A COMPARATIVE STUDY ON POSE ESTIMATION ALGORITHMS USING VISUAL DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

GÜVEN ÇETİNKAYA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

FEBRUARY 2012

Approval of the thesis:

A COMPARATIVE STUDY ON POSE ESTIMATION ALGORITHMS USING VISUAL DATA

submitted by **GÜVEN ÇETİNKAYA** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. A. Aydın Alatan _____
Supervisor, **Electrical and Electronics Eng. Dept., METU**

Examining Committee Members:

Prof. Dr. Uğur HALICI _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. A. Aydın ALATAN _____
Electrical and Electronics Engineering Dept., METU

Asst Prof. Dr. Afşar SARANLI _____
Electrical and Electronics Engineering Dept., METU

Asst. Prof. Dr. Erol ŞAHİN _____
Computer Engineering Dept., METU

Dr. Erkan YAVUZ _____
Image Processing Department, ASELSAN, MGEO

Date: **February 9, 2012**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Güven ÇETİNKAYA

Signature :

ABSTRACT

A COMPARATIVE STUDY ON POSE ESTIMATION ALGORITHMS USING VISUAL DATA

Çetinkaya, Güven

M.Sc., Department of Electrical and Electronics Engineering

Supervisor : Prof. Dr. A. Aydın Alatan

February 2012, 108 pages

Computation of the position and orientation of an object with respect to a camera from its images is called pose estimation problem. Pose estimation is one of the major problems in computer vision, robotics and photogrammetry. Object tracking, object recognition, self-localization of robots are typical examples for the use of pose estimation.

Determining the pose of an object from its projections requires 3D model of an object in its own reference system, the camera parameters and 2D image of the object. Most of the pose estimation algorithms require the correspondences between the 3D model points of the object and 2D image points.

In this study, four well-known pose estimation algorithms requiring the 2D-3D correspondences to be known a priori; namely, *Orthogonal Iterations*, *POSIT*, *DLT* and *Efficient PnP* are compared. Moreover, two other well-known algorithms that solve the correspondence and pose problems simultaneously; *Soft POSIT* and *Blind-PnP* are also compared in the scope of this thesis study. In the first step of the simulations, synthetic data is formed using a realistic motion scenario and the

algorithms are compared using this data. In the next step, real images captured by a calibrated camera for an object with known 3D model are exploited.

The simulation results indicate that *POSIT* algorithm performs the best among the algorithms requiring point correspondences. Another result obtained from the experiments is that *Soft-POSIT* algorithm can be considered to perform better than *Blind-PnP* algorithm.

Keywords: 2D-3D Pose Estimation, 2D-3D Correspondence

ÖZ

POS KESTİRİM ALGORİTMALARININ GÖRSEL VERİLER KULLANILARAK KARŞILAŞTIRILMASI

Çetinkaya, Güven
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. A. Aydın Alatan

Şubat 2012, 108 sayfa

Bir nesnenin görüntülerini kullanarak o nesnenin kameraya gore pozisyon ve yöneliminin hesaplanmasına poz kestirim problemi denir. Poz kestirimi, bilgisayarla görü, robotik ve fotogrametri alanlarındaki önemli problemlerden birisidir. Nesne takibi, nesne tanıma, robotların kendi konumlarını bulması gibi uygulamalar poz kestirim probleminin kullanım alanlarına örnek olarak verilebilir.

Bir nesnenin pozunu, o nesnenin görüntülerini kullanarak bulmak için nesnenin kendi referans sisteminde üç boyutlu modeline, kamera parametrelerine ve iki boyutlu görüntüye ihtiyaç duyulmaktadır. Poz kestirim algoritmalarının çoğu üç boyutlu model noktaları ile iki boyutlu görüntü noktaları arasındaki ilişkinin tam olarak bilinmesine ihtiyaç duyarlar.

Bu çalışmada, önceden önerilmiş ve model noktaları ile görüntü noktaları arasındaki ilişkinin bilindiğini varsayan dört poz kestirim algoritması; *Orthogonal Iterations*, *POSIT*, *DLT* ve *Efficient PnP*, kodlanmış ve karşılaştırılmıştır. Ayrıca

poz ve eşleştirme problemlerini aynı anda çözen iki bilinen algoritmanın; *Soft-POSIT* ve *Blind-PnP*, kodlanması ve karşılaştırılması da bu çalışmada yer almaktadır. Simülasyonların ilkinde, gerçek hareket senaryoları kullanılarak sentetik veriler üretilmiş ve algoritmalar bu veriler kullanılarak karşılaştırılmıştır. Sonraki adımda, üç boyutlu modeli bilinen bir nesnenin kalibre edilmiş bir kamera ile çekilmiş gerçek görüntülerinden faydalanılmıştır.

Simülasyon sonuçlarına göre, model noktaları ve görüntü noktaları arasındaki ilişkinin bilindiğini varsayan algoritmalar arasında *POSIT* algoritmasının en iyi performansı gösterdiği görülmüştür. Deneyler sonucu elde edilen bir başka sonuç da *Soft-POSIT* algoritmasının *Blind-PnP* algoritmasına göre daha iyi bir performans gösterdiğidir.

Anahtar Kelimeler: 2B–3B Poz Kestirimi, 2B–3B Eşleştirme

To my family

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere thanks to my supervisor Prof. Dr. A. Aydın ALATAN, for his support, friendly attitude and encouragement at each stage of this thesis study.

I would like to thank ASELSAN, MGEO Image Processing Department for the support given throughout this study.

I would like to forward my appreciation to all my friends and colleagues for their continuous encouragement.

I would like to present my thanks to Erkan YAVUZ and O. Serdar GEDİK for sharing their knowledge.

I would like to express my appreciation to my friends Selçuk SANDIKÇI, Özgür YILMAZ and Ümit AYDIN for their valuable comments and support throughout the development of this thesis.

Finally, I like to express very special thanks to my dear family for their endless support and love throughout my education.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Definition and Motivation

This thesis study is dedicated to comparison of some well-known pose-estimation algorithms by using synthetic and real data. Pose is defined as the orientation and position of an object with respect to a camera. Pose estimation problem is the process of determining the pose of an object from its image. In order to calculate a pose at least a 2D image and a 3D object is required. Most of the algorithms also require some camera parameters and the correspondences between points on the 3D object and points on the 2D image.

Pose estimation is widely used in robotics applications. In [1], the visual servoing application of a robot arm is realized. The robot arm can grasp three boxes one by one and place each one on top of the other. In another application example, the pose of the robot and the opening angles of its arms can be calculated using monocular images [2].

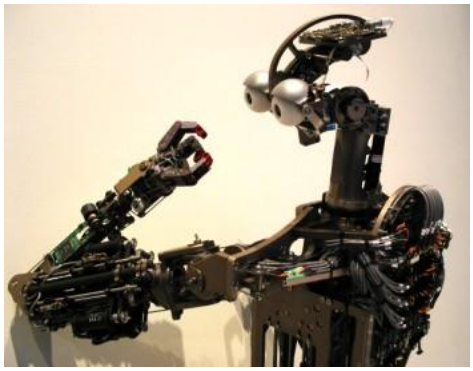

Figure 1 : Visual Servoing Application

Augmented reality applications also use pose estimation algorithms widely. In [3], a real-time tracking algorithm for helmet-mounted displays is proposed. The person wearing helmet-mounted display stands inside a box-like room surrounded by the cameras and the head movements of him are tracked by the system.

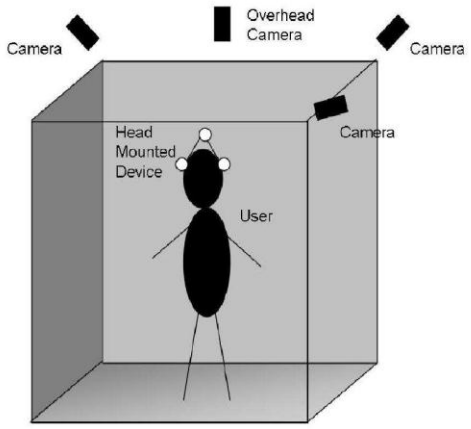

Figure 2 : Real-Time Optical Head Tracker Application [3]

The scope of this thesis is as follows: The algorithms compared in this thesis study use point features obtained from a single image. Comparison is performed according to pose accuracy and computational complexity of these algorithms. Visual and synthetic data are both utilized during the experiments for comparison.

Visual data is obtained by using images of infrared light emitting diodes (IR-LEDs) located on the head, whereas synthetic data is generated by using the realistic head motion data. Robustness of the algorithms against noise, occlusion and clutter cases are also investigated. The suitability of the algorithms for real time head tracking application is evaluated.

1.2 Related Work

There are different approaches to pose estimation in computer vision literature. The algorithms can be classified in many different groups.

One classification can be achieved according to the utilized features (points or lines) in the algorithm. Many algorithms use point features for pose estimation, but there are some algorithms [4, 5, 6] that also use line features. In this thesis study, algorithms using point features will be focused.

Algorithms can also be classified with respect to their input requirements. Most of the algorithms [7-21] require 2D – 3D point correspondences for pose calculation. Some of the algorithms [22, 23] calculate both pose and correspondences at the same time. There are also algorithms [24, 25, 26] that do not need 3D model of the object to be known. These algorithms use motion information to estimate the 3D structure and the pose of the object at the same time.

The computational method could be another issue during the classification of algorithms. Some algorithms [9, 11, 16] calculate pose from 2D-3D point correspondences iteratively. The other group consists of non-iterative algorithms [4, 7, 8, 10, 12, 13, 14, 15 17, 18] which again calculate pose from 2D-3D point correspondences.

The *DLT* algorithm [7], which is published in 1971, is the earliest study for pose estimation. It is not directly designed for pose estimation problem, but it can be used for calculation of transformations between 2D-2D or 2D-3D point pairs. Studies about *PnP* problem start in 1981 with [10]. The other methods proposed for *PnP* problem are [17, 20] in 1989, [19] in 1991, [13, 14] in 1999, [12] in 2001, [4, 18] in 2003 and [15] in 2008. Among these, [18, 13] are specialized for *P3P* problem and produce at most four solutions in general; and the others can handle arbitrary values of *n*. Algorithms for *DLT* and *PnP* problems are closed-form solutions and they are fast methods compared to iterative methods. But most of these algorithms are sensitive to noise and outliers.

First iterative algorithms for pose estimation problem used variances of optimization algorithms such as *Gauss-Newton* or *Gradient Descent*. [21, 19] in 1991 are the examples to these algorithms. In 1994, [5] came up with an algorithm based on constraints on image lines. *POSIT* algorithm [6], which was published in 1995, came up with an idea of using weak-perspective approximation for initialization. Lu *et.al.* published *Orthogonal Iterations* algorithm in 2000, and formulated pose estimation as minimizing an error metric based on collinearity in object rather than image. These iterative algorithms are more accurate than closed formed solutions and they are less sensitive to noise and outlier. But they typically suffer from slow convergence when they are badly initialized or convergence to local minima. For stability and accuracy, the iterative algorithms require more point correspondences compared to non-iterative ones.

First algorithms calculating pose and correspondences simultaneously worked out the problem by hypothesis and test approach such as RANSAC algorithm [10]. The problem is solved by hypothesizing small sets of 2D-3D correspondences between a number of 3D and 2D points. Using indexing methods such as *geometric hashing* is another method for determining pose and correspondences together. [27] in 1988 and [28] in 1993 are examples of this kind of algorithms. [29] in 1997 and [30] in

1999 are pose clustering approaches to calculating pose and correspondences together. [31] in 1995, [32] in 1996 and *Soft-POSIT* [23] in 2004 are algorithms which solve pose and correspondences iteratively by minimizing a global cost function. In 2008, *Blind-PnP* algorithm [22], which models the priors for camera pose with *Gaussian Mixture Model* [33] and refines them by hypothesizing new correspondences, was published.

The algorithms which are compared in this thesis study can be divided in two groups: algorithms that require point correspondences for pose estimation and algorithms which calculate pose and correspondences together. For the algorithms with known correspondences, two non-iterative and two iterative algorithms are selected. *DLT* algorithm is selected as the first non-iterative algorithm, because it is the simplest and oldest algorithm used in pose estimation. *Efficient PnP* algorithm is selected as the second non-iterative algorithm, because there is an immense study about *PnP* problem in the literature and it is the most recent among the proposed algorithms. *POSIT* algorithm is selected as one of the iterative algorithms, because it is one of the fastest and most accurate algorithms among the proposed algorithms. *ORTHIT* is selected as another algorithm with iterative nature, because of its accuracy and the different approach (minimizing errors in 3D space) used in pose calculation. Among the algorithms which calculate pose and correspondences simultaneously, *Soft-POSIT* is selected because it is considered as the most accurate, stable and fastest algorithm falling in this category. *Blind-PnP* is the other selected algorithm for comparison because it is one the most recent studies in this area.

1.3 Outline of the Thesis

In this thesis the second chapter is devoted to the background information; first, information about perspective projection and pinhole camera model are given. Then camera calibration and lens distortion topics are explained.

The third chapter contains detailed description of the algorithms compared in this thesis study. The chapter is divided in two parts. In the first part, algorithms with known point correspondences (*DLT*, *Efficient PnP*, *ORTHIT* and *POSIT*) are explained. In the second part, algorithms with unknown correspondences (*Soft-POSIT* and *Blind-PnP*) are detailed.

In the fourth chapter, information is given about the test environment used in the experiments. Later, the details about the experiments are given; finally, at the end of this chapter, the results of the experiments are given.

In the last chapter, the conclusion remarks of this study are presented.

CHAPTER 2

BACKGROUND TOPICS FOR POSE ESTIMATION

In this chapter, basic information about pose estimation problem such as perspective projection, camera calibration and lens distortion are given.

2.1 Perspective Projection

A camera can be considered as the mapping between a 3D object and its 2D image. Pinhole camera model is the simplest of other camera models and it does not take into account the lens distortion, de-focusing, blurring etc. Pinhole camera model describes the perspective projection.

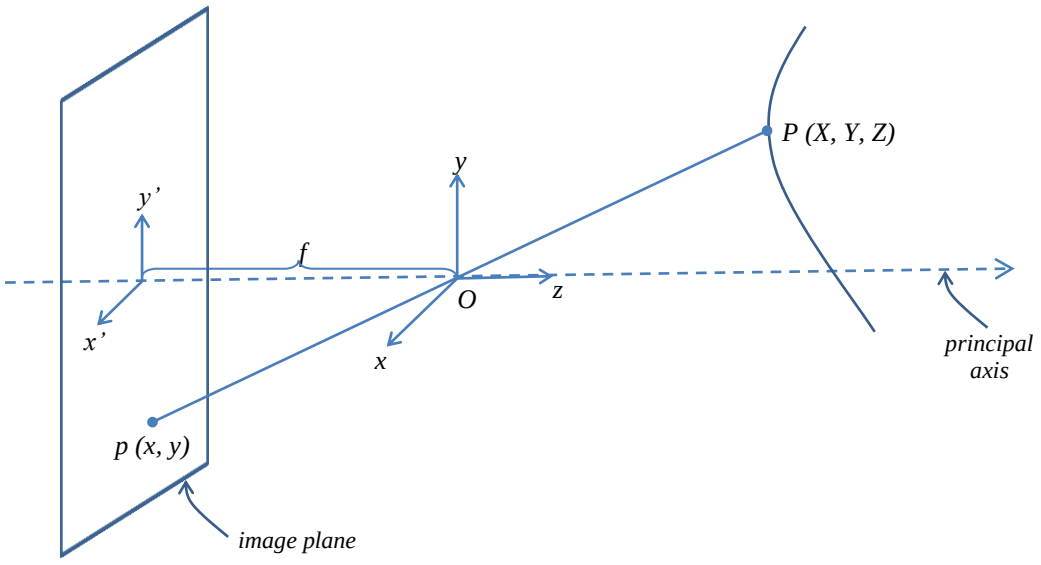

Figure 3 : Pinhole Camera Model

In Figure 3, pinhole camera model is shown. P is the 3D object point with coordinates (X, Y, Z) and p is its 2D projection onto the image plane with coordinates (x, y). The point O is the *camera center*. The distance between the camera center and image plane is called *focal length*. The relation between the coordinates of a 3D point and a 2D point is as shown below:

$$x = f\,\frac{X}{Z} \quad , \quad y = f\,\frac{Y}{Z} \tag{2.1}$$

The 2D image coordinates and 3D world coordinates can be represented in homogenous coordinates. The matrix representation of equation (2.1) in homogenous coordinates is shown below:

$$\begin{bmatrix} Zx \\ Zy \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

Let $\boldsymbol{x}$ represent the homogenous image coordinates vector and $\boldsymbol{X}$ represent the homogenous world coordinates vector. The given 3x4 matrix in (2.2) is called the homogenous *camera projection matrix* and can be represented as $\boldsymbol{P}$. Then equation (2.2) can be written as:

$$x = PX \tag{2.3}$$

In the equations for pinhole camera model above, the origin of image coordinates is considered as the center of image plane. However, for most of the cameras, this assumption is not exactly valid. If we represent the coordinates of center of image

plane (principal point) by (p_x, p_y), the generalized version of (2.2) can be written as in (2.4).

$$\begin{bmatrix} Zx \\ Zy \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.4}$$

In (2.4), the partial 3 x 3 matrix is represented as $\boldsymbol{K}$ and is called the *camera calibration matrix* [8].

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_x \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

In definition of pinhole camera model, the coordinates of 3D point is expressed in terms of *camera coordinate frame* where the origin is the camera center. However, in most applications, the object points are expressed in another reference system called as *world coordinate frame.* If one takes into account the rotation and translation relating the camera reference system and world reference system, the projection matrix for general pinhole camera model should be rewritten as:

$$P = K\,[R \mid t] \tag{2.6}$$

where K is the camera calibration matrix given in (2.5) and (R, t) gives the transformation between camera coordinate frame and world coordinate frames.

In digital cameras, the image plane is digitized in small regions called as pixels. In a pinhole camera model, the pixels are considered to be square. However, in some CCD cameras, the pixels can also be rectangular; hence the number of pixels in unit

length can be different along *x* and *y* directions. Another issue is that the pixels can sometimes be skewed (i.e. the *x*- and *y*- axis of pixels are not perpendicular to each other). Representing the number of pixels in x direction by m_x and the number of pixels in y direction by m_y and adding the effect of skew by *s* the camera calibration matrix in equation (2.5) can be modified as:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

In equation (2.7), $\alpha_x = f\, m_x$ and $\alpha_y = f\, m_y$ are focal length of camera in terms of pixel dimensions in *x*- and *y*- axis. $x_0 = m_x p_x$ and $y = m_y p_y$ are principal point coordinates in terms of pixel dimensions [8].

As a summary, the relation between a point in 3D world coordinates and its projection onto the image plane can be expressed by a 3 x 4 matrix called as *projection matrix (P)* in the form:

$$P = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} [R \mid t] \tag{2.8}$$

The projection matrix given in equation (2.8) has 11 unknowns: 5 for *K* matrix (α_x , α_y , s, x_0 *and* y_0), 3 for rotation matrix *R* (φ, θ, ψ) and 3 for translation (t_x , t_y ,t_z). The parameters for the camera calibration matrix *K* are called *internal (intrinsic) camera parameters.* The rotation and translation parameters are denoted as *external (extrinsic) camera parameters.* If intrinsic camera parameters are known for a camera, it is termed as a *calibrated camera.* The calibration parameters can be calculated using calibration tools as will be mentioned in Section 2.2. The extrinsic parameters should be calculated by using pose estimation algorithms.

2.2 Camera Calibration

Camera calibration is calculation of intrinsic parameters of a camera. Focal length, principal point and skew can be calculated using camera calibration algorithms. Camera calibration algorithms are beyond the scope of this thesis study; therefore, they will not be examined in detail. Instead, the *Camera Calibration Toolbox for MATLAB* which is used in calibration of test system's camera will be mentioned. The calibration algorithm implemented in this toolbox is described in [34]. For calibration, several images of a calibration pattern shown in Figure 4 are used.

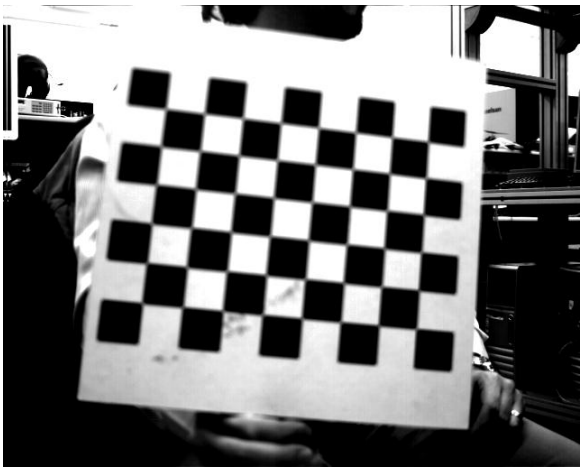

Figure 4 Calibration Pattern utilized during simulations

The calibration pattern is imaged from different view angles and distances. Approximately 20 images are used during calibration. The resulting parameters are obtained for the camera that is utilized during simulations and kept fixed during the whole tests:

Focal Length: $[\alpha_x \ \ \alpha_y]$ = [662.49534 664.67735]

Principal point: $[x_0 \ \ y_0]$ = [306.51289 241.75111]

Skew: s = [0.00000]

2.3 Lens Distortion

Pinhole camera model assumes that the projection of a 3D point falls on a straight line starting from 3D point, passing through the camera center and falling onto the image plane. However, for real cameras, lenses are used which cause deviations in their pixel locations. The deviations are caused by radial and tangential lens distortion. By knowing the non-linear relation between the ideal and distorted pixel locations, one might correct the effect of distortion [34]. Equations (2.9) and (2.10) describe the approximations for *radial distortion* and *tangential distortion* respectively.

$$\begin{bmatrix} \delta u_i^{(r)} \\ \delta v_i^{(r)} \end{bmatrix} = \begin{bmatrix} \tilde{u}_i \, (k_1 \, r_i^2 + \, k_2 \, r_i^4 + \cdots) \\ \tilde{v}_i \, (k_1 \, r_i^2 + \, k_2 \, r_i^4 + \cdots) \end{bmatrix} \tag{2.9}$$

where k_1, k_2, ... are coefficients for radial distortion and $r_i = \sqrt{\tilde{u}_i{}^2 + \tilde{v}_i{}^2}$.

$$\begin{bmatrix} \delta u_i^{(t)} \\ \delta v_i^{(t)} \end{bmatrix} = \begin{bmatrix} 2p_1 \tilde{u}_i \, \tilde{v}_i \, + \, p_2 \left(r_i^2 + 2 \, \tilde{u}_i{}^2 \right) \\ 2p_2 \tilde{u}_i \, \tilde{v}_i \, + \, p_1 \left(r_i^2 + 2 \, \tilde{v}_i{}^2 \right) \end{bmatrix} \tag{2.10}$$

There are many algorithms for estimating the parameters for distortion. However, in this thesis study, as in camera calibration process, the *Camera Calibration Toolbox for MATLAB* is used for calculating distortion parameters. The toolbox outputs five parameters for lens distortion. Three of these parameters (k_1, k_2 and k_5) are for radial distortion component and two of them (k_3 k_4) are for tangential distortion component. For the camera, utilized during simulations, distortion parameters calculated by the calibration pattern shown in Figure 4 are given as:

k_1 = -0.27908, k_2 = 0.32025, k_3 = 0.00050, k_4 = 0.00028 and k_5 = 0.00000

The distorted image coordinates x_d (u_i', v_i') and ideal image coordinates x_n $(\tilde{u}_i, \tilde{v}_i)$ can be expressed as follows: The relation between x_d and x_n is expressed in (2.11), where $\left(\delta u_i^{(r)}, \delta v_i^{(r)}\right)$ and $\left(\delta u_i^{(t)}, \delta v_i^{(t)}\right)$ can be calculated from (2.9) and (2.10) respectively [34].

$$\begin{bmatrix} u_i' \\ v_i' \end{bmatrix} = \begin{bmatrix} \tilde{u}_i + \delta u_i^{(r)} + \delta u_i^{(t)} \\ \tilde{v}_i + \delta v_i^{(r)} + \delta v_i^{(t)} \end{bmatrix} \tag{2.11}$$

CHAPTER 3

DETAILS OF THE ALGORITHMS

In this chapter, the algorithms compared in this thesis study are explained in more detail. The compared algorithms can be classified in two groups: First group of algorithms are point-based pose estimation algorithms with known point correspondences, which are: *Direct Linear Transformation (DLT)* [7], *Efficient Perspective-n-Point (EPnP)* [15], *Orthogonal Iterations (ORTHIT)* [9] and *Pose from Orthography and Scaling with Iterations (POSIT)* [11]. The second group of methods consists of point based algorithms that calculate pose and correspondences simultaneously, listed as *Simultaneous Pose and Correspondence Determination (Soft-POSIT)* [23] and *Blind-PnP* [22].

3.1 Algorithms with Known Correspondences

3.1.1 Direct Linear Transformation (DLT)

Direct linear transformation (DLT) algorithm aims to calculate the linear transformation between given 2D and 3D point matches. It is also used to find the homography between given 2D images. DLT algorithm is first mentioned in 1971 by Abdel-Aziz and Karara [7]. In this thesis study, the form of algorithm that is explained in the book of Hartley and Zisserman [8] will be investigated.

Assume that we are given a set of point correspondences between 2D image points and 3D world points. Let's denote the homogenous coordinates of each 2D image point by x_i *and* the homogenous coordinates of each 3D world point by X_i. The relation between x_i and X_i is given in (3.1), where the 3x4 matrix P is called the *projection matrix*.

$$x_i = PX_i \tag{3.1}$$

The equation (3.1) can be expressed in terms of the vector cross-product as $x_i \times PX_i = 0$. Denoting the elements of x_i as (x_i, y_i, w_i), this cross-product equation can be rewritten as in (3.2).

$$\begin{bmatrix} 0^T & -w_i X_i^T & y_i X_i^T \\ w_i X_i^T & 0^T & -x_i X_i^T \\ -y_i X_i^T & x_i X_i^T & 0^T \end{bmatrix} \begin{pmatrix} P^1 \\ P^2 \\ P^3 \end{pmatrix} = 0 \tag{3.2}$$

In equation (3.2), the term $P_i{}^T$ denotes a 4-vector which is the i^{th} row of P. The three equations of (3.2) are linearly dependent, so the first two of them can be taken as in (3.3) [8].

$$\begin{bmatrix} 0^T & -w_i X_i^T & y_i X_i^T \\ w_i X_i^T & 0^T & -x_i X_i^T \end{bmatrix} \begin{pmatrix} P^1 \\ P^2 \end{pmatrix} = 0 \tag{3.3}$$

For each point correspondences, one has 2 equations and the matrix in (3.3) has 2 rows and 12 columns. Hence, for n point correspondences, the matrix will be of size $2n$ x 12 and it can be denoted as A. The problem turns into the solution of a set of equations $Ap = 0$, where p is a vector that contains the elements of projection matrix P [8].

Since the projection matrix P has 12 elements and 11 degrees of freedom, at least 11 equations are required to solve (3.3). This requirement means that at least 5,5 correspondences (5 correspondences with *x*- and *y*- coordinates, single correspondence with only *x*- or *y*- coordinate) is strictly needed. Using this minimum number of correspondences with noiseless 2D and 3D coordinates, the solution will be exact. However, if the data is noisy, 6 correspondences will not be sufficient. If more than 6 correspondences are utilized, the solution is over-determined and can be obtained by minimizing $\|Ap\|$.

The *A* matrix is formed by using the image and world point correspondences. The solution for the vector p that contains the elements of projection matrix can be obtained by using the singular value decomposition of matrix *A*. In (3.4), the last column of vector *V* gives the solution.

$$A = UDV^T \tag{3.4}$$

In order to avoid the effects of arbitrary choices of coordinate systems and similarity transformation, it is better to normalize the inputs of DLT algorithm. For normalization, image points are translated, so that their centroid is at origin and scaled so that their root-mean-squared (RMS) distance from the origin is $\sqrt{2}$. Similarly for world points, their centroid is carried to origin and they are scaled to make their RMS distance from origin equal to $\sqrt{3}$. The result obtained by *DLT* algorithm using normalized inputs should be de-normalized [8].

DLT algorithm computes the internal calibration parameters of the camera and the pose (external parameters) at the same time. If internal camera calibration parameters are known, then these parameters can be inserted in the algorithm. This version of DLT algorithm is called *Clamped DLT* [15].

Typically, the output of DLT algorithm is given to an optimization algorithm minimizing the re-projection error (the error between image points and projection of world points). Levenberg-Marquardt or Gauss-Newton algorithms can be used for optimization. The Gold Standard algorithm given in [8] includes the normalization and optimization processes with DLT algorithm and it can be summarized as follows:

(i). Linear Solution: Compute an initial estimate of *P* by using a linear method.

a) Normalization: Normalize the 2D and 3D points using the similarity transformations *T* and *U* respectively as in (4.5)

$$\tilde{x}_i = Tx_i$$
$$\tilde{X}_i = UX_i \tag{4.5}$$

where, $\tilde{x}_i$ and $\tilde{X}_i$denotes normalized image points and normalized world points respectively.

b) DLT: Form the 2*n x* 12 matrix *A* by using the correspondences. Obtain SVD of *A* as $A = UDV^T$. The unit singular vector corresponding to the smallest singular value (the last column of *V* vector) is the required solution.

(ii). Minimize geometric error: Using the linear estimate obtained in step (i) as starting point minimize the error between the image points and the projection of world points (re-projection error) as depicted in (4.6). Levenberg-Marquardt algorithm can be used to minimize this error function. The obtained resulting $\widehat{P}$ is the projection matrix for the normalized coordinates.

$$\min_{P} \sum_{i} d\left(\tilde{x}_i, \hat{P}\tilde{X}_i\right)^2 \sum_{i} d\left(\tilde{x}_i, \widehat{P}\tilde{X}_i\right)^2 \tag{4.6}$$

iii. De-normalization: The projection matrix for the original coordinates can be calculated as below:

$$P = T^{-1}\,\widehat{P}\,U \tag{4.7}$$

where *T* and *U* are the similarity transformations used during normalization step.

3.1.2 Efficient Perspective-n-Point Algorithm (EPnP)

Perspective-*n*-Point (*PnP*) problem is calculation of pose of a calibrated camera from *n* 3D-2D point correspondences. In the literature there are many methods proposed for PnP problem [10, 13, 14, 17, 18, 19, 20]. At least three correspondences are required for pose calculation. *P3P* (*PnP* with three correspondences), produces four solutions to the pose problem. A unique solution can be found by adding one more correspondence (*P4P*). Detailed description about *P3P* algorithm is given in APPENDIX A.

Efficient PnP algorithm was published by Lepetit *et.al.* in 2008 [15]. It is a non-iterative solution and is claimed to be as accurate as iterative algorithms but much faster.

The object points are represented by the combination of four control points. By this way, the problem is reduced to calculate the coordinates of these four control points in camera coordinate system. Since the relation between control points and object points are known, rotation and translation of object points can be calculated. In (4.8), the relation between object points and control points is given.

$$X_i^w = \sum_{j=1}^{4} \alpha_{ij} C_j^w \quad with \quad \sum_{j=1}^{4} \alpha_{ij} = 1 \tag{4.8}$$

where;

X_i^w , *i=1,..., n* are the 3D coordinates of *object points* in world coordinate system.

C_j^w , *j=1,..., 4* are the 3D coordinates of *control points* in world coordinate system.

α_{ij} are homogenous Barycentric coordinates, uniquely defined and can easily be estimated.

The relation between reference points and control points is valid and same in both camera coordinate frame and world coordinate frame. The selection of control points may be arbitrary, but it is claimed that selecting one of the points as the centroid of object points and the other three points to form a basis aligned with the principal directions of object, would increase stability.

The relation between the object points and image points can be expressed as follows:

$$\forall i, \quad w_i \begin{bmatrix} x_i \\ 1 \end{bmatrix} = K X_i^c = K \sum_{j=1}^{4} \alpha_{ij} C_j^c \tag{4.9}$$

Here, x_i are 2D image points and X_i^c are the object points in camera coordinate system. K denotes the internal camera calibration matrix and w_i are scalar projective parameters. Equation (4.10) gives an open form of the formula in (4.9).

$$w_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} \quad \forall i, \tag{4.10}$$

In (4.18), f_x and f_y are focal length of camera in *x*- and *y*- axes respectively. (x_0, y_0) denotes the principal point. The unknowns in this equation are:

i. Coordinates of control points (12 unknowns)
ii. *N* projective parameters w_i (*N* is number of object points)

As it can be seen in (4.10), the projective parameters can be expressed as in (4.11).

$$w_i = \sum_{j=1}^{4} \alpha_{ij} z_j^c \tag{4.11}$$

Placing this expression in (4.10), two equations (4.12) and (4.13) can be obtained for each object point.

$$\sum_{j=1}^{4} \alpha_{ij} f_x x_j^c + \alpha_{ij} (x_0 - x_i) z_j^c = 0 \tag{4.12}$$

$$\sum_{j=1}^{4} \alpha_{ij} f_y y_j^c + \alpha_{ij} (y_0 - y_i) z_j^c = 0 \tag{4.13}$$

Equations (4.12) and (4.13) can be re-written in matrix form as in equation

$$Mc = 0 \tag{4.14}$$

where M is a $2n$ x 12 matrix containing the coefficients in (4.12) and (4.13), c is a vector containing the 12 unknown parameters. The solution belongs to the null space of the matrix M and can be expressed as in (4.15).

$$c = \sum_{i=1}^{N} \beta_i v_i \tag{4.15}$$

v_i are the eigenvectors corresponding to the N null singular values of M. The values of v_i can be calculated from M^TM matrix. The dimension of null-space of M^TM can vary from 1 to 4 depending on the configuration of the reference points, the focal length of the camera and the amount of noise. In the algorithm, solutions for all cases are calculated and the one producing the smallest re-projection error is chosen. N is fixed to 1, 2, 3 and 4 respectively and solutions for $\beta_i's$ are calculated for each value independently. After calculating $\beta_i's$, the coordinates of control points in camera coordinate frame can be calculated from (4.15). Then the coordinates of object points in camera coordinate frame can be calculated from control points. The absolute orientation between object coordinates in world coordinate frame and object coordinates in camera coordinate frame gives the rotation and translation between the object and the camera.

Efficient PnP algorithm can be summarized as follows:

i. Equations (4.12) and (4.13) are formed and M matrix is generated.
ii. The null eigenvectors of M^TM are calculated. For N= 1, 2, 3, 4 the values of $\beta_i's$ are calculated and the solution giving the smallest re-projection error is chosen.

iii. With known values of $\beta_i's$ and null eigenvectors, the coordinates of control points in camera reference system can be calculated from equation (4.15).

iv. The coordinates of object points in camera reference frame can be calculated from equation (4.8).

v. The coordinates of object points in both camera reference system and world reference system are available. Solving the absolute orientation between these two point sets gives the rotation and translation between the object and camera.

3.1.3 Orthogonal Iterations (ORTHIT)

Orthogonal Iterations algorithm is an iterative pose estimation algorithm that is proposed by Chien-Ping Lu in 2000 [9]. Most of the pose estimation algorithms deal with the problem in 2D domain, whereas Lu's algorithm tries to solve the problem in 3D domain. Starting from an initial pose, the object points in 3D coordinates are estimated from the 2D image points and the distances between the real 3D coordinates and these estimated 3D coordinates are tried to be minimized. This problem is known as *absolute orientation problem*.

Let's denote the 2D image points by x_i and 3D object points defined in object reference frame by X_i. Let $\widetilde{X}_i$ denote the object points defined in camera reference frame. The relation between X_i and $\widetilde{X}_i$ is as depicted in (4.16).

$$\tilde{X}_i = RX_i + t \tag{4.16}$$

where R is a rotation matrix and t is a translation vector. If i^{th} row of R is expressed by r_i^T and translation in x-, y- and z- coordinates is expressed by t_x, t_y and t_z, respectively, the relation between x_i and X_i is shown in (4.17).

$$x_i = \frac{1}{r_3^T X_i + t_z} (RX_i + t) \tag{4.17}$$

The *line-of-sight projection matrix* is the transformation that projects an object point to the line of sight defined by an image point. The line-of-sight projection matrix can be calculated as in (4.18).

$$V_i = \frac{v_i v_i^T}{v_i^T v_i} \tag{4.18}$$

The orthogonal iterations algorithm uses the fact that, the orthogonal projection of $\widetilde{X}_i$ on x_i must be equal to itself [9]. The problem is reduced to find the optimal rotation matrix (R) and translation vector (t) that minimizes the distance between object points in camera reference frame and their orthogonal projection on the image points. The error function is shown in (4.19).

$$e_i = (I - V_i)(RX_i + t) \tag{4.19}$$

The orthogonal iterations algorithm calculates R matrix and t vector that minimizes the squared error as depicted in (4.20).

$$E(R,t) = \sum_{i=1}^{n} \|e_i\|^2 = \sum_{i=1}^{n} \|(I - V_i)(RX_i + t)\|^2 \tag{4.20}$$

The solution of the least squares problem defined in (4.20) can be obtained by using singular value decomposition. The *sample cross-covariance matrix (M)* between X_i and $\widetilde{X}_i$ is calculated by

$$M = \sum_{i=1}^{n} \tilde{X}_i{}' \, Xi'^{t} \tag{4.21}$$

where $\tilde{X}_i{}'$ denotes the object points defined in camera reference system with centroids subtracted and Xi' is the object points defined in object reference system and centroid is subtracted. The solution to the minimization problem in (4.20) is identical to finding the rotation matrix maximizing $tr(R^T M)$. The solution can be obtained using the singular value decomposition of M that is $M = UDV^T$. Rotation matrix can be calculated by (4.22).

$$R = VU^T \tag{4.22}$$

The translation can be calculated from R by (4.23).

$$t(R) = \frac{1}{n}\left(I - \frac{1}{n}\sum_j V_j\right)^{-1} \sum_j \left(V_j - I\right) R\, X_j \tag{4.23}$$

The summary of the *Orthogonal Iterations* algorithm is as follows:

i. Using an initial estimate of R, calculate t from (4.23).
ii. From (4.16), calculate the object points in camera reference frame.
iii. Calculate the sample cross-covariance matrix from (4.21) and find SVD of M.
iv. Calculate the next estimate of R using (4.22).

This is an iterative process and the iterations are performed until the squared error in (4.20) falls below a threshold or no improvement occurs between two consecutive iterations.

3.1.4 Pose from Orthography and Scaling with Iterations (POSIT)

POSIT is a pose estimation algorithm that is first published by Daniel F. DeMenthon *et.al.* in 1995 [11], and is an iterative algorithm that requires at least four correspondences between 2D image points and 3D object points. *POSIT* algorithm consists of two steps. The first step is "*Pose from Orthography and Scaling (POS)"* algorithm that calculates rotation and translation using linear equations from scaled-orthographic projection approximation. The second step iteratively refines the rotation and translation calculated by the POS algorithm. One of the main properties of *POSIT* algorithm that differs it from other iterative algorithms is that it does not require an initial estimate of the pose.

Scaled orthographic projection (also known as *weak perspective projection*) is an approximation to the perspective projection that assumes the distance between the object points is much smaller than the distance of object from the camera center. In scaled orthographic projection, the points are assumed to be on the same plane parallel to the image plane. The depth (distance from camera center in *z*- axis) of all the object points is assumed to be fixed and equal to Z_0. Then the relation between image points and object points can be obtained for scaled orthographic projection as below:

$$x' = f\,\frac{X}{Z_0} \quad , \quad y' = f\,\frac{Y}{Z_0} \tag{4.24}$$

where $s = f/Z_0$ is a constant called "*scaling factor"*.

The POSIT coordinate system is shown in Figure 5. X_0 is the center of object and X_i is a point on object. x_0 is the perspective projection of X_0 *and* x_i' is the scaled

orthographic projection of X_i. G is the image plane and K is the plane where all the object points are assumed to be on for scaled orthographic projection. (w-, v-, u-) are the axes of object coordinate system and (x-, y-, z-) are the axes of camera coordinate system.

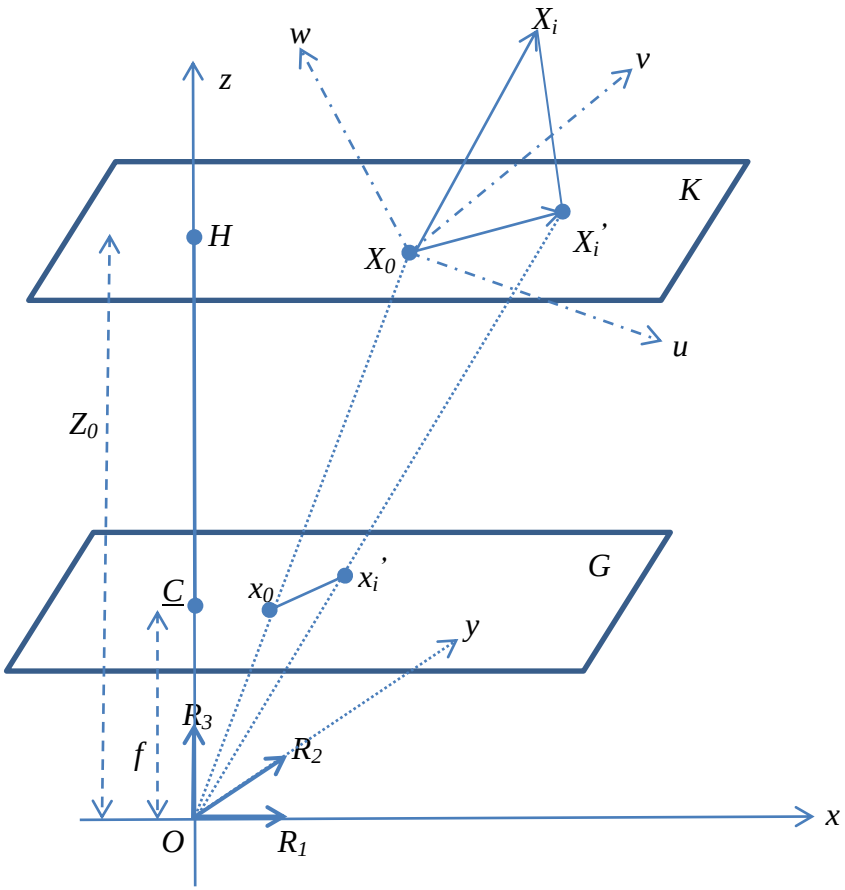

Figure 5 : *POSIT* Coordinate System

The rotation matrix can be written as (4.25).

$$R = \begin{bmatrix} R_{1u} & R_{1v} & R_{1w} \\ R_{2u} & R_{2v} & R_{2u} \\ R_{3u} & R_{3v} & R_{3w} \end{bmatrix} = \begin{bmatrix} R_1^T \\ R_2^T \\ R_3^T \end{bmatrix} \tag{4.25}$$

The coordinates of 3D object point X_i with respect to object coordinate system is (X_0u, X_1u, X_2u). If R_1 and R_2 are obtained, R_3 can be calculated from the cross-product of R_1 and R_2 (R_1 x R_2). X_0O is the vector between the origin of object coordinate system and center of projection and it defines the translation vector T.

The relation between object points and image points can be written as in equation (4.26). The parameter w is the scale factor, whereas f *is* the focal length.

$$\begin{bmatrix} wx_i \\ wy_i \\ w \end{bmatrix} = \begin{bmatrix} fR_1^T & fT_x \\ fR_2^T & fT_y \\ R_3^T & T_z \end{bmatrix} \begin{bmatrix} X_0X_i \\ 1 \end{bmatrix} \tag{4.26}$$

If both sides of (4.26) are multiplied by $\left({}^{1}/_{T_z}\right)$ and letting $s = {}^{f}/_{T_z}$, then the resulting relation is obtained:

$$\begin{bmatrix} wx_i \\ wy_i \end{bmatrix} = \begin{bmatrix} sR_1^T & sT_x \\ sR_2^T & sT_y \end{bmatrix} \begin{bmatrix} X_0X_i \\ 1 \end{bmatrix} \tag{4.27}$$

where w in (4.27) can be calculated from (4.28), and $R_3 \cdot X_0X_i$ is the projection of X_0X_i onto the optical axis.

$$w = {}^{R_3 \cdot X_0X_i}/_{T_z} + 1 \tag{4.28}$$

When the depth between object points is quite small compared to the distance between camera center and object, w value can be estimated as 1. This is the *scaled orthographic projection* (equation (4.29)). When s value is equal to 1, this case is called "*orthographic projection*".

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} sR_1^T & sT_x \\ sR_2^T & sT_y \end{bmatrix} \begin{bmatrix} X_0 X_i \\ 1 \end{bmatrix} \tag{4.29}$$

Equation (4.29) gives a set of linear equations where the image point and object point coordinates are known parameters, whereas sR_1, sR_2, sT_y *and* sT_z are unknown. After solving this linear equation, we can calculate rotation (R_1, R_2, R_3) and translation (T_x, T_y, T_z) from (4.30)

$$\begin{gathered} s = \sqrt{|sR_1||sR_2|}, \\ R_1 = {}^{(sR_1)}/_{s}, \quad R_2 = {}^{(sR_2)}/_{s}, \qquad R_3 = R_1 \times R_2 \\ T_X = {}^{(sT_X)}/_{s}, \quad T_y = {}^{(sT_y)}/_{s}, \qquad T_z = {}^{f}/_{s} \end{gathered} \tag{4.30}$$

The general perspective equation can be obtained as:

$$[X \quad Y \quad Z \quad 1] \begin{bmatrix} sR_1 & sR_2 \\ sT_y & sT_y \end{bmatrix} = [wx \quad wy] \tag{4.31}$$

The summary of *POSIT* algorithm is as follows:

i. Initially set *w* to 1.
ii. Estimate the pose by solving (4.29) and (4.30).
iii. Update *w* by (4.28).
iv. If improvement of *w* is smaller than a threshold or maximum iteration number is reached, stop iterations; else, go back to step (ii).

3.2 Pose Estimation Algorithms with Unknown Correspondences

3.2.1 Simultaneous Pose and Correspondence Determination (Soft-POSIT)

Soft-POSIT algorithm is published by Philip David, *et.al.* in 2004 [23]. It is a pose estimation algorithm that requires an initial estimate of the pose and solves both pose and correspondence problems at the same time. *Soft-POSIT* algorithm combines two iterative algorithms: Gold's *SoftAssign* algorithm [35] for the correspondence problem and *POSIT* algorithm [11] for the pose estimation problem.

The details of *POSIT* algorithm are already given in Section 3.1.4. In this section, the *POSIT* algorithm is investigated in terms of geometry and derivation of an objective function is given. In Figure 6, another illustration of scaled orthographic projection is given.

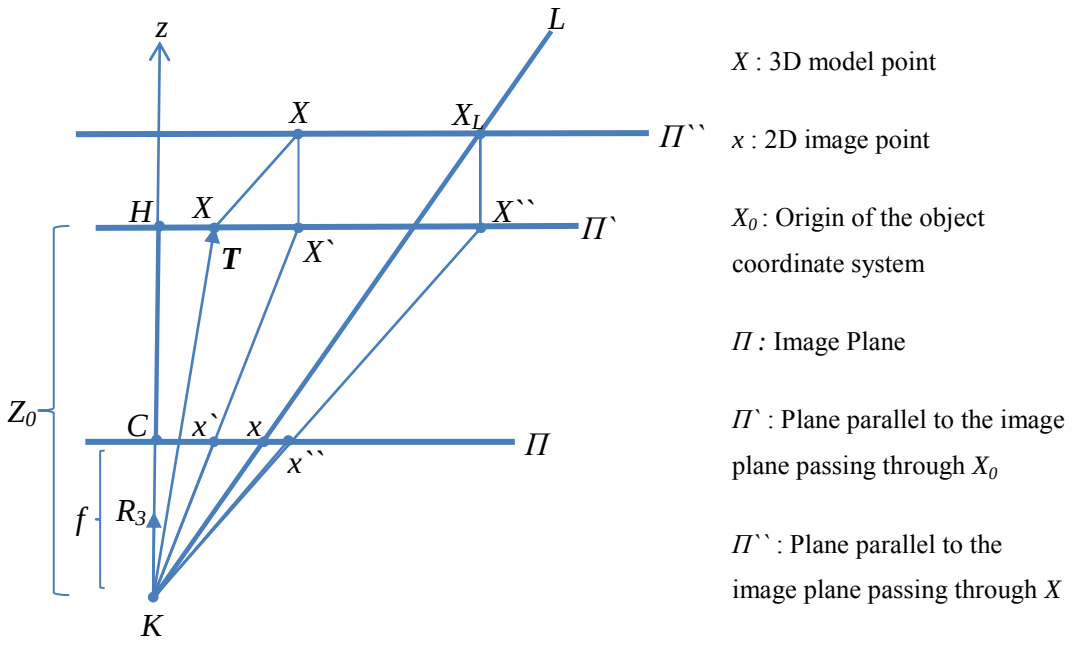

Figure 6 : Scaled Orthographic Projection

For the pose estimation problem, 3D coordinates of object points and 2D coordinates of image points are available. If Figure 6 is examined, *x* denotes an image point coordinate and *X* denotes an object point coordinate. *x`* is the scaled orthographic projection of object point *X* and X_L is an object point on the line of sight of image point *x*. The scaled orthographic projection of X_L is denoted by *x``*. The idea here is that *x`* and *x``* must be superposed, so that *X* will fall on the line of sight of image point *x*. This condition can be satisfied by minimizing the distance between *x`* and *x``*. This distance can be expressed as $|Cx` - Cx``|$.

If we have a set of 3D object points and 2D image points, this condition must be satisfied for all point pairs. By minimizing a global objective function as given in (4.32), one can find a solution.

$$\begin{aligned} E &= \sum_i {d_i}^2 = \sum_i |Cx` - Cx``|^2 \\ &= \sum_i ((Q_1 \cdot X_i - w_i x_i)^2 + (Q_2 \cdot X_i - w_i y_i)^2) \end{aligned} \tag{4.32}$$

where Q_1 and Q_2 are *pose vectors* with length four in homogenous coordinates as given in (4.33). X_i denotes the vector denoting the homogenous coordinates of object points.

$$\begin{aligned} Q_1 &= s(R_1, T_x) \\ Q_2 &= s(R_2, T_y) \end{aligned} \tag{4.33}$$

Minimizing (4.32) means minimizing the sum of squared distances of object points to line of sight. The objective is to find the pose vectors minimizing this cost function.

If the correspondences are unknown, the objective function in (4.32) can be modified as in (4.34) where the number of image points is *N* and number of object points is *M*.

$$\begin{aligned} E &= \sum_{j=1}^{N}\sum_{i=1}^{M} m_{ji}\left(d_{ji}^2 - \alpha\right) \\ &= \sum_{j=1}^{N}\sum_{i=1}^{M} m_{ji}\left(\left(Q_1 \cdot X_i - w_i x_j\right)^2 + \left(Q_2 \cdot X_i - w_i y_j\right)^2 - \alpha\right) \end{aligned} \tag{4.34}$$

In (4.34), m_{ji} are the correspondence variables that define the assignments between the object points and image points. The parameter α is a penalty term that moves the minimum away from the trivial solution $m_{ji}=0$. The global objective function is minimized iteratively with the initial estimation of the pose is given and initial value for w_i is selected as 1. The summary for *Soft-POSIT* algorithm is as follows:

i. Compute the correspondence variables assuming everything else (Q_1, Q_2 and w_i) is fixed (described in Section 3.2.1.1)
ii. Compute pose vectors (Q_1 and Q_2) assuming everything else (m_{ji} and w_i) is fixed (described in Section 3.2.1.2).
iii. Compute the correction term w_i using Q_1 and Q_2 by (4.35).

$$w_i = {R_3 \cdot P_i}/{T_z} + 1 \tag{4.35}$$

3.2.1.1 Calculating Correspondences

If the parameters d_{ji}^2 in expression E of (4.34) are known, optimal values for the correspondence variables m_{ji} can be calculated. An *assignment matrix* (***m***) that has one row for each image point and one column for each object point is formed. An extra row called *slack row* and an extra column, namely *slack column,* are added to this assignment matrix. The sum of each row and each column of assignment matrix must be equal to 1. The aim here is to form an assignment matrix that consists of 1's and 0's in each row and coloumn.

The elements of assignment matrix are initialized to $exp(-\beta(d_{ji}^2 - \alpha))$ and the slack elements are initialized to a very small number. In this equation, β is a very small number and $\alpha = 9.21 \times \sigma^2$, where σ is the standard deviation of the image noise.

The values for d_{ji}^2 are calculated using the initial estimates of pose variables. The value of β is initialized at the beginning of the algorithm and it depends on the quality of the initial pose. If there is no idea of the initial pose, β can be selected as 0.0004 and if the initial pose can be guessed β can be chosen larger.

The assignment matrix can be calculated iteratively by performing these two steps at each iteration step:

i. Matrix normalization with Sinkhorn's method [36].
ii. Deterministic annealing known as s*oftmax [37]*. Increasing β at each iteration step, the term m_{ji} with the smallest d_{ji}^2 tends to converge to 1 and the other elements tend to converge to 0. The iterations are stopped when the value of β exceeds 0.5.

3.2.1.2 Calculating Pose

When the correspondence variables are known, the pose variables Q_1 and Q_2 can be calculated by using (4.36).

$$Q_1 = \left(\sum_{i=1}^{M} m_i' X_i X_i^T\right)^{-1} \left(\sum_{j=1}^{N}\sum_{i=1}^{M} m_{ji} w_i x_j X_i\right)$$
$$Q_2 = \left(\sum_{i=1}^{M} m_i' X_i X_i^T\right)^{-1} \left(\sum_{j=1}^{N}\sum_{i=1}^{M} m_{ji} w_i y_j X_i\right) \quad (4.36)$$

The scale factor, rotation vectors and translation vector can be calculated from Q_1 and Q_2 as given in (4.37).

$$s = (\|Q_1^1, Q_1^2, Q_1^3\| \, \|Q_2^1, Q_2^2, Q_2^3\|)^{1/2}$$
$$R_1 = {(Q_1^1, Q_1^2, Q_1^3)^T}/{s} \;, \quad R_2 = {(Q_2^1, Q_2^2, Q_2^3)^T}/{s} \;, \quad R_3 = R_1 \times R_2 \quad (4.37)$$
$$T_x = Q_1^4/s \;, \qquad T_y = Q_2^4/s \;, \qquad T_z = f/s$$

where Q_1^k is k^{th} is row of Q_1 and Q_2^k is the k^{th} row of Q_2.

3.2.2 Blind-PnP

Blind-PnP algorithm is a pose estimation algorithm that is proposed by Francesc Moreno, *et.al.* in 2008 [22]. This algorithm determines the pose and correspondences simultaneously in two steps. In the first step pose priors are calculated by using some clues about the constraints on the camera position and the 3D model of the object. Pose priors are calculated offline for one time and the parameters are stored. In the second step, by using these pose priors, the final pose and correspondences are calculated.

3.2.2.1 Calculating Pose Priors

The pose priors are obtained by forming a torus that is composed of possible camera locations around an arbitrary object. Then, these pose priors are modeled as the sum of a number of Gaussian distributions and the most probable pose priors are selected. During the experiments, in this thesis study, the pose priors are modeled by 20 Gaussian distributions to obtain 20 pose priors. These pose priors are given as input to the algorithm.

For the formation of the torus, two radius parameters R_{major} and R_{minor} are used. In the experiments, R_{major} is determined as the arithmetic mean of the minimum and maximum camera distances from the center of object points (The COG of the object model points is {0, 0, 0} during the simulations). The minimum distance is assumed as 10mm and maximum distance is thought as 800mm; hence, R_{major} is determined as 405mm, whereas R_{minor} is assumed to be 100mm.

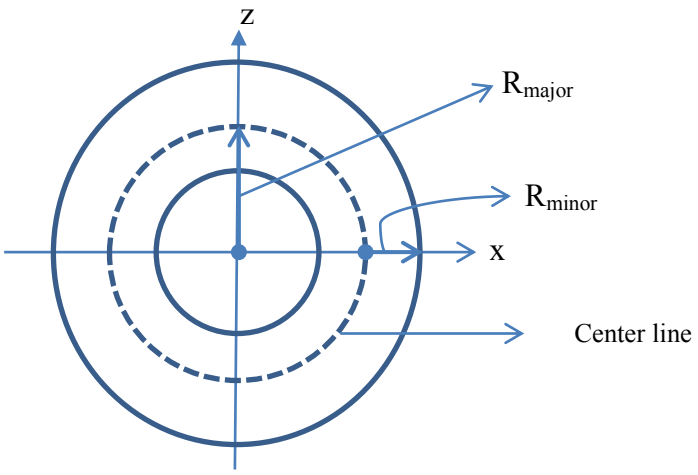

Figure 7 : Torus Parameters

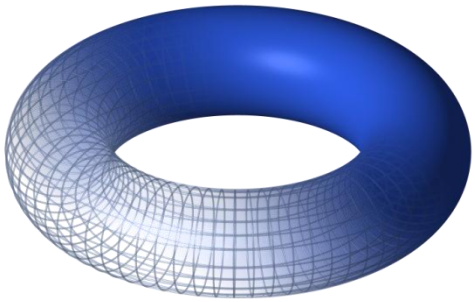

Figure 8 : A sample torus shape

The other parameters for the generation of pose priors are the number of samples for each axis (n_1, n_2, $n_{samples\_rotation}$). A point is selected inside a sphere containing the model points, and another point is selected on the torus. Then the orientation of the vector pointing from the point on the torus to the point on the sphere is calculated and this is taken as pose sample. This process is applied for a number of sample points on the sphere (n_1) and for a number of points on the torus (n_2). (In this case, a small modification is performed during the selection of points on the sphere: it was guaranteed that one of the selected points is exactly at the center of the model points). The model points are projected onto the image plane for each

pose calculated in this manner and if the number of points falling into the image plane is below a threshold, then the corresponding pose is eliminated. During the experiments n_1 is selected as 10 whereas n_2 is selected as 1000; finally $n_{samples\_rotation}$ is set to 60.

After the formation of torus, maximum n_1 x n_2 x $n_{samples\_rotation}$ pose samples are obtained. These pose samples are approximated by a number of Gaussian distributions. After modeling, the same number of pose priors with Gaussian distributions is left and there is 6x6 covariance matrix for each pose prior.

As a summary, the parameters for the formation of pose priors are the radius values for the torus, the number of samples for each axis and the number of Gaussians. Once these parameters are determined, the pose priors can be calculated once and can be saved into a file. There is no need to calculate the pose priors in every experiment unless the model points and possible camera locations do not change.

3.2.2.2 Calculating Pose and Correspondences

The pose and correspondences are calculated using a hypothesis and test approach in *Blind-PnP* algorithm. The aim of the algorithm is to find the correct correspondences between 2D points and 3D points and also the correct pose. For this purpose the cost function given in (4.38) is minimized.

$$E(p) = \sum_{(X,x)\epsilon Matches} \|x - Proj(p;X)\| + |Not\ Detected|\tau \qquad (4.38)$$

In (4.38), X is the set of 3D object points, x is the set of 2D points and $Proj(p; X)$ is the projection of X with pose p. *Not Detected* is the set of 3D points which do not

have corresponding 2D points and τ is the constant penalty term. The aim here is to minimize the re-projection error and increase the number of found correspondences.

For each Gaussian component (pose prior) and for each 3D object point, a corresponding 2D point is searched. Each pose prior corresponds to a small elliptical region on the image plane for a 3D point and the corresponding 2D point must be on this elliptical region. Hence, the search region is reduced from the entire image to a small region. This is one of the most critical points of the algorithm. The possible matches for a 3D point are 2D points satisfying the condition in (4.39), where $\hat{x}_i$ is the projection of 3D object point X_i.

$$\left(\hat{x}_i - x_j\right)^T \sum_i^{\hat{x}} \left(\hat{x}_i - x_j\right) \leq M^2 \tag{4.39}$$

In (4.39), $\sum_i^{\hat{x}}$ is the covariance of 3D the object point X_i and can be calculated from (4.40).

$$\sum\nolimits_i^{\hat{x}} = J(X_i) \sum\nolimits_g^p \left(J(X_i)\right)^T \tag{4.40}$$

In (4.40), $J(X_i)$ is the Jacobian of projection of X_i. $\sum_g^p$ is the covariance matrix of pose prior and $\sum_i^{\hat{x}}$ determines a search region for point X_i.

Among the 3D points that have at least one potential match, we chose the one with smallest number of matches. By taking the 3D point and taking one of its potential matches, the pose prior and its covariance matrix is updated by using Kalman filter by using (4.41) and (4.42) respectively.

$$p_g^+ = p_g + K\left(x_j - Proj\left(p_g; X_i\right)\right) \tag{4.41}$$

$$\sum_g^{p\ +} = \left(I - KJ(X_i)\right) \sum_g^p \tag{4.42}$$

This updated pose prior and its covariance matrix is used for determining the second correspondence. This time, since the pose prior has improved, the search region is smaller and it is easier to find the second match.

This process continues till the number of correspondences reaches to the value of 3. When the number of correspondences is larger than or equal to 3, the remaining correspondences can be obtained by projecting the model points onto the image plane. This situation coincides by the fact that pose can be calculated from at least three correspondences.

The iterations continue until the error in the cost function given in (4.38) decreases below a threshold value.

CHAPTER 4

EXPERIMENTS

The algorithms compared in this thesis study are divided into two groups. First group consists of algorithms requiring 2D-3D point correspondences to be known. The other group of algorithms can solve both 2D-3D point correspondences and pose at the same time. For comparing each group of algorithms, the experiments are designed separately.

For the experiments, a 3D object that can be attached to a person's head is used. There are 24 infrared light emitting diodes (IR LEDs) on the object. Each LEDs position is known preciously with respect to the center of gravity of LED locations. The 3D object that is used in the experiments is shown in Figure 9.

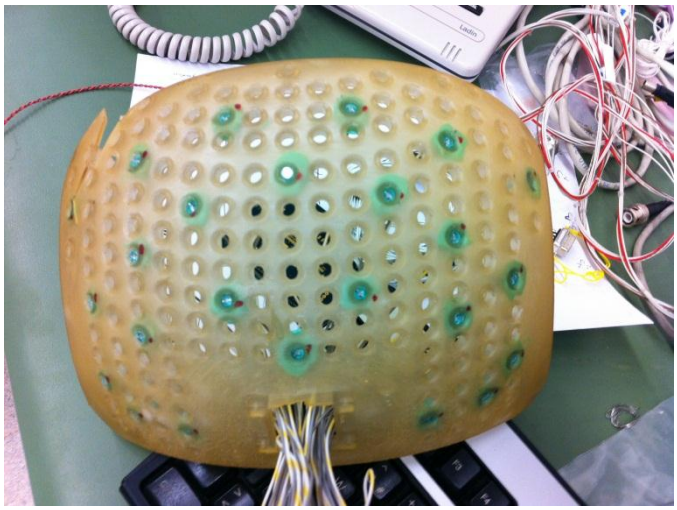

Figure 9 : Object used in the experiments

The IR-LEDs placed on the object are imaged by a near-IR camera. Mikrotron's EoSens CL camera is used for this purpose during the experiments. The IR-LEDs used during the experiments have wave-length beyond visible light and a band-pass filter that does not pass visible light is placed in front of the camera. By this combination, only the LEDs can be seen by the camera and this reduces the segmentation effort in feature extraction process. The images are taken in 50 fps and the size of the images is 640 x 512 pixels. In Figure 10, the camera and optical filter used in the experiments are illustrated.

Figure 10 : Mikrotron's EoSens CL Camera and Filter Used in Experiments

For visual test data, the object shown in Figure 9 is attached to the back of a helmet. While wearing the helmet and making real head movements, the images are captured behind the helmet. The test bench is designed as shown in Figure 11. An example of taken images is shown in Figure 12.

The experiments are performed using *MATLAB* environment. Personal computers with Intel Core2-Quad Q9400 2.66 GHz CPU, 4GB RAM and 500 GB memory are utilized during the experiments.

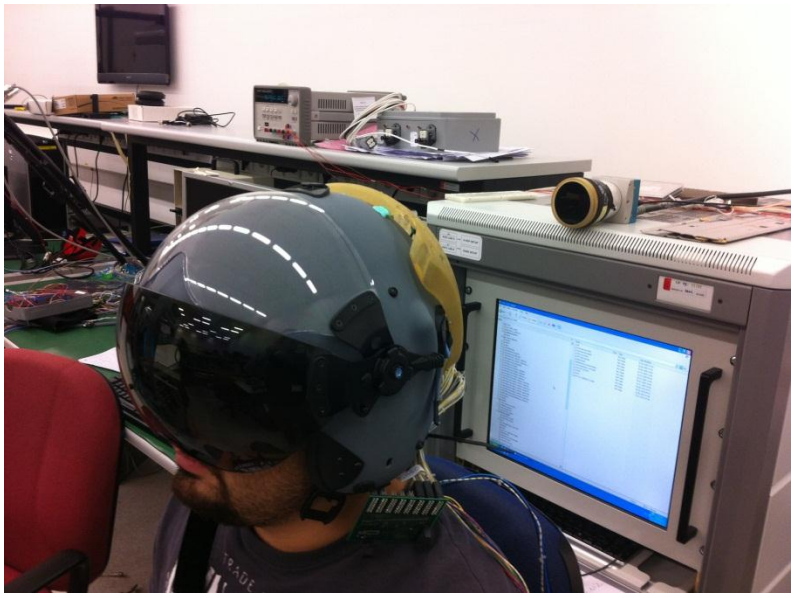

Figure 11 : Test Bench Designed for Taking Visual Data

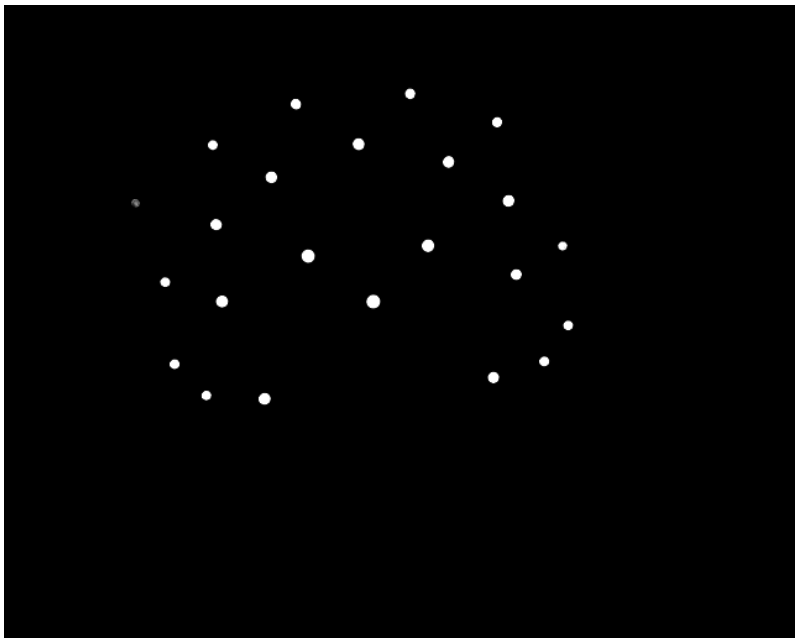

Figure 12 : An Example of Visual Data

4.1 Comparison of Algorithms with Correspondence

In this part of experiments, four well-known algorithms that calculate rotation and translation from 2D-3D point correspondences are compared. These algorithms are *POSIT*, *Orthogonal Iterations (ORHIT)*, *Efficient PnP (EPnP)* and *Direct Linear Transformation (DLT)*. The experiments are carried out in two steps: first using synthetic data, and then, using real visual data.

4.1.1 Experiments Using Synthetic Data

Inputs of pose estimation algorithms are 3D model points and corresponding 2D image points. The performance of algorithms is affected by the amount of noise in both model points and image points. The mismatch between model points and image points affects the performance of the algorithms, too. Synthetic data is generated for each of these cases to see how algorithms are affected. A set of previously collected realistic head movement data is used for synthetic data generation. Flow-chart for experiments using synthetic data is given in Figure 13.

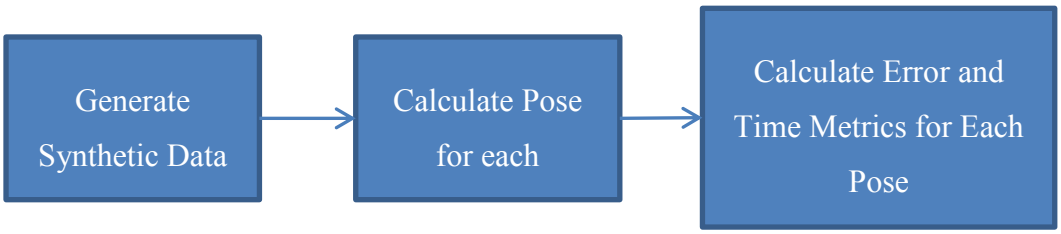

Figure 13 : Synthetic Data Experiment Flow Chart

Synthetic data is generated by projecting model points onto the image plane by a realistic head pose data. When investigating the effect of model point noise, first noise is added to the model points then the model points are projected to form synthetic image points. If effect of image noise is being investigated, first original model points are projected, then noise is added to the 2D points obtained. When

investigating outlier effect, original model points are projected to form image points and then coordinates of desired number of image points are changed. The visible model points at a given pose are calculated by using the method in [38]. Flow-chart for synthetic data generation is shown in Figure 14.

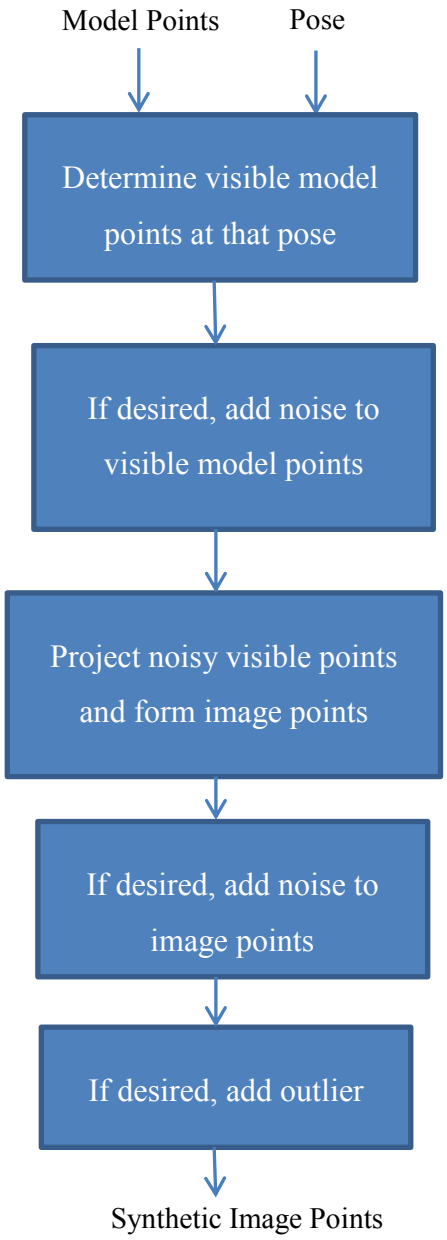

Figure 14 : Flow-Chart of Synthetic Data Generation

4.1.1.1 Effect of Image Noise

The effect of image noise to the performance of pose estimation algorithms is investigated by performing several experiments. For each experiment, zero-mean Gaussian distributed noise with different standard deviations (0, 1, 2, 3, 4 pixels) is added to the image points given to the algorithms. An example of synthetic image points with zero mean and 3 pixels standard deviation noise is shown in Figure 15. Re-projection error, rotation error in x-, *y*-, *z*- axes, translation error in *x*-, *y*-, *z*- axes and execution time values are recorded for each case. The results are summarized in Table 1, Table 2, Table 3, Table 4, Table 5, Table 6, Table 7 and Table 8.

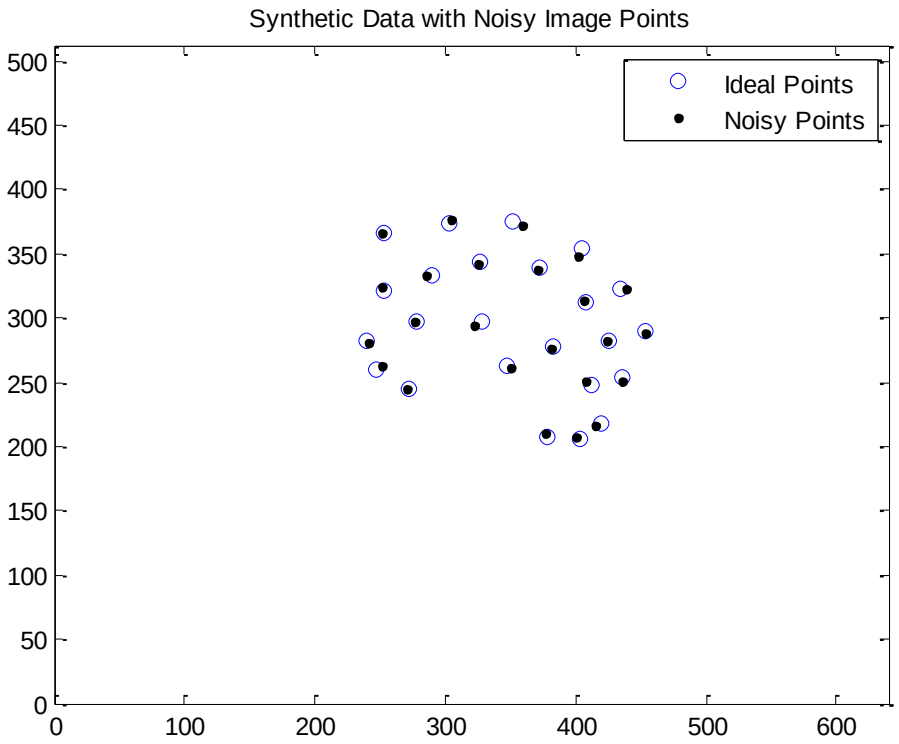

Figure 15 : Image Points with Mean = 0, Variance = 9 Noise

Table 1 : Re-projection Error (pixel) vs. Image Noise

	0 pixel	1 pixel	2 pixels	3 pixels	4 pixels
POSIT	0.03	1.13	2.26	3.38	4.51
ORTHIT	2.0e-5	1.16	2.32	3.48	4.65
EPnP	2.4e-9	1.61	3.25	5.19	7.94
DLT	6.9e-13	2.59	5.37	8.28	11.61

Table 2 : Rotation Error around *X*-Axis (mrad) vs. Image Noise

	0 pixel	1 pixel	2 pixels	3 pixels	4 pixels
POSIT	0.42	6.52	13.08	19.15	25.92
ORTHIT	2.7e-4	4.24	8.49	12.96	17.45
EPnP	2.9e-8	13.85	26.80	81.08	81.08
DLT	4.6e-12	16.85	42.04	159.71	159.71

Table 3 : Rotation Error around *Y*-Axis (mrad) vs. Image Noise

	0 pixel	1 pixel	2 pixels	3 pixels	4 pixels
POSIT	0.70	10.28	20.64	30.41	41.65
ORTHIT	7.2e-4	6.61	13.26	20.03	27.42
EPnP	6.9e-8	25.49	49.64	74.32	98.71
DLT	7.3e-12	23.49	49.04	82.08	111.78

Table 4 : Rotation Error around *Z*-Axis (mrad) vs. Image Noise

	0 pixel	1 pixel	2 pixels	3 pixels	4 pixels
POSIT	0.52	13.00	25.90	39.01	52.11
ORTHIT	3.2e-4	8.56	17.14	26.13	34.73
EPnP	4.7e-8	31.70	62.39	95.20	127.70
DLT	7.5e-12	27.10	57.71	92.32	122.13

Table 5 : Translation Error along *X*-Axis (mm) vs. Image Noise

	0 pixel	1 pixel	2 pixels	3 pixels	4 pixels
POSIT	0.02	0.26	0.52	0.78	1.04
ORTHIT	9.0e-6	0.22	0.44	0.68	0.92
EPnP	1.2e-9	0.60	1.19	1.89	2.71
DLT	3.6e-13	1.11	2.51	5.08	8.08

Table 6 : Translation Error along *Y*-Axis (mm) vs. Image Noise

	0 pixel	1 pixel	2 pixels	3 pixels	4 pixels
POSIT	0.01	0.27	0.55	0.82	1.11
ORTHIT	1.0e-5	0.24	0.48	0.74	1.03
EPnP	1.3e-9	0.66	1.35	2.13	3.01
DLT	2.8e-13	1.00	2.25	4.88	8.00

Table 7 : Translation Error along *Z*-Axis (mm) vs. Image Noise

	0 pixel	1 pixel	2 pixels	3 pixels	4 pixels
POSIT	0.04	2.64	3.30	4.90	6.57
ORTHIT	6.4e-5	1.30	2.68	4.19	5.95
EPnP	9.8e-9	4.42	9.16	14.74	21.23
DLT	2.9e-12	9.28	20.73	42.64	68.23

Table 8 : Execution Time (msec) vs. Image Noise

	0 pixel	1 pixel	2 pixels	3 pixels	4 pixels
POSIT	0.76	0.68	0.68	0.68	0.68
ORTHIT	237.36	122.94	111.14	105.00	101.39
EPnP	1.60	1.20	1.23	1.51	1.24
DLT	0.30	0.30	0.30	0.30	0.30

The variation of re-projection error, rotation error (norm of rotation errors in *x*-, *y*-,*z*- axes), translation error (norm of translation errors in *x*-, *y*-, *z*- axes) and execution time with different amounts of image noise can be seen from graphs shown in Figure 16, Figure 17, Figure 18 and Figure 19.

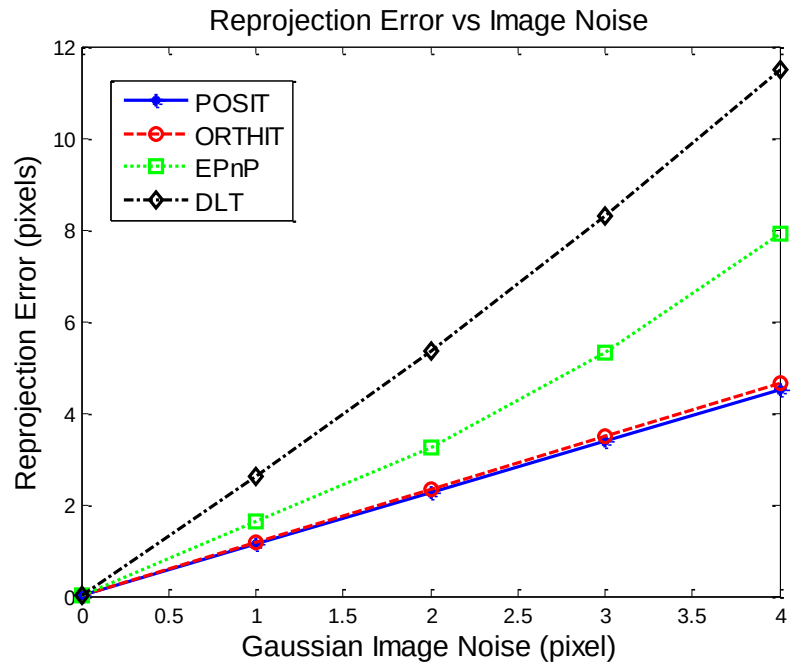

Figure 16 : Re-projection Error vs. Image Noise

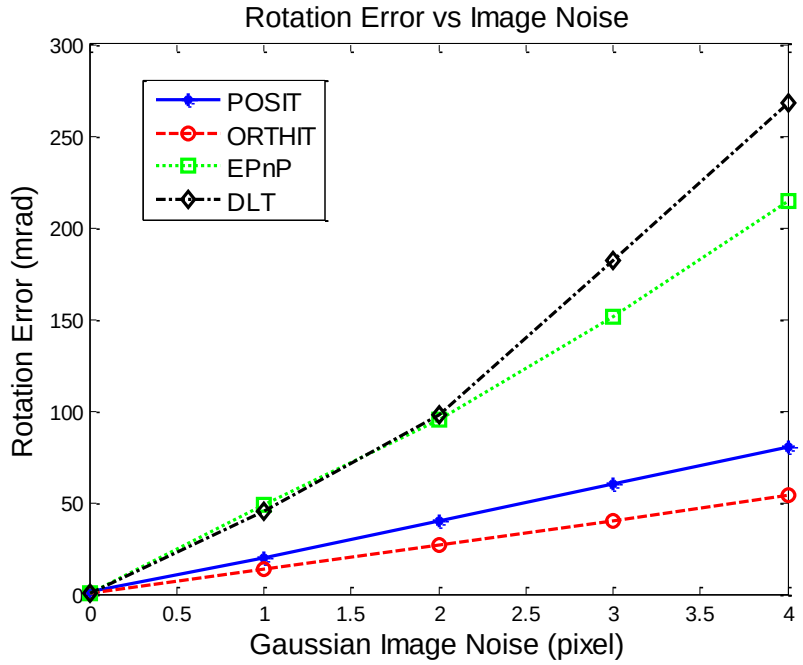

Figure 17 : Rotation Noise vs. Image Noise

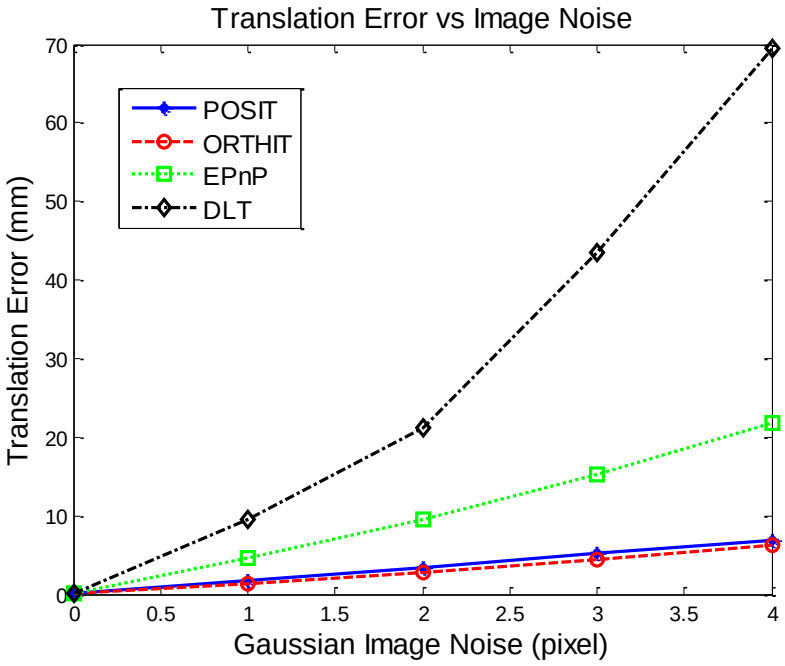

Figure 18 : Translation Error vs. Image Noise

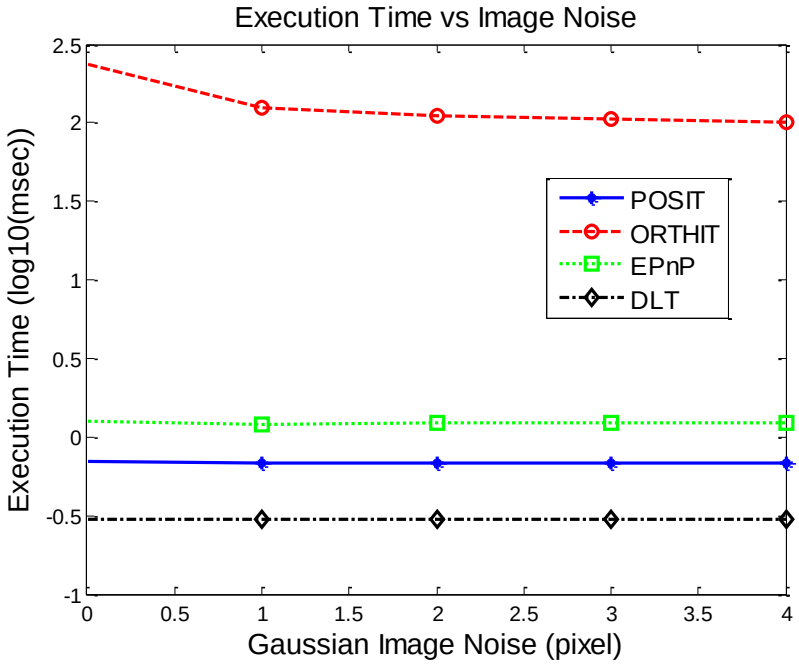

Figure 19 : Execution Time vs. Image Noise

4.1.1.2 Effect of Model Point Noise

The effect of the noise in model points to the performance of pose estimation algorithms is investigated by performing several experiments. For each experiment, zero-mean Gaussian distributed noise with different standard deviations (0, 0.5, 1, 1.5, 2 mm) is added to the model points and synthetic image points are generated using these noisy points. An example of synthetic image points generated from model points with zero mean and 1.5 mm standard deviation noise is shown in Figure 20. Re-projection error, rotation error, translation error and execution time values are recorded for each case. The results are summarized in Table 9, Table 10, Table 11, Table 12, Table 13, Table 14, Table 15 and Table 16.

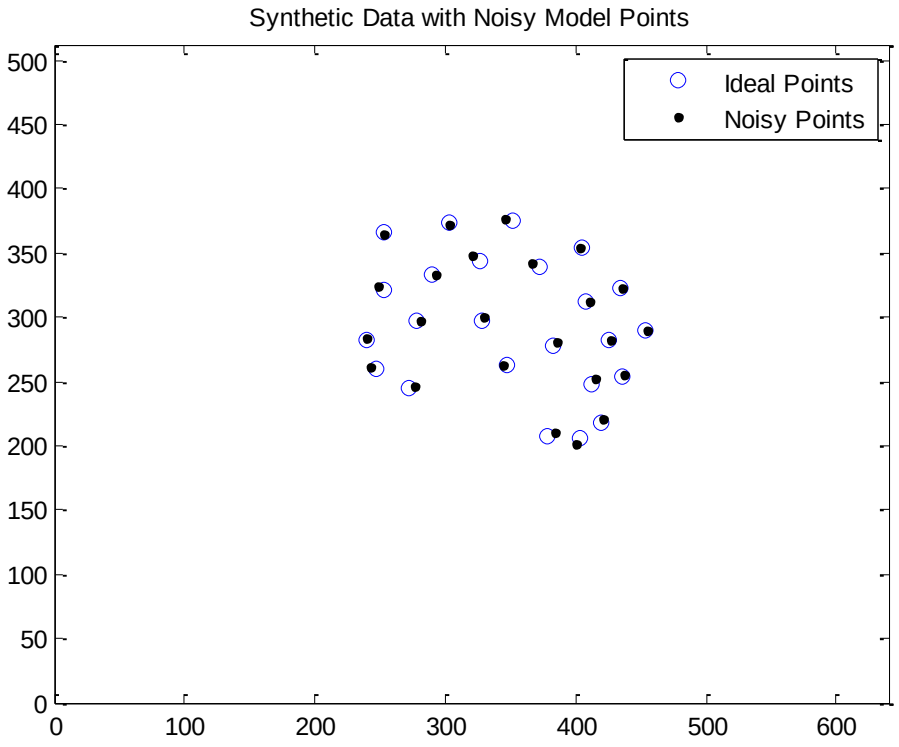

Figure 20 : Image Points with Mean = 0, Variance = 2.25 Model Point Noise

Table 9 : Re-projection Error (pixel) vs. Model Point Noise

	0 mm	1 mm	2 mm	3 mm	4 mm
POSIT	0.03	0.32	1.29	2.91	5.17
ORTHIT	2.0e-5	0.33	1.32	2.99	5.33
EPnP	2.4e-9	0.45	1.82	4.24	9.20
DLT	6.9e-13	0.74	2.98	6.93	13.52

Table 10 : Rotation Error around *X*-Axis (mrad) vs. Model Point Noise

	0 mm	1 mm	2 mm	3 mm	4 mm
POSIT	0.42	1.87	7.26	16.29	29.53
ORTHIT	2.7e-4	1.20	4.85	10.81	20.12
EPnP	2.9e-8	3.97	15.47	35.39	93.96
DLT	4.6e-12	4.79	19.43	61.83	190.19

Table 11 : Rotation Error around *Y*-Axis (mrad) vs. Model Point Noise

	0 mm	1 mm	2 mm	3 mm	4 mm
POSIT	0.70	2.94	11.51	25.95	46.31
ORTHIT	7.2e-4	1.86	7.42	16.77	30.33
EPnP	6.9e-8	7.29	28.42	62.82	109.80
DLT	7.3e-12	6.76	26.60	62.99	125.69

Table 12 : Rotation Error around *Z*-Axis (mrad) vs. Model Point Noise

	0 mm	1 mm	2 mm	3 mm	4 mm
POSIT	0.52	3.71	14.56	32.81	58.87
ORTHIT	3.2e-4	2.42	9.70	22.00	39.71
EPnP	4.7e-8	8.94	35.65	77.97	142.83
DLT	7.5e-12	7.55	30.93	73.92	139.28

Table 13 : Translation Error along *X*-Axis(mm) vs. Model Point Noise

	0 mm	1 mm	2 mm	3 mm	4 mm
POSIT	0.02	0.07	0.29	0.66	1.17
ORTHIT	9.0e-6	0.06	0.24	0.57	1.06
EPnP	1.2e-9	0.17	0.67	1.55	3.15
DLT	3.6e-13	0.32	1.26	3.64	9.76

Table 14 : Translation Error along *Y*-Axis (mm) vs. Model Point Noise

	0 mm	1 mm	2 mm	3 mm	4 mm
POSIT	0.01	0.07	0.30	0.69	1.21
ORTHIT	1.0e-5	0.06	0.26	0.60	1.13
EPnP	1.3e-9	0.18	0.73	1.71	3.35
DLT	2.8e-13	0.27	1.11	3.15	8.86

Table 15 : Translation Error along *Z*-Axis (mm) vs. Model Point Noise

	0 mm	1 mm	2 mm	3 mm	4 mm
POSIT	0.04	0.46	1.82	4.18	7.34
ORTHIT	6.4e-5	0.37	1.48	3.47	6.89
EPnP	9.8e-9	1.25	5.01	12.07	24.72
DLT	2.9e-12	2.69	10.66	29.98	81.77

Table 16 : Execution Time (msec.) vs. Model Point Noise

	0 mm	0.5 mm	1 mm	1.5 mm	2 mm
POSIT	0.76	0.70	0.70	0.7132	0.71
ORTHIT	237.36	151.76	126.10	112.12	205.32
EPnP	1.6054	1.26	1.25	1.57	1.29
DLT	0.30	0.30	0.30	0.30	0.30

The variation of re-projection error, rotation error (norm of rotation errors in *x*-, *y*-,*z*- axes), translation error (norm of translation errors in *x*-, *y*-, *z*- axes) and execution time with different amounts of model point noise can be seen from graphs shown in Figure 21, Figure 22, Figure 23 and Figure 24.

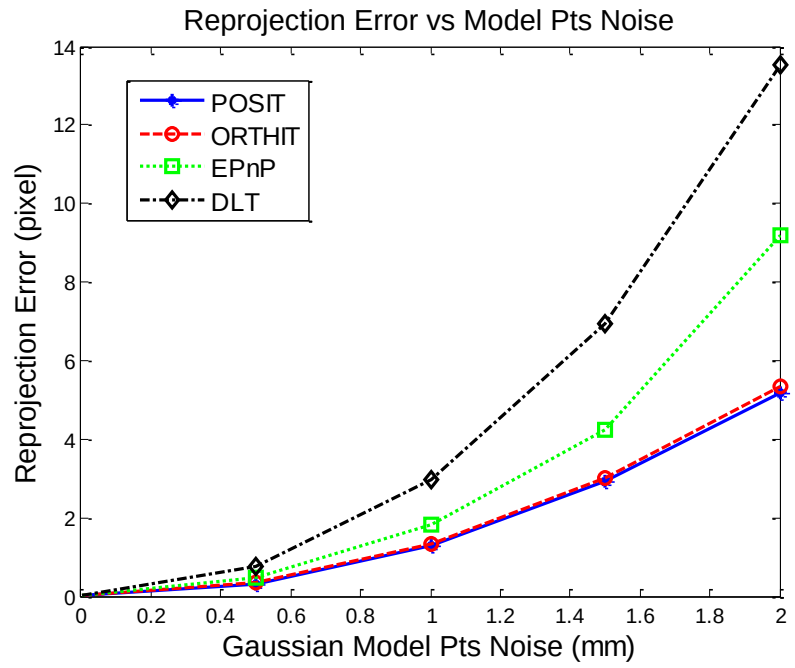

Figure 21 : Re-projection Error vs. Model Point Noise

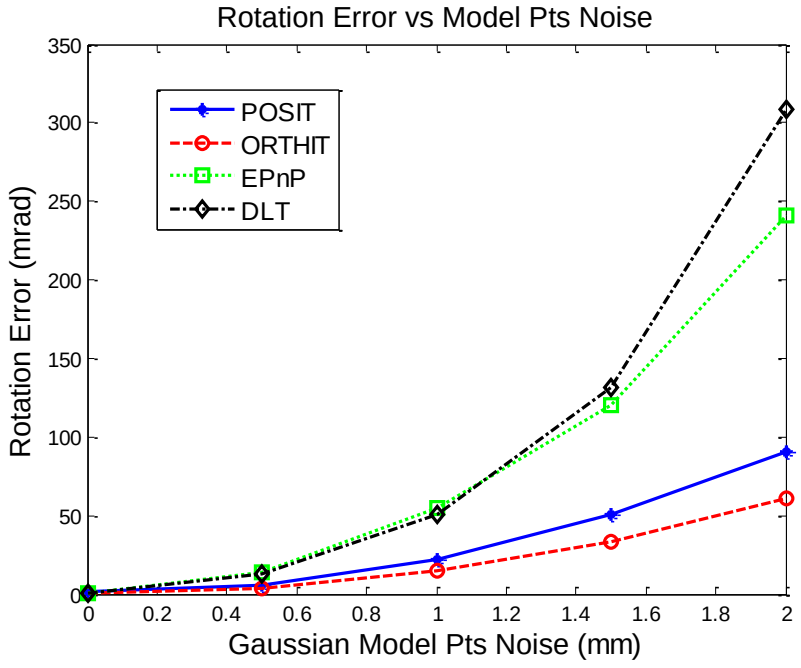

Figure 22 : Rotation Error vs. Model Point Noise

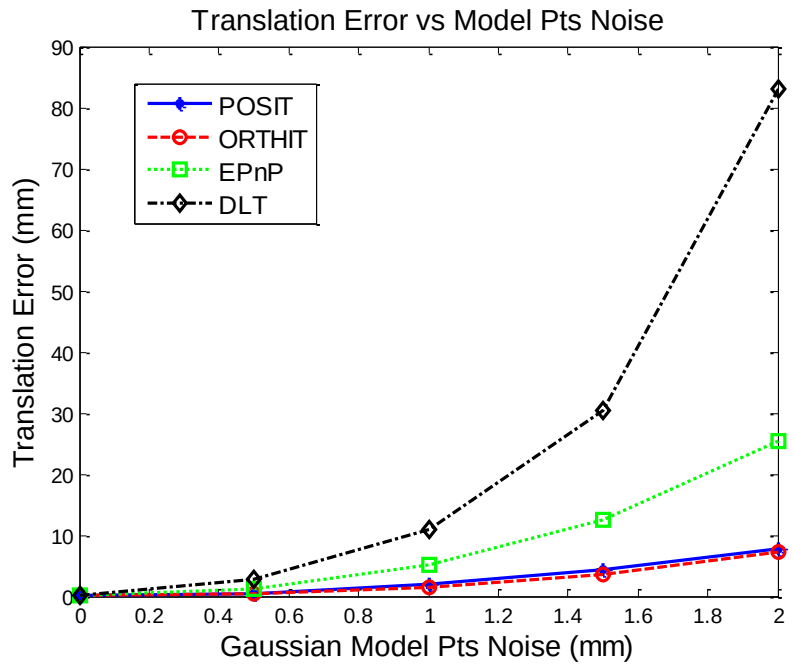

Figure 23 : Translation Error vs. Model Point Noise

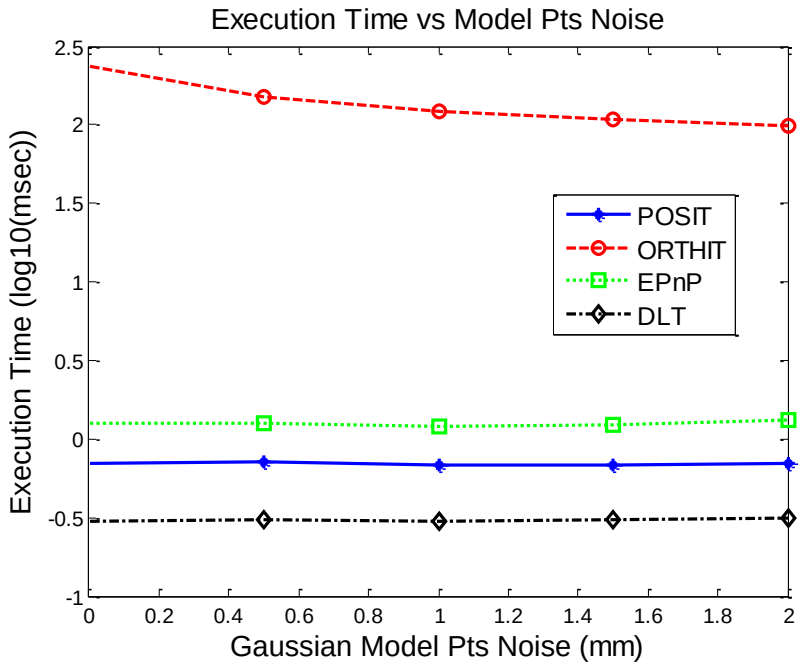

Figure 24 : Execution Time vs. Model Point Noise

4.1.1.3 Effect of Outlier

The effect of the mismatch between model points and image points to the performance of pose estimation algorithms is investigated by performing several experiments. For each experiment, different amount of outlier case is generated. Outlier ratio defines the percentage of model points and image points that match incorrectly. Outlier case is generated by changing the coordinates of an amount of image points obtained by projecting model points. An example of synthetic image points generated from model points with % 16 outlier ratios is shown in Figure 25. Re-projection error, rotation error, translation error and execution time values are recorded for each case. The results are summarized in Table 17, Table 18, Table 19, Table 20, Table 21, Table 22, Table 23 and Table 24

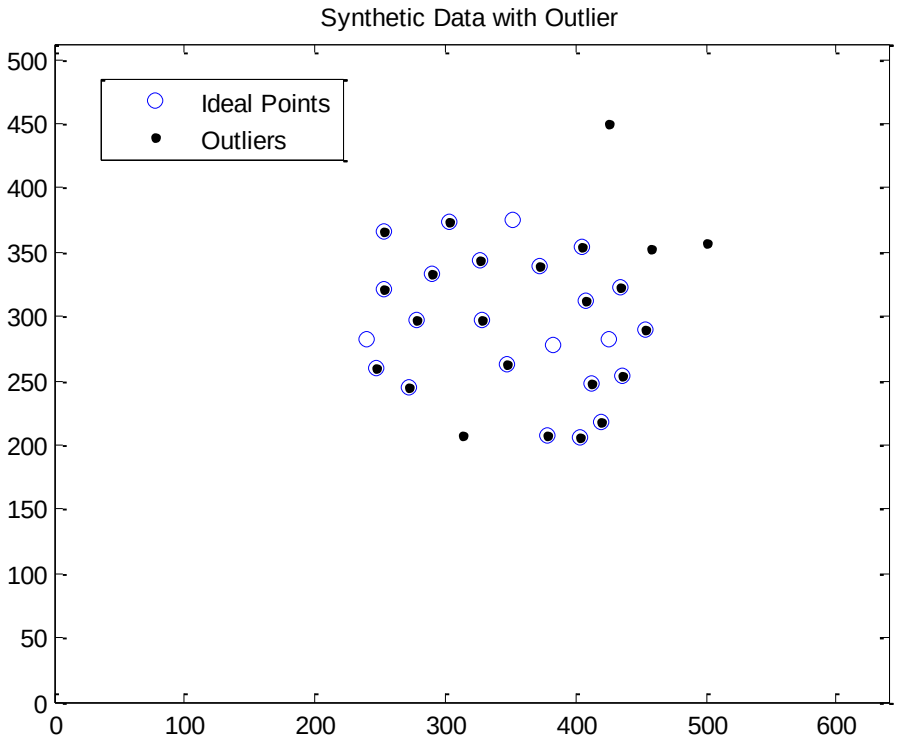

Figure 25 : Synthetic Image Points with %16 Outlier

Table 17 Re-projection Error (pixel) vs. Outlier

	%0	%4	%8	%12	%16
POSIT	0.03	11.17	17.26	22.46	27.57
ORTHIT	2.0e-5	14.37	24.95	33.01	41.43
EPnP	2.4e-9	66.30	78.23	106.27	119.53
DLT	6.9e-13	233.66	903.26	1631.30	2834.50

Table 18 : Rotation Error around *X*-Axis (mrad) vs. Outlier

	%0	%4	%8	%12	%16
POSIT	0.42	117.82	144.41	165.76	188.15
ORTHIT	2.7e-4	105.36	154.13	205.95	253.96
EPnP	2.9e-8	803.30	808.02	869.26	887.22
DLT	4.6e-12	1.40e3	1.48e3	1.49e3	1.46e3

Table 19 : Rotation Error around *Y*-Axis (mrad) vs. Outlier

	%0	%4	%8	%12	%16
POSIT	POSIT	169.85	215.19	255.05	283.22
ORTHIT	ORTHIT	185.31	304.49	395.66	468.77
EPnP	EPnP	339.02	364.28	403.92	419.15
DLT	DLT	702.49	782.69	793.79	763.38

Table 20 : Rotation Error around *Z*-Axis (mrad) vs. Outlier

	%0	%4	%8	%12	%16
POSIT	0.52	218.76	282.00	342.78	393.98
ORTHIT	3.2e-4	205.31	315.27	429.71	542.83
EPnP	4.7e-8	505.30	544.69	582.61	609.37
DLT	7.5e-12	965.77	1.56e3	1.79e3	1.81e3

Table 21 : Translation Error along *X*-Axis (mm) vs. Outlier

	%0	%4	%8	%12	%16
POSIT	0.02	3.97	5.76	7.44	9.24
ORTHIT	9.0e-6	4.01	6.40	8.63	10.59
EPnP	1.2e-9	15.01	16.88	19.65	20.93
DLT	3.6e-13	57.19	56.43	53.66	51.18

Table 22 : Translation Error along *Y*-Axis (mm) vs. Outlier

	%0	%4	%8	%12	%16
POSIT	0.01	3.83	5.01	5.93	6.83
ORTHIT	1.0e-5	6.92	11.24	15.04	18.27
EPnP	1.3e-9	15.81	18.68	21.14	22.62
DLT	2.8e-13	59.01	59.68	58.38	56.78

Table 23 : Translation Error along *Z*-Axis (mm) vs. Outlier

	%0	%4	%8	%12	%16
POSIT	0.04	26.81	35.06	41.74	48.31
ORTHIT	6.4e-5	56.42	97.01	132.27	162.18
EPnP	9.8e-9	115.77	140.04	161.50	173.97
DLT	2.9e-12	540.19	558.69	541.87	521.98

Table 24 : Execution Time (msec.) vs. Outlier

	%0	%4	%8	%12	%16
POSIT	0.76	0.61	0.65	0.61	0.62
ORTHIT	237.36	91.03	94.94	85.93	75.52
EPnP	1.60	2.61	2.66	2.50	2.54
DLT	0.30	0.26	0.27	0.26	0.26

The variation of re-projection error, rotation error (norm of rotation errors in *x*-, *y*-,*z*- axes), translation error (norm of translation errors in *x*-, *y*-, *z*- axes) and execution time with different amounts of outlier can be seen from graphs shown in Figure 26, Figure 27, Figure 28 and Figure 29.

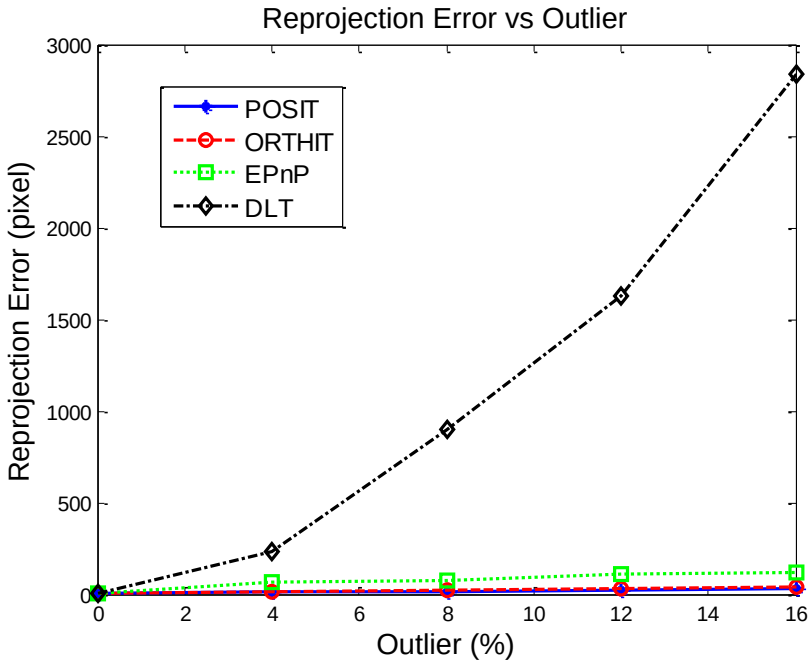

Figure 26 : Re-projection Error vs. Outlier

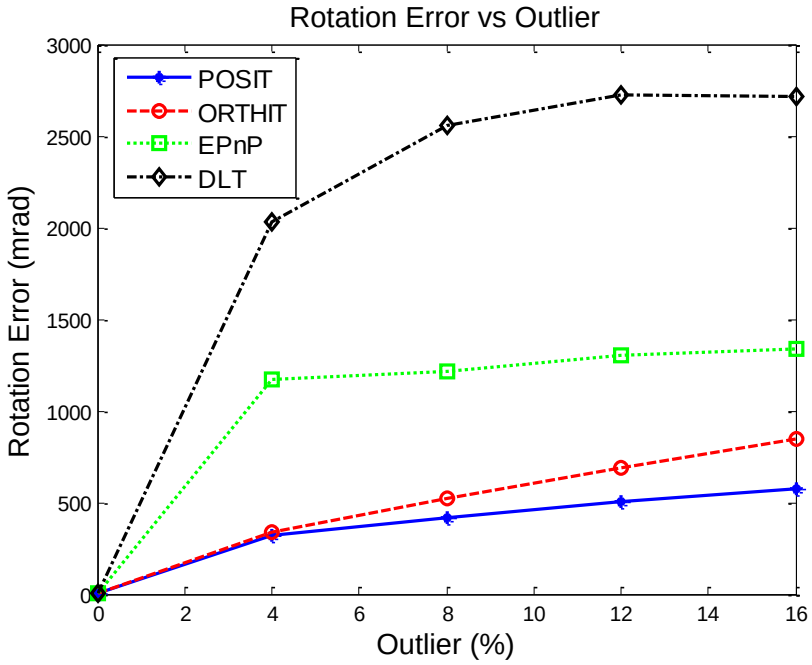

Figure 27 : Rotation Error vs. Outlier

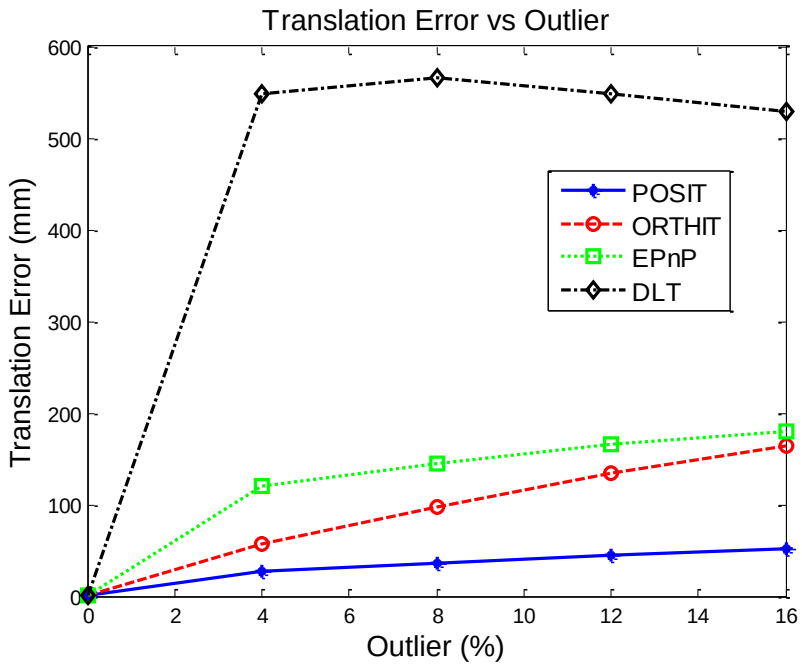

Figure 28 : Translation Error vs. Outlier

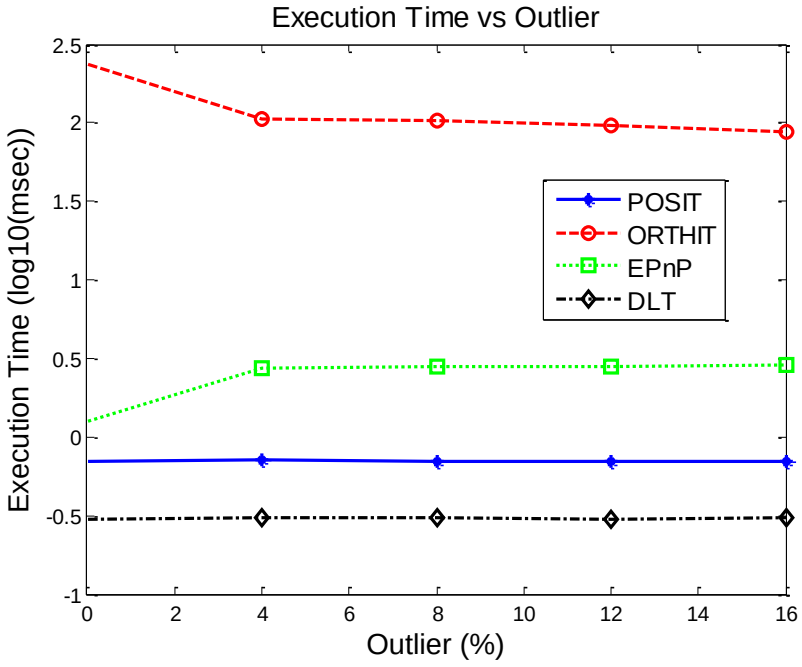

Figure 29 : Execution Time vs. Outlier

4.1.2 Experiments Using Visual Data

The algorithms are also tested with real images taken from the system shown in Figure 11. In this case, the noise in model points and image points is unknown. Testing with real visual data helps us to understand the behavior of algorithms with realistic inputs. The 2D image points are obtained by calculating the centroids of the LED points seen on the image. The centroids are corrected by removing the effect of lens distortion.

When the image points are obtained, the 2D-3D point correspondences are not known. Hence, the corresponding model point for each image point must be calculated first. Soft-Posit algorithm is used for this purpose. At first, model points and obtained image points are given to Soft-Posit to calculate the correspondences, and then the obtained correspondences are preserved by tracking the visible LEDs in each frame. As the LEDs disappear from the view-angle of the camera, they cannot be tracked anymore. This causes a decrease in the number of known correspondences according to the head movements. When the number of correspondences falls below 6, Soft-Posit algorithm is again called to calculate new correspondences.

The correct pose of the object is not known, while capturing the images. It requires very accurate mechanical test benches to obtain ground-truth data for this kind of tests. Hence, the only error metric that can be used in comparison of the algorithms when working with real data is re-projection error. Execution time is another performance metric for comparison.

Flow-chart of the visual data experiments for comparison of the pose algorithms requiring correspondences is given in Figure 30.

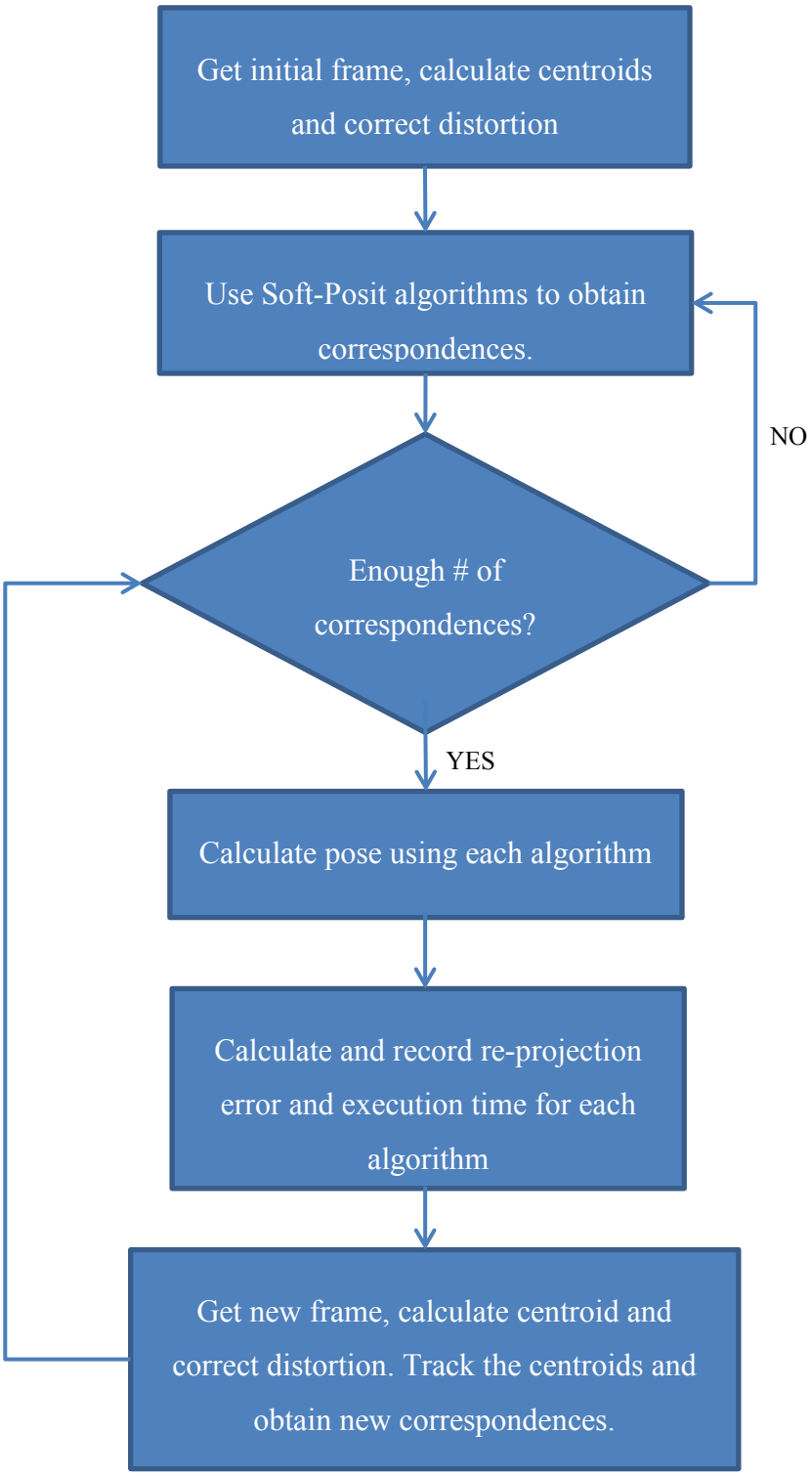

Figure 30 : Flow-Chart for Visual Experiments

The algorithms were tested on 2500 sequential frames. For each frame, rotation and translation is calculated. For the comparison of the algorithms, re-projection error and execution time is calculated for each frame. In Table 25, the mean values of re-projection errors and execution times for each algorithm are given.

Table 25 : Results for Experiments using Visual Data

	Re-projection Error (pixel)	Execution Time (msec)
POSIT	0.31	0.81
ORTHIT	0.33	179.65
EPnP	0.91	1.39
DLT	6.08	0.30

For visualizing, the results are shown with bar graphs, giving information about the mean values and standard deviation for re-projection errors and execution times. The graph for re-projection error is given in Figure 31 and the graph for execution time is given in Figure 32.

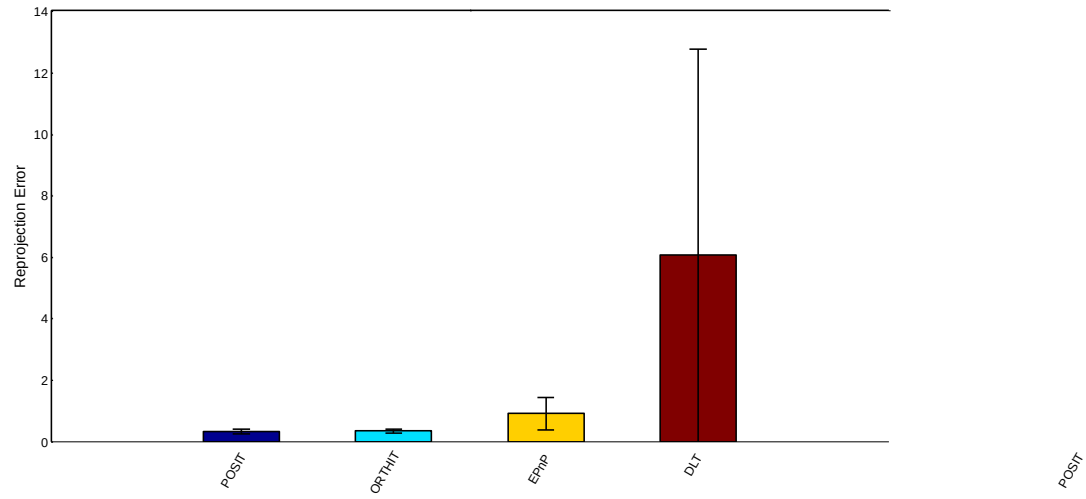

Figure 31 : Re-projection Error for Visual Data Experiment

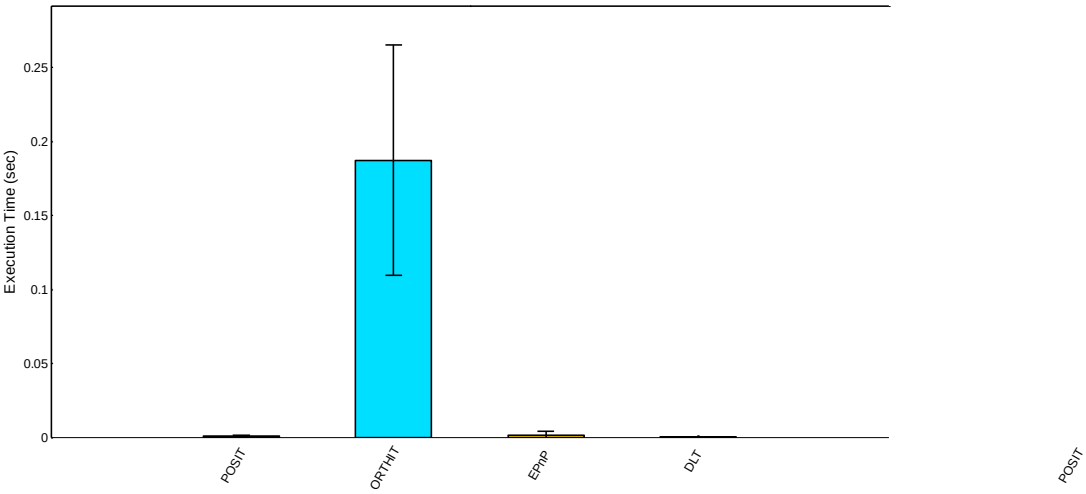

Figure 32 : Execution Time for Visual Data Experiment

4.2 Comparison of Algorithms without Correspondence

In this part of experiments, two well-known algorithms that solve both pose and correspondence simultaneously are compared. These algorithms are Soft-POSIT and Blind-PnP. The experiments are carried out in two steps: first using synthetic data and then using real visual data.

4.2.1 Experiments Using Synthetic Data

Soft-Posit and Blind-PnP algorithms get image points and model points as input and calculate rotation and translation between object and camera. The image points and model points can be in any order and they do not need to be in pairs. The performance of algorithms are affected by the noise in image and model points, the number of image points without corresponding model points (clutter) and the number of model points without corresponding image points (occlusion). Each of these cases is generated by using synthetic data. Synthetic data is formed by projecting model points by previously collected head pose data. The experiments using synthetic data are performed just as the experiments performed for the algorithms with correspondences. Flow-chart of experiments is given in Figure 13. However, there are some differences in synthetic data generation. The model point noise and image point noise cases are generated by the same way. Clutter case is generated by projecting model points onto the image plane and adding additional points to obtained image points. Occlusion case is generated by projecting model points onto the image plane and removing some of the obtained image points. The flow chart for synthetic data generation is given in Figure 33.

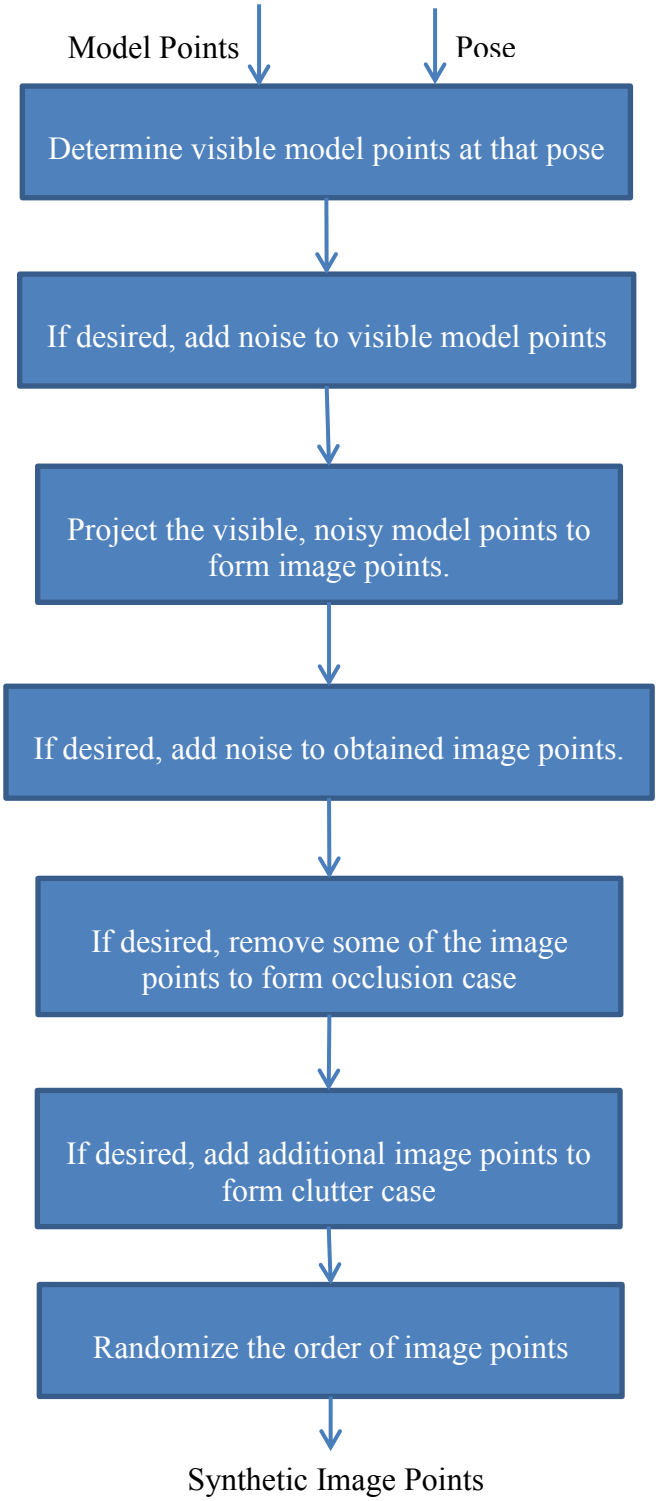

Figure 33 : Flow-Chart for Synthetic Data Generation

4.2.1.1 Effect of Image Noise

The effect of image noise to the performances of Soft-Posit and Blind-PnP algorithms is investigated by performing several experiments. In each experiment, different amount of white Gaussian noise is added to the projections of model points. The created cases are the same as the experiments performed for the algorithms with correspondences. For each experiment, re-projection error, rotation error in x-, *y*-, *z*- axes, translation error in x-, *y*-, *z*- axes and execution time are recorded. When the correspondences are calculated correctly, the algorithm is marked as converged. The convergence ratio indicates the number of experiments that the algorithms converged. An example of the 2D noisy input is given in Figure 15. The results for the performed experiments are given in Table 26, Table 27, Table 28, Table 29, Table 30, Table 31, Table 32, Table 33 and Table 34 .

Table 26 : Re-projection Error (pixel) vs. Image Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.02	1.14	2.39	3.75	5.04
Blind-PnP	0.78	1.29	3.28	4.08	5.59

Table 27 : Rotation Error around *X*-Axis (mrad) vs. Image Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.24	21.43	44.14	74.06	90.11
Blind-PnP	6.31	23.47	46.74	63.03	120.80

Table 28 : Rotation Error around *Y*-Axis (mrad) vs. Image Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.31	18.11	21.00	46.67	70.52
Blind-PnP	7.29	16.36	71.55	62.36	75.06

Table 29 : Rotation Error around *Z*-Axis (mrad) vs. Image Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.17	10.93	14.24	39.00	57.66
Blind-PnP	0.58	8.74	35.17	31.58	39.87

Table 30 : Translation Error along X-Axis (mm) vs. Image Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.24	0.30	0.62	1.24	1.32
Blind-PnP	0.25	0.81	1.14	2.34	4.86

Table 31 : Translation Error along Y-Axis (mm) vs. Image Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	7.7e-2	0.48	0.94	1.47	1.41
Blind-PnP	0.25	0.45	1.21	1.69	2.77

Table 32 : Translation Error along Z-Axis (mm) vs. Image Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.01	1.88	3.66	4.80	7.15
Blind-PnP	0.25	1.44	6.90	4.13	9.26

Table 33 : Execution Time (sec) vs. Image Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.91	0.88	0.82	0.80	0.73
Blind-PnP	41.78	39.31	35.65	40.48	42.16

Table 34 : Convergence Rate (%) vs. Image Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	%76	%74	%72	%70	%72
Blind-PnP	%60	%68	%68	%68	%64

The variation of re-projection error, rotation error (norm of rotation errors in *x*-, *y*-,*z*- axes), translation error (norm of translation errors in *x*-, *y*-,*z*- axes), execution time and convergence rate with different amounts of image noise can be seen from graphs shown in Figure 34, Figure 35, Figure 36, Figure 37 and Figure 38.

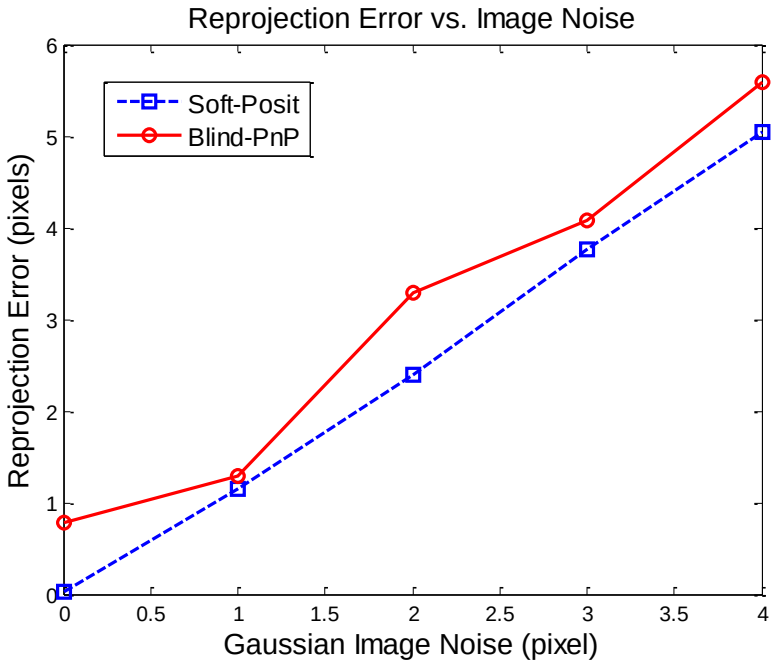

Figure 34 : Re-projection Error vs. Image Noise

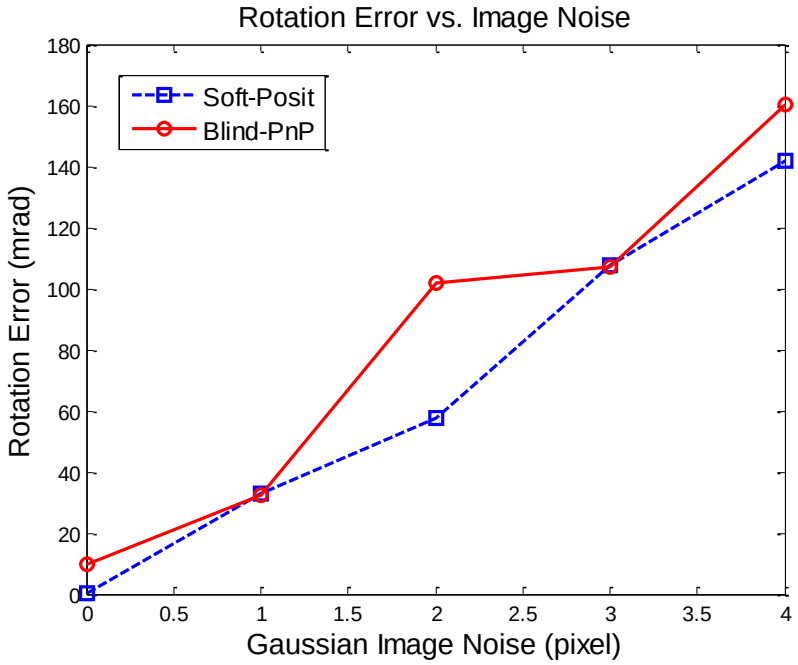

Figure 35 : Rotation Error vs. Image Noise

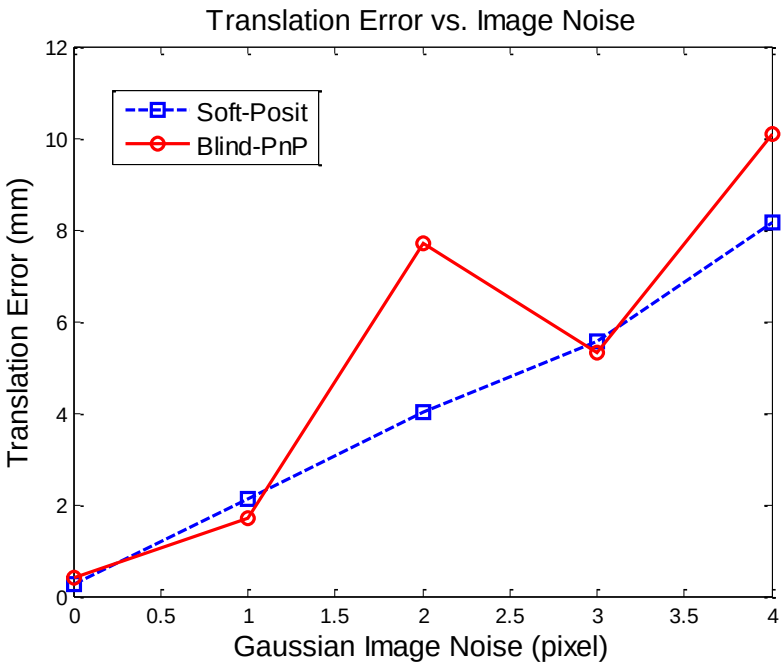

Figure 36 : Translation Error vs. Image Noise

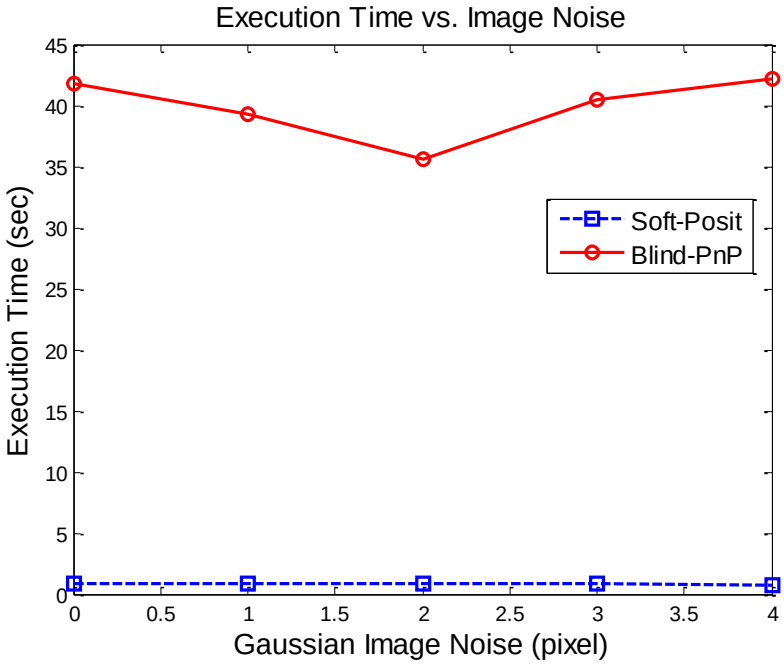

Figure 37 : Execution Time vs. Image Noise

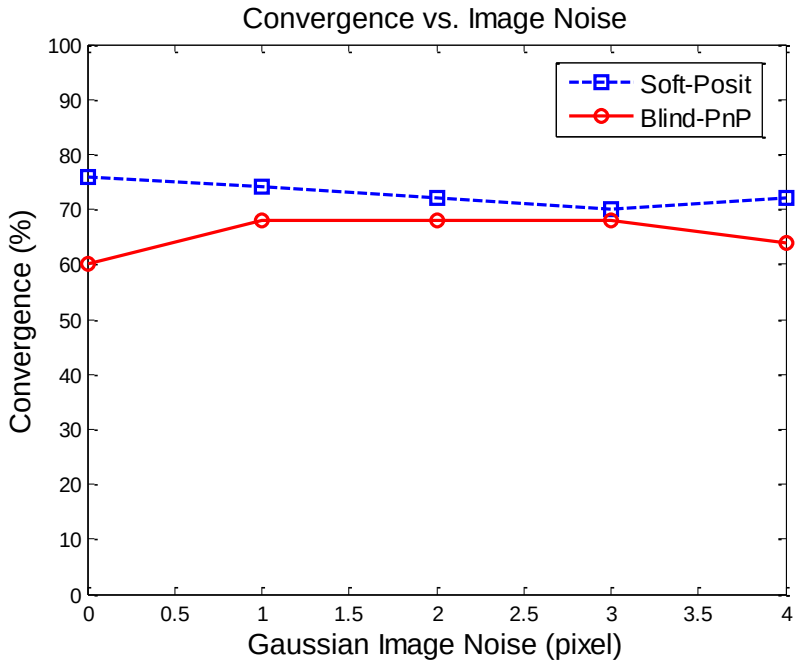

Figure 38 : Convergence Rate vs. Image Noise

4.2.1.2 Effect of Model Point Noise

The effect of model point noise to the performances of Soft-Posit and Blind-PnP algorithms is investigated by performing several experiments. In each experiment, different amount of white Gaussian noise is added to the model points and synthetic data is formed using these noisy points. The created cases are the same as the experiments performed for the algorithms with correspondences. For each experiment, re-projection error, rotation error in x-, *y*-, *z*- axes, translation error in x-, *y*-, *z*- axes and execution time are recorded. When the correspondences are calculated correctly, the algorithm is marked as converged. The convergence ratio indicates the number of experiments that the algorithms converged. An example of the 2D noisy input formed with noisy model points is given in Figure 20 . The results for the performed experiments are given in Table 35, Table 36, Table 37, Table 38, Table 39, Table 40, Table 41, Table 42 and Table 43.

Table 35 : Re-projection Error (pixel) vs. Model Point Noise

	0 mm	0.5 mm	1 mm	1.5 mm	2 mm
Soft-Posit	0.02	0.37	1.43	3.32	5.76
Blind-PnP	0.78	0.92	1.64	3.52	5.80

Table 36 : Rotation Error around *X*-Axis (mrad) vs. Model Point Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.24	5.45	29.08	56.87	65.63
Blind-PnP	6.31	7.45	18.74	61.51	94.63

Table 37 : Rotation Error around *Y*-Axis (mrad) vs. Model Point Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.31	3.85	15.35	46.50	52.39
Blind-PnP	7.29	15.68	11.41	43.56	68.51

Table 38 : Rotation Error around *Z*-Axis (mrad) vs. Model Point Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.17	2.52	14.33	40.89	33.90
Blind-PnP	0.58	5.36	6.89	32.98	66.49

Table 39 : Translation Error along *X*-Axis (mm) vs. Model Point Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.24	0.26	0.77	1.71	1.74
Blind-PnP	0.09	0.34	0.29	1.39	2.58

Table 40 : Translation Error along *Y*-Axis (mm) vs. Model Point Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	7.7e-2	0.14	0.49	0.78	1.90
Blind-PnP	0.25	0.34	0.53	0.98	1.81

Table 41 : Translation Error along *Z*-Axis (mm) vs. Model Point Noise

	0 pixel	1 pixel	2 pixel	3 pixel	4 pixel
Soft-Posit	0.01	0.62	2.41	5.39	4.58
Blind-PnP	0.25	0.96	1.15	4.40	9.38

Table 42 : Execution Time (sec) vs. Model Point Noise

	0 mm	0.5 mm	1 mm	1.5 mm	2 mm
Soft-Posit	0.91	0.92	0.89	0.71	0.97
Blind-PnP	41.78	46.93	48.12	36.37	50.91

Table 43 : Convergence Rate vs. Model Point Noise

	0 mm	0.5 mm	1 mm	1.5 mm	2 mm
Soft-Posit	%76	%76	%68	%68	%76
Blind-PnP	%60	%60	%64	%72	%64

The variation of re-projection error, rotation error (norm of rotation errors in *x*-, *y*-,*z*- axes), translation error (norm of translation errors in *x*-, *y*-,*z*- axes), execution time and convergence rate with different amounts of model point noise can be seen from graphs shown in Figure 39, Figure 40, Figure 41, Figure 42 and Figure 43.

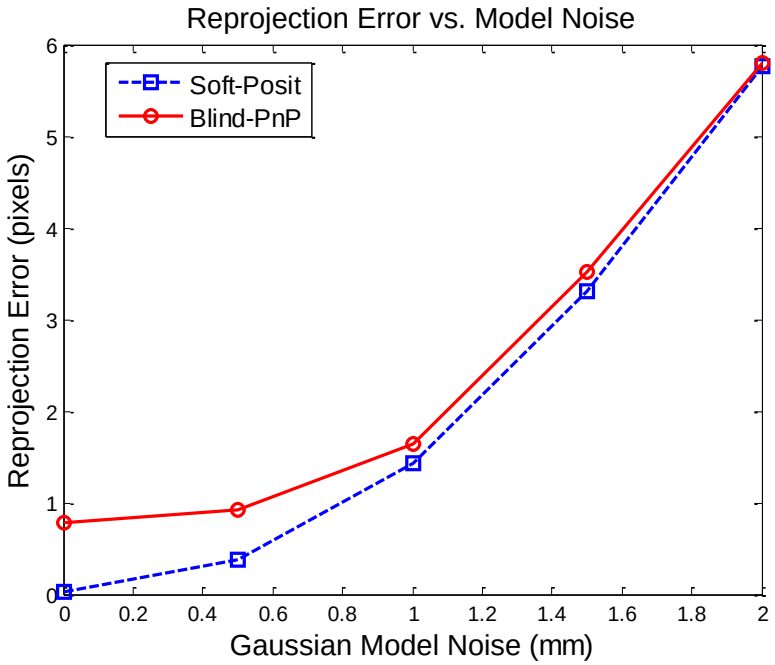

Figure 39 : Re-projection Error vs. Model Point Noise

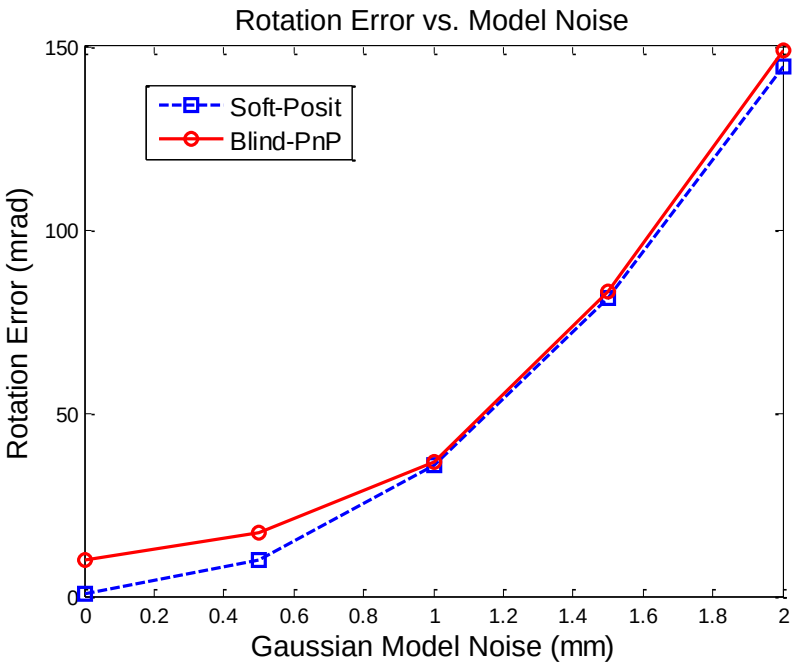

Figure 40 : Rotation Error vs. Model Point Noise

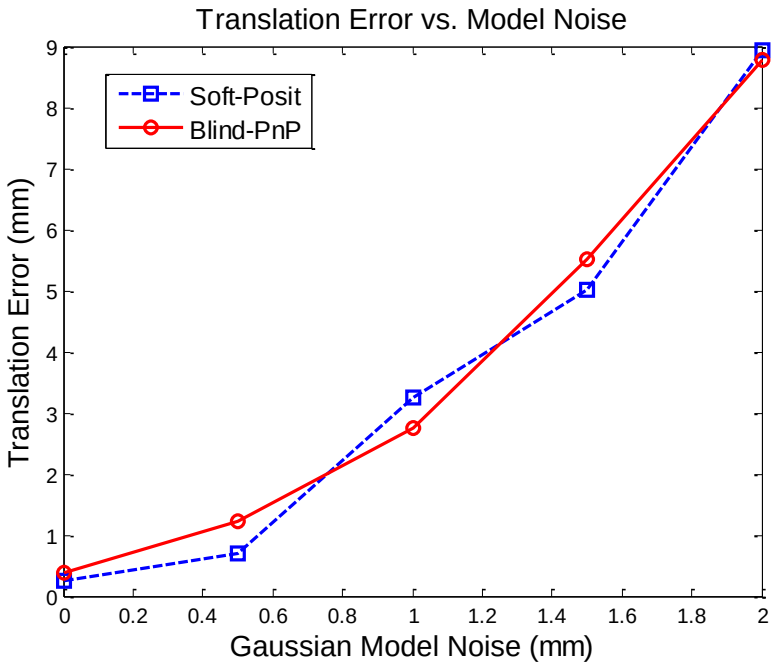

Figure 41 : Translation Error vs. Model Point Noise

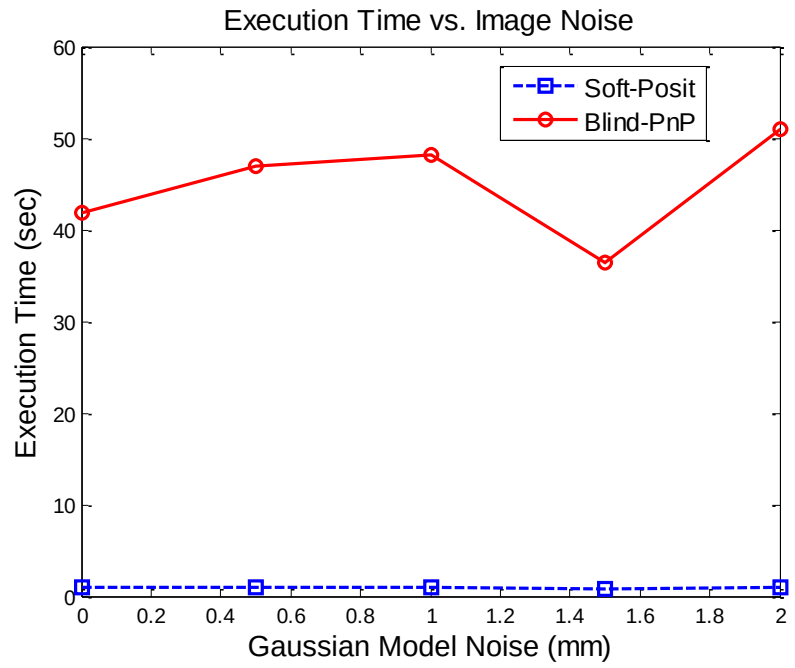

Figure 42 : Execution Time vs. Model Point Noise

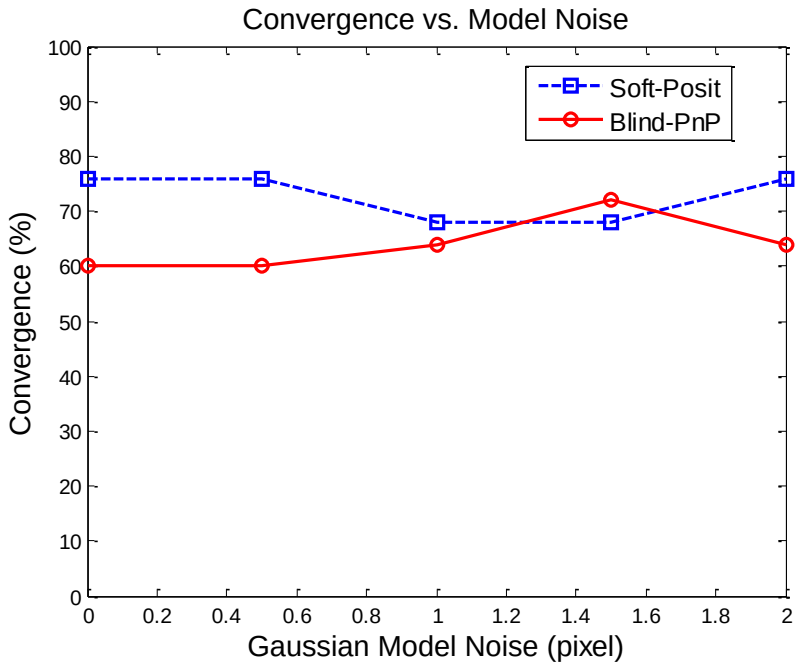

Figure 43 : Convergence Rate vs. Model Point Noise

4.2.1.3 Effect of Clutter

The image points that have no corresponding model points are called *clutter*. Clutter is caused by the errors during segmentation and feature extraction process. In order to investigate the effect of clutter level to the performance of Soft-Posit and Blind-PnP algorithms, several experiments are performed. In each experiment different amount of clutter cases are generated by projecting model points onto the image plane and then inserting additional points. An example for synthetic image points with %30 clutter level is shown in Figure 44. The clutter level is the ratio of additive image points to the number of projected model points. The results for the performed experiments are given in Table 44, Table 45, Table 46, Table 47, Table 48, Table 49, Table 50, Table 51 and Table 52.

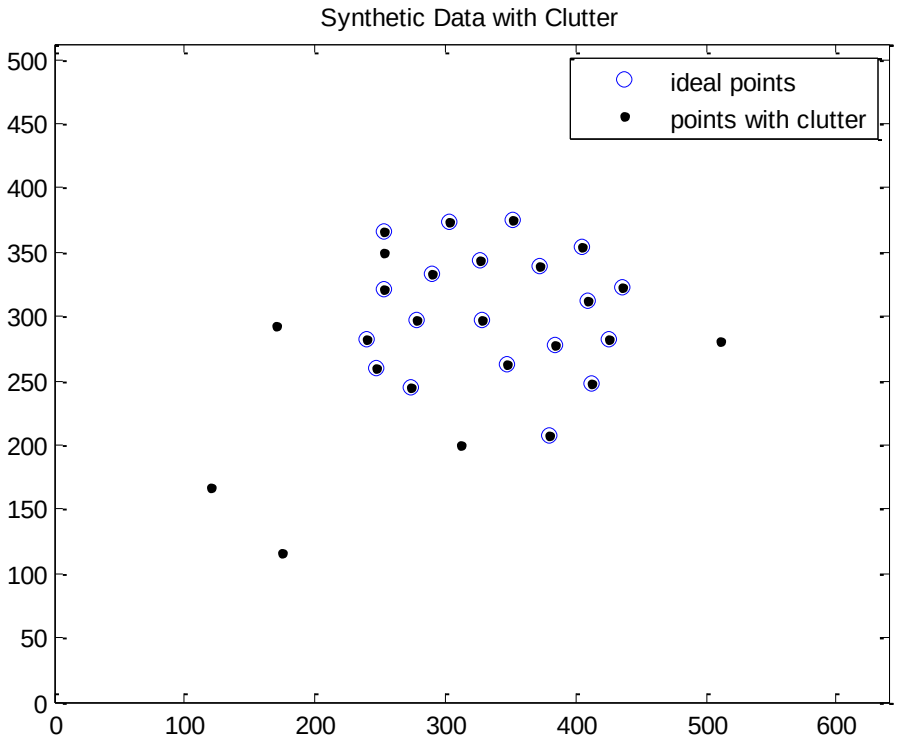

Figure 44 : Image Points with %30 Clutter Level

Table 44 : Re-projection Error (pixel) vs. Clutter

	%0	%10	%20	%30	%40
Soft-Posit	0.02	0.02	0.02	0.03	0.03
Blind-PnP	0.78	0.61	0.84	1.29	1.04

Table 45 : Rotation Error around *X*-Axis (mrad) vs. Clutter

	%0	%10	%20	%30	%40
Soft-Posit	0.24	0.32	0.58	0.31	0.66
Blind-PnP	6.31	4.10	6.74	9.24	7.92

Table 46 : Rotation Error around *Y*-Axis (mrad) vs. Clutter

	%0	%10	%20	%30	%40
Soft-Posit	0.31	0.29	0.61	0.36	0.65
Blind-PnP	7.29	4.19	5.51	7.92	4.53

Table 47 : Rotation Error around *Z*-Axis (mrad) vs. Clutter

	%0	%10	%20	%30	%40
Soft-Posit	0.17	0.19	0.38	0.18	0.36
Blind-PnP	0.58	0.50	0.72	5.32	3.76

Table 48 : Translation Error along *X*-Axis (mm) vs. Clutter

	%0	%10	%20	%30	%40
Soft-Posit	0.24	0.25	0.25	0.25	0.25
Blind-PnP	0.09	0.08	0.74	1.14	0.19

Table 49 : Translation Error along *Y*-Axis (mm) vs. Clutter

	%0	%10	%20	%30	%40
Soft-Posit	7.7e-2	7.8e-2	0.01	70.e-2	0.01
Blind-PnP	0.25	0.14	0.58	0.74	0.32

Table 50 : Translation Error along *Z*-Axis (mm) vs. Clutter

	%0	%10	%20	%30	%40
Soft-Posit	0.01	0.01	0.02	0.03	0.03
Blind-PnP	0.25	0.20	0.60	0.83	1.16

Table 51 : Execution Time (sec) vs. Clutter

	%0	%10	%20	%30	%40
Soft-Posit	0.91	0.88	1.29	1.41	1.69
Blind-PnP	41.78	35.90	36.72	42.64	44.04

Table 52 : Convergence Rate vs. Clutter

	%0	%10	%20	%30	%40
Soft-Posit	%70	%70	%70	%80	%40
Blind-PnP	%60	%70	%80	%60	%60

The variation of re-projection error, rotation error (norm of rotation errors in x-, *y*-, *z*- axes), translation error (norm of translation errors in x-, *y*-, *z*- axes), execution time and convergence rate with different amounts of clutter can be seen from graphs shown in Figure 45, Figure 46, Figure 47, Figure 48 and Figure 49.

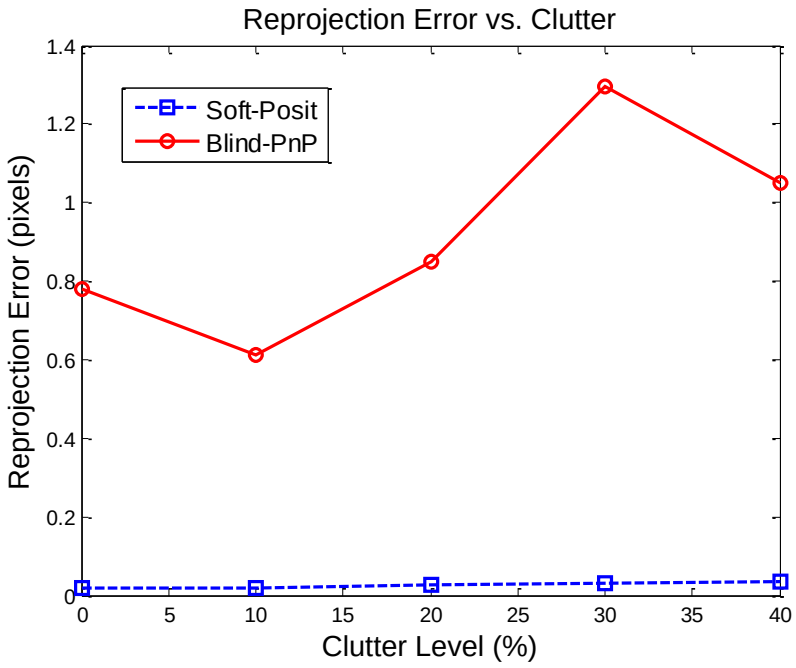

Figure 45 : Re-projection Error vs. Clutter

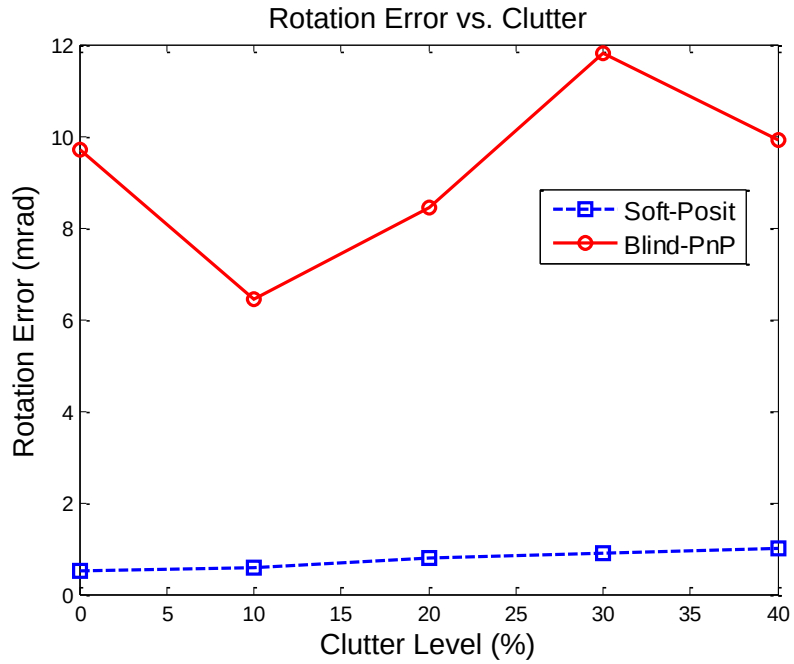

Figure 46 : Rotation Error vs. Clutter

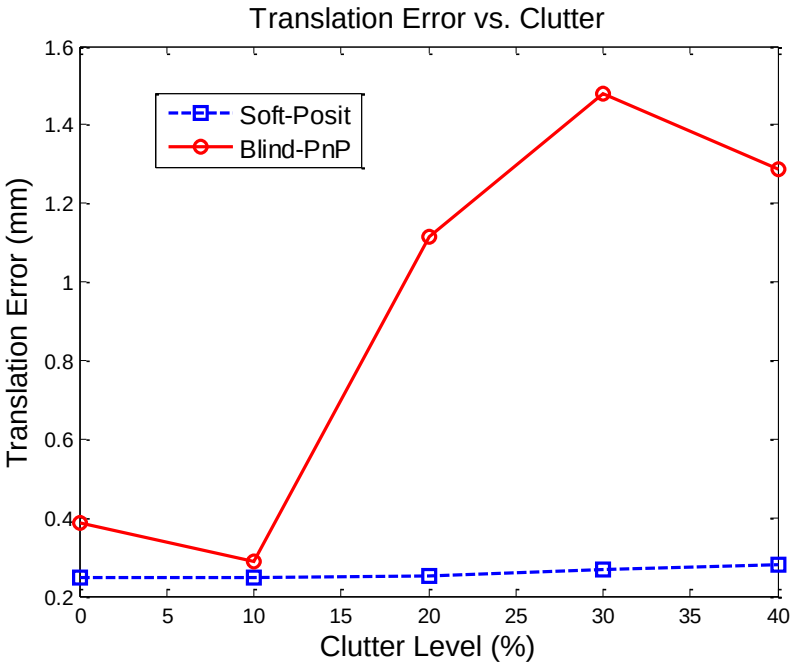

Figure 47 : Translation Error vs. Clutter

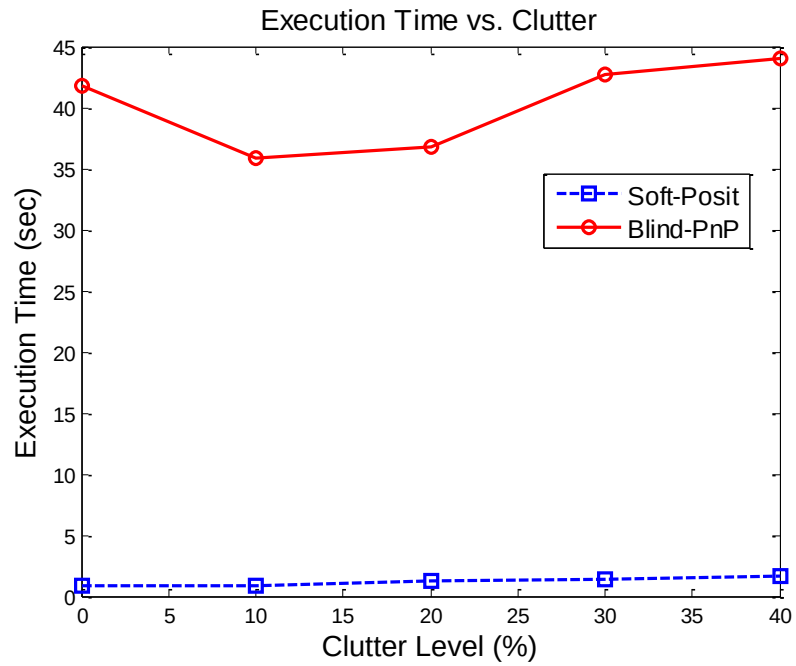

Figure 48 : Execution Time vs. Clutter

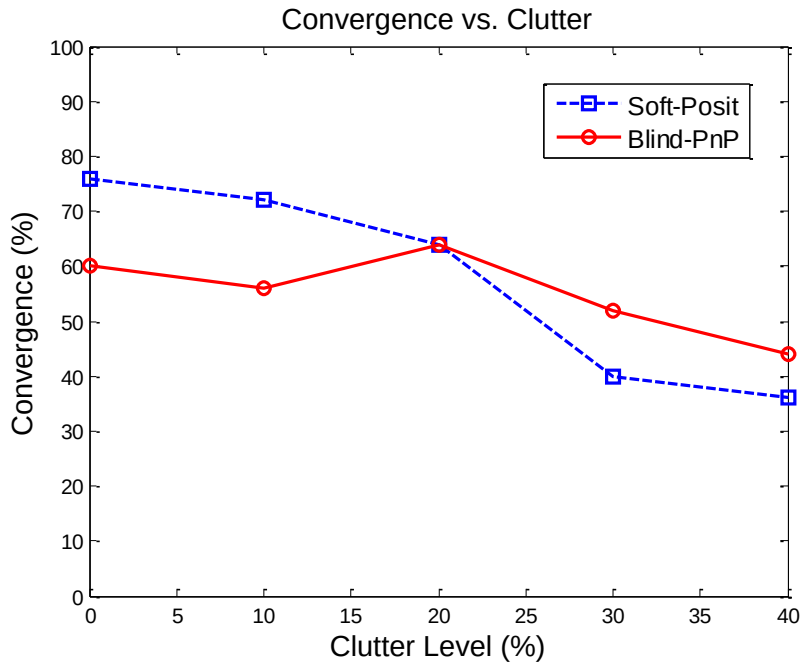

Figure 49 : Convergence Rate vs. Clutter

4.2.1.4 Effect of Occlusion

The model points without corresponding image points are called as occluded points. The occlusion appears when the model points are not observed by the camera, since they are blocked by another object or the object itself. In order to examine the effect of occlusion to performance of Soft-Posit and Blind-PnP algorithms, several experiments are performed. Occlusion case is generated by projecting model points onto the image plane and then removing some of the obtained points. In Figure 50, the occlusion case, when %30 of the model points cannot be observed is shown. The results for the performed experiments are given in Table 53, Table 54, Table 55, Table 56, Table 57, Table 58, Table 59, Table 60 and Table 61.

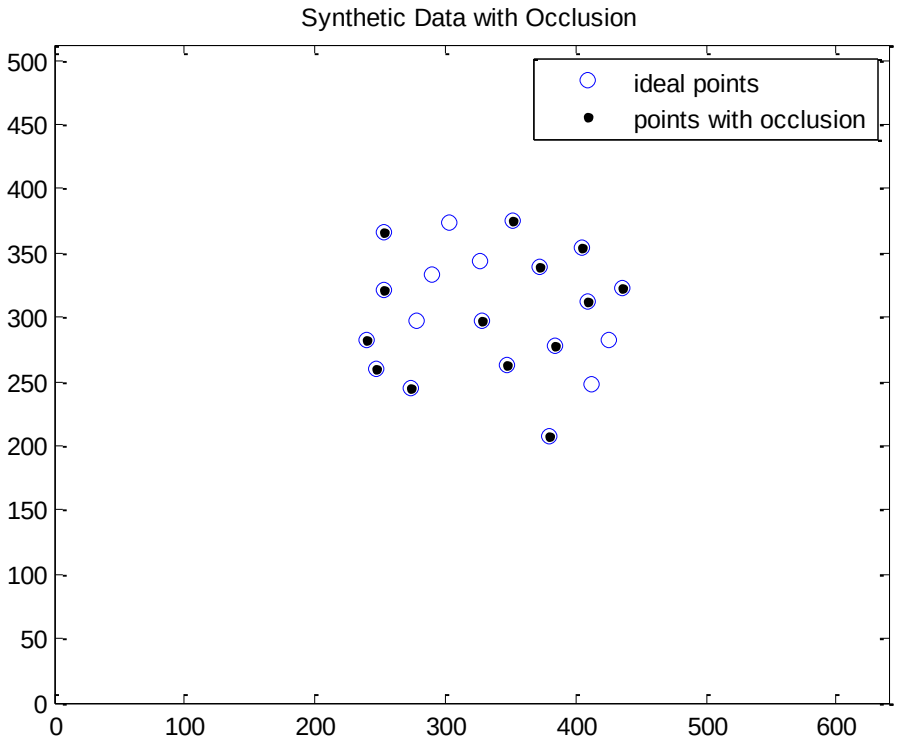

Figure 50 : Image Points with %30 Occlusion Level

Table 53 : Re-projection Error (pixel) vs. Occlusion

	%0	%10	%20	%30	%40
Soft-Posit	0.02	0.02	0.04	0.05	0.06
Blind-PnP	0.78	0.56	0.95	1.22	1.30

Table 54 : Rotation Error around X-Axis (mrad) vs. Occlusion

	%0	%10	%20	%30	%40
Soft-Posit	0.24	0.69	0.54	1.62	0.84
Blind-PnP	6.31	3.94	3.77	13.57	12.06

Table 55 : Rotation Error around Y-Axis (mrad) vs. Occlusion

	%0	%10	%20	%30	%40
Soft-Posit	0.31	0.39	0.33	1.24	0.61
Blind-PnP	7.29	3.59	0.84	15.20	3.74

Table 56 : Rotation Error around Z-Axis (mrad) vs. Occlusion

	%0	%10	%20	%30	%40
Soft-Posit	0.17	0.18	0.36	1.22	0.35
Blind-PnP	0.58	3.47	3.30	8.07	1.60

Table 57 : Translation Error along X-Axis (mm) vs. Occlusion

	%0	%10	%20	%30	%40
Soft-Posit	0.24	0.25	0.24	0.27	0.25
Blind-PnP	0.09	0.14	0.02	0.33	0.04

Table 58 : Translation Error along Y-Axis (mm) vs. Occlusion

	%0	%10	%20	%30	%40
Soft-Posit	7.7e-2	0.02	0.01	0.02	0.02
Blind-PnP	0.25	0.17	0.04	0.62	0.22

Table 59 : Translation Error along Z-Axis (mm) vs. Occlusion

	%0	%10	%20	%30	%40
Soft-Posit	0.01	0.06	0.04	0.06	0.07
Blind-PnP	0.25	0.62	0.39	1.55	1.21

Table 60 : Execution Time (sec) vs. Occlusion

	%0	%10	%20	%30	%40
Soft-Posit	0.91	0.87	0.91	0.77	0.74
Blind-PnP	41.78	33.87	28.69	22.87	17.88

Table 61 : Convergence Rate vs. Occlusion

	%0	%10	%20	%30	%40
Soft-Posit	%76	%72	%52	%44	%20
Blind-PnP	%60	%64	%56	%52	%40

The variation of re-projection error, rotation error (norm of rotation errors in x-, *y*-, *z*- axes), translation error (norm of translation errors in x-, *y*-, *z*- axes), execution time and convergence rate with different amounts of clutter can be seen from graphs shown in Figure 51, Figure 52, Figure 53, Figure 54 and Figure 55.

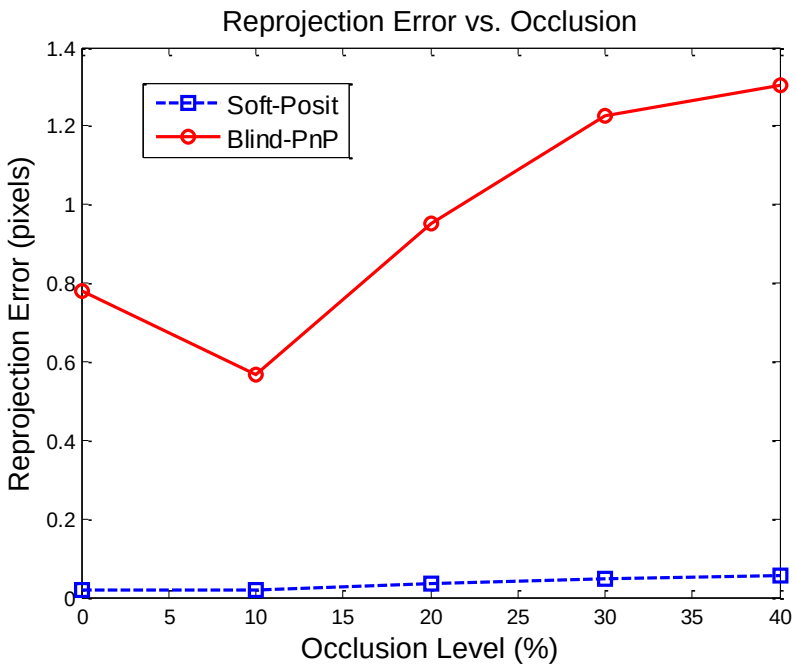

Figure 51 : Re-projection Error vs. Occlusion

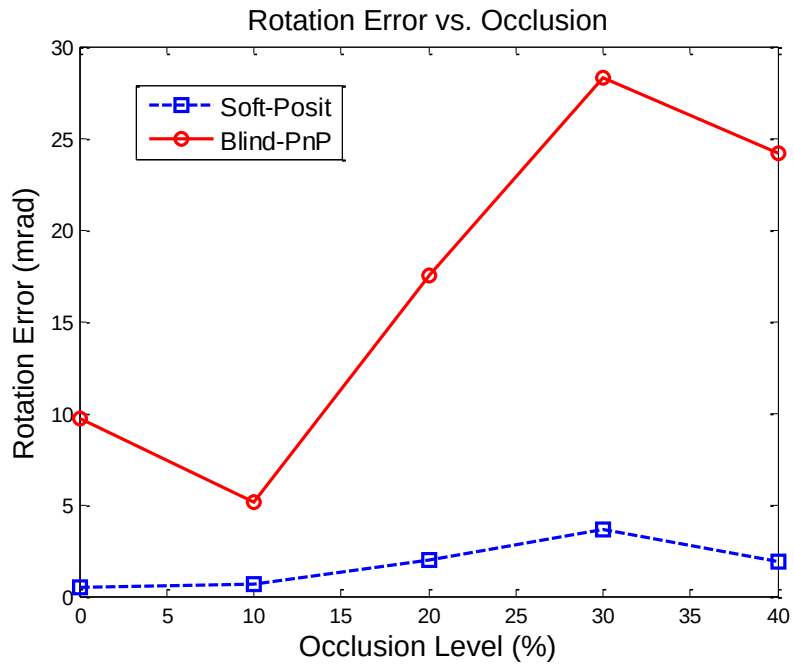

Figure 52 : Rotation Error vs. Occlusion

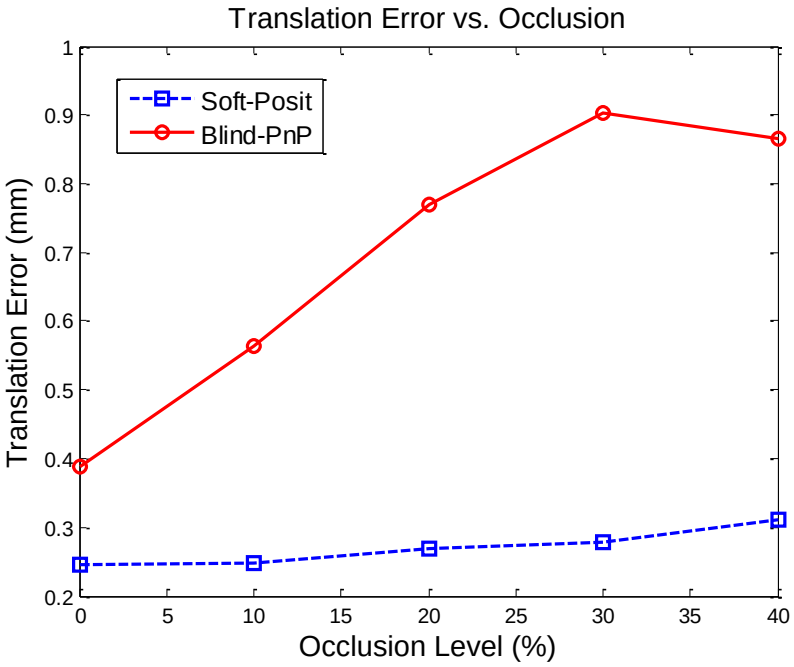

Figure 53 : Translation Error vs. Occlusion

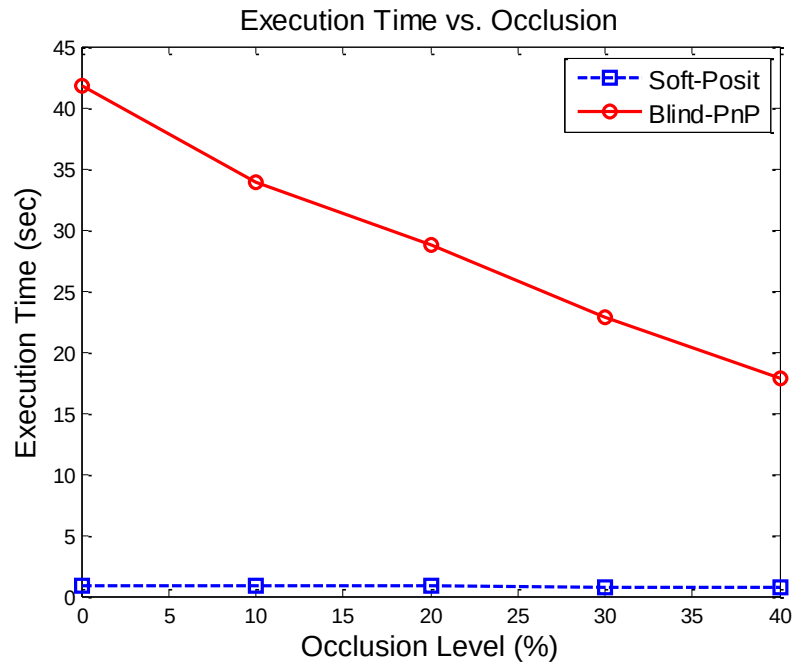

Figure 54 : Execution Time vs. Occlusion

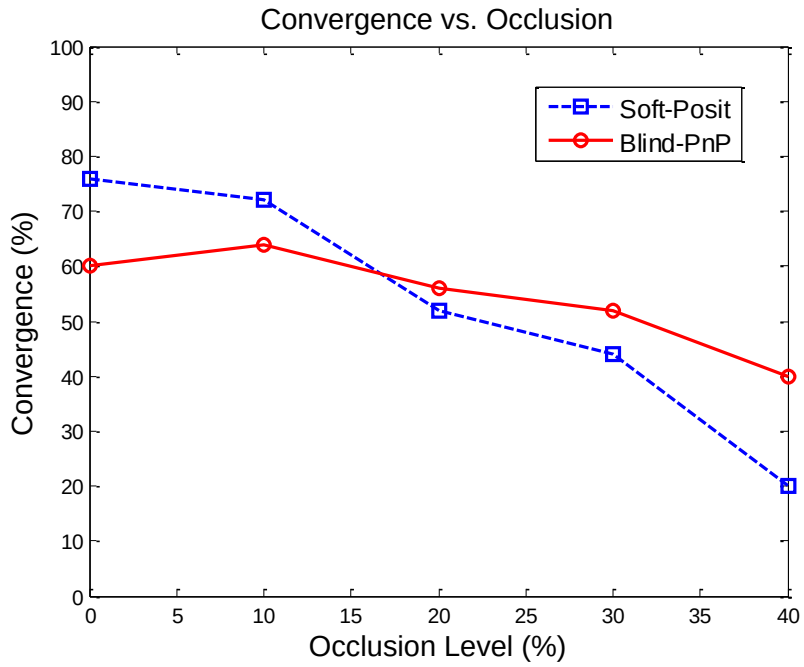

Figure 55 : Convergence Rate vs. Occlusion

4.2.2 Experiments Using Visual Data

Soft-Posit and Blind-PnP algorithms are also tested with some real video sequences captured from the test setup shown in Figure 11. The advantage of experiments using real images is that, one can see the performance of algorithms with realistic inputs. The 2D feature points are obtained by calculating the centroids of the LED points seen on the image. The distortion caused by the lens is removed to calculate the exact centroid locations. In this case, there is no need to give model points and image points in pairs because Soft-Posit and Blind-PnP algorithms can calculate correspondences.

The exact rotation and translation of the head are unknown when the images are taken. Hence, there is no ground-truth information for comparison. The algorithms are compared with respect to their re-projection error and execution time. Flow-chart for experiment using visual data is given in Figure 56

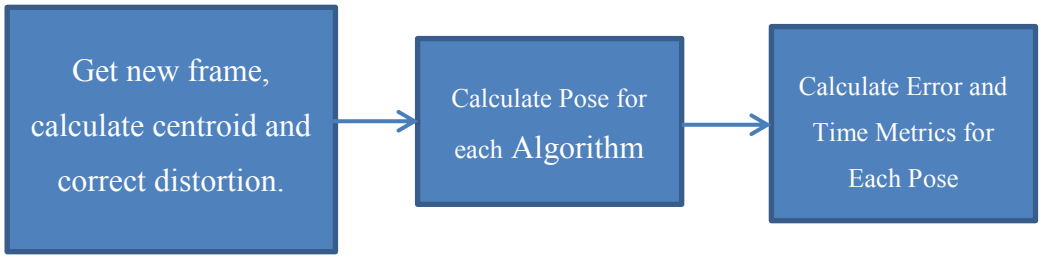

Figure 56 : Flow-Chart for Experiments Using Visual Data

If the algorithm calculates the correspondences correctly, it is marked as converged. Convergence rate defines the ratio of experiments that the algorithm converged to the total number of experiments. The results of the performed experiments are given in Table 62.

Table 62 Results for Experiments Using Visual Data

	Re-projection Error (pixel)	Execution Time (sec)	Convergence Rate (%)
Soft-POSIT	2.31	1.23	52
BPnP	3.96	38 396	64

For visualizing, the results are shown with bar graphs, giving information about the mean values and standard deviation for re-projection errors, execution times and convergence rates Bar graph for re-projection error is given in Figure 57, the graph for execution time is given in Figure 58 and the graph for convergence rate is given in Figure 59.

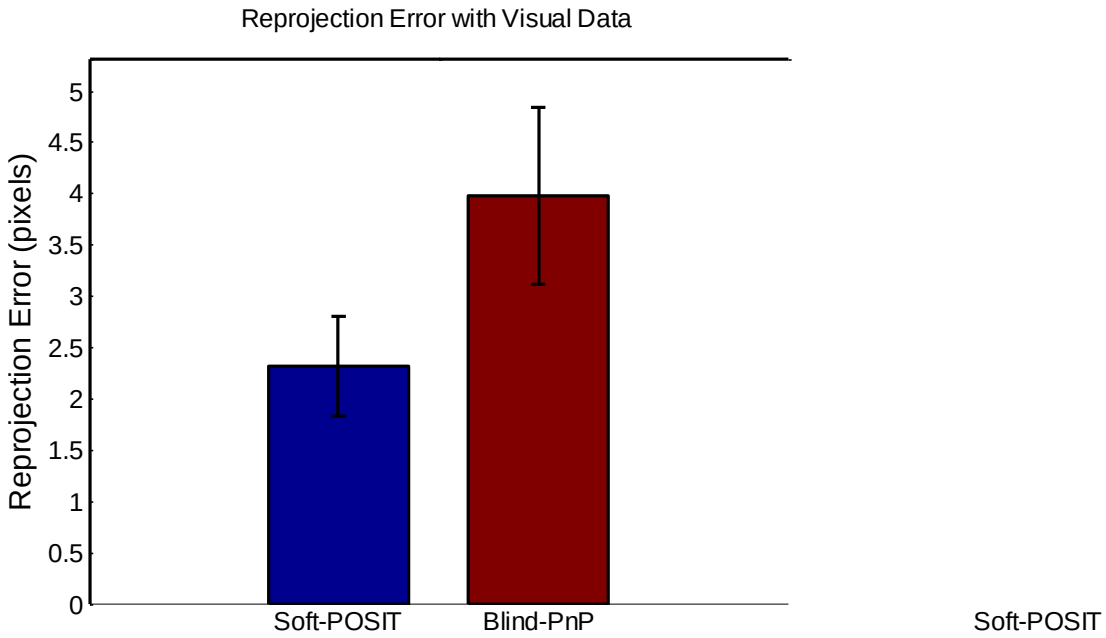

Figure 57 : Re-projection Errors for Visual Data Experiment

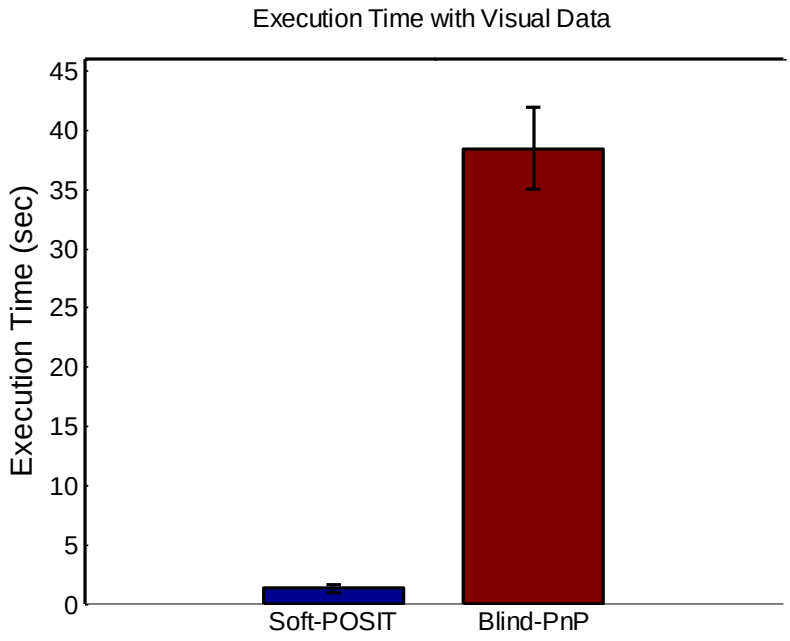

Figure 58 : Execution Times for Visual Data Experiment

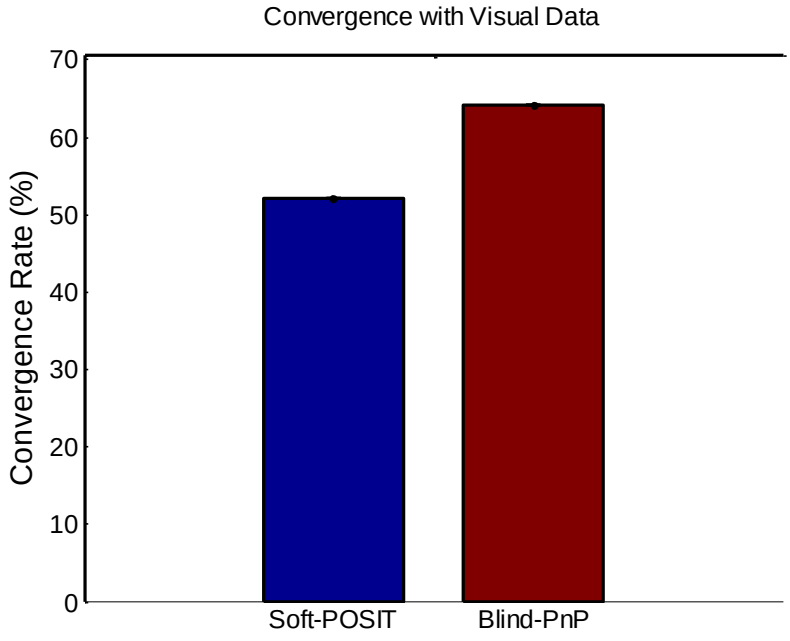

Figure 59 : Convergence Rates for Visual Data Experiment

CHAPTER 5

CONCLUSIONS

The purpose of this thesis study is to compare some well-known pose estimation algorithms and investigate their performance under several circumstances such as image noise, model point noise and outlier. The compared algorithms can be classified in two groups: algorithms with known correspondences and algorithms without correspondences. The experiments for comparison of the algorithms are performed in two steps: first using synthetic data and then using real visual images.

The algorithms with known correspondences are *POSIT*, *DLT*, *EPnP* and *ORTHIT*. *POSIT* and *ORTHIT* algorithms are iterative algorithms, which coarsely minimize re-projections errors. *EPnP* and *DLT* algorithms are closed form algorithms, which approach the solution of the pose estimation problems by forming linear equations and solving them. Several amounts of image point noise, model point noise and outlier are introduced to the algorithms and their performances under these conditions are observed. The conclusions based on simulation results can be summarized as follows:

- ✓ In ideal case, when there is no image noise, no model point noise and no outlier, it is observed that closed form algorithms perform better than iterative ones. From the results it can be said that *DLT* is the best algorithm according to the error values and *POSIT* algorithm could be the worst.

- ✓ As the image noise level is increased, the re-projection, rotation and translation error values for all algorithms also increase. It can be observed that closed form algorithms are affected more than iterative algorithm. The performances of *POSIT* and *ORTHIT* algorithms are nearly the same, but it can be said that *ORTHIT* algorithm performs best under noisy conditions. The performance of *DLT* algorithm decreased dramatically with increasing noise level and it can be considered as the worst algorithm in noisy cases. As expected, the noise in model points affects the algorithms just as the image noise does.

- ✓ The performances of all algorithms are mostly affected by outlier level. A small increase in outlier level causes huge degradations in performances. The most affected algorithm is *DLT* as expected. Again it can be stated that iterative algorithms are less susceptible to outlier level than closed form algorithms. *POSIT* algorithm can be considered to perform best with increasing outlier level.

- ✓ The experiments using visual data is expected to give the most reliable results for comparison of algorithms. The problem while using real image is that, the ground-truth values for rotation and translation values are not known, and hence, errors in rotation and translation cannot be calculated. The only error metric that can be used for comparison is re-projection error. It can be observed from the results that with a realistic input, *POSIT* performs better than any of the compared algorithms. The performance of *ORTHIT* algorithm is slightly inferior to *POSIT*, but still be considered to perform well. The *DLT* algorithm can be stated to be the worst algorithm with realistic input.

- ✓ If the algorithms will be used in real-time applications, the execution times are also important. From the experiment results, it can be said that *DLT*

algorithm has the best real-time performance of all. Although the accuracy of *ORTHIT* algorithm is good, the real-time performance of it is very poor compared to the others.

- ✓ As a summary, considering the accuracy and real-time performances, *POSIT* algorithm can be stated as the best algorithm among the compared algorithms. In terms of the accuracy, *DLT* can be said to be the worst algorithm, whereas with real-time concern, *ORTHIT* algorithm can be considered to be the worst performing method.

The algorithms which calculate the pose and correspondences simultaneously are *Soft-POSIT* and *Blind-PnP*. These two algorithms are compared using synthetic data with different amounts of circumstances such as image noise, model point noise, clutter and occlusion. Experiments with real images are also used for comparison purpose. The conclusions can be summarized as follows:

- ✓ The comparison of algorithms without correspondences is more problematic. This is due to the fact that the pose estimation performances of the algorithms are also affected by their correspondence calculation performances. If an algorithm calculates the correspondences erroneously, then the re-projection, rotation and translation error values become meaningless for comparison. Hence, the metrics for comparison of the algorithms are calculated only when the algorithms succeeds to calculate enough number of correspondences. *Convergence Rate* metric shows the performances of algorithms for correspondence problem.

- ✓ As the noise levels in image and model point increase, the re-projection, rotation and translation error values also tend to increase for both algorithms. The convergence rate of *Soft-POSIT* algorithm decreases as the noise level increases while convergence of *Blind-PnP* algorithm can be

stated as irresponsive. The accuracy of Soft-Posit algorithm can be considered to be slightly better than *Blind-PnP* according to results.

- ✓ The increasing occlusion and clutter levels make the correspondence problem harder for both algorithms. The error metrics are calculated when the algorithms calculates correspondences successfully; hence, it can be observed that error values are slightly affected by occlusion and clutter levels. The convergence performance of *Soft-POSIT* algorithm is more susceptible to occlusion and clutter levels. Considering the convergence rate, *Blind-PnP* can be stated as immune to clutter, but with increasing levels of occlusion, its convergence rate decreases.

- ✓ The experiments using real images can give more reliable results. According to the results, the accuracy of the calculated pose values can be stated to be better for *Soft-POSIT* algorithm. The dominant disturbance case while working with real images of such a curved 3D object is the occlusion. It can be seen from the results that the convergence rate for *Blind-PnP* algorithm is better than *Soft-POSIT*.

- ✓ While using the algorithms in real-time applications, execution time is one of the main concerns. According to the results, *Blind-PnP* execution time performance can be considered to be unacceptable for a real-time application. *Soft-POSIT* real time performance is much better.

- ✓ Both algorithms are tested without prior knowledge of initial pose of the object. For *Blind-PnP*, only the pose prior values are used and for Soft-Posit an arbitrary initial pose value is given. During the experiments, it is observed that, the performances of both algorithms are very dependent on the initial pose values given. When the real rotation and translation values get far away from the initial values, the algorithms usually fail to compute the correspondence and pose values. It can be concluded that, the accuracy

and real-time performances of both algorithms can be improved by giving meaningful initial pose estimates.

- ✓ As summary, considering accuracy and real-time performances, *Soft-POSIT* algorithm can be stated to perform better than *Blind-PnP*. According to the experience gained from the experiments, both algorithms are inappropriate for the use in head tracking applications directly. Further processes on input data must be performed before using these algorithms. Occlusion cases must be minimized by removing the model points which cannot be seen by the cameras. The previously calculated pose value can be given as an initial estimate to the algorithms. With these improvements in the input data, both algorithms should perform better for both accuracy and time concerns.

REFERENCES

1. G. Taylor & L. Kleeman, “Flexible Self-Calibrated Visual Servoing for a Humanoid Robot”, In Proc. of the Australian Conference on Robotics and Automation, pp. 79–84, Sydney, Australia, November 2001.

2. B. Rosenhahn, O. Granert & G.Sommer, “Monocular Pose Estimation of Kinematic Chains”, Applied Geometric Algebras for Computer Science and Engineering, Birkhauser Verlag, pp.371–381, 2002.

3. L. Tessens, R. Kehl, A. Pizurica, L. Van Gool & W. Philips, “A Real-Time Optical Head Tracker Based on 3D Prediction and Correction”, Proceedings of SPS-DARTS, 2006.

4. A. Ansar & K. Daniilidis, “Linear Pose Estimation from Points or Lines”, Proc. European Conf. Computer Vision, vol. 4, pp. 282-296, May 2002.

5. R. Kumar & A. R. Hanson, “Robust Methods for Estimating Pose and a Sensitivity Analysis”, Computer Vision and Image Analysis, vol. 60, pp. 313-342, 1994.

6. P. David, D. DeMenthon, R. Duraiswami & H. Samet, “Simultaneous Pose and Correspondence Determination Using Line Features”, In Computer Vision and Pattern Recognition, Vol. 2, pp. 424-431, 2003.

7. Y. L. Abdel-Aziz & H. Karara, “Direct Linear Transformation from Comparator Coordinates into Object Space Coordinates in Close-Range

Photogrammetry", In Proc. ASP/UI Symp. Close-Range Photogrammetry, pp. 1-18, 1971.

8. R. Hartley & A. Zisserman, "Multiple View Geometry in Computer Vision", Cambridge University Press, 2003.

9. C. Lu, G. Hager & E. Mjolsness, "Fast and Globally Convergent Pose Estimation from Video Images", IEEE Trans. Pattern Analysis and Machine Intelligence, vol.11, no. 6, pp. 610-622, June 2000.

10. M. Fischler & R. Bolles, "The Random Sample Consensus Set: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", Comm. ACM, vol. 24, no. 6, pp. 381-395, 1981.

11. D. DeMenthon & L. S. Davis, "Model-Based Object Pose in 25 Lines of Code", International Journal of Computer Vision, vol. 15, pp. 123-141, 1995.

12. P. D. Fiore, "Efficient Linear Solution of Exterior Orientation", IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 23, pp. 140-148, 2001.

13. L. Quan & Z. Lan, "Linear N-Point Camera Pose Determination", IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 21, pp. 774-780, 1999.

14. B. Triggs, "Camera Pose and Calibration from 4 or 5 Known 3D Points", International Conference on Computer Vision, pp. 278-284, 1999.

15. V. Lepetit, F. Moreno-Noguer & P. Fua, “EPnP: An Accurate O(n) Solution to the PnP Problem”, International Journal of Computer Vision, vol. 81, pp. 155–166, 2009.

16. D. Oberkampf, D. DeMenthon & L. S. Davis, “Iterative Pose Estimation Using Coplanar Feature Points”, Computer Vision and Image Understanding, vol. 63, pp. 495-511, 1996.

17. M. Dhome, M. Richetin, & J.–T. Lapreste, “Determination of the Attitude of 3D Objects from A Single Perspective View”, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 11, pp. 1265-1278, 1989.

18. X. S. Gao, X. R. Hou, J. Tang & H. F. Cheng, “Complete Solution Classification for the Perspective-Three-Point Problem”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, pp. 930-943, 2003.

19. R.M. Haralick, C. Lee, K. Ottenberg and M. Nolle, “Analysis and Solutions of the Three Point Perspective Pose Estimation Problem,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 592-598, 1991.

20. R. Horaud, B. Canio, and O. Leboullenx, “An Analytic Solution for the Perspective-Four-Point Problem”, Computer Vision, Graphics and Image Processing, vol. 47, pp. 33 - 44, 1989.

21. D.G Lowe, “Fitting Parametrized Three-Dimensional Models to Images”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, pp. 441-450, 1991.

22. F. Moreno-Noguer, V. Lepetit & P. Fua, “Pose Priors for Simultaneously Solving Alignment and Correspondence”, In 10th European Conference on Computer Vision, 2008.

23. P. David, D. DeMenthon, R. Duraiswami & H. Samet, “Soft-POSIT: Simultaneous Pose and Correspondence Determination”, International Journal of Computer Vision, Vol. 49, pp. 259-284, 2004.

24. J. Oliensis & R. Hartley, “Iterative Extensions of the Sturm/Triggs Algorithm: Convergence and Nonconvergence”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 29, pp. 2217-2233, December 2007.

25. P. Sturm & B. Triggs, “A Factorization Based Algorithm for Multi-Image Projective Structure and Motion”, Proc. European Conference on Computer Vision, pp. 709-720, 1996.

26. C. Tomasi & T. Kanade, “Shape and Motion from Image Streams under Orthography: A Factorization Method”, International Journal of Computer Vision, vol. 9, pp. 137–154, 1992.

27. Y. Lamdan, H. J. Wolfson, “Geometric Hashing: A General and Efficient Model Based Recognition Scheme”, Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 238-249, 1988.

28. J. B. Burns, R. S. Weiss, E. M. Riserman, “View Variation of Point-Set Line Segment Features”, IEEE Trans. Pattern Analysis and Machine Intelligence, vol.15, pp. 51-68, 1993.

29. C. F. Olson, “Efficient Pose Clustering Using a Randomized Algorithm”, International Journal of Computer Vision, vol. 23, pp. 131–147, 1997.

30. F. Jurie, “solution of the Simultaneous Pose and Correspondence Problem Using Gaussian Error Model”, Computer Vision and Image Understanding, vol. 73, pp. 357-373, 1999.

31. J. R. Beveridge & E. M. Riserman, “Optimal Geometric Model Matching Under Full 3D Perspective”, Computer Vision and Image Understanding, vol. 61, pp. 351-364, 1995.

32. P. Wunsch & G. Hirzinger, “Registration of CAD Models to Images by Iterative Inverse Perspective Matching”, International Conference on Pattern Recognition, vol.1, pp. 78-83, 1996.

33. M. Figueiredo & A.K.Jain, "Unsupervised Learning of Finite Mixture Models", IEEE Trans. Pattern Analysis and Machine Intelligence, vol.24, pp. 381-396, 2002.

34. J. Heikkilä & O. Silvén, “A Four-Step Camera Calibration Procedure with Implicit Image Correction”, Proc. IEEE Comput. Soc. Conf. Computer Vision and Pattern Recognition (CVPR'97), pp. 1106 - 1112, 1997.

35. S. Gold, A. Rangarajan, C.P. Lu, S. Pappu & E. Mjolsness. "New Algorithms for 2D and 3D Point Matching: Pose Estimation and Correspondence," Pattern Recognition, vol. 31, pp. 1019-1031, August 1998.

36. R. Sinkhorn, “A Relationship between Arbitrary Positive Matrices and Doubly Stochastic Matrices”, Annals of Mathematical Statistics, vol. 35, pp. 876-879, 1964.

37. J. S. Bridle, “Training Stochastic Model Recognition as Networks can Lead to Maximum Mutual Information Estimation of Parameters”, Proc. Advances in Neural Information Processing Systems, pp. 211-217, 1990.

38. L. Davis, F. G. Hamza-Lup & J. P. Rolland, “A Method for Designing Marker-Based Tracking Probes”, in Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR), pp. 120-129, 2004.

39. Niu, X. M. & Sakurai, T., “A Method for Finding the Zeros of Polynomials Using a Companion Matrix”, Japan J. Industr. Appl. Math., vol.20, pp. 239-256, 2003.

APPENDIX A

SOLUTION OF P3P PROBLEM

Determining the positions of the three vertices of a triangle from their projections is called the *P3P* problem. Positions of the vertices with respect to each other, the image coordinate for each vertex and internal parameters of the camera are known values for the *P3P* problem [10].

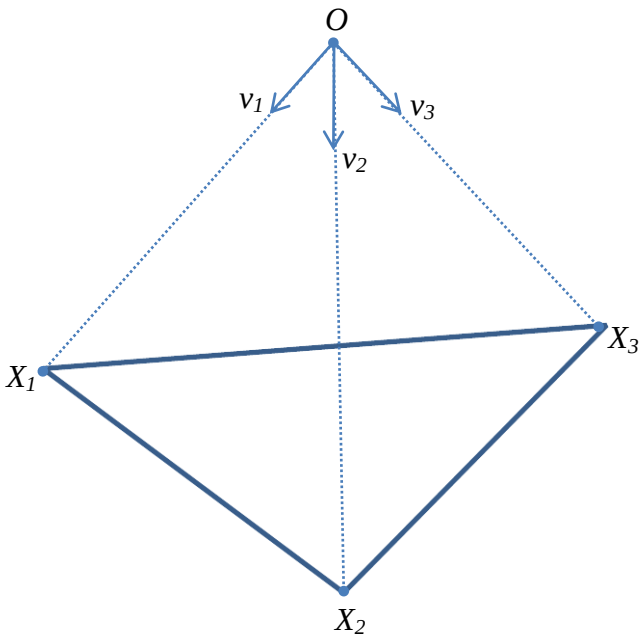

Figure 60 : The *P3P* Problem

In Figure 60, X_1, X_2 and X_3 are the scene points, O is *center of projectivity*. Let α denote the angle between v_1-v_2, β denote the angle between v_2-v_3 and γ denote the angle between v_1 and v_3. The distance between X_1-X_2 is represented by R_{ab}, distance between X_2-X_3 is represented by R_{bc} and distance between X_1-X_3 is represented by R_{ac}. The aim is to calculate the distances, $a = \|X_1 - O\|$, $b = \|X_2 - O\|$ and $c = \|X_3 - O\|$. One can form three set of quadratic equations using the law of cosines as in (A.1) [10].

$$\begin{aligned} R_{ab}^2 &= a^2 + b^2 - 2ab\cos\alpha \\ R_{ac}^2 &= a^2 + c^2 - 2ac\cos\gamma \\ R_{bc}^2 &= b^2 + c^2 - 2bc\cos\beta \end{aligned} \tag{A.1}$$

By introducing new variables x and y such that $b = xa$ and $c = ya$, one can rewrite (A.1) as below:

$$\begin{aligned} R_{ab}^2 &= a^2 + x^2a^2 - 2a^2x\cos\alpha \\ R_{ac}^2 &= a^2 + y^2a^2 - 2a^2y\cos\gamma \\ R_{bc}^2 &= x^2a^2 + y^2a^2 - 2a^2xy\cos\beta \end{aligned} \tag{A.2}$$

(A.2) can be written as in (A.3).

$$\begin{aligned} R_{bc}^2(1 + x^2 - 2x\cos\alpha) &= R_{ab}^2(x^2 + x^2 - 2xy\cos\beta) \\ R_{bc}^2(1 + y^2 - 2y\cos\gamma) &= R_{ac}^2(x^2 + x^2 - 2xy\cos\beta) \end{aligned} \tag{A.3}$$

Denote $K_1 = (R_{bc}/R_{ac})^2$ and $K_2 = (R_{bc}/R_{ab})^2$

$$\begin{aligned} 0 &= (1-K_1)y^2 + 2(K_1 \cos\gamma - x\cos\beta)y + (x^2 - K_1) \\ 0 &= y^2 + 2(-x\cos\beta)y + [x^2(1-K_2) + 2xK_2\cos\alpha - K_2] \end{aligned} \tag{A.4}$$

By replacing the polynomials in *x* with *m, p, q, m', p', q',* the following equation is obtained [10]:

$$\begin{aligned} 0 &= my^2 + py + q \\ 0 &= m'y^2 + p'y + q' \end{aligned} \tag{A.5}$$

Eliminating the terms with y^2 and then eliminating *y* yields:

$$0 = (m'q - mq')^2 - (pm' - p'm)(p'q - pq') \tag{A.6}$$

After regrouping the terms, the following equation is obtained

$$0 = G_4x^4 + G_3x^3 + G_2x^2 + G_1x + G_0 \tag{A.7}$$

where,

$$\begin{aligned} G_4 &= (K_1K_2 - K_1 - K_2)^2 \\ G_3 &= 4(K_1K_2 - K_1 - K_2)K_2(1-K_1)\cos\alpha \\ &\quad + 4K_1\cos\beta\,[(K_1K_2 - K_1 + K_2)\cos\gamma \\ &\quad + 2K_2\cos\alpha\cos\beta] \end{aligned} \tag{A.8}$$

$$G_2 = [2K_2(1-K_1)\cos\alpha]^2 + 2(K_1K_2 - K_1 - K_2)(K_1K_2 + K_1 - K_2) + 4K_1[(K_1 - K_2)(\cos\beta)^2 + K_1(1-K_2)(\cos\gamma)^2 - 2(1+K_1)K_2\cos\alpha\cos\beta\cos\gamma]$$

$$G_1 = 4(K_1K_2 + K_1 - K_2)K_2(1-K_1)\cos\alpha + 4K_1[(K_1K_2 - K_1 + K_2)\cos\beta\cos\gamma + 2K_1K_2\cos\alpha(\cos\gamma)^2]$$

$$G_0 = (K_1K_2 + K_1 - K_2)^2 - 4{K_1}^2K_2(\cos\gamma)^2$$

The solution of the *P3P* problem can be summarized as follows:

i. Compute G_0, G_1, G_2, G_3, G_4 by using (A.8).
ii. Find the roots of (A.7) by *companion matrix method* [39]. Up to four real solutions are obtained.
iii. Compute *a* by using (A.2), compute *y* by using (A.5) and compute *b*, *c*.
iv. Compute the point *O* from *a*, *b*, *c* by using *trilateration* [10].
v. Compute λ_1 by using the equation below:

$$|\lambda_1|\|v_i\| = \|x_i - O\|, \quad i = 1,2,3 \tag{A.9}$$

vi. Compute matrix *R* by using the following equation:

$$\lambda_1 v_i = R(x_i - O), \quad i = 1,2,3 \tag{A.10}$$

Multiple solutions can be obtained from this solution. The correct solutions can be selected by using the physical properties of the camera, scene and images.