Proceedings of the 2002 IEEE
International Conference on Robotics & Automation
Washington, DC • May 2002

Performance Tests of Partitioned Approaches to Visual Servo Control

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 has been a viable method of robot manipulator control for more than a decade. Image-Based Visual Servoing (IBVS), in particular, has seen considerable development in recent years. Recently, a number of researchers have reported tasks for which traditional IBVS methods fail, or experience serious difficulties. In response to these difficulties, several methods have been devised that partition the control scheme, allowing troublesome motions to be handled by methods that do not rely solely on the image Jacobian.

To date, there has been little research that explores the relative strengths and weaknesses of these methods. In this paper we present such an evaluation. We have chosen three recent visual servo approaches for evaluation, in addition to the traditional IBVS approach. We posit a set of performance metrics that measure quantitatively the performance of a visual servo controller for a specific task. We then simulate each of the candidate visual servo methods for four canonical tasks, under perfect and non-ideal experimental conditions.

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) [1]. 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. They are robust to calibration errors and do not require a full 3D reconstruction of the environment. It also becomes a simple matter to regulate the trajectory of image features, for instance preventing them from leaving the field of view.

However, IBVS has its own weaknesses. 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 on 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 methods which partition the image Jacobian along specific degrees of freedom [2–4]. These partitioned methods use classic, Jacobian-based IBVS to control certain end-effector motions while using other techniques to control the remaining degrees of freedom. We briefly describe these methods, using a single notational framework, in Section 2.

Martinet explored differences between PBVS and IBVS methods [5]. However, there is little information comparing individual IBVS systems. More importantly, there has not been a significant attempt to create a set of metrics to compare performance of different systems nor to determine a valid set of benchmark tests.

In this paper we present an evaluation for the three methods presented in [2–4] along with the classic IBVS approach. To arrive at a meaningful evaluation, we posit a set of performance metrics that measure quantitatively the performance of a visual servo controller for a specific task. These metrics are described in Section 4. We then evaluate each of the candidate visual servo methods for four canonical tasks, under a range of experimental conditions. The set of canonical tasks are described in Section 3, and the experimental conditions are described in Sections 5 and 6. An unwieldy amount of data was collected so only results of particular interest will be reported here, in Section 7. The entirety of the data will be available at http://www-cvr.ai.uiuc.edu/~ngans/vs.html.

2 Candidate Visual Servo Methods

In this paper, we consider the visual servo problem of positioning the camera relative to specified features in the scene. Throughout the paper, we will let $(x, y, z)^T$ represent coor-

dinates of a 3D point that is observed by the camera (expressed relative to the camera coordinate frame), and let $\dot{\mathbf{r}} = (T_x, T_y, T_z, \omega_x, \omega_y, \omega_z)^T$ represent the velocity of the camera frame, composed of a linear velocity $\mathbf{v} = (T_x, T_y, T_z)^T$ and angular velocity $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$. We will use $\mathbf{f} = (u, v)^T$ to represent the image-plane coordinates of the image of the scene point and $\dot{\mathbf{f}} = (\dot{u}, \dot{v})^T$ to represent the corresponding image point velocities.

2.1 Traditional IBVS

At the heart of traditional IBVS is the image Jacobian, given by

$$\dot{\mathbf{f}} = \mathbf{J}(u, v, z)\dot{\mathbf{r}}, \tag{1}$$

with

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

in which λ is the focal length of the camera. Derivations of this can be found in a number of references including [1, 6, 7]. The simplest approach to IBVS is to merely use (1) to construct the control law

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

in which $\dot{\mathbf{f}}$ is the desired feature motion on the image plane, Γ is a (typically diagonal) gain matrix, and $\mathbf{u} = \dot{\mathbf{r}}$ is the control input, an end-effector velocity (this can be converted to joint velocities via the manipulator Jacobian). Since images are acquired at discrete time instants, one typically defines $\dot{\mathbf{f}} = \mathbf{f}^* - \mathbf{f}$, in which the superscript * denotes the goal value for a quantity.

This approach assumes that the image Jacobian is square and nonsingular, and when this is not the case, a generalized inverse, $\mathbf{J}^+$, is used [1]. Since (3) essentially represents a gradient descent on the feature error, when this control law is used, feature points tend to move in straight lines to their goal positions.

Equation (1) can be decomposed, and written as

$$\dot{\mathbf{f}} = \mathbf{J}_{\mathbf{v}}(u, v, z)\mathbf{v} + \mathbf{J}_{\omega}(u, v)\boldsymbol{\omega} \tag{4}$$

in which $\mathbf{J}_{\mathbf{v}}(u, v, z)$ contains the first three columns of the image Jacobian, and is a function of both the image coordinates of the point and its depth, while $\mathbf{J}_{\omega}(u, v)$ contains the last three columns of the image Jacobian, and is a function of only the image coordinates of the point (i.e., it does not depend on depth).

This decomposition is at the heart of the partitioned methods that we discuss below.

2.2 2.5D Visual Servo Control

The 2.5D approach, described in [2], exploits the epipolar geometry that relates a pair of images to determine the rotational component of the desired motion. In particular, if the 3D scene is planar, then the initial and desired images are related by a homography matrix, which can be decomposed into the translational and rotational components of the motion between the two camera configurations.

If $\mathbf{f}$ denotes the feature coordinates in the initial image and $\mathbf{f}^*$ denotes the feature coordinates in the desired image, then the homography matrix satisfies $\tilde{\mathbf{f}} = \mathbf{H}\tilde{\mathbf{f}}^*$, in which the $\tilde{\mathbf{f}}$ and $\tilde{\mathbf{f}}^*$, denote the homogeneous coordinates of $\mathbf{f}$ and $\mathbf{f}^*$, respectively. This homography matrix can be computed from a set of corresponding points in the initial and desired images [8].

The homography can be expressed in terms of the rigid body transformation that relates the two camera configurations as

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

in which $\mathbf{R}$ is the rotation matrix describing the relative orientation of the current and desired camera coordinate frames, $\mathbf{t}$ is the displacement between the two frames, d is the distance from the camera frame in the goal configuration to the plane containing the points, and $\mathbf{n}$ is the normal to the plane containing the four points, expressed relative to the goal camera frame [8].

Using the method described in [9], this homography can be decomposed into the rotational component and a translational component. The translational component can be recovered only up to scale, and therefore, depth must be estimated if the translational component is to be used in a visual servo scheme. However, it should be noted that for most visual servo schemes the depth acts merely as a gain on the translational motion, and therefore exact knowledge of the depth value is not necessary.

It should also be noted that the 2.5D approach does not use the same features as traditional IBVS. In particular, for 2.5D visual servo we have

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

in which θ and $\mathbf{u}$ are the angle and axis of rotation extracted from $\mathbf{R}$, ρ is the ratio $\frac{z}{z^*}$ and can be directly calculated from the homography. The control is given by

$$\dot{\mathbf{r}} = -\Gamma\tilde{\mathbf{J}}^{-1}(r)\dot{\mathbf{f}}, \quad \tilde{\mathbf{J}}^{-1} = \begin{bmatrix} \hat{d}^*\rho\mathbf{J}_{\mathbf{v}}{}^{-1} & -\hat{d}^*\rho\mathbf{J}_{\mathbf{v}}{}^{-1}\mathbf{J}_{\omega} \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix} \tag{7}$$

with $\mathbf{J}_{\mathbf{v}}^{-1}$ and $\dot{\mathbf{f}}$ computed from the information of a single point. Thus, the rotational component of the control is computed directly from the computed desired rotation in 3D, and the translational component is computed by subtracting from the traditional IBVS control a term that accounts for the translational motion in the image induced by the rotation.

Of course, the performance of any approach that relies on decomposing the homography $\mathbf{H}$ will depend on both the quality of the estimate of $\mathbf{H}$ and on the quality of the resulting solution for the decomposition.

There are numerous methods for computing $\mathbf{H}$ given a set of corresponding points from two images. Visual servoing typically requires quicker calculations than iterative methods may provide [10], so we have chosen to use a linear least-squares solution for $\mathbf{H}$ when evaluating this approach. A 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 [9,11]. Decomposing the homography is not a trivial exercise and generally cannot be solved to a unique solution without using additional information or views. In our performance evaluation experiments, at each iteration we use information from the previous iteration to resolve this ambiguity.

2.3 The method of Deguchi (KD)

Deguchi [3] takes a complementary approach to the 2.5-D scheme of Malis et al. In particular, he uses the decomposition of the homography matrix (5) to compute the translation velocity as

$$\mathbf{r_v} = \hat{d}\frac{\mathbf{t}}{d} \tag{8}$$

where $\hat{d}$ is the estimated distance to the plane that contains the 3D reference points, and the ratio t/d^* is the scaled translation that is directly yielded by the decomposition of the homography matrix (5). This leads to the control law

$$\dot{\mathbf{r}}_{\boldsymbol{\omega}} = [\omega_x \omega_y \omega_z]^T = -\mathbf{J}_{\boldsymbol{\omega}}^{+}(\dot{\mathbf{f}}) + \mathbf{J_v}\dot{\mathbf{r}}_{\mathbf{v}} \tag{9}$$

Thus, the translational component of the control is computed directly from the estimated desired translation in 3D, and the rotational component is computed by subtracting from the traditional IBVS control a term that accounts for the motion in the image that is induced by the translation.

In our evaluation of Deguchi's method, we compute the homography and its decomposition as described above in Section 2.2. Deguchi states that a precise depth estimate is not needed. We used the average depth of all image points.

2.4 The method of Corke and Hutchinson (PC&SH)

The method of Corke and Hutchinson (PC&SH) [4] partitions the classical IBVS of (1) so that

$$\dot{\mathbf{f}} = \mathbf{J}_{xy}\dot{\mathbf{r}}_{xy} + \mathbf{J}_z\dot{\mathbf{r}}_z \tag{10}$$

where $\dot{\mathbf{r}}_{xy} = [T_x,\ T_y,\ \omega_x,\ \omega_y]$, $\dot{\mathbf{r}}_z = [T_z\ \omega_z]$, and $\mathbf{J}_{xy}$ and $\mathbf{J}_z$ are respectively columns $\{1,\ 2,\ 4,\ 5\}$ and $\{3,\ 6\}$ of $\mathbf{J}$. We can write (10) as

$$\dot{\mathbf{r}}_{xy} = \mathbf{J}_{xy}^{+}\left\{\dot{\mathbf{f}} - \mathbf{J}_z\dot{\mathbf{r}}_z\right\} \tag{11}$$

where $\dot{\mathbf{f}}$ is the feature point coordinate error as in the traditional IBVS scheme.

In [12], the Z-axis velocity, $\dot{\mathbf{r}}_z$, is based on two image features that are simple and inexpensive to compute (although many other choices for features could have been made). The first feature, σ is defined as the square root of the area enclosed by the feature points. Here we make a slight departure from the technique described in [12], and define

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

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

The second feature, is the angle between the u-axis of the image plane and the directed line segment joining feature points i and j. For numerical conditioning we select the longest line segment that can be constructed from the feature points. The rotational rate is simply

$$\omega_z = \gamma_\theta(\theta^* - \theta), \tag{13}$$

in which γ_θ is a scalar gain coefficient.

3 Canonical Tasks

As discussed above, many of the partitioned approaches have been developed in response to specific problems that are task dependent. Therefore, to evaluate the various methods, we have selected a set of four tasks that we believe are representative, and that present the most interesting tasks encountered by visual servo systems.

Task 1: The first task that we consider corresponds to a pure rotation about the optic axis. The difficulty of this task was first noted in [13]. We evaluate performance over rotations ranging from 30°to 210°.

Task 2: The second task is a pure translation along the optic axis, with initial positions ranging from one meter retreated from goal to one meter advanced through the goal. We have chosen to isolate this direction of motion, since many of the methods depend on the depth from camera to object, and since translation parallel to the image plane generally does not present difficulties to visual servo control.

Task 3: The third task corresponds to a pure rotation of the camera about its y-axis of the camera coordinate frame. The initial values range from 10°to 80°of rotation.

Task 4: The final task is to rotate the 3D feature points about the y-axis of the world frame. The visual servo system will then need to perform rotation and translation in order to zero the image error. The feature points will be rotated through a range of positions from 10°to 80°.

4 Performance Metrics

Before a quantitative evaluation can be performed, it is necessary to posit a set of performance metrics that can be quantitatively evaluated. We have chosen the following performance metrics for our analysis:

Number of iterations to convergence: Visual servoing was considered successful and halted if the average feature point error was less than 1 pixels. If the average error varied less than 0.1 pixels for five iterations the system was considered to have converged to a constant value and servoing was halted.

The number of iterations can obviously be increased or decreased by altering the coefficients of the gain matrix. Thus, the number of iterations itself is not extremely meaningful but proves insightful when comparing the performance of multiple systems or a system vs. task errors.

Error at Termination: At the halt of visual servoing, the remaining pixel error of each point from it's goal position was calculated. These results were averaged to give the average error at termination.

Visual servoing was halted if the the error was successfully zeroed or converged to steady state, as discussed above. Additionally, if over 300 iterations had been performed without convergence visual servoing was halted. Finally, VS was halted if the camera had retreated more then ten meters from goal, advanced through zero meters depth, or the feature points moved to more then 3000 pixels from the the principle point.

Maximum Feature Excursion: At each iteration while visual servoing, the current norm of each feature point to the principle point (center of image) was calculated. The maximum value attained over the entire process was then reported.

Maximum Camera Excursion: At each iteration, the current Cartesian distance of the camera from it's goal position was calculated. The norm of the translation vectors was then calculated to give a current distance. The maximum value attained was reported.

Maximum Camera Rotation: Maximum camera rotation can be difficult to track as singular positions can give rise to enormous angle measures (roll/pitch/yaw angles, for instance) while resulting in little or no actual camera rotation. To gain a useful measure we transformed the rotations into single axis/rotation form and tracked the magnitude of the angle of rotation. The maximum value obtained was reported.

5 Test Conditions

For each of the methods, we evaluated the performance metrics under the following conditions.

Noise in measured pixel coordinates: We simulated errors in feature detection by adding zero-mean Gaussian noise to the ideal image feature coordinates. The variance of this noise is an input parameter for the simulations and ranges from 0 to 0.8 pixels.

Since noise is a random process, the experiment was performed for the full range of motions and variance 100 times the results averaged in order to smooth the results and reduce the effect of outliers.

These results are presented as 3D graphs with variance and the rotation/translation appropriate for the task as independent parameters.

Method of depth estimation: As mentioned above, image-based approaches require some estimate of depth. Several methods have been previously investigated in the literature, including using a constant depth value (often the depth value for the goal configuration), using an an estimation scheme to reconstruct the depth or, in the case of simulations, using the exact depth. The method of depth estimation is an input parameter for our simulations. The results of these tests are 2D graphs showing the performance of each system for each task using the three different depth estimation methods with the appropriate rotation/translation as independent variable.

6 Simulation Methodology

The whole of our experiments were conducted in simulation using Matlab and the Machine Vision Toolbox and Robotics Toolbox [14] (publicly available at http://www.cat.csiro.au/cmst/staff/pic/). For each simulation, feature points consist of the corner points of a square in 3D. The square was simulated as .1m by .1m, and the camera was positioned 1.4 meters from the plane.

Images were projected using a simulated camera with $0.00001m^2$ pixels and a focal length of $0.0078m$. The camera plane was allowed to be infinite. However, if a feature point strayed more then 3000 pixels from the principle point, visual servoing was halted to prevent the presence of extreme outliers in the data set. Additionally, if a system had not zeroed the error or converged to steady state error within 500 iterations, visual servoing was halted. 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 Cartesian freedom. The gains were selected in order to zero an error for the corresponding degree of freedom in approximately 30 iterations, while motion in the remaining degrees of freedom were held at zero. This is significantly faster than VS systems are often run, but represents a realistic goal of zeroing an error in one second of video signal.

7 Notable Results

The four systems in section 2, tested for the four tasks in section 3, under each of the two different test conditions in section 5, produced a vast amount of data. Results of particular interest will be presented here, and the entirety of the results are available at http://www-cvr.ai.uiuc.edu/~ngans/vs.html.

7.1 Rotation About the Optical Axis with Noise

During pure rotation about the optical axis, the maximum camera translation is arguably the most interesting topic. Traditional IBVS has a documented tendency to pull the camera backwards along the z-axis in an attempt to produce straight feature point trajectories. This is seen in Figure 1.

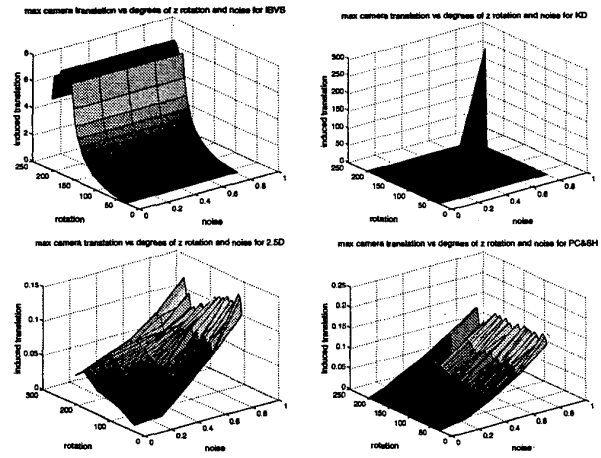

Figure 1: Average max camera translation vs degrees of rotation vs noise variance

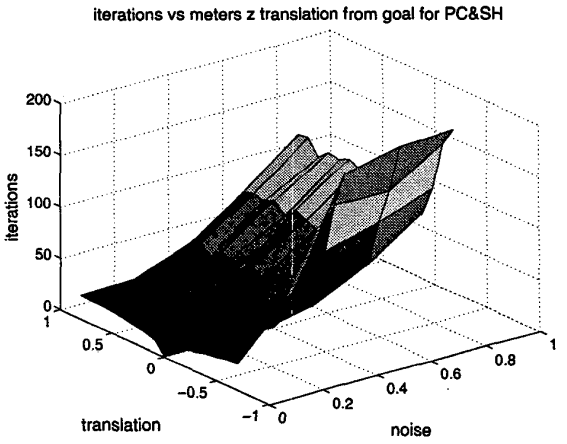

Figure 2: Average iterations vs meters of translation vs noise variance

For IBVS, the max camera translation increases in a roughly exponential fashion as the camera rotation increases. KD has one extremely sharp point of camera motion at 180° of rotation and .8 pixels standard deviation white noise; i.e. the most strenuous conditions experienced and is likely the result of system instability and unbounded output. Both PC&SH and 2.5D suffer negligible camera motion, while experiencing slight increases as noise increases, and the 2.5 D showing a slight increase as rotation approaches 180°.

The results in Figure 1 are closely mirrored by the results for remaining pixel error. IBVS has extremely large error above approximately 160°, and KD can no longer zero the error at 180°. All systems show increased remaining error as noise increases, and 2.5D and KD also show increased error as rotation approaches 180°.

7.2 Translation Along the Optical Axis with Noise

All systems showed good performance during this test, zeroing the error or converging to just a few pixels even under heavy noise conditions. IBVS, 2.5D and KD all showed symmetric decreases in iterations as the initial distance from goal approaches zero from either the positive or negative direction. PC&SH has a slightly non-symmetric graph, requiring more iterations when moving from the negative direction (initially retreated from goal), as seen in Figure 2.

7.3 Rotation About a Parallel Axis with Noise

Each system performs quite differently during this test. Figure 3 focuses on the remaining pixel error for each system, while Figure 4 displays the number of iterations performed. IBVS has extremely little error up to about 35°, at which point it becomes very unstable, leaving extremely large and irregular amount error. The graph of maximum translation looks quite similar; large translation occurring after 35°, which undoubtedly results in the large errors. The number of iterations drops dramatically at this point as well, indicating the instability occurs very quickly.

2.5D performs well; the maximum remaining error is under seven pixel and occurs at 80 degrees of rotation. For lower amounts of camera rotation it performs even better, and seems to be little affected by increasing noise levels. Indeed, it actually appears to have less error at larger noise levels for large rotations. Inversely, the number of iterations for 2.5D becomes extremely large during noisy conditions. KD performs extremely well, reducing error to less then two pixels for camera rotations up to 70°, at which point the remaining error skyrockets to the thousands. Pixel error shows little dependence on noise, although the number of iteration does rise as noise increases.

Finally, PC&SH shows good stability, but performance is poorer than the other partitioned systems, generally reducing error to seven to ten pixels. The brief spike in the graph is likely an outlier that was not perfectly smoothed by the monte carlo simulation. The number of iterations for PC&SH rises quickly as both rotation and noise increase.

7.4 Rotation of feature points with Noise

Figures 5 and Figure 6 show the average remaining pixel error and iterations to convergence respectively. Both the IBVS and 2.5D systems show very good performance for this test, reducing the remaining error below one pixel. Additionally, while the error increases with noise levels, it does not appear dependent on the amount the feature points were rotated. The number of iterations for 2.5D looks very similar, increasing with noise but very little with rotation, whereas for IBVS, the number of iterations climbs dramatically with rotation angle.

KD reduces error well up to 40°of feature point rotation, at which point it becomes unstable and experiences large, irregular error levels. The number of iteration drops at this point

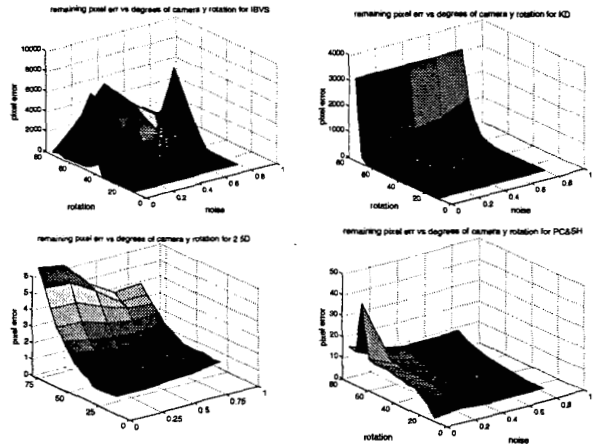

Figure 3: Average remaining pixel error vs degrees of rotation vs noise variance

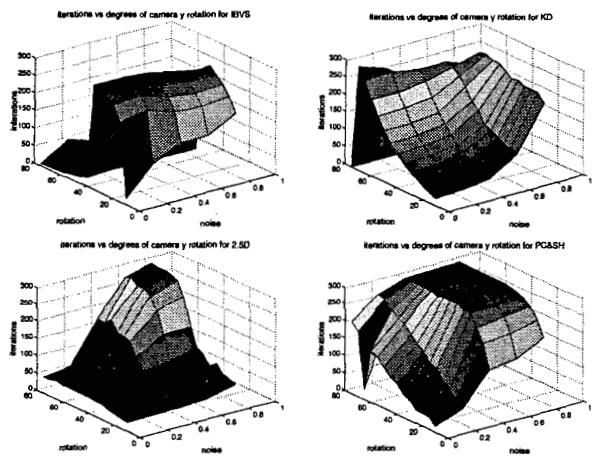

Figure 4: Average iterations to convergence vs degrees of rotation vs noise variance

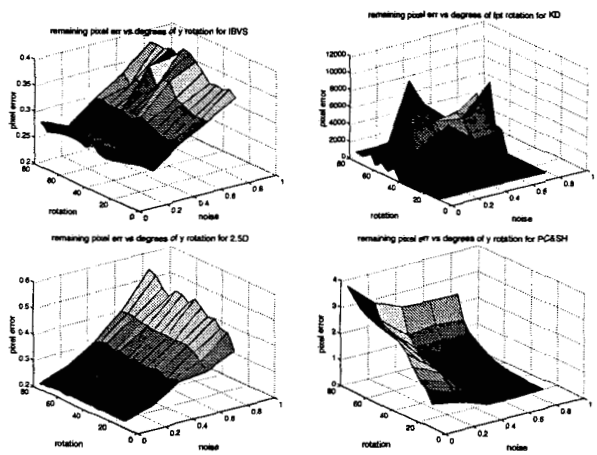

Figure 5: Average remaining pixel error vs degrees of rotation vs noise variance

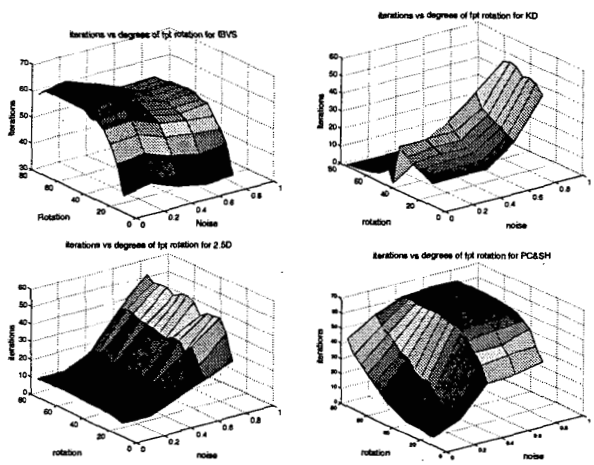

Figure 6: Average iterations to convergence vs degrees of rotation vs noise variance

as well, indicating that the instability surfaces during the first few iterations. PC&SH performs well for smaller rotations, but remaining error rises with the angle of rotation, to almost four pixels at 75°. The number of iterations rises with both noise and angle of rotation

7.5 Rotation About the Optical Axis with Differing Depth Estimation

For the most part, using different depth estimation methods results in little change during pure rotation about the optical axis. This is not surprising as the goal and initial distance are the same, and in the absence of translation are the same as the true depth at every iteration. The one exception is IBVS for severe rotations. As seen in Figure 7, using the initial or goal depth results in less camera retreat than true depth as the angle of rotation increases. in fact, using the goal or initial depth allows for successful visual servoing for all rotations excluding 180°, at the expense of more iterations.

7.6 Translation Along the Optical Axis with Differing Depth Estimation

Different depth estimation methods do not have a dramatic effect on system performance during translation along the optical axis. For PC&SH, there is absolutely no affect since motion along the optical axis is not calculated using the image Jacobian. For the other systems, it only significantly affects the number of iterations. As seen in Figure 8, using the initial point depths raises the number of iterations universally, but has little overall effect on the remaining pixel error. The other error metrics show even less variation between the methods.

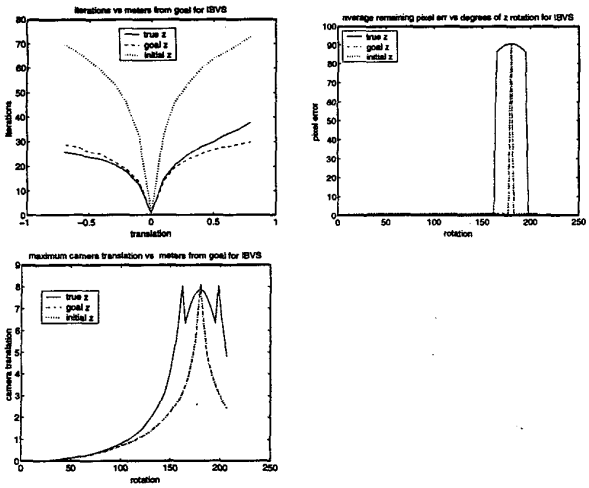

Figure 7: IBVS performance for different depth estimations schemes

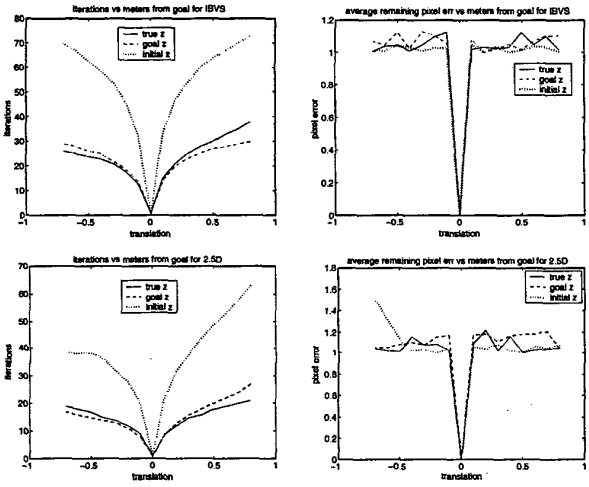

Figure 8: Examples of effects of differing depth estimation schemes while translating along the optical axis

7.7 Rotation About a Perpendicular Axis with Differing Depth Estimation

As seen in Figure 9 the IBVS and 2.5D systems experience dramatic changes under different depth estimation methods. With IBVS, using a constant depth rather than true depth allows the system to zero the error for almost ten more degrees, but the remaining pixel error is dramatically larger (note this is a log scale) after this point. 2.5D shows much worse performance over almost all angles of rotation when estimating depth rather than using true depth. KD shows the same performance regardless of the depth estimation method and is able to reduce the error below one pixel for the majority of

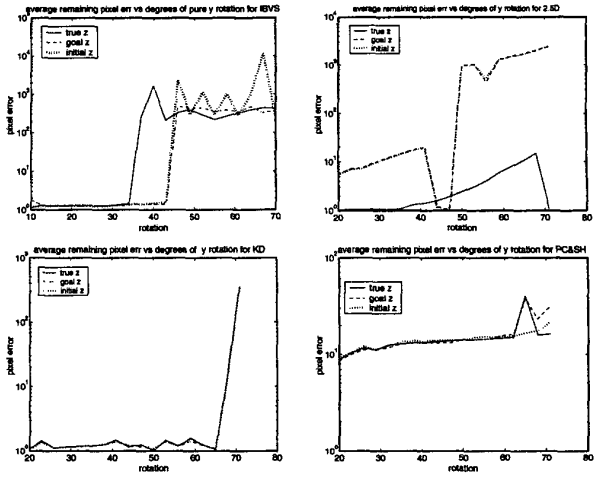

Figure 9: Remaining Pixel Error for Y-Axis Rotation with Differing Depth Estimation

rotation angles. KS constantly fails after about65°. PC&SH has a very slowly increasing level of error, remaining close to 10 pixels. The constant depth estimation methods result in a slight increase in a sharp error increase after about 65°of rotation.

7.8 Rotation of Feature Points with Differing Depth Estimation

Different depth estimation schemes begin to have effects for most systems only for large rotation of the feature points. As seen in Figure 10, IBVS shows little variation for each estimation method. However, both 2.5D and PC&SH have dramatic increases in pixel error after specific angles. This is most apparent in the case of 2.5D, which is able to zero the error even at extremely large angles when using true depth. KD experiences a dramatic rise in pixel error at about 35 degrees of rotation. Using a constant depth estimation delays this increase by about 5°.

8 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.

References

[1] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics*

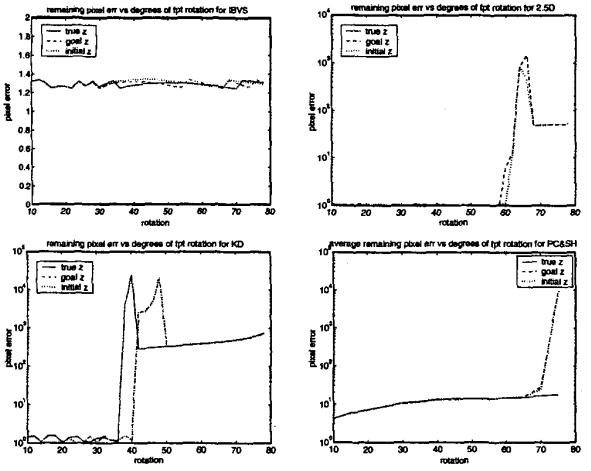

Figure 10: Remaining Pixel Error for Y-Axis Rotation with Differing Depth Estimation

and Automation, vol. 12, pp. 651–670, Oct. 1996.

[2] E. Malis, F. Chaumette, and S. Boudet, "2-1/2-d visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 238–250, Apr. 1999.

[3] K. Deguchi, "Optimal motion control for image-based visual servoing by decoupling translation and rotation," in *Proc. Int. Conf. Intelligent Robots and Systems*, pp. 705–711, Oct. 1998.

[4] P. I. Corke and S. A. Hutchinson, "A new partitioned approach to image-based visual servo control," in *Proc. 39th Conf. on Decision and Control*, pp. 2521–2526, Dec. 2000.

[5] P. Martinet, "Comparison of visual servoing techniques: Experimental results,"

[6] J. Aloimonos and D. P. Tsakiris, "On the mathematics of visual tracking," *Image and Vision Computing*, vol. 9, pp. 235–251, Aug. 1991.

[7] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*. Addison Wesley, 1993.

[8] O. Faugeras, *Three-Dimensional Computer Vision*. Cambridge, MA: MIT Press, 1993.

[9] O. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, no. 3, pp. 485–508, 1988.

[10] E. Malis and F. Chaumette, "2 1/2d visual servoing with respect to unknown objects through a new estimation scheme of camera displacement," *International Journal of Computer Vision*, vol. 37, no. 1, pp. 79–97, 2000.

[11] Z. Zhang and A. Hanson, "reconstruction based on homography mapping," 1996.

[12] 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.

[13] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The confluence of vision and control* (D. Kriegman, G. Hager, and S. Morse, eds.), vol. 237 of *Lecture Notes in Control and Information Sciences*, pp. 66–78, Springer-Verlag, 1998.

[14] P. I. Corke, "Robotics toolbox for MATLAB," *IEEE Robotics & Automation Magazine*, vol. 3, no. 1, pp. 24–32, 1996.