by minimizing quadratic error.

$$\sum_{i=1}^{n} \|\mathtt{M}_i\mathtt{X} - \mathtt{Y}\mathtt{N}_i\|_F$$

Where, $\|.\|_f$ is Frobenoius norm. The above algorithm is called QR-24 according to [20].

Orthonormalization Note that matrices computed not necessarily orthogonal therefore we need to orthonormalize using singular value decomposition (SVD)

Calibration error Having finished the hand-eye calibration, It is necessary to investigate the accuracy of the obtained results. Accuracy can be obtained by rearranging the equation 2.20 as,

$$({}^{\mathrm{EE}}\mathtt{T}_{\mathrm{CB}})^{-1}({}^{\mathrm{Base}}\mathtt{T}_{\mathrm{EE}})^{-1\mathrm{Base}}\mathtt{T}_{\mathrm{Cam}}{}^{\mathrm{Cam}}\mathtt{T}_{\mathrm{CB}} = \mathtt{I}_{4\times4}$$

The above equation will not hold in reality due to noisy data, camera calibrations error etc. RHS of the above equation will be fully populated matrix instead of an identity matrix. two error metrics can be defined to individually evaluate the rotation and translation parts per [20].

$$\begin{aligned} \textit{Let } \mathtt{A}'_{4\times4} = ({}^{\mathrm{EE}}\mathtt{T}_{\mathrm{CB}})^{-1}({}^{\mathrm{Base}}\mathtt{T}_{\mathrm{EE}})^{-1\mathrm{Base}}\mathtt{T}_{\mathrm{Cam}}{}^{\mathrm{Cam}}\mathtt{T}_{\mathrm{CB}} \\ \mathtt{SVD}\ (\mathtt{A}'_{4\times4}) = \mathtt{UDV}\ \textit{then}, \\ \mathtt{A}_{4\times4} = \mathtt{U}\mathtt{V}^T \end{aligned}$$

then, translation error can be defined as,

$$\boldsymbol{e}_{trans}[\mathtt{A}] = \sqrt{\mathtt{A}^2_{4\times1} + \mathtt{A}^2_{4\times2} + \mathtt{A}^2_{4\times3}}$$

let (k,θ) be axis angle representation [22] of $\mathtt{R}[\mathtt{A}]$ then the rotational error metric is,

$$\boldsymbol{e}_{rot}[\mathtt{A}] = |\theta|$$

in the rotation metric the axis of rotation is neglected.

2.5 Point cloud registration

The objective of any point cloud registration algorithm is to find the best transformation that aligns 2 point clouds (source and target). A common point cloud registration technique used is the iterative closest point (ICP) algorithm [14]. A brief introduction to ICP is provided in this section.

Let $P = \{p_1, p_2, ..., p_m\}$ be the source point cloud set that needs to be aligned with the target point cloud set $Q = \{q_1, q_2, ..., q_n\}$ with correspondences $C = \{(i, j)\}$.

objective : find the best transformation $T = [R|t]$ that aligns the source point cloud set to the target point cloud set which minimizes the mean square objective function given below.

$$\begin{aligned} \mathtt{E}(\mathtt{R}|\mathtt{t}) &= \sum_{(i,j)\in C} \|\boldsymbol{q}_i - \mathtt{R}\boldsymbol{p}_j - \boldsymbol{t}\|^2 \\ &\textit{or} \\ \mathtt{E}(\mathtt{R}|\mathtt{t}) &= \sum_{(i,j)\in C} \|\boldsymbol{q}_i - \mathtt{T}[\mathtt{R}|\mathtt{t}]\boldsymbol{p}_j\|^2 \end{aligned} \tag{2.22}$$

where, R is a 3×3 rotation matrix, t is a 3×1 translation vector, and T[R|t] is a 4×4 homogeneous transformation matrix.

Case 1: Known data association. In this case each point in source point cloud set p_i corresponds to q_j point in the target point cloud set with same index or to any known index. Then, eq. (2.22) can be solved by finding the center of mass and singular value decomposition.

The center of mass for the source point cloud set and target point cloud set is

$$\begin{aligned} \mu_p &= \frac{1}{\|C\|} \sum_{(i,j)\in C} p_i \\ \mu_q &= \frac{1}{\|C\|} \sum_{(i,j)\in C} q_i \end{aligned} \tag{2.23}$$

By subtracting the center of mass from every point in the source and target point cloud, we get

$$\begin{aligned} Q' &= \{q_i - \mu_q\} = \{q_i'\} \\ P' &= \{p_j - \mu_p\} = \{p_j'\} \end{aligned} \tag{2.24}$$

minimizing eq. (2.22) is equivalent to minimizing,

$$\mathtt{E}'(\mathtt{R}) = \sum_{(i,j)\in C} \|[q_1', q_2', ..., q_n'] - R[p_1', p_2', ..., p_m']\|_F^2 \tag{2.25}$$

and is called orthogonal procrustes problem and can be solved by constructing cross-covariance matrix of 2 point cloud sets and SVD [23]. The cross-covariance matrix of 2 point cloud sets is,

$$\begin{aligned} W &= \sum_{(i,j)\in C} \{q_i - \mu_q\}\{p_i - \mu_p\}^T \\ W &= \sum_{(i,j)\in C} q_i' {p_j'}^T \end{aligned} \tag{2.26}$$

Use the SVD to decompose the cross-covariance matrix hence find the transformation between 2 point clouds.

$$\begin{aligned} W &= UDV^T \; then, \\ R &= UV^T \\ t &= \mu_q - R\mu_p \end{aligned} \tag{2.27}$$

In conclusion, a point cloud matching algorithm estimates the 3×3 rotation matrix R and 3×1 translation vector t.

$$(T[R|t], MSE) = PointCloudRegistration(P, Q)$$

where, T[R|t] is the 4×4 homogeneous transformation between source and target point cloud and MSE is the mean square error of matching as in eq. (2.22).

Case 2: Unknown data association. When the correspondence between source & target point cloud set is unknown, we resort to iterative closest point algorithm. Where for each point in the source point cloud set $P = \{p_1, p_2, ..., p_m\}$ corresponding closest point in the target point cloud $Q = \{q_1, q_2, ..., q_n\}$ is calculated before proceeding to the point cloud matching.

The Euclidean distance $d(\vec{p}_1, \vec{p}_2)$ between two points $\vec{p}_1 = (x_1, y_1, z_1)$ and $\vec{p}_2 = (x_2, y_2, z_2)$ is

$$d(\vec{p}_1, \vec{p}_2) = \|\vec{p}_1 - \vec{p}_2\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

The distance between a point p in the source point cloud set and the target point cloud set $Q = \{q_1, q_2, ..., q_n\}$, $d(\vec{p}, Q)$ is

$$\begin{aligned} d(\vec{p}, Q) &= \min_{i \in Q} \; d(\vec{p}, \vec{q}_i) \\ d(\vec{p}, Q) &= \min_{i \in Q} \|\vec{p} - \vec{q}_i\| \end{aligned} \tag{2.28}$$

The point in target point cloud $q* \in Q$ that yields a minimum distance is the closest point. The unknown data association is solved when closest point computation is performed for each point in the source point cloud set $P = \{p_1, p_2, ..., p_m\}$.

$$C = ComputeClosestPoint(P, Q) \tag{2.29}$$

where $C = \{(i, j)\}$ is the computed correspondence set for each point in the source point cloud. Having computed the correspondence set, one can proceed with point cloud registration. Each point the source point cloud is updated via $P_{k+1} = T[R|t]P_k$

$$P_{k+1} = ApplyRegistration(T[R|t], P_k) \tag{2.30}$$

where, P_{k+1} are the points after applying the registration (homogeneous representation is used here to enable matrix multiplication).

2.5.1 Iterative closest point algorithm

1. Given $P = \{p_1, p_2, ..., p_m\}$ be the source point cloud set that needs to be aligned with the target point cloud set $Q = \{q_1, q_2, ..., q_n\}$ with unknown data association.
2. Initialize the iteration $P_0 = P$, $T[R|t]$ = 4X4 identity [], $k = 0$
   a) $C_k = ComputeClosestPoint(P_k, Q)$

b) $(T[R|t]_k, MSE_k) = PointCloudRegistration(P_0, C_k)$

c) $P_{k+1} = ApplyRegistration(T[R|t]_k, P_0)$

d) Terminate the iteration when

   i. change in the mean square error is less than a predefined threshold $\tau > 0$ i.e. $MSE_k - MSE_{k+1} < \tau$.

   ii. or when iteration exceeds the predefined number.

2.6 Simultaneous Localization and Mapping

2.6.1 Map

Formally, a map m is a list of objects in the environment along with their properties [24]. Especially, in this case, the map is a collection of electrodes and their locations on the EEG cap.

$$m = \{m_1, m_2, ...m_N\}$$

where, N is the total number of electrodes on the cap and m_n where $1 \leq n \leq N$ represents the 3D location of the n^{th} electrode.

2.6.2 Localization

In simple words, localization is the pose estimation problem. It is the process of determining the pose of a camera relative to the given map of the electrodes using a wide variety of sensor data i.e RGB-D images and odometry etc. Localization is also the problem of establishing the correspondence between the coordinate system based on which the map is defined (often, global coordinate system) and the camera's local coordinate system. For the problem addressed in this thesis, this correspondence enables the camera to express the electrodes in the map in its local coordinate system. Unfortunately, sensor measurements are noisy, therefore pose has to be inferred from the noisy data which makes localization a difficult problem to solve. There are many probabilistic approaches to solve localization problems, the reader is encouraged to refer [24] for more details.

2.6.3 SLAM

Simultaneous localization and mapping (SLAM) is a special case, for the problem at hand, where the camera neither has the pre-build map of the electrodes nor does it know its pose relative to the map. The camera can however be moved along a trajectory either with a help of a robot or manually within the environment and also collect RGB and depth images along the way. Finally, the task is to build the map of the electrodes and simultaneously localize the camera with this map. Since measurements are prone to noise, SLAM problems are modeled probabilistically.

Probabilistic formulation of SLAM. The camera moves with in an unknown environment along a trajectory. Camera poses along the trajectory are described by the

random variables $x_{1:t} = \{x_1, ..., x_t\}$. A set of odometry can be generated along the trajectory $u_{1:t} = \{u_1, ..., u_t\}$ and perceived 3D electrode position can be expressed by $z_{1:t} = \{z_1, ..., z_t\}$. SLAM can be broadly classified in to 2 types, online-SLAM and full-SLAM.

Online-SLAM. Estimating the current pose and the map.

$$p(x_t, m|z_{1:t}, u_{1:t})$$

where x_t is the camera pose at current time t, m is the map, $z_{1:t}$ and $u_{1:t}$ are measurement and control inputs until the current time t respectively. Online-SLAM is essentially an online state estimation problem where the state is the current camera pose and the map. The pose and the map is estimated and refined as and when the new measurements are available. Online-SLAM problem is modeled and solved using filtering techniques such as Kalman and information filters [25] [26] and particle filters [27].

Full-SLAM. Estimating the entire camera trajectory and the map instead of just momentary pose.

$$p(x_{1:t}, m|z_{1:t}, u_{1:t})$$

Full-SLAM estimates $x_{1:t}$ the entire trajectory of the camera and simultaneously mapping the electrodes by incorporating a full set of measurements and control inputs $z_{1:t}$ and $u_{1:t}$. Smoothing techniques like [28] [29] are employed to solve the full-SLAM problems. The modern and intuitive approach for solving a full-SLAM problem is to use graph-based formulation. Due to the improvements in the areas of computational power and efficient linear algebra solvers, graph-based formulations have gained popularity over the years [28].

2.7 Graph-Based SLAM (Pose-Graph SLAM)

The graph-based approach uses an intuitive graph that consists of nodes and edges to represent the full-SLAM problem. Each node corresponds to the camera pose along the trajectory and an edge between two nodes corresponds to the spatial constraints (odometry/loop closure constraints) between them. These constraints are inherently uncertain in nature. The idea of the graph-based approach is to first construct the pose-graph and determine an optimized node configuration (camera poses) that minimizes the error introduced by spatial constraints. Finally, render a map based on these optimized poses as shown in fig. 2.7. The underlying mathematics and equations are referenced from the literature [15], [34].

2.7.1 Pose-Graph construction

The graph consists of n nodes $x_{1:t}$ which corresponds to the camera pose at time t. Every time the camera moves from x_i to x_{i+1} position an edge(odometry) constraint is added to the graph connecting nodes x_i and x_{i+1}. If the camera revisits the previously visited

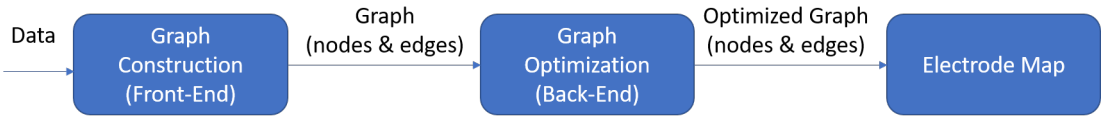

Fig. 2.7: Graph based SLAM in a nutshell.

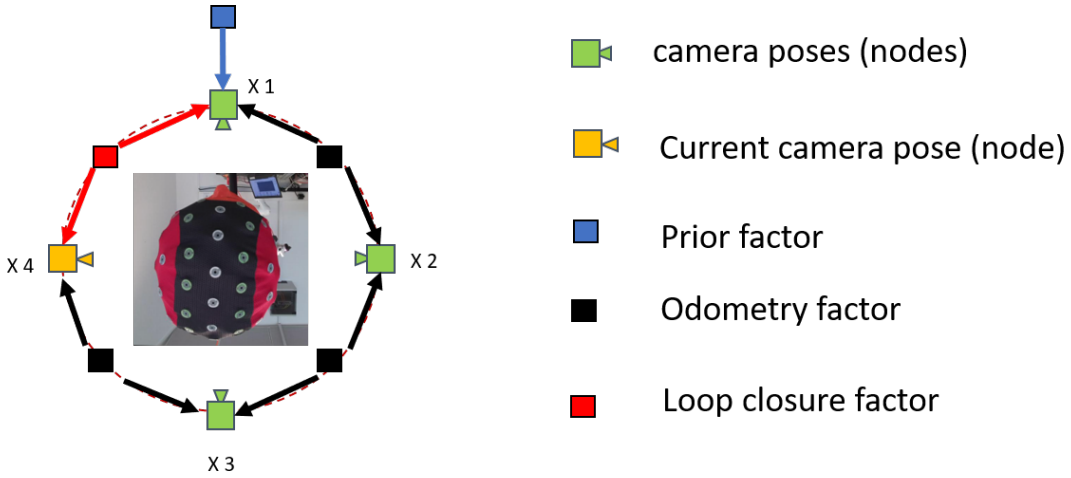

Fig. 2.8: Pose-Graph consists of nodes and edges.

position, for example x_4, an loop closure constraint is added to the graph connecting the current node x_t and x_j as shown in fig. 2.8.

Since spatial constraints associated with edges (sensor measurement of some sort) are inherently noisy, these are modeled probabilistically. Let Z_{ij} and Ω be mean and information matrix of a measurement (please see appendix on Gaussian distribution) between node x_i and x_j. Let $\hat{Z}_{ij}$ the prediction of this measurement given a node configuration x_i and x_j as shown in fig. 2.9.

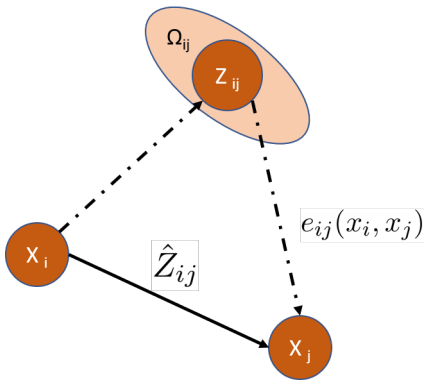

Fig. 2.9: Probabilistic model of measurements [15].

The log-likelihood l_{ij} of a measurement Z_{ij} is therefore,

$$l_{ij} \propto [Z_{ij} - \hat{Z}_{ij}(x_i, x_j)]^T \Omega_{ij} [Z_{ij} - \hat{Z}_{ij}(x_i, x_j)] \tag{2.31}$$

Let $e_{ij}(x_i, x_j)$ be the function that computes difference between real and expected observation/measurement,

$$e_{ij}(x_i, x_j) = [Z_{ij} - \hat{Z}_{ij}(x_i, x_j)] \tag{2.32}$$

Let $\mathcal{C}$ be the set with many nodes for which observation Z_{ij} exists then, the objective of maximum likelihood approach is to find the configuration X^* that minimizes the negative log likelihood of the function $F(x)$ of all observations.

$$F(x) = \sum_{(i,j)\in\mathcal{C}} e_{ij}^T \Omega_{ij} e_{ij}$$

which can be rewritten as,

$$X^* = \underset{x}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{C}} e_{ij}^T \Omega_{ij} e_{ij} \tag{2.33}$$

2.7.2 Pose-Graph optimization

The eq. (2.33) is a non-linear least square problem and solution can be obtained using Gauss-Newton or Lavenberg-Marquardt algorithms [15]. Dellaert and Kaess [29] have shown the connection between factor graphs and non-linear least square problems. A Factor graphs encodes the probabilistic nature of SLAM. A factor graph represents joint probabilities of all the factors with factor nodes $f_i \in \mathcal{F}$, variable nodes $x_i \in \mathcal{X}$.

$$g(\mathcal{X}) = \prod_i f_i(\mathcal{X}_i) \tag{2.34}$$

each factor f_i consists of measurement function $h_i(\mathcal{X}_i)$ and a measurement z_i. Assuming Gaussian measurement models f_i can be given as

$$f_i(\mathcal{X}_i) \propto exp(\frac{1}{2}\|h_i(\mathcal{X}_i) - z_i\|^2_\Sigma)$$

with $\|e\|^2_\Sigma = e^T\Sigma^{-1}e$ being the Mahalanobis distance with covariance matrix Σ. A set of node configurations $\mathcal{X}^*$ that maximizes eq. (2.34) can be posed as a nonlinear least square problem.

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmin}} \frac{1}{2} \sum_{(i,j)\in\mathcal{C}} \|h_i(\mathcal{X}_i) - z_i\|^2_\Sigma \tag{2.35}$$

The fig. 2.8 describes simple scenario where there are four camera positions $x_{1:4}$ corresponding to the time stamps $t_{1:4}$. A factor graph models the pose-graph as fig. 2.10. Prior factor describes prior distribution on the very first pose x_1 and encodes the relation with world coordinate frame. Odometry factors encodes the spatial constraint between subsequent nodes x_i with x_{i+1} and loop factor encodes the relative spatial constraint between x_4 and x_1.

$$g(\mathcal{X}) = \underbrace{f_i(x_1)}_{\text{Prior}} \underbrace{f_i(x_1, x_2) f_i(x_2, x_3) f_i(x_3, x_4)}_{\text{Odometry}} \underbrace{f_i(x_4, x_1)}_{\text{Loop closure}}$$

Fig. 2.10: Pose-Graph as a factor graph model.

2.8 Electrode detection and localization

Electrodes captured in a sequence of RGB images need to be detected and simultaneously localized as shown in fig. 2.11 in order to create 3D point clouds for sequential ICP registration and SLAM. YOLO(You Only Look Once) being one of the most popular convolutional neural network based object detection and localization algorithm has been employed for this very purpose. YOLO has been trained on a large data-set of EEG cap and trained weights are available as a part of the pipeline. The center point of YOLO bounding box is considered as the 2D electrode position.

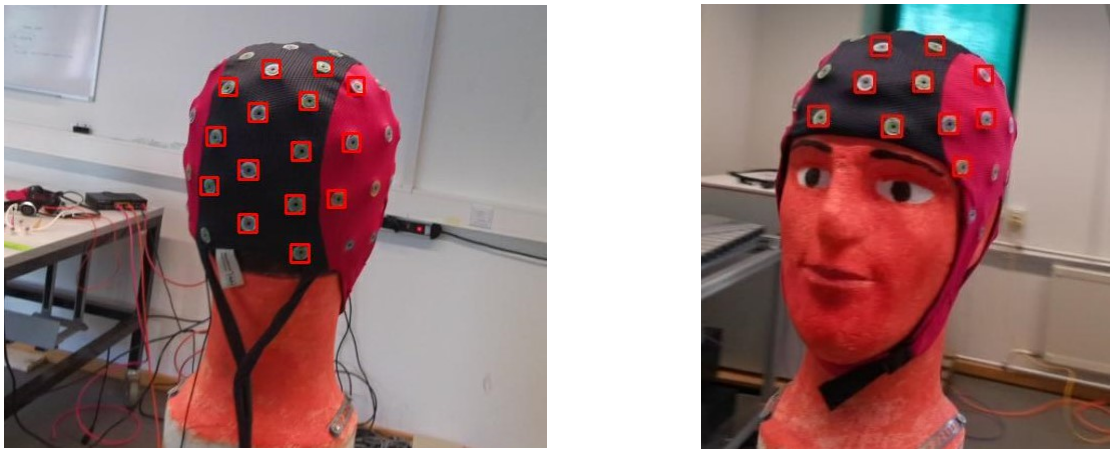

Fig. 2.11: Electrode detection and localization using YOLO.

2.9 Cluster processing

Two mainly used cluster processing algorithms in this thesis are DBSCAN and K-Means provided by Scikit-learn.

2.9.1 DBSCAN

DBSCAN is a density based clustering algorithm, it differentiates the clusters based on the high and low density of points. It labels samples(points) that are in the high density as core samples, A cluster, therefore, a collection of many core samples. In this thesis, DBSCAN is used to estimate the number of clusters in the electrode map. *eps* and *min_samples* are the main arguments required by DBSCAN. *eps* is the maximum distance between two samples to be considered as in the neighborhood of each other and *min_samples* is the number of samples to be considered as a core point [30].

```
DBSCAN(eps, min_samples, ...)
output: labels, n_clusters
```

2.9.2 K-means

K-Means on the other hand tries to group the samples of equal variance together by minimizing the inertia or within-cluster sum-of-squares. However, K-Means requires number of clusters to be specified before hand. K-Means divides $\mathcal{N}$ samples in to $\mathcal{K}$ disjoint clusters $\mathcal{C}$ where each cluster contains $\mathcal{X}$ samples and described by centroids of the cluster μ_j. K-Means aims to determine the μ_j that minimizes the inertia defined by eq. (2.36). The argument *n_clusters* is number of clusters to form as well as number of centroids to generate [30].

$$\sum_{i=0}^{n} \min_{\mu_j \in C} \|x_i - \mu_j\|^2 \tag{2.36}$$

```
KMeans(n_clusters,...)
output: labels, cluster_centers
```

3 Methods and material

3.1 Algorithms for electrode digitalization

In this section, we provide an overview of the algorithms required for electrode digitalization process which consists of various modules. The individual modules and their input/outputs are shown in fig. 3.1. The robot guided data set and electrode detection module have been previously implemented and are used in this thesis as a part of the pipeline (orange boxes). The RGB-D camera provides color image and corresponding calibrated depth map. The module *Electrode detection* detects the visible electrodes in each camera frame and looks up the corresponding depth value for each electrode. These two values are combined to form 3D point clouds of electrodes in each frame for further processing. The module *Scan registration* estimates the relative camera movement between current and previous camera frame by performing the ICP registration between two 3D point clouds. For each camera frame, this module saves the current position relative to the very first camera position (time t =0), relative camera motion estimated previously, and 3D clouds corresponding to current frame (more details on the data structure to follow). The *SLAM* module is responsible for pose-graph creation and optimization. Output is sent to the *Cluster processing* module which finds the final 3D position of electrodes expressed in the first camera frame.

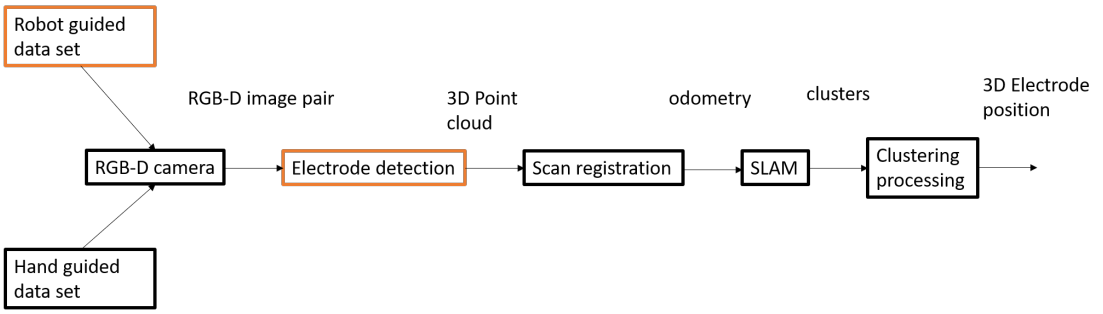

Fig. 3.1: Overview of algorithms for electrode digitalization.

Several libraries have been used to implement each module used for electrode digitalization. Therefore, we give a brief introduction to various libraries used in the rest of this section.

3.1.1 ROS

Robot operating system (ROS) [31] is an open-source software which serves as middleware for robotic software development across many platforms. ROS offers a flexible framework of numerous libraries, software tools for communication, motion planning, navigation, localization and mapping and for robust robotic software development. ROS supports many programming languages including C++, Python, Matlab, etc. A modular

and distributed software platform along with a vibrant user community all over the world makes ROS very popular among roboticists, students and researchers. In this project, communication between camera, robot and modules are implemented by exploiting ROS features like publishers and subscribers and visualization tool Rviz has been employed effectively.

3.1.2 OpenCV

OpenCV [16] is a open source computer vision and machine learning library. It provides a flexible framework and highly optimized codes to execute computer vision and machine learning tasks. OpenCV can be used with many programming languages like C++, Python, and comes with Matlab interface. It offers more than 2500 highly optimized codes for performing classical computer vision applications, for example, single/stereo camera calibration, 3D reconstruction, depth mapping etc. and state of the art machine learning tasks such as object detection, human face recognition in the video etc. In this thesis, OpenCV is used mainly for camera calibration and chess board pose estimation.

3.1.3 Matlab

Matlab [32] is a popular platform to design, analyze and simulate real world problems using matrix based language. Matlab provides numerous technology packages/apps to solve problems in fields of science and engineering. In this thesis, Matlab is used for camera calibration.

3.1.4 Open3D

Open3D [33] is an open-source library loaded with set of tools for 3D data processing such as scan matching (ICP), point cloud visualization, RGB-D image based odometry, etc. Open3D supports development using C++ and Python programming languages. In this thesis, ICP scan registration and point cloud visualization is implemented using open3D.

3.1.5 miniSam

miniSam [34] is an open-source library which provides factor graph based framework to solve non-linear least square optimization problem. While front-end of miniSam supports c++ and python programming languages, back-end is highly optimized for efficient computation. The task of pose-graph creation and optimization has been efficiently used in this thesis work.

3.1.6 Scikit-learn

Scikit-learn [35] is a open-source python based machine learning framework that supports classification, Regression, Clustering, Dimensionality reduction, etc. Scikit-learn offers various algorithms for cluster processing such as K-Means, DBSCAN, etc. Scikit-learn is mainly used for cluster processing.

3.2 Experimental setup

The setup consists of a RGB-D camera (Azure kinect), a tracking camera (fusionTrac 500 by Atarcsys or Cambar B2 by Axios 3D), a robot (KUKA), one phantom head, multiple EEG caps, one chess board, and two reflective markers.

3.2.1 Kinect camera

The Azure Kinect camera [36] comes with many sophisticated sensors integrated. For example, 12Megapixel RGB sensor, 1 Megapixel time of flight (Tof) depth sensors both capable of recording images at 30 fps, motion sensor (accelerometer and gyroscope), and a microphone as shown in fig. 3.2. Azure Kinect offers many configurations (Resolution, fps, FOV) for acquiring RGB and depth images. Different camera modes and associated errors are shown in fig. 3.3. Azure Kinect comes with a software development kit (SDK) which can be integrated with robot operating system (ROS). In this thesis, Kinect is sometimes referred to as RGB-D or kCam especially in the graphics.

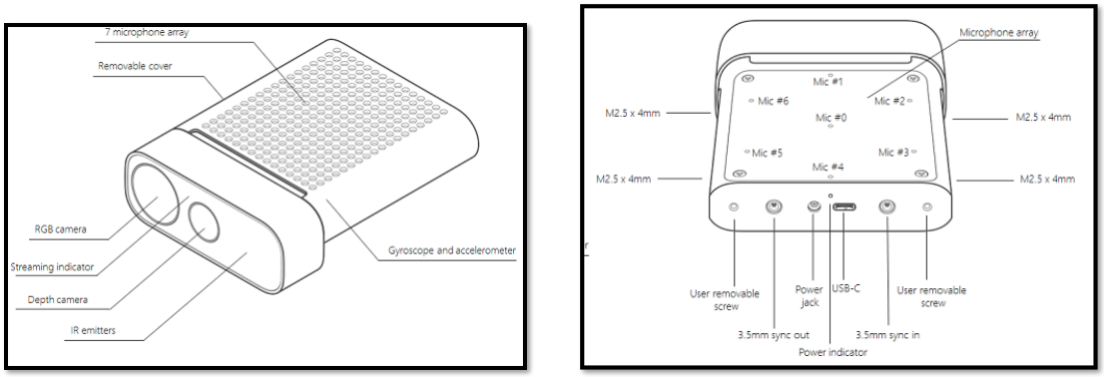

Fig. 3.2: Microsoft azure kinect camera features [37].

3.2.2 Tracking camera

The fusionTrac 500 by Atarcsys and Cambar B2 by Axios 3D fig. 3.4 are active and passive, real-time optical pose-tracking system specially designed to detect and track reflective spheres, disks, and IR-LEDs in real-time [38] [39]. These consists of stereo set-up to observe reflective and/or active fiducials (IR LEDs) simultaneously, and triangulates the detected markers to calculate their locations with high precision and measurement rates. These systems can be used to estimate the pose when several fiducials are arranged to form a specific shape. The fusionTrac 500 is used in robotic-guided case while Cambar B2 is used during hand-guided case. In this thesis both fusionTrac and Cambar is some times interchangeably used and referred as tracking camera or tCam especially in the graphics.

3.2.3 Robot, phantom, EEG caps

The KUKA LBR iiwa 14 med robot has been employed in order to guide the RGB-D camera along predefined trajectories shown in fig. 3.5. Several EEG caps are provided by eemagine Medical Imaging Solutions GmbH, Berlin, Germany. One cap with 63

Depth camera supported operating modes

Azure Kinect DK integrates a Microsoft designed 1-Megapixel Time-of-Flight (ToF) depth camera using the image sensor presented at ISSCC 2018. The depth camera supports the modes indicated below:

Mode	Resolution	FoI	FPS	Operating range*	Exposure time
NFOV unbinned	640x576	75°x65°	0, 5, 15, 30	0.5 - 3.86 m	12.8 ms
NFOV 2x2 binned (SW)	320x288	75°x65°	0, 5, 15, 30	0.5 - 5.46 m	12.8 ms
WFOV 2x2 binned	512x512	120°x120°	0, 5, 15, 30	0.25 - 2.88 m	12.8 ms
WFOV unbinned	1024x1024	120°x120°	0, 5, 15	0.25 - 2.21 m	20.3 ms
Passive IR	1024x1024	N/A	0, 5, 15, 30	N/A	1.6 ms

*15% to 95% reflectivity at 850nm, 2.2 µW/cm^2/nm, random error std. dev. ≤ 17 mm, typical systematic error < 11 mm + 0.1% of distance without multi-path interference. Depth may be provided outside of the operating range indicated above. It depends on an object's reflectivity.

Fig. 3.3: Figure highlights the different modes offered by Kinect. The random error standard deviation $\leq$ 17mm, typical systematic error $<$ 11mm + 0.1% of distance without multi-path interference. [37].

Fig. 3.4: Left: fusionTrac 500 by Atarcsys [38], Right: Cambar B2 by Axios 3D [39] mounted on the ceiling

electrodes and two with 23 electrodes each are used in this thesis. These caps are strapped on a dummy phantom head as shown in fig. 3.6.

3.2.4 Markers

A Chessboard with grid size (5 × 8) is used for camera and hand-eye calibration. One reflective marker which has a specific arrangement of fiducials is rigidly attached to the Kinect. The pose can be estimated when the marker is exposed to the tracking camera. Since this marker is attached on an arbitrary position, exact transformation between kinect coordinate frame and the marker is unknown and will be estimated using eye-in-hand calibration. Another reflective marker with a different arrangement of fiducials is used for electrode digitalization. The tracking camera records the 3D position

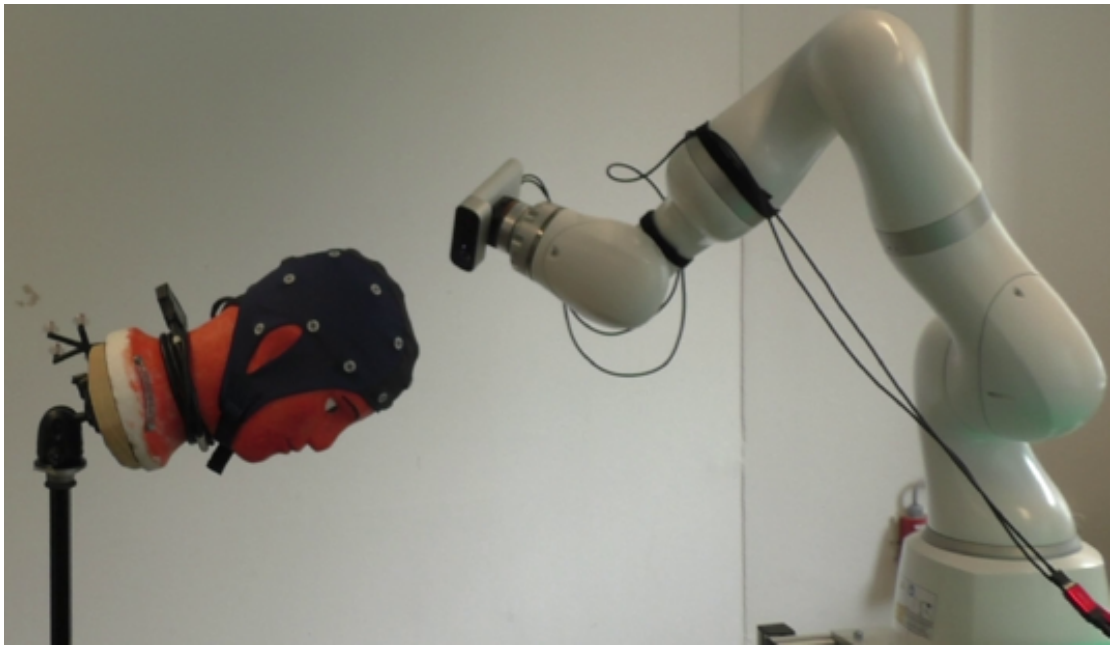

Fig. 3.5: The KUKA LBR iiwa 14 med robot.

Fig. 3.6: Left to right, 63 electrodes with size M (*Cap_63*), 23 electrodes size L (*Cap_23_1*), 23 electrodes size M (*Cap_23_2*)

of marker tip. Please note that these markers have to be calibrated with tracking camera before hand. The fig. 3.7 shows different markers used.

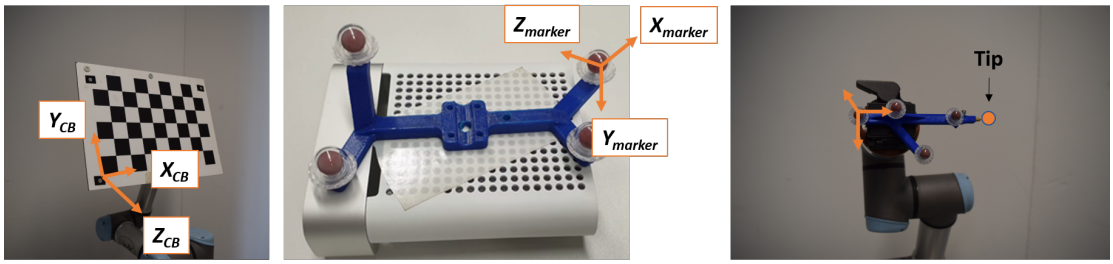

Fig. 3.7: chessboard along with reflective markers serving specific purpose.

3.3 Camera calibration and pixel value out-projection with chessboard

3.3.1 Camera calibration

Camera calibration is carried out using open-source computer vision library OpenCV and using a readily available calibration app from Matlab. Camera calibration is based on 2D/2D point correspondences with the chessboard as a 2D planar object. The homography is calculated using the square corners of a chessboard (world points) and its image points. OpenCV needs arrays of world points, image points, and grid size of the chessboard (in our case its 5 rows, 8 columns). Several images at different depths in steps of $\approx$ 200mm starting $\approx$ 400mm from the camera were captured and an array of world points (x,y) location of chessboard corners [(0,0), (40,0), (80,0)...] was fed to the algorithm. OpenCV automatically detects these chessboard corners from the images refines them accordingly.

```
cv2.calibrateCamera(object_points, image_points, ...)
output: rms, camera_matrix, dist_coeffs, rot_vecs, trans_vecs
```

The OpenCV function cv2.calibrateCamera takes in these world points (object points), image points and some more arguments then output geometric error of re-projection (rms), intrinsic parameters (camera_matrix) distortion coefficients (dist_coeffs) and extrinsic parameters ($rot\_vecs, trans\_vecs$). Matlab on the other hand requires only single square size of the chessboard along with images.

3.3.2 Chessboard pose estimation

Having completed the camera calibration we can now make use of the intrinsic parameters and the distortion coefficients as an input to the pose estimation algorithm provided by OpenCV.

```
cv2.solvePnP(object_points, image_points, intr_mat, dist_coeffs)
output: rot_vecs, trans_vecs
```

The OpenCV function cv2.solvePnP takes object points, image points, intrinsic matrix, and distortion coefficients as the arguments and computes rotation and translation vectors. The rotation vector can be converted to 3×3 rotational matrix using cv2.Rodrigues function provided by OpenCV. By combining rotation matrix and translation vector we can form 4×4 homogeneous matrix for further manipulation. One can perform a sanity check on the pose estimation using the output of cv2.solvePnP. We pretend that image points for chessboard pose is unknown for which output of the cv2.solvePnP are known. Image points can be calculated back using OpenCV function cv2.ProjectPoints and obtained points can be projected on an image for visual sanity check as shown in fig. 3.8.

```
cv2.ProjectPoints(object_points,rot_vecs,trans_vecs,intr_mat,dist_coeffs)
output: image_points
```

Fig. 3.8: Chessboard visual sanity check on three different depth values.

3.3.3 Pixel points out-projection

As shown in eq. (2.19) pixel values out-projected to form 3D points (X_c, Y_c, Z_c) directly proportional to the depth and indirectly proportional to the estimated focal length of the camera. Depth values obtained from a TOF camera depends on various factors such as, the reflectivity of the material, lighting conditions, etc. The Kinect has a random error standard deviation $\leq$ 17mm, typical systematic error $<$ 11mm + 0.1% of distance without multi-path interference. Having finished the camera calibration, a chessboard can be used to observe the above phenomena. Several images of the chessboard at different depths in steps of $\approx$ 200mm starting $\approx$ 400mm from the camera were captured. At each depth distance, images with different orientations of the chessboard at 30fps were captured. Firstly, for each image, the chessboard pose is estimated using OpenCV solvePnP method and ground truth object points are calculated in camera coordinates. Secondly, corresponding image points are out-projected using raw depth value (λ) from the Kinect as shown in fig. 3.9.

For each image, out-projected values (estimated object points) are then compared to the ground truth values with L2 norm as the metric. Since there are many images at a particular depth distance, L2 norm will be averaged over all the images. Refer results section for more details.

3.4 Data acquisition

3.4.1 Robot guided trajectory

Although robot-guided data set is available as a part of the pipeline, the important aspects will be discussed here. As shown in fig. 3.10 setup consists of 2 cameras (RGB-D and a tracking camera), a robot, a phantom head with EEG cap. The Kinect is mounted directly on the end-effector of the robot while the tracking camera is rigidly mounted to the ceiling (Cambar B2 is shown in the graphic instead of fusionTrac 500). The position of the tracking camera from robot base ${}^{\text{Base}}\mathbf{T}_{\text{tCam}}$ is determined using hand-eye calibration. Since the Kinect is mounted directly on the end-effector, a variant form of hand-eye calibration known as eye-in-hand calibration is performed to calculate the position of Kinect from robot base ${}^{\text{Base}}\mathbf{T}_{\text{kCam}}$. First, all the electrodes are digitized using

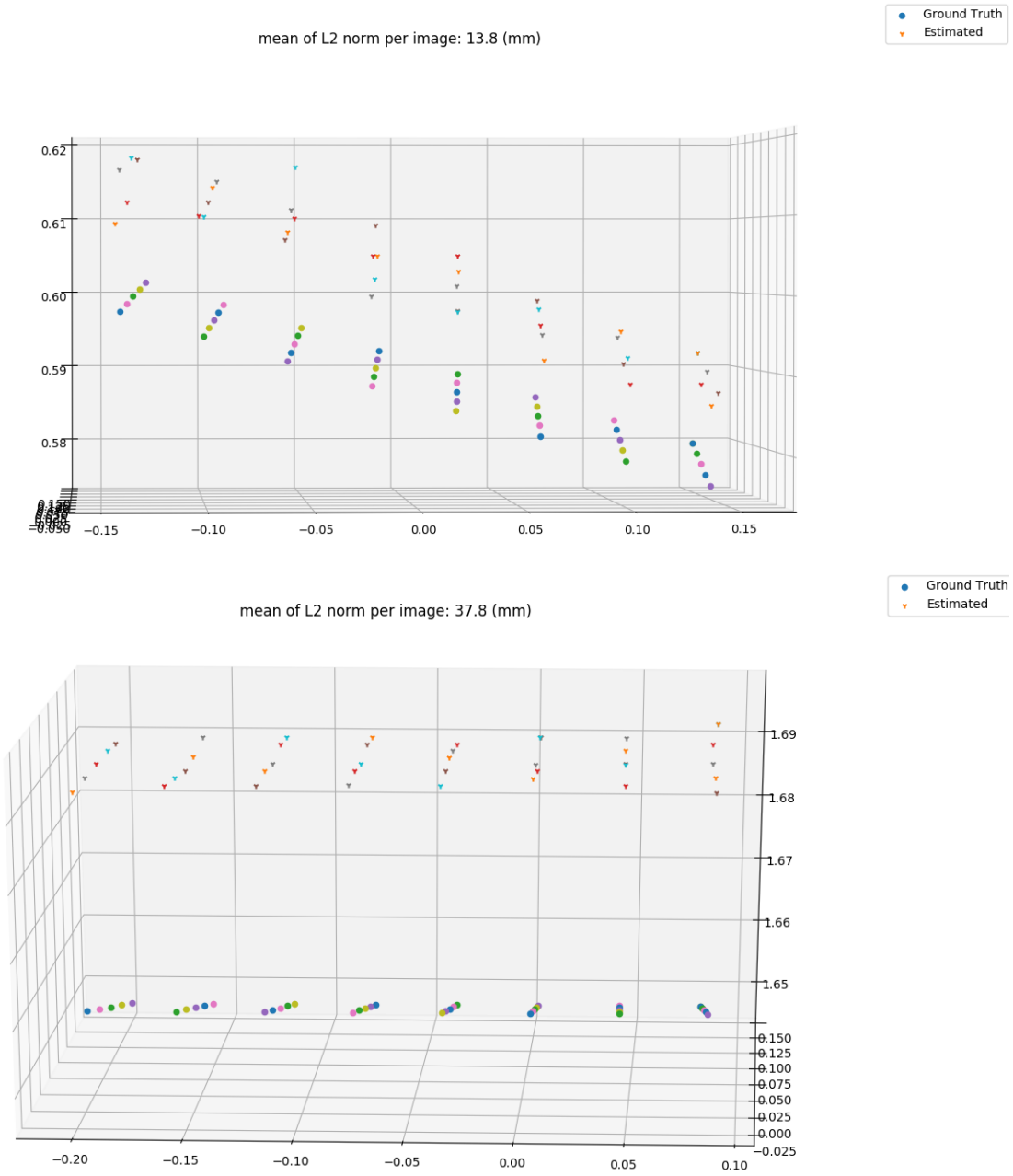

Fig. 3.9: Out-projection error between ground truth chessboard corners and estimated using raw kinect depth at 600 and 1600mm from the camera. Vertical axis the distance from the camera.

a reflective marker and fusionTrac. Then the robot is guided around the phantom head on a circular trajectory while RGB and depth images are captured using the point and shoot method. The process is repeated several times with the same circular trajectory each time with a re-positioned phantom head.

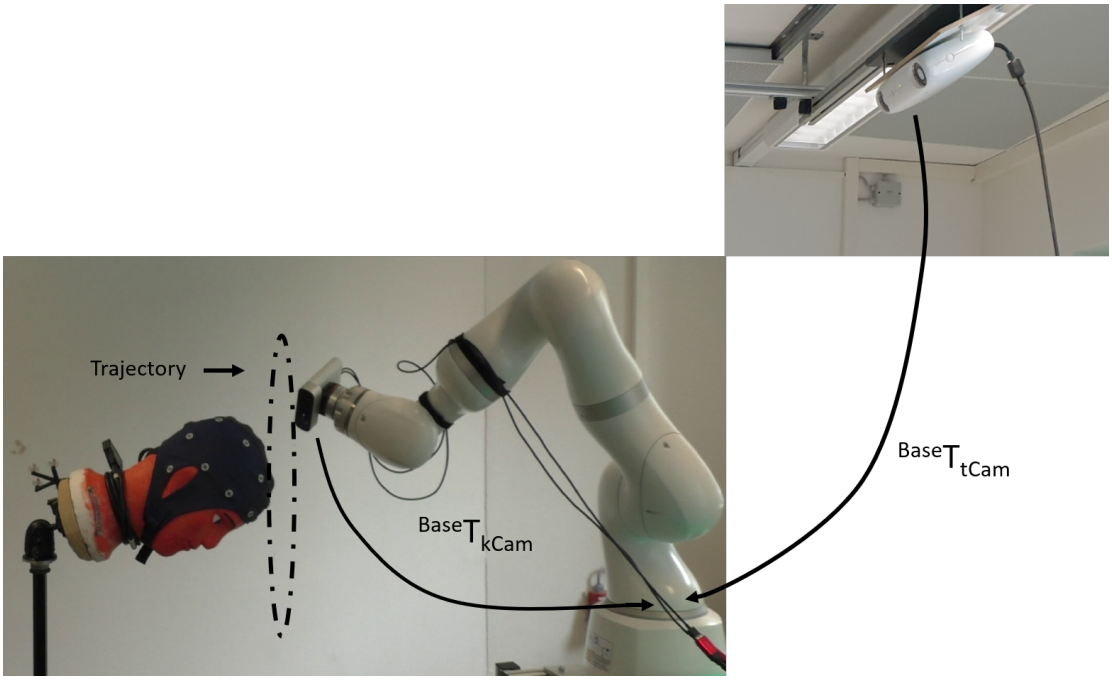

Fig. 3.10: Robot-guided data acquisition system.

3.4.2 Hand guided trajectory

As shown in fig. 3.11 setup consists of a tracking camera and an RGB-D camera as in the robot-guided setup. However, a reflective marker is attached to Kinect in order to be tracked from the tracking camera in the absence of a robot. The Kinect is hand-guided around the phantom head trying to maintain a circular trajectory while continuously capturing RGB and depth images at 30fps. The process is repeated several times with different trajectories each time and recorded as ROS .bag files for later processing.

One of the limitations during hand-guided data acquisition is the limited field of view (FOV) of the tracking camera. fig. 3.12 shows an approximate radius of the phantom head is $R \approx 70$mm and recommended distance for depth is 250 - 2210 mm for azure Kinect in $WFOV\ 2x2\ binned$ mode. Hence the hand-guided radius is maintained $>$ 350mm. However, due to the limited FOV of the tracking camera, only a portion of the trajectory is recorded. A top view of an example trajectory is shown in fig. 3.13 where it starts from 1 and ends at 6. The segments 1-2, 3-4 and 5-6 are seen while 2-3, 4-5, 6-1 are not, $nav\_msgs/Path$ message type from ROS automatically fills these gaps with straight lines. One of the consequences is that full trajectory is not available for comparison. Therefore, only visible segments are used for trajectory comparison and other metrics.

3.4.3 Eye-in-hand calibration for hand guided trajectory

In this section, we provide details to estimate the transformation between the reflective maker and the Kinect coordinate system. The only difference between the setup presented in the general hand-eye calibration and the hand-guided system is the absence of a robot. however, the eq. (3.1) is still holds with ${}^{\mathrm{tCam}}\mathrm{T}_{\mathrm{marker}}$ as M, ${}^{\mathrm{marker}}\mathrm{T}_{\mathrm{kCam}}$ as X, ${}^{\mathrm{tCam}}\mathrm{T}_{\mathrm{CB}}$ as Y and ${}^{\mathrm{CB}}\mathrm{T}_{\mathrm{kCam}}$ as N as shown in fig. 3.14.

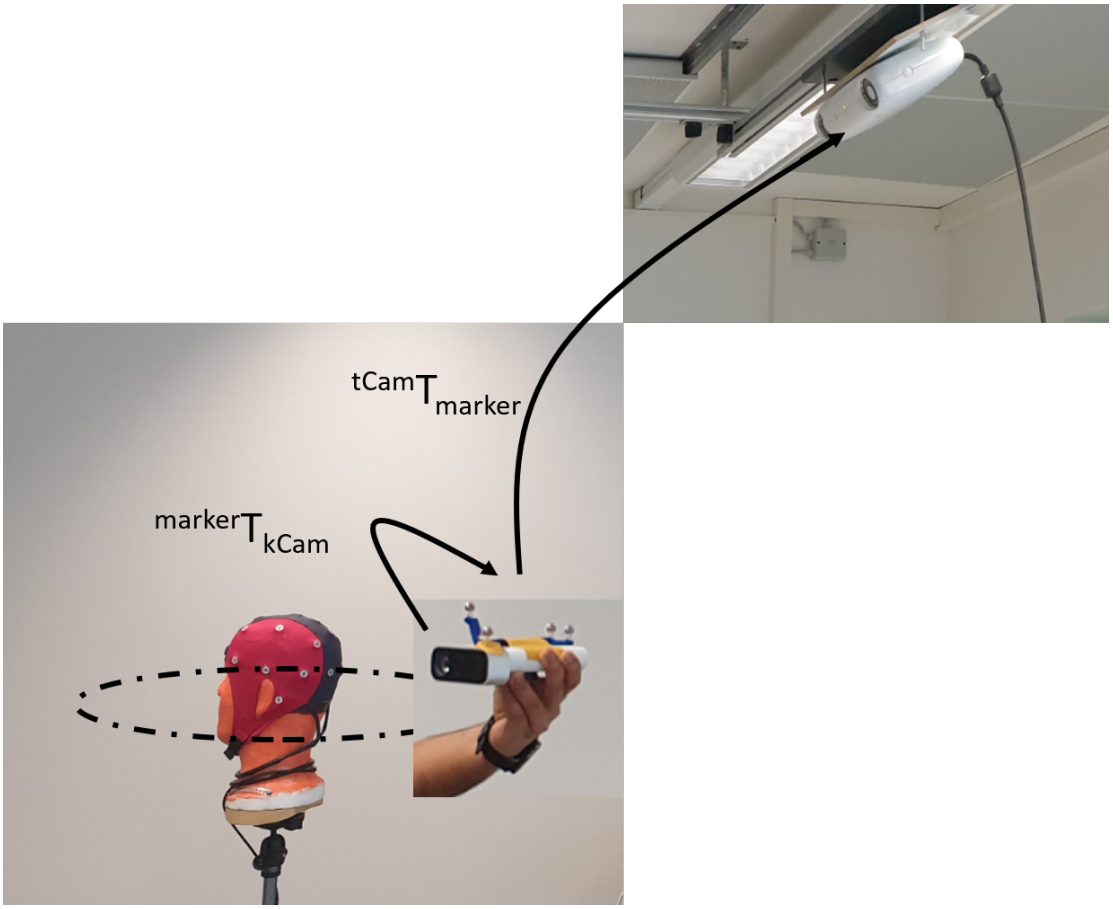

Fig. 3.11: Camera is hand guided around the phantom head.

$$\mathtt{M}_i\mathtt{X} - \mathtt{Y}\mathtt{N}_i = 0 \tag{3.1}$$

The tracking camera and chessboard were static while kinect moved to different position M_i and corresponding chessboard pose is estimated N_i.

3.4.4 Time synchronization

The Cambar B2 communication server, pose tracking software, and ROS, Kinect software run on two separate computers. The cambar B2 software provides a time stamp in milliseconds relative to starting time of the camera (does not use system time as basis) with alternating 62.5 and 66.66 fps while Kinect data is recorded at 30 fps with ROS timestamps as shown in fig. 3.15. Although data acquisition on both computers was manually started together, time delay from pressing the start button to actual starting is expected. As both cameras do not have a common time as a reference, the first entry of both cameras my not corresponds to each other. Therefore, time synchronization is necessary. The first task is to determine which Cambar entry corresponds to the first entry of Kinect. This may prompt to skip the first few entries from the Cambar data. Once the correct entry is found, all the other time stamps will be expressed relative to just found entry. Then, at time t, time stamps both Kinect and Cambar will be

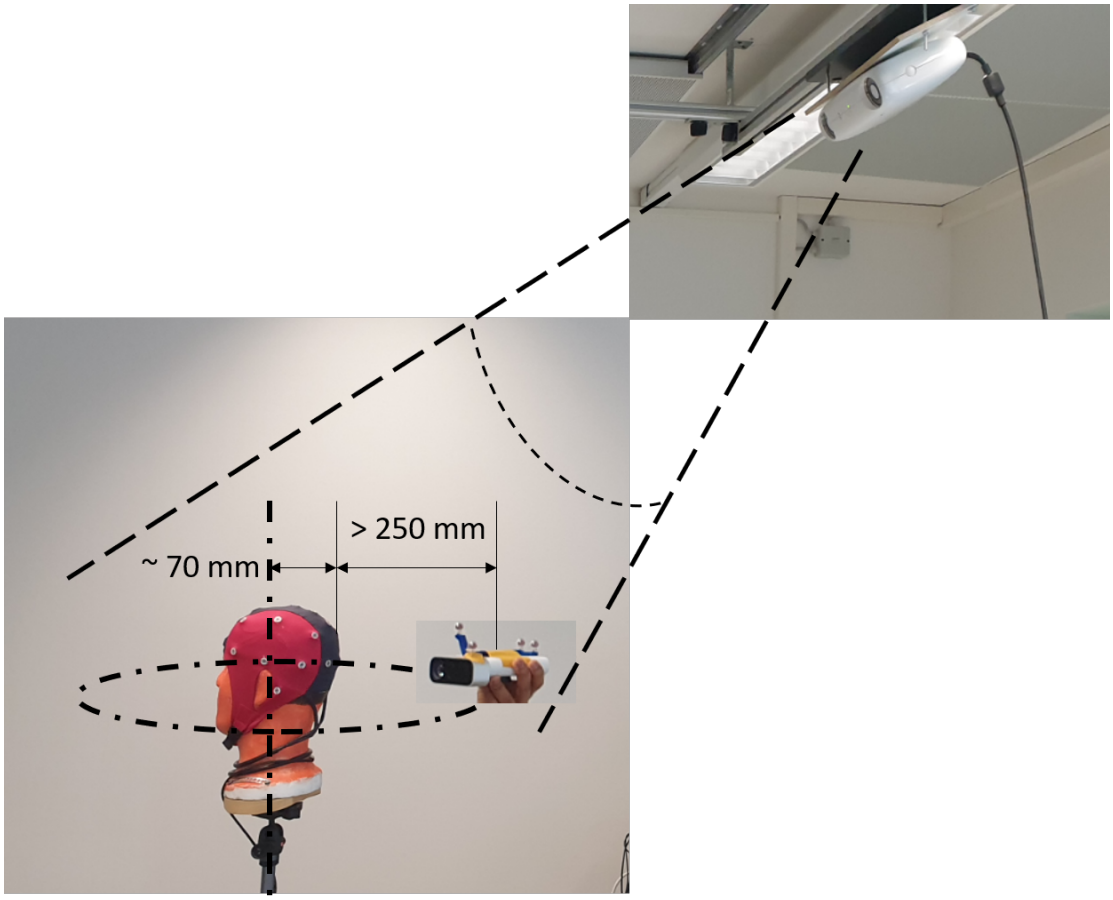

Fig. 3.12: Hand-guided radius is > 350mm.

matched. If the Kinect pose recorded in Cambar at the exact time is not available, the pose closest to time t is chosen. In order to choose the correct first entry in Cambar data, we first map the point clouds using Kinect poses and iterate over first 60 frames (assuming time delay is not more than 1 second), at each frame, the cluster centers, least value of cluster inertia as per eq. (2.36)(sum of distance squared) along with the corresponding entry in the Cambar is stored. Since the change in Kinect raw depth value also affects cluster inertia, later two optimizations will be performed together (details to follow). The fig. 3.16 shows an extreme case where the first 48 frames had to be skipped to achieve the desired goal.

3.5 Point cloud for scan registration

As previously discussed, point cloud for scan registration is generated by combining 2D pixel values from YOLO detection and depth value from Kinect. The middle point of the YOLO bounding box is taken as pixel value and corresponding depth value is retrieved to form a 3D point cloud. Only a few electrodes are visible at each camera position and the same electrode can be seen from multiple adjacent images. Therefore slight variations on the bounding box position, thereby on the middle point of the bounding boxes is expected. There are also cases where electrodes are not detected in any of the

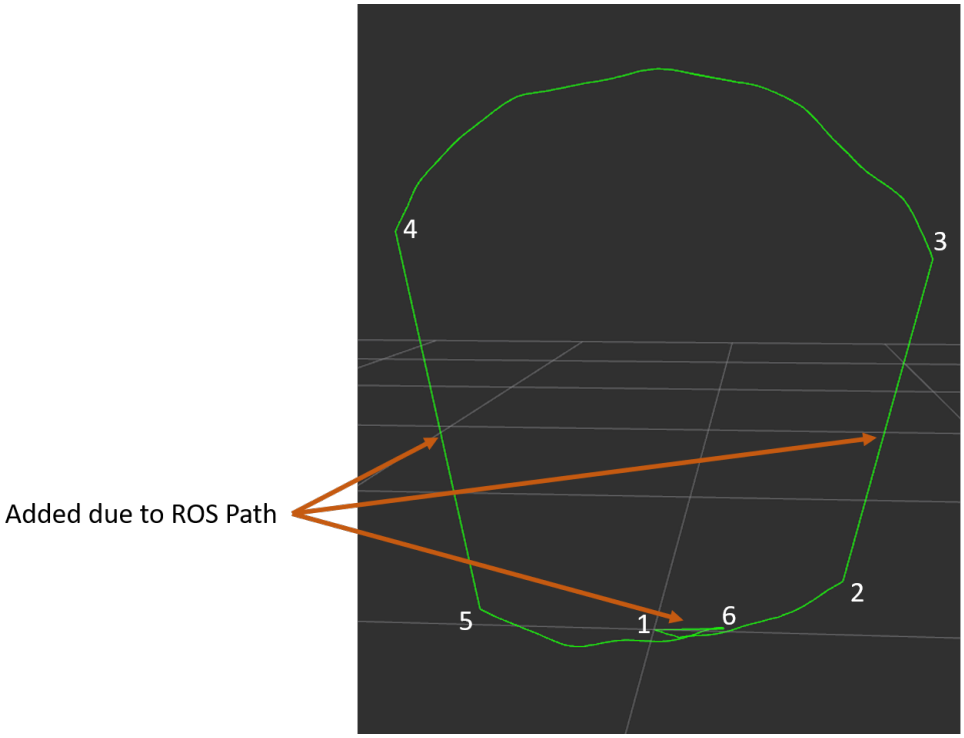

Fig. 3.13: An example for ground truth hand guided trajectory (1-6).

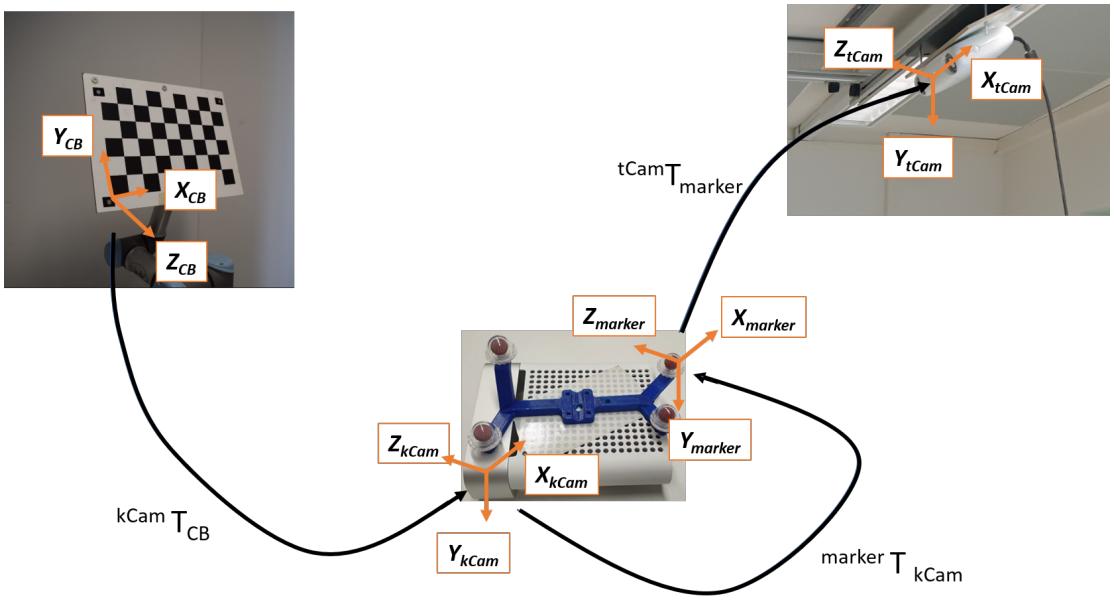

Fig. 3.14: Eye-in-hand calibration for hand guided setup. Coordinate frames are arbitrarily shown for illustration purposes only.

camera frames by YOLO and few false positives as shown in fig. 3.17.

Kinect Data		Cambar Data		Cambar Data	
time_stamp (milli sec)	Point Cloud	time_stamp (milli sec)	Marker Pose	time_stamp (milli sec)	Marker Pose
0	PC1	356349	Pose1	0	Pose1
33.33	PC2	356365	Pose2	16	Pose2
66.66	PC3	356380	Pose3	31	Pose3
99.99	PC4	356396	Pose4	47	Pose4
133.32	PC5	356411	Pose5	62	Pose5
166.65	PC6	356427	Pose6	78	Pose6
199.98	PC7	356443	Pose7	94	Pose7
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

Fig. 3.15: left: Kinect data consists of time stamp and point cloud, center : Cambar data consists of time stamp and pose of a marker attached to Kinect (hence pose of Kinect), right: Cambar time stamps relative to first entry.

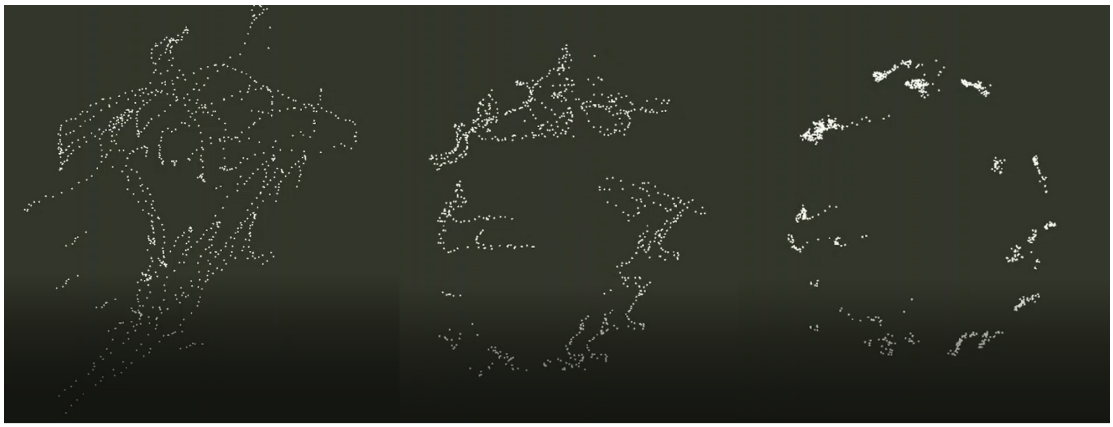

Fig. 3.16: Time synchronization using raw Kinect depth value.

3.6 Ground truth data acquisition

A reflective marker is used to acquire the 3D position of all the electrodes. The marker tip is carefully placed on each electrode and the 3D position is recorded in the tracking camera. As the Kinect frame is chosen to be the single frame of reference for the entire project, points recorded in the tracking camera are then transferred to the Kinect frame via series of transformations. The associated systematic errors can be visualized in fig. 3.18. In robot-guided case, these systematic errors consist of two hand-eye calibration errors i.e between tracking camera-robot base and Kinect-robot base. For the hand guided case one eye-in-hand calibration error between Kinect-marker. When out-projected points are compared with ground truth values, there will be an additional out-projection error.

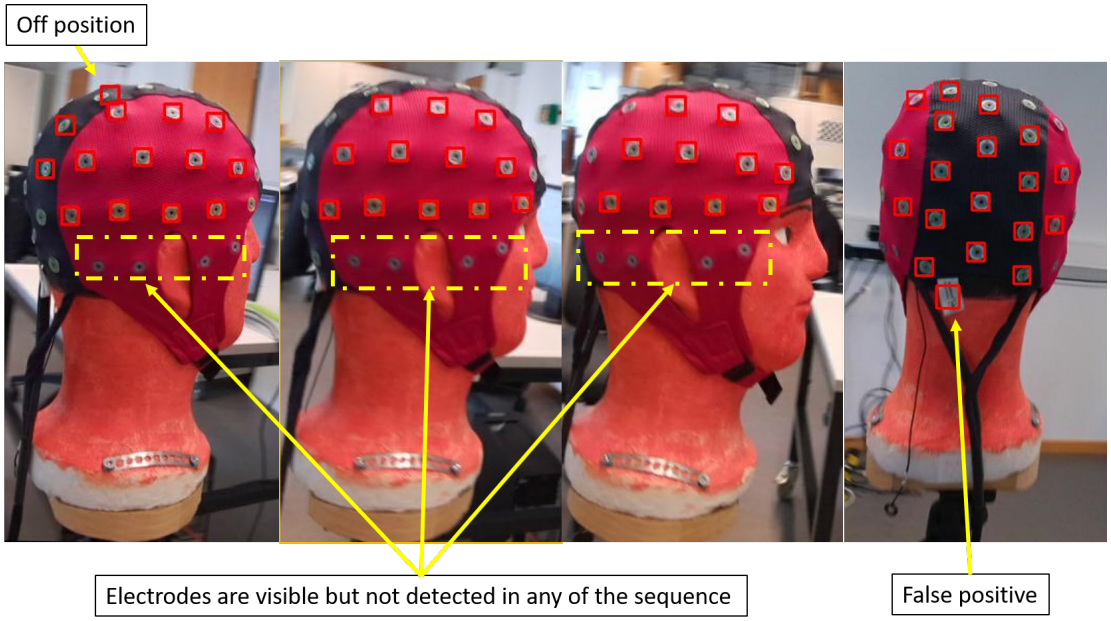

Fig. 3.17: Examples of undetected, false positives and bounding box off position.

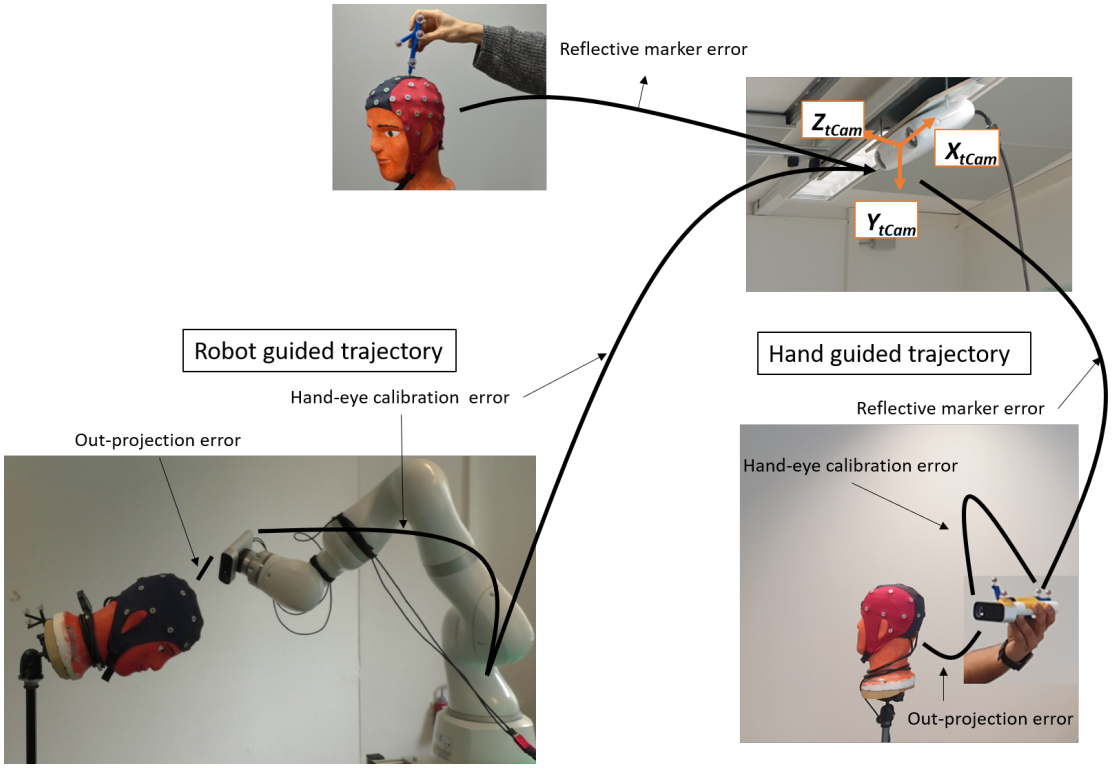

Fig. 3.18: Systematic error in acquiring ground truth data.

3.6.1 Out-projection error determination

One way to understand the out-projection error is to directly compare the out-projected and ground truth values and by measuring the difference in them. It is already known

that chessboard out-projection using raw depth value led to over estimation of the object points depth value. Therefore, we intend to find the parameter β that minimizes the error between ground truth and out-projected values with [raw depth - β]. This can be addressed in two ways, as shown in fig. 3.19. First, at each camera position $x_{1:t}$ both ground truth and out-projected point clouds are known (only those electrodes that are visible from a particular camera pose) and an iterative closest point registration can be performed and output root mean squared error (RMSE) can be used as a metric and averaged over all camera poses. Second, one can stitch the point clouds and find the cluster centers as we did before and compare them to the ground truth values and measure L2 norm.

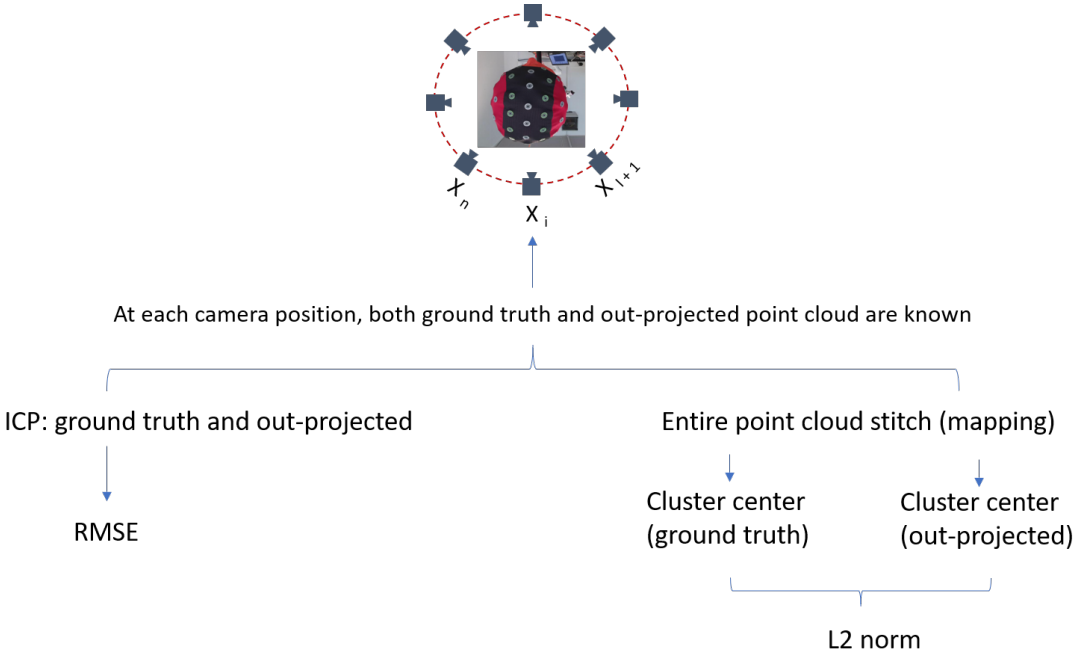

Fig. 3.19: Depth offset β determination process.

3.7 Pose-graph construction and optimization

In this section, the unique data structure used to store and provide easy access to the relevant information throughout the electrode digitalization process will be discussed and then an overview of pose-graph construction and optimization is provided. The fig. 3.20 shows a dictionary, each camera position has an associated number and stored in "$viewID$" or "$scanID$". The "$absTform$" or Absolute position is relative to the world coordinate frame, in this thesis, the very first camera position is treated as the world frame. 3D point cloud associated with each camera scan is stored as "$pointCloud$" in the dictionary. The system consists of 2 main layers and previously acquired scans as shown in fig. 3.21. At the very first scan, Pose-graph is initialized with a prior factor (black square) which represents the initial camera position relative to the world frame. The node X_1 with "$viewID$" $= 0$ "$absTform$" $= identity(4 \times 4)$ and point cloud "$pointCloud$" $= PC_1$ associated with camera position C_1 is added to the pose-graph. A map is simultaneously being built each time a node is added to the pose-graph. Every

time a new scan is available, scan registration (ICP) finds the best transformation that aligns it with the previous scan. A new node X_2, between factor (odometry) i.e. output of ICP (gold squares), and the position of current camera frame relative to very first frame (world) are added to the pose-graph. When the system detects a loop closure (see section on loop closure) a loop factor between the current scan and loop closure candidate is added to the pose-graph and optimization is triggered.

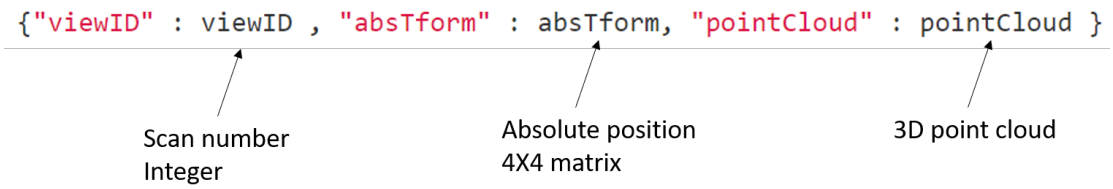

Fig. 3.20: Data structure to store and provide easy access to the information.

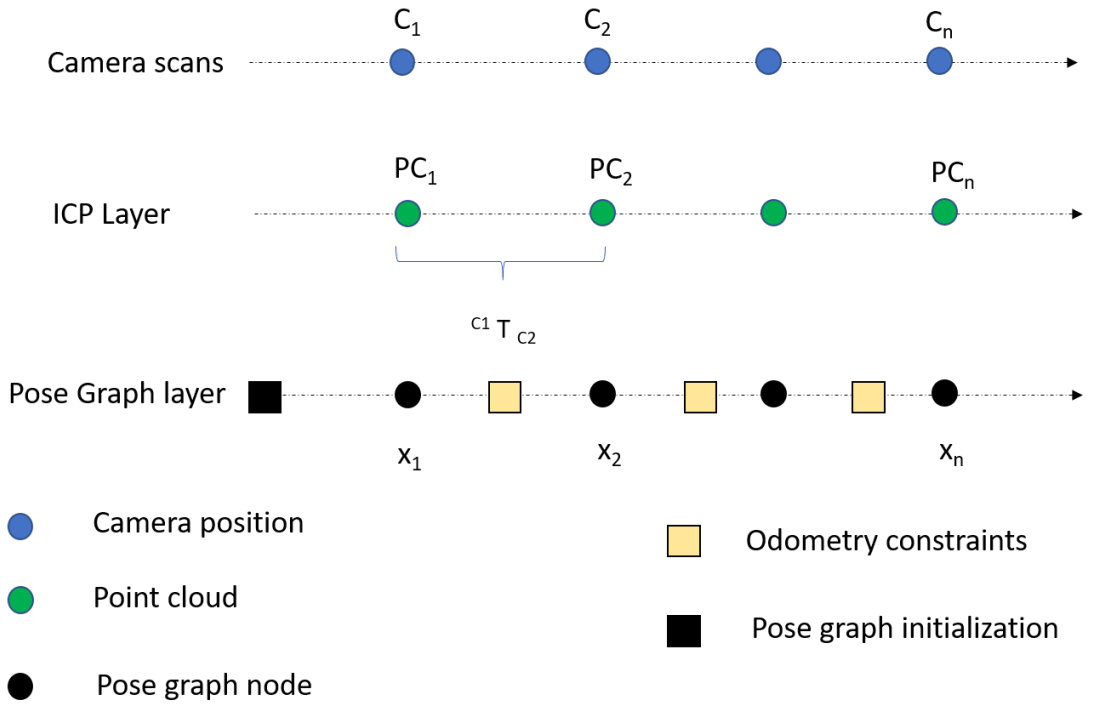

Fig. 3.21: System overview along with pose-graph layer. All the layers are shown on a straight line instead of circular trajectory just for the illustration purposes only

3.7.1 Loop closure

Every time the new node is added to the pose-graph, the system evaluates if the loop can be closed by computing the euclidean distance between current node $x_{current}$ and all other nodes that are already present in the pose-graph. Please note that the recent m number of scans are not considered as potential loop closure candidates thus avoiding

local loops and enabling longer loop creation. These recent m number of nodes are called *no loop closure window* . Hence we always start with the first node on the pose-graph until [$x_{current}$ - m] nodes. The fig. 3.22 illustrates the loop closure candidate selection process. We compute the 3D euclidean distance between the nodes as there is a high chance of elevation difference when visiting the previously visited places, especially with hand-guided trajectory. If a node x_i is under a certain norm threshold from the current node, a loop closure ICP is performed between the current node and x_i. All the nodes that are within the norm threshold and exceed certain loop closure ICP fitness threshold are considered potential loop closure candidates. The node with the highest ICP fitness value is considered as loop closure candidate. We check if this candidate has been already chosen before and closed loop successfully, in that case, we ignore this candidate and we start new iteration with $x_{current}$ +1. This is to avoid multiple loop closure for the same candidate because the subsequent camera frame is only few millimeters away and there is a high possibility that same electrodes are visible in both camera frames. After successful loop closure, a loop closure constraint (output of the loop closure ICP) is added to the pose-graph between the current node and loop closure candidate before the optimization is triggered. The map will be updated with the best estimates of the camera positions resulting from the optimization. Thanks to the data structure, updating the map is achieved by just replacing the absolute position with the optimized result.

Open3D defines ICP (source cloud, target cloud) fitness as eq. (3.2). The number of inlier correspondences usually one to many i.e. one point in the target point cloud can have multiple correspondences in the source point cloud. However, in this case, it is one to one due to the sparsity of point cloud. Therefore ICP fitness value for example 0.8 means 8 out of 10 points in the target point cloud have one to one correspondences with source point cloud.

$$ICPfitness = \text{ \# of inlier correspondences / \# of points in target point cloud} \quad (3.2)$$

3.7.2 Optimization

The popular non-linear optimization algorithm Lavenberg-Marquardt offered by MiniSam is used for pose-graph optimization. By using the optimized camera trajectory, the electrode positions from each frame can be transformed into a common coordinate system. For this purpose, the coordinate system of the very first frame (time $t = 0$) is used. After the transformation of all electrodes detected over the duration of trajectory into the initial coordinate system, individual electrodes which were detected in several frames do not lie exactly on top of each other. This is due to errors in the detection and error in out-projection etc. As a result, clusters of electrodes appear in the 3D space. The last step of electrode digitalization is calculating cluster centers. First, the number of clusters is determined by the algorithm DBSCAN. Then, the K-Means algorithm is applied to determine the centroids of each cluster as shown in fig. 3.23.

3.8 Performance metrics

This section provides an overview of metrics used to evaluate the performance of algorithms. Firstly, we compute the absolute 3D position error between estimated and

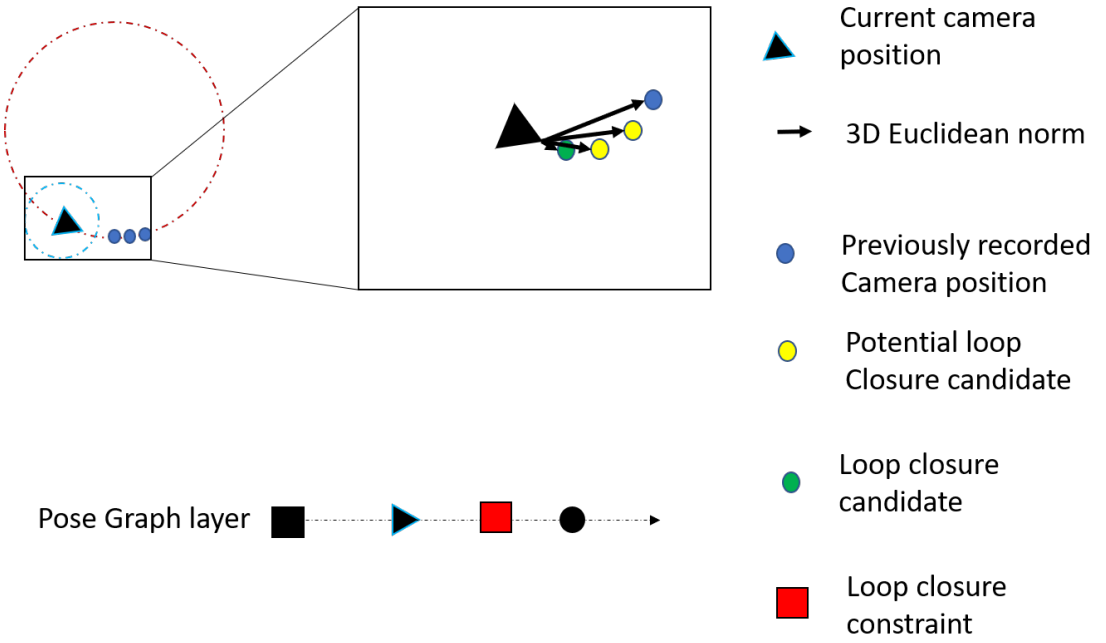

Fig. 3.22: Loop closure candidate selection process.

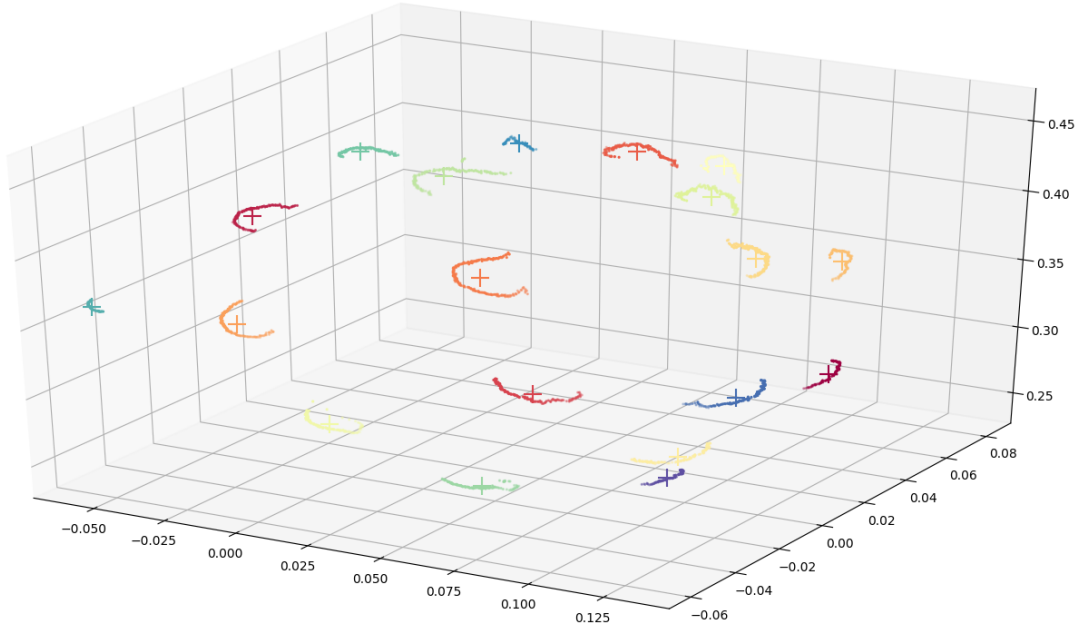

Fig. 3.23: Clusters along with centers (plus sign) are calculated via K-Means analysis. Each cluster is plotted with different colors for visualization.

ground truth electrodes. As we saw earlier that there are series of errors involved in gathering ground truth data and transferring it to Kinect for frame error computation. Therefore, we also perform an ICP registration between estimated and ground truth electrodes. Quite an insight can be obtained by comparing the estimated and ground truth trajectory (see appendix : pose comparison). As previously mentioned, all the

camera poses are expressed with respect to the first camera pose (which is also the global reference frame) before we compare the trajectory as shown in fig. 3.24. Lastly, we compute the difference between ICP estimated transformation for subsequent camera poses and ground truth poses. As shown in fig. 3.25, we skip one pose in order to have sufficient distance between 2 camera pose (>4mm @ 30fps).

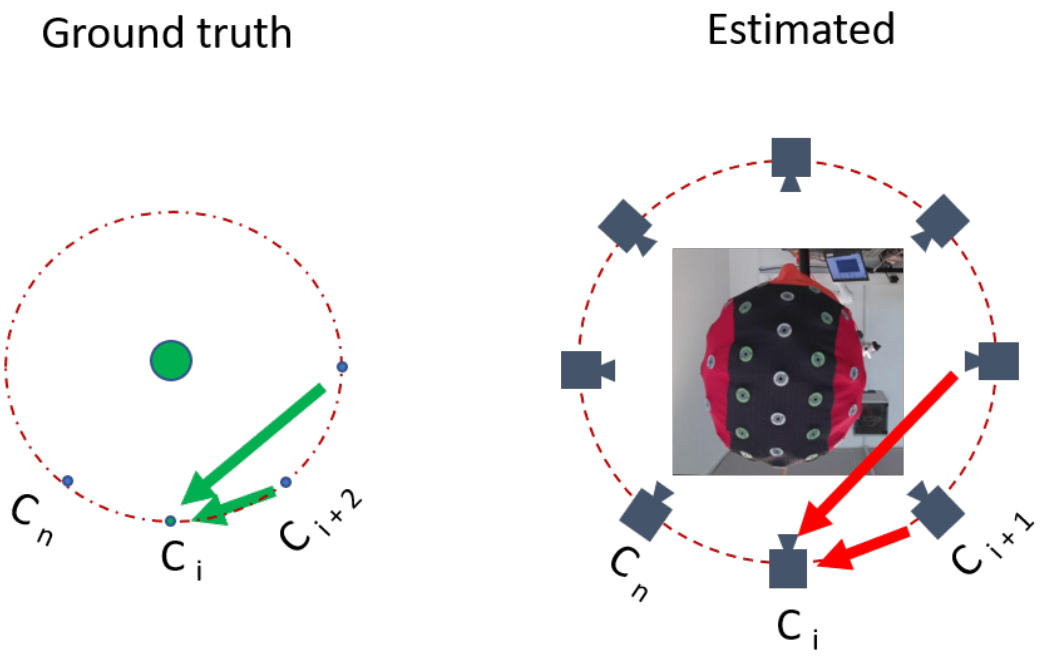

Fig. 3.24: Left: an example of ground truth trajectory (robot guided/hand guided). Right: estimated trajectory via SLAM. All the poses are expressed relative to first frame. Fewer poses are shown for illustration purpose.

Summary of all metrics.

1. Absolute 3D Position error
   a) $L2Norm_i = \sqrt{(x_{gi} - x_{ei})^2 + (y_{gi} - y_{ei})^2 + (z_{gi} - z_{ei})^2}$
   b) L2 Norm list $=[L2Norm_i, ..., L2Norm_N]^T$
   c) where gi and ei are ground truth and estimated electrodes respectively. N is total number of electrodes in the cap.
   d) Mean +/- STD Deviation of L2 Norm list are reported.
2. Relative 3D position error
   a) Post registration RMSE = ICP (Estimated electrode cloud, ground truth electrode cloud)
3. Trajectory comparison
   a) Rotation angle (axis-angle representation) Mean +/- STD Deviation
   b) Translation (L2 norm) Mean +/- STD Deviation

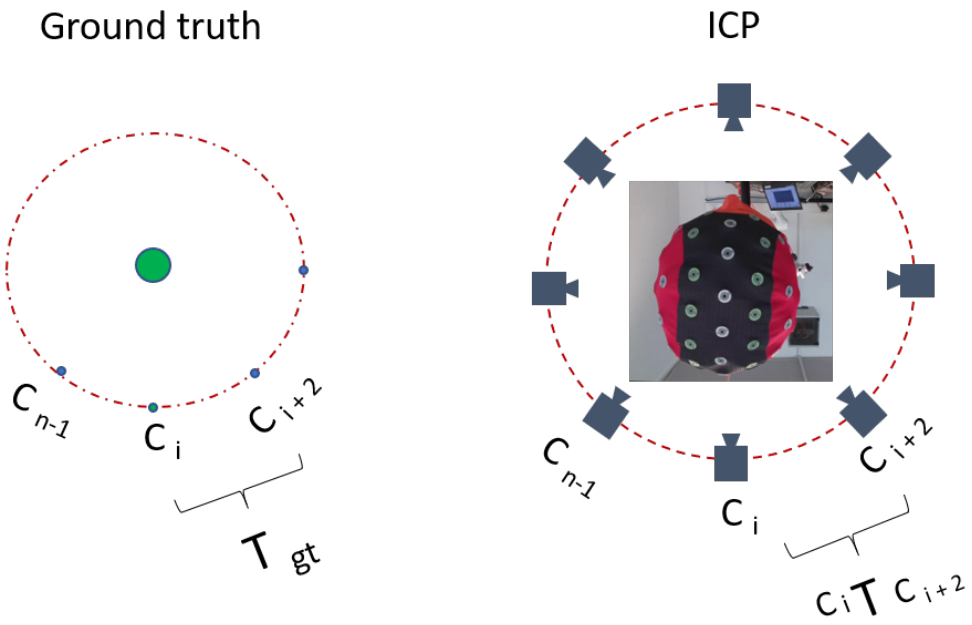

Fig. 3.25: Left: an example of ground truth trajectory (robot guided/hand guided). Right: estimated trajectory via SLAM. Relative transformation between successive frames of ground truth trajectory is being compared with ICP estimation. Fewer poses are shown for illustration purpose.

4. ICP vs ground truth comparison
   a) Rotation angle (axis-angle representation) Mean +/- STD Deviation
   b) Translation (L2 norm) Mean +/- STD Deviation

4 Results

This section provides detailed results on various aspects of the thesis.

4.1 Camera calibration

Several images at different depths in steps of ≈ 200mm starting ≈ 400mm from the camera were used for camera calibration. The estimated camera intrinsic parameters using OpenCV, Matlab along with values in the robot-guided data set is provided in tab. 4.1. Image size 1280×720.

Tab. 4.1: Microsoft azure kinect calibration result

Parameters	Calibrated values (OpenCV)	Calibrated values (Matlab)	Value in robot-guided data set
α_x	601.06	604.58	607.49
α_y	601.33	604.48	607.44
S	0.0	0.0	0.0
C_x	636.54	639.25	638.92
C_y	360.29	358.54	364.35

4.2 Hand-eye and Eye-in-hand calibration

For eye-in-hand calibration (hand-guided case) the Kinect along with firmly attached marker was moved to >40 different positions making sure that the tracking camera can detect the marker and estimate the pose correctly. The collected data set is divided into training (60%) and evaluation sets (40%) randomly. The errors are calculated based on the evaluation data set and (Mean±STD Deviation) separately for translation and rotation is reported. The tab. 4.2 shows errors for robot-guided and hand-guided case.

Tab. 4.2: Hand-eye, Eye-in-hand calibration results

Camera	Translation error (mm)	Rotational error (deg)	Between
Kinect	8.8± 0.2	0.7682 ± 0.0204	Marker-Kinect
Tracking camera	$0.18\pm 4.5 \times 10^{-6}$	$0.093 \pm 5.4 \times 10^{-5}$	Robot-Tracking Camera
Kinect	$1.28\pm 2.1 \times 10^{-4}$	$0.211 \pm 2.1 \times 10^{-4}$	Robot-Kinect

4.3 Pixel points out-projection

For the pixel points out-projection experiment, several images of the chessboard at different depths in steps of ≈ 200mm starting ≈ 400mm from the camera and at each depth, different orientations were captured. Firstly, ground truth object points are calculated in camera coordinates and corresponding image points are out-projected using

raw depth provided by Kinect. For each image, out-projected values are then compared to the ground truth values with L2 norm as the metric. Since there are many images at a particular depth distance, L2 norm will be averaged over all the images. The fig. 4.1 shows the results.

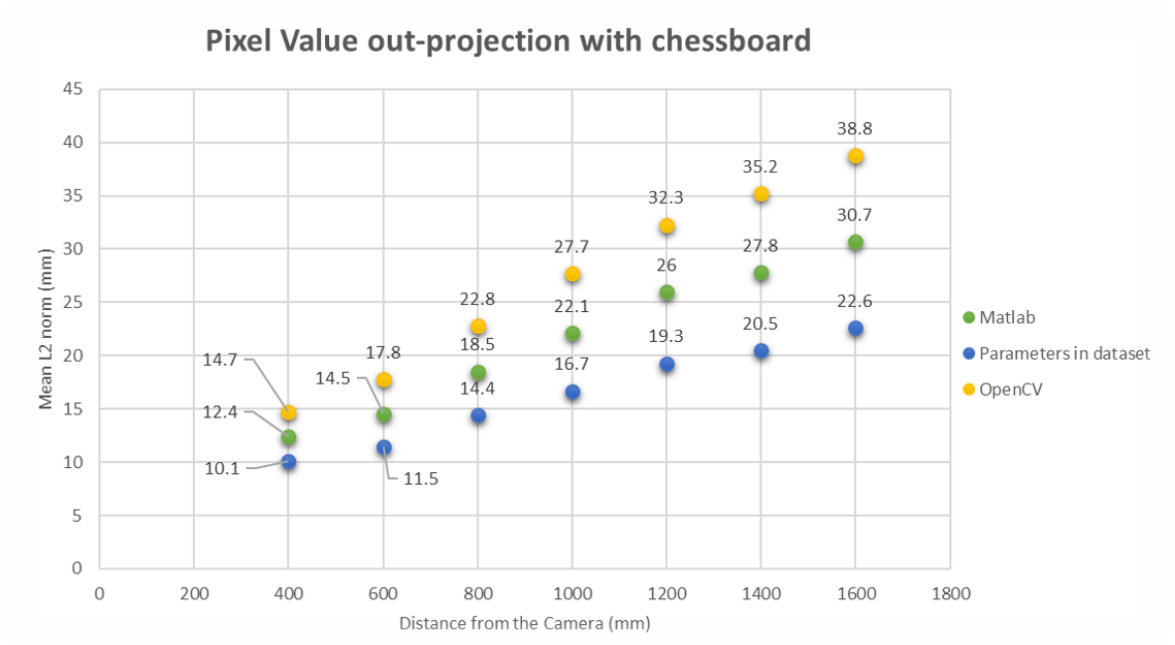

Fig. 4.1: Overview of mean L2 norm with different distance from the camera.

4.4 Depth offset determination

For the robot-guided case, first at each camera position $x_{1:t}$ both ground truth and out-projected point clouds are known (only those electrodes that are visible from a particular camera pose) and an iterative closest point registration can be performed and output root mean squared error (RMSE) can be used as a metric and averaged over all camera poses. Second, one can stitch the point clouds and find the cluster centers and compare them to the ground truth values and measure L2 norm. For the ICP scan registration, β is varied from 10 to 29 mm (i.e. out-projected points will be raw Kinect depth - β) while performing grid search with a minimum grid size of 1 mm. The fig. 4.2 shows result for 6 robot-guided trajectories (2/cap, for 3 caps). Similarly, with the same β variation for cluster center comparison, fig. 4.3 shows the results.

K-Means aims to determine the centroids μ_j that minimizes the inertia defined by eq. (2.36). The effect of depth offset can also be seen in inertia estimated by k-means. The fig. 4.4 shows inertia at different $\beta > 20$ values. Effect of β on the point cloud can be visualized in fig. 4.5.

For the hand-guided case, optimum β is found by combining time synchronization as it also affects the cluster inertia. Therefore, For the combined optimization, β is varied from 0 to 29 mm and skip is varied from 0 to 60. Where $\beta = 0$ refers to the direct use of Kinect raw depth value $skip = 0$ means the first Cambar entry corresponds to the first Kinect entry. At each iteration of β, we first the solve time synchronization

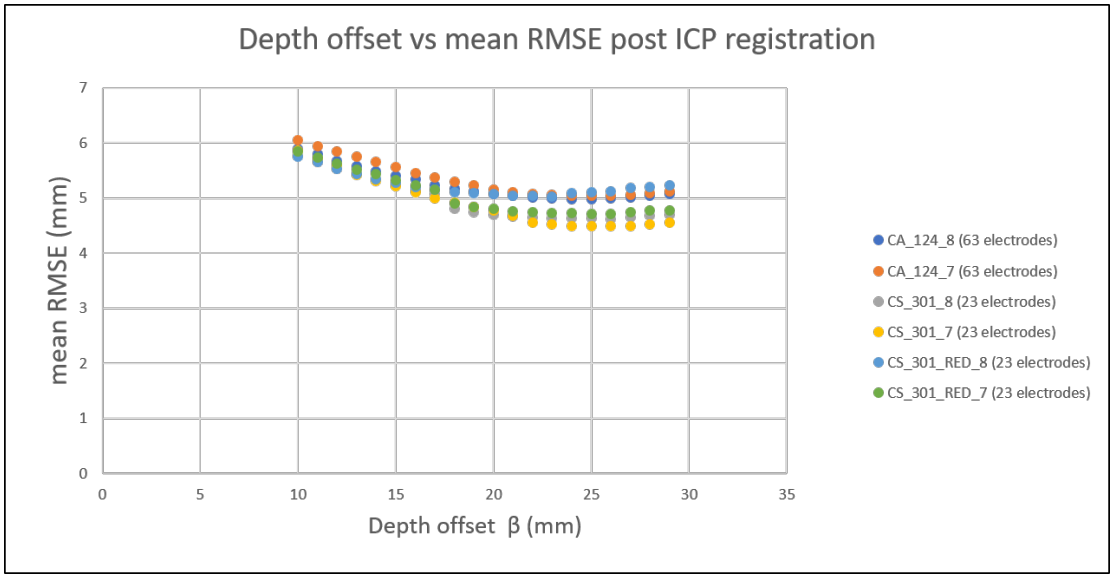

Fig. 4.2: mean RMSE vs depth offset β.

Tab. 4.3: Loop closure thresholds for robot-guided scenario

Use case	Euclidean Norm	ICP fitness threshold
robot-guided	100 (mm)	0.8

problem and note the lowest inertia and skip values. The fig. 4.6 shows the result of the combined optimization. The numbers in the parenthesis are the average skip value at which minimum inertia was achieved. On the other hand, higher error in static transform estimation between marker and Kinect led to the higher error in absolute 3D position of estimated cluster centers and ground truth electrodes. Therefore, one more ICP between them was necessary. The cluster center comparison result is shown in fig. 4.7.

4.5 Robot-guided trajectory

The euclidean norm and ICP fitness threshold used to select the loop closure candidate are shown in tab. 4.3. Forthcoming results are for 2 trajectories for each cap (2/cap, for 3 caps) at $\beta = 25mm$. Absolute and relative 3D position errors are calculated only for visible electrodes as some are not seen/detected in any of the frames, for example (55/63) means (average electrodes visible/total number of electrodes in the cap) as shown in tab. 4.4. Trajectory comparison and the ICP performance are shown in tab. 4.5 and tab. 4.6 respectively. The fig. 4.8 and fig. 4.10 summarizes the ICP and ICP + loop closure performance for one of the caps with 63 and 23 electrodes respectively. Clusters along with the 3D position of estimated cluster centers and ground truth after post registration are shown in fig. 4.9 and fig. 4.11 for one of the caps with 63 and 23 electrodes respectively.

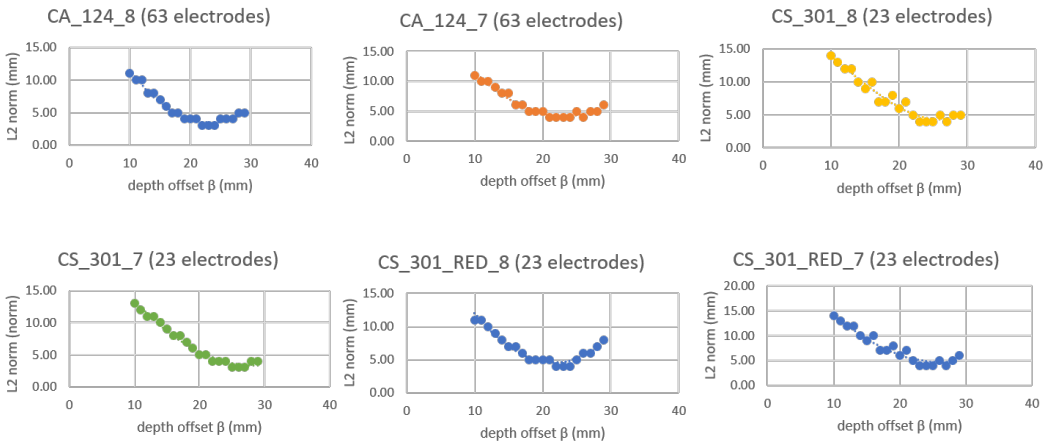

Fig. 4.3: mean L2 norm vs depth offset β.

Tab. 4.4: 3D position error

Metric	CAP_63 (55/63)	CAP_23_1 (20/23)	CAP_23_2 (19/23)
Absolute L2 norm	11.5 mm ± 5.0 mm	11.7 mm ± 4.0 mm	12.6 mm ± 5.5 mm
Post registration RMSE	3.7 mm ± 0.3	6.8 mm± 2.0 mm	5.6 mm ± 1.2 mm

4.6 Hand-guided trajectory

The euclidean norm and ICP fitness threshold used to select the loop closure candidate are shown in tab. 4.7. Forthcoming results are for 2 trajectories for each cap (2/cap, for 3 caps) at $\beta = 10mm$. High eye-in-hand calibration errors, i,e estimated static transformation between marker attached to the Kinect is not so accurate therefore only relative 3D position errors are shown in tab. 4.8. Trajectory comparison and ICP performance are shown in tab. 4.9 and tab. 4.10 respectively. The fig. 4.12 and fig. 4.14 shows the summary of ground truth and ICP estimated trajectories for caps with 63 and 23 electrodes respectively. For both images, the top-left section shows the front and side view of ground truth trajectory (green) recorded by attaching a marker. Sections 1-2, 3-4, and 5-6 are visible while sections 2-3, 4-5 are not. The top-right section shows front and side view of ICP estimated trajectory (red) which starts from 1 and ends at 2. The bottom-left section shows the front and side view of both ground truth and ICP estimated trajectory on top of each other for easy visualization. The bottom-right section on the other hand shows a closer look at starting and ending portion of each trajectory. Clusters along with the 3D position of estimated cluster centers and ground truth after post registration are shown in fig. 4.13 and fig. 4.15 for one of the caps with 63 and 23 electrodes respectively.

Tab. 4.5: Trajectory Comparison

Metric	CAP_63	CAP_23_1	CAP_23_2
Translation error	-1.4 mm ± 9.3 mm	16.3 mm ± 10.1 mm	12.1 mm ± 14.5 mm
Rotational error	4.1°± 1.7°	4.6°± 1.7°	6.7°± 3.4°

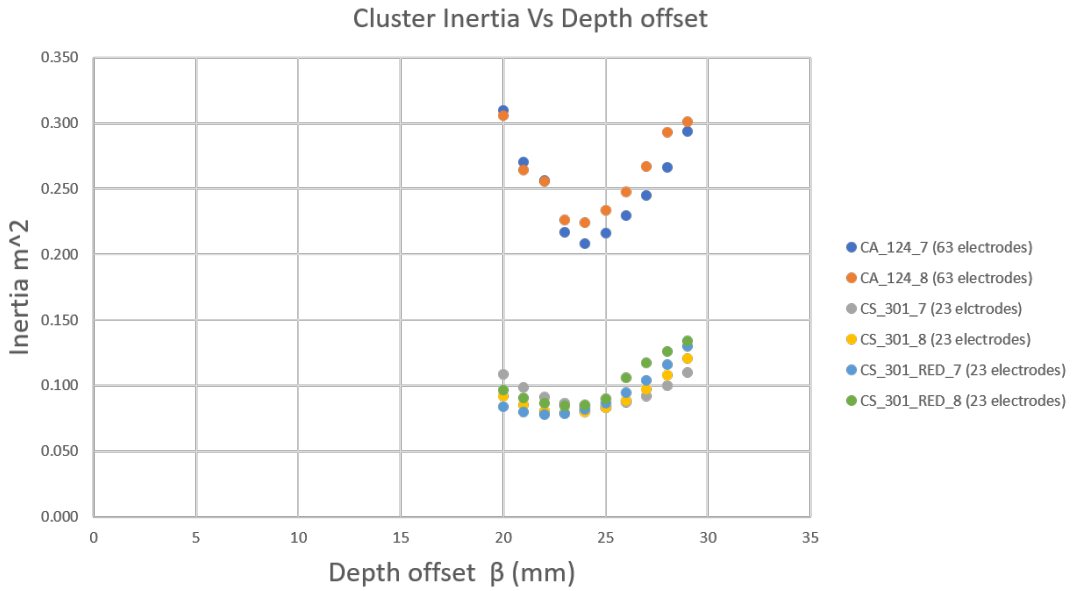

Fig. 4.4: Average cluster Inertia for each trajectory is calculated, note that, inertia is not a normalized metric. Lower the better and zero is optimum

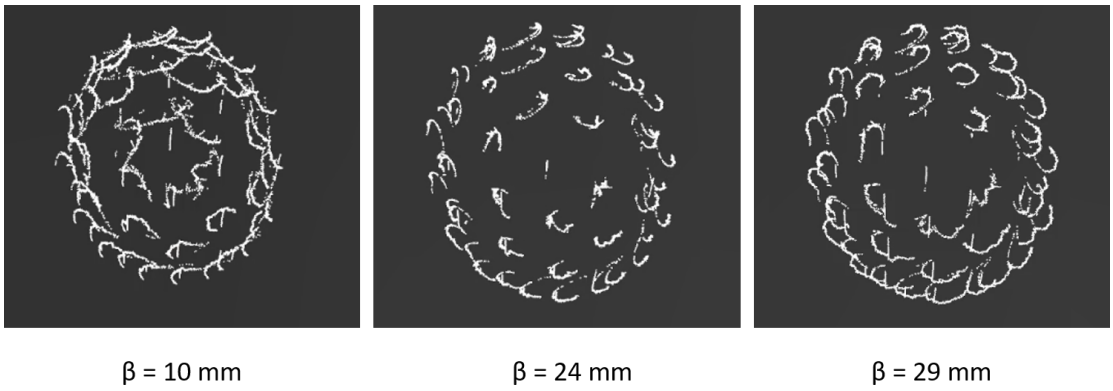

Fig. 4.5: Effect of change in β on point cloud.

Tab. 4.6: ICP vs ground truth comparison

Metric	CAP_63	CAP_23_1	CAP_23_2
Translation error	0.5 mm ± 2.1 mm	0.8 mm ± 2.9 mm	0.9 mm ± 3.3 mm
Rotational error	0.5°± 0.3°	0.7°± 0.4°	0.9°± 0.6°

Tab. 4.7: Loop closure thresholds for hand-guided scenario

use case	Euclidean Norm	ICP fitness threshold
hand-guided	100 (mm)	0.6

Tab. 4.8: 3D position error

Metric	CAP_63 (50/63)	CAP_23_1 (21/23)	CAP_23_2 (22/23)
Post registration RMSE	10.7 mm ± 2.7	14.6 mm ± 3.4 mm	19.3 mm ± 2.3 mm

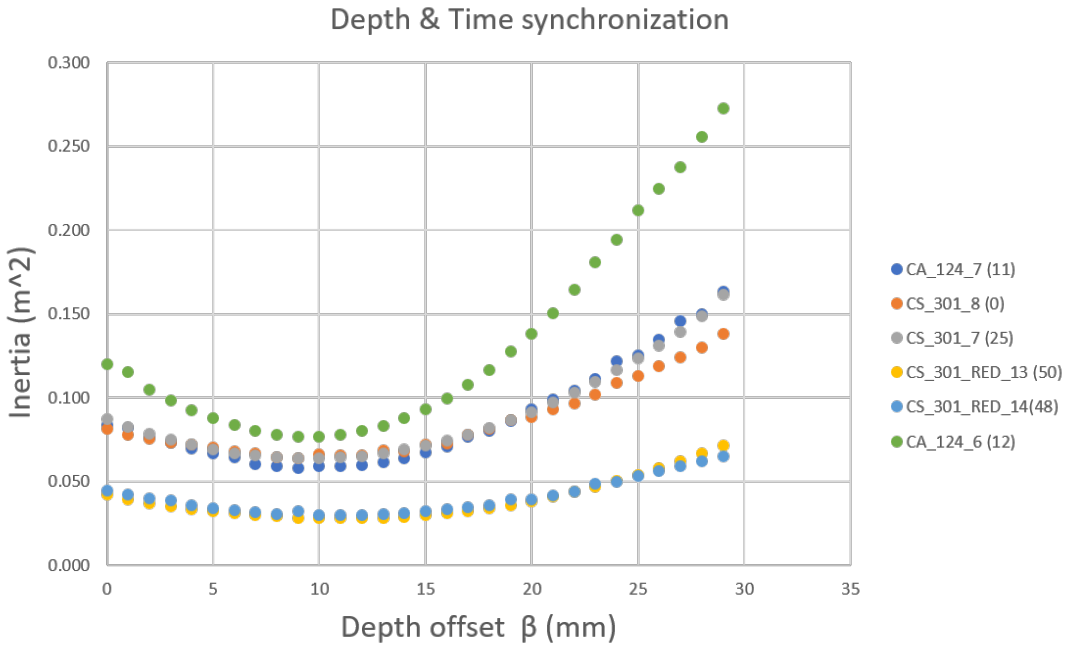

Fig. 4.6: Depth and time synchronization.

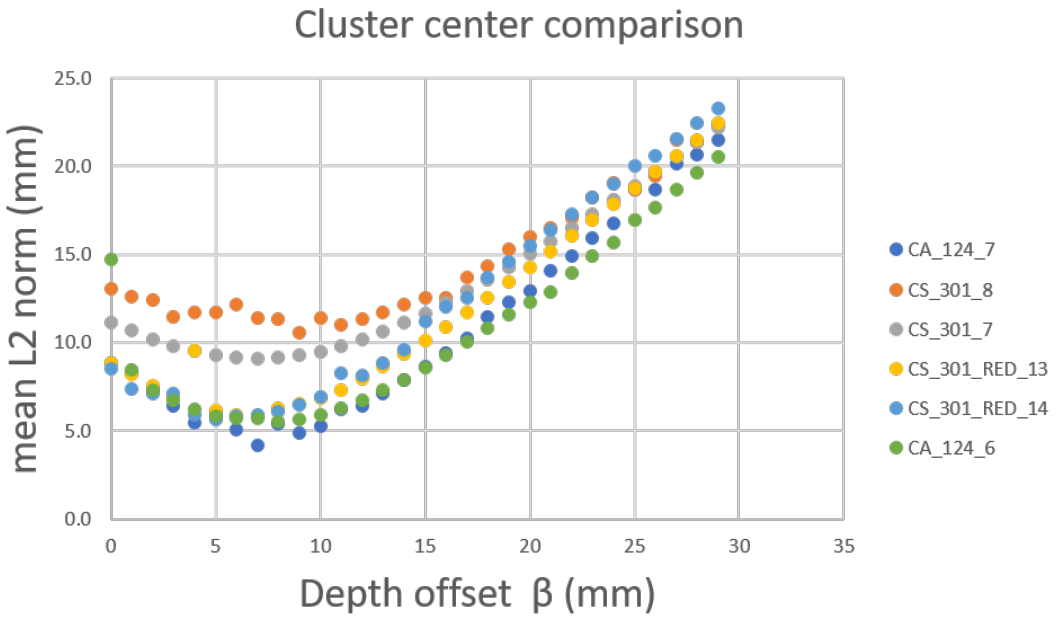

Fig. 4.7: mean L2 norm vs depth offset β.

Tab. 4.9: Trajectory Comparison

Metric	CAP_63	CAP_23_1	CAP_23_2
Translation error	-10.6 mm ± 50 mm	-15 mm ± 91 mm	-18.8 mm ± 60 mm
Rotational error	11°± 4°	16.8°± 11°	15.3°± 4°

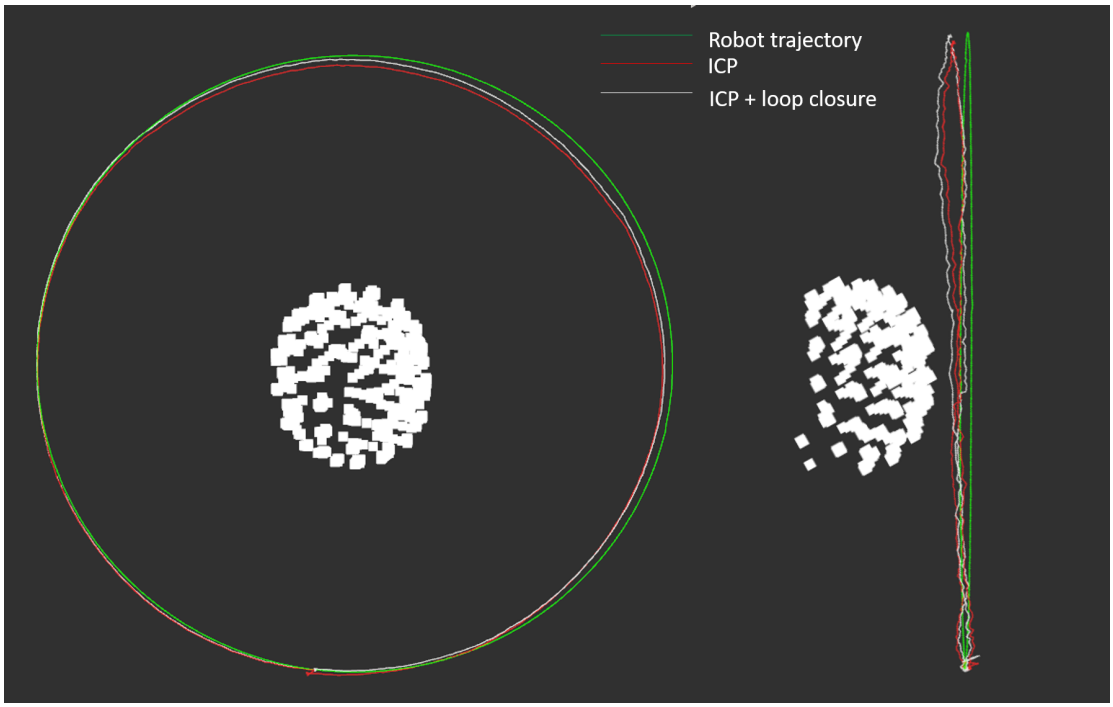

Fig. 4.8: Front and side view of trajectories for one of the cap with 63 electrodes. Green: Ground truth, Red: ICP alone, White: ICP+Loop closure,robot trajectory radius is 300mm.

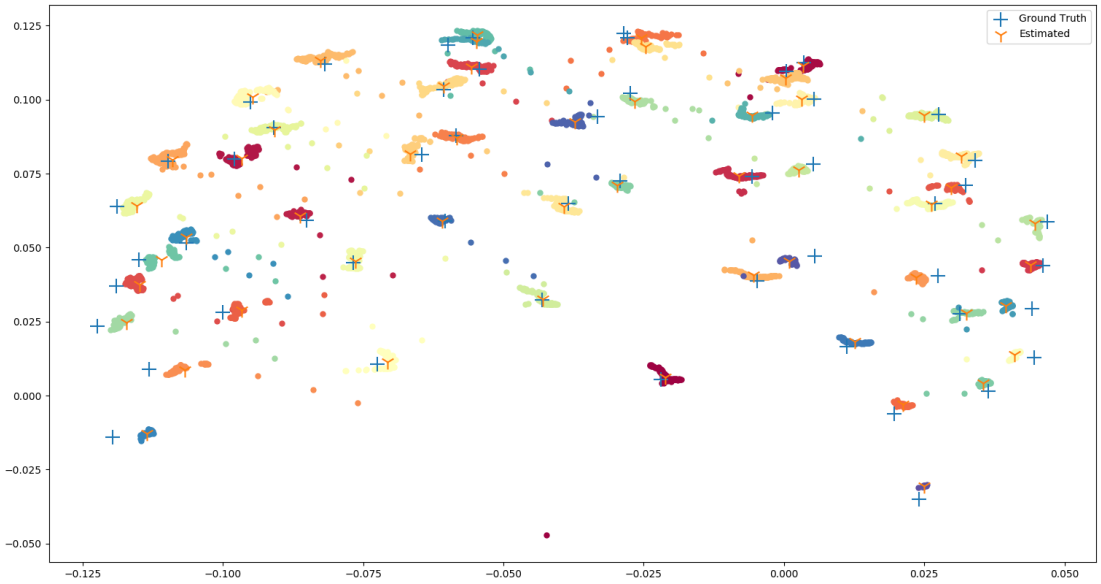

Fig. 4.9: Post registration results, Mean L2 norm : 3.06 (mm), Plus sign : ground truth , *tri_down* sign : estimated

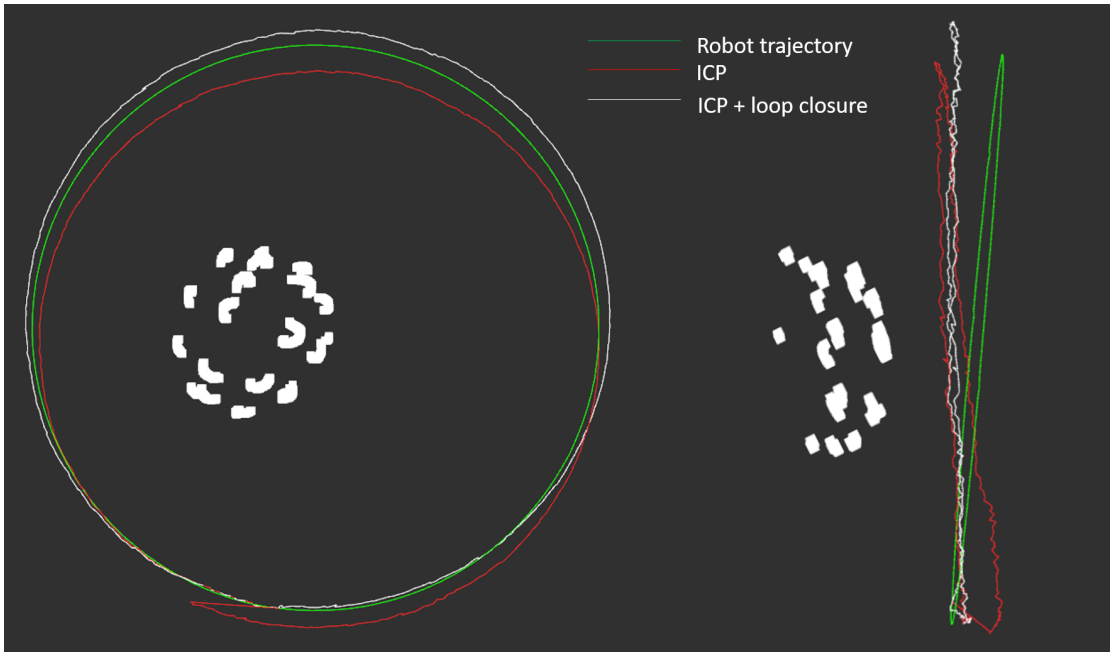

Fig. 4.10: Front and side view of trajectories for one of the cap with 23 electrodes. Green: Ground truth, Red: ICP alone, White: ICP+Loop closure, robot trajectory radius is 300mm.

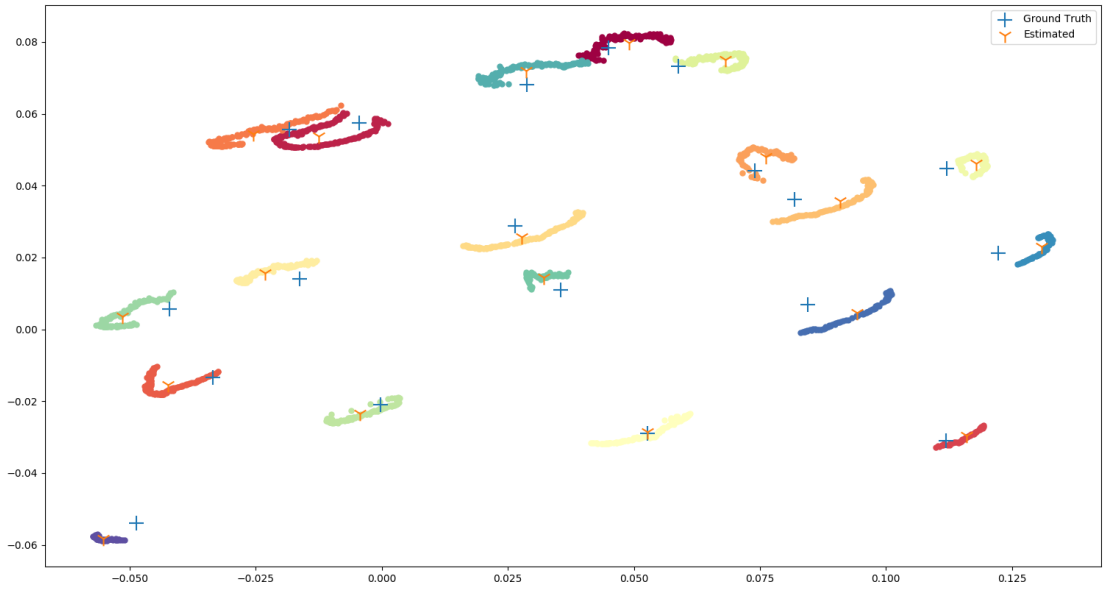

Fig. 4.11: Post registration results, Mean L2 norm : 8.6 (mm), Plus sign : ground truth , *tri_down* sign : estimated

Tab. 4.10: ICP vs ground truth comparison

Metric	CAP_63	CAP_23_1	CAP_23_2
Translation error	4 mm ± 7 mm	7.1 mm ± 9 mm	8 mm ± 7 mm
Rotational error	1.1°± 0.6°	1.5°± 1°	1.4°± 0.8°

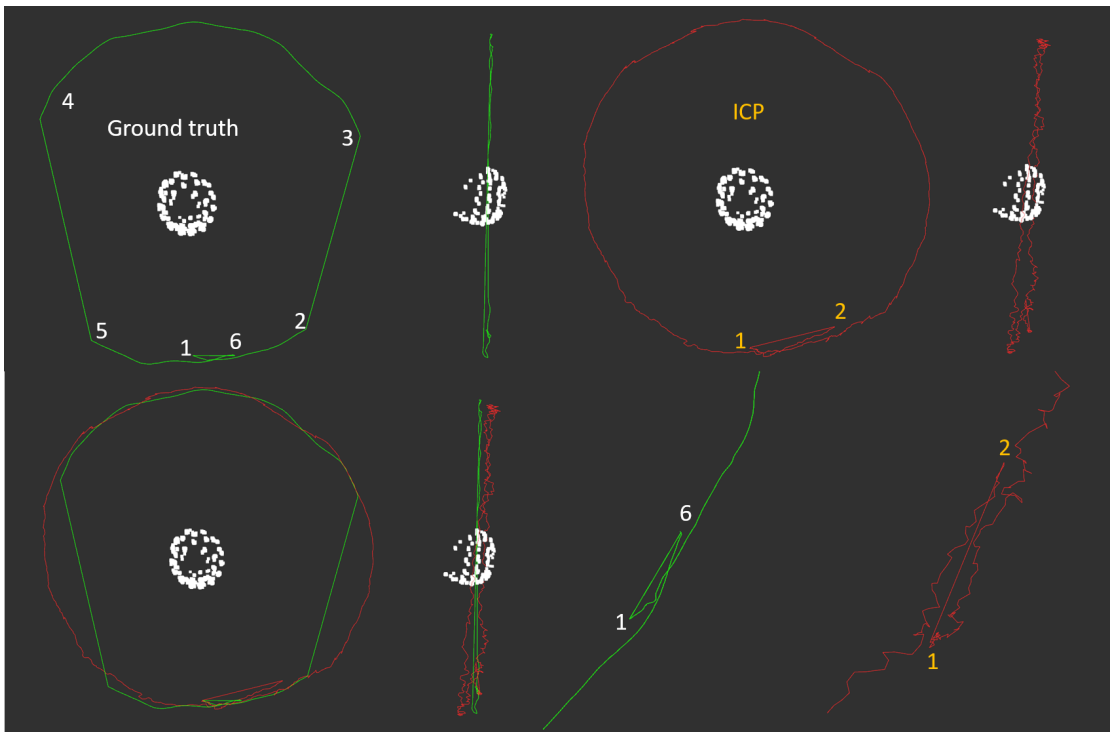

Fig. 4.12: Front and side view of trajectories for one of the cap with 63 electrodes. The sections 1-2, 3-4, and 5-6 are visible while sections 2-3, 4-5 are not. Hand trajectory radius is ≈ 500mm.

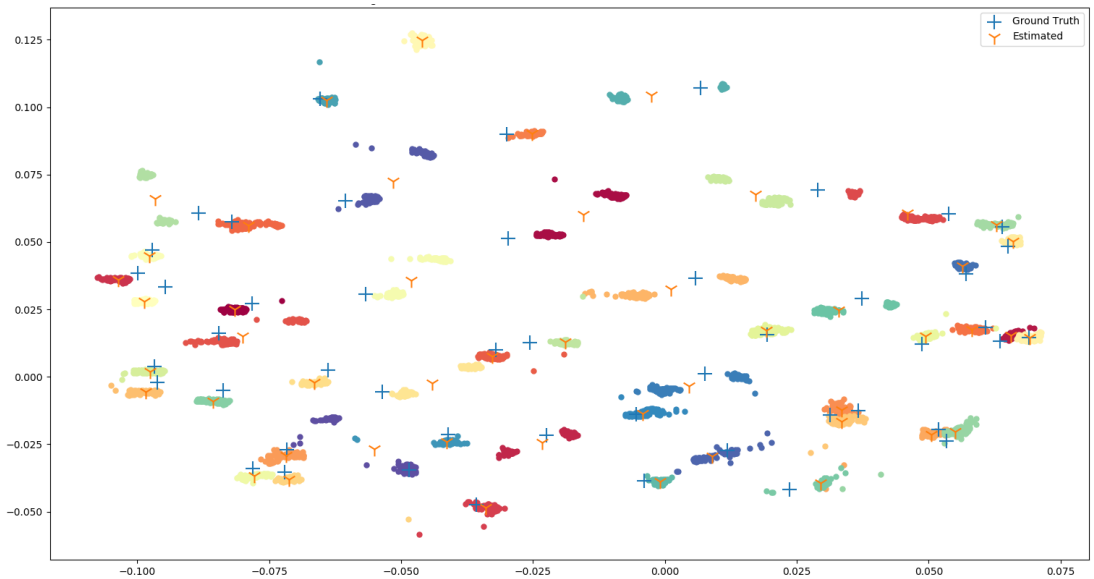

Fig. 4.13: Post registration results, Mean L2 norm : 8 (mm), Plus sign : ground truth , *tri_down* sign : estimated

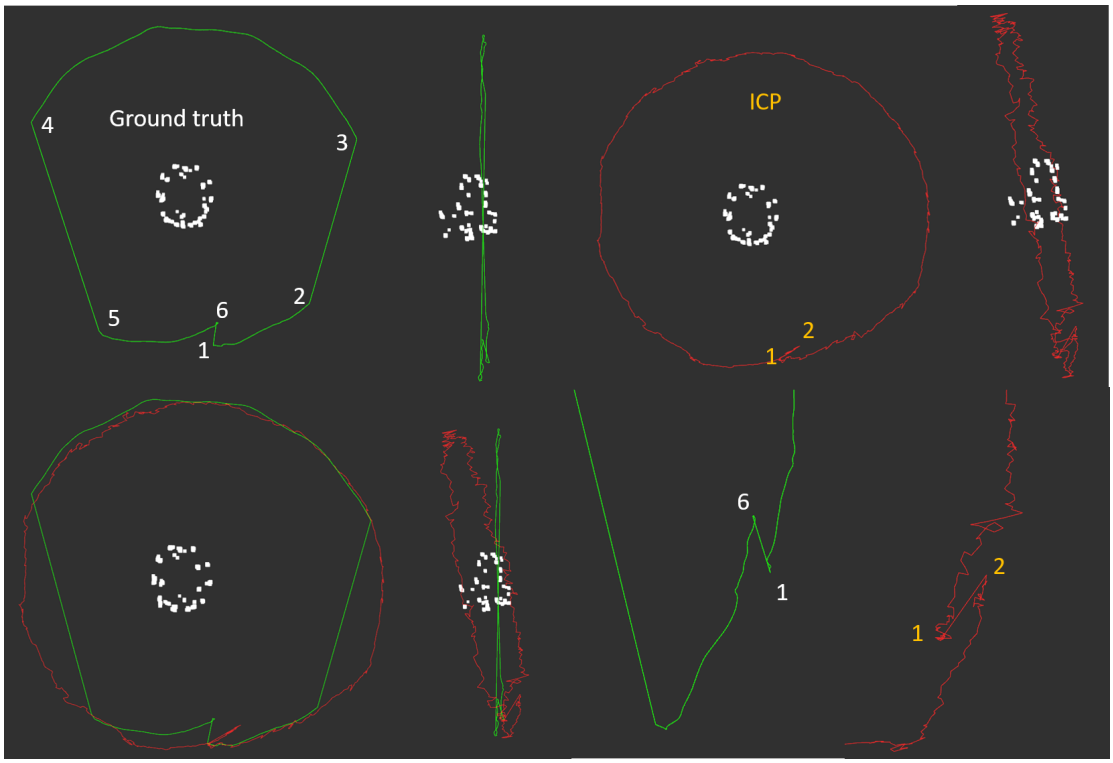

Fig. 4.14: Front and side view of trajectories for one of the cap with 63 electrodes. The sections 1-2, 3-4, and 5-6 are visible while sections 2-3, 4-5 are not. Hand trajectory radius is ≈ 500mm.

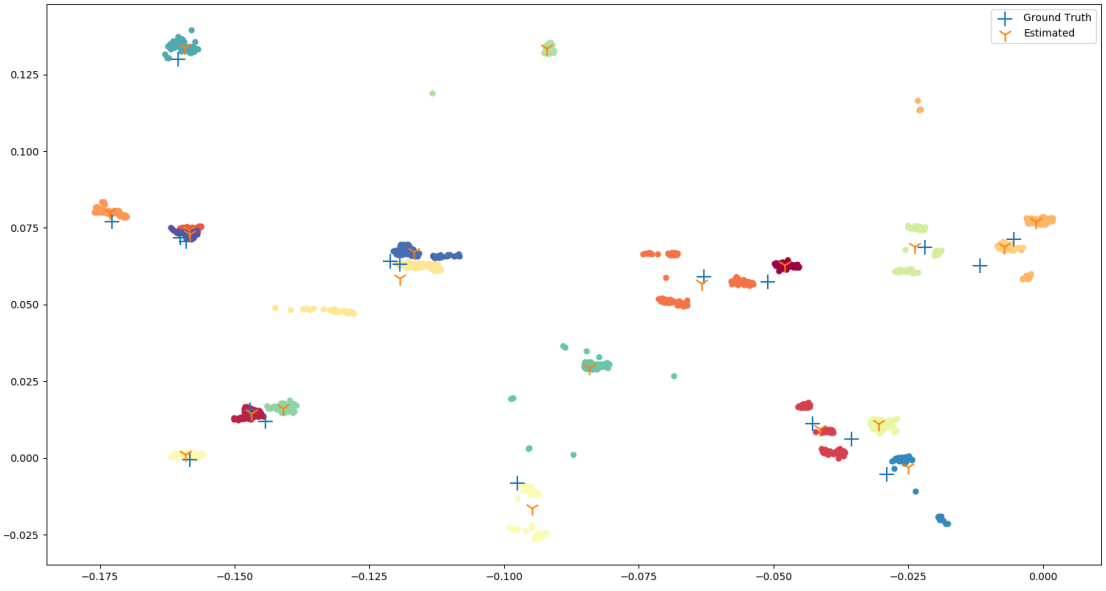

Fig. 4.15: Post registration results, Mean L2 norm : 10.7 (mm), Plus sign : ground truth , *tri_down* sign : estimated

5 Discussion

5.1 Calibrations

There are differences in camera intrinsic matrices estimated using OpenCV, Matlab, and values in the robot-guided data set as shown in tab. 2.1. The effect of these differences can be seen when pixel values are out-projected using camera intrinsic parameters. The fig. 4.1 shows the mean L2 norm with different distances from the camera. Two points can be observed here, lower the focal length higher the over estimation. L2 Norm tends to increase with the distance from the camera.

The tab. 4.2 summarizes all hand-eye and eye-in-hand calibrations. The hand-eye calibration between robot-tracking camera (fusionTrac 500) resulted in mean positional error < 0.2 millimeters and mean rotational error < 0.1 degrees on a hold-out validation set and the eye-in-Hand calibration between robot-kinect resulted in mean positional error < 1.5 millimeters and mean rotational error < 0.3 degrees on a hold-out validation set. In the hand-guided case, eye-in-hand calibration between kinect-marker resulted in mean positional error < 9 millimeters and mean rotational error < 0.25 degrees on a hold-out validation set. This is significantly higher compared to the calibration using fusionTrac and robot where movements are precisely controlled. In the hand-guided case, the camera along with the marker were moved to different positions by hand (keeping the hand steady while acquiring poses) and marker poses are acquired by Cambar B2. The same process is repeated while the camera along with the marker is mounted on a tripod and moved to different positions which offers better stability while acquiring poses compared to holding by hand and obtained similar calibrations errors. The higher errors are probably attributed to the less accurate pose tracking capability of the Cambar. Therefore, it is strongly recommended to perform eye-in-hand calibration using robot and Cambar B2 to get more insight on baseline position tracking errors.

5.2 Depth offset determination

For the robot-guided case, both ICP scan registration and cluster center comparison revealed that RMSE is lower at depth offset β between 24 - 26 mm. For the inertia calculation, clusters are well separated at $\beta > 20$ especially for the cap with 63 electrodes. At $\beta = 10$ clusters are overlapped with each other, at $\beta = 24$ clusters are well separated and tightly bound, where at $\beta > 24$ starts to spread again. The $\beta = 25$ is used for the robot-guided case.

For the hand-guided case, inertia based combined depth and time synchronization led to the lowest inertia at β between 9 - 10 mm. This value is inconsistent with previously seen behavior that over estimation tends to increase with the distance from the camera using simple depth estimation experiment using chessboard (hand guided trajectory radius is $\approx$ 150-200 mm bigger than robot-guided trajectory). It is suspected that this is due to high eye-in-hand calibration error, time synchronization. In addition, the

entire ground truth trajectory is not visible hence, the majority of cluster points are not included in combined optimization. The time synchronized trajectory is used for cluster center comparison and results are shown in fig. 4.7. The graph shows the average minimum L2 norm between 7 - 9 mm. Therefore, $\beta = 10$ is used for the hand-guided case.

5.3 SLAM

In the robot-guided case, very controlled motion is achieved and we can command the robot to visit the previously visited place with exact position and orientation. The tab. 4.3 shows criteria used for detecting the loop closure candidate. These numbers are arrived at based on heuristics. The euclidean norm and ICP fitness threshold for robot-guided case is 100mm and 0.8 respectively. Recall that while constructing pose-graph, all the camera position is expressed relative to the very first camera frame including the current frame from which loop closure is being attempted. This enables one to calculate the 3D position of the coordinate frame of all the previously acquired scans. Note that we compute the 3D euclidean distance between the current frame and a potential loop closure candidate as there is a high chance of elevation change. This means that the candidate has to be within a euclidean distance of 100mm and has to have at least ICP fitness of 80 % for it to be considered as a potential loop closure candidate. The fig. 4.8 and fig. 4.10 shows both front and side view of robot-guided (ground truth), ICP estimation, and ICP + loop closure trajectories. While using ICP alone, the inherent accumulation of error at each scan registration leads to the drift in overall trajectory estimation. Hence SLAM is an essential step in correcting this drift. One can observe drift is higher in the case of a cap with 23 electrodes in comparison with 63 as there are fewer electrodes available per image for scan registration. The translation and rotational error in the case of cap with 23 electrodes are higher as shown in tab. 4.6.

However, in the hand-guided case, where controlled motion is unlikely to achieve and in most of the cases, it is not guaranteed to revisit the same place at exact position and orientation as before. One can also expect elevation changes while revisiting the places. These practical issues make loop closure hard in hand-guided case. Each ICP registration results in a large drift and this drift is accumulated over the entire trajectory which is beyond correction. Please note, experiments have shown that further reduction of loop closure thresholds will results in point cloud distortion and clusters will be overlapped to an extent that different clusters cannot be differentiated. Therefore, all the results for hand-guided case are presented without loop closure.

5.4 Electrode Digitalization

In the robot-guided case, the mean absolute 3D position error of 11.5 mm $\pm$ 5 mm for the cap with 63 electrodes, 11.7 mm $\pm$ 4.0 mm, and 12.6 mm $\pm$ 5.5 mm for caps with 23 electrodes were achieved respectively. This 3D position error includes the systematic error in acquiring the ground truth positions. The post ICP registration resulted in a mean error of 3.7 mm $\pm$ 0.3 for the cap with 63 electrodes, and 6.8 mm$\pm$ 2.0 mm, 5.6 mm $\pm$ 1.2 mm for caps with 23 electrodes respectively. For the hand-guided case, due to high errors in eye-in-hand calibration and the time synchronization, one to one correspondences between estimated and ground truth electrodes could not be found for

3D absolute position comparison. The post ICP registration resulted in mean errors of 10 mm ± 2.7 mm for the cap with 63 electrodes, 14.6 mm ± 3.4 mm and 19.3 mm± 2.3 mm, for caps with 23 electrodes respectively. In comparison with various other methods summarized in the tab. 1.1, digitalization errors (post registration) in the robot-guided case is on par with that of the positioning tool. However, errors in the hand-guided case are significantly higher.

6 Conclusion and Future scope

The electrode digitalization system purely based on the inexpensive RGB-D camera using machine learning based electrode detection framework, pose-graph based simultaneous localization and mapping, and various cluster processing algorithm was developed and evaluated. The different software modules integrating all the hardware have been implemented using open source software/libraries like miniSam, Open3D, and OpenCV, etc. Static offsets are determined using hand-eye, eye-in-hand calibrations. The camera calibration is carried out to estimate the intrinsic parameters. A simple chessboard depth estimation experiment using image point out-projection using raw depth and estimated intrinsic parameters has shown an offset in depth estimation by the camera. For the particular problem at hand, this offset is corrected by comparing the estimated and ground truth points. In the robot-guided case, the average digitalization error 3.7 mm $\pm$ 0.3 for the cap with 63 electrodes, 6.8 mm $\pm$ 2.0 mm and 5.6 mm $\pm$ 1.2 mm for two caps with 23 electrodes were achieved respectively. For the hand-guided case, 10 mm $\pm$ 2.7 mm for the cap with 63 electrodes, 14.6 mm $\pm$ 3.4 mm, and 19.3 mm$\pm$ 2.3 mm, for caps with 23 electrodes respectively.

Few electrodes are not seen in any of the frames either due to poor camera angle or YOLO failed to detect them. On the other hand, there were false positives and bounding box off position which led to an increased error. Solving these problems could be explored further. The developed algorithm relies on only one single 3D point for each electrode per camera frame for localization. Although this is beneficial in terms of computation time, a small variation in the electrode detection may have an adverse affect on localization. Especially when there are extreme cases where only 3 electrodes are seen per frame. Therefore, it is recommended to include at least few points around the YOLO bounding box center and evaluate its effects on the localization. The Kinect has an inbuilt IMU (inertial measurement unit) and provides a way to record the raw data. Although Microsoft does not provide any readily available tools/algorithms for camera localization using the Kinect IMU data at the time of the thesis, popular approaches like extended Kalman filters (EKF) can be employed to calculate the camera position. Addition of IMU measurement along with ICP and loop closure may lead to better localization. The developed algorithm waits till the end to close the loop and correct the drift. By that time, the accumulation of drift in each sequential ICP registration may be large and beyond correction as in the case of hand held trajectory. Therefore, it is recommended that frequent loop closure is attempted. In this thesis, all the experiments were carried out using a static phantom head which eliminates head movements that occur in real life cases. It would be interesting to apply RGB-D and SLAM based digitalization techniques simulating real life cases.

A Listings

A.1 Least square problem

The least square problem and solutions are referenced from the literature [17]. There are 2 types of least square problems, non-homogeneous $\mathtt{A}\boldsymbol{x} = \boldsymbol{b}$ (type I) and homogeneous $\mathtt{A}\boldsymbol{x} = \mathbf{0}$ (type II).

A.1.1 Type I

Consider A system of equations $\mathtt{A}\boldsymbol{x} = \boldsymbol{b}$ where $\mathtt{A}$ is m $\times$ n matrix.

1. if m $<$ n then, there are more unknowns than the number of equations. In this case, there are infinitely many solutions.
2. if m $=$ n then, In this case, there is an exact solution (unique).
3. if m $>$ n then, there are more equations (data points) than the number of unknowns which is usually the case most of the times. In this case there is no exact solution but we can minimize the algebraic error by solving $\min_{x \in \mathtt{R}^n} \|Ax - b\|^2 = 0$ s.t. $\|b\| \neq 0$.
4. assuming matrix $\mathtt{A}$ is invertable,the solution is $\boldsymbol{x} = (\mathtt{A}^\mathtt{T}\mathtt{A})^{-1}\mathtt{A}^T\boldsymbol{b}$

A.1.2 Type II

Consider A system of equations $\mathtt{A}\boldsymbol{x} = \mathbf{0}$ where $\mathtt{A}$ is m $\times$ n matrix. we will see what is the solution to case m $>$ n where there are more equations (data points) than the number of unknowns.

1. As there is no exact solution, we can minimize the geometric error by solving $\min_{x \in \mathtt{R}^n} \|Ax\|^2 = 0$ s.t. $\|x\| = 1$.
2. the solution $\boldsymbol{x}$ is the right null space of matrix $\mathtt{A}$ and is obtained from unit singular value of A corresponding to the smallest singular value i.e if SVD of $\mathtt{A} = \mathtt{UDV}^T$, then $\boldsymbol{x}$ is the last column of $\mathtt{V}$.

A.2 Rotation matrix

Task is to find the rotation matrix $\mathtt{R}$ which is closest approximation to the given matrix $\mathtt{Q}$. The "best" here means that the Frobenius norm of the $[\mathtt{R} - \mathtt{Q}]$ is minimized [18].

$$\min_R \|R - Q\|_F^2 \quad \text{s.t. } R^T R = I$$

if SVD of $\mathtt{Q} = \mathtt{UDV}^T$, then Matrix $\mathtt{R}$ which satisfies above condition is $\mathtt{R} = \mathtt{UV}^T$.

A.3 Gaussian distribution

This section gives an overview of Gaussian distribution as per [24]. A random variable possess probability density functions (PDF's) like one dimensional normal distribution function with mean μ and variance σ^2. PDF of the normal distribution function is given by,

$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\}$$

PDF of a vector valued variable is called multivariate normal distribution with mean μ and a positive semi-definite and symmetric matrix called covariance matrix Σ. PDF of a multivariate is given by

$$p(x) = det(2\pi\Sigma)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}$$

Canonical representation of the a multivariate is given by a information matrix Ω and an information vector ξ

$$\Omega = \Sigma^{-1}$$
$$\xi = \Sigma^{-1}\mu$$

With few rearrangement canonical form of PDF for multivariate normal distribution can be given by

$$p(x) = \eta \exp\{-\frac{1}{2}x^T\Omega x + x^T\xi\}$$

A.4 Pose comparison

Let `P` be the source pose and `Q` be the target pose, which are 4×4 homogeneous matrices with $[R|t]$. R is the rotation and t is the translational part. Let $\mathtt{P}_q$, $\mathtt{Q}_q$ be the quaternion form of above poses and $\mathtt{E}_q = \mathtt{P}_q * \mathtt{Q}_q^{-1}$ be the error in quaternion form. Task is to compare them and form a metric.

Let (k_E,θ_E) be axis angle representation [22] of rotational part $\mathtt{R}[E_q]$. Then the metric for the rotational part is eq. (A.1), the axis part is neglected.

$$|\theta_E| \tag{A.1}$$

Let t_p and t_q be the translational part of `P` and `Q` respectively. Then the metric for translational part is eq. (A.2).

$$||t_p||_2 - ||t_q||_2 \tag{A.2}$$

Bibliography

[1] en.wikipedia.org. (1999). "MS Windows NT kernel description," [Online]. Available: `https://en.wikipedia.org/wiki/Electroencephalography` (visited on 05/20/2020).

[2] S. Qian and Y. Sheng, "A single camera photogrammetry system for multi-angle fast localization of eeg electrodes," *Annals of biomedical engineering*, vol. 39, pp. 2844–56, Aug. 2011. DOI: 10.1007/s10439-011-0374-6.

[3] C. Binnie, E Dekker, A Smit, and G Van der Linden, "Practical considerations in the positioning of eeg electrodes," *Electroencephalography and Clinical Neurophysiology*, vol. 53, no. 4, pp. 453–458, 1982.

[4] J. De Munck, P. Vijn, and H. Spekreijse, "A practical method for determining electrode positions on the head," *Electroencephalography and clinical Neurophysiology*, vol. 78, no. 1, pp. 85–87, 1991.

[5] D. Khosla, M. Don, and B. Kwong, "Spatial mislocalization of eeg electrodes–effects on accuracy of dipole estimation," *Clinical neurophysiology*, vol. 110, no. 2, pp. 261–271, 1999.

[6] J. Le, M. Lu, E. Pellouchoud, and A. Gevins, "A rapid method for determining standard 10/10 electrode positions for high resolution eeg studies," *Electroencephalography and clinical neurophysiology*, vol. 106, no. 6, pp. 554–558, 1998.

[7] S Steddin and K Bötzel, "A new device for scalp electrode localization with unrestrained head," *J. Neurol*, vol. 242, p. 65, 1995.

[8] L. Koessler, T. Cecchin, O. Caspary, A. Benhadid, H. Vespignani, and L. Maillard, "Eeg–mri co-registration and sensor labeling using a 3d laser scanner," *Annals of Biomedical Engineering*, vol. 39, no. 3, pp. 983–995, 2011.

[9] H. Bauer, C. Lamm, S. Holzreiter, I. Holländer, U. Leodolter, and M. Leodolter, "Measurement of 3d electrode coordinates by means of a 3d photogrammetric head digitizer," *NeuroImage*, vol. 11, no. 5, S461, 2000.

[10] G. S. Russell, K. J. Eriksen, P. Poolman, P. Luu, and D. M. Tucker, "Geodesic photogrammetry for localizing sensor positions in dense-array eeg," *Clinical Neurophysiology*, vol. 116, no. 5, pp. 1130–1140, 2005.

[11] L. Koessler, L. Maillard, A. Benhadid, J.-P. Vignal, M. Braun, and H. Vespignani, "Spatial localization of eeg electrodes," *Neurophysiologie Clinique/Clinical Neurophysiology*, vol. 37, no. 2, pp. 97–102, 2007.

[12] U. Baysal and G. Şengül, "Single camera photogrammetry system for eeg electrode identification and localization," *Annals of biomedical engineering*, vol. 38, no. 4, pp. 1539–1547, 2010.

[13] S. Chen, Y. He, H. Qiu, X. Yan, and M. Zhao, "Spatial localization of eeg electrodes in a tof+ ccd camera system," *Frontiers in neuroinformatics*, vol. 13, p. 21, 2019.

[14] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, International Society for Optics and Photonics, vol. 1611, 1992, pp. 586–606.

[15] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[16] opencv.org. (1999). "MS Windows NT kernel description," [Online]. Available: `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html` (visited on 05/20/2020).

[17] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003, ISBN: 0521540518.

[18] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000, ISSN: 0162-8828. DOI: 10.1109/34.888718.

[19] *Hand-eye calibration — rc_visard 21.01.1 documentation*, `https://doc.rc-visard.com/latest/en/handeye_calibration.html`, (Accessed on 04/13/2021).

[20] F. Ernst, L. Richter, L. Matthäus, V. Martens, R. Bruder, A. Schlaefer, and A. Schweikard, "Non-orthogonal tool/flange and robot/world calibration," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 8, no. 4, pp. 407–420, 2012.

[21] Wikipedia. (1999). "MS Windows NT kernel description," [Online]. Available: `https://en.wikipedia.org/wiki/QR_decomposition` (visited on 05/20/2020).

[22] ——, (1999). "MS Windows NT kernel description," [Online]. Available: `https://en.wikipedia.org/wiki/Axis%E2%80%93angle_representation` (visited on 05/20/2020).

[23] I. Söderkvist, "Using svd for some fitting problems," *University Lecture*, 2009.

[24] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005, ISBN: 0262201623.

[25] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous robot vehicles*, Springer, 1990, pp. 167–193.

[26] J. A. Castellanos, J. Montiel, J. Neira, and J. D. Tardós, "The spmap: A probabilistic framework for simultaneous localization and map building," *IEEE Transactions on robotics and Automation*, vol. 15, no. 5, pp. 948–952, 1999.

[27] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[28] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[29] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[30] *Sklearn.cluster.dbscan — scikit-learn 0.24.2 documentation*, `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html`, (Accessed on 04/28/2021).

[31] www.ros.org. (1999). "MS Windows NT kernel description," [Online]. Available: `https://www.ros.org/about-ros/` (visited on 05/20/2020).

[32] *What is matlab? - matlab & simulink*, `https://de.mathworks.com/discovery/what-is-matlab.html`, (Accessed on 05/01/2021).

[33] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3d: A modern library for 3d data processing," *arXiv preprint arXiv:1801.09847*, 2018.

[34] J. Dong and Z. Lv, "Minisam: A flexible factor graph non-linear least squares optimization framework," *arXiv preprint arXiv:1909.00903*, 2019.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] microsoft.com. (1999). "MS Windows NT kernel description," [Online]. Available: `https://docs.microsoft.com/en-us/azure/kinect-dk/hardware-specification` (visited on 05/20/2020).

[37] *Azure kinect dk hardware specifications | microsoft docs*, `https://docs.microsoft.com/en-us/azure/kinect-dk/hardware-specification`, (Accessed on 05/06/2021).

[38] atracsys.com. (1999). "MS Windows NT kernel description," [Online]. Available: `https://www.atracsys-measurement.com/products/fusiontrack-500/` (visited on 05/20/2020).

[39] *Cambar b2 c4 and cambar b2 c8 - axios 3d® services gmbh*, `https://axios3d.de/optische-messsysteme/cambar-b2-c4-und-cambar-b2-c8/`, (Accessed on 04/07/2021).

Declaration

Hiermit versichere ich, dass ich meine Abschlussarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum: .....

(Unterschrift)