Proceedings of the 4th
International Conference on Robotics and Mechatronics
October 26-28, 2016, Tehran, Iran

Visual Servoing Simulator by Using ROS and Gazebo

Parisa Masnadi Khiabani, Babak Sistanizadeh Aghdam,
Javad Ramezanzadeh and Hamid D. Taghirad,*Senior Member IEEE*
Advanced Robotics and Automated Systems (ARAS), Industrial Control Center of Excellence (ICCE),
Faculty of Electrical Engineering, K.N. Toosi University of Technology, Tehran, Iran,
Email: p.masnadi@ee.kntu.ac.ir, babaksistani@email.kntu.ac.ir,
j.ramezanzadeh@ee.kntu.ac.ir, taghirad@kntu.ac.ir.

***Abstract*—In this paper, a simulator for five degree of freedom (DOF) visual servoing robot is presented with eye-in-hand configuration. This simulator has been developed in Robot Operating System (ROS) and Gazebo environment. The designed simulator eases the process of testing and debugging visual servoing schemes, and robot controllers. Among different methods, one of the existing Image based visual servoing schemes, image moments, has been implemented to verify the functionality and performance of designed simulator.**

I. Introduction

A robotics simulator is the facility that creates a desirable environment to test a robot in different conditions and for different applications such as mobile and behavior-based robotics. Robotic simulator makes 3D modeling of a robot and its environment possible. It facilitates creation of simple environment of rigid objects and light sources to interact with the robot. Many simulation engines such as Gazebo, Webots, V-rep have been developed so far. In this study, Gazebo is used since it readily integrates with ROS which is one of the most celebrated system software for robotic developments. ROS makes the task of creating robotic platforms with robust behavior much easier. Gazebo provides common tools and sensors that are used by most robots. The start of developing Gazebo was in 2002, while Dr. Andrew Howard and Nate Koenig created the Gazebo in the University of Southern California. In 2009, ROS was amalgamated with Gazebo at Willow. Afterward, From year 2012 till now OSRF is responsible for further development of the Gazebo [1].

Visual servoing utilizes vision sensor information to control robot's motion [2]. It has been used in robot motion tracking and regulation objectives, especially for featureless object. The aim of visual servoing is controlling the system in such a way that minimizes the vision error. Finding the best solution for this purpose needs a lot of repetitive tests and analyses. Although, the software for controlling robot have been progressed widely, testing the real robot in the controller development phase, continuously increases the risk of robot damage due to mechanical or electrical failures. In addition, creating a general robust software which consists every aspects of robot experiments is very challenging [3]. Based on our knowledge, there is no simulation environmemt to test the proposed visual serviong schemes. Motivated by these issues, a visual servoing simulator is implemented based on ROS and Gazebo in this paper. The camera sensor and advanced control plug-in, which are crucial for visual servoing, have been added to this simulator to meet the robots requirement.

First approaches in visual servoing uses geometrical features such as points, straight lines or eclipses to match and track visual measurements [4], [5]. In those approaches the number of objects that may be tracked were very limited. In addition, no analytical form for interaction matrix were defined. In [6] neural network has been utilized to estimate the interaction matrix. Pre-learning process was the weakness of this method. In [7], the author utilized image moments as visual features. Furthermore, the analytical form of interaction matrix for image moments has been determined. Author in [8] has improved the suggested features in previous work. In this paper the recommended features in [8] are used.

A visual servoing robot is simulated in Gazebo simulator facilitates any hardware modification such as camera emplacement with desired properties such as wide, or fish-eye lenses. Moreover, moment based visual servoing is implemented on the simulator to confirm it. The distribution of this simulator eases the debugging and developing process of visual servoing software.

The paper is organized as follows: in section II, the simulator developments in ROS and Gazebo are described in more details. In section.III features extraction, and also the relation that links camera velocity to features' variation are explained. Finally, the implementation results of proposed method is presented in section IV.

II. Model Implementation in ROS and Gazebo

ROS is an open source framework for developing robotics program. This framework contains libraries such as slam, navigation stack etc that makes simulator development process much easier. Moreover, developers all over the world can cooperate with each other and contribute to ROS libraries. Consequently, ROS users benefit from these rich libraries and use them for their robots. In what follows, an explanation on ROS elements will be described briefly.

A node in ROS is a part (process) of whole software which is responsible for computing the specific task. In addition, nodes communicate with each other in order to pass the

information and data. The combination of nodes builds a graph. Each node has a name which is known to all other nodes in the graph. Moreover, the type of each node is originated from its package name. Therefore ROS find executable node location in file system easily.

Nodes communicate with each other through topics. Many nodes can subscribe to a topic, and also publish data to a topic. The topics use TCP/IP and UDP protocols to convey data. It has to be mentioned that ROS topics are used for transmitting one directional continuous data. If the node needs to receive a response or a request, it may use services. Topics uses messages to transmit data. A message is a data structure which may consist of many fields such Integer, floating point, Boolean, etc.

Gazebo is an Open Source software that makes it possible to simulate a robot in an accurate and efficient way in complicated indoor and outdoor environments. It can easily represent the 3D world as the environment in which the robot operates in. In addition, it includes variety of sensors, different robot model and environment, and some physics engines like Open Dynamics Engine (ODE) ,Bullet , Simbody and Dart, to simulate dynamic behavior of objects in the environment.

The proposed simulator mainly contains three packages $aras\ visual\ servo\ gazebo$, $aras\ visual\ servo\ camera$, and $aras\ visual\ servo\ controller$. A package may include one or more nodes. It may consists launch files, configuration files etc. The $aras\ visual\ servo\ gazebo$ package is responsible for launching the robot in gazebo environment through robot's Unified Robot Description Format file (URDF). 3D drawings of all the robot elements are prepared by software such as Solid Works, and then exported into the URDF file. This file is an XML file format which has a tree structure. A joint is defined in terms of a parent and its links and attributes such as origin, axis, limits (lower, upper, velocity, effort) are defined as the children. In addition, camera link and figure box which are crucial parts in visual servoing are defined in URDF file. The $aras\ visual\ servo\ gazebo$ package also contains configuration file for PID controller of the robot. The model for visual servoing robot is shown in Fig. 1.

As it is shown in this figure, a five DOF robot with one degree of redundancy in y direction is considered in this simulation. This robot resembles completely the industrial robot being used in ARAS robotic Lab. The camera has been mounted on the robot's end effector to simulate an eye-in-hand visual servoing configuration. Visual servoing routines fetch the camera data and current joint position to create the connection between Gazebo and VS controller program. For this purpose, subscriber and publisher functions are used in the simulator.

The publisher continually broadcast a message to a specific channel called topic. The subscriber function binds to a specific topic that will be called back, whenever a new message is arrived. The $aras\ visual\ servo\ camera$ package obtains the images from gazebo's camera plug-in and utilizes Open CV's library in order to extract image features and subsequently publishes data to $aras\ visual\ servo\ controller\ node$ based on specific update rate that is defined by user. The graph of the whole simulator is illustrated in Fig. 2. The way Gazebo is communicating with the other nodes is illustrated in this figure. The before-mentioned nodes are represented in circles. The two topics of joint states and camera data are published by Gazebo node and subscribed by $robot\ state\ publisher$ and $aras\ visual\ sevo\ camera$ nodes, respectively. After the $aras\ visual\ servo\ controller$ receives the data, it updates interaction matrix and calculates error signals and control signals and finally publishes the appropriate commands to move the joints.

The flowchart of the whole simulator is presented in Fig. 3. As it is shown in this diagram the structure of whole software is divided into seven sections. The Gazebo simulator generates the camera image with the help of gazebo plug-in and publishes it to $camera\ image$ topic. Later, $aras\ visual\ servo\ camera$ node receives the image and publishes it to $converted\ camera\ image\ topic$. In addition, $aras\ visual\ servo\ camera$ node extracts appropriate features from received images and publishes to relevant topics. Subsequently, $ros\ controller$ receives the commands and publishes the PID coefficients for $ros\ control$. Afterward, the $ros\ control$ node publishes commands which is applicable for $gazebo\ ros\ control$ node. Subsequently, $gazebo\ ros\ control$ node sends the commands for controlling the robot in the simulator (gazebo).

III. Image Moments Modeling

Image moments have been utilized in computer vision for a long time. They can describe any objects or complex shapes effectively. In addition, the combination of moments could be invariant with respect to specific transformation which makes them suitable for visual servoing purposes. The moment m_{ij} may be defined by [9]:

$$m_{ij}(t) = \iint_{R(t)} f(x,y)\, x^i y^j \, dx\, dy \tag{1}$$

if f is considered as a function in $x - y$ plane which is projected in the area R, m will represent the moment of f and

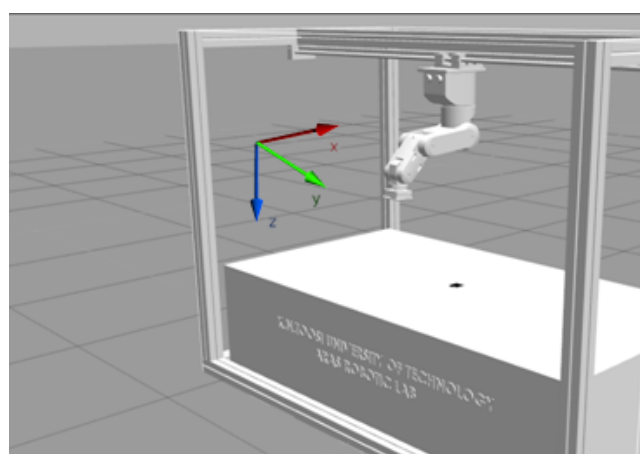

Fig. 1: The model of VS robot

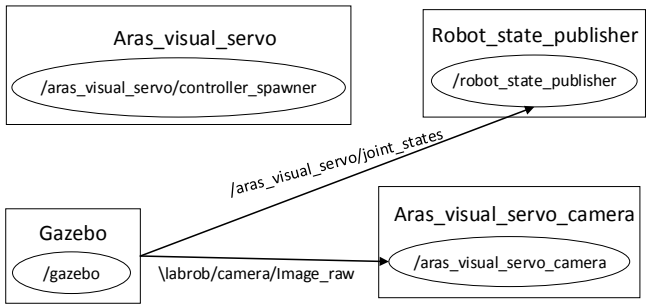

Fig. 2: The simulator graph

ij will determine the order of moment. In a binary image f is considered as intensity of pixel in x-y coordinates, therefore its value could be one or zero. In other words equation (2) may be written in [7]

$$m_{ij}(t) = \iint_{R(t)} f(x,y)\,dx\,dy \tag{2}$$

in which f is considered as $x^i y^j$. The centered moments which are invariant with respect to x-y transformation are defined by [7]:

$$\mu_{ij}(t) = \iint_{R(t)} (x-x_g)^i (y-y_g)^j\,dx\,dy \tag{3}$$

where x_g and y_g are the coordinates of image's center. In order to relate the time variation of moment $\dot{m}_{ij}$ to relative velocity between camera and object, a linear mapping is defined as follow [7]:

$$\dot{m}_{ij} = \mathbf{L}_{m_{ij}}\,\mathbf{v} \tag{4}$$

where $\mathbf{v}$ represents transitional and rotational velocity of camera $\mathbf{v} = (v, \omega)$, and $\mathbf{L}_m$ is the interaction matrix that links camera velocity to the rate of moment changes. The only part in (2) that changes by time is R. The time derivative of the moments may be simplified by use of Green's theorem (for further information refer to [7]) and is defined by:

$$\dot{m}_{ij} = \iint_R (t) \left[\frac{\partial f}{\partial x}\dot{x} + \frac{\partial f}{\partial y}\dot{y} + f(x,y)\left(\frac{\partial \dot{x}}{\partial x} + \frac{\partial \dot{y}}{\partial y}\right)\right] dxdy \tag{5}$$

On the other hand, velocity of any point in image coordinates $\mathbf{x} = (x, y)$ with known depth represented by Z, links to the camera velocity with the well-known relation

$$\dot{\mathbf{x}} = \mathbf{L_x}\mathbf{v} \tag{6}$$

where $\mathbf{L_x}$ is written by

$$\begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -1-x^2 & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{7}$$

If the object is considered planar, the depth of scene points will be related to image point by [7]:

$$\frac{1}{Z} = Ax + By + C \tag{8}$$

A, B, and C are scalar parameters that describe the orientation of plane. When the camera plane is parallel to the image plane $A = B = 0$ [7]. By applying (8) in (7) $\dot{x}$ and $\dot{y}$ are obtained as below

$$\begin{aligned} \dot{x} =& -(Ax+By+C)\cdot v_x + x(Ax+By+C)\cdot v_z \\ &+ xy\cdot\omega_x - (1+x^2)\cdot\omega_y + y\cdot\omega_z \\ \dot{y} =& -(Ax+By+C)\cdot v_y + y(Ax+By+C)\cdot v_z \\ &+ (1+y^2)\cdot\omega_x - xy\cdot\omega_y - x\cdot\omega_z \end{aligned} \tag{9}$$

By replacing (9) in (5) and simplifying the final equation, interaction matrix of moment is obtained. The simplified form of interaction matrix which includes four degree of camera motion may be represented by:

$$\begin{aligned} \mathbf{L}_{m_{ij}}^{v_x} =& -A(i+1)m_{i,j} - Bim_{i-1,j+1} - Cim_{i-1,j} \\ \mathbf{L}_{m_{ij}}^{v_y} =& -Ajm_{i+1,j-1} - B(j+1)m_{i,j} - Cjm_{i,j-1} \\ \mathbf{L}_{m_{ij}}^{v_z} =& A(i+j+3)m_{i+1,j} + B(i+j+3)m_{i,j+1} \\ &+ C(i+j+2)m_{i,j} \\ \mathbf{L}_{m_{ij}}^{\omega_z} =& im_{i-1,j+1} - jm_{i+1,j-1} \end{aligned} \tag{10}$$

similarly, for centered moment one may obtain:

$$\begin{aligned} L_{\mu_{ij}}^{v_x} &= 0, \\ L_{\mu_{ij}}^{v_y} &= 0, \\ L_{\mu_{ij}}^{v_z} &= C\,(i+j+2)\,\mu_{i,\,j}, \\ L_{\mu_{ij}}^{\omega_z} &= i\,\mu_{i-1,j+1} - j\,\mu_{i+1,\,j-1}. \end{aligned} \tag{11}$$

As mentioned earlier, centered moments are invariant with respect to x-y transformation. Therefore, the interaction matrix of them has no elements along camera transitional motion in $x-y$ plane. The zero order moment(m_{00}) calculates the area that is utilized to control the camera motion in z direction. Noting that in parallel situation $\mathbf{Z}^*\sqrt{a^*} = \mathbf{Z}\sqrt{a}$ [8]; therefore, the depth may be calculated. The selected features for controlling the camera velocity in x and y directions are the coordinates of image's mass center($x_g = m_{10}/m_{00}$,

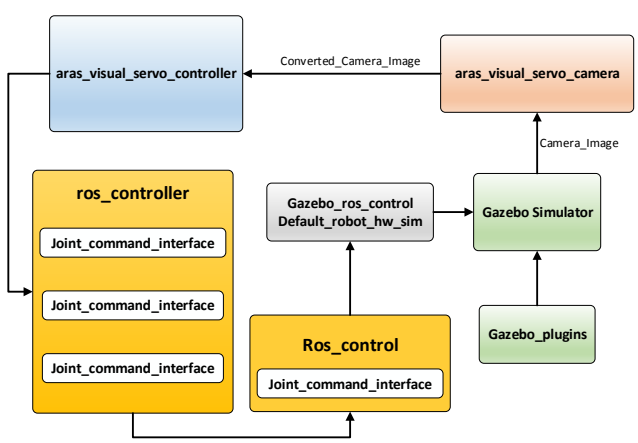

Fig. 3: The simulator diagram

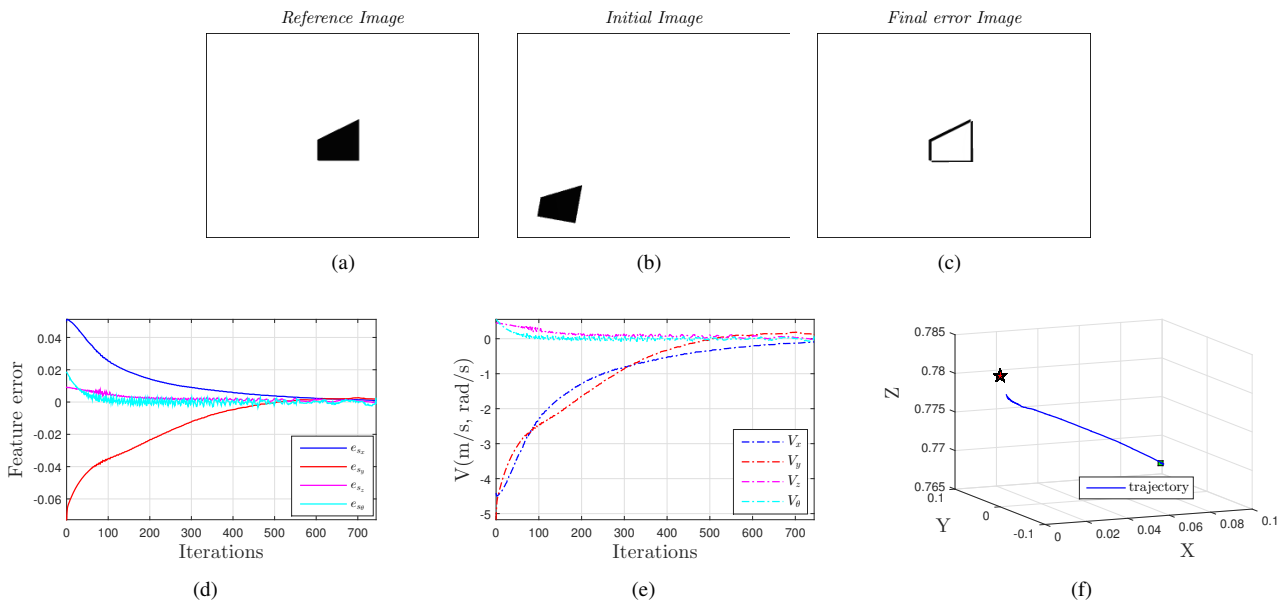

Fig. 4: Simulation results 1. (a) Desired image. (b) Initial image. (c) Difference between final and desired images. (d) Moment errors. (e) Robot velocities. (f) The camera 3D trajectory.

$y_g = m_{01}/m_{00}$). In order to make these coordinates invariant to depth variation, they are normalized [8]. There is a well-known feature for object orientation around optical axes which is defined [10].

The suggested features for visual servoing may be written in:

$$X_n = \frac{m_{10}}{m_{00}}\, a_n \tag{12}$$

$$Y_n = \frac{m_{01}}{m_{00}}\, a_n \tag{13}$$

$$a_n = Z^* \sqrt{\frac{m_{00}^*}{m_{00}}} \tag{14}$$

$$\alpha = \frac{1}{2} tan^{-1} \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \tag{15}$$

where Z^* is the depth of the point relative to the camera frame and the area in target position and m_{00}^* is the area of the object image in target position. By utilizing equations (10) and (11) the interaction matrix of moment for four DOF of camera motion may be written as

$$L = \begin{bmatrix} -1 & 0 & 0 & Y_n \\ 0 & -1 & 0 & -X_n \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{16}$$

by considering equation (4), the control signal is defined by [2]:

$$\mathbf{v} = \mathbf{L}_{m_{ij}}{}^{-1} \dot{m}_{ij} \tag{17}$$

If the target point is considered constant, $\dot{m}$ may be replaced by $\dot{e}$ to support exponential decrease of error signal in equation (18) $\dot{e} \rightarrow -\lambda e$. Therefore, the control signal may be written in

$$\mathbf{v} = -\mathbf{\Lambda}\, \mathbf{L}_{m_{ij}}{}^{-1}\, \mathbf{e} \tag{18}$$

IV. Simulation Results

To examine the accuracy and efficiency of the developed simulator, image moments based method has been implemented on the proposed simulator. The control signals in equation (18) are described in Cartesian space. A five DOF robot has many singular points in its workspace. By utilizing gantry's redundancy target points may choose far from robot's singular points. Therefore, desired Cartesian target points are converted to desired joint target points through the singularity avoidance function which has been developed by ARAS Visual Servo members [11].

The camera provides images with 240×320 resolution at the rate of 30 frame per second. The captured images become binary images by Threshold function [12]. The simulator has been tested for different initial conditions. In all the experiments $\lambda_x = 90$, $\lambda_y = 70$, $\lambda_z = 50$, $\lambda_\theta = 30$ and the sample time is 0.01 second. The control loop ceases when the moment error signals along x and θ direction are below 0.001, along y is below 0.003 and along z is below 0.005. The developed simulator is accessible to everyone through GitHub site [13].

Among several experiments that have been performed, two set of experiments are shown in Fig. 4 and Fig. 5. In Fig. 4(c) the difference between target position and final position has been visually illustrated which are very suitable. As shown in Fig. 4(a) the target position has been chosen at the center of image which makes the coordinates of mass center zero. In Fig. 4(d) feature errors are shown that decreases exponentially as expected. In Fig. 4(e) the computed control signals have been shown and in Fig. 4(f) 3D trajectory has been illustrated, in which, the initial and target point have been marked by circle and star, respectively. The mean of tracking errors

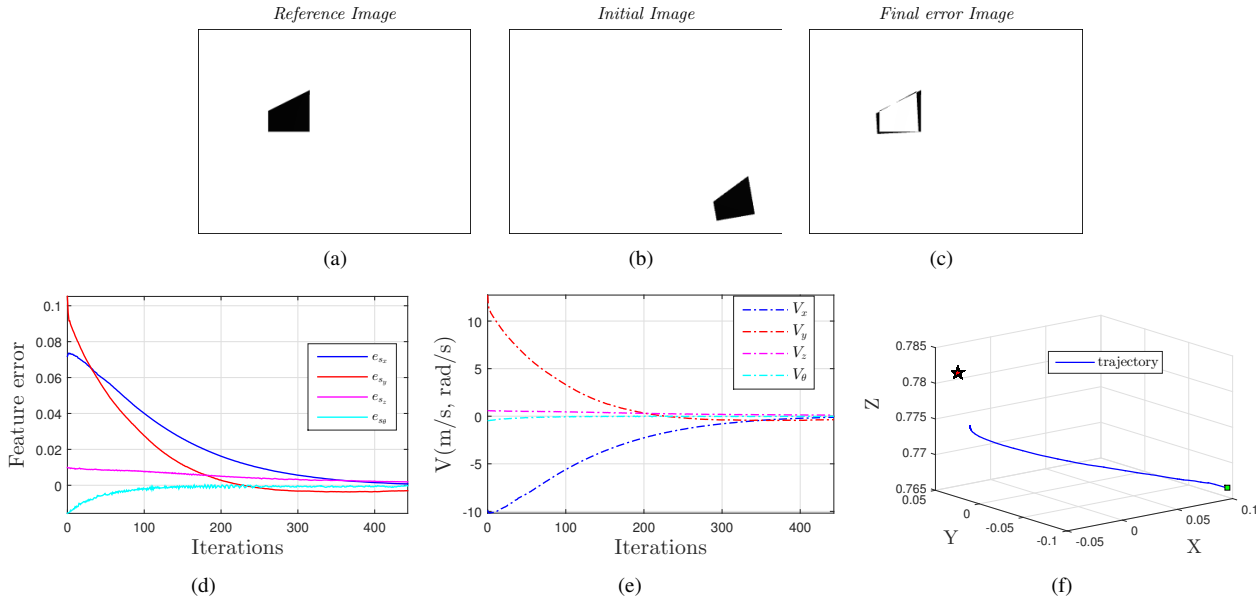

Fig. 5: Simulation results 2. (a) Desired image. (b) Initial image. (c) Difference between final and desired images. (d) Moment errors. (e) Robot velocities. (f) The camera 3D trajectory.

along $x, y, z,$ and θ are 2 mm, 4.7 mm, 2.6 mm, and 0.003 $degree$, respectively. In second set of experiments, the target point is not at the center of image as illustrated in Fig. 5(a). As seen in Fig. 4(c) the difference between final image and target image is more than first set of experiments. In Fig. 4(d) the exponential decreases of error signal are observed. In Fig. 5(f) the trajectory and the before mentioned errors are visible. The mean of tracking errors in second set are 4 mm, 8.9 mm, 7.2 mm, and 0.03 $degree$, respectively. The simulation results ensures the effectiveness of the visual servoing scheme presented in this paper, and motivates to implement it in practice on our real robot in ARAS robotic Lab.

V. Conclusions

In this paper, a visual servoing simulator for a five DOF robot in ROS and Gazebo environment was presented. This simulator facilitates the development of different methods of feature tracking, visual servoing routines, and control algorithm. To validate the proposed simulator, image-based visual servoing for four degree of camera motions have been implemented. The method have been checked for different initial and target points. It is planned to develop six DOF robot simulator in future to assist development of feature selection, and other visual servoing routines for simultaneous six DOF camera motion. This would be a crucial issue in the development and test of the state-of-the-art visual servoing algorithm, before its implementation on real robotic manipulators.

Acknowledgment

The authors would like to thank all visual servoing team member of ARAS Robotic Lab for their kind assistance in preparing VS program. Special thanks goes to Ebrahim Abedloo for providing Solid Work model of the Robot used in this paper.

References

[1] "The gazebo robot simulation," http://gazebosim.org/, 2014, accessed: 2014-10-22.
[2] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 4, pp. 82–90, 2006.
[3] "Ros wiki," http://wiki.ros.org/, 2015, accessed: 2015-07-03.
[4] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 3, pp. 313–326, 1992.
[5] S. Hutchinson, G. D. Hager, P. Corke *et al.*, "A tutorial on visual servo control," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 5, pp. 651–670, 1996.
[6] G. Wells, C. Venaille, and C. Torras, "Vision-based robot positioning using neural networks," *Image and Vision Computing*, vol. 14, no. 10, pp. 715–732, 1996.
[7] F. Chaumette, "Image moments: a general and useful set of features for visual servoing," *Robotics, IEEE Transactions on*, vol. 20, no. 4, pp. 713–723, 2004.
[8] O. Tahri and F. Chaumette, "Point-based and region-based image moments for visual servoing of planar objects," *Robotics, IEEE Transactions on*, vol. 21, no. 6, pp. 1116–1127, 2005.
[9] A. K. Jain, *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.
[10] R. Mukundan and K. Ramakrishnan, *Moment functions in image analysis: theory and applications*. World Scientific, 1998, vol. 100.
[11] H. Taghirad, M. Shahbazi, S. Atashzar, and S. RayatDoost, "A robust pose-based visual servoing technique for redundant manipulators," *submitted to Robotica*, 2012.
[12] R. C. Gonzalez, *Digital image processing*. Pearson Education India, 2009.
[13] "Vs simulator," https://github.com/babaksit/aras_visual_servo/, 2015, accessed: 2015-11-22.