Proc. 2001 Australian Conference on Robotics and Automation
Sydney, 14-15 November 2001

Comparison of Robustness and Performance of Partitioned Image Based Visual Servo Systems

Nicholas R. Gans[1*], Peter I. Corke[2] and Seth A. Hutchinson[1]

[1] Department of Electrical and Computer Engineering
The Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
405 N. Mathews Avenue
Urbana, IL USA 61801
{ngans,seth}@uiuc.edu

[2] CSIRO Manufacturing Science & Technology
Pinjarra Hills
AUSTRALIA 4069.
pic@cat.csiro.au

Abstract

Visual Servoing (VS) has been a viable method of robot manipulator control for almost two decades. Image-Based Visual Servoing (IBVS), in particular, has seen considerable development in recent years. Numerous techniques have been devised to partitioned the control scheme along degrees of freedom, allowing troublesome motions to be handled by methods other than the use of a Jacobian. However, very little independent research has been done to explore the strengths or weaknesses of these methods. In this paper we apply strict tests to several Partitioned Image-Based Visual Servo (PIBVS) systems to test for robustness under the effects of common imaging and robot control errors, as well as conduct tests of performance during difficult control tasks.

1 Introduction

Visual servo control allows for the closed loop control of a robot end effector through the use of image data. In general, there are two approaches to visual servo control: Image-Based Visual Servo (IBVS) and Position-Based Visual Servo (PBVS) [Hutchinson *et al.*, 1996]. In IBVS, an error signal is measured in the image and is mapped directly to actuator commands. In PBVS systems, features are detected in an image and used to generate a 3D model of the environment. An error is then computed in the Cartesian task space, and it is this error that is used by the control system.

IBVS systems enjoy several advantages over PBVS systems. Since the task error is calculated from the image, it is not necessary to have *a priori* knowledge of the 3D task space and goal position. It also becomes a simple matter to regulate the trajectory of image features, for instance preventing them from leaving the field of view. IBVS has its own weaknesses, however. Singularities in the image Jacobian lead to control problems. Image based systems also surrender direct control of the Cartesian velocities, thus while the task error may be quickly reduced to zero, complicated and unnecessary motions may be performed. Finally, the Jacobian is dependent upon feature point depth, which may be unavailable or difficult to estimate accurately.

The problems associated with IBVS systems have led to the creation of several partitioned systems [Malis *et al.*, 1999; Deguchi, 1998; Corke and Hutchinson, 1999]. Partitioned methods use classic, Jacobian-based IBVS to control certain end effector motions while using other techniques to control the remaining degrees of freedom. However, following their introduction there is very little independent investigation of these methods. Most of these methods are designed to address one specific flaw of IBVS, thus it is uncertain how they perform outside of the conditions they were designed for, how they perform in the face of common error conditions, or how well they handle particularly difficult tasks.

In Section 2 we will examine the techniques used in Jacobian-based IBVS, the basic techniques of estimating motion using image data techniques employed by the partitioned systems, and briefly introduce each system tested here. Section 3 will detail the tests performed to test system robustness in the face of common imaging and kinematic errors. Section 4 explains tests performed to test performance when faced with tasks which IBVS systems have generally had a great deal of trouble handling. Rigid protocols will be established to insure the accuracy and merit of all tests, and we will provide probable explanations for the observed results.

2 Background

In this section we present basic concepts involved in Image-Based Visual Servoing. We will also review the established methods of determining camera motion from images using a homography matrix, a method used in several partitioned IBVS system. Finally, we will introduce the IBVS systems being tested and discuss the methods, motivations, and conspicuous strengths and weaknesses of each.

*This work was completed while the first author was visiting CSIRO Manufacturing Science and Technology.

2.1 IBVS

Let $r = (x, y, z)^T$ represent coordinates of the end-effector, and $\dot{r} = (T_x, T_y, T_z, \omega_x, \omega_y, \omega_z)^T$ represent the corresponding end-effector velocity. Let $f = (u, v)^T$ be the image-plane coordinates of a point in the image and $\dot{f} = (\dot{u}, \dot{v})^T$ the corresponding velocities. The image Jacobian relationship is given by

$$\dot{f} = J(r)\dot{r}, \tag{1}$$

with

$$J = \begin{bmatrix} \frac{\lambda}{z} & 0 & \frac{-u}{z} & \frac{-uv}{\lambda} & \frac{\lambda^2+u^2}{\lambda} & -v \\ 0 & \frac{\lambda}{z} & \frac{-v}{z} & \frac{-\lambda^2-v^2}{\lambda} & \frac{uv}{\lambda} & u \end{bmatrix} \tag{2}$$

in which λ is the focal length for the camera. Derivations of this can be found in a number of references including [Hutchinson *et al.*, 1996].

The simplest approach to IBVS is to merely use (1) to construct the proportional feedback control

$$\mathbf{u} = \Gamma J^{-1}(r)\dot{f} \tag{3}$$

in which $\dot{f}$ is the desired feature motion on the image plane, Γ is a gain matrix, and $\mathbf{u} = \dot{r}$ is the control input, an end-effector velocity. In the general case that the Jacobian is not square, the pseudo-inverse, J^+, is used.

2.2 Camera Motion from a Homography Matrix

Define $\mathbf{f_i^*}$, $\mathbf{f_i}$, as coordinates in two images of the 3D points $\mathbf{F_i}, i = 1, .., n$ lying on the plane π. These points are related by

$$\mathbf{f_i} = \mathbf{H}\mathbf{f_i^*} \tag{4}$$

where $\mathbf{H}$ is the $3x3$ homography matrix, which can further be decomposed as

$$\mathbf{H} = \mathbf{R}(\mathbf{I}_3 - \frac{\mathbf{t}\mathbf{n}^T}{d}) \tag{5}$$

Where $\mathbf{I}$ is a $3x3$ identity matrix and $\mathbf{R}$ and $\mathbf{t}$ are the rotation matrix and translation vector, respectively, relating the two camera views; $\mathbf{n}$ and $\mathbf{d}$ are the normal of the plane π and the distance from the second camera origin to the plane π, respectively.

There are numerous methods for computing $\mathbf{H}$ given two sets of image points. We have focused our attention on linear solutions since visual servoing, in general, requires quicker calculations than iterative methods may provide. The major drawback to linear methods is that they are susceptible to noise.

After solving the homography, it is necessary to decompose $\mathbf{H}$ as in (5). Methods to perform this decomposition are detailed in [Faugeras and Lustman, 1988; Zhang and Hanson, 1996]. Decomposing the homography is not a trivial exercise and generally cannot be solved to a unique solution. Additional information or views are required to select form multiple solutions.

2.3 Partitioned IBVS Methods

We will investigate three partitioned IBVS systems, those developed by Malis, *et al* [Malis *et al.*, 1999], Corke and Hutchinson [Corke and Hutchinson, 1999], and Deguchi [Deguchi, 1998]. Each one, addressing one of the drawbacks of classical IBVS, isolates specific degrees of freedom and calculates these motions using distinct methods. The remaining motions are calculated using a reduced Jacobian. To increase clarity, we will not present the precise notation of each author, but rather present each system using a common notation.

2.5D Visual Servoing

Malis, *et al*, [Malis *et al.*, 1999] focus on two things: The necessity of depth estimation in calculating an accurate Jacobian and the fact that IBVS often does not converge to zero error when the initial position is far removed from the goal position. Deriving the rotation $\mathbf{R}$ from the homography matrix (5), and inserting it into the control law, this system is coined *2.5D Visual Servoing* (2.5D).

The feature vector is defined $f = [x, y, z, \theta\mathbf{u}^T]^T$ where x and y are the image coordinates of a selected feature point, z is the depth of the point, and θ and $\mathbf{u}$ are the angle and axis of rotation extracted from $\mathbf{R}$. The task error then becomes

$$\dot{f} = [x - x^*, y - y^*, log\rho, \theta\mathbf{u}^T]^T \tag{6}$$

where ρ is the ratio $\frac{z}{z^*}$ and can be directly calculated from the homography as

$$\rho = \det(\mathbf{H})\frac{n^{*T}m^*}{n^T m} \tag{7}$$

Malis defines the motion control law as

$$\mathbf{u} = -\Gamma J^{-1}(r)\dot{f}, \quad J^{-1} = \begin{bmatrix} \hat{Z}^*\hat{\rho}J^{-1} & -\hat{Z}^*\hat{\rho}J_v J_\omega^{-1} \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix} \tag{8}$$

We expect the 2.5D system to perform well given little or no information on the distance to goal. Additionally, using motion estimation methods to compute rotations should allow it to perform well in the face of extreme rotations. However, reliance on $\mathbf{H}$ would appear to be an Achilles heel as well, adding a great deal of complexity to calculate and decompose. $\mathbf{H}$ can also add an extreme susceptibility to noise and introduces its own deficiencies, as discussed in section 2.2.

Note as well, that while four or more points are required to solve the homography, the Jacobian and velocity are determined using only one point. This may cause motions that would not be present in a least squares solution.

The Method of Deguchi

Deguchi [Deguchi, 1998] addresses the issue that the translation portion of the Jacobian is dependent upon the distance to the goal. Deguchi's method (KD) partitions the system into translational and rotational components. Solving the planar homography, as in (5), delivers the scaled translation $\frac{\mathbf{T}}{d}$. The translational velocity of the end effector is then computed as

$$\dot{r_T} = [T_x T_y T_z]^T = \hat{d}(\frac{\mathbf{T}}{d}) \quad (9)$$

where $\hat{d}$ is an estimate of the distance from the camera goal position to the plane containing the image features. The equation for the rotational motions becomes

$$\dot{r_\omega} = [\omega_x \omega_y \omega_z]^T = -J_\omega^+(\dot{f} + J_T \dot{r_T}) \quad (10)$$

where J_ω and J_T are the rotational and translation portions of the image Jacobian, respectively.

Continuous depth estimation is no longer required, the distance $\hat{d}$ is a constant and only needs to be generated once. Additionally, $\hat{d}$ appears simply as a gain factor to the translational motion component, thus it is not necessary that the estimate be highly accurate.

We expect that Deguchi's Method will have similar strengths and weaknesses to 2.5D. It does not require explicit depth estimation, however the use of the homography matrix adds complexity and increased noise susceptibility.

The Method of Corke and Hutchinson

The method of Corke and Hutchinson (PC&SH)[Corke and Hutchinson, 1999] was designed to avoid the task problem due to a rotation about the optical axis very near 180°. Traditional IBVS systems will attempt to zero each feature error in a straight line, resulting in an attempted backwards motion to infinity. To counter this, Corke and Hutchinson decouple the T_z and ω_z motions from the Jacobian, and calculate these motions using simple image features.

To calculate T_z, σ is defined as the square root of the area inclosed by the feature points. Here we take a departure from the technique described in [Corke and Hutchinson, 1999]. Rather than defining

$$T_z = \gamma_{T_z}(\sigma^* - \sigma) \quad (11)$$

we define

$$T_z = \gamma_{T_z} \ln(\frac{\sigma^*}{\sigma}) \quad (12)$$

Where $*$ indicates a feature in the goal image, and γ_{T_z} is a scalar gain coefficient. This delivers a T_z which varies linearly with motion along the optical axis.

Regarding ω_z, the variable θ is defined as the angle between the u axis of the image and the line segment connecting two points in the image. This leads to

$$\omega_z = \gamma_{\omega_z}(\theta^* - \theta). \quad (13)$$

The equation for x and y motions is

$$\dot{r}_{xy} = \Gamma J_{xy}^+ \left\{ \dot{f} - J_z \dot{r}_z \right\} \quad (14)$$

We thus expect this system to perform well when presented with extreme rotation about z and require little additional computation than traditional IBVS. One readily apparent drawback is the fact that rotations about x and/or y will tend to reduce both the area σ and angle θ in the image, leading to erroneous motions along and about z.

3 Robustness

There are numerous internal and external errors that can assault a visual servo system. We have conducted a series of tests to check the performance the systems presented in 2.3 under the effects of some of these errors.

3.1 General Standards

The whole of our experiments were conducted in simulation using Matlab, the Machine Vision Toolbox and the Robotics Toolbox [Corke, 1996] developed by Peter Corke and publicly available at http://www.cat.csiro.au/cmst/staff/pic/pic.html. Feature points consist of points on the outside of a square in 3D, providing a large possible selection of planer points. The square was simulated as .1m by .1m, and the camera was positioned 1.5 meters from the plane.

Images were projected using a simulated Pulnix TN-6 camera with an 8mm lens and a Datacube Digimax digitizer. The location of feature points are floating point numbers and were not rounded.

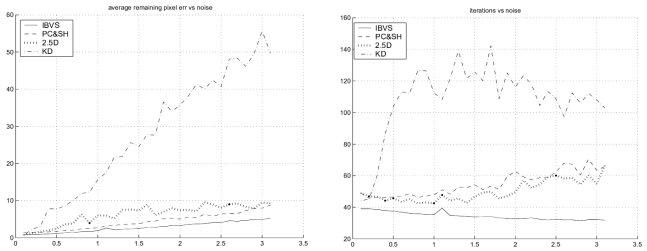
The gain for each system was a 6×6 diagonal matrix, allowing for the individual tuning of each degree of freedom. The gains were selected in order to zero an error for the corresponding degree of freedom in approximately 30 iterations, while errors in the remaining degrees of freedom are held at zero. This is significantly faster than VS systems are typically run, but represents a realistic goal of zeroing an error in one second of video signal.

In the 2.5D and KD systems, the homography matrix was calculated using a linear, least-squares solution. The matrix was decomposed using the methods described by Faugeras [Faugeras and Lustman, 1988]. Ambiguity of multiple solutions was solved by using information from an extra view for the first iteration, and from the previous view during visual servoing.

Visual servoing was considered successful and halted if the average feature point error was less than 0.5 pixels. If the average error varied less than 0.1 pixels for 5 iterations the system was considered to have converged to a constant value and servoing was halted. A system was considered to have failed, and VS halted, if any feature points left the image, the camera retreated more then ten meters or advanced through the image plane, or if the error neither zeroed nor converged within 300 iterations. Techniques exist to keep feature points in the image [Corke and Hutchinson, 1999], but were not incorporated in these tests.

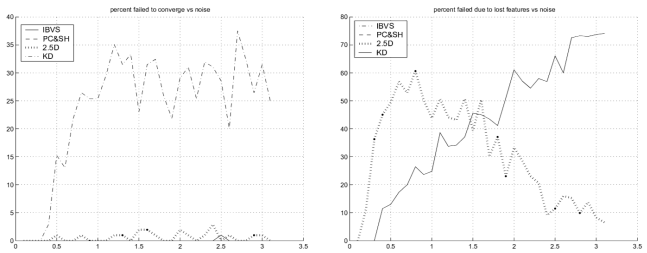
3.2 White Noise

We tested each system versus Gaussian white noise with zero mean and standard deviation increasing from 0 to 3 pixels at steps of 0.1 pixels. Ten feature points were used to over-determine the linear equations for each system. Since white noise is a random process, this experiment was run until each system succeeded or converged 100 times. We tracked the average pixel error left after success or convergence and the number of iterations needed to reach that point. We also tracked the number of failures each system exhibited and calculated the failure rate.



(a) (b)

Figure 1: Performance of IBVS systems vs. increasing white noise. (a)Average remaining pixel error, (b) Iterations to success or convergence



(a) (b)

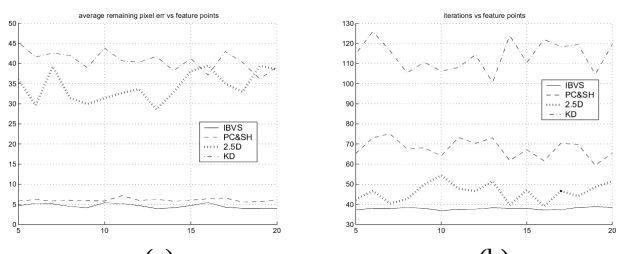
Figure 2: Failure rates of IBVS systems vs. increasing white noise. (a)Percent failed to converge, (b) Percent image features left image

Figure 1 shows the remaining pixel error after convergence as a function of white noise. IBVS, PC&SH and 2.5D all show a fairly gradual, linear increase in pixel error and iterations required to converge. KD experiences a dramatic increase in both pixel error and the number of iterations.

Figure 2 shows failure rates of each system versus white noise. IBVS and PC&SH never fail to converge. 2.5D experiences sporadic failures due to failing to converge within 300 iterations, and a large number of failures due to losing feature points. KD experiences a large number of failures in both categories starting at a standard deviation of 0.7 pixels. The failure rates of these two systems is due to the susceptibility of the homography matrix to noise. It is likely that an iterative solution to $\mathbf{H}$ would give better results, with a trade off for processing time. The fact that the translational component of the homography seems more harshly affected by noise accounts for the poorer performance of KD.

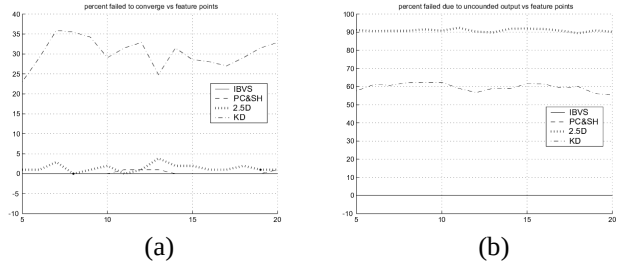
3.3 Number of Feature Points

The flip-side to performance in the presence of noise is how system performance improves as more feature points are added and the system becomes over-determined. For this test, constant white noise with zero mean and standard deviation of 2 pixels was applied to each image. Starting with four feature points, the minimum for solutions for KD and 2.5D, we increased the number of feature points by one up to twenty points. We ran the experiment until each system succeeded or converged 100 times, and tracked the average remaining pixel error, the number of iterations needed to cease VS and the success rate.



(a) (b)

Figure 3: Performance of IBVS systems under noise vs. increasing feature points. (a)Average remaining pixel error, (b) Iterations to success or convergence



(a) (b)

Figure 4: Failure rates of IBVS systems under noise vs. increasing feature points. (a)Percent failed to converge, (b) Percent image features left image

Figures 3 and 4 show the remaining pixel error and iterations to convergence, and the failure rates as functions of the number of feature points, respectively. There appears to be no significant benefit to any system by increasing the number of feature points.

4 Severe Task Performance

There are several control tasks that typically confound IBVS systems. These include positions that give rise to Jacobian singularities and large motions. We investigated performance given three such tasks. The same general standard outlined in Section 3.1 were used here.

4.1 Affine motion

The first test is a substantial translation along x,y and z and a rotation about z. This results in an affine transformation of a planar image, and is a motion any IBVS should be expected to perform. Since this is a fairly simple, though large, motion, it is a good benchmark for system performance. Graphs of the camera motion parameters for each system are seen in Figure 5. The number of iterations each system needed to zero the task error, along with notable features of the trajectory, are presented in Table 1.

4.2 Chaummette Conundrum

The Chaummette Conundrum is the name given to an IBVS failure occurring under a pure rotation about z of near 180°. Using the corners of the square as feature points we increase the rotation from 150°to 180°at steps of 0.5°and observe at which point each system fails. To avoid the singularity of

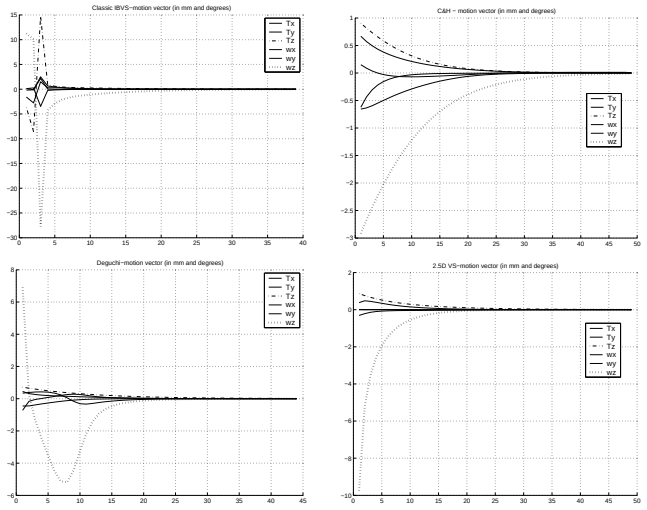


Figure 5: Camera Trajectories During Affine Motion

	reps	notes
IBVS	38	sharp motions
PC&SH	49	smooth motions
KD	48	large ω_z overshoot
2.5D	48	smooth motions, large initial ω_z

Table 1: Performance During Affine Motion

H during pure rotation, a slight motion was induced in the $-z$ direction as well.

Figure 6 shows the feature points trajectories for each system for $\omega_z = 150°$, shedding some light on the motion tendencies. IBVS creates straight feature point trajectories through rotation and and translation on the optical axis. PC&SH features pure rotation about z while KD shows strong ω_z along with ringing by the ω_y ω_x motions. Finally, 2.5D undergoes a complex motion induced by z rotation and fairly strong oscillations in T_x and T_y. Table 2 details the largest angle each system was able to successfully recover from, along with the general cause of failure.

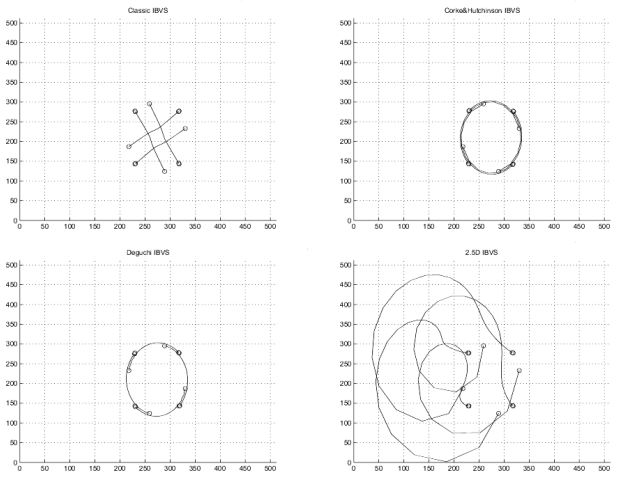


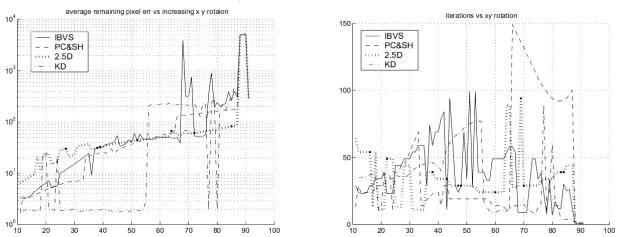
Figure 6: Image feature point trajectories during severe optical axis rotation

	max angle	general failure
IBVS	162	extreme camera retreat
PC&SH	180	successful for all rotations
KD	195	cannot converge
2.5D	160	lost feature points

Table 2: Performance During Chaummette Conundrum

4.3 Severe Rotation

IBVS systems have traditionally shown poor performance when the task involves correcting large rotations about x and/or y. We rotate the feature point plane about both x and y from 10°to 90°at steps of 1°and track each system's performance in terms of remaining pixel error and the number of iterations needed for success or convergence.



(a) (b)

Figure 7: Performance of IBVS systems vs. increasing x and y rotation. (a)Average remaining pixel error, (b) Iterations to success or convergence

Figure 7 shows results from the xy rotation test. There is little evident trend in the number of iterations. However the remaining pixel error shows some interesting features. The average converged pixel error for IBVS increases fairly smoothly to 65°at which point it IBVS becomes unstable and continuously looses feature points. PC&SH follows a fairly linear rise with two rapid increases in average pixel error. KD generally zeroes the error up to 55°, at which point it jumps to an average converged error of about 110 pixels. 2.5D shows smoothly increasing average error until about 87°. All systems are completely unstable after 87°.

5 Future Research

Additional tests need to be designed and performed, and much data remains to be collected. There are robustness issues outside of noise in the image. These include camera calibration errors, error in depth estimation and robot kinematic errors. In addition to the tasks presented here, tasks can be further broken down and investigated along each degree of freedom. Future research will focus on additional robustness tests and motion tasks. Additionally, we will combine these tests, to investigate task performance while under the effects of errors, this will provide much deeper insight into the abilities of these methods.

Following simulation, real experimental results can be obtained. Most of the robustness tests, such as increasing levels of white noise or purposely miscalibrating the camera, will not make for practical experiments. Thus, real ex-

periments should generally revolve around the task performance tests. Even under extremely controlled conditions, experiments will be subject to signal noise, calibration error and distortion of the image due to camera motion, focusing issues and lighting issues. Additionally, discretization of the image into a pixels will add quantization noise that may drastically effect performance.

6 Conclusion

Visual servoing, and robotics in general, is a constantly evolving field. As innovations continue to be made, it becomes increasingly important to explore the different methods in order to gain insight into the characteristics, strengths and weaknesses of each. Focusing on the field of Partitioned Image-Based Visual Servo systems, we have performed several standardized tests of robustness in the face of imaging error and system performance against difficult tasks. This data can be used to select appropriate visual servo systems for specific tasks and conditions or provide direction for future research.

7 Acknowledgements

This material is based in part upon work supported by the National Science Foundation under Award Nos. CCR-0085917 and IIS-0083275.

References

[Corke and Hutchinson, 1999] P. I. Corke and S. A. Hutchinson. A new partitioned approach to image-based visual servo control. In *Proc. on 31st Int'l Symposium on Robotics and Automation*, 1999.

[Corke, 1996] Peter I. Corke. Robotics toolbox for MATLAB. *IEEE Robotics & Automation Magazine*, 3(1):24–32, 1996.

[Deguchi, 1998] K. Deguchi. Optimal motion control for image-based visual servoing by decoupling translation and rotation. In *Proc. Int. Conf. Intelligent Robots and Systems*, pages 705–711, October 1998.

[Faugeras and Lustman, 1988] O.D. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(3):485–508, 1988.

[Hutchinson *et al.*, 1996] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, October 1996.

[Malis *et al.*, 1999] E. Malis, F. Chaumette, and S. Boudet. 2-1/2-d visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, April 1999.

[Zhang and Hanson, 1996] Z. Zhang and A. Hanson. 3d reconstruction based on homography mapping, 1996.