TUHH
Hamburg University of Technology

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

MTEC
Medical Technology | Science | Systems

A project paper 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: March 22, 2021

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

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, accurate 3D position of the electodes has to be known. Currently available EEG electrode localization 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 recent developments in the areas of high precision time of flight (TOF) depth cameras have led to the possibility of combing High resolution industrial camera and a TOF depth camera for the EEG electrode digitalization. One such system is used in [13] which captures images from five different perspectives (five fixed camera positions) color (RGB) and depth information is fused together to localize the EEG electrodes. The tab. 1.1 gives an overview of typical methods.

Variety of ododmetry based positioning techniques are available in the field of robotics. The techniques which relies on the 3D point clouds employ scan registration algorithms which finds the best transformation that aligns 2 subsequent point clouds (source and target). Especially in the era of autonomous driving, these algorithms serves as the basis for localizing the car and simultaneously build the map of the environment also to estimate the car's trajectory by fusing the data from variety of sensor suites (sensor fusion). The main scan registration technique is iterative closest point (ICP) algorithm introduced in [14] and many varients have been proposed over the years and selecting the appropriate parameters and configuration differs based on the problem. While using ICP, similar to any odometry systems, inherent accumulation of error at each scan registration leads to the drift in the trajectory estimation. Simultaneous localization and mapping (SLAM) technique is 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 probalistic form can be addressed via graph-based formulation by constructing a graph whose nodes represent the robot pose and the edge between 2 nodes act as a noisy odometry constraints and loop closure contraints can be added if the robot revisits the previously visited place. Solution to such a graph based SLAM invloves solving a large error minimization problem

Tab. 1.1: Comaparision 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)

[15].

In this thesis work, An inexpensive RGBD (color+TOF) sensor such as Microsoft Azure Kinect is guided around the phantom head with EEG cap in an specific trajectory while performing ICP registration between the scans for camera localization. The 3D point clouds for scan registration is generated by detecting electrodes on a RGB image using a state of the art convolutional neural network and corresponding depth value is added. A pre-Trained YOLO network for electrode detection is available as a previous step of the pipeline. The Generated point clouds are sparse as there are fewer electrodes on the EEG cap. A pose-graph is created on the fly whose nodes represent each camera pose along the trajectory. Output of the ICP scan registration is added as a edge connecting 2 subsequent nodes which serves as odmoetry constraints between 2 subsequent camera poses. Each camera pose along with the associated point cloud is added to the map which is being built simultaneously. Pose-graph is solved each time camera revisits the previously visited places and electrode map will be rendered with the corrected poses. Even after having contruscted the electrode map with the best estimates of the camera poses, several systematic errors will result in as many unlabelled clusters as the number of electrodes hence centroids of these clusters will be treated as the electorde position.

The above process is first applied to the dataset created by guiding the camera around phantom head using a 6 DOF robot where the camera motion is precisely controlled. Second part focusses on free hand guided trajectories. The objective of this thesis work is to develop algorighms to accuaratly estimate 3D position of the EEG electrodes by simulataneously localizing the camera which is either guided by a robot or by a hand and building an EEG electrode map based on scan registration and the pose-graph formulation.

2 Background

2.1 Camera fundamenals

A gentle introduction to the pinhole camera model, projection matrix, calibration (internal and external), the underlying mathematics.

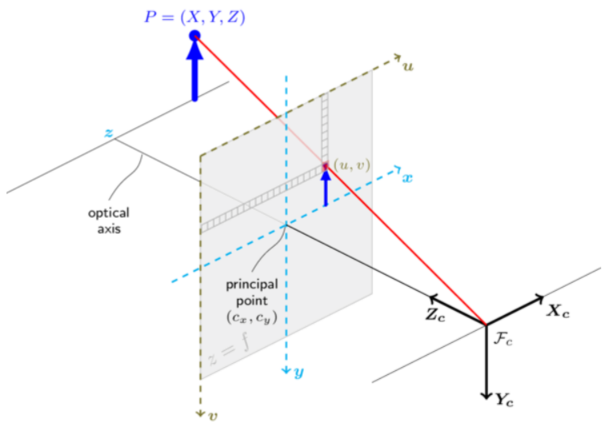

Fig. 2.1: Pinhole camera geometry. *$\boldsymbol{F_c}$ is the camera center, (C_x, C_y) is the principal point, $\boldsymbol{P}$ $(X, Y, Z)^T$ is world point, $\boldsymbol{P_c}$ $(X_c, Y_c, Z_c)^T$ is world point measured in camera coordinate system.[16]*

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) 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.

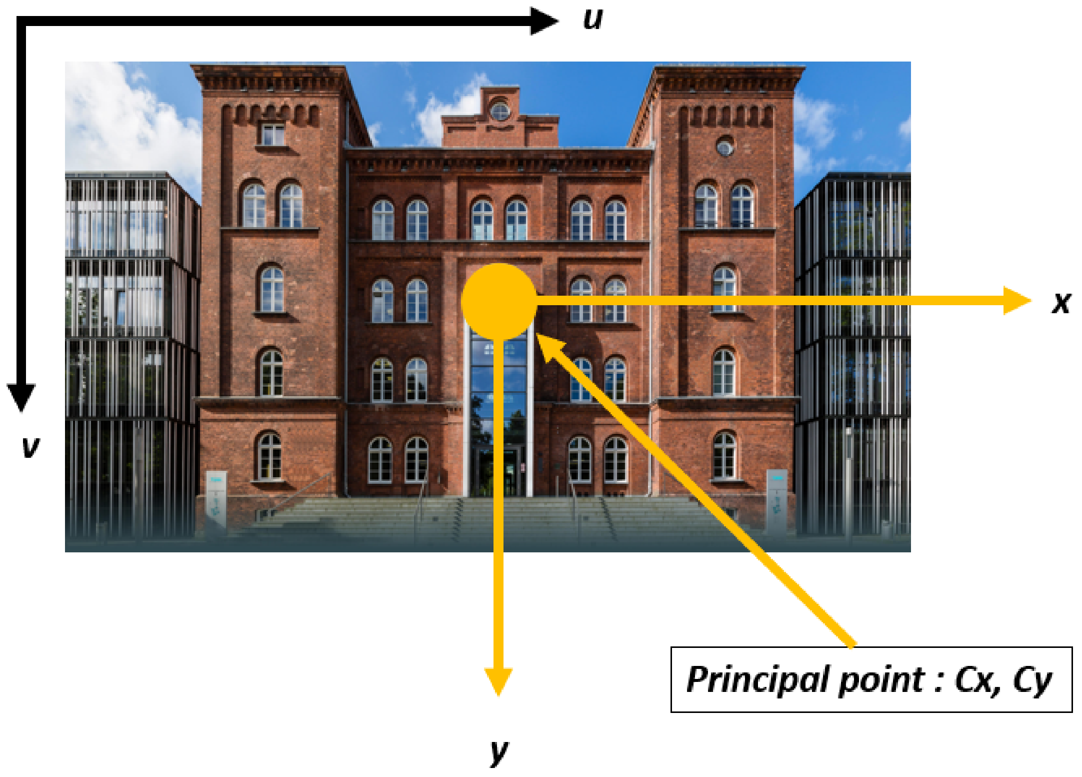

Fig. 2.2: Pixel coordinates (u, v) and the camera coordinates (x, y).

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).

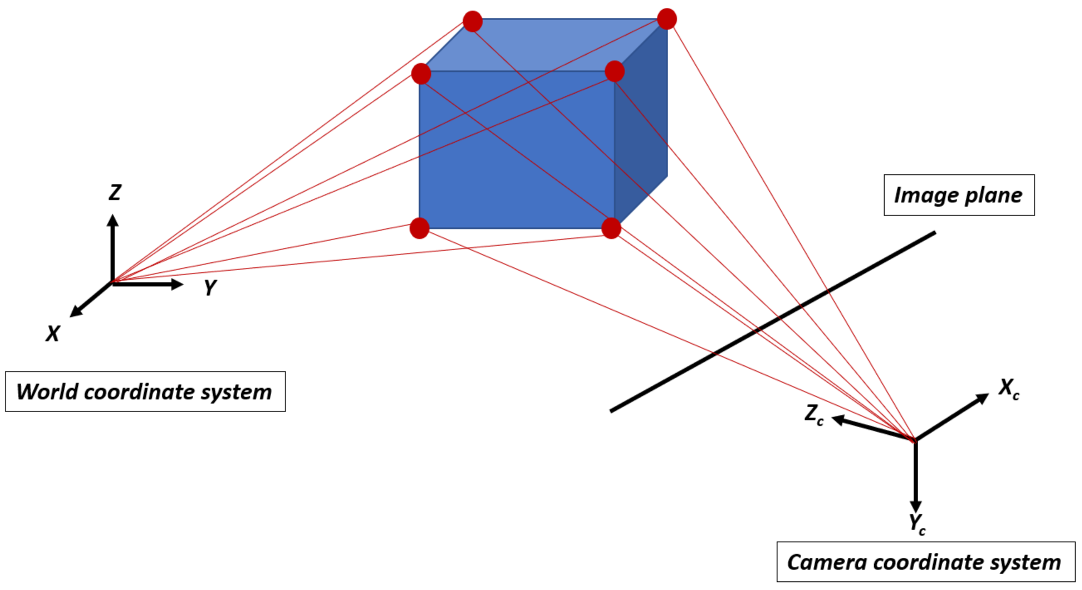

Fig. 2.3: Mapping 3D world points on to a 2D image plane.

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 $\boldsymbol{p}$.

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

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

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 [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

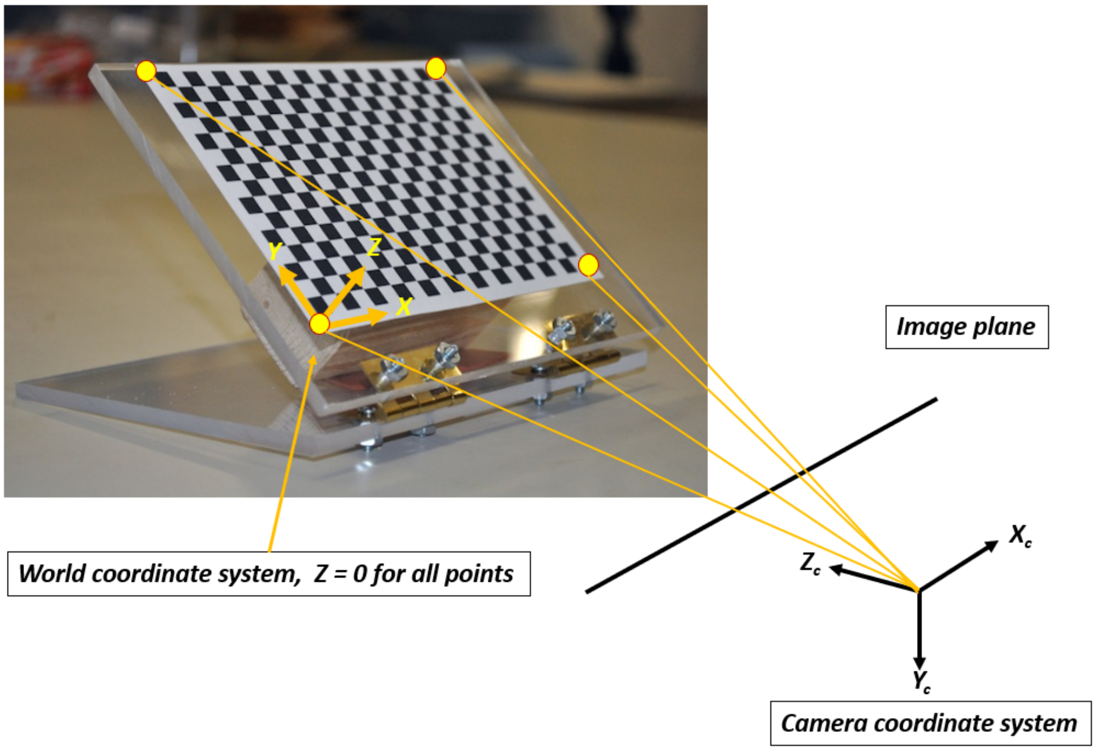

Fig. 2.4: Mapping 2D planar world points (Z = 0) on to a 2D image plane.

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 easy method to estimate the projection matrix thereby intrinsic and extrinsic camera parameters, 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

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

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,

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

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

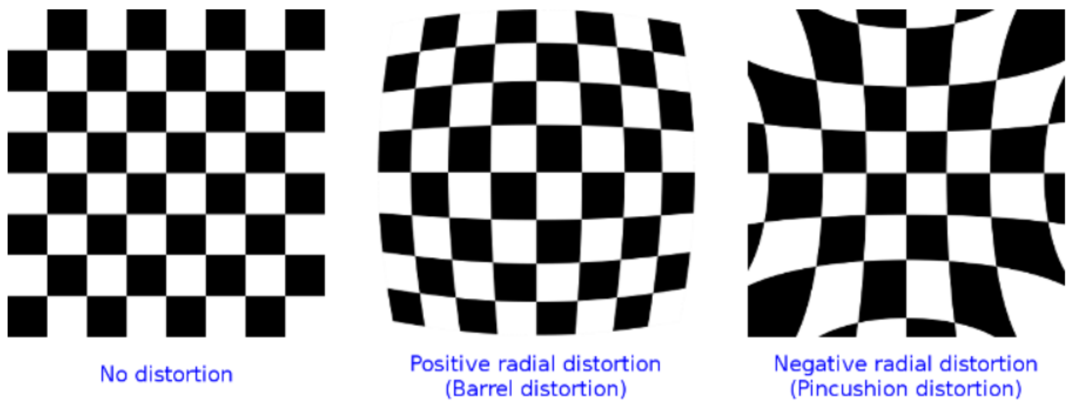

Fig. 2.5: Barrel distortion (*typically* $k_1 > 0$) and pincushion distortion ($typically k_1 < 0$) [16]

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 Camera calibration and pose estimation

Camera calibration is carried out using open-source computer vision library OpenCV [16]. 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. 2.6 and refines them accordingly.

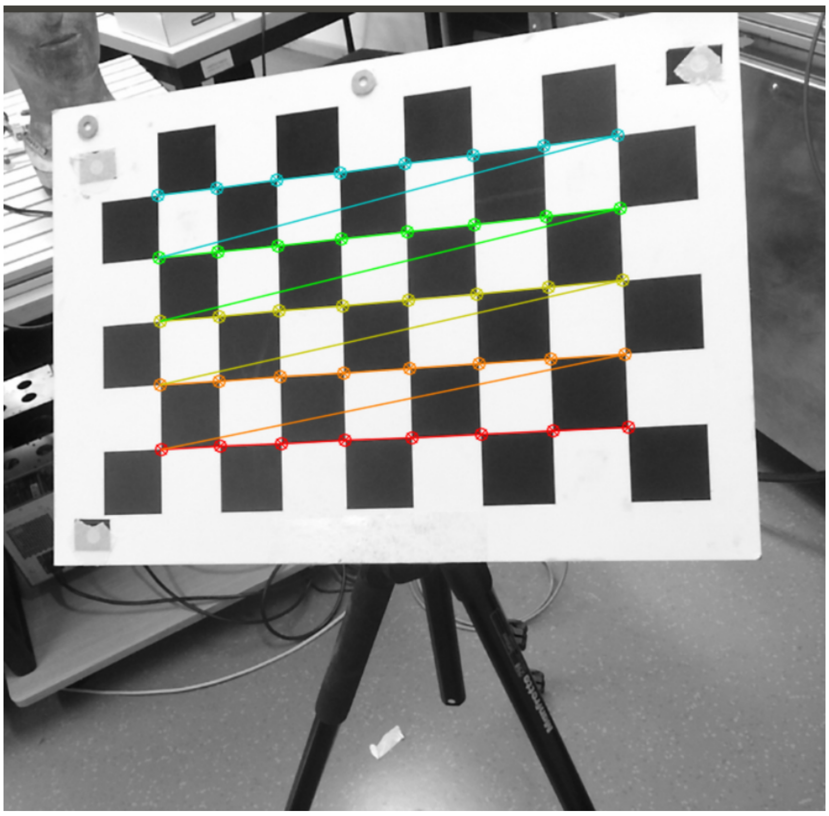

Fig. 2.6: Chessboard corners(image points) detection in OpenCV

```
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.

2.4 Pixel to 3D points

Having obatined 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. (2.1) a 3D world point $(X_c, Y_c, Z_C)^T$ is mapped to the pixel coordinates and can be expressed as a 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 rearanging the above equation $(X_c, Y_c, Z_C)^T$ can be expressed as below.

$$X_c = \frac{u - C_x}{\alpha_x} * \lambda$$

$$Y_c = \frac{v - C_y}{\alpha_y} * \lambda \tag{2.19}$$

$$Z_c = \lambda$$

$$TODO : spaceoutcorrectly$$

2.5 Hand-eye calibration

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

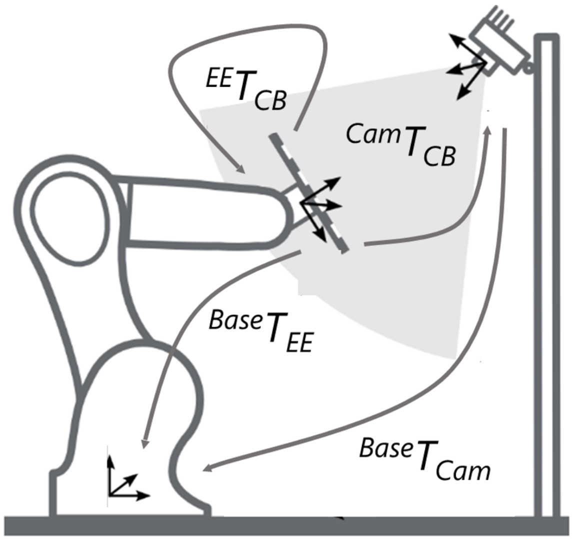

Fig. 2.7: Generic hand-eye calibration setup

Figure fig. 2.7 depicts the robot and camera set up required for hand-eye calibration. There are four transformations, namely end-effector to base $^{\text{Base}}\text{T}_{\text{EE}}$, chessboard to end-effector $^{\text{EE}}\text{T}_{\text{CB}}$, chessboard to camera $^{\text{Cam}}\text{T}_{\text{CB}}$ and camera to base $^{\text{Base}}\text{T}_{\text{Cam}}$. Defining new variables $^{\text{Base}}\text{T}_{\text{EE}}$ as M, $^{\text{EE}}\text{T}_{\text{CB}}$ as X, $^{\text{Base}}\text{T}_{\text{Cam}}$ as Y and $^{\text{Cam}}\text{T}_{\text{CB}}$ as N we can write the transformation equation as

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

per [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. (2.20) i.e the chessboard position with respect to the robot's end effector $\mathtt{X}$ and the camera position with respect to the robot's 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,

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

and

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

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 2.20 is a form of least square of type I (see appendix) and can be solved using QR factorisation [20] by minimizing quadratic error.

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

Where, $||.||_f$ is frobenoius norm. The above algorithm is called QR-24 according to [19].

Orthonormalisation Note that matrices computed not necessarily orthogonal therefore we need to orthonormalise 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 2.20 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 [19].

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

then, translation error can be defined as,

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

let (k,θ) be axis angle represention [21] of $\mathtt{R[A]}$ then the rotarional error metric is,

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

in the rotation metric the axis of rotation is neglected.

2.6 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). The main point cloud registration technique is iterative closest point (ICP) algorithm introduced in [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 correspondenses $C = \{(i, j)\}$.

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.

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

where, R is a 3X3 rotaion matrix, t is a 3X1 tranlation vector, and T[R|t] is a 4X4 homogeneous tranformation matrix.

Case 1: Known data association. where each point in source point cloud set p_i coressponds 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 as follows.

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,

$$\mathtt{E'(R)} = \sum_{(i,j)\in C} \|[q'_1, q'_2, ..., q'_n] - R[p'_1, p'_2, ..., p'_m]\|_F^2 \tag{2.25}$$

and is called orthogonal procrustes problem and can be solved by constructing cross-covariance matrix of 2 point cloud sets and SVD [22]. 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 the 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 3X3 rotation matrix R and 3X1 traslation vector t.

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

where, T[R|t] is the 4X4 homogeneous transformation between source and target point cloud and MSE is the mean square error of matching as in eq. (2.22).

Case 2: Unnown data association. when the correspondece 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 $P = \{p_1, p_2, ..., p_m\}$ coressponding closest point in the target point cloud $Q = \{q_1, q_2, ..., q_n\}$ is calculated before proceeding to the point cloud matching.

The Eucledian 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 corresponce 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 represetation is used here to enable matrix multiplication).

2.6.1 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.

2.7 Simultaneous Localization and Mapping

2.7.1 Map

Formally, a map m is a list of objects in the environment along with their properties [23]. Especially in this case, 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.

2.7.2 Localization

In simple words, localization is the pose estimation problem. It is the process of determining the pose of a camera with respect to the given map of the electrodes using wide vareity of sensor data i.e RGBD images and odometry etc. Localization is also the problem of establishing the correspondence between coordinate system based on which the map is defined (often, global coordinate system) and the camera's local coordinate system. This correspondence enables the camera to express the electrodes in the map in it's local coordinate system to be able to successfully navigate through the environment. 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 probablistic approaches to solve localization problems, the reader is encouraged to refer [23] for more details.

2.7.3 SLAM

Simultaneous localization and mapping (SLAM) is a special case where the camera neither have the pre-build map of the electrodes nor does it know its pose with respect to the map. Camera can however be moved along a trajectory either with a help of a robot or manually with in the environment and also collect RGB and depth images along the way. Finally, task is to build the map of the electrodes and simultaneously localize the camera with this map. Due to the fact that measurements are prone to noise, SLAM problems are modelled probabalistically.

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 percieved 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 map, $z_{1:t}$ and $u_{1:t}$ are measurement and control inputs untill the current time t respectively. Online-SLAM is essentially a online state estimation problem where 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 avaiable. Online-SLAM problem is is modelled and solved using filtering techninques such as Kalman and information filters [24] [25] and partical filters [26].

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 simulataneously mapping the electrodes by incororating full set of measurements and control inputs $z_{1:t}$ and $u_{1:t}$. Smoothing techniques like [27] [28] are employed to solve the full-SLAM problems. The modern and intuitive approach for solving 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 [27].

2.8 Graph-Based SLAM (Pose-Graph SLAM)

Graph-based approach uses an intuitive graph which 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 graph-based approach is to first construct the pose-graph and determine an optimised 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. 2.8.

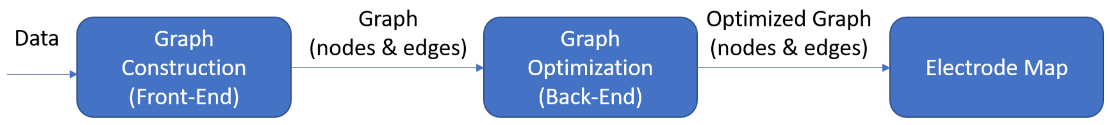
Data
Graph Construction (Front-End)
Graph (nodes & edges)
Graph Optimization (Back-End)
Optimized Graph (nodes & edges)
Electrode Map

Fig. 2.8: Graph based SLAM in a nutshell.

2.8.1 Pose-Graph construction

Graph consists of n nodes $x_{1:t}$ which corresponds to the camera pose at time t. Every time 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 camera revisits the previously visited position, for example x_j, an loop closure constraint is added to the graph connecting the current node x_t and x_j as shown in fig. 2.9.

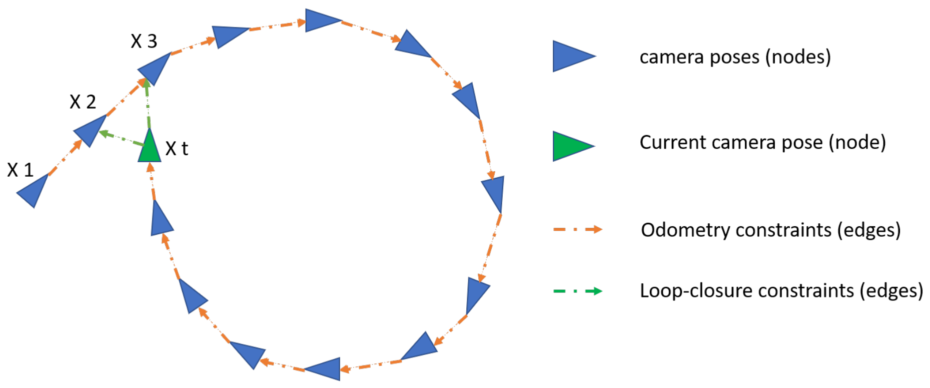
X 1
X 2
X 3
X t
camera poses (nodes)
Current camera pose (node)
Odometry constraints (edges)
Loop-closure constraints (edges)

Fig. 2.9: Pose-Graph consits of nodes and edges.

Since spatial constraints associated with edges (sensor measurement of some sort) are inherently noisy, these are modelled probabalistically. Let Z_{ij} and Ω be mean and information matrix of an 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. 2.10.

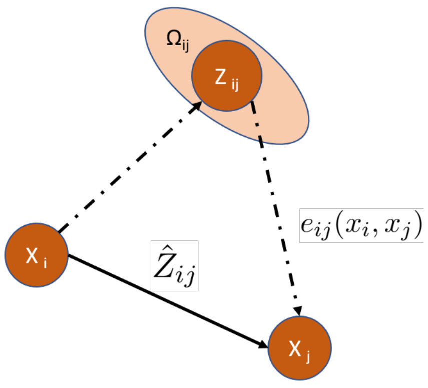

Fig. 2.10: Probabalistic model of measurements.

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 configuation 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,

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

2.8.2 Pose-Graph optimization

The eq. (2.33) is nonlinear least square problem and solution can be obtained using Gauss-Newton or Lavenberg-Marquardt alogorithms [15]. Dellaert and Kaess [28] have shown connection between factor graphs and nonlinear least square problems. A Factor graphs easily encodes probabalistic nature of SLAM and the sparsity of a SLAM problem can be easily visualised. A factor graph represents joint probabalities 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 fuction $h_i(\mathcal{X}_i$ and a measurement z_i. Assuming Gaussian measurement models f_i can be given as

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

with $\|e\|_\Sigma^2 = e^T \Sigma^{-1} e$ 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.

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

2.9 Electrode detection and localization

Electrodes captured in sequence of RGB images needs to be detected and simultaneously localised as shown in fig. 2.11 inorder to create 3D point clouds for sequential ICP registration and SLAM. YOLO(You Only Look Once) being one of 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 dataset of EEG cap and trained weights are available as a part of the pipileine.

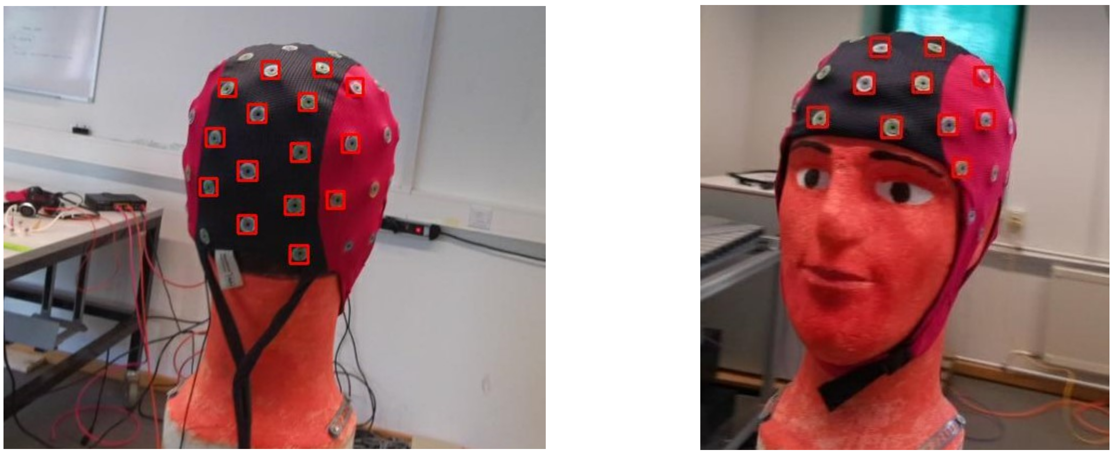

Fig. 2.11: Electrode detection and localization.

2.10 Cluster centers

A single electrode is seen from multiple adjacent camera poses and even after having contruscted the electrode map with the best estimates of the camera poses, errors associated with camera calibration, hand-eye calibration, YOLO localization, ICP drift, SLAM etc. will result in as many unlabelled clusters as the number of electrodes. Centroids of these clusters will be treated as the electorde position.

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 $\min\limits_{x \in \mathtt{R}^n} \|Ax - b\|^2 = 0$ 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 $\min\limits_{x \in \mathtt{R}^n} \|Ax\|^2 = 0$ 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.

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

if SVD of $\mathtt{Q} = \mathtt{UDV}^T$, then Matrix $\mathtt{R}$ which satisfies above condition is $\mathtt{R} = \mathtt{UV}^T$ per [18].

A.3 Gaussian distribution

A random variable possess probability density fucntions (PDF's) like one dimensional normal distribution fucntion 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 postive semidefinite and symmetric matrix called cavariance 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 cannocical form of PDF for multivariate normal distribution can be given by

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

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] 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.

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

[21] ——, (1999). "MS Windows NT kernel description," [Online]. Available: `https://en.wikipedia.org/wiki/Axis%E2%80%93angle_representation` (visited on 05/20/2020).

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

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

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

[25] 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.

[26] 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.

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

[28] 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.

Declaration

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

Datum: ....

(Unterschrift)