Robotica (2017) volume 35, pp. 2278–2296. © Cambridge University Press 2017
doi:10.1017/S0263574716000874

Mirage: an $O(n)$ time analytical solution to 3D camera pose estimation with multi-camera support

Semih Dinc[†]*, Farbod Fahimi[‡] and Ramazan Aygun[†]

[†]*Department of Computer Science, University of Alabama in Huntsville, Huntsville, Alabama*
[‡]*Mechanical & Aerospace Engineering, University of Alabama in Huntsville, Huntsville, Alabama.*
E-mails: ff0002@uah.edu, aygunr@uah.edu

(Accepted December 20, 2016. First published online: February 16, 2017)

SUMMARY
Mirage is a camera pose estimation method that analytically solves pose parameters in linear time for multi-camera systems. It utilizes a reference camera pose to calculate the pose by minimizing the 2D projection error between reference and actual pixel coordinates. Previously, Mirage has been successfully applied to trajectory tracking (visual servoing) problem. In this study, a comprehensive evaluation of Mirage is performed by particularly focusing on the area of camera pose estimation. Experiments have been performed using simulated and real data on noisy and noise-free environments. The results are compared with the state-of-the-art techniques. Mirage outperforms other methods by generating fast and accurate results in all tested environments.

KEYWORDS: Camera pose estimation; Perspective-N-Point (PnP); Object localization; Analytical solution; $O(n)$ time complexity.

1. Introduction

Camera pose estimation is the problem of calculating six pose (position and orientation) parameters of calibrated camera(s) using a set of point correspondences in 3D space and their projections on 2D image plane. Pose estimation plays a critical role in many applications such as target tracking, robotics, and augmented reality.[1] In target tracking and robotic applications, estimated pose generates a feedback to localize the vehicle, or assists some other sensors for decision making during the motion. For augmented reality applications, it is crucial to estimate the pose of the camera precisely, so that the artificial images can be inserted into the display properly. In the literature, the problem has been studied as the *Perspective-n-Point (PnP)*[2] or *object localization*[3] depending on the problem domain.

The number of 2D to 3D point correspondences is an important factor in the pose estimation. It is often preferred to solve the pose estimation problem with small number of points to reduce the complexity of the system. The general case of the problem is PnP where n is an arbitrary number. If the number of points is restricted to 3 ($n = 3$), the problem becomes P3P. Similarly, it is named as P4P if minimum number of 4 points ($n = 4$) are necessary. The methods to solve the PnP problem can be categorized as *numerical* and *analytical* methods. Majority of the existing solutions in the literature are based on numerical solutions, where an iterative approach is expected to minimize the position error and converge to a good solution.[4] Iterative methods can quickly converge to a solution with large number of points. However, they require a good initial estimation of the parameters. Also they are susceptible to be trapped in local minima in the solution space.[5] On the other hand, analytical methods provide closed form solutions, in which the pose parameters are solved directly as unknowns (i.e., pose parameters) of equations that are derived from matching feature points. Nevertheless, the noise sensitivity of analytical methods and their high time complexity ($O(n^k), k \geq 2$) make them less suitable for real-time applications. Despite the development of new linear analytical methods,[6,7] reliability and processing time for real-time applications are still issues for pose estimation.

* Corresponding author. E-mail: sd0016@uah.edu

Another important factor in the domain has been emerged by the recent advancements of vision systems. As hardware prices were reduced, multi-camera systems have become available at reasonable prices to users. There is a clear advantage of using a multi-camera system over a single-camera system since the multi-camera systems provide an extended field of view and allow depth calculations via stereo vision.[8] However, multi-camera PnP problem has rarely been studied in the literature. Majority of existing studies were designed for a single-camera structure. Clearly, this does not indicate that the existing techniques cannot be extended, but rather the extension is unclear and not tested. For instance, some methods[9,10] may require the same feature points visible in all cameras, while some others[8,11,12] do not have this requirement. Therefore, the new techniques should explicitly address how to deal with feature points from multiple cameras.

In this paper, we present a comprehensive evaluation of Mirage pose estimation method. Mirage has been applied to trajectory tracking (or visual servoing) problem and satisfactory results have been achieved on a 3DOF non-holonomic vehicle.[13] The goal of this study is to investigate the robustness and speed of the algorithm under various realistic conditions. Mirage analytically solves six pose parameters in $O(n)$ time for a multi-camera system. Traditional techniques use direct 2D to 3D correspondence of the feature points to calculate the (actual) pose. However, Mirage follows an indirect approach by defining a reference pose for the camera. Using the reference pose, the 3D feature points are projected on the 2D image plane to produce reference pixel coordinates.[14] Mirage takes advantage of the incremental relations[15] between 3D pose of the camera and the pixel coordinates of the feature points. This idea allows us to estimate the relation between the actual pose and the reference pose by minimizing the error between actual and reference pixel coordinates. Mirage uses the pixel errors (incremental pixel coordinates) to calculate how far the camera is with respect to its reference pose (incremental 3D pose). Then, this relative pose (or pose error) is added to the reference pose to calculate the actual pose. Computing the pose error is beneficial in robotic systems, since only the relative pose (or pose error) would be sufficient to follow a desired trajectory or complete a given motion. Mirage involves analytical derivations, which finally yield a linear equation system to estimate the pose. This low complexity allows us to achieve a linear time solution. Moreover, Mirage is designed to have capability of supporting one or multiple cameras without significant overhead while improving its reliability as the field-of-view is extended.

After detecting 2D image features and removing the outliers using methods such as SIFT and RANSAC,[16] a 3D reference pose and its corresponding 2D pixels are given to the Mirage to calculate the actual camera pose. In our experiments, we observed that Mirage is robust to the noise on the image, has efficient time complexity, and estimates the pose with fewer number of features compared to other state-of-the-art methods. The major contributions and features of the Mirage may be summarized as follows: Mirage

- utilizes a reference camera pose and estimates the relative pose of the camera with respect to its reference pose,
- has flexible structure supporting one or multiple cameras,
- is an analytical PnP method with $O(n)$ time complexity for multi-camera systems,
- has resiliency to noise by yielding the lowest translational and rotational error among the compared methods,
- is more reliable than other methods for few number of feature points.

Remainder of the paper is organized as follows. In Section 2, we present a detailed review on the recent studies in the literature and give comparison based on significant aspects. In Section 3, we describe *Mirage* for a single camera system and then generalize the idea to a multi-camera system. In Sections 4 and 5, we show the experimental results and comparisons by using both simulated and real data, respectively. Section 6 provides a discussion about preconditions and potential limitations of the Mirage. Finally, Section 7 concludes our paper.

2. Related Work

There has been significant work on the PnP problem in the last three decades due to its practical significance in many applications. A good number of studies, which provide survey of pose estimation,[17–19] exist in the literature. We can broadly classify the PnP methods as numerical and analytical solutions. Numerical methods are iterative methods that minimize a specific error

Table I. Comparison of well-known and recent studies regarding some important aspects.

Paper name	Complexity	Depth sensor	#cameras
Dementhon (1995) - [POSIT]	$O(mn)$ + Iterative	No	1
Lu (2000) - [LHM]	Iterative	No	1
Ansar (2003)	O(n)	No	1
Chang (2004)	Iterative	No	Multiple
Stewenius (2006)	Iterative[a]	No	2[b]
Lepetit (2009) - [EPnP]	$O(n)$	No	1
Chen (2009)	Iterative	No	Multiple[b]
Noell (2010)	Iterative[a]	No	1
Hesch (2011) - [DLS]	$O(n)$	No	1
Choi (2012)	$O(n^2)$ + Iterative	Yes	1
Li (2012) - [RPnP]	$O(n)$	No	1
Jaramillo (2013)	Iterative	No	1
Lee (2013)	$O(n^2)$ + Iterative	No	Multiple
Dryanovski(2013)	Iterative	Yes	1
Kneip (2013) - [gPnP]	$O(n)$	No	Multiple
Zheng (2013)	$O(n)$	No	1
Ferraz (2014)	$O(n)$	No	1
Fabian (2014)	$O(n)$	Yes	1
Vandenhouten (2015)	Iterative	No	1
Mirage	$O(n)$	No	Multiple

[a]Partially iterative approach is used in small part of the solution.
[b]All cameras are expected to capture the same set of points.

(geometric or algebraic[4]) by optimizing an objective function. These methods are robust to noise and they can benefit from high number of matching points since they can iteratively search the solution space. However, they are susceptible to be trapped in local minima, which prevents them converging to the optimal solution.[5] For instance, the general case of the iterative closest point (ICP) algorithm often converges to a bad local minima in the search space.[20] Another example is the classical POSIT algorithm[21] that converged to a local minimum in some of our experiments. An exhaustive search can be initiated to overcome the local minima entrapment problem, but it would have high computational cost. On the other hand, analytical methods directly solve the parameters as unknowns of an equation system to reach the optimal solution. However, the time complexity of these methods may not be tractable for real-time systems. In addition, these methods are usually sensitive to noise, which causes an incorrect pose estimation. The noise in this context appears due to inaccurate pixel positions of the feature points. Another categorization can be made based on multi-camera support and determining the 3D positions of feature points. Next, we briefly look into related work from these perspectives and give a comparison of methods in Table I.

Time Complexity. The minimal case of PnP is P3P, where only three matching points are employed to solve unknowns in the equation system. Mathematically, this requires solving an 8-degree polynomial that may yield up generally two but up to four solutions, which causes an ambiguity.[1] Thus, in later studies, additional feature points (four points in ref. [22], five points in ref. [23]) have been included into the solutions to clarify this ambiguity. Theoretically, using a small number of points is sufficient to calculate the pose, however, recent studies show that abundance of the points helps to avoid planarity and reduces the negative effect of the noise.[24] Many existing studies target to exploit abundance of the feature points.[2,25,26] The time complexity of these methods include $O(n^8)$ (method by Ansar *et al.*[6]), $O(n^5)$ (method by Quan *et al.*[27]), and $O(n^2)$ (methods by refs. [11,28]).

Finding the optimal solution in *real-time* is as important as reaching the optimal solution. The methods with time complexity of $O(n^b)$ where $b > 1$ are not suitable for real-time applications. The first well-known linear time solution is DLT (Direct Linear Transform),[29] where the transformation parameters are solved without involving any constraints. Even though it is relatively a fast algorithm, it is very sensitive to noise, therefore not applicable for real-time applications. To increase resiliency to noise, Lepetit *et al.* (2009)[7] proposed the EPnP method with $O(n)$ time complexity. EPnP calculates four reference points that are generated from n points. The method solves the pose parameters using those four reference points. Based on EPnP, subsequent linear time solutions were proposed such as Hesh *et al.*,[30] Li *et al.*,[31] Zheng *et al.*,[24] and Ferraz *et al.*[4] Although these methods increased the robustness of the system, our experiments show that there is still room for further improvements for high noise rates and low number of feature points.

Multi-camera Support. The multi-camera structure enlarges the field of view and provides more information about the environment such as depth. It is clearly beneficial to use multi-camera structure without sacrificing the processing speed or computational load. The solutions given in the previous section were not originally designed to support multi-camera systems. Although there are a few number of research studies employing multiple cameras in the literature, some of these studies[9] require the same points should be visible in all cameras. Such a limitation prevents to get benefit from enlarged field of view. While methods proposed by Chang *et al.*[12] and Lee *et al.*[11] do not have such a limitation, they have another drawback of being numerical solutions thus they may be trapped in local minima. Kneip *et al.*[8] proposed the gPnP method with $O(n)$ time complexity with multi-camera support. Although their method is more resilient to noise than the EPnP method, the translational error of the gPnP is roughly three times of the EPnP method with 100 feature points available under their experimental setting.

Determination of 3D Positions. The PnP methods assume the availability of the 3D positions of the feature points. The studies can be divided into two groups in that sense. The first group of studies uses a specific equipment (e.g., depth sensors) in their system to determine 3D positions of feature points on the object.[32] Dryanovski *et al.*[33] proposed a camera trajectory estimation algorithm using an RGB-D camera. After extracting feature points on the image and estimating 3D positions using their uncertainty model, their method tries to register 3D positions to a model built using previous frames. In another study, Choi *et al.*[28] propose an iterative method for object pose estimation using multiple point clouds of objects. In their evaluations, they pay more attention to accuracy of detection and recognition rather than pose estimation errors. In the second group, methods employ a complete or sparse 3D model of a reference object to determine the 3D coordinates of feature points.[34] In this study, we prefer using second approach (3D model based solution) since there is no requirement for an additional depth sensor. And, constructing the 3D model of any object is not a significant burden to the system, since it is done only once offline.

3. Mirage: Analytic Pose Estimation

Mirage analytically solves six pose parameters in $O(n)$ time for multi-camera systems. It assumes the availability of a basic 3D object model and utilizes a given reference pose to calculate the actual camera pose. We refer the real pose parameters as "actual pose" throughout the paper. And the "reference pose" can be determined arbitrarily in the world space independent from the actual pose. "Actual pixels" are extracted from real images, while "reference pixels" can be obtained by projecting the 3D feature points with respect to the reference camera pose. Our algorithm aims to minimize the 2D projection error between the actual and reference pixel coordinates and returns relative pose of the camera with respect to its reference pose. The relative pose can be directly fed into the system (e.g., visual servoing) or the actual pose can be calculated by adding the reference pose to the relative pose.

Mirage can be applied to camera systems with any number of cameras. In a multi-camera system, rather than calculating the pose of every camera, a single pose is calculated with respect to a central point (usually the carrier vehicle). Similarly, in this study, we calculate the pose of a *vehicle* on which, all cameras are mounted with known relative positions and orientations. As single-camera systems are common, we start explaining formulation of Mirage for single-camera systems in detail. Later, we show the generalization of our method to multi-camera systems.

In the following sections of the paper, we adopt the following notation. Scalars are denoted by non-bold italic letters. Vectors are noted by lowercase bold letters (e.g., $\mathbf{m}$, $\mathbf{n}$, $\mathbf{o}$). And matrices are represented by uppercase bold letters (e.g., $\mathbf{M}$, $\mathbf{K}$, $\mathbf{V}$).

3.1. Single-camera systems

Figure 1 illustrates a sample scenario showing world (I), vehicle (B), camera (C) spaces, and a *target object* having four 3D feature points (r^B) in the actual vehicle space. Similarly, reference vehicle space (Bd) and corresponding reference camera space (Cd) are given along with the 3D feature points

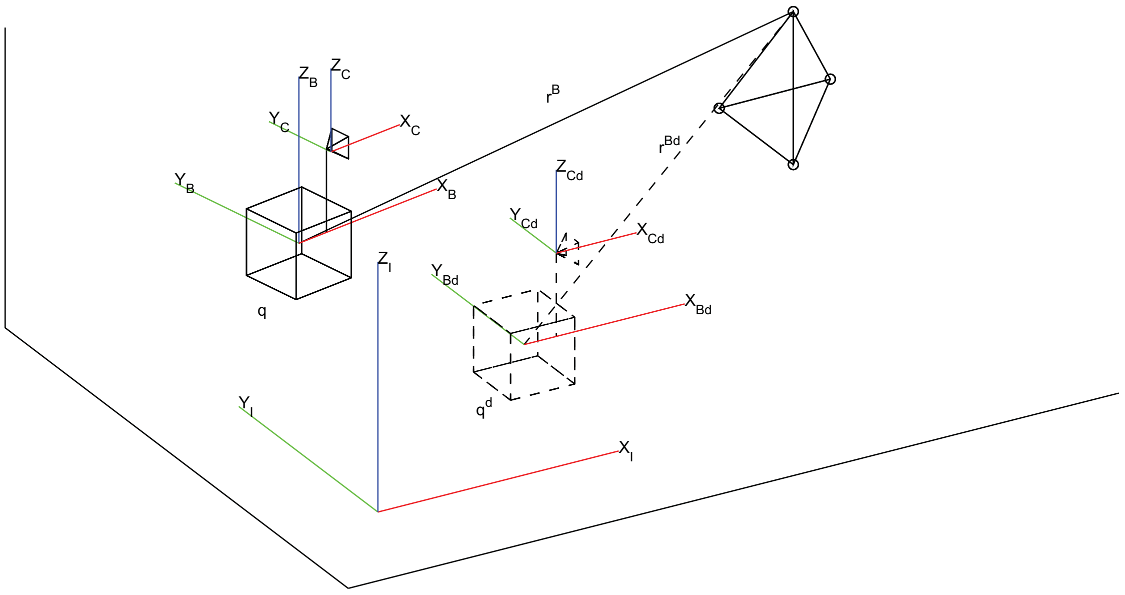

Fig. 1. Sample scene for Mirage. $\mathbf{q}$ is the actual pose of the vehicle and $\mathbf{r}^B$ is 3D position of a target point with respect to the actual pose. Similarly, $\mathbf{q}^d$ is the reference pose of the vehicle and $\mathbf{r}^{Bd}$ is 3D position of the same target point with respect to the reference pose.

in the reference vehicle space (r^{Bd}). The actual pose is denoted by $\mathbf{q} = [x, y, z, \theta, \phi, \psi]$ and shown by solid lines in the figure. Similarly, the reference pose is denoted by $\mathbf{q^d} = [x^d, y^d, z^d, \theta^d, \phi^d, \psi^d]$ and shown by dashed lines. Mirage calculates the relative pose by determining a 4×4 transformation matrix $\tilde{\mathbf{T}}^B_{Bd}$, which has rotational ($\mathbf{R}^B_{Bd}$) and translational ($\mathbf{t}^B_{Bd}$) parts such that $\tilde{\mathbf{T}}^B_{Bd} = [\mathbf{R}^B_{Bd}|\mathbf{t}^B_{Bd}]$, where $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in R^3$.

$\tilde{\mathbf{T}}^B_{Bd}$ transforms the vehicle's local space at its actual pose (X_B, Y_B, Z_B) to its local space at its reference pose X_{Bd}, Y_{Bd}, Z_{Bd}. This relation can be represented such that

$$\mathbf{T}^B_I = \tilde{\mathbf{T}}^B_{Bd}\mathbf{T}^{Bd}_I, \tag{1}$$

where $\mathbf{T}^B_I = [\mathbf{R}^B_I|\mathbf{t}^B_I]$ is a transformation matrix that transform a given coordinate in world space into the actual vehicle space. Actual pose of the vehicle, $\mathbf{q}$, can be determined by manipulating rotation and translational parts of this matrix. Similarly, $\mathbf{T}^{Bd}_I = [\mathbf{R}^{Bd}_I|\mathbf{t}^{Bd}_I]$ is the transformation matrix to transform a given coordinate in world space into the reference vehicle space and it corresponds to $\mathbf{q^d}$, the reference pose of the vehicle. Equation (1) states that the actual pose is conveniently determined if $\mathbf{T}^{Bd}_I$ is known and $\tilde{\mathbf{T}}^B_{Bd}$ is calculated.

Mirage solves $\tilde{\mathbf{T}}^B_{Bd}$ by minimizing image projection errors. Thus, by utilizing a given reference pose, we can calculate the actual pose of the vehicle in world space. Our calculations are based on the 2D pixel differences between reference and actual points. Assume that $\mathbf{r}^B = [r^B_1\ r^B_2\ r^B_3\ 1]^T$ is the 3D position of the object feature point with respect to the vehicle at its actual position. Similarly, $\mathbf{r}^{Bd} = [r^{Bd}_1\ r^{Bd}_2\ r^{Bd}_3\ 1]^T$ indicates the relative position of the same point when the vehicle is at its reference position. $\mathbf{r}^B$ is equal to $\mathbf{r}^{Bd}$ when the vehicle is at its reference pose. Otherwise a 3D translation and rotation is necessary for the vehicle to transform the reference points to the actual points. This transformation is done by $\tilde{\mathbf{T}}^B_{Bd}$ transformation matrix as

$$\mathbf{r}^B = \tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}. \tag{2}$$

Let $\mathbf{p}^C$ be a 3D point in camera space and $\mathbf{M}^C$ be a 4×4 transformation matrix that transforms a point from the vehicle space to the camera space,

$$\mathbf{p}^C = \mathbf{M}^C \mathbf{r}^B, \tag{3}$$

where $\mathbf{M}^C = [\mathbf{m}_1\ \mathbf{m}_2\ \mathbf{m}_3\ 1]^T$ and $\mathbf{m}_i$'s are the rows of the matrix. $\mathbf{M}^C$ comprises two different transformation matrices $\mathbf{K}^C$ and $\mathbf{T}_B^C$:

$$\mathbf{M}^C = \mathbf{K}^C \mathbf{T}_B^C, \tag{4}$$

where $\mathbf{T}_B^C$ is a 4×4 transformation matrix including extrinsic parameters of the camera that transforms a point from the vehicle space to the camera space. $\mathbf{K}^C$ is also a 4×4 matrix of intrinsic camera parameters (focal length, principle points, and distortion parameters), which are available *a priori* via camera calibration. $\mathbf{K}^C$ projects the point in camera space into image plane coordinates. Similar to Eq. (3), reference points can be transformed from vehicle to camera space as follows:

$$\mathbf{p}^{Cd} = \mathbf{M}^C \mathbf{r}^{Bd}. \tag{5}$$

Using the reference and actual camera points, the error can be measured. Normally, a simple subtraction operation $\mathbf{p}^C - \mathbf{p}^{Cd}$ would be performed, however, subtraction is not closed under 3D camera space. Instead, the error can be derived as

$$\tilde{\mathbf{p}}^C = \begin{bmatrix} \tilde{p}_1^C \\ \tilde{p}_2^C \\ \tilde{p}_3^C \end{bmatrix} = \begin{bmatrix} p_3^{Cd} p_1^C - p_1^{Cd} p_3^C \\ p_3^{Cd} p_2^C - p_2^{Cd} p_3^C \\ p_3^{Cd} p_3^C \end{bmatrix}, \tag{6}$$

where $\tilde{\mathbf{p}}^C$ represents the error between actual and reference values of the target points in the camera space. Expression in Eq. (6) is substituted with corresponding expressions in Eqs. (3) and (5) as follows:

$$\tilde{\mathbf{p}}^C = \begin{bmatrix} \tilde{p}_1^C \\ \tilde{p}_2^C \\ \tilde{p}_3^C \end{bmatrix} = \begin{bmatrix} \mathbf{m}_3\mathbf{r}^{Bd}\mathbf{m}_1\mathbf{r}^B - \mathbf{m}_1\mathbf{r}^{Bd}\mathbf{m}_3\mathbf{r}^B \\ \mathbf{m}_3\mathbf{r}^{Bd}\mathbf{m}_2\mathbf{r}^B - \mathbf{m}_2\mathbf{r}^{Bd}\mathbf{m}_3\mathbf{r}^B \\ \mathbf{m}_3\mathbf{r}^{Bd}\mathbf{m}_3\mathbf{r}^B \end{bmatrix}. \tag{7}$$

Then if we move the common factor $\mathbf{r}^B$ out of the parenthesis, Eq. (8) is derived from Eq. (7):

$$\tilde{\mathbf{p}}^C = \begin{bmatrix} \tilde{p}_1^C \\ \tilde{p}_2^C \\ \tilde{p}_3^C \end{bmatrix} = \begin{bmatrix} \mathbf{m}_3\mathbf{r}^{Bd}\mathbf{m}_1 - \mathbf{m}_1\mathbf{r}^{Bd}\mathbf{m}_3 \\ \mathbf{m}_3\mathbf{r}^{Bd}\mathbf{m}_2 - \mathbf{m}_2\mathbf{r}^{Bd}\mathbf{m}_3 \\ \mathbf{m}_3\mathbf{r}^{Bd}\mathbf{m}_3 \end{bmatrix} \mathbf{r}^B. \tag{8}$$

For the sake of simplicity, the rows of Eq. (8) are called $\mathbf{n}_i$. Using this notation, following form of the equation is obtained:

$$\tilde{\mathbf{p}}^C = \begin{bmatrix} \mathbf{n}_1 \\ \mathbf{n}_2 \\ \mathbf{n}_3 \end{bmatrix} \mathbf{r}^B = \mathbf{N}^C \mathbf{r}^B. \tag{9}$$

After substitution of $\mathbf{r}^B$ with $\tilde{\mathbf{T}}_{Bd}^B \mathbf{r}^{Bd}$ in Eq. (2), Eq. (10) can be derived as follows:

$$\tilde{\mathbf{p}}^C = \mathbf{N}^C \tilde{\mathbf{T}}_{Bd}^B \mathbf{r}^{Bd}. \tag{10}$$

Equation (10) implies that $\tilde{\mathbf{T}}_{Bd}^B$ can analytically be calculated if $\tilde{\mathbf{p}}^C$ is known, since all other parameters in the equation are known. $\tilde{\mathbf{p}}^C$ can be calculated directly by comparing the reference and actual pixel coordinates. Because 2D image errors can be related to $\tilde{\mathbf{p}}^C$. The relation between 2D and 3D image errors can be shown as

$$\mathbf{e}^C = \begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} \tilde{p}_1^C / \tilde{p}_3^C \\ \tilde{p}_2^C / \tilde{p}_3^C \end{bmatrix}, \tag{11}$$

where $\mathbf{e}$ represents the error between actual and reference 2D pixel coordinates of the points. In this case, $\mathbf{e}$ is a known vector because both the reference and actual images are known. If we separate the errors for both x and y dimensions on the image, we obtain

$$\tilde{p}_3^C e_x - \tilde{p}_1^C = 0, \tag{12}$$

$$\tilde{p}_3^C e_y - \tilde{p}_2^C = 0. \tag{13}$$

If we substitute $\tilde{p}_i^C$ with the corresponding value in Eq. (10), the new set of error Eqs. (A1) and (A2) are obtained as follows:

$$\mathbf{n}_3 \tilde{\mathbf{T}}_{Bd}^B \mathbf{r}^{Bd} e_x - \mathbf{n}_1 \tilde{\mathbf{T}}_{Bd}^B \mathbf{r}^{Bd} = 0, \tag{14}$$

$$\mathbf{n}_3 \tilde{\mathbf{T}}_{Bd}^B \mathbf{r}^{Bd} e_y - \mathbf{n}_2 \tilde{\mathbf{T}}_{Bd}^B \mathbf{r}^{Bd} = 0. \tag{15}$$

It is possible to reach Eq. (A3) after a series of derivations, which are presented in Appendix A:

$$\bar{\mathbf{e}}^C = \mathbf{V}^C \tilde{\mathbf{t}}_{Bd}^B. \tag{16}$$

In Eq. (A3), $\bar{\mathbf{e}}^C$ refers to pixel errors between actual and desired image pixels. $\tilde{\mathbf{t}}_{Bd}^B = [\tilde{t}_{11}\ \tilde{t}_{12}\ \tilde{t}_{13}\ \ldots\ \tilde{t}_{33}\ \tilde{t}_{34}]^T$ is a 12×1 vector that is actually the reshaped form of $\tilde{\mathbf{T}}_{Bd}^B$ matrix. And finally, $\mathbf{V}^C$ is 2×12 matrix that satisfies the corresponding equations. Equation (A3) is a linear equation system. Twelve unknowns in $\tilde{\mathbf{t}}_{Bd}^B$ can be solved if there are at least 12 independent equations in the system. In a single camera system, one target point generates two equations since one equation exists per image dimension x and y as in Eq. (11). Therefore, in theory, a minimum of six distinctive target points are necessary in a single camera system to satisfy all 12 equations. However, due to calibration error and dimensional inaccuracies, 12 equations do not yield a solvable system using single camera. Adding extra equations ($\mathbf{t}_i \mathbf{t}_j = 0$, for $1 \le i, j \le 3$, $i \ne j$) or camera(s) is necessary to compensate for these errors. Nevertheless, adding such equations makes the system non-linear for single camera. Finally, n target points generate $2n$ different equations where $2n > 12$. Equation (A3) is stacked n times to obtain

$$\begin{bmatrix} \mathbf{V}_1^C \\ \mathbf{V}_2^C \\ \vdots \\ \mathbf{V}_n^C \end{bmatrix} \tilde{\mathbf{t}}_{Bd}^B = \begin{bmatrix} \bar{\mathbf{e}}_1^C \\ \bar{\mathbf{e}}_2^C \\ \vdots \\ \bar{\mathbf{e}}_n^C \end{bmatrix}. \tag{17}$$

In a real application, it is most likely to have a large number of matching points, which may also have some amount of noise. For such cases, direct solutions are not applicable since the number of rows of $\mathbf{V}^C$ matrix will be much larger than 12. To overcome this limitation, a "least squares" optimization method is beneficial. Let us rewrite Eq. (17) such that $\mathbf{Q}$ be a $2n \times 12$ coefficient matrix on the left-hand side and $\mathbf{W}$ be $2n \times 1$ constant vector in the right-hand side:

$$\mathbf{Q} \tilde{\mathbf{t}}_{Bd}^B = \mathbf{W}, \tag{18}$$

where symbols **Q** and **W** refer to corresponding matrices in Eq. (17). Considering large number of points, **Q** will not be a square matrix, thus Eq. (18) may be solved using the least squares method. The error,

$$E = \frac{1}{2}(\mathbf{Q}\tilde{\mathbf{t}}^B_{Bd} - \mathbf{W})^T(\mathbf{Q}\tilde{\mathbf{t}}^B_{Bd} - \mathbf{W}), \tag{19}$$

is minimized using the formula in Eq. (20), which also solves the unknowns in $\tilde{\mathbf{t}}^B_{Bd}$ vector:

$$\tilde{\mathbf{t}}^B_{Bd} = (\mathbf{Q}^T\mathbf{Q})^{-1}(\mathbf{Q}^T\mathbf{W}). \tag{20}$$

Note that $\tilde{\mathbf{t}}^B_{Bd}$ includes 12 elements of the transformation matrix $\tilde{\mathbf{T}}^B_{Bd}$. If the purpose of the application was to estimate the relative pose of the vehicle, then using directly $\tilde{\mathbf{T}}^B_{Bd}$ would be sufficient, but we seek the actual pose of the vehicle. Thus, additional operation is necessary such that

$$\mathbf{T}^B_I = \tilde{\mathbf{T}}^B_{Bd}\mathbf{T}^{Bd}_I \tag{21}$$

which was described in Eq. (1) earlier. The multiplication in Eq. (21) returns the actual pose parameters of the vehicle as a transformation matrix:

$$\mathbf{T}^B_I = \left[\begin{array}{ccc|c} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]. \tag{22}$$

At this stage, a conversion is necessary to calculate the actual pose $\mathbf{q} = [x, y, z, \theta, \phi, \psi]$ using 12 elements of $\mathbf{T}^B_I$ matrix. The values t_{14}, t_{24}, and t_{34} are conveniently assigned to translation parameters x, y, and z, respectively. For the rotational parameters, however, the following conversions are necessary to compute the angles. In this conversion, quadrant-dependent function *atan2* is used instead of a simple *arctangent* function. The sign of $\cos(\theta)$ determines the quadrant of ϕ and ψ, so it must not be dropped from the denominators.

$$\theta = \operatorname{atan2}(-t_{31}, \sqrt{{t_{11}}^2 + {t_{21}}^2}), \tag{23}$$

$$\phi = \operatorname{atan2}(\frac{t_{32}}{\cos\theta}, \frac{t_{33}}{\cos\theta}), \tag{24}$$

$$\psi = \operatorname{atan2}(\frac{t_{21}}{\cos\theta}, \frac{t_{11}}{\cos\theta}). \tag{25}$$

3.2. Generalization to multi-camera systems

Our system treats the multi-camera system as “one”.[8] It can easily be adapted to multi-camera systems without the need of major modifications and significant overhead (e.g., bundle adjustment[35]). There are no major restrictions on the number and specifications of cameras. The cameras do not have to be identical, and they can be installed on the vehicle at known arbitrary poses. All cameras can capture different target points and non-overlapping setup (as in ref. [8]) is not required for Mirage.

Moreover, our derivations yield another significant advantage. Increasing the number of cameras decreases the number of target points required for the solution. Based on the fact that one camera can provide two equations from a target point, theoretically, the inequality $mn \geq 6$ must be satisfied, where m is the number of cameras and n is the number of target points. If n target points are required for a single camera system, multi-camera system with m cameras may work with $\lceil n/m \rceil$ target points. However, in practice, at least four non-planar points are necessary to localize the vehicle and perform reliable pose calculation. Equation (17) formulates this logic for a single camera system. If we generalize this formula for m number of cameras, we have

$$\begin{bmatrix} \mathbf{V}_1^{C_1} \\ \mathbf{V}_1^{C_2} \\ \vdots \\ \mathbf{V}_1^{C_m} \\ \mathbf{V}_2^{C_1} \\ \mathbf{V}_2^{C_2} \\ \vdots \\ \mathbf{V}_2^{C_m} \\ \vdots \\ \mathbf{V}_n^{C_1} \\ \mathbf{V}_n^{C_2} \\ \vdots \\ \mathbf{V}_n^{C_m} \end{bmatrix} \tilde{\mathbf{t}}_{Bd}^{B} = \begin{bmatrix} \bar{\mathbf{e}}_1^{C_1} \\ \bar{\mathbf{e}}_1^{C_2} \\ \vdots \\ \bar{\mathbf{e}}_1^{C_m} \\ \bar{\mathbf{e}}_2^{C_1} \\ \bar{\mathbf{e}}_2^{C_2} \\ \vdots \\ \bar{\mathbf{e}}_2^{C_m} \\ \vdots \\ \bar{\mathbf{e}}_n^{C_1} \\ \bar{\mathbf{e}}_n^{C_2} \\ \vdots \\ \bar{\mathbf{e}}_n^{C_m} \end{bmatrix}, \tag{26}$$

where $\mathbf{V}_j^{C_i}$ represents the $\mathbf{V}$ matrix for ith camera and jth target point. Similarly, symbol $\bar{\mathbf{e}}_j^{C_i}$ represents $\bar{\mathbf{e}}$ pixel errors for ith camera and jth target point. Using the equation system in Eq. (26) instead of Eq. (17), the pose parameters can be conveniently solved for a multi-camera system having m cameras.

4. Performance of Mirage on Synthetic Data

In this section, we investigated the pose calculation accuracy and processing time of Mirage for variety of conditions using synthetic data. We compared Mirage with available state-of-the-art techniques, Efficient PnP (EPnP), Efficient PnP with Gauss Newton (EPnP-G), Classic POSIT (Posit-C), Modern POSIT [(1)] (Posit-M), DLT, Lu–Hager–Mjolsness (LHM), Robust PnP (RPnP), Direct Least Squares (DLS). Each method is evaluated with respect to processing time, robustness to noise, and number of feature points via three experiments: noise rate vs. pose error, the number of points vs. pose error, and the number of points vs. processing time.

In this simulated environment, the actual camera was placed in a known position of the world space and n number of 3D target points were randomly generated (in the interval of $[-5,5]$ in X, Y, and Z dimensions). Then, the target points were projected on the 2D image plane to calculate pixel locations. We also generated pixel noise randomly using normal distribution model. Noise values were added to the pixel coordinates (projections of 3D points), since we wanted to simulate a real problem (imprecise feature detection) in the feature extraction stage. 3D points and their projections were provided to all methods. Only in Mirage, a reference pose was also defined to be used in calculations. For a given dataset, every method estimated a camera pose. Since the points were randomly generated, we ran every simulation 20 times and get the average error for the final evaluation. In all experiments, we used the intrinsic parameters of a real camera (Logitech HD Webcam C525) and image size of 640×480 pixels. Our code was implemented in MATLAB 2013a.

4.1. Noise rate vs. pose error

In real applications, it is a common problem to have fair amount of noise on feature extraction stage. A desirable pose estimation method should handle this problem without sacrificing the accuracy

[(1)]Projection center is adjusted to object origin.

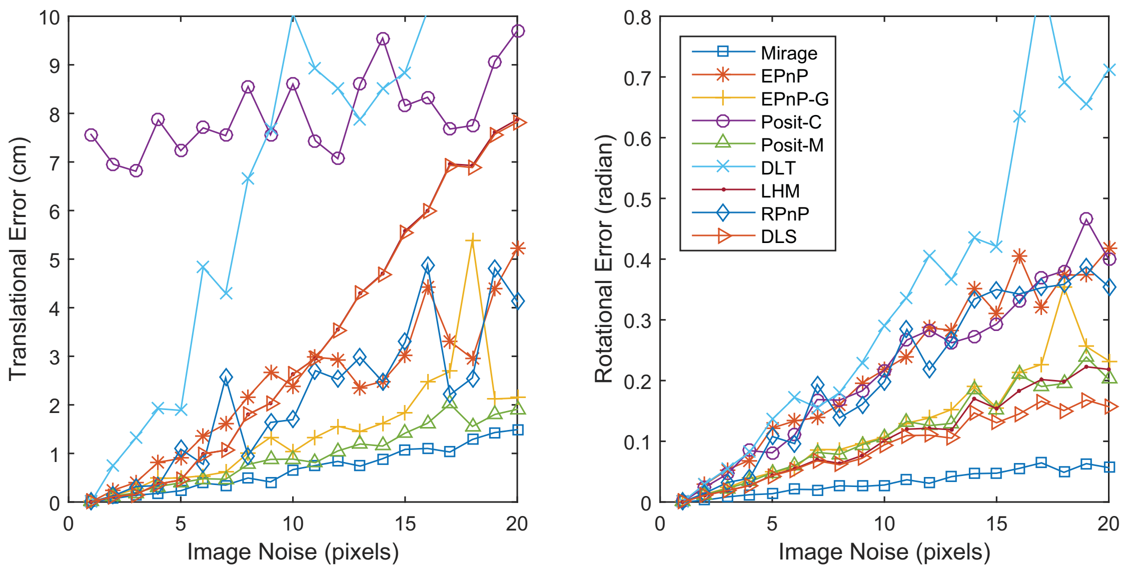

Fig. 2. Translational and rotational pose error with respect to the pixel noise for $n = 50$ feature points and $\sigma = 0$ to 20 noise rate.

significantly. The first experiment targets to simulate this condition, where 2D pixel coordinates cannot be perfectly detected. The results show the noise resilience analysis of each method. In the experiments, the number of feature points n was set to 50, and 20 different gaussian noise rate ($\sigma = 0$ to 20) were added to the pixel coordinates of each point. The noise was generated randomly for each feature point and varies. Figure 2 shows translational (cm) and rotational (radian) pose errors with respect to the image pixel noise.

According to the results, DLT and Posit-C return very high error rates, which imply they are very sensitive to the noise. Particularly, DLT is very unstable with respect to the noise. LHM and DLS give low rotational errors but their translational error rapidly increases as the noise increases. EPnP and EPnP-G generate acceptable results, yet a few spikes can be seen around $\sigma = 12$, 16, and 20, which make them unstable on these cases. RPnP yields relatively better results in translation but is still not as good as the Posit-M and Mirage in rotation. Interestingly, an iterative approach Posit-M generates low error rate and more stable results than other compared methods in both translational and rotational parameters. However, Posit-M has another limitation that is not shown in the figure. During our experiments, on some specific cases (special combinations of random points), Posit methods have fallen to local minima and never converged to a solution. This is a critical limitation that makes most of the iterative methods impractical. Our method, on the other hand, yields stable results and the lowest translational and rotational errors.

4.2. *Number of points vs. pose error*

Number of feature points is another critical factor in pose estimation problem. It is desirable to estimate the pose with less number of points. In the second experiment, we analyze the effect of number of feature points to the pose error. We evaluate the number of matching points, n, between 10 and 200 for a fixed amount of gaussian noise, where $\sigma = 10$.

As expected, all methods generate lower error results as the number of points increases. Figure 3 shows that DLT gives unstable results and the error rate merely decreases by increasing the number of points. Posit-C also returns high error rate, which is more than eight times of Mirage. Increasing the number of points does not affect the results of LHM and DLS in translational error; however, DLS yields better improvement than LHM in rotational error. EPnP and RPnP generate relatively good results, yet they are not stable enough compared to the remaining methods. EPnP-G and Posit-M yield good results in translational error, but not as good as Mirage in rotational error. Mirage generates significantly good and stable results in both translational and rotational pose error. Our experiments show that using 30–40 number of points would be sufficient to calculate the pose for gaussian noise with $\sigma = 10$ noise rate, which is a realistic assumption for real applications.

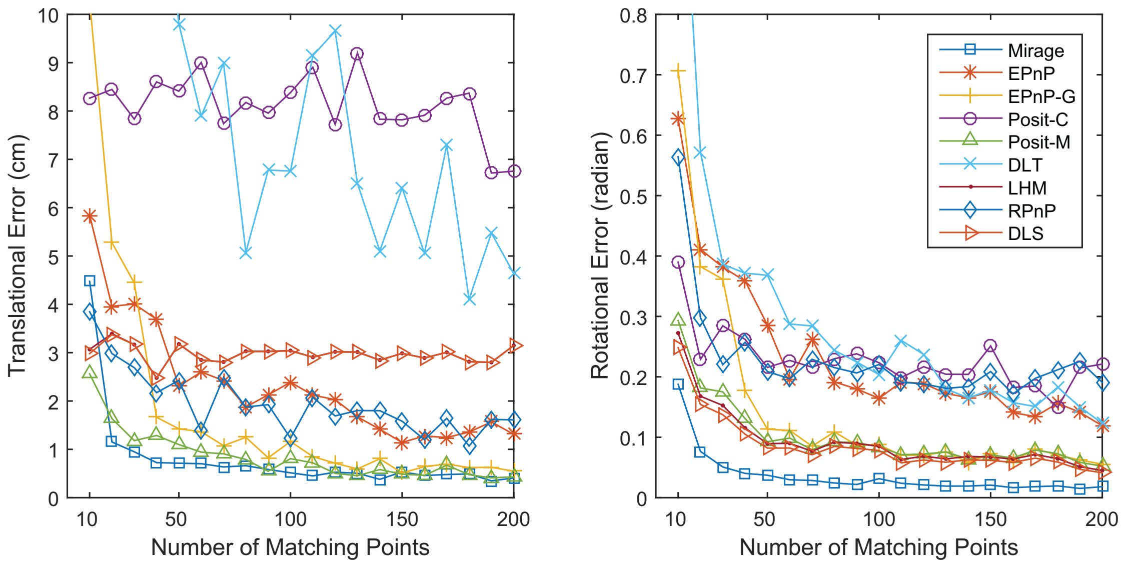

Fig. 3. Translational and rotational error with respect to the number of points $n = 10$ to 200 and $\sigma = 10$ noise rate.

4.3. Number of points vs. processing time

Previous experiment results show that increasing number of points decreases the pose error. Although there is a clear advantage of using more points, the running time of the algorithm is negatively affected from large data size. In this experiment, our goal is to analyze the processing times of the methods with respect to number of matching points. So, we measured the run-time of each method for increasing number of points from $n = 10$ to 200, under fixed amount of pixel noise, where $sigma = 10$.

According to the results in Fig. 4, LHM requires very high computational time (0.07 s for 200 points), which is larger than the plot range. DLS gives slightly better results than LHM but it is not as efficient as the others. Remaining methods run in reasonable time. EPnP, EPnP-G, and Mirage show similar performance and RPnP is slightly better. It is hard to benefit from fast convergence of Posit-C, Posit-M, and DLT methods to a solution, since Posit methods may fall into local minima with the probability of high error and DLT yields high pose errors. The results clearly show that Mirage is a linear solution since the processing time is linearly increasing. In addition, it runs faster than EPnP and EPnP-G until 100 points that is larger than the sufficient number of points shown in previous results. In a typical real application using the same image size, it is expected to extract around 100–150 key features on the average. In such a system, Mirage can calculate the pose around 4 ms, which is feasible for real-time requirements.

5. Performance of Mirage on Real Data

We also made experiments in a real environment, where we employed a multi-camera system (having two Logitech HD Webcam C525 cameras) as shown in Fig. 5(a). We used 640×480 pixels resolution images. In our real experiments, we constructed a grid surface (formed with markers on the table shown in Fig. 5(b)) which allowed us to create a reference 3D space (world space) that has the origin (0,0,0) at the bottom right of the table. Before each experiment, we placed the camera and the target to known (actual) positions as the ground truth. Later, the camera pose is estimated using the 2D images of the target and corresponding 3D point coordinates.

In this section, we made two sets of experiments using two different target object models. In the first experiment, we used a target object that has four feature points as shown in Fig. 6(a). Some of the available methods require more than four points in their algorithms. Therefore, in the second set of experiments we used a target object that has eight feature points as shown in Fig. 6(b) to produce reliable results for all techniques.

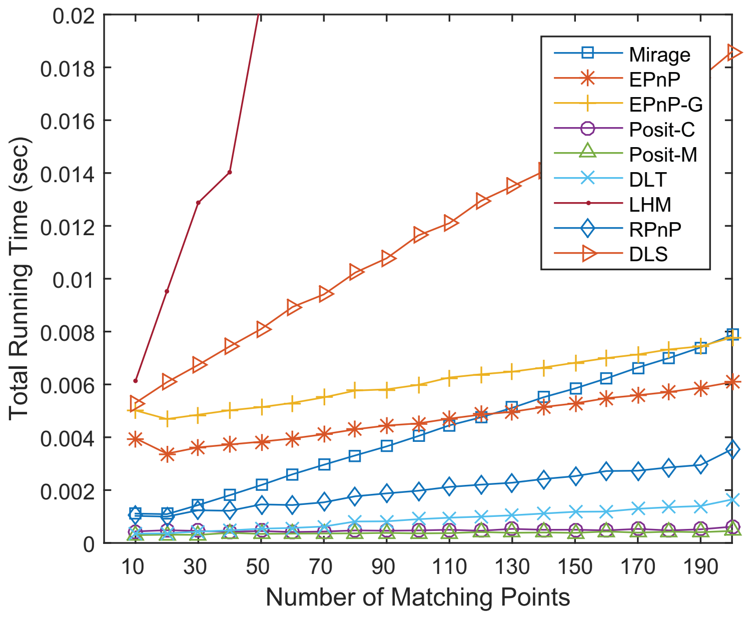

Fig. 4. Total processing time with respect to number of points $n = 10$ to 200 and $\sigma = 10$ noise rate.

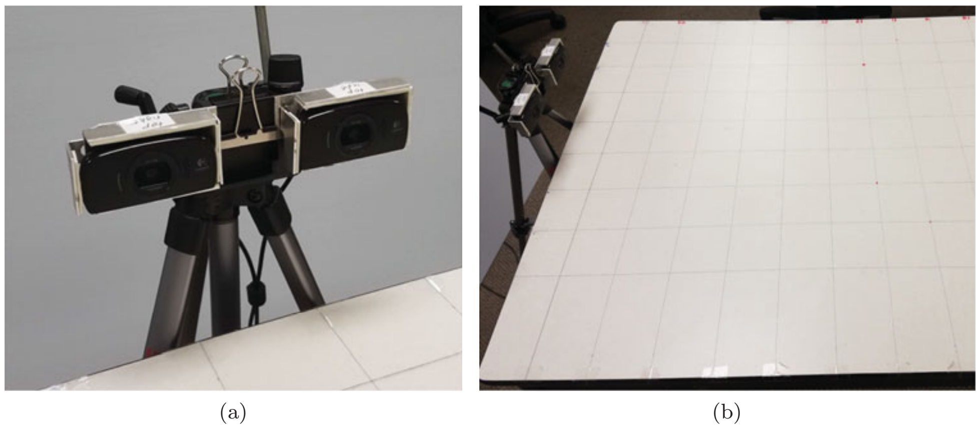

Fig. 5. (a) Cameras, (b) Grid surface.

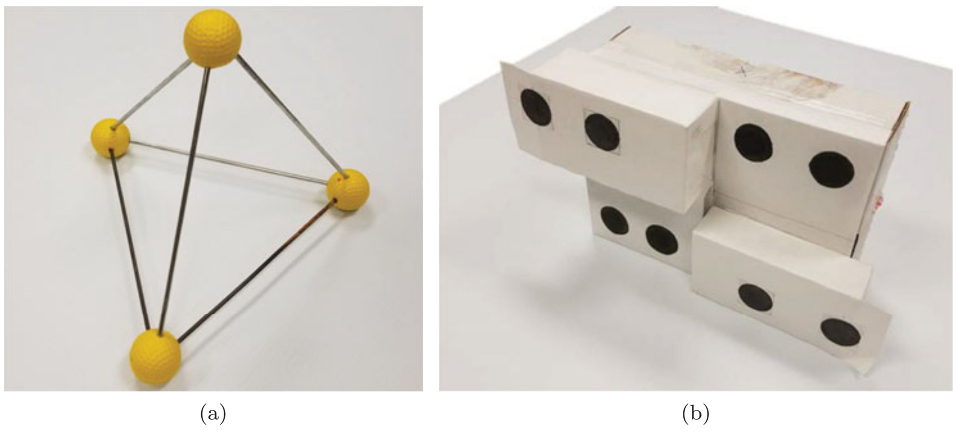

Fig. 6. (a) 4 points target object, (b) 8 points target object.

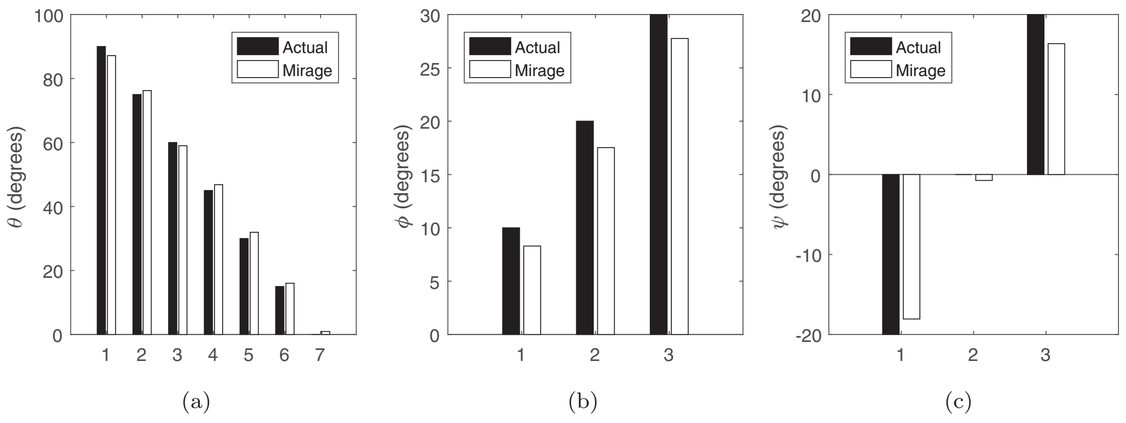

Fig. 7. Rotational pose calculation results. (a) θ (rotation around x axis), (b) ϕ (rotation around y axis), and (c) ψ (rotation around z axis).

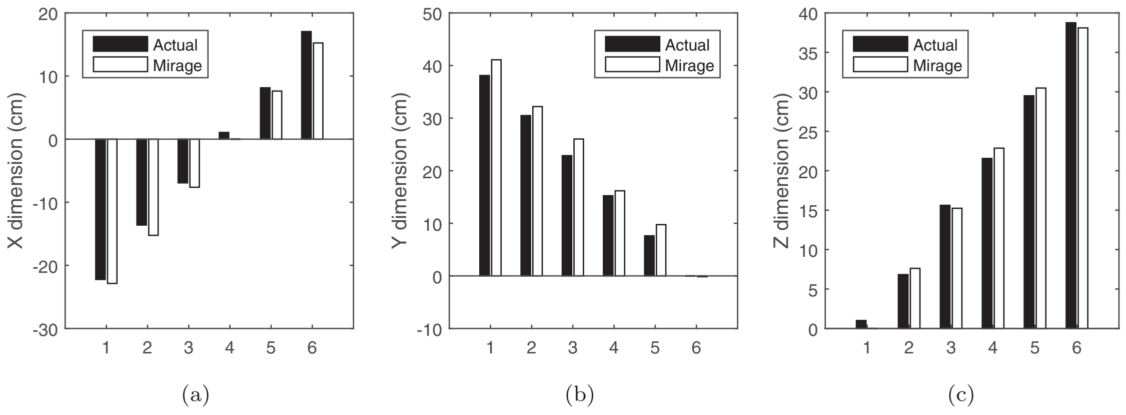

Fig. 8. Translational pose calculation results. (a) x dimension, (b) y dimension, (c) z dimension.

5.1. Experiments with 4-point target

Mirage pose estimation allows 4 points target object using a two-camera system. Sine other methods did not generate good results with 4 points, we performed these experiments using only Mirage pose estimation. In this experiment, we placed the target object to several known positions and estimated six pose parameters of the camera. We repeated the experiments for each dimension individually. The calculations of each dimension are compared with the actual values in our 3D world space.

Figure 7 shows the Mirage estimations and the actual values of rotational pose parameters (θ, ϕ, ψ) in degrees. And Fig. 8 shows the same comparison results for the translational parameters x, y, and z in cm. Results show that Mirage can calculate the pose of the camera with minor (around 5–7%) rotational and translational errors. This is an acceptable error rate when considering the minor inaccuracies in the calibration stage and other minor imperfections in ground truth measurement.

5.2. Experiments with 8-point target

In this set of experiments, we used different target model that has eight feature points and we tested the pose estimation performance of all techniques. We repeated the experiment for 10 different target positions in a systematic way that each position has different distances to the camera. For each case, we produced the estimation results of the methods and then measured the total translational and rotational errors.

The results are shown in Fig. 9. Posit-C and DLT produce acceptable rotational errors but they are very inconsistent in translation. They produce high amount of errors in translation as the distance between camera and the target is increasing. EPnP and EPnP-G produce good results in translational parameters. However, in rotational errors, there are a few unexpected spikes that make these two methods unstable. RPnP produce low translational error. But its rotational error is still high. Posit-M,

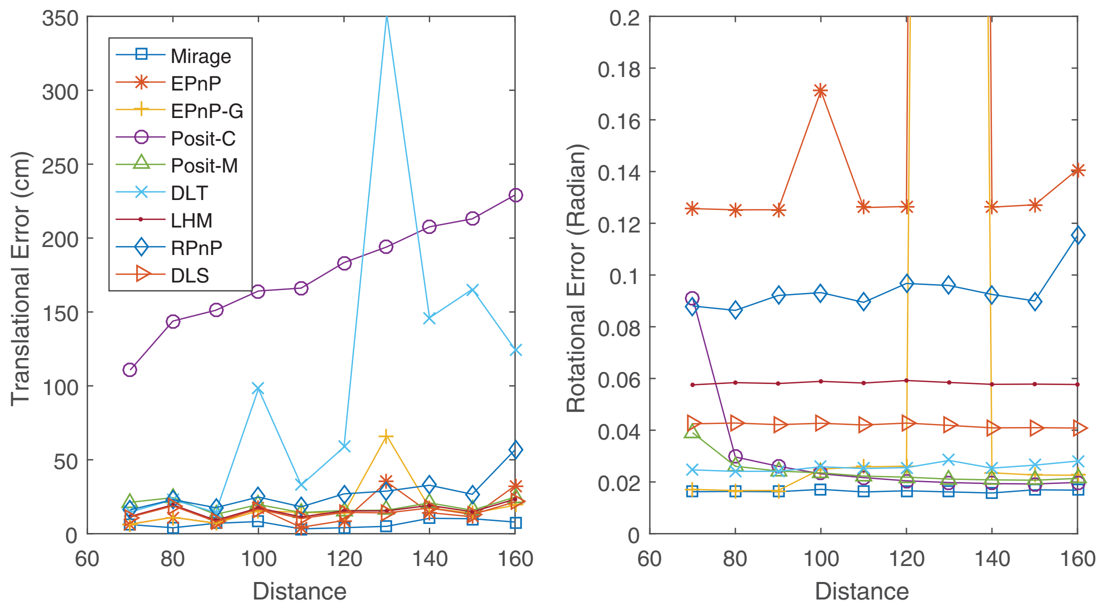

Fig. 9. Real experimental results using 8-point target model. Experiments were performed at different positions by increasing the distance between camera and the target.

DLS, and LHM produce acceptable results in both translational and rotational parameters; however, Mirage produces the lowest errors.

6. Discussion

Mirage is an efficient method that analytically calculates the pose parameters of a multi-camera system in linear time. The performance of the Mirage was tested for various conditions and promising results were obtained. However, there are several preconditions and limitations in the methodology that may require a precaution to prevent unexpected results. Most of these cases can be avoided with simple solutions, however, some of them require special attention.

Pose estimation methods assume the availability of 2D to 3D feature correspondence, which means feature extraction stage is not part of the pose estimation. This condition is also true for Mirage. So Mirage user must employ a feature extraction and mapping technique to generate initial feature set from images. Another precondition for Mirage is the availability of a colored (or textured) 3D model of the desired object. It is recommended to use high resolution 3D models, since the desired images are generated using 3D model. A 3D CAD model of the model is recommended to avoid scaling problems and produce reliable reference images.

Although, Mirage is very robust to the noise in 2D pixel coordinates, its calculations are sensitive to camera calibration errors. Inaccuracy in the calibration stage will introduce an error into the equations due to the incorrect intrinsic camera parameters. To minimize this error, an accurate calibration process is recommended.

Mirage does not support planar objects. 3D points on the object model must be non-planar to avoid under-determined equation system. However, today almost all 3D object models are non-planar, thus, this is not a significant limitation for real-world applications.

Finally, the current version of the Mirage does not have the linear time complexity for single camera system. Additional non-linear equations need to be included into the system to solve the unknowns. In multi-camera systems, there is no such a requirement. In the future, we plan to improve Mirage algorithm to achieve linear time complexity for single camera systems.

7. Conclusion

In this study, we present an evaluation of Mirage pose estimation various realistic conditions using simulated and real environments. Mirage is an analytic solution that linearly solves the pose parameters for multi-camera systems. It utilizes an additional reference camera pose in the calculations, rather than using only 2D to 3D point correspondence. Using the reference pose, Mirage estimates the camera pose by minimizing the 2D projection error between reference and actual pixel coordinates.

To evaluate the performance of Mirage, we conducted our experiments in various conditions. We also tested eight other well-known pose estimation techniques in the literature and compared the results of each method with Mirage. We obtained promising results in all experiments. Mirage outperforms other techniques by producing lowest translational and rotational errors even with low number of target points.

References

1. T. Petersen, A Comparison of 2D-3D Pose Estimation Methods *Master's Thesis* (Lautrupvang: Aalborg University-Institute for Media Technology Computer Vision and Graphics, 2008).
2. T. Nöll, A. Pagani and D. Stricker, "Real-Time Camera Pose Estimation using Correspondences with High Outlier Ratios," *VISAPP 2010: International Conference on Computer Vision Theory and Applications*, Angers, France (2010) pp. 381–386.
3. C. Jaramillo, I. Dryanovski, R. G. Valenti and J. Xiao, "6-DOF Pose Localization in 3D Point-Cloud Dense Maps Using a Monocular Camera," *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, Shenzhen, China, (2013) pp. 1747–1752.
4. L. Ferraz, X. Binefa and F. Moreno-Noguer, "Very Fast Solution to the PnP Problem with Algebraic Outlier Rejection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Columbus, Ohio, USA, (2014) pp. 501–508.
5. R. Tron, X. Zhou and K. Daniilidis, "A Survey on Rotation Optimization in Structure from Motion," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Las Vegas, Nevada, USA (2016) pp. 77–85.
6. A. Ansar and K. Daniilidis, "Linear pose estimation from points or lines," *Pattern IEEE Trans. Anal. Mach. Intell.* **25**(5), 578–589 (2003).
7. V. Lepetit, F. Moreno-Noguer and P. Fua, "EPnP: An Accurate O(n) Solution to the PnP Problem," *Int. J. Comput. Vis.* **81**(2), 155–166 (2009).
8. L. Kneip, P. Furgale and R. Siegwart, "Using Multi-Camera Systems in Robotics: Efficient Solutions to the nPnP Problem," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Karlsruhe, Germany, (2013) pp. 3770–3776.
9. C. Chen and D. Schonfeld, "Robust 3D pose estimation from multiple video cameras," *Proceedings of the 16th IEEE International Conference on Image Processing (ICIP)*, IEEE, Cairo Egypt (2009) pp. 541–544.
10. H. Stewenius, C. Engels and D. Nistér, "Recent developments on direct relative orientation," *ISPRS J. Photogramm. Remote Sens.* **60**(4), 284–294 (2006).
11. G. H. Lee, B. Li, M. Pollefeys and F. Fraundorfer, "Minimal Solutions for Pose Estimation of a Multi-Camera System," *Proceedings of the International Symposium on Robotics Research (ISRR)*, Singapore (2013) pp.1–16.
12. W. Y. Chang and C. S. Chen, "Pose estimation for multiple camera systems," *Proceedings of the 17th International Conference on Pattern Recognition, ICPR*, vol. 3, IEEE, Cambridge, UK (2004) pp. 262–265.
13. S. Dinc, F. Fahimi and R. Aygun, "Vision-based trajectory tracking for mobile robots using mirage pose estimation method," *IET Computer Vision* (Institution of Engineering and Technology) **10**(5), 450–458 (2016).
14. S. Dinc, F. Fahimi and R. Aygun, "Vision-Based Trajectory Tracking Approach for Mobile Platforms in 3D World using 2D Image Space," *Proceedings of the ASME International Mechanical Engineering Congress and Exposition, (IMECE)*, vol. 4 B, San Diego, CA, United States (2013).
15. S. Leonard, Learning Feed-Forward Control for Vision-Guided Robotics *PhD Thesis* (University of Alberta, Computing Science, Alberta, Canada, 2008).
16. O. Chum and J. Matas, "Matching with Prosac - Progressive Sample Consensus," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR*, vol. 1, San Diego, CA, USA, (2005) pp. 220–226.
17. B. Rosenhahn, Pose Estimation Revisited *PhD Thesis* (Inst. für Informatik und Praktische Mathematik, Kiel, Germany, 2003).
18. D. Grest, T. Petersen and V. Krüger, "A Comparison of Iterative 2D-3D Pose Estimation Methods for Real-Time Applications," **In:** *Image Analysis* (A. Salberg, J. Y. Hardeberg and R. Jenssen, eds.), (Springer, Oslo, Norway, 2009) pp. 706–715.
19. T. Nöll, A. Pagani and D. Stricker, "Markerless Camera Pose Estimation-an Overview," **In:** *OASIcs-OpenAccess Series in Informatics* (A. Middel, I. Scheler and H. Hagen, eds.) vol. 19 (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2011) pp. 45–54.
20. P. J. Besl and H. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.* **14**(2), 239–256 (1992).
21. D. F. Dementhon and L. S. Davis, "Model-based object pose in 25 lines of code," *Int. J. Comput. Vis.* **15**(1–2), 123–141 (1995).

22. Y. Guo, “A novel solution to the p4p problem for an uncalibrated camera,” *J. Math. Imaging Vis.* **45**(2), 186–198 (2013).
23. J. Tang, W.-S. Chen and J. Wang, “A novel linear algorithm for P5P problem,” *Appl. Math. Comput.* **205**(2), 628–634 (2008).
24. Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom and M. Okutomi, “Revisiting the pnp Problem: A Fast, General and Optimal Solution,” *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, IEEE, Sydney, Australia (2013) pp. 2344–2351.
25. C.-P. Lu, G. D. Hager and E. Mjolsness, “Fast and globally convergent pose estimation from video images,” *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(6), 610–622 (2000).
26. R. Vandenhouten, T. Kistel and O. Wendlandt, “A method for optical indoor localization of mobile devices using multiple identifiable landmarks,” *Trans. IoT Cloud Comput.* **1**(1) 1–10 (2015).
27. L. Quan and Z. Lan, “Linear n-point camera pose determination,” *IEEE Trans. Pattern Anal. Mach. Intell.* **21**(8), 774–780 (1999).
28. C. Choi and H. I. Christensen, “3D Pose Estimation of Daily Objects using An rgb-d Camera,” *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Vilamoura, Portugal (2012) pp. 3342–3349.
29. R. Szeliski, *Computer Vision: Algorithms and Applications* (Springer Science & Business Media, Springer-Verlag, NY, USA 2010).
30. J. Hesch, S. Roumeliotis, “A Direct Least-Squares (DLS) Method for PnP,” *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, IEEE, Barcelona, Spain (2011) pp. 383–390.
31. S. Li, C. Xu and M. Xie, “A robust O(n) solution to the perspective-n-point problem,” *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(7), 1444–1450 (2012).
32. J. Fabian and G. Clayton, “Error analysis for visual odometry on indoor, wheeled mobile robots with 3-d sensors, Mechatronics,” *IEEE/ASME Trans.* **19**(6), 1896–1906 (2014).
33. I. Dryanovski, R. G. Valenti and J. Xiao, “Fast Visual Odometry and Mapping from RGB-D Data,” *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Shenzhen, China (2013) pp. 2305–2310.
34. L. Svarm, O. Enqvist, M. Oskarsson and F. Kahl, “Accurate Localization and Pose Estimation for Large 3D Models,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, Ohio, USA, (2014) pp. 532–539.
35. C. Engels, H. Stewénius and D. Nistér, “Bundle adjustment rules,” *Photogramm. Comput. Vis.* **2** 124–131 (2006).

Appendix: Derivations

Derivation of Eq. (A3) from Eqs. (A1) and (A2) is not mentioned in Section 3. In this section, we present intermediate steps of this derivation. We will consider derivations of x component in the following parts of this section. Derivations regarding y component will be provided as needed.

$$\mathbf{n}_3\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_x - \mathbf{n}_1\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd} = 0, \tag{A1}$$

$$\mathbf{n}_3\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_y - \mathbf{n}_2\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd} = 0, \tag{A2}$$

$$\bar{\mathbf{e}}^C = \mathbf{V}^C\tilde{\mathbf{t}}^B_{Bd}. \tag{A3}$$

Observe that Eqs. (A1) and (A2) have multiplication with $\mathbf{n}_1^T$, $\mathbf{n}_2^T$, and $\mathbf{n}_3^T$ vectors which are the components of the matrix $\mathbf{N}^C$ as mentioned before in Eq. (9). Expanded form of the matrix $\mathbf{N}^C$ is

$$\mathbf{N}^C = \begin{bmatrix} \mathbf{n}_1 \\ \mathbf{n}_2 \\ \mathbf{n}_3 \end{bmatrix} = \begin{bmatrix} n_{11} & n_{12} & n_{13} & n_{14} \\ n_{21} & n_{22} & n_{23} & n_{24} \\ n_{31} & n_{32} & n_{33} & n_{34} \end{bmatrix}. \tag{A4}$$

By substituting $\mathbf{n}_1$ and $\mathbf{n}_3$ vectors with the ith row of matrix $\mathbf{N}^C$ in the first term of Eq. (A1), we obtain

$$\mathbf{n}_3\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_x = [n_{31}\ n_{32}\ n_{33}\ n_{34}] \left[\begin{array}{ccc|c} \tilde{\mathbf{t}}_1 & \tilde{\mathbf{t}}_2 & \tilde{\mathbf{t}}_3 & \tilde{\mathbf{t}}_4 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \mathbf{r}^{Bd}e_x, \tag{A5}$$

where $\tilde{\mathbf{t}}_i$ represents $[\tilde{t}_{1i}\ \tilde{t}_{2i}\ \tilde{t}_{3i}]^T$ vector of the matrix $\tilde{\mathbf{T}}^B_{Bd}$. If we separate n_{34} from the other n_{ij} values, we get

$$\mathbf{n}_3\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_x = [n_{31}\ n_{32}\ n_{33}\ 0]\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_x + [0\ 0\ 0\ n_{34}]\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_x. \tag{A6}$$

The result of multiplication in the second term yields

$$n_{34}e_x = [0\ 0\ 0\ n_{34}]\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_x. \tag{A7}$$

So, we obtain Eq. (A8) by making the corresponding substitution in Eq. (A6) with Eq. (A7)

$$\mathbf{n}_3\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_x = [n_{31}\ n_{32}\ n_{33}\ 0]\tilde{T}^B_{Bd}\mathbf{r}^{Bd}e_x + n_{34}e_x. \tag{A8}$$

Then, the terms with the matrix $\tilde{\mathbf{T}}^B_{Bd}$ is moved to the same side of equation yielding

$$n_{34}e_x = \mathbf{n}_1\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd} - [n_{31}\ n_{32}\ n_{33}\ 0]\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_x. \tag{A9}$$

Now, we can divide each side of the equation by n_{34} to isolate e_x

$$e_x = \frac{\mathbf{n}_1}{n_{34}}\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd} - \frac{[n_{31}\ n_{32}\ n_{33}\ 0]}{n_{34}}\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_x. \tag{A10}$$

For further simplification, assume

$$\mathbf{o}_1 = \frac{\mathbf{n}_1}{n_{34}}, \tag{A11}$$

$$\mathbf{o}_2 = \frac{\mathbf{n}_2}{n_{34}}, \tag{A12}$$

$$\mathbf{o}_3 = \frac{[n_{31}\ n_{32}\ n_{33}\ 0]}{n_{34}}, \tag{A13}$$

where $\mathbf{o}_i$ is a 1×4 vector. And, by substituting the $\mathbf{o}_1$ and $\mathbf{o}_3$ values in Eqs. (A11) and (A13) with the corresponding terms in Eq. (A10), we obtain

$$e_x = \mathbf{o}_1\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd} - \mathbf{o}_3\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_x. \tag{A14}$$

Similarly, the error in y component may be presented using the same derivations

$$e_y = \mathbf{o}_2\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd} - \mathbf{o}_3\tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}e_y. \tag{A15}$$

Now, if we multiply $\mathbf{o}_i$ vectors with the matrix $\tilde{\mathbf{T}}^B_{Bd}$, we obtain

$$e_x = \begin{bmatrix} \mathbf{o}_{1,(1\to3)}\tilde{\mathbf{t}}_1 \\ \mathbf{o}_{1,(1\to3)}\tilde{\mathbf{t}}_2 \\ \mathbf{o}_{1,(1\to3)}\tilde{\mathbf{t}}_3 \\ \mathbf{o}_{1,(1\to3)}\tilde{\mathbf{t}}_4 + o_{14} \end{bmatrix}^T \begin{bmatrix} r_1^{Bd} \\ r_2^{Bd} \\ r_3^{Bd} \\ 1 \end{bmatrix} - e_x \begin{bmatrix} \mathbf{o}_{3,(1\to3)}\tilde{\mathbf{t}}_1 \\ \mathbf{o}_{3,(1\to3)}\tilde{\mathbf{t}}_2 \\ \mathbf{o}_{3,(1\to3)}\tilde{\mathbf{t}}_3 \\ \mathbf{o}_{3,(1\to3)}\tilde{\mathbf{t}}_4 \end{bmatrix}^T \begin{bmatrix} r_1^{Bd} \\ r_2^{Bd} \\ r_3^{Bd} \\ 1 \end{bmatrix}. \tag{A16}$$

Similarly, after the multiplication with the $\mathbf{r}^{Bd}$, we obtain

$$\begin{aligned} e_x =&(r_1^{Bd}\mathbf{o}_{1,(1\to3)}\tilde{\mathbf{t}}_1 + r_2^{Bd}\mathbf{o}_{1,(1\to3)}\tilde{\mathbf{t}}_2 \\ &+ r_3^{Bd}\mathbf{o}_{1,(1\to3)}\tilde{\mathbf{t}}_3 + \mathbf{o}_{1,(1\to3)}\tilde{\mathbf{t}}_4 + o_{14}) \\ &- e_x(r_1^{Bd}\mathbf{o}_{3,(1\to3)}\tilde{\mathbf{t}}_1 + r_2^{Bd}\mathbf{o}_{3,(1\to3)}\tilde{\mathbf{t}}_2 \\ &+ r_3^{Bd}\mathbf{o}_{3,(1\to3)}\tilde{\mathbf{t}}_3 + \mathbf{o}_{3,(1\to3)}\tilde{\mathbf{t}}_4). \end{aligned} \tag{A17}$$

Now, we can reorder the terms of Eq. (A17) and then group the common terms as follows:

$$\begin{aligned} e_x - o_{14} =&\left(r_1^{Bd}\mathbf{o}_{1,(1\to3)} - e_x r_1^{Bd}\mathbf{o}_{3,(1\to3)}\right)\tilde{\mathbf{t}}_1 \\ &+ \left(r_2^{Bd}\mathbf{o}_{1,(1\to3)} - e_x r_2^{Bd}\mathbf{o}_{3,(1\to3)}\right)\tilde{\mathbf{t}}_2 \\ &+ \left(r_3^{Bd}\mathbf{o}_{1,(1\to3)} - e_x r_3^{Bd}\mathbf{o}_{3,(1\to3)}\right)\tilde{\mathbf{t}}_3 \\ &+ \left(\mathbf{o}_{1,(1\to3)} - e_x\mathbf{o}_{3,(1\to3)}\right)\tilde{\mathbf{t}}_4. \end{aligned} \tag{A18}$$

This form of Eq. (A18) enables us to rewrite $\vec{r_i}$ and $\tilde{\mathbf{t}}_4$ parameters as vector format which is actually equal to $\tilde{\mathbf{t}}_{Bd}^B$ as mentioned in Eq. (A3). In this way, unknown parameters of matrix $\tilde{\mathbf{t}}_{Bd}^B$ is separated from known variables as

$$e_x - o_{14} = \begin{bmatrix} r_1^{Bd}\mathbf{o}_{1,(1\to3)} - e_x r_1^{Bd}\mathbf{o}_{3,(1\to3)} \\ r_2^{Bd}\mathbf{o}_{1,(1\to3)} - e_x r_2^{Bd}\mathbf{o}_{3,(1\to3)} \\ r_3^{Bd}\mathbf{o}_{1,(1\to3)} - e_x r_3^{Bd}\mathbf{o}_{3,(1\to3)} \\ \mathbf{o}_{1,(1\to3)} - e_x\mathbf{o}_{3,(1\to3)} \end{bmatrix}^T \begin{bmatrix} \tilde{\mathbf{t}}_1 \\ \tilde{\mathbf{t}}_2 \\ \tilde{\mathbf{t}}_3 \\ \tilde{\mathbf{t}}_4 \end{bmatrix}. \tag{A19}$$

In a similar way, the equation regarding y component can be written as

$$e_y - o_{24} = \begin{bmatrix} r_1^{Bd}\mathbf{o}_{2,(1\to3)} - e_y r_1^{Bd}\mathbf{o}_{3,(1\to3)} \\ r_2^{Bd}\mathbf{o}_{2,(1\to3)} - e_y r_2^{Bd}\mathbf{o}_{3,(1\to3)} \\ r_3^{Bd}\mathbf{o}_{2,(1\to3)} - e_y r_3^{Bd}\mathbf{o}_{3,(1\to3)} \\ \mathbf{o}_{2,(1\to3)} - e_y\mathbf{o}_{3,(1\to3)} \end{bmatrix}^T \begin{bmatrix} \tilde{\mathbf{t}}_1 \\ \tilde{\mathbf{t}}_2 \\ \tilde{\mathbf{t}}_3 \\ \tilde{\mathbf{t}}_4 \end{bmatrix}. \tag{A20}$$

In this form of equation, we clearly isolated unknowns from the known values. In order to simplify the notation, let $\mathbf{v}_x$ and $\mathbf{v}_y$ be 1×12 vectors which composes of known values in the right side of Eqs. (A19) and (A20):

$$\mathbf{v}_x = \begin{bmatrix} r_1^{Bd}\mathbf{o}_{1,(1\to3)} - e_x r_1^{Bd}\mathbf{o}_{3,(1\to3)} \\ r_2^{Bd}\mathbf{o}_{1,(1\to3)} - e_x r_2^{Bd}\mathbf{o}_{3,(1\to3)} \\ r_3^{Bd}\mathbf{o}_{1,(1\to3)} - e_x r_3^{Bd}\mathbf{o}_{3,(1\to3)} \\ \mathbf{o}_{1,(1\to3)} - e_x\mathbf{o}_{3,(1\to3)} \end{bmatrix}^T, \tag{A21}$$

$$\mathbf{v}_y = \begin{bmatrix} r_1^{Bd}\mathbf{o}_{2,(1\to 3)} - e_y r_1^{Bd}\mathbf{o}_{3,(1\to 3)} \\ r_2^{Bd}\mathbf{o}_{2,(1\to 3)} - e_y r_2^{Bd}\mathbf{o}_{3,(1\to 3)} \\ r_3^{Bd}\mathbf{o}_{2,(1\to 3)} - e_y r_3^{Bd}\mathbf{o}_{3,(1\to 3)} \\ \mathbf{o}_{2,(1\to 3)} - e_y \mathbf{o}_{3,(1\to 3)} \end{bmatrix}^T. \tag{A22}$$

By substituting $\mathbf{v}_x$ and $\mathbf{v}_y$ values with corresponding fields in Eqs. (A19) and (A20), we reach the final representations of error in x and y components:

$$e_x - o_{14} = \mathbf{v}_x \tilde{\mathbf{t}}_{Bd}^B, \tag{A23}$$

$$e_y - o_{24} = \mathbf{v}_y \tilde{\mathbf{t}}_{Bd}^B. \tag{A24}$$

Consequently, the combination of these two equations gives us Eq. (A3):

$$\bar{\mathbf{e}}^C = \mathbf{V}^C \tilde{\mathbf{t}}_{Bd}^B, \tag{A3}$$

where

$$\bar{\mathbf{e}}^C = \begin{bmatrix} e_x - o_{14} \\ e_y - o_{24} \end{bmatrix}. \tag{A25}$$