TUHH
Hamburg University of Technology

Pavan Vishwanath

Developement of a Robotic Data Acquisition Setup for Camera Based Electrode Digitalization

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

mTEC
Medical Technology | Science | Systems

A project paper written at the Institute of Medical Technology and submitted in partial fulfillment of the requirements for the degree Master of Science.

Author: Pavan Vishwanath

Title: Developement of a Robotic Data Acquisition Setup for Camera Based Electrode Digitalization

Date: June 2, 2020

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

Contents

List of Figures

Abstract

Electroencephalography (EEG) is used to record electrical activities of the brain using an electrode cap which is placed on the scalp. There are accurate methods developed to localize the electrodes, for example, using optical scanners, however, these methods are expensive and time-consuming. On the other hand, an inexpensive and faster solution is currently developed using a conventional camera and deep learning network. The development and evaluation of the method require a large amount of ground truth data for the electrode positions. In this work, a robotic system for the systematic acquisition of data for this purpose is developed. A phantom head wearing the EEG cap is mounted on to a robot and a head coordinate system using anatomical features of the phantom face is created. The position of all the electrodes in the EEG cap is recorded using a stylus with attached reflective markers and then transformed to the head coordinate system. The possible sources of error while using the camera, robot, and stylus for recording the positions of the electrodes have been examined. The data acquisition system can capture the electrode positions with respect to the head coordinate system with an accuracy of $\pm$5mm. Three 30-45 seconds of video sequences with 30fps for three different EEG caps have been recorded with the system. This data can not only be used to train the convolutional neural networks (CNNs) and also to evaluate the performance of the camera-based electrode detection algorithm in general.

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]. It is therefore evident to know the electrode locations on the scalp.

Several electrode detection methods based on cheap cameras and RGBD images have been developed so far [2] [3] which are suitable for the fixed head position. A convolutional neural network (CNNs) based method has been proposed [4] which is suitable when head movement is inevitable and fast detection is needed. The latter method also highlights a large amount of ground truth data generation using a robotic setup. In this thesis, a complete robotic set up required for data generation has been realized. The primary goal of this project is to generate ground truth data set i.e. the exact position of the electrodes in head coordinate systems for different electrode caps and different cap positions which can be used to evaluate algorithms for electrode detection using conventional camera.

A phantom head wearing an EEG cap is mounted on to the robot's end-effector followed by head coordinate system creation and manual electrode mapping. RGBD images of the phantom head along with the end-effector position will be captured while the robot is moved along specified trajectories. The process is repeated with three different caps and orientation.

The thesis structure is as follows : In chapter 2 we provide a brief introduction of pinhole camera model, projection matrix, the process involved in the camera calibration.

In chapter 3 we present the experimental setup required along with hardware descriptions and the purpose of the hardware used. Then we explain what are the key software packages used in the project. A detailed explanation of hand-eye calibration, the corresponding algorithms, head coordinate creation, and the electrode mapping are given.

In chapter 4 we present the results achieved for camera calibration, hand-eye calibration, and head coordinate system created using anatomical features of the face. We also discuss the effects of orientation of the marker and movement of the phantom and resulting errors on the estimation of electrode position. We graphically present the position of electrodes for easy visualization. We present the data for 5 EEG cap wearing instances to show the extent to which position of the electrodes varies.

In the last chapter, we conclude by discussing the limitations of the robotic setup and provide the scope for improvements.

2 Background

2.1 Camera fundamenals

Since the camera is an integral part of this project, we would like to provide a gentle introduction to the pinhole camera model, projection matrix, calibration (internal and external), the underlying mathematics.

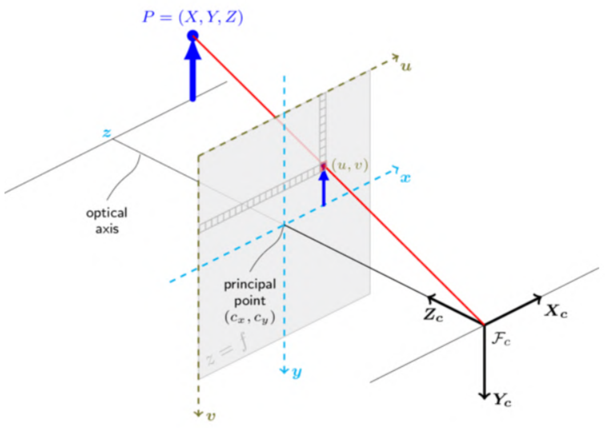


Fig. 2.1: Pinhole camera geometry. $\boldsymbol{F_c}$ *is the camera center,* (C_x, C_y) *is the principal point,* $\boldsymbol{P}$ $(X, Y, Z)^T$ *is world point,* $\boldsymbol{P_c}$ $(X_c, Y_c, Z_c)^T$ *is world point measured in camera coordinate system.[5]*

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) and 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, being able 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 $\mathtt{P}$ 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.

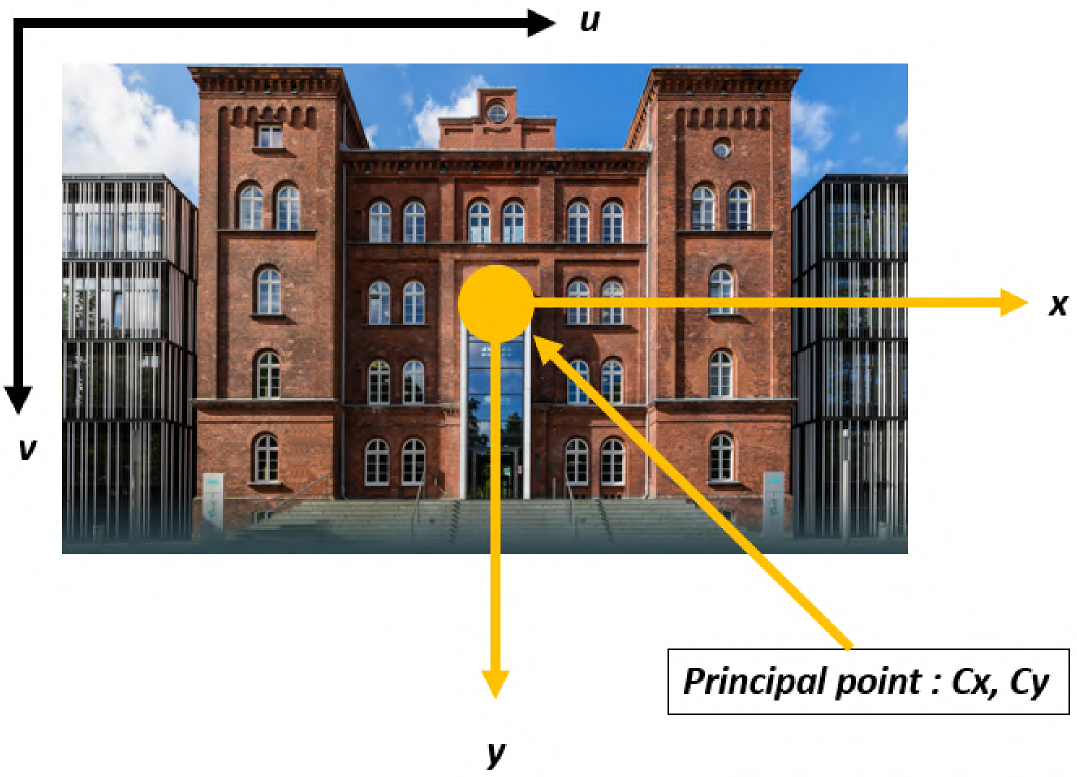


Fig. 2.2: Pixel coordinates (u, v) and the camera coordinates (x, y).

The equation 2.2 depicts mapping image to pixel coordinates. wehere (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 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 and there are many ways to do so. Two approaches will be presented here, one using projection matrix from equation 2.7 (2D/3D correspondence) and using homography (2D/2D correspondence).

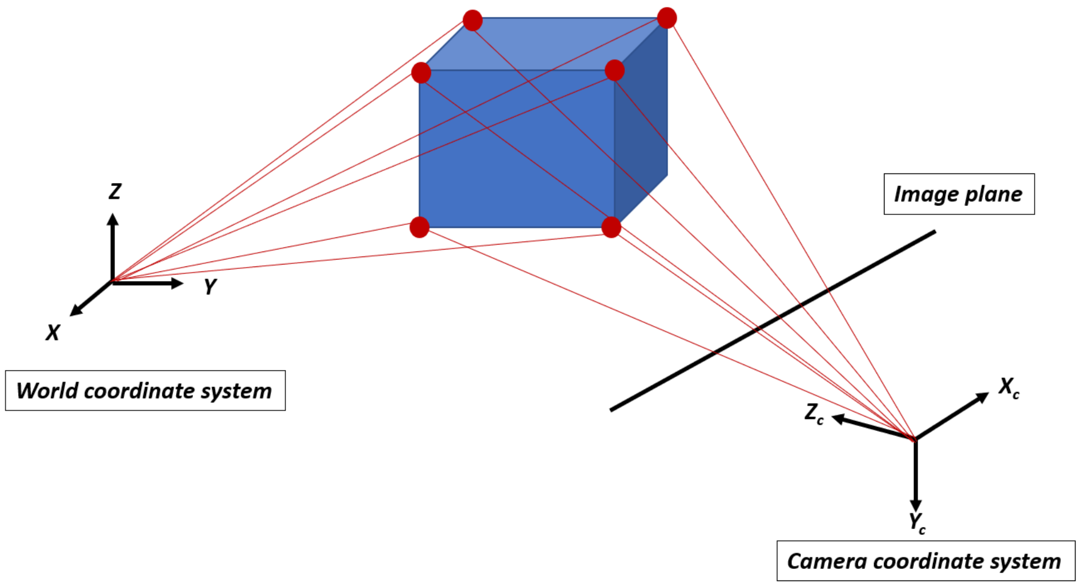


Fig. 2.3: Mapping 3D world points on to a 2D image plane.

2.2.1 2D/3D correspondence

The fig. 2.4 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 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 yileds

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

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

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 $\boldsymbol{p}$.

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

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

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).

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

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 Ap = 0 is the right nullspace 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 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 [6]. 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

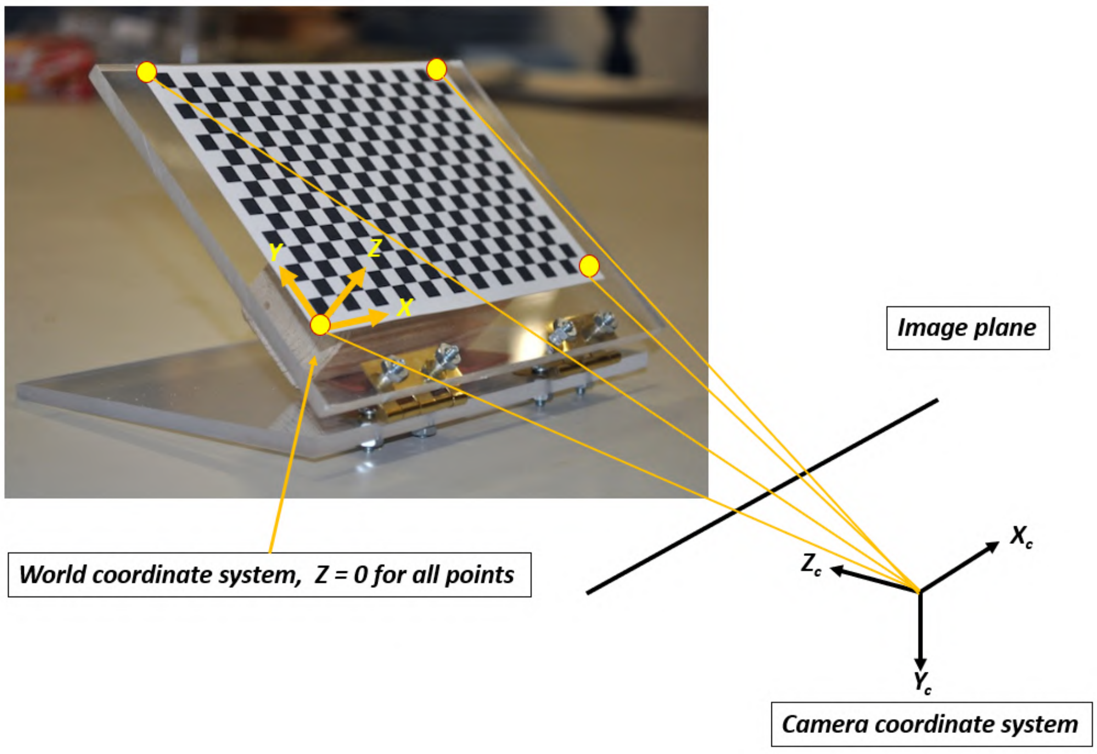


Fig. 2.4: Mapping 2D planar world points (Z = 0) on to a 2D image plane.

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 easy method to estimate the projection matrix thereby intrinsic and extrinsic camera parameters, developed by Zhang [7] 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 nullspace 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 projrction 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 soution obtained from 2.11.

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,

$$\mathbf{r}_1^T \mathbf{r}_2 = 0$$
$$\mathbf{r}_1^T \mathbf{r}_1 = \mathbf{r}_2^T \mathbf{r}_2 = 1$$

rewriting with K gives,

$$\mathbf{p}_1^T K^{-T} K^{-1} \mathbf{p}_2 = 0$$
$$\mathbf{p}_1^T K^{-T} K^{-1} \mathbf{p}_1 = \mathbf{p}_2^T K^{-T} K^{-1} \mathbf{p}_2 = 1$$

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 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 therfore 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 [7] 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 theoritically equal to $[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]$. Since the rotational matrix is a orthogonal matrix, third column is othogonal 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 theoritically 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 necessary to find a best 3x3 rotation matrix R from a given 3x3 matrix Q, here "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 normalised 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

$$\begin{aligned} x^{'} &= x \times \frac{1 + k_1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^6} + 2p_1xy + p_2(r^2 + 2x^2) \\ y^{'} &= y \times \frac{1 + k_1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^6} + p_1(r^2 + 2y^2) + 2p_2xy \end{aligned} \tag{2.18}$$

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 $(u, v, z)^T = (x^{'} + C_x, y^{'} + C_y, f)^T$.

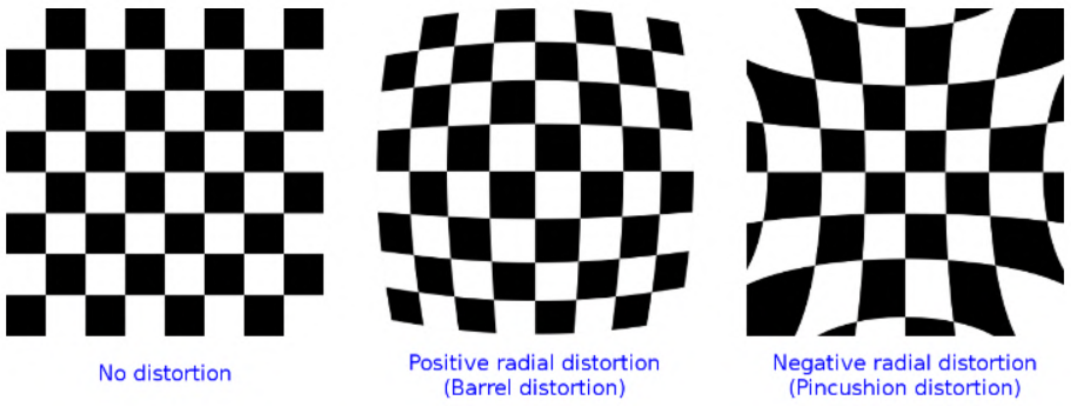


Fig. 2.5: Barrel distortion (*typically* $k_1 > 0$) and pincushion distortion (*typicallyk*$_1 < 0$) [5]

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 Hand-eye calibration

Having finished camera calibration, the calibrated cameras can be used for hand-eye calibration.

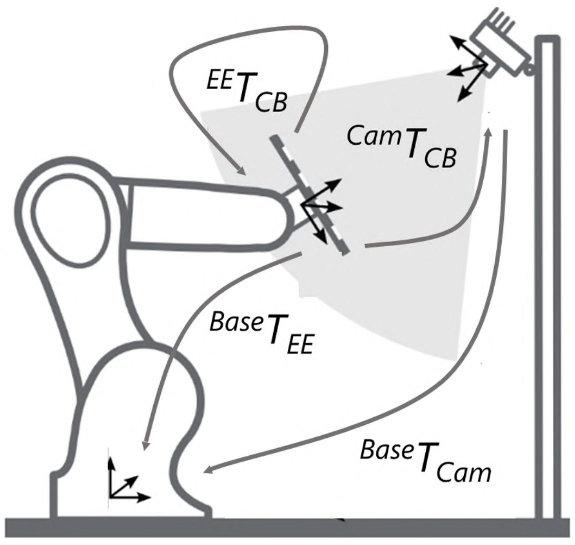


Fig. 2.6: Generic hand-eye calibration setup.

The fig. 2.6 depicts the robot and camera set up required for hand-eye calibration. 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 per [8]

$$\mathtt{M}_i\mathtt{X} - \mathtt{Y}\mathtt{N}_i = 0 \tag{2.19}$$

Where $\mathtt{M}$ is obtained through forward kinematics, $\mathtt{N}$ is obtained by calculating the pose of the calibration pattern from the camera image. Hand-eye calibration aims to solve the two unknowns in eq. (2.19) i.e the chessboard position with respect to the robot's end effector $\mathtt{X}$ and the camera position with respect to the robot's base $\mathtt{Y}$.

The equation 2.19 is transformed to A system of linear equations as

$$\mathtt{A}\boldsymbol{w} = \boldsymbol{b} \tag{2.20}$$

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,

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

and

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

Here, $\mathtt{R[M]}_i \in \mathbb{R}^{3\times3}$ is rotaional part of $\mathtt{M}_i$ and $\mathtt{T[M}_i\mathtt{]} \in \mathbb{R}^3$ 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 2.19 is a form of least square of type I (see appendix) and can be solved using QR factorisation [9] by minimizing quadratic error.

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

Where, $||.||_f$ is Frobenius norm. The above algorithm is called QR-24 according to [8].

Orthonormalization. Note that the matrices computed are not necessarily orthogonal therefore, we need to orthonormalize using singular value decompostion (SVD).

Calibration error. Having finished the hand-eye calibation, It is ncessary to investigate the accuracy of the obtained results. Accuarcy can be obained by rearranging the equation 2.19 as,

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

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 [8]. This fully populated matrix has to be orthonormalised using SVD before calcluating the error as shown below.

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

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 represention [10] of $\mathtt{R}[\mathtt{A}]$ then the rotarional error metric is,

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

in the rotation metric the axis of rotation is neglected.

3 Methods and material

3.1 Experimental setup

As shown in the fig. 3.1 the setup consists of 2 cameras (Kinect by Microsoft and fusionTrac 500 by Atarcsys), A robot (UR3 by Universal Robots), phantom head, multiple EEG caps, 2 markers as shown in fig. 3.8 each serving a specific purpose. The purpose of each hardware used is briefly described below.

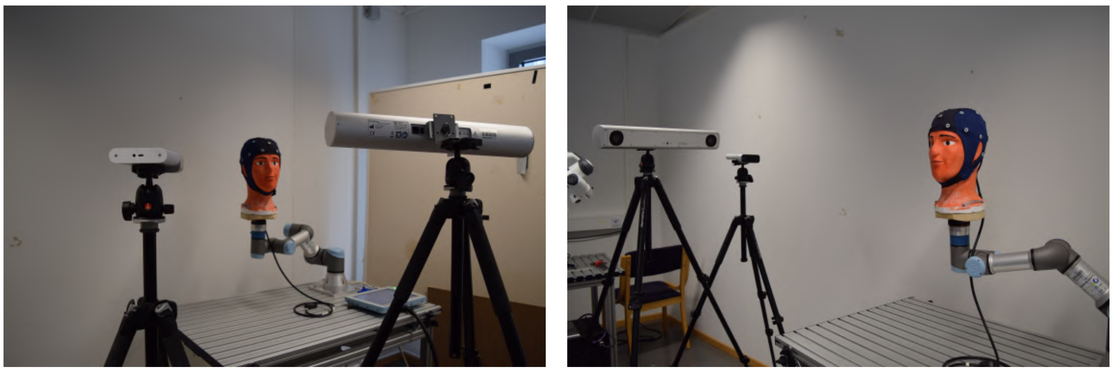


Fig. 3.1: Experimental setup consists of Microsoft kinect, Atracsys fusionTrac 500, UR3 robot.

1. **Kinect camera**
    - Record information such as robot movement, point cloud data, RGBD images, etc.
2. **fusionTrac 500**
    - Head coordinate system creation.
    - Electrode mapping.
3. **UR3**
    - Simulate the patient's head movement.
4. **EEG caps**
    - Caps with different electorde patterns are used to generate ground truth data.
5. **Chessboard and reflective marker**
    - Hand-eye calibration.
    - Electrode mapping.

3.1.1 Kinect camera

Microsoft azure Kinect camera [11] comes with many sophisticated sensors integrated. For example, 12Megapixel RGB sensor, 1 Megapixel time of flight (Tof) depth sensor, motion sensor (accelerometer and gyroscope) and microphone array which provides excellent computer vision and speech models. The camera is compact in size and comes with an USB interface to the computer as shown in fig. 3.2. Microsoft also provides a software development kit (SDK) and easy to use interface for accessing all the sensors. SDK also provides an interface to the robot operating system (ROS).

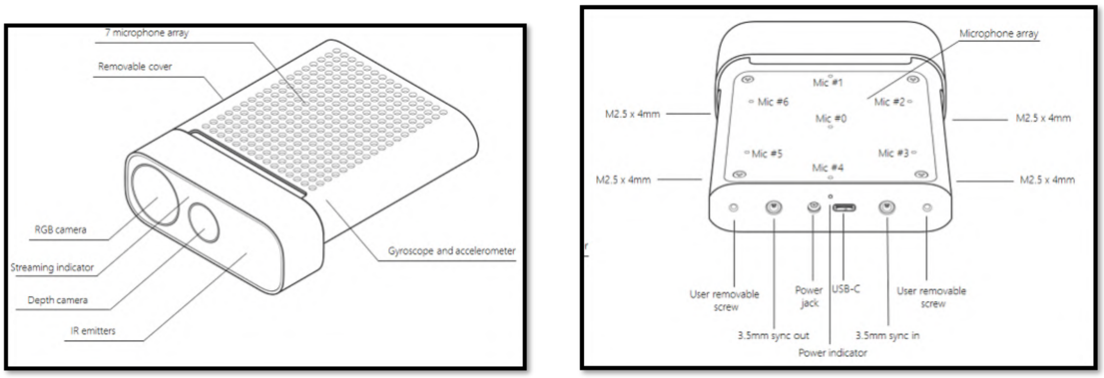


Fig. 3.2: Microsoft azure kinect camera features.

3.1.2 fusionTrac 500

The fusionTrac 500 by Atracsys fig. 3.3 is a passive and active, real-time optical pose-tracking system specially designed to detect and track reflective spheres, disks, and IR-LEDs in real-time. The fusionTrack has two cameras that observe reflective and/or active fiducials (IR LEDs) simultaneously, and it triangulates the detected markers to calculate their locations with high precision and a measurement rate of 335 Hz. The fusionTrac system can be used to estimate the pose when several fiducials are arranged to form a specific shape.

Fig. 3.3: fusionTrac 500 [12]

fusionTrac provides software development kit (SDK) to seamlessly access the data via TCP/IP server and client based system.

3.1.3 UR3 Robot

To simulate the patient's head movements the phantom head has been mounted to ur3 robot from Universal Robotics fig. 3.4 and moved along predefined trajectories. It comes with an onboard controller and an easy tab interface to control. The robot prevents itself against any unforeseen forces by the operator and the environment. Universal robotics also provides a ROS interface from where the robot can be controlled via scripts. The important specifications of the UR3 robot is given in table 3.1.

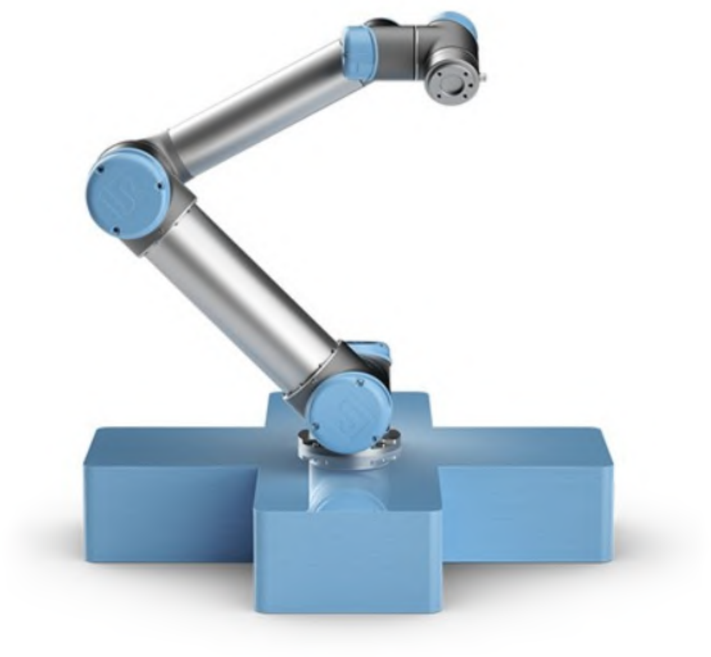


Fig. 3.4: UR3 robot [13]

Tab. 3.1: Important specifications of UR3 robot

Specification	UR3
Repeatability	± 0.1 mm
Payload	3 Kg
Reach	500mm
Degrees of freedom	6 rotating joitns

3.1.4 Phantom head

In order to simulate the patient's head, a dummy phantom head has been used all through the project as shown in the fig. 3.5

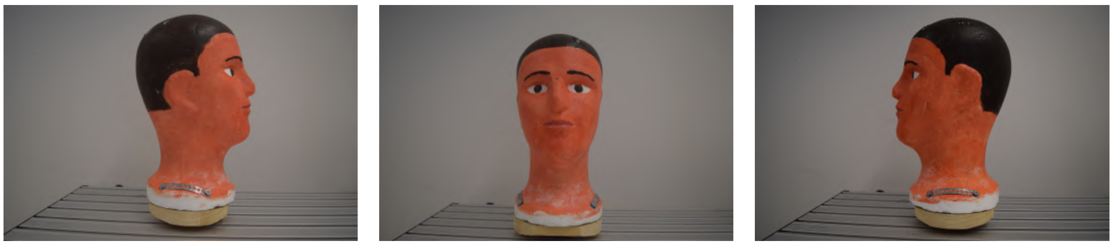


Fig. 3.5: Phantom head used in the experiments.

3.1.5 Electode caps

Several EEG caps provided by eemagine Medical Imaging Solutions GmbH, Berlin, Germany are used to generate the data as shown in fig. 3.6 and fig. 3.7.

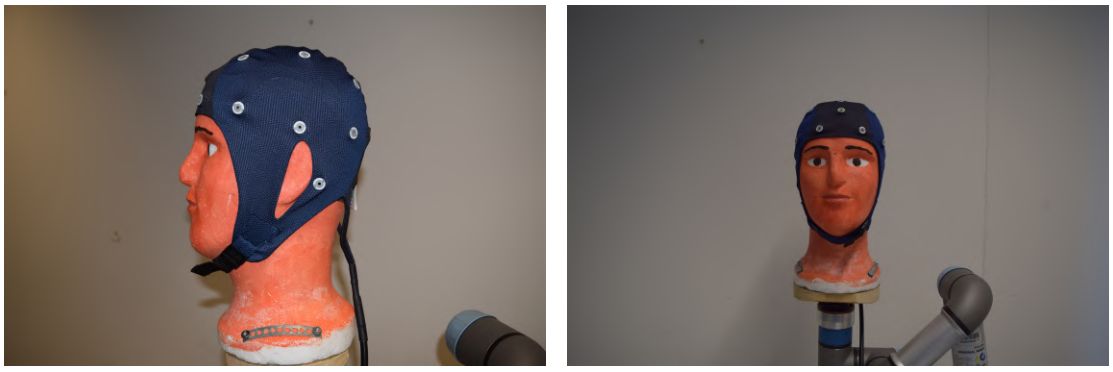


Fig. 3.6: EEG cap on the phantom head.

3.1.6 Markers

A Chessboard with grid size (5 × 8) is used for camera and hand-eye calibration of Kinect while reflective marker is used with fusionTrac 500. Both markers can be seen in the fig. 3.8.

3.2 Software architecture

3.2.1 ROS

Robot operating system [14] 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.

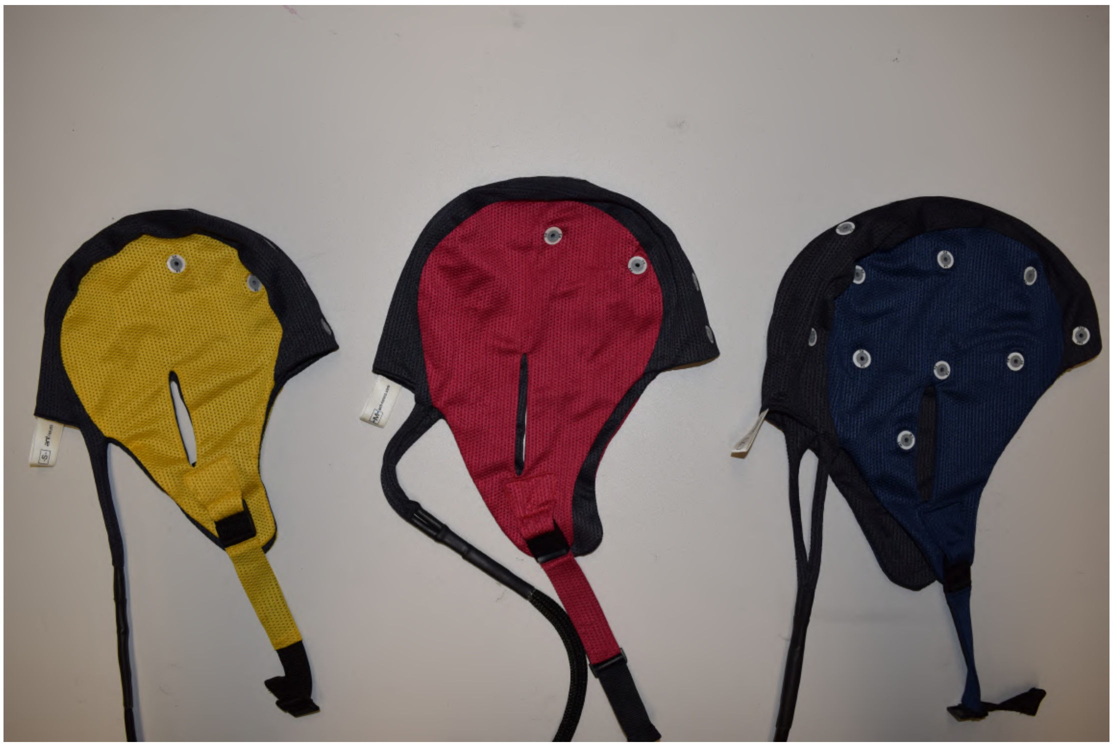


Fig. 3.7: Multiple EEG caps with different elctrode location

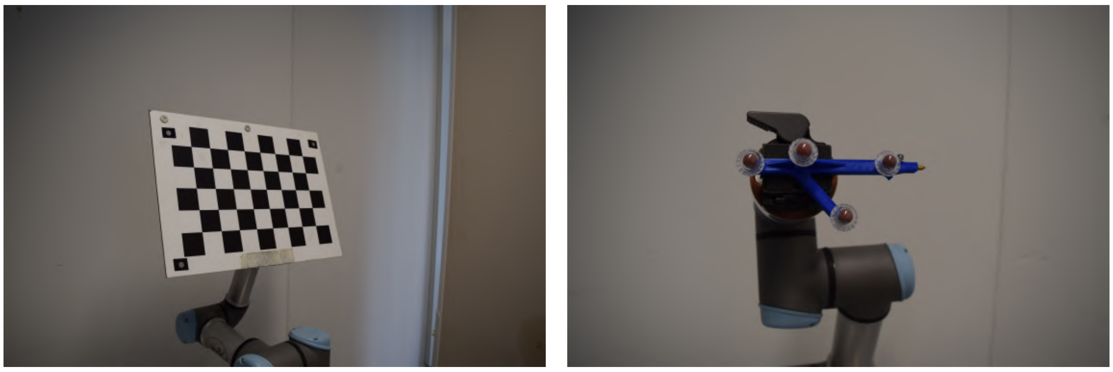


Fig. 3.8: Two types of markers used in the experiments, chessboard and active reflective marker

Robot geometry library

One of the main challenges in the complex robotic system is to know the location of each part of the robot from a common frame of reference. To keep track of both moving and static parts of the robotic system, ROS provides tf transform library. tf allows one to define static frames such as camera, laser sensor location along with dynamic parts such as moving joints in the robot. These tf transform information can be accessed via ROS messages.

Robot description language

Another challenge in the robotic software development is to have the correct geometrical, structural information of the robot and its environments. ROS provides a unique XML based format called URDF (unified robot description format) which helps to capture the realistic representation of the robot and its environments. URDF format is understood by underlying ROS geometrical libraries tf, visualization tools such as rviz.

Communication

ROS is well known for its well developed, robust communication system. It is often the case in a large robotic system live information exchange is a must, for example, mapping the electrodes in the camera frame. The camera needs to know the pose of the phantom head all the time during the motion. These live information exchange between different nodes of the system happens via messages, services, and actions in a publisher/subscriber concept. ROS handles all the hassles of the communication between distributed system so that one can focus on software development.

3.2.2 MoveIt

MoveIt is an open-source manipulation framework for robotic software development. MoveIt has developed robust tools for motion planning to generate highly accurate trajectories for the robot in a messy environment. Manipulation solution provided by moveIt makes it easy to analyze the interaction between the different environments. MoveIt also provides solutions for inverse kinematics, control, 3D perception, and collision checking problems. The advantage of MoveIt is it can be integrated into ROS so that all the ROS tools can be used, it also integrates the physics-based Gazebo simulation. The combination of ROS, MoveIt, Gazebo simulator enables one to develop sophisticated robotic software solutions.
MoveIt provides easy setup assistant tool to configure any robot of choice not only that, there are already moveIt packages available for most popular robots on the market. It supports scripting via c++ and python and comes with many tutorials and clean documentation.
The setup assistant along with easy tutorials guides the user step by step for configuring any robot.

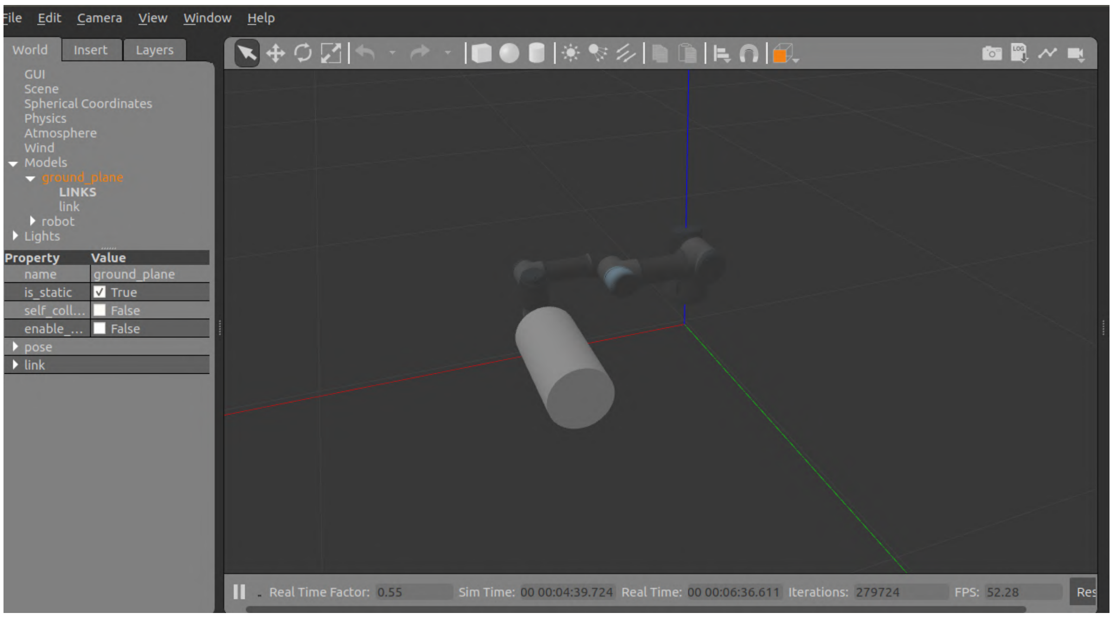
3.2.3 MoveIt configuration

The primary objective of configuring the MoveIt is to generate semantic Robot Description Language (SRDF) [15] along with other important files needed to exploit MoveIt's capabilities. The important steps (not all of them) involved are given a brief introduction below.

Self-Collision matrix

MoveIt checks if there is a possibility of collision between parts of the robot and this takes away the computation time. However, generating a self-collision matrix defines the pairs of links on the robot which are always in contact thus disabling these pairs from collision checking. Also, it defines the pairs which are never in contact so these also can be eliminated from collision checking.

Since the phantom head is mounted to the robot's end-effector, it is necessary to account for collision checking. An approximate model of the phantom head is added to the URDF of the robot as shown in fig. 3.9.



Property	Value
name	ground_plane
is_static	✓ True
self_coll...	False
enable_...	False

Real Time Factor: 0.55 Sim Time: 00 00:04:39.724 Real Time: 00 00:06:36.611 Iterations: 279724 FPS: 52.28

Fig. 3.9: Approximate URDF model of the phantom head.

Virtual joints

Virtual joint attach the robot to the world therefore it is important to define it. Joint type and parent joint must be defined to create a virtual joint.

Planning groups

Planning groups define the robot kinematic chain semantically and these groups are used later to move the robot. Appropriate kinematic solver has to be defined at this point to let MoveIt know which kinematic solver needs to be used. Popular choices are KDL, universal kinematics, or more powerful IK solver.

MoveIt offers many advantages when it comes to commanding the robot such as,

1. **Getting basic info** basic information such as end-effector position, move group details, planning frame, end-effector link, and joint states can be easily queried using MoveIt.
2. **Joint pose goal** robot can be commanded to a specific joint goal by defining an array of joint values.
3. **Pose goal** robot can be commanded to a specific pose by defining a homogeneous 4×4 pose matrix.
4. **Cartesian goal** robot can be commanded to follow a cartesian path by defining a list of waypoints for end-effector to go through.

5. **Trajectory display** we can visualize the path planned for the robot via any of the commanding methods.

6. **Rviz & Gazebo simulation** MoveIt seamlessly integrates Rviz and Gazebo simulation to enhance visualization.

3.3 Camera calibration and pose estimation

Camera calibration is carried out using open-source computer vision library OpenCV [5]. Camera calibration is based on 2D/2D point correspondences with the chessboard as a 2D planar object. The homography is calculated using square corners of a chessboard (world points) and its image points. OpenCV needs an arrays of world points and image points and the grid size of the chessboard (in our case its 5 rows, 8 columns). Therefore, 10 RGB images of the chessboard at different position and orientation was recorded 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 as shown in fig. 3.10 and refines them accordingly.

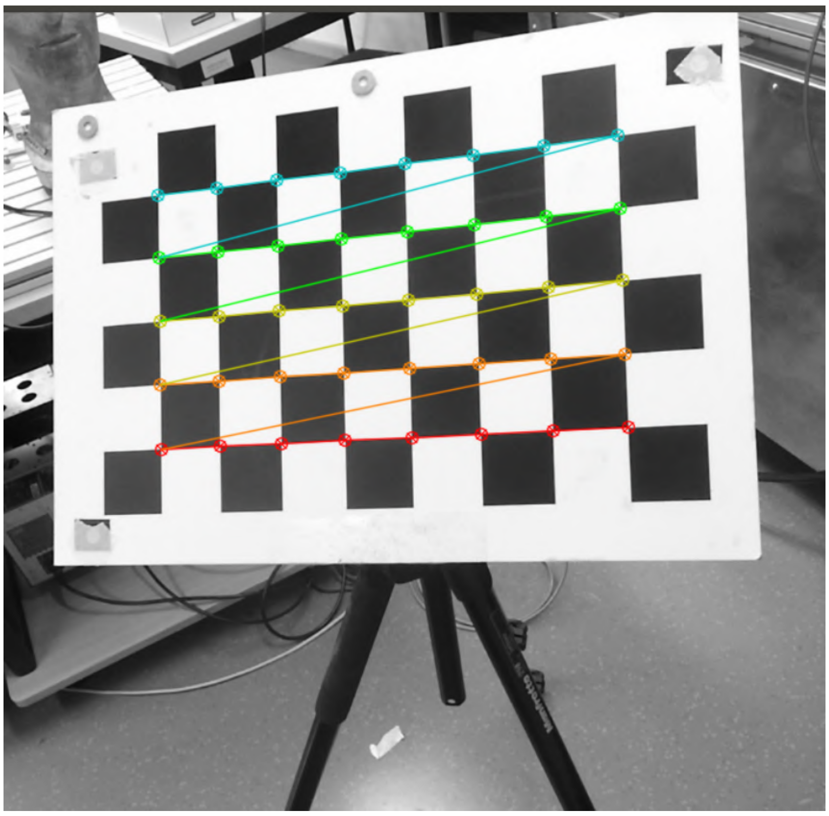


Fig. 3.10: Chessboard corners(image points) detection in OpenCV

```
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 arguements then outputs geometric error of reprojection (rms), intrinsic parameters (camera_matrix) distortion coefficients (dist_coeffs) and extrinsic parameters ($rot\_vecs, trans\_vecs$).

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 coeffcients as the arguments and computes rotation and translation vectors. 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 homogenious matrix for further manipulation.

3.3.1 Kinect and fusionTrac 500 hand-eye calibration setup

The fig. 3.11 shows the hand-eye calibration setup for Kinect with chessboard. Over 40 independent robot poses `M` were defined for hand-eye calibration. The robot drives to each of the defined positions where an image was taken using the Kinect camera. One should make sure that the chessboard is completely visible in the camera image. The pose of the chessboard in camera coordinates `N` is determined for each position by using the pose estimation algorithm very similar to camera extrinsic calibration. 30 poses were used for hand-eye calibration and rest 10 poses (which were not part of hand-eye calibration) were used for error estimation to avoid overfitting.

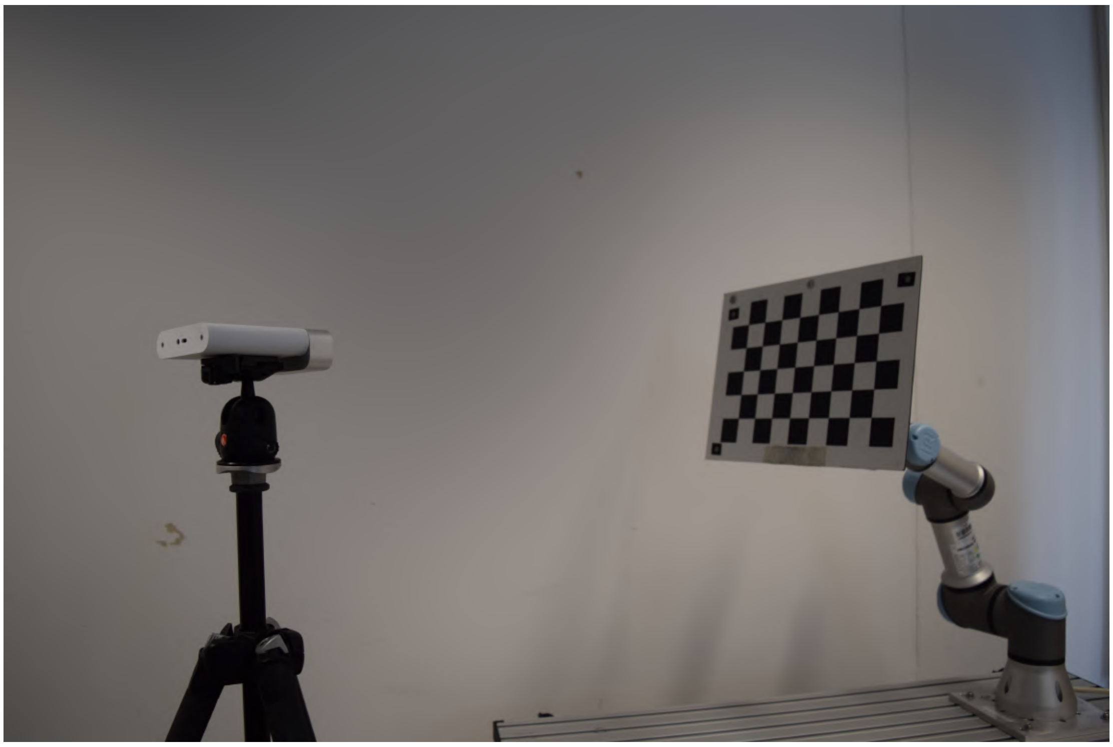


Fig. 3.11: Kinect camera hand-eye calibration setup.

Same strategy goes for the fusionTrac 500 camera also with only exception that reflective marker has been used instead of chessboard as shown in the fig. 3.12.

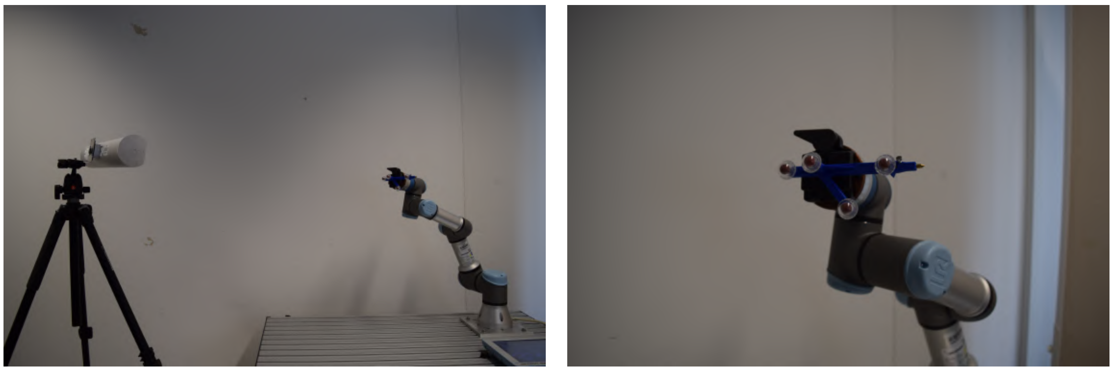


Fig. 3.12: fusionTrac 500 camera hand-eye calibration setup.

3.4 Head coordinate system creation

All the electrodes in the EEG cap have to be mapped to a single coordinate system. we will use BTi or 4D neuroimaging coordinate system defined as per [16]. The BTi/4D coordinate convention uses external anatomical features in the face such as the nasion, left, and right pre-auricular points (LPA & RPA respectively) as a basis for the coordinate system as shown in fig. 3.13.

1. The origin is between LPA & RPA
2. The $\boldsymbol{x}$ axis goes through nasion.
3. The $\boldsymbol{y}$ goes through LPA, orthogonal to $\boldsymbol{x}$
4. The $\boldsymbol{z}$ axis is orthogonal to both $\boldsymbol{x}$ & $\boldsymbol{y}$

fusionTrac 500 by Attarcsys along with marker is used to capture the points nasion, LPA, and RPA of the phantom head as shown in fig. 3.13b and the coordinate system is created using vector algebra.

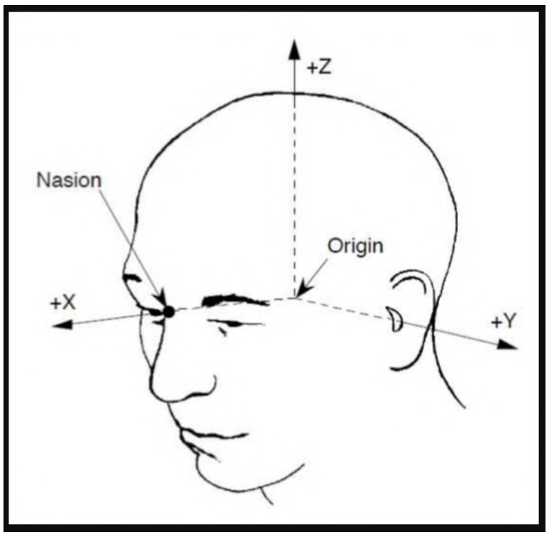
The points a, b, c are nasion, LPA, RPA respectively in the camera coordinate system, vectors $\boldsymbol{V}_a, \boldsymbol{V}_b, \boldsymbol{V}_c$ are created accordingly as depicted in 3.14a. Then $\boldsymbol{V}_a - \boldsymbol{V}_p$ and an unit vector is computed from point a to b.

$$\boldsymbol{n} = \frac{\boldsymbol{V}_b - \boldsymbol{V}_a}{\|\boldsymbol{V}_b - \boldsymbol{V}_a\|}$$

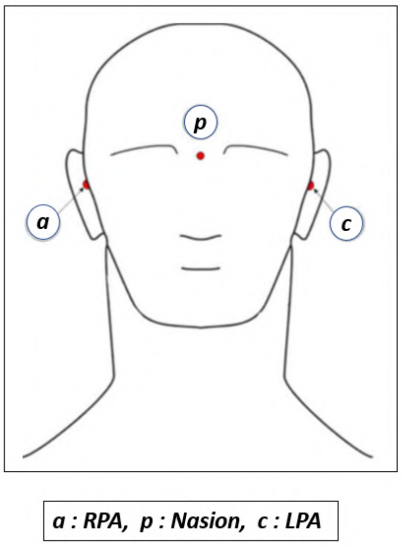
The vector $\boldsymbol{V}_a - \boldsymbol{V}_p$ is then projected on to the unit vector $\boldsymbol{n}$ along point a, b and vector $\boldsymbol{t}$ is computed.

$$Projection \;= ((\boldsymbol{V}_a - \boldsymbol{V}_p)n) * n)$$

$$\boldsymbol{t} = (\boldsymbol{V}_a - \boldsymbol{V}_p) - (((\boldsymbol{V}_a - \boldsymbol{V}_p)n) * n)$$



(a) The BTi/4D coordinate system convention.



(b) Anatomical features of the face.

Fig. 3.13: Head coordinate system.

we can calculate the origin of the coordinate system $\boldsymbol{o}$ as

$$\boldsymbol{o} = \boldsymbol{V}_p + \boldsymbol{t}$$

its evident that $\boldsymbol{x}$ axis is along $\boldsymbol{t}$ but in a opposite direction. It is important to normalise $\boldsymbol{x}$ after computation.

$$\boldsymbol{x} = -\boldsymbol{t} = -((\boldsymbol{V}_a - \boldsymbol{V}_p) - (((\boldsymbol{V}_a - \boldsymbol{V}_p)n) * n))$$

$$\hat{\boldsymbol{x}} = -\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|} \tag{3.1}$$

similarly $\boldsymbol{y}$ axis and its normalised version can be computed as,

$$\boldsymbol{y} = -(\boldsymbol{V}_b - \boldsymbol{o})$$

$$\hat{\boldsymbol{y}} = \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|} \tag{3.2}$$

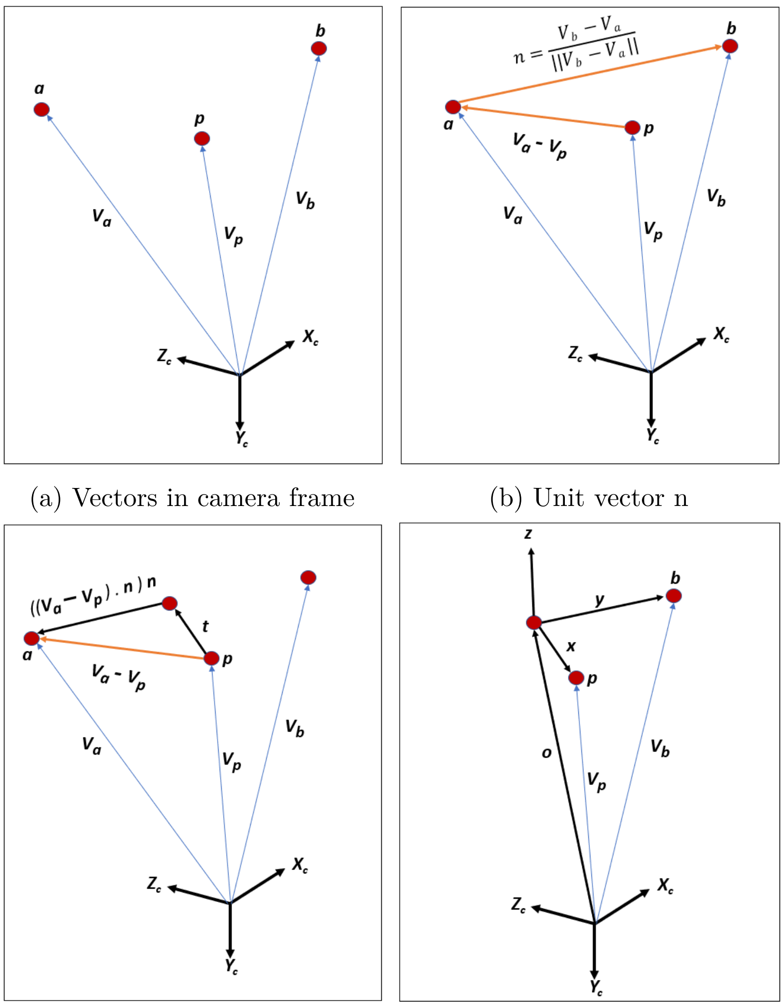
$\hat{\boldsymbol{z}}$ is a vector cross product of $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$

$$\hat{\boldsymbol{z}} = \hat{\boldsymbol{x}} \times \hat{\boldsymbol{y}} \tag{3.3}$$

A coordinate matrix can be created by combining $[\hat{\boldsymbol{x}}\ \hat{\boldsymbol{y}}\ \hat{\boldsymbol{z}}]$ also we can compute an orthonormal basis for this matrix using SVD.

3.5 Electrode mapping

Having created a head coordinate system, we now present strategies for mapping all the electrodes in the EEG cap. Please note that the process of head coordinate system



(a) Vectors in camera frame (b) Unit vector n

(c) Unit vector $\boldsymbol{n}$, $\boldsymbol{V}_b - \boldsymbol{V}_a$ (d) vector $\boldsymbol{t}$ & $((\boldsymbol{V}_b - \boldsymbol{V}_a).\boldsymbol{n}) * \boldsymbol{n}$

Fig. 3.14: Vector algebra in head coordinate system creation.

creation and electrode mapping are coupled and had to be done in the same setting without disturbing the head position but due to the limited working volume of the fusionTrac 500 camera as shown in fig. 3.15 it is evident that we cannot map all the electrodes in the cap without disturbing the head position especially the electrodes at the back of the head and this prompted to decouple the above 2 process.

Decoupling can be achieved by mapping both the head coordinate system and electrode's position to a common frame of reference the one which we always know the position of. Robot's end-effector frame was selected as a frame of reference as we can enquire about the end-effector position via MoveIt. First, we will map the already created head coordinate system to robot end-effector ${}^{\mathrm{EE}}\mathrm{T}_{\mathrm{H_{cs}}}$, let us recall the transformation between the camera and the robot's effector,

$$ {}^{\mathrm{EE}}\mathrm{T}_{\mathrm{H_{cs}}} = ({}^{\mathrm{Base}}\mathrm{T}_{\mathrm{EE}})^{-1}\,{}^{\mathrm{Base}}\mathrm{T}_{\mathrm{fCam}}\,{}^{\mathrm{fCam}}\mathrm{T}_{\mathrm{H_{cs}}} $$

where ${}^{\mathrm{Base}}\mathrm{T}_{\mathrm{EE}}$ is forward kinematics obtained via MoveIt, ${}^{\mathrm{Base}}\mathrm{T}_{\mathrm{fCam}}$ is evaluated via hand-eye calibration, ${}^{\mathrm{fCam}}\mathrm{T}_{\mathrm{H_{cs}}}$ is the head coordinate system created via fusionTrac 500

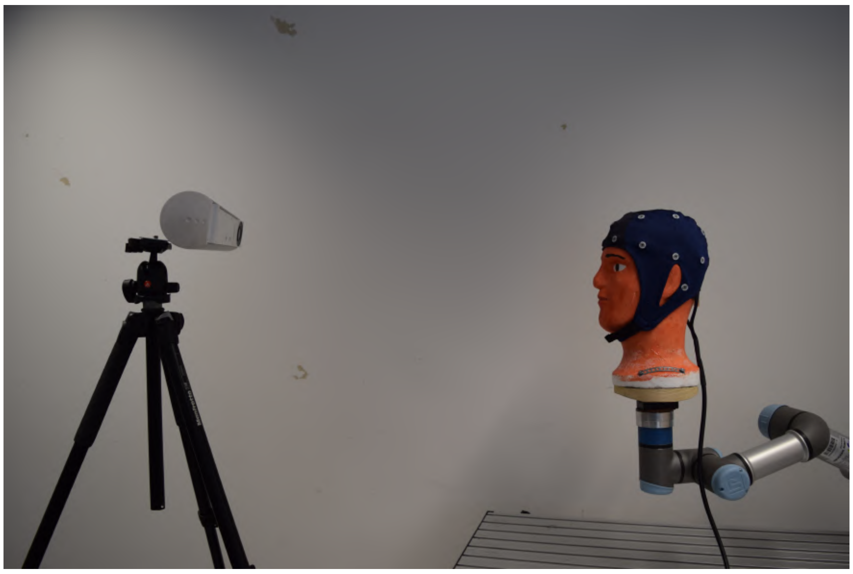


Fig. 3.15: Difficult to capture electrodes at the back of the phantom head.

camera. The same process is applied for mapping the electrodes also. The advantage now is the head can be moved by commanding the robot until all the electrodes are covered.

$$^{\mathrm{EE}}\mathrm{T}_{\mathrm{electrodes}} = \left(^{\mathrm{Base}}\mathrm{T}_{\mathrm{EE}}\right)^{-1} {}^{\mathrm{Base}}\mathrm{T}_{\mathrm{fCam}} {}^{\mathrm{fCam}}\mathrm{T}_{\mathrm{electrodes}}$$

Finally, all the electrodes can be mapped to head coordinate system via simple transformation.

$$^{\mathrm{H_{cs}}}\mathrm{T}_{\mathrm{electrodes}} = \left(^{\mathrm{EE}}\mathrm{T}_{\mathrm{H_{cs}}}\right)^{-1} {}^{\mathrm{EE}}\mathrm{T}_{\mathrm{electrodes}}$$

3.6 Robot movement

The phantom head wearing EEG is mounted to the robot and moved along spcified trajectories before recording the video in the Kinect camera. The fig. 3.16 shows the robot movement along the single axis. The phantom is rotated 360 degrees clockwise first and then anti-clockwise in order to fully capture all the electrodes in the video thereby in the RGB images.

3.7 Ground truth data generation

Having electrode mapping strategy figured out, we are set to generate ground truth data generation and recording in Kinect frame.

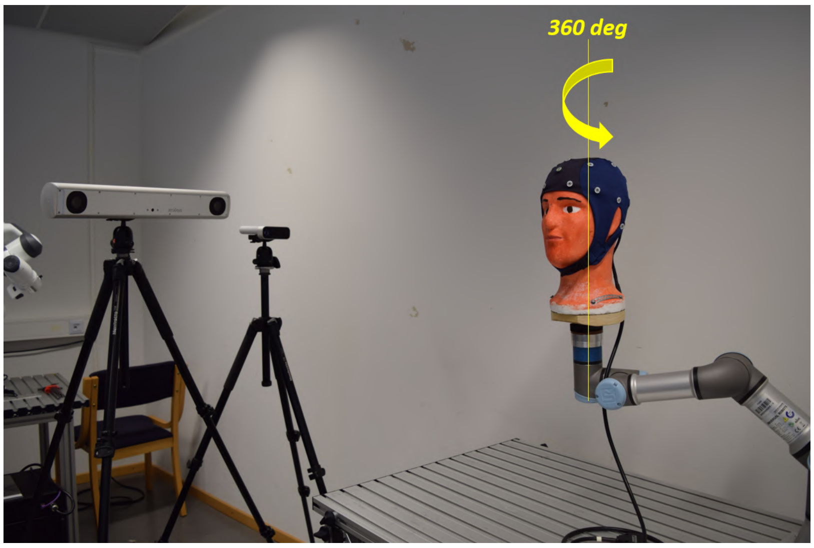


Fig. 3.16: 360 degrees movement of the phantom head.

Overview :
Here we would like to summarise all the steps involved in ground truth data generation.

1. Microsoft kinect :
   a) Camera calibaration using chessboard.
   b) Hand-eye calibtaion using chessboard.
2. Atracsys fusionTrac 500 :
   a) Hand-eye calibtaion using active marker.
   b) Head coordinate generation.
3. Ground truth data recording in Kinect:
   a) Move the robot in a predetermined trajectory and while moving, record below highlighted ROS topics.
      i. $/k4a/depth\_to\_rgb/camera\_info$.
      ii. $/k4a/depth\_to\_rgb/image\_rect$.
      iii. $/k4a/rgb/camera\_info$.
      iv. $/k4a/rgb/image\_rect\_color$.
      v. $/k4a/points2$.
      vi. $joints\_states$.

3.8 Accuracy of measurement

While using the reflective marker, we are only interested in the tip position of the marker which is independent of the orientation of marker geometry. However, we tried to evaluate the calibration of the marker tip relative to the marker geometry by measuring the same electrode with 10 different orientations of the reflective marker and the error is recorded.

Recall that we had to disturb the head position in order to map all the electrodes and we did it by introducing a common frame of reference i.e. robots end-effector frame at the cost of a measurement error. We also evaluated the positional error due to rotating head along a single axis and multiple axes to cover all electrodes. First, we rotated the phantom head to 10 different angles along a single axis while measuring the position of the single electrode at each of these 10 angles. Then, we repeated the same procedure but moved the phantom head along multiple axes as shown in fig. 3.17 and the positional error is recorded.

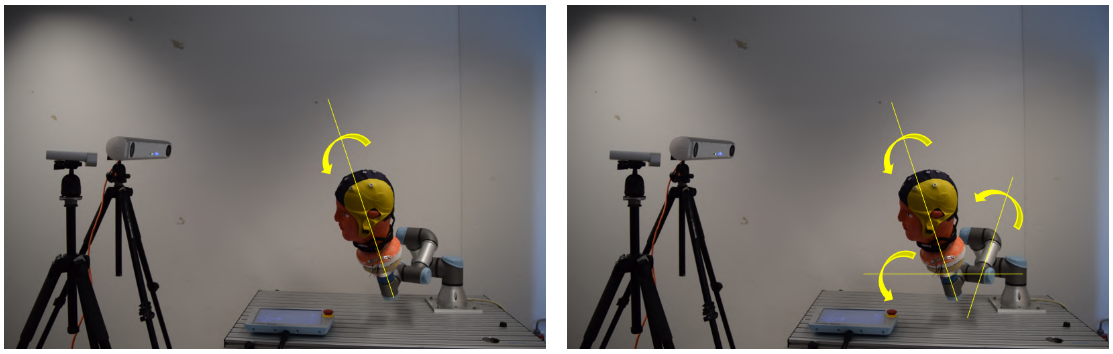


Fig. 3.17: Phantom head rotation along single and multiple axes.

4 Results

4.1 Camera calibration

10 RBG images were fed into the OpenCV algorithm for camera calibration. Calibration results along with the factory setting values are given in the below table.

Tab. 4.1: Microsoft azure kinect calibration result

Training set	calibrated values	Factory calibration settings
α_x	968.7	971.99
α_y	968.9	971.906
S	0.0	0.0
C_x	1026	1022.58
C_y	774.0	775.268
K_1	0.075 88	0.781 141
K_2	0.042 85	−3.0437
P_1	−0.000 64	0.000 616
P_2	0.001 32	-2.35×10^{-5}
K_3	−0.257 94	1.7224

4.2 Hand-eye calibration

We have achieved a high degree of accuracy in estimating 2 unknowns in the hand-eye calibration equation i.e camera position and marker position. Both Microsoft Kinect and Atracsys fusionTrac 500 hand-eye calibration resulted in positional error < 1.5 millimeters and rotational error < 0.3 degrees on a hold-out validation set as shown in the tab. 4.2.

Tab. 4.2: Microsoft azure kinect hand-eye calibration result

Camera	Translation error (m)	Rotational error (deg)
Kinect	$0.001 \pm 1.40 \times 10^{-7}$	0.258 ± 0.0
fusionTrac	$0.001 \pm 2.8 \times 10^{-7}$	0.205 ± 0.0

4.3 Head coordinate system

Head coordinate system is created with the 3 anatomoical feature of the face (RPA, nose & LPA) We can map above points to coordinate system to check whether we have right coordinate system. By observing the translational part expressed in head coordinate system ${}^{\mathrm{H_{cs}}}\mathrm{T}_{\mathrm{RPA}}$, ${}^{\mathrm{H_{cs}}}\mathrm{T}_{\mathrm{nasion}}$ & ${}^{\mathrm{H_{cs}}}\mathrm{T}_{\mathrm{LPA}}$ respectively it is indeed that and RPA, nasion, LPA make the 3 perpendicular axis. The fig. 4.1 shows these points graphically, RPA is at 56.5mm along -$\boldsymbol{y}$ axis, nasion is at 97.6mm along $\boldsymbol{x}$ axis, and LPA is at 74.6mm along $\boldsymbol{y}$ axis.

$$\begin{bmatrix} 0.0 \\ -0.0565 \\ 2.775 \times 10^{-17} \end{bmatrix}$$

$$\begin{bmatrix} 0.0976 \\ 2.775 \times 10^{-17} \\ 0.0 \end{bmatrix}$$

$$\begin{bmatrix} 0.0 \\ 0.0746 \\ 0.0 \end{bmatrix}$$

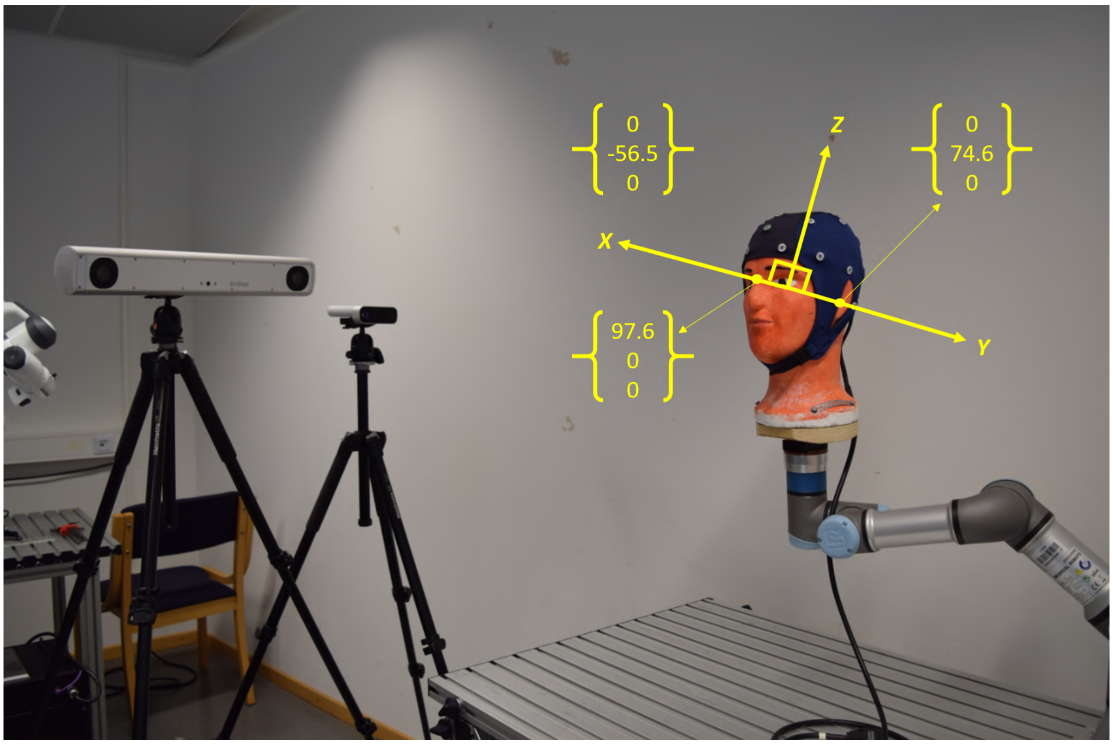


Fig. 4.1: Head coordinate system using RPA, nose and LPA

4.4 Accuracy of the data acquisition system

The fig. 4.2 shows the ten measurements of the same electrode with different marker orientation with mean value 137.05 ± 1 mm, therefore, we conclude that the measurement error due to the orientation of the marker is negligible.

The fig. 4.3 shows the ten measurements for single axis phantom rotation with mean value 118.26±4.5mm.

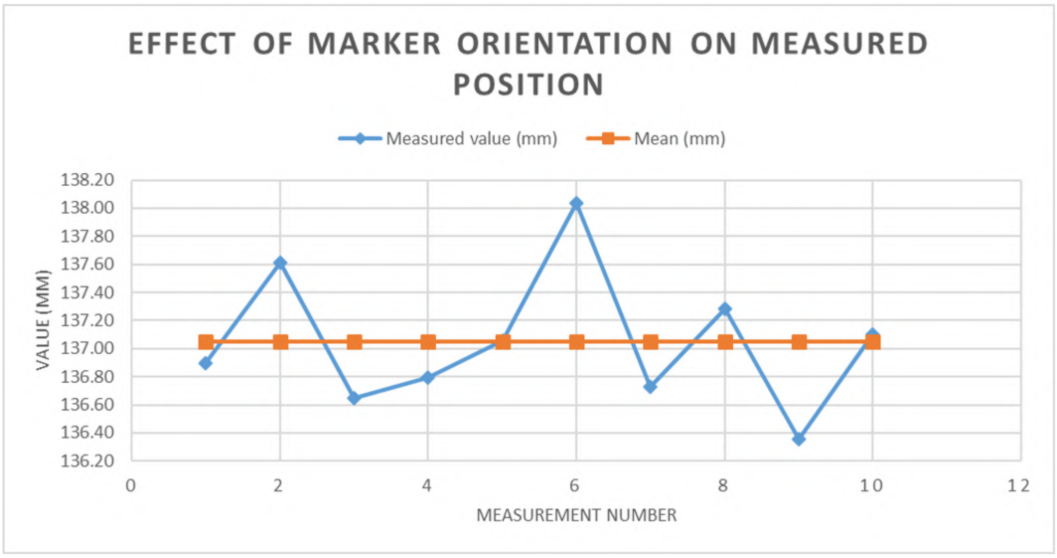


Fig. 4.2: Measurement of single electrode with ten different marker orientation

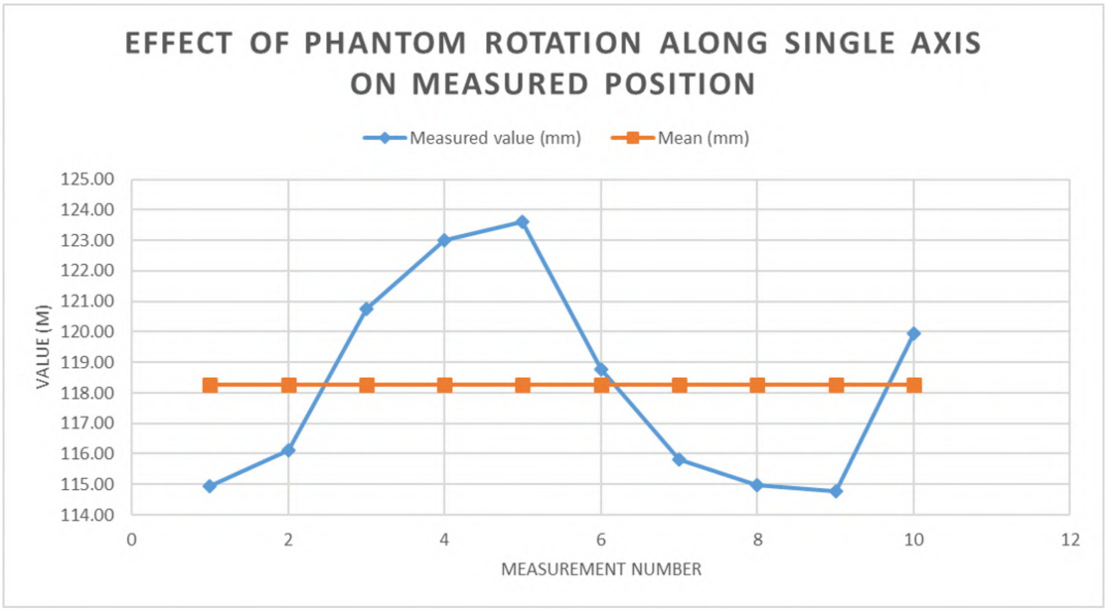


Fig. 4.3: Measurement of single electrode when phantom head roatated to ten different angles along an axis.

The fig. 4.4 shows the ten measurements for multiple axis phantom movement with 116.5± 1.5mm therefore, we conclude this robotic system can generate ground truth data with accuracy of ≈ ± 5mm.

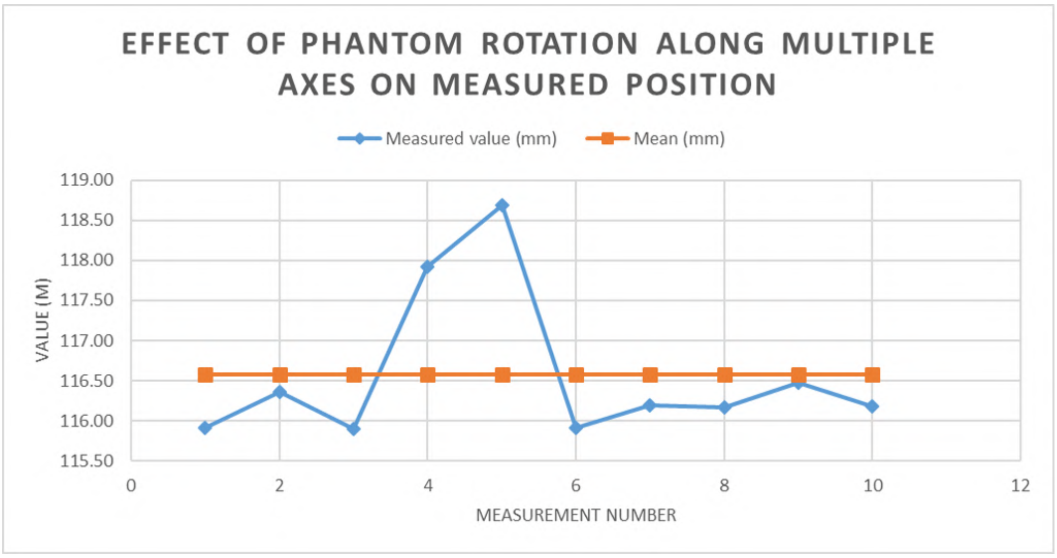


Fig. 4.4: Measurement of single electrode when phantom head roatated to ten different angles along multiple axes.

4.5 Range of electrode position variation

To evaluate the extent to which electrodes position varies at different wearing instances of the EEG cap, we tried to simulate attaching the EEG cap on the phantom for 5 times and measure the variance of the electrode position. fig. 4.5 and fig. 4.6 shows the electrode position at 5 such instances.

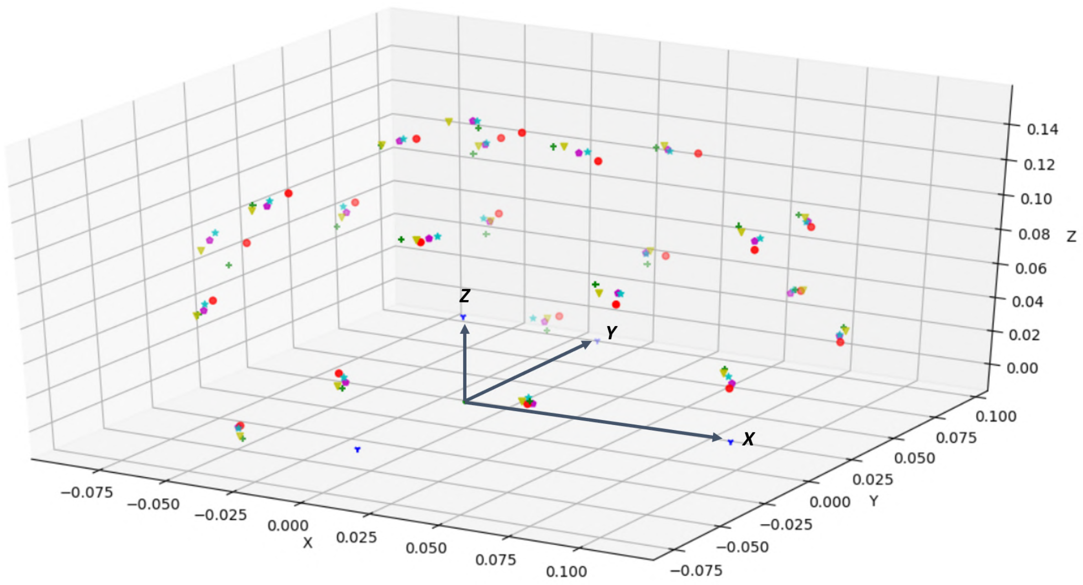


Fig. 4.5: 3D view of electrodes at 5 different EEG cap wearing instances

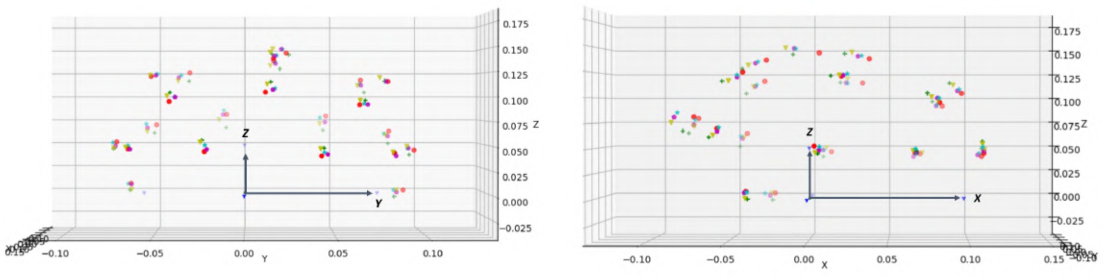


Fig. 4.6: 2D side views of electrodes at 5 different EEG cap wearing instances

The fig. 4.7 shows the range of electrode position variation during 5 electrode wearing instances to be with mean value 7.8 ± 5.8 mm.

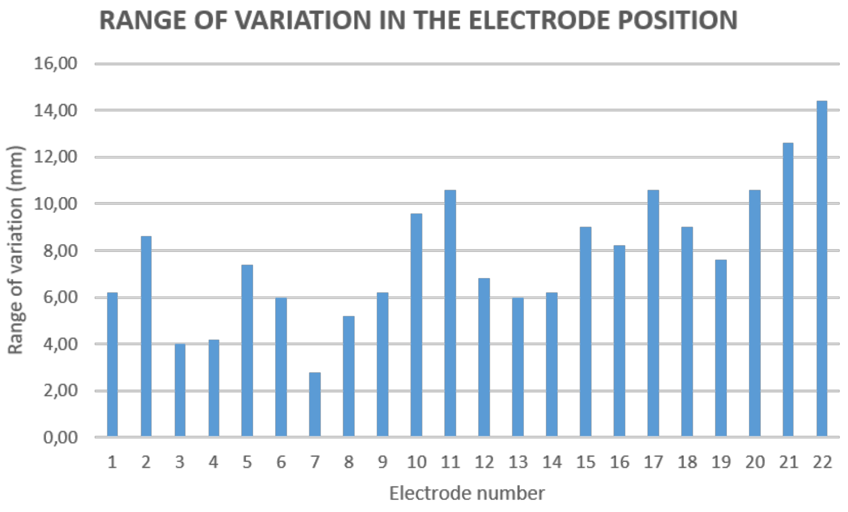


Fig. 4.7: Range of electrode position variation at 5 different EEG cap wearing instances

5 Discussion and Conclusion

We developed a robotic data acquisition system to generate ground truth data for the EEG cap electrode's position using the universal robotics UR-3 model, Microsoft azure Kinect, atracsys fusionTrac 500 camera. A robust software, integrating all the hardware has been implemented using open source software/libraries like ROS, MoveIt, and OpenCV, etc. We perform camera calibration, hand-eye calibration using a chessboard and a reflective marker before localizing the electrodes. A phantom head wearing the EEG cap is mounted on to a robot and a head coordinate system using anatomical features of the phantom face is created. While mapping the electrodes, the introduction of a common reference frame enabled us to map all the electrodes especially those at the back of the phantom head. We were able to cover all the electrodes at the cost of a measurement error of $\pm$5mm. This measurement error can be avoided by redesigning the reflective marker that can reach all the electrodes without having to move the phantom head. To evaluate the extent to which electrodes position varies at different wearing instances of the EEG cap, we generated data for 5 such instances. Finally, the phantom head is moved along specified trajectories using a robot while we record a video (30-45 seconds, 30fps) of the entire movement along with other important information. A Large amount of RGB images from the video can be used to train neural networks to detect electrodes by labeling all the electrodes in the images. Not all the electrodes can be seen in a single RGB image, on the other hand, the position of all the electrode is recorded in the Kinect frame even the ones that are not visible. This could pose a problem while creating a label to train the neural networks, therefore there is a scope for developing a filtering system to filter all the invisible electrodes positional information. These generated ground truth data can be used to evaluate any camera-based electrode detection and position estimation algorithms.

A Listings

A.1 Least square problem

There are 2 types of least square problems, non-homogenious $\mathtt{A}\boldsymbol{x} = \boldsymbol{b}$ (type I) and homogenious $\mathtt{A}\boldsymbol{x} = \mathbf{0}$ (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 $\min\limits_{x \in \mathbb{R}^n} \|Ax - b\|^2 = 0$ 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 $\min\limits_{x \in \mathbb{R}^n} \|Ax\|^2 = 0$ s.t. $\|x\| = 1$.
2. the solution $\boldsymbol{x}$ is the right nullspace of matrix $\mathtt{A}$ and is obtained from unit sigular 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 coulmn 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.

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

if SVD of $\mathtt{Q} = \mathtt{UDV}^{T}$, then Matrix $\mathtt{R}$ which satisfies above condition is $\mathtt{R} = \mathtt{UV}^{T}$ per [7].

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] P. M. R. Reis and M. Lochmann, “Using a motion capture system for spatial localization of eeg electrodes,” *Frontiers in Neuroscience*, vol. 9, 2015.

[4] N. Gessert, M. Gromniak, M. Bengs, L. Matthäus, and A. Schlaefer, “Towards deep learning-based eeg electrode detection using automatically generated labels,” in *CURAC 2019 Tagungsband*, Reutlingen, 2019, pp. 176–180, ISBN: 978-3-00-063717-9. [Online]. Available: `https://arxiv.org/abs/1908.04186`.

[5] 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).

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

[7] 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`.

[8] 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.

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

[10] ——, (1999). “MS Windows NT kernel description,” [Online]. Available: `https://en.wikipedia.org/wiki/Axis%E2%80%93angle_representation` (visited on 05/20/2020).

[11] 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).

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

[13] www.universal robots.com. (1999). “MS Windows NT kernel description,” [Online]. Available: `https://www.universal-robots.com/products/ur3-robot/` (visited on 05/20/2020).

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

[15] wiki.ros.org. (1999). “MS Windows NT kernel description,” [Online]. Available: `https://wiki.ros.org/srdf/review` (visited on 05/20/2020).

[16] fieldtriptoolbox.org. (1999). “MS Windows NT kernel description,” [Online]. Available: `http://www.fieldtriptoolbox.org/faq/how_are_the_different_head_and_mri_coordinate_systems_defined/#details-of-the-bti4d-coordinate-system` (visited on 05/20/2020).

Declaration

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

Datum: ....

(Unterschrift)