Vision-Based Trajectory Tracking for Mobile Robots using Mirage Pose Estimation Method

Semih Dinc[a,*], Farbod Fahimi[b], Ramazan Aygun[a]

University of Alabama in Huntsville, Huntsville, Alabama

[a] *Department of Computer Science*
[b] *Mechanical & Aerospace Engineering Department*

Abstract

Unmanned vehicles are autonomous robotic systems that are fully or partially controlled by an operator remotely from a station. In the last 2 decades, massive amount of advancements have been observed regarding unmanned vehicles for both military and civilian purposes. Today majority of these vehicles require human guidance even for basic missions, thus, minimizing the human intervention on such systems is one of the emerging research topics. To serve this purpose, this study proposes a new trajectory tracking algorithm using *Mirage* pose estimation method. Mirage employs target pixel errors in 2D image plane and analytically calculates the robot's pose in 3D Euclidean space. Therefore, complex computations are not needed and undesirable Euclidean trajectories are avoided since the vehicle pose is directly controlled. We performed both simulations and real experiments to verify the effectiveness of our method. The results show that the proposed method is a feasible alternative for vision-based Euclidean trajectory tracking with high accuracy and low complexity.

Keywords: Trajectory tracking, Vision-based control, Mirage pose estimation, $O(n)$ time complexity

1. Introduction

Target tracking, recognition of the environment, obstacle avoidance, and trajectory following have been investigated for autonomous systems for over two decades. Various types of sensors are installed on robotic or autonomous systems such as robotic arms, unmanned (ground, aerial, and underwater) vehicles, and space exploration robots or rovers. While using these sensors may increase reliability, possible failures and limitations necessitate robust and versatile methods. One of the major challenges for autonomous vehicles is *trajectory tracking* problem that is normally solved by GPS (Global Positioning System) technology [1]. However, using GPS is not a feasible solution when the signal is weak, jammed, or lost. Also, accuracy of pose measurement using GPS is not satisfactory for missions requiring high precision. In such cases, *vision-based* methods are good alternatives with high precision and robustness [2]. In vision-based trajectory tracking, an unmanned vehicle is expected to follow a trajectory by getting visual feedback from camera(s). The vehicle reconstructs its position/velocity and angle/rotation rates with respect to a reference (target) object at any instant of time during the motion.

The studies regarding vision-based trajectory tracking problem can be classified with respect to camera placement and pose error computation [3]. There are three types of camera placement approaches: 1) *eye-to-hand*: the camera is placed on a fixed position [4], 2) *eye-in-hand*: the camera is mounted on the moving vehicle [5], and 3) *hybrid systems*: these support both mechanisms [6]. Another categorization is made regarding the method of pose error measurement [7]. In the first group of studies, the vehicle is guided such that pixel errors in 2D image space vanish *(image based visual servoing)*. In that approach, visual features (edges [8] or corners [9]) on the image are employed with corresponding interaction matrix to determine new linear and angular velocity of the vehicle [10]. These methods are free from camera

*Corresponding Author, Semih Dinc (sd0016@uah.edu), +1-256-200-8472
Email addresses: sd0016@uah.edu (Semih Dinc), ff0002@uah.edu (Farbod Fahimi), aygunr@uah.edu (Ramazan Aygun)

calibration and usually considered to be robust, however, 3D motion of the vehicle cannot be controlled directly, because the Euclidean pose of the vehicle is not calculated. Marchand et al. [11] proposed *virtual visual servoing* framework that adopted an alternative image based (2D visual servoing) approach for real time augmented reality applications. In 2007, Mariottini et al. [12] proposed an image-based control method for non-holonomic mobile robots. Their algorithm was established based on epipolar geometry calculations using current and desired camera views. In 2008 Choi et al. [13] presented another image-based tracking system based on Kande-Lucas-Tomasi (KLT) tracker. Their method runs under the assumption of known 3D geometric model with SIFT description of an object. A recent study in 2014 [14] employs homography decomposition to formulate the kinematic model of the mobile robot after determining the fixed depth parameter. The method was tested on simulated experiments and promising results have been received.

The second group of methods minimize the pose error in 3D Euclidean space by using actual 3D pose of the vehicle *(position based visual servoing)* [15]. This procedure often requires a pose estimation method that calculates the vehicle pose in 3D space using visual features. Beavidez et al. [16] proposed a tracking system by utilizing MS Kinect RGB-D camera, which simplifies the pose measurement problem since a direct depth measurement is possible. A fuzzy logic controller is used to control the robot. Elqursh et al. [17] investigated the robot pose estimation by using 2 parallel lines and 1 more line orthogonal to the others in the image. Estimated pose is employed to control the mobile robot. However, pose estimation part usually involves complex calculations or numerical (iterative) solutions, which reduces the preferability of the method [18]. Even though there exist $O(n)$ time solutions such as [19], [20], our previous experiments show that the robustness is still a challenging problem to solve.

In this study, we propose a novel trajectory tracking algorithm for mobile robots using Mirage pose estimation method, which has linear time complexity and analytically calculates the Euclidean pose of the vehicle in real time. The foundation of the Mirage pose estimation is presented in [21], where we used 2 identical cameras and 1 target to calculate the pose errors on a grid surface. However, it is not limited to these conditions and it allows one or more cameras. It can be integrated as an independent module that replaces navigational sensors in unmanned aerial vehicles as well as underwater or ground vehicles. In our tracking system, the vehicle is able to track its trajectory using only 2 information. The first one is the target image(s) that are captured by the individually calibrated camera(s) mounted on the vehicle. And the second one is the desired pose of the vehicle for that time instant. No explicit 3D pose measurement or depth-analysis (stereo-vision structure) is required to calculate vehicle pose as long as the target points are in the field of view. Our solution can be categorized as an eye-in-hand position based tracking method since the vehicle is controlled in 3D space and a pose estimation technique is employed. The major advantage of our system is that it computes the 3D pose error directly without computing actual 3D pose by reducing the error on 2D image plan using linear calculations and feeding the pose error directly to the controller. The contributions of this paper may be summarized as follows: Our solution

- employs Mirage pose estimation, which is robust and fast compared to the other state-of-art methods,
- performs 6 DOF (degrees of freedom) motion in Euclidean space,
- only needs analytical computations, so it has approximately the same computational economy as that of image based systems, while lacks the problem of undesirable Euclidean trajectories.
- needs only single vision to operate, but supports multi-camera systems without using stereo-vision, and
- is platform independent.

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., $\mathbf{M}$, $\mathbf{K}$, $\mathbf{V}$).

2. Background: Mirage Pose Error Estimation

Mirage analytically solves 6 pose parameters in $O(n)$ time with multi-camera support. It assumes the availability of a basic 3D object model and calculates the pose parameters by minimizing the 2D projection

error between the actual and desired image pixel coordinates and returns relative pose of the camera with respect to the object. The relative pose can be fed into the controller to determine next linear and angular velocities. Moreover, Mirage is designed to support multiple cameras without significant overhead while improving its reliability as the field-of-view is extended. We start explaining our method for single-camera systems and later, we show the generalization for multi-camera systems.

Figure 1 is an informative illustration of a sample scenario showing world, vehicle, camera spaces, and target object in the scene. The process starts by a user defining the *desired pose* of the vehicle at any given time ($\mathbf{q}^d = [x^d, y^d, z^d, \theta^d, \phi^d, \psi^d]$), which represents the desired trajectory of the vehicle. In Fig. 1, the vehicle is shown at its desired pose by dashed lines. Suppose that the real vehicle is not at its desired pose. The pose of the real vehicle is called the *actual pose* is denoted by $\mathbf{q} = [x, y, z, \theta, \phi, \psi]$. In Fig. 1, the actual pose is shown by solid lines. There exists a 4×4 transformation that transforms the vehicle's local frame at its actual pose (X_B, Y_B, Z_B) to its local frame at its desired pose X_{Bd}, Y_{Bd}, Z_{Bd}. This matrix is denoted by $\tilde{\mathbf{T}}^B_{Bd} = [\mathbf{R}^B_{Bd} | \mathbf{t}^b_{Bd}]$. Mirage's aim is to calculate this transformation matrix. Once this transformation matrix is known, the pose error between the actual and the desired vehicle pose ($\tilde{\mathbf{q}}$) will be calculated.

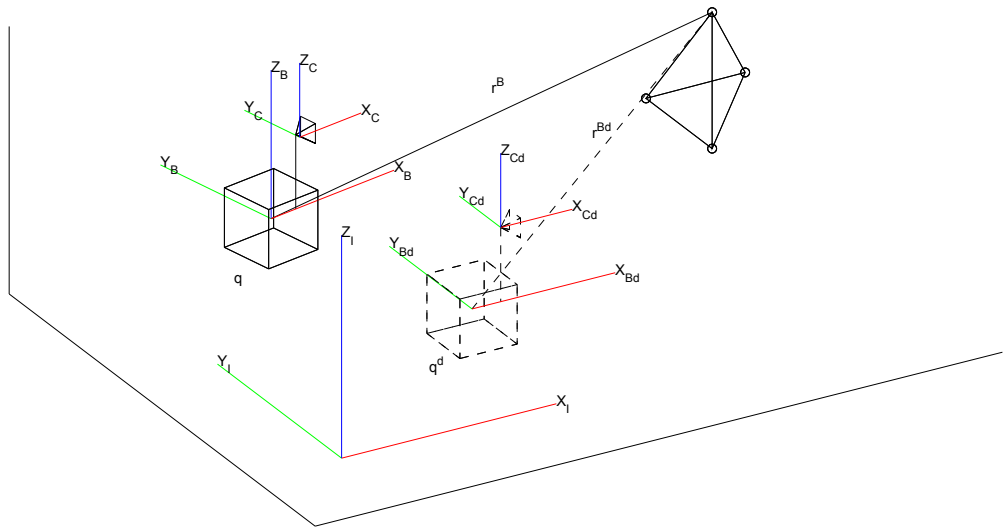


Figure 1: Sample Scene for Mirage. $\mathbf{q}$ and $\mathbf{r}^B$ are actual positions of the vehicle and target object. Similarly, $\mathbf{q}^d$ and $\mathbf{r}^{Bd}$ represent desired coordinates of the vehicle and object points, respectively. Mirage calculates the relative pose of the vehicle ($\tilde{\mathbf{T}}^B_{Bd}$) with respect to its desired position. The desired position $\mathbf{q}^d$, is defined by the user beforehand, and actual position of the vehicle $\mathbf{q}$, can be calculated. I, B, and C represent *world*, *vehicle*, and *camera* spaces, respectively.

When the vehicle is at its desired pose, it has an image of the target in its camera(s). This image is called the desired image, and pixel coordinates of the target's feature points in this desired image are called the *desired 2D* points. When the vehicle is not at its desired pose (i.e. it is at its actual pose), the camera(s) see an image different than the desired image. The pixel coordinates of the target's feature points this image are called the *actual 2D* points.

Our calculations are based on the 2D pixel differences between desired and actual 2D points. Let $\mathbf{r}^B = [r^B_1 \; r^B_2 \; r^B_3 \; 1]^T$ be a 3D point in the actual vehicle space and $\mathbf{r}^{Bd}$ be the corresponding point in desired vehicle space, where $\mathbf{r}^B = \tilde{\mathbf{T}}^B_{Bd}\mathbf{r}^{Bd}$ is satisfied. Even though $\mathbf{r}^{Bd}$ is available, $\mathbf{r}^B$ is unknown, therefore, our goal is to calculate $\tilde{\mathbf{T}}^B_{Bd}$ matrix using actual 2D image pixels that have a relation with $\mathbf{r}^B$. Let $\mathbf{p}^C$ be a 3D point in camera space and $\mathbf{M}^C$ be a 4×4 transformation matrix satisfying following equation,

$$\mathbf{p}^C = \mathbf{M}^C\mathbf{r}^B \tag{1}$$

where $\mathbf{M}^C = [\mathbf{m}_1\ \mathbf{m}_2\ \mathbf{m}_3\ 1]^T$. $\mathbf{M}^C$ is a composite matrix of two different transformation matrices such that,

$$\mathbf{M}^C = \mathbf{K}^C \mathbf{T}_B^C \tag{2}$$

where $\mathbf{K}^C$ and $\mathbf{T}_B^C$ are matrices of extrinsic and intrinsic camera parameters, respectively. $\mathbf{K}^C$ projects the point from camera space into image plane coordinates and the parameters of $\mathbf{K}^C$ (focal length, principle points and distortion parameters) are available a priori via camera calibration. Using the similar idea in Eq. (1), the point in the desired camera space can be calculated by $\mathbf{p}^{Cd} = \mathbf{M}^C \mathbf{r}^{Bd}$. The 3D pose error of the vehicle depends on the actual and desired points. 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

$$\tilde{\mathbf{p}}^C = \begin{bmatrix} \tilde{p}_1^C \\ \tilde{p}_2^C \\ \tilde{p}_3^C \end{bmatrix} = \begin{bmatrix} p_3^{Cd} p_1^C - p_1^{Cd} p_3^C \\ p_3^{Cd} p_2^C - p_2^{Cd} p_3^C \\ p_3^{Cd} p_3^C \end{bmatrix} \tag{3}$$

where $\tilde{\mathbf{p}}^C$ represents the error between actual and desired values of the target points in the camera space. Expression in Eq. (3) is substituted with corresponding values in Eq. (1), then the common factor $\mathbf{r}^B$ is moved out of the parenthesis. New form of the equation becomes

$$\tilde{\mathbf{p}}^C = \begin{bmatrix} \tilde{p}_1^C \\ \tilde{p}_2^C \\ \tilde{p}_3^C \end{bmatrix} = \begin{bmatrix} \mathbf{m}_3 \mathbf{r}^{Bd} \mathbf{m}_1 - \mathbf{m}_1 \mathbf{r}^{Bd} \mathbf{m}_3 \\ \mathbf{m}_3 \mathbf{r}^{Bd} \mathbf{m}_2 - \mathbf{m}_2 \mathbf{r}^{Bd} \mathbf{m}_3 \\ \mathbf{m}_3 \mathbf{r}^{Bd} \mathbf{m}_3 \end{bmatrix} \mathbf{r}^B \tag{4}$$

Let $\mathbf{n}_i$ represent the i^{th} row of Eq. (4). Using this notation, equation can be simplified such that

$$\tilde{\mathbf{p}}^C = \begin{bmatrix} \mathbf{n}_1 \\ \mathbf{n}_2 \\ \mathbf{n}_3 \end{bmatrix} \mathbf{r}^B = \mathbf{N}^C \mathbf{r}^B \tag{5}$$

where $\mathbf{N}^C$ is the matrix of $\mathbf{n}_i$ rows. After substitution of $\mathbf{r}^B$ with $\tilde{\mathbf{T}}_{Bd}^B \mathbf{r}^{Bd}$, Eq. (6) can be derived.

$$\tilde{\mathbf{p}}^C = \mathbf{N}^C \tilde{\mathbf{T}}_{Bd}^B \mathbf{r}^{Bd} \tag{6}$$

Equation (6) 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 from desired and actual pixel coordinates because 2D image errors can be related to $\tilde{\mathbf{p}}^C$. So we can obtain a relation between 2D and 3D image errors as

$$\mathbf{e}^C = \begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} \tilde{p}_1^C / \tilde{p}_3^C \\ \tilde{p}_2^C / \tilde{p}_3^C \end{bmatrix} \tag{7}$$

where $\mathbf{e}$ represents the error between actual and desired 2D pixel coordinates of the 3D points. In this case, $\mathbf{e}$ is a known vector because both the desired and actual images are known. So we may use 2D image errors instead of 3D errors. Following this path, it is possible to reach Eq. (8) after a series of derivations.

$$\bar{\mathbf{e}}^C = \mathbf{V}^C \tilde{\mathbf{t}}_{Bd}^B \tag{8}$$

where $\tilde{\mathbf{t}}_{Bd}^B = [\tilde{t}_{11}\ \tilde{t}_{21}\ \tilde{t}_{31}\ ...\ \tilde{t}_{24}\ \tilde{t}_{34}]^T$ represents a 12×1 vector, which is actually the reshaped form of $\tilde{\mathbf{T}}_{Bd}^B$. $\mathbf{V}^C$ is 2×12 matrix that satisfies the corresponding equations. Equation (8) suggests that $\tilde{\mathbf{t}}_{Bd}^B$ can be computed, since all other parameters in the equation are known.

Equation (8) clearly a linear system of equations, which can be solved by various techniques in the literature. Since there are 12 unknowns in $\tilde{\mathbf{t}}_{Bd}^B$, at least 12 equations are necessary to solve the equation system, in other words $\mathbf{V}^C$ matrix must have a full rank. In a single camera system, one target point generates two equations for 2 image dimensions x and y as (Eq. (7)). Hence, a minimum of 6 distinctive target points are necessary in order to satisfy all 12 equations. However, due to calibration error and dimensional inaccuracies, 12 equations may not lead to the best results. Therefore, adding extra target points to the system is highly recommended to compensate for the camera calibration error and dimensional

inaccuracies. Considering recent feature extracting techniques in the literature, number of matching points will most likely be around hundreds, which eliminates the singularity problem of $\mathbf{V}^C$.

In a real application, handling large number of matching points with noise may be challenging. For such cases, direct solutions are not applicable since the number of rows of $\mathbf{V}^C$ matrix will be much larger than 12 and the noise may cause significant error in calculations. Therefore, a "least squares" based optimization is preferred for solving the equation system. Let us rewrite the Eq. (8) such that $\mathbf{Q}$ becomes a $2n \times 12$ coefficient matrix on the left hand side and $\mathbf{W}$ becomes a $2n \times 1$ constant vector on the right hand side,

$$\mathbf{Q}\tilde{\mathbf{t}}_{Bd}^{B} = \mathbf{W} \tag{9}$$

Considering large number of points, $\mathbf{Q}$ will not be a square matrix, thus Eq. (9) may be solved using linear least squares method such that,

$$\tilde{\mathbf{t}}_{Bd}^{B} = (\mathbf{Q}^T\mathbf{Q})^{-1}(\mathbf{Q}^T\mathbf{W}) \tag{10}$$

Note that, $\tilde{\mathbf{t}}_{Bd}^{B}$ includes 12 elements of the transformation matrix $\tilde{\mathbf{T}}_{Bd}^{B}$. Therefore, a conversion is necessary of 12 elements to the actual rotation angles (in radian) and translation distances. Suppose the calculated unknowns are rewritten as follows,

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

where the values $\tilde{t}_{14}$, $\tilde{t}_{24}$, and $\tilde{t}_{34}$ is conveniently assigned to translation parameters $\tilde{x}$, $\tilde{y}$, and $\tilde{z}$, respectively. For the rotational parameters, however, the following conversions are necessary to compute the angles. Note that the quadrant-dependent function *atan2* is used instead of a simple *arctangent* function. The sign of $\cos(\tilde{\theta})$ determines the quadrant of $\tilde{\phi}$ and $\tilde{\psi}$, so it must not be dropped from the denominators.

$$\tilde{\theta} = \text{atan2}(-\tilde{t}_{31}, \sqrt{\tilde{t}_{11}^2 + \tilde{t}_{21}^2}) \tag{12}$$

$$\tilde{\phi} = \text{atan2}(\frac{\tilde{t}_{32}}{\cos\tilde{\theta}}, \frac{\tilde{t}_{33}}{\cos\tilde{\theta}}) \tag{13}$$

$$\tilde{\psi} = \text{atan2}(\frac{\tilde{t}_{21}}{\cos\tilde{\theta}}, \frac{\tilde{t}_{11}}{\cos\tilde{\theta}}) \tag{14}$$

After the final step, the pose error of the vehicle, $\tilde{\mathbf{q}} = [\tilde{x}, \tilde{y}, \tilde{z}, \tilde{\theta}, \tilde{\phi}, \tilde{\psi}]$, is calculated. This can be easily employed in a position based vehicle controller. Note that all the derivations in this section are analytical. So, numerical (iterative) calculations are not necessary for the proposed method.

Our method can easily be adopted to multi-camera systems without major modifications. Since our algorithm requires fixed number of equations, it is not crucial how many cameras or target points are employed. Moreover, our derivations yield a significant advantage, in which increasing the number of cameras decreases the number of target points required for the solution. Based on the fact that 1 camera can provide 2 equations from a target point, theoretically, the equation $m * n \geq 6$ must be satisfied, where m and n are the number of cameras and target points, respectively. However, in practice at least 4 non-planar points are necessary to localize the vehicle and perform reliable tracking. Using the equations coming from the cameras, $\bar{\mathbf{e}}^C$ vector and $\mathbf{V}^C$ matrix is found and then unknowns can be solved conveniently with the new equation system. There are no major restrictions on the setup 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, which means they do not have to capture the same set of points.

3. Trajectory Tracking using Mirage

In vision based trajectory tracking problems, a moving vehicle is expected to follow a pre-defined (desired) trajectory using actual and reference views of the target. Visual features are extracted from captured images, then a control law is employed to compute necessary velocities to manipulate the vehicle's pose. Depending on the type of the system, the control law may be image based or position based controller. In Fig. 2 (a) and (b), traditional image based and position based trajectory tracking systems are depicted, respectively. As discussed in more detail in the introduction, the major difference between these systems originates from the pose estimation method. Image based systems only minimize the pixel error in 2D image plane, while position based methods can determine the vehicle pose in 3D space. In this study, we offer an alternate trajectory tracking system (Figure 2 (c)), which also has the ability of full motion control as in position based methods. However, our method does not require direct pose measurement via additional equipment and avoids numerical calculations. Also unlike position based methods, our method directly calculates the pose error, not the actual pose. Estimated pose error is used by a closed-loop feedback controller to determine $\mathbf{u} = [\upsilon, \omega]^T$, the linear and angular velocity of the robotic vehicle. Repeating this process for each time instant, the vehicle follows a given trajectory. Please note that, no assumptions are made on the characteristics of desired trajectory as long as the target points are visible. It can be any arbitrary set of continuous positions in time domain.

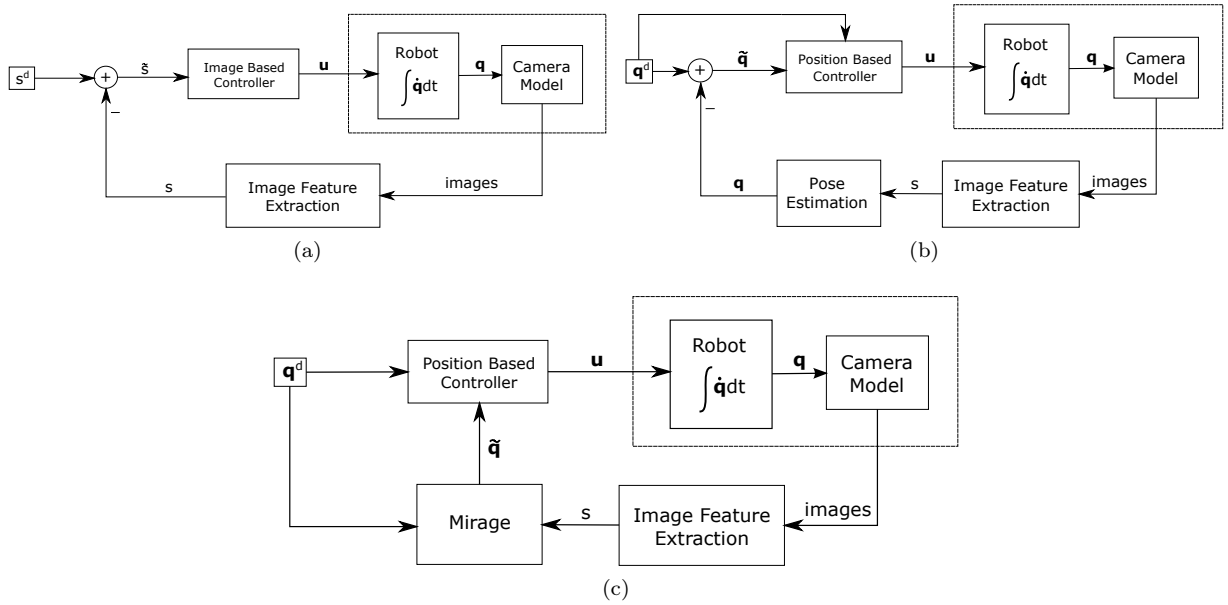


Figure 2: Overview of typical a) Image based, b) Position based and c) Proposed (Mirage based) trajectory tracking systems. $\mathbf{q}$, s, and $\mathbf{u}$ represent the 3D pose, set of image features, and velocity of the vehicle, respectively.

As mentioned in earlier sections, the proposed system offers a generic framework that is capable of controlling the vehicle in 3D with 6 degrees of freedom. So our approach can be applied to robotic systems with no modifications. General form of the kinematic model for a robotic system can be expressed as,

$$\dot{\mathbf{q}} = \mathbf{S}\mathbf{u} \tag{15}$$

where $\dot{\mathbf{q}}$ is the time derivative of the vehicle pose and $\mathbf{S}$ is the vehicle specific matrix. Depending on the motion constraints of the system, $\mathbf{S}$ changes. In the following, characteristics of two sample systems are described. The first system is a 6DOF vehicle with no motion constraints. And the latter is a 3DOF two-wheeled nonholonomic mobile robot. Both systems are selected to demonstrate particular difficulties to be solved.

3.1. 6DOF Unconstrained System

We prefer to start with a 6DOF robotic vehicle (e.g. an underwater robot). In such a system, **S** matrix in Eq. (15) is an identity matrix since no motion constraints are specified. Hence the new form of system model becomes $\dot{\mathbf{q}} = \mathbf{u}$. The controller of the system can be specified as

$$\mathbf{u} = \lambda \tilde{\mathbf{q}} \tag{16}$$

where $\mathbf{u}$ the velocity vector fed into system model, λ is small negative constant and $\tilde{\mathbf{q}}$ is the pose error generated by Mirage. After substituting $\mathbf{u}$ in given equations, the new form of the system model becomes $\dot{\mathbf{q}} = \lambda \tilde{\mathbf{q}}$, which suggests that the motion is realized by direct integration of pose error after multiplying by λ. Please note that this system is applicable to limited real applications, however, it has very simple and straightforward structure, which makes these systems suitable for simulation purposes.

3.2. 2-Wheeled Nonholonomic Mobile Robot

Planar vehicles can also benefit from our tracking system. We applied our approach to a nonholonomic two-wheeled robot in a virtual 2D system with 3 degrees of freedom. Figure 3 demonstrates the schematic view of the chosen robot, where x, y are coordinates of the axis center, and θ is the orientation of the robot, such that $\mathbf{q} = [x, y, \theta]^T$.

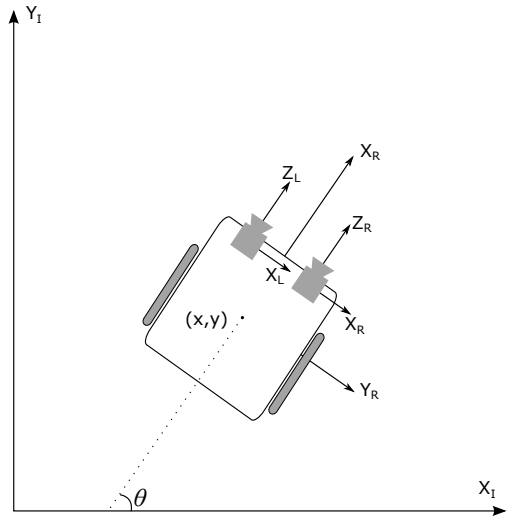


Figure 3: Two-wheeled nonholonomic mobile robot

This system may also be represented by the kinematic model given in Eq. (15) with modified **S** matrix such that

$$\dot{\mathbf{q}} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u} \tag{17}$$

where $\dot{\mathbf{q}}$ is the time derivative of the pose and $\mathbf{u} = [v, \omega]^T$. In simulation, we consider the robot moving along a desired trajectory using the vision system containing two calibrated cameras (not stereo vision mode) mounted on the robot body. For each time instant, robot checks its pose with respect to the reference pose, then moves with the objective of reducing pose error. In this paper, we briefly describe the derivations of controller. The detailed information is provided in [22]. In order to derive the $\mathbf{u}$ vector, the following equation may be used [23]:

$$\mathbf{u} = \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} v^d \cos \tilde{q}_\theta + v^b \\ w^d + w^b \end{bmatrix} \tag{18}$$

where v^d and w^d are the desired input velocities and v^b and w^b are feedback signals which are explained in [22]. The inertial posture error $\tilde{\mathbf{q}}_I = [\tilde{q}_{I_x}\ \tilde{q}_{I_y}\ \tilde{q}_{I_\theta}]^T$ is necessary for computation but the errors calculated by Mirage are only available in local coordinate system not in global coordinate system. Hence, following transformation is necessary

$$\begin{bmatrix} \tilde{q}_{I_x} \\ \tilde{q}_{I_y} \\ \tilde{q}_{I_\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tilde{\mathbf{q}} \tag{19}$$

where $\tilde{\mathbf{q}}$ is the local pose error calculated by Mirage and $\theta = \theta^d + \tilde{\theta}$. In this manner, the posture error model can be shown as

$$\begin{bmatrix} \dot{\tilde{q}}_{I_x} \\ \dot{\tilde{q}}_{I_y} \\ \dot{\tilde{q}}_{I_\theta} \end{bmatrix} = \begin{bmatrix} \cos\tilde{q}_{I_\theta} & 0 \\ \sin\tilde{q}_{I_\theta} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v^d \\ w^d \end{bmatrix} + \begin{bmatrix} -1 & \tilde{q}_{I_y} \\ 0 & -\tilde{q}_{I_x} \\ 0 & -1 \end{bmatrix} \mathbf{u} \tag{20}$$

It is shown in [22] that the error can vanish by employing the feedback signals v^b and w^b as follows:

$$v^b = k_x \tilde{q}_x \tag{21}$$

$$w^b = k v^d \tilde{q}_y (1 + \frac{e_{cos}}{a})^2 + k_s e_{sin} [(1 + \frac{e_{cos}}{a})^2]^n \tag{22}$$

where $\tilde{q}_x$ and $\tilde{q}_y$ are local errors, k_x and k_s are positive values and $n \in Z$. [22] also suggests that n should be chosen as a small number such as -2, -1, 0, 1 or 2 for practical reasons. Parameters k and a are constants larger than zero. And finally directional error parameters, e_{sin} and e_{cos} in Eq. (22) are presented as follows:

$$e_{sin} = \sin(\tilde{q}_{I_\theta}) \tag{23}$$

$$e_{cos} = \cos(\tilde{q}_{I_\theta}) - 1 \tag{24}$$

4. Experimental Results

In this section, we present the simulation and real experiment results of our algorithm described in Section 3.

4.1. Simulation: 6DOF Unconstrained System

The first simulation is applied to the 6DOF vehicle. Using such an unconstrained system, we address a well-known benchmark problem [24] on which image based systems usually generate undesirable motion. This is an unavoidable issue in such systems due to their limited solution space (2D image plane). Optimal minimization on the image plane does not yield optimal solution in 3D space.

The problem involves a square shape target object that is placed on a fixed position and the robotic vehicle is rotated around only one dimension with respect to its desired position. When the system runs, the motion of the vehicle is analyzed until convergence to the desired pose. To test this problem on the proposed method, the initial position of our robot is rotated 30 degrees and placed 2 m behind the desired position. An illustration of this scenario can be seen in Figure 4 with initial and desired positions of the vehicle, and target images for both cases.

This scenario is simulated using traditional image based, position based, virtual visual servoing approach[1] [11], and the proposed approach. Detailed comparison results are shown in Figure 5. Note that only differences between initial and desired pose are in x dimension and ϕ angle. Therefore, the expected motion change must be in only x and ϕ. According to the results, all 4 methods converge to the desired pose. However, there are significant differences in the image based and virtual visual servoing results. First,

[1]Source Code can be downloaded at http://www.irisa.fr/lagadic/visp/visp.html

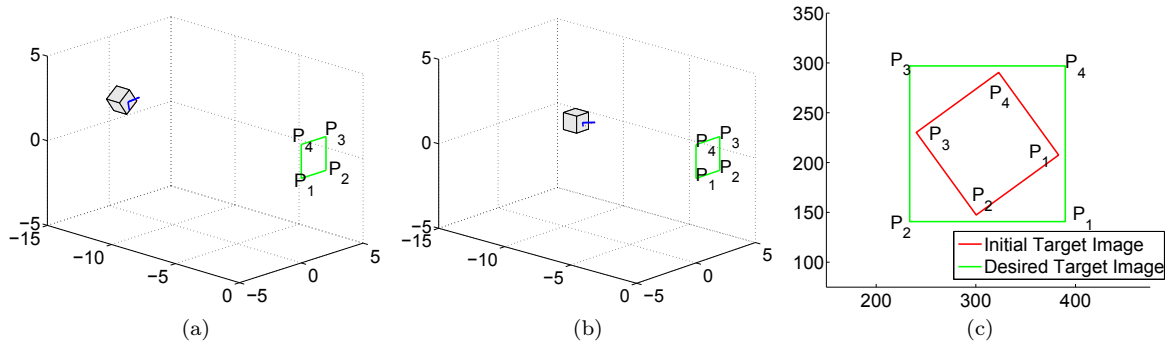


Figure 4: Initial and desired pose of the robot: a) initial position: rotated 30 degrees and placed 2 m behind the desired position, b) desired position, c) initial and desired target images

they converge later than the other 2 methods. Second, the robot makes undesirable motion in almost all dimensions due to 2D error minimization process. Particularly, on x direction, the robot moves back for some time to correct its position. On the other hand, the position based method yields satisfactory results for all dimensions. Majority of position based methods calculate the 3D pose of the robot via numeric (iterative) solutions, which are usually slow for real time implementation. Also there is a possibility to be trapped in local minima during iterative process [25]. Since the 3D pose of the robot is directly fed into the controller, the position based solution generates good results. Our proposed approach provides very similar results with the position based method, while only using the 2D image pixels of the target and obtaining the motion parameters in Euclidean Space with low computational complexity. This is a noticeable advantage of our method over other solutions.

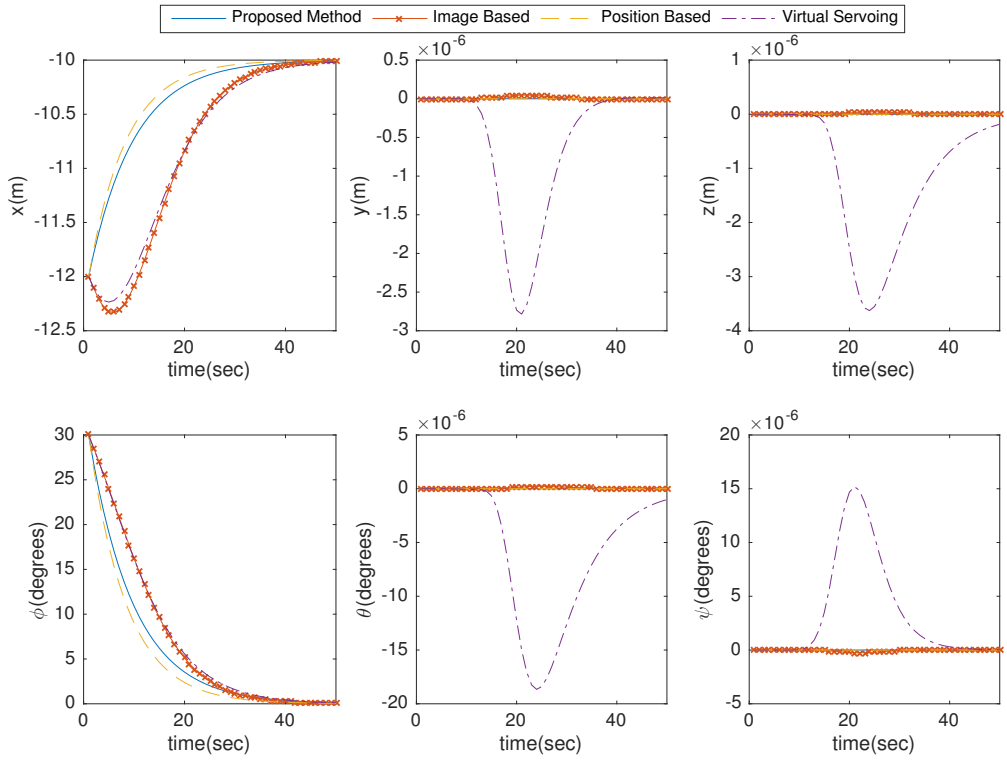


Figure 5: Comparison of 3 methods with respect to 6 degrees of freedom. In image based system and virtual servoing, undesirable motion exist on x, y, and z dimensions, also on θ and ψ angles.

4.2. Simulation: 3DOF Planar System

Second simulation is presented using 3DOF planar system described in Section 3.2. The reference trajectory is defined as a function of time in a 2D environment and the initial pose of the robot is set. The objective is to reduce the pose error by moving the robot as time passes. A sine wave function is selected to represent the trajectory, since it has sufficient complexity for such systems. The initial position of the desired trajectory is $\mathbf{q}_0^d = [-30\ 2\ 0.5610]^T$ and the initial position of the robot is $\mathbf{q}_0 = [-32\ \ -2\ 0]^T$. The simulation is run for 25 seconds. This experiment has been repeated for noise-free and noisy scenarios and the results of 5 state-of-art pose estimation techniques are compared on noisy environment

4.2.1. Noise-Free Environment

In the first case, we consider an ideal condition, where there is no noise in the environment. We intend to see whether our method generates unexpected results under perfect conditions. The controller gains are selected as $k = 1$, $k_x = 1$, $k_s = 1$, $a = 3$, and $n = 0$. Figure 6 illustrates the simulation results.

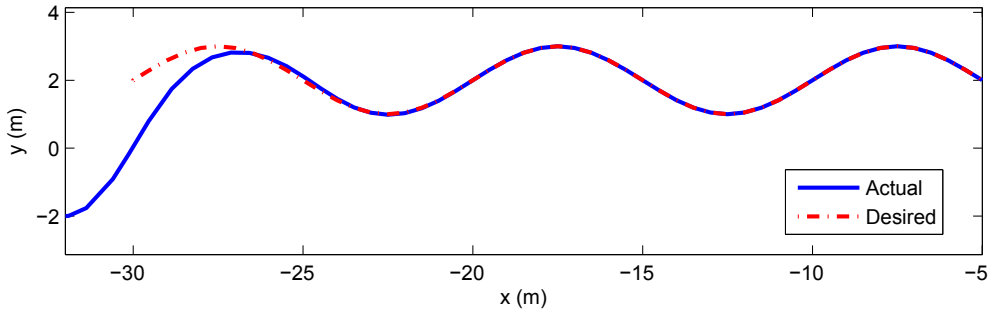
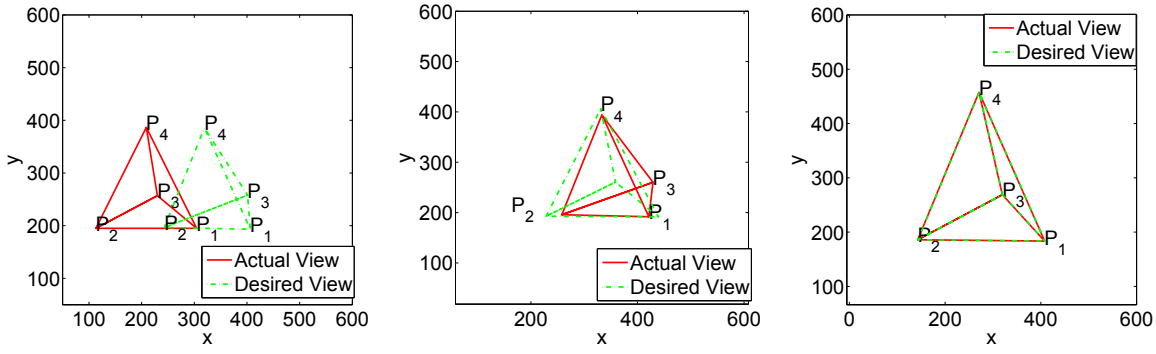


Figure 6: Tracking results under noise-free environment. Convergence is achieved at 9^{th} second of the motion.

As shown in Figure 6, mobile robot converges to the reference trajectory and finally follows it with no error. Figure 7 shows how the image of the target in the left camera looks like when the robot is at actual and desired positions in 3 different frames. The actual image of the target converges to the desired image of the target approximately at the 9^{th} second of the simulation.



(a) (b) (c)

Figure 7: Target image in the left camera; a) at time 0, b) at time 5, c) at time 9

4.2.2. Noisy Environment

In the second case, we added certain amount of gaussian noise ($\sigma = 20$) to the target pixel coordinates and ran the same simulation on 3DOF mobile robot. This time we also tested 5 pose estimation methods: Direct Linear Transform (DLT) [26], Lu-Hager-Mjolsness (LHM) [27], Modern POSIT (Posit) [28], Efficient PnP (EPnP) [19], and Robust PnP (RPnP) [29]. Selected methods include numeric and analytic solutions

with varying complexities. The controller gains are selected as $k = 1$, $k_x = 1$, $k_s = 1$, $a = 3$, and $n = 0$. Figure 8 shows the tracking results of each method.

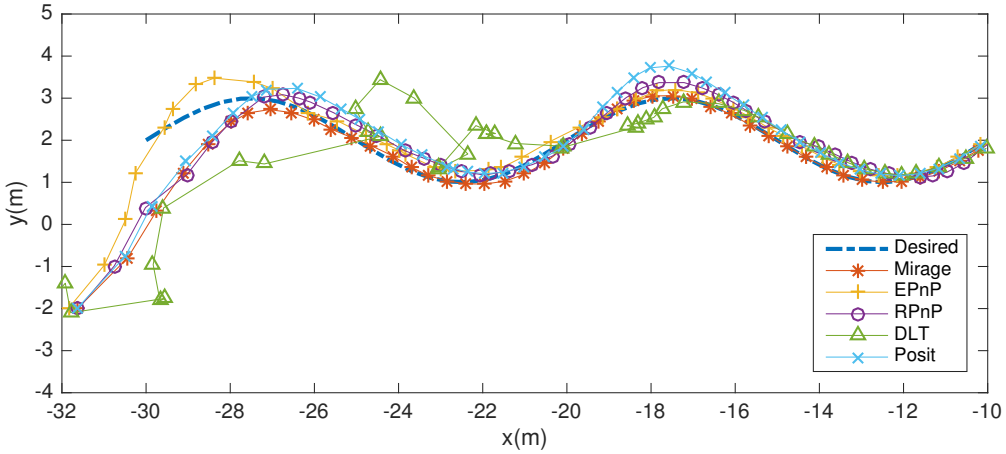


Figure 8: Tracking results under noisy environment

In Figure 8 it can be seen that, DLT generated inconsistent motion, which cannot be applicable to a real system. EPnP, RPnP, and LHM can converge to the desired trajectory with high level of error, which may be critical for some applications. Posit returns relatively good results but it is not as good as Mirage. Our method is the most robust pose estimation method in the given scenario.

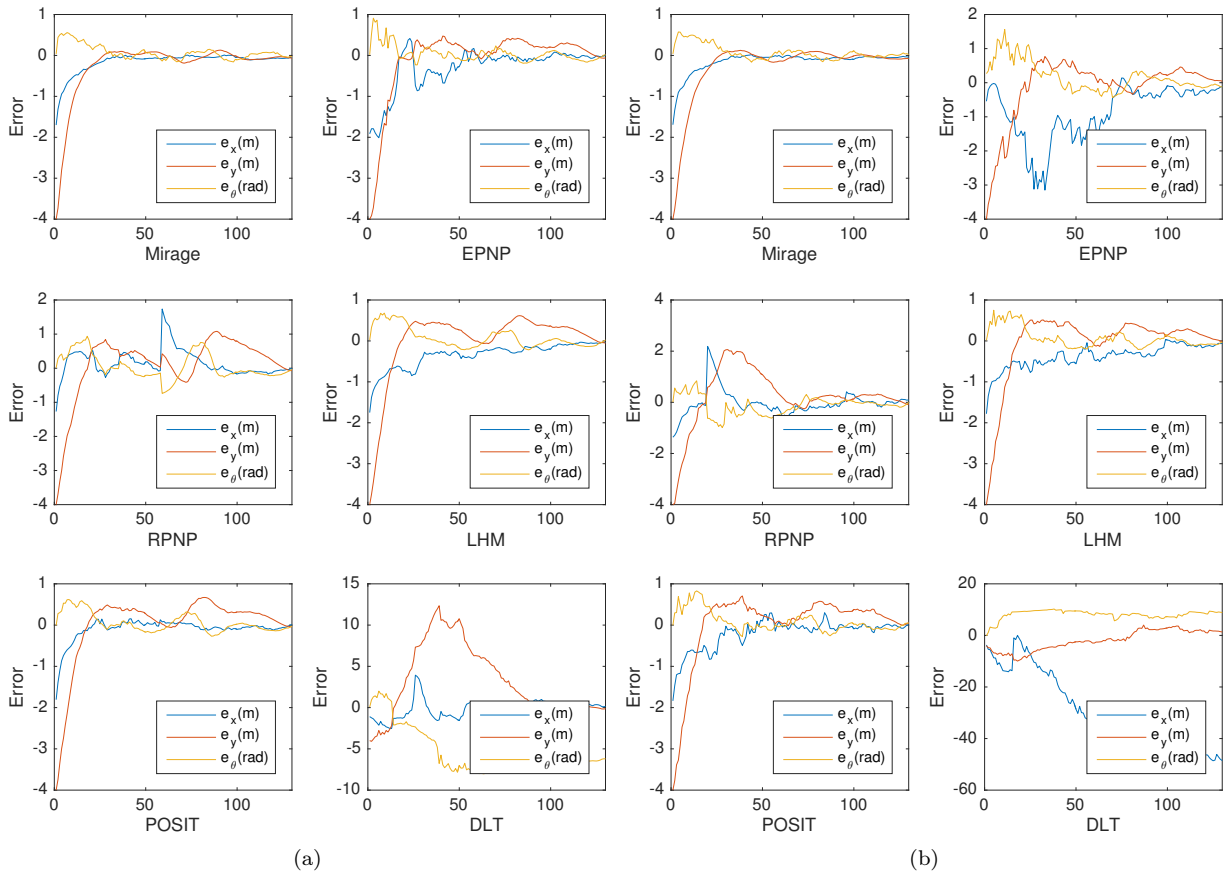


Figure 9: Pose error minimization during the simulation using (a) 20 target points, (b) 5 Target Points

We also track the minimization of pose errors by the time. Figure 9 (a) shows the results of each method using 20 target points. The results clearly indicate that Mirage is very stable under the noise while others produce significant errors. Also, using 20 points is much more than necessary points for Mirage. When we reduce the number of points to 4 (minimum for Mirage) the error rate of some methods becomes very high. Particularly, Posit cannot even calculate the pose in 4 point case. We found that 5 is the minimum number that runs for all methods. So, the same scenario is tested using 5 points and the results are provided in Figure 9 (b). The results show that Mirage generate very similar results using low number of target points while some other methods, such as EPnP and RPnP, generate significant errors compared to the 20 point case.

To investigate the effect of the number of points, we conducted further experiments from 5 to 100 points and provide the results in Table 1. Table entries show the sum of squared errors (SSE) of a method for given number of points. The results indicate that the error rate of the Mirage is very low for any number of target points and it is independent of the number of points. This is a significant advantage when only limited number of features can be extracted from the images. On the other hand, EPnP and DLT produce very high error on small set of target points and the error decreases by increasing the target points. RPnP and LHM are more robust to the change of target points but the errors are still not as low as Mirage. Posit cannot calculate the pose using 4 points, but then generates relatively good results.

Table 1: Sum of squared errors of pose estimation methods for each dimension with respect to the number of target points

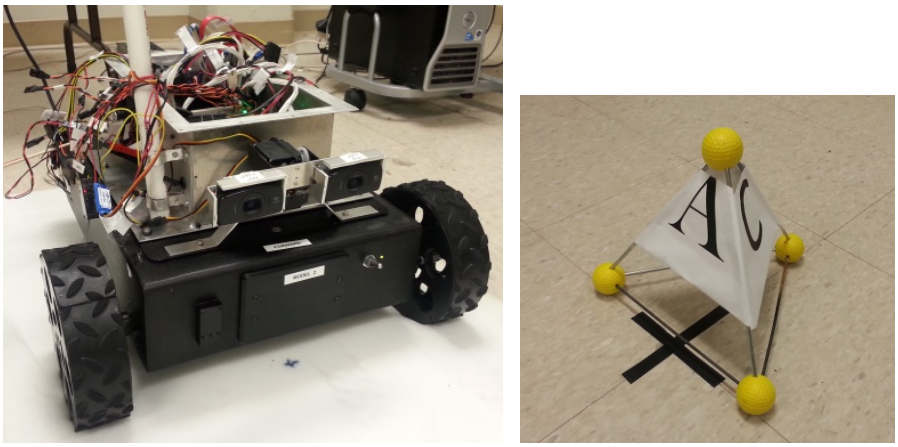
	4 Target Points			20 Target Points			50 Target Points			100 Target Points		
	e_x	e_y	e_θ	e_x	e_y	e_θ	e_x	e_y	e_θ	e_x	e_y	e_θ
Mirage	3.249	8.7845	2.3101	3.3537	8.8294	2.1606	3.4167	8.7393	2.1728	3.1275	8.9024	2.2154
EPnP	50.018	36.722	15.973	8.5741	10.446	4.1263	6.1572	10.016	2.9885	4.6746	9.9953	2.841
RPnP	7.4068	11.912	3.3667	3.7396	9.6026	3.1745	4.0028	10.264	2.8311	2.868	9.787	2.8544
LHM	9.5819	23.235	7.3379	5.8029	9.8064	2.6997	6.0816	9.8986	3.014	5.4213	9.7372	2.8033
Posit	N/A	N/A	N/A	4.1916	9.5632	2.678	3.0467	9.7196	2.7939	3.2983	9.5816	2.5117
DLT	143.62	47.683	70.489	15.984	17.509	8.9795	10.221	12.357	3.8159	5.5212	10.531	4.8191

4.3. Real Time Tracking using Non-holonomic Vehicle

Experimental Setup. Proposed tracking algorithm has also been applied on a real non-holonomic robotic system to present its efficiency with real equipment. In the experiments, a 2 wheeled ground vehicle holding 2 cameras is utilized (Figure 10(a)). 640×480 resolution images are captured from the cameras and then images are processed by onboard computer to detect the target points. The points are fed into Mirage to calculate the pose errors that are converted to velocity using the controller system provided in Section 3.2. Our experiments show that the controller gains in Eq.(21) and Eq. (22) should be $k = 0.3$, $k_x = 1.5$, $k_s = 1.5$, $a = 3$, and $n = 0$ in order to receive accurate results for our robotic system. The time consumed for image acquisition and feature extraction stages is about $285ms$. The time for Mirage pose estimation using 4 feature points and two cameras is about $15ms$.

We defined a 3-meter path and desired trajectory. The vehicle moves on x direction with varying velocity by the time. On this planar space, initial pose of the vehicle is $\mathbf{q}_0 = [-4\ 0\ 0]^T$ and a 36-second desired trajectory is defined such that the vehicle moves forward with sinusoidally increasing velocity for 9-second to the pose $\mathbf{q}_1 = [-2.5\ 0\ 0]^T$ and then continues with sinusoidally decreasing velocity for 9-second to zero, the pose $\mathbf{q}_2 = [-1\ 0\ 0]^T$. After that it stops and moves backwards to $\mathbf{q}_0$ following the same pattern in the remaining 18 *sec.* A 4 point target object in Figure 10(b) is constructed and placed at the origin.

Results. The experiment is repeated 2 times and in total of 150 seconds. The robot is initially at rest at $x = -4m$ and $y = 0m$. The control is switched to automatic tracking at $t = 11$ and $t = 95$ seconds, when the robot starts moving for a period of 36 seconds. The robot is commanded to move straight forward and backward along in the x direction for 3-meter, while increasing and decreasing its speed in a sinusoidal pattern. Considering two 36-second periods in Figure 11, the vehicle follows the trajectory in all 3 dimensions without significant sum of squared errors (SSE) such that $e_x = 4.5204m^2$, $e_y = 0.8261m^2$, and $e_\theta = 0.4942rad^2$.



(a) (b)

Figure 10: a) Non-holonomic vehicle employed in the real experiments, b) 4 point target object

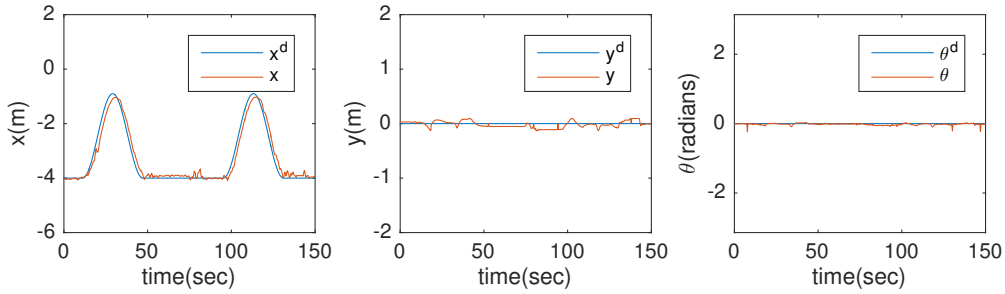


Figure 11: Experiment results with real robotic system

5. Conclusion

This study presents a new vision based trajectory tracking algorithm that uses 2D target image features, however, unlike image based methods, proposed method analytically calculates the pose in 3D Euclidean space with 6 motion parameters. This is a noticeable advantage that provides high accuracy in Euclidean trajectory tracking problems as in position based methods. Moreover, unlike position based methods, the Euclidean pose is calculated analytically without iterative solutions in Mirage. The proposed method can be applied to various types of mobile vehicles having spatial motions (e.g., aircraft, spacecraft, underwater robots). However, it can also be easily applied to vehicles with planar motion. In this study, we demonstrated two sample applications, and provide real environment and simulation results that verify the effectiveness of our method. Our method converges to the trajectory with less error in shorter time than other methods and it is still robust in noisy environments. Both simulations and actual experiments successfully confirm the advantages of the proposed methodology. Based on these promising results, in the future, we aim to improve our system for more realistic and challenging environments such as low illumination and poor detection or undetected target points.

References

[1] C. C. Liebe, K. A. Brown, S. Udomkesmalee, C. W. Padgett, M. P. Brenner, A. M. Howard, T. R. Wysocky, D. I. Brown, S. C. Suddarth, VIGIL: a GPS-based target-tracking system, Proc. SPIE 3365 (1998) 10–21.

[2] X. Shao-Rong, L. Jun, R. Jin-Jun, G. Zhen-Bang, Computer vision-based navigation and predefined track following control of a small robotic airship, Acta Automatica Sinica 33 (3) (2007) 286–291.

[3] Y. H. Huang, M. Hu, H. L. Chong, X. M. Jia, J. X. Ma, W. L. Liu, A survey of robot visual tracking, Key Engineering Materials 501 (2012) 577–582.

[4] B. Bethke, M. Valenti, J. How, Cooperative vision-based estimation and tracking using multiple UAVs, in: Conference of Cooperative Control and Optimization, Gainesville, FL, 2007.

[5] R. Kelly, R. Carelli, O. Nasisi, B. Kuchen, F. Reyes, Stable visual servoing of camera-in-hand robotic systems, IEEE/ASME Transactions on Mechatronics 5 (1) (2000) 39 –48.

[6] J. Feddema, C. Lee, O. Mitchell, Weighted selection of image features for resolved rate visual feedback control, IEEE Transactions on Robotics and Automation 7 (1) (1991) 31–47.

[7] G. Palmieri, M. Palpacelli, M. Battistelli, M. Callegari, A comparison between position-based and image-based dynamic visual servoings in the control of a translating parallel manipulator, Journal of Robotics 2012.

[8] T. Zhao, R. Nevatia, Tracking multiple humans in complex situations, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2004) 1208–1221.

[9] H. Song, M. Shen, Target tracking algorithm based on optical flow method using corner detection, Multimedia Tools and Applications 52 (2011) 121–131.

[10] E. Royer, M. Lhuillier, M. Dhome, J.-M. Lavest, Monocular vision for mobile robot localization and autonomous navigation, Int. J. Comput. Vision 74 (3) (2007) 237–260.

[11] É. Marchand, F. Chaumette, Virtual visual servoing: a framework for real-time augmented reality, in: Computer Graphics Forum, Vol. 21, Wiley Online Library, 2002, pp. 289–297.

[12] G. Mariottini, G. Oriolo, D. Prattichizzo, Image-based visual servoing for nonholonomic mobile robots using epipolar geometry, IEEE Transactions on Robotics 23 (1) (2007) 87–100.

[13] C. Choi, S.-m. Baek, S. Lee, F. Member, Real-time 3D object pose estimation and tracking for natural landmark based visual servo, 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (2008) 3983–3989.

[14] W. MacKunis, N. Gans, A. Parikh, W. E. Dixon, Unified tracking and regulation visual servo control for wheeled mobile robots, Asian Journal of Control 16 (3) (2014) 669–678.

[15] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, H. Durrant-Whyte, Simultaneous localization, mapping and moving object tracking, International Journal of Robotic Research 26 (9) (2007) 889–916.

[16] P. Benavidez, M. Jamshidi, Mobile robot navigation and target tracking system, in: 2011 6th International Conference on System of Systems Engineering, IEEE, 2011, pp. 299–304.

[17] A. Elqursh, A. Elgammal, Line-based relative pose estimation, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2011) 3049–3056.

[18] V. Lippiello, B. Siciliano, L. Villani, Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration, Robotics, IEEE Transactions on 23 (1) (2007) 73–86.

[19] V. Lepetit, F. Moreno-Noguer, P. Fua, Epnp: An accurate O(n) solution to the PnP problem, International journal of computer vision 81 (2) (2009) 155–166.

[20] L. Ferraz, X. Binefa, F. Moreno-Noguer, Very fast solution to the pnp problem with algebraic outlier rejection, in: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE, 2014, pp. 501–508.

[21] S. Dinc, F. Fahimi, R. Aygun, Vision-based trajectory tracking approach for mobile platforms in 3D world using 2D image space, in: ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE), Vol. 4 B, San Diego, CA, United states, 2013.

[22] S. Blaič, A novel trajectory-tracking control law for wheeled mobile robots, Robot. Auton. Syst. 59 (11) (2011) 1001–1007.

[23] Y. Kanayama, Y. Kimura, F. Miyazaki, T. Noguchi, A stable tracking control method for an autonomous mobile robot, in: Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on, 1990, pp. 384 –389 vol.1.

[24] F. Chaumette, S. Hutchinson, Visual servo control, Part I: Basic approaches, IEEE Robotics and Automation Magazine 13 (4) (2006) 82–90.

[25] S. Dinc, F. Fahimi, R. Aygun, Mirage: An O(n) time analytical solution to 3D camera pose estimation with multi-camera support, Computer Vision and Image Understanding (Submitted).

[26] R. Szeliski, Computer vision: algorithms and applications, Springer Science & Business Media, 2010.

[27] C.-P. Lu, G. D. Hager, E. Mjolsness, Fast and globally convergent pose estimation from video images, Pattern Analysis and Machine Intelligence, IEEE Transactions on 22 (6) (2000) 610–622.

[28] D. F. Dementhon, L. S. Davis, Model-based object pose in 25 lines of code, International journal of computer vision 15 (1-2) (1995) 123–141.

[29] S. Li, C. Xu, M. Xie, A robust O(n) solution to the perspective-n-point problem, Pattern Analysis and Machine Intelligence, IEEE Transactions on 34 (7) (2012) 1444–1450.