2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Uncalibrated Visual Servoing for Intuitive Human Guidance of Robots

Matthew Marshall[†*] Michael Matthews[†] Ai-Ping Hu[†] Gary McMurray[†] Harvey Lipkin[‡]
[†]Georgia Tech Research Institute
[‡]Georgia Institute of Technology
Atlanta, GA 30332 USA
[*]matthew.marshall@gtri.gatech.edu

***Abstract*— We propose a novel implementation of visual servoing whereby a human operator can guide a robot relative to the coordinate frame of an eye-in-hand camera. Among other applications, this can allow the operator to work in the image space of the eye-in-hand camera. This is achieved using a gamepad, a time-of-flight camera (an active sensor that creates depth data), and recursive least-squares update with Gauss-Newton control. Contributions of this paper include the use of a person to cause the control action in a visual-servoing system, and the introduction of uncalibrated position-based visual servoing. The system's efficacy is evaluated via trials involving human operators in different scenarios.**

I. INTRODUCTION

Visual servoing (VS) is a vision-guided robot control method that has been intently studied over the last two decades. The reader is referred to [1] and [2] (and the citations therein) for a recent overview. The present paper describes a novel application of visual servoing to human guidance of a robot arm. The new method uses a gamepad and an eye-in-hand camera display for navigating to an object in the robot's workspace. The method developed permits the human operator to command the robot in the camera's image space by using the gamepad to generate the desired pose (defined relative to the current location of the camera frame) that drives the visual servoing algorithm. In test trials, the implementation has been shown to provide more intuitive robot user operation and significantly faster task completion times.

The results reported here stem from recent work performed at the Georgia Tech Research Institute to provide a more natural user interface for operators of explosive ordinance disposal (EOD) robots [3]. The standard configuration for an EOD robot is an articulated arm mounted on top of a mobile platform; the robot is manually controlled by the operator via a remote pendant and visual displays showing the view of on-board camera's. Currently, a user operates an EOD robots via joint-based control, rather than directly controlling the end-effector motion in Cartesian space. This control suffers from the fact that a general commanded joint motion changes the image relayed by an eye-in-hand camera in a non-intuitive fashion since the motion transformation is governed by the robot's nonlinear forward kinematics. This impediment in EOD (and similar) robots has been noted by various researchers (refer, for example, to [4], [5], and [6]).

The result is that to make EOD robots easier/feasible to use, the operator will strive to maintain line-of-sight with the robot. By observing the robot directly in this way, the operator is able to develop a mental picture of the effect that actuating a given joint has on the robot motion relative to the world's Cartesian space. Removal of this line-of-sight requirement, in order to get the operator as far as possible from harm's way, was the driving motivator for the work presented here. From the point of view of a user tasked with guiding the robot end-effector motion while looking only at video from an eye-in-hand camera, it is most natural to control the motion in image space, rather than in joint space. For example, if it is desired to position the end-effector slightly to the left of an object appearing in the image (say, to try to peer around it), then an optimal user interface implements that motion by allowing the user to push a LEFT button. This has a clear advantage over joint-based control, a claim validated later in this paper using the experimental results from user trials.

II. BACKGROUND

One major aspect of accomplishing visual servoing is transforming camera data into variables that are meaningful to the robot, either in Cartesian space or joint space. Techniques in VS are commonly categorized into two groups, based on the chosen transformation. The first group is position-based visual servoing (PBVS), it treats the vision system as a 3-D sensor to determine the Cartesian coordinates of the features. The second group is image-based visual servoing (IBVS) which uses "features that are immediately available in the image data" [1]. There are advantages and disadvantages to each.

A primary benefit of employing PBVS is that it can provide optimized trajectories in Cartesian space. On the other hand, PBVS typically requires precise calibration. Here calibration deals with both the extrinsic (position and orientation) and intrinsic (lens and sensor) parameters of the camera, and the kinematic parameters of the robot. A meritorious aspect of IBVS is that it is more accurate in the presence of imprecisely estimated system parameters. A drawback to IBVS is that the robot trajectory in Cartesian space can be unpredictable [1].

Visual-servoing systems can also be categorized as either calibrated or uncalibrated. In a calibrated system, camera parameters and the kinematic model of the robot are both utilized to generate robot commands. Uncalibrated visual servoing is able to control the robot without knowledge of

978-1-4673-1405-3/12/$31.00 ©2012 IEEE

these parameters. In Marchand et al. [7], a type of hybrid VS controller is reported where robot and camera parameters are used but the robot itself lacks instrumentation, that is, proprioceptive sensors. The work presented here is also a hybrid, in that it is founded on uncalibrated (image-based) visual servoing, but uses a calibrated, time-of-flight 3-D camera. The result is that the kinematics of the system are not needed, but a calibrated sensor is employed. Thus, this paper presents an implementation of uncalibrated, position-based visual servoing. To the best of the authors' knowledge this is the first such system to be reported.

The reason for basing this work on uncalibrated VS is to address the many possible applications of VS technology to unstructured environments: those where calibration is not feasible. An example of this could be cameras mounted on mobile robots that are deployed to a scene and provide additional visual information for an EOD robot. It would not be reasonable to calibrate the extrinsic parameters of these mobile cameras relative to the EOD robot (and as stated above the kinematic parameters of these robots are typically not used).

Various methods for circumventing the calibration process have been presented in the literature. One example is the use of neural networks ([8] and [9]). These systems do not require prior knowledge of the hardware parameters, the controller learns the relationship between image data and robot coordinates via an extensive teaching phase. The Jacobian matrix for IBVS is a descriptor for this relationship. It is a composite of the robot Jacobian (relating movement in the robot joints to changes in its pose) and the image Jacobian (relating motion in Cartesian space to movement in the image plane). Another example of uncalibrated VS [10] defines state variables formed from elements of the Jacobian matrix and employs a Kalman-Bucy filter to estimate them on-line. Undesirably, large performance losses were noted in this type of controller as system noise increased [11]. In a different approach to the uncalibrated VS problem, the transformation from robot joint coordinates to image plane coordinates is estimated on-line via Broyden's update so changes to camera or robot parameters during operation can be handled without the need for system recalibration. This type of system is shown to be able to track moving targets in Piepmeier et al. [12], and it forms the foundation for the algorithms presented here.

Traditionally, visual servoing acts on an error that is defined as the difference between the current coordinates of features and their desired coordinates:

$$f = y - y^* \tag{1}$$

where y represents the location of the robot end-effector and y^* its desired location, for example, that of a target being tracked. For IBVS these coordinates are given in pixels of the image plane. For PBVS they are Cartesian coordinates. Previously published instances of VS determine the control action based on this error function toward the goal of minimizing error.

In the application of VS described herein, there is no explicit set of desired features from which to compute an error like in (1). Instead, the control action is based on inputs from a human operator, issued via a hand-held gamepad. This distinction is expounded upon in the following section.

III. Method

The aim of this section is to describe the technique used here for human-guided visual servoing. The control algorithm is given, in general terms that can apply to either traditional VS or the implementation in this work, and then the distinction is drawn between the two approaches. Concluding this section is a description of the perception component of this work.

A. Control algorithm

The visual servoing system model is assumed to be linear and able to be expressed as

$$\delta y \approx J\,\delta\theta$$

where the output y is some measurable value and θ describes the system. The model used for the control algorithm is

$$h_y = \hat{J} h_\theta$$

where, at the k-th iteration, $h_{y_k} \equiv y_k - y_{k-1}$ and $h_{\theta_k} \equiv \theta_k - \theta_{k-1}$. The term $\hat{J}$ denotes an estimate of J.

After each iteration and subsequent observation of the system state θ and output y, the Jacobian model is updated according to the following, which is based on work presented in [12].

$$\hat{J}_k = \hat{J}_{k-1} + \frac{\left(h_{y_k} - \hat{J}_{k-1} h_{\theta_k}\right) h_{\theta_k}^{\mathsf{T}} P_{k-1}}{\lambda + h_{\theta_k}^{\mathsf{T}} P_{k-1} h_{\theta_k}} \tag{2}$$

$$P_k = \frac{1}{\lambda}\left(P_{k-1} - \frac{P_{k-1} h_{\theta_k} h_{\theta_k}^{\mathsf{T}} P_{k-1}}{\lambda + h_{\theta_k}^{\mathsf{T}} P_{k-1} h_{\theta_k}}\right)$$

where P may be initialized as the identity, and the term λ is called the "forgetting factor." This is a bit of a misnomer since the Jacobian update reacts to new data more slowly as λ increases, thus the system forgets old information more quickly with smaller λ [13].

Given the new observations, the control action is governed by the Gauss-Newton method as

$$\theta_{(k+2)_c} = \theta_{(k+1)^-} + \hat{J}_k^{+} h_{y_{d_{(k+1)^-}}} \tag{3}$$

where $\hat{J}^+$ is the pseudo-inverse of $\hat{J}$, h_{y_d} is the desired output change, the minus sign on $(k+1)^-$ indicates values at a moment just prior to $k+1$, and the subscript c indicates that this will not necessarily be the joint position at $k+2$ but rather it is the commanded value. The possible difference is due to the fact that the robot is operating in velocity mode and the control period is dependent on the (variable) image processing time.

B. Difference between traditional VS and gamepad-driven

As described in §II, in the context of traditional PBVS the system output y is given in Cartesian coordinates and θ is robot joint angles. Conventional visual servoing would have the desired output change in (3) as $h_{y_{d(k+1)}} = -f_k$, where f is the pose based error from (1), thus commanding the system toward zero error in the image plane. However, for the implementation presented here, the user commands the robot relative to the camera image by specifying motion in six degrees-of-freedom (three translational and three rotational).

For example, there is a joystick direction for camera movement to the right that will correspond to a translation command in the positive x direction of the camera's frame, with similar commands along/about the other five camera degrees of freedom (DOF). The 6×1 vector describing this desired motion is denoted g. The visual servoing algorithm resolves the user-commanded motion into the proper joint movements. It follows that here $h_{y_{d_k}} = g_k$, where g_k is the current operator input to the gamepad.

C. Perception

To control in all six camera DOF as described above, the vision system must solve for the Cartesian offset of the camera (its relative pose) from one image to another, h_{p_k}. The 3-D time-of-flight (TOF) camera, which outputs a 3-D point location for each pixel, allows for a simple transformation solution using standard computer vision methods.

The TOF camera yields intensity, confidence, and 3-D images. The intensity image is similar to a standard grayscale image. 3-D positions of each pixel are returned in the 3-D image. The confidence image is a grayscale image that indicates the amount of error in the 3-D solution for each pixel, and is critical for accurate data analysis. Distinct feature points, or keypoints, are found in the images, which are then matched from one image to the next. The 3-D data at each point is then subsequently used to compute a transformation solution. This process is illustrated in Fig. 1 and by Algorithm 1, which relies on many function calls from [14].

After the images are obtained, the confidence image is thresholded and eroded, and used as a mask for detecting feature points with reliable 3D data. Feature points are detected in the 2-D grayscale image using the FAST feature detector [15], and descriptions of these keypoints are found with the SURF descriptor [16]. Next, these 2-D keypoints are matched with keypoints found in the previous image using K-Nearest-Neighbors on the high dimensional space of the descriptors. For each current keypoint the nearest k previous keypoints are located, and all become initial matches. These are then filtered to only the single best cross correlated matches, and to those satisfying the epipolar constraint (a fundamental matrix solution with RANSAC). Finally, using the 3D coordinates of the current keypoint matches, the 3-D transformation solution is computed using a 3D-3D transformation solver [17] (using RANSAC for further filtering). This final 3-D transformation, consisting of rotations (roll, pitch, yaw) and translations (x, y, z), of the camera with respect to the previous camera pose, is the feedback input into the model update portion of the VS algorithm as $h_{y_k} = h_{p_k}$. At the start of each cycle of the VS algorithm, the camera is triggered and this method is run.

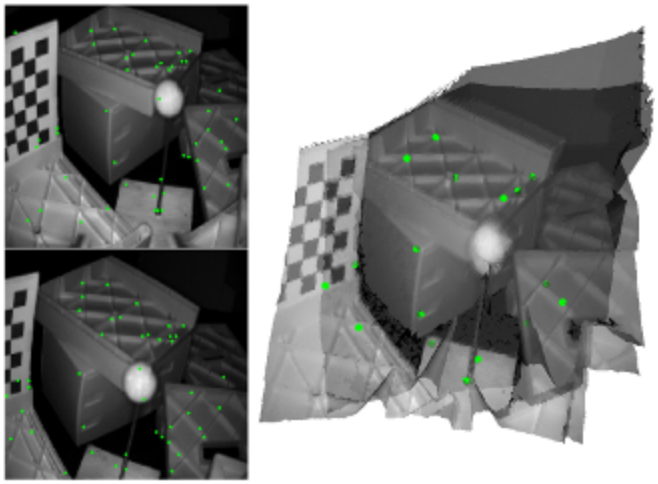

Fig. 1. Relative Camera Pose Solution: left, previous and current intensity images in 2-D, matching keypoints highlighted in green; right, previous and current 3-D meshes overlaid illustrating registering of the image data.

IV. EXPERIMENTAL RESULTS

A. Setup

In the current implementation, a six degree-of-freedom articulated robot arm (shown in Fig. 2) is used as the testbed. A 3-D time-of-flight camera is affixed to the end of the robot arm (i.e., eye-in-hand). The robot used for experiments is the KR 15 SL manufactured by Kuka Robotics. The 3-D TOF camera used is the SwissRanger SR4000.

This camera uses active-pulsed infrared lighting and multiple frames of the returned light, taken at different times, to solve for the depth at each pixel. Its optics are pre-calibrated by the manufacturer to accurately convert the depth data into a 3-D position image. The camera resolution is 176×144 pixels. For image analysis this provides roughly 300 feature points, yielding 50–200 matches per iteration, and takes 50–70 ms processing time. Analysis of image data takes place on a Windows 7 PC with an Intel Core i7-870 processor and 8 GB of RAM. This PC communicates with the robot joint-level controller using a DeviceNet connection, which updates every 12 ms.

The gamepad is a Sony Playstation 3 DualShock controller, with floating point axis feedback allowing for smooth user control. Motion-in-Joy drivers are used to connect it as a Windows joystick. NI LabVIEW reads the current gamepad state, whose values are then sent to the VS controller over TCP.

Before starting to servo the robot an initial estimate of the Jacobian is made by jogging the joints individually and recording the resulting measurement as a column of $\hat{J}$. This is not a necessary step, but done to minimize the learning time, in the interest of safety. Also the gamepad position and

begin

(1) Get images: intensity, confidence, and 3-D.

(2) Threshold and Erode TOF Confidence Image to Mask Keypoint Detection.
KeypointMask = cvThreshold(ConfImage, Threshold);
KeypointMask = cvErode(KeypointMask, ErodeSize);

(3) Detect Feature Points, Get Descriptors, and 3D Positions.
CurrKeypoints = cvFASTFeatureDetector(GrayImage, Mask);
CurrKeypointsDesc = cvSurfFeatureDesciptor(GrayImage, CurrKeypoints);
CurrKeypoints3D = Get3DPositions(CurrKeypoints, Image3D);

(4) Perform the 2D Keypoint Matching.
Matches = cvknnMatch(CurrKeypointsDesc, PrevKeypointsDesc, KNN);
Matches = CrossCheckMatch(Matches);
Matches = cvFundamentalMatrixMatch(Matches, CurrKeypoints, PrevKeypoints, RANSAC);

(5) Perform 3D Keypoint Matching for Final Pose Solution.
RelativePose = ThreeDMatchSolver(Matches, CurrKeypoints3D, PrevKeypoints3D, RANSAC);

(6) Swap Curr/Prev.

end

Algorithm 1: Perception algorithm pseudo-code.

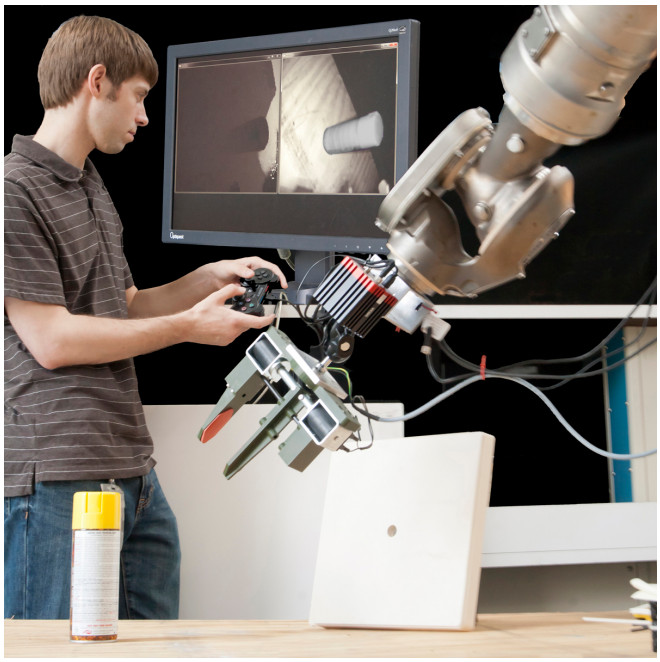

Fig. 2. Experimental testbed consisting of a six degree-of-freedom articulated arm robot controlled via a hand-held gamepad. Computer monitor shown displaying, from left to right, 3-D and intensity images.

joint angles are read and stored as g^- and θ^- respectively. This constitutes the system description at $k = 0$. The initial action, θ_{1_c}, is computed using these three values and equation (3).

To begin a general iteration, the controller first issues a command for the robot motion. As stated before the robot is operating in velocity mode so this is a motion in the direction of θ_c. The perception subsystem (described in Section III-C) is then immediately triggered. The joint angles θ_k are read and the controller awaits the measurement $h_{y_k} = h_{p_k}$, the measured relative pose of the camera from $k-1$ to k. Once this is received, the Jacobian estimate is updated according to (2). Next, the joint angles and gamepad position are re-read, as $\theta_{(k+1)^-}$ and $h_{y_{d_{(k+1)^-}}} = g_{(k+1)^-}$ respectively (again, the minus sign indicates that they represent the values at a moment just prior to the robot reaching $k+1$). The concluding task for an iteration is to compute the next desired joint position, $\theta_{(k+2)_c}$, using (3).

B. Assigned manipulation task

To demonstrate the effectiveness of using visual servoing for teleoperation, trials are performed with human operation of the robot performing an object manipulation task, with eleven members of our research team as volunteers. The visual servoing method is compared to traditional joint-based guidance for two different scenarios: 1) line of sight, and 2) camera view. Thus each volunteer performs four tests. A brief explanation of the four cases is provided for clarity.

- Line of sight, VS mode: The operator only has line of sight to the robot. The buttons on the gamepad are mapped to the Cartesian frame of the camera
- Line of sight, joint mode: The operator only has line of sight to the robot. The buttons on the gamepad are mapped to the robot axes.
- Camera view, VS mode: The operator sees only the monitor displaying the intensity image from the eye-in-hand camera. The buttons on the gamepad are mapped to the Cartesian frame of the camera
- Camera view, joint mode: The operator sees only the monitor displaying the intensity image from the eye-in-hand camera. The buttons on the gamepad are mapped to the robot axes.

In each case the operator servos to and grasps (using a custom end-of-arm gripper, see Fig. 2) a two-inch diameter ball. The gripper is able to open to a width of two-and-a-

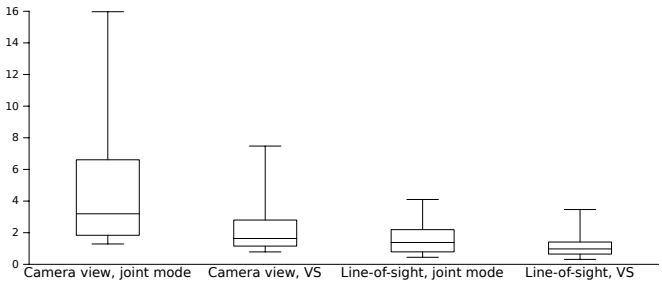

Fig. 3. Time to complete task (minutes), eleven trials.

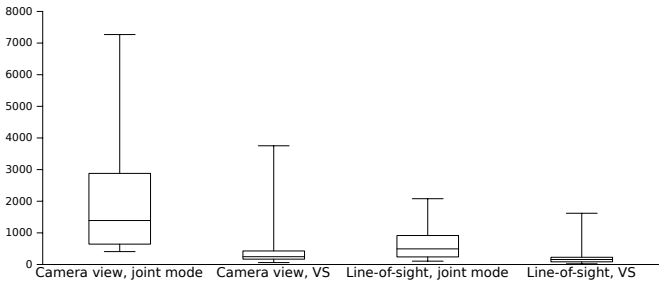

Fig. 4. Number of times user input changed direction, eleven trials.

half inches. The robot and ball start in the same positions for each operator. These positions are such that the ball is in the camera's field of view at the start of the task and are about one meter apart. A trial was deemed complete when the user had closed the gripper on the ball.

C. Results of human trials

All participants completed the task with both control modes in both scenarios. Analysis of the time required to complete the task in the four different situations shows that when using VS in the line-of-sight scenario, operator speed increases by an average of 15% compared to using joint mode. When using VS in the camera-view situation, the operator completed the task 227% faster than in joint mode. The data regarding time to complete the task is summarized in Fig. 3. Box plots depict smallest observation, lower quartile, median, upper quartile, and the largest observation.

Besides time-to-complete, another metric regarding ease of use for the operator is a count of the times the user input (gamepad position) changes direction during the task. Roughly speaking, that means an instance when the operator moved from pressing one button to another (besides buttons, analog joysticks were available so this cannot be applied too strictly). As shown in Fig. 4, there is, on average, a twofold decrease in the number of direction changes for the line-of-sight scenario, and a fourfold decrease for the camera-view scenario.

For both modes of operation (joint and VS) in the camera-view scenario, information regarding the 3-D path taken by the robot gripper for a representative operator is shown in Figures 5, 6, and 7. In Fig. 5 the X, Y, and Z coordinates of the gripper in the world Cartesian system are plotted vs time. Fig. 6 traces this path in a 3-D plot. The distance between the gripper and the ball (the target) is plotted versus time in

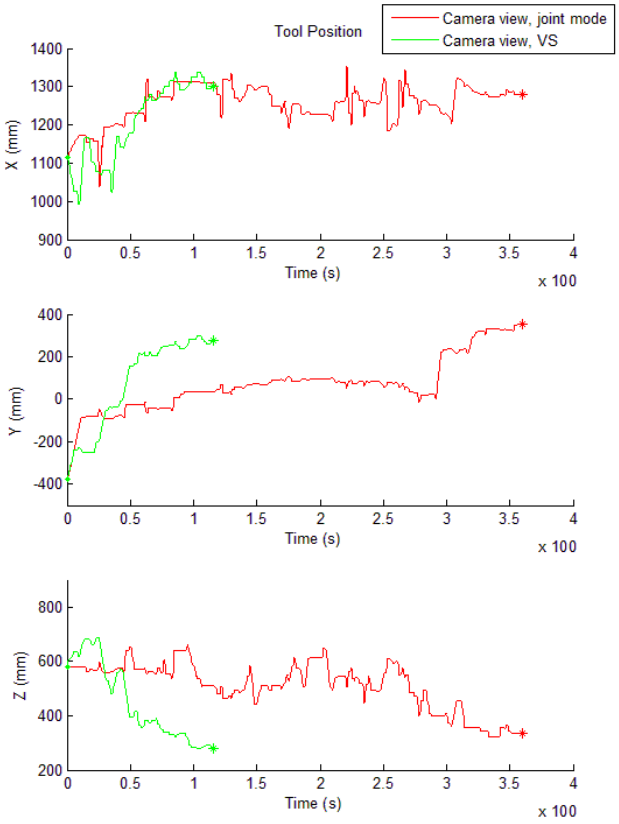

Fig. 5. Gripper position relative to the world's Cartesian space w.r.t. time.

Fig. 7. This distance is normalized with respect to its starting value. These figures together serve to demonstrate that the operator is able to guide the robot to the goal more directly when using VS than when using joint mode.

V. DISCUSSION AND FUTURE WORK

This paper presents a novel control method based on uncalibrated visual servoing for teleoperation of a robot. The described method uses commands issued by the operator via the buttons and joysticks on a hand-held gamepad; these manual inputs drive the robot joint updates. This differs from a standard visual servoing implementation that would use image plane errors to update the robot joints. The operator in our experimental trials is looking at a computer monitor that displays the view from a camera affixed to the robot end-effector.

We have reported the results of human trials in operating a six degree-of-freedom articulated arm robot performing a simple manipulation task. Four different scenarios were tested: our modified visual servoing control with and without line-of-sight and joint-based control with and without line-of-sight. A significant improvement was observed for the visual servoing trials. Operators were consistently able to complete a manipulation task faster and with fewer commands, and also the path of the gripper was more direct.

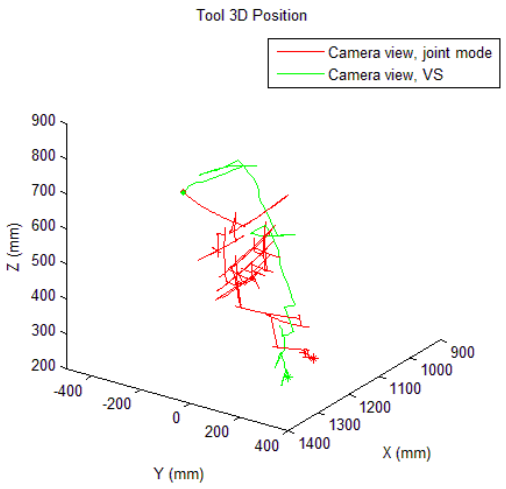

Fig. 6. Gripper position relative to world Cartesian space, 3-D plot.

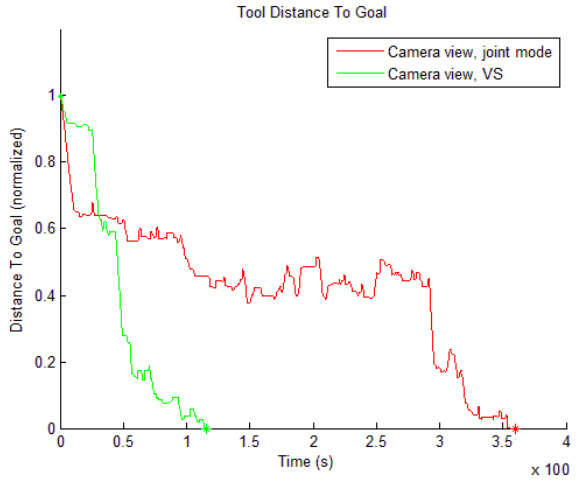

Fig. 7. Change in distance between gripper and ball as a function of time.

This 6-DOF Cartesian control can be implemented with a stereo camera, a 3-D camera, or a 2-D camera with a 3-D pose solution (e.g., using structure from motion techniques). In addition, the work presented here need not be limited to Cartesian control with a 3-D sensor. It is a flexible platform in that it allows a user to guide a robot relative to whatever the frame of the measurements is. For example, in earlier, unreported work, we provided 3-DOF control relative to the image plane using a standard 2-D eye-in-hand camera. It need not even be limited to eye-in-hand camera scenarios. The only requirement is that the user interface and vision system are capable of control and feedback of the desired coordinates. Another possibility is control of the end-effector from an external eye-to-hand camera's viewpoint. Future work will seek to explore these alternatives to explore potential benefits.

REFERENCES

[1] F. Chaumette and S. Hutchinson, "Visual servo control, part I: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, dec 2006.

[2] ——, "Visual servo control, part II: Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, mar 2007.

[3] M. Marshall, M. Matthews, A.-P. Hu, H. Lipkin, and G. McMurray, "Using visual servoing to improve eod robot control," *AUVSI Unmanned Systems North America*, aug 2011.

[4] D. Handelman, G. Franken, and H. Komsuoglu, "Agile and dexterous robot for inspection and eod operations," in *Proc. SPIE*, vol. 7692, USA, 2010, p. 769211 (11 pp.).

[5] L. Cai, F. Chang, S. Li, and X. Zhang, "Control system of the explosive ordnance disposal robot based on active eye-to-hand binocular vision," in *Proc. AICI'11 - Volume Part III*, 2011.

[6] D. Anderson, T. Howard, D. Apfelbaum, H. Herman, and A. Kelly, "Coordinated control and range imaging for mobile manipulation," in *Proc. ISER '08*, 2008.

[7] E. Marchand, F. Chaumette, F. Spindler, and M. Perrier, "Controlling an uninstrumented manipulator by visual servoing," *The International Journal of Robotics Research*, vol. 21, pp. 635–647, jul 2002.

[8] H. Hashimoto, T. Kubota, M. Kudou, and F. Harashima, "Self-organizing visual servo system based on neural networks," *Control Systems Magazine, IEEE*, vol. 12, no. 2, pp. 31–36, Apr 1992.

[9] J. Carusone and G. D'Eleuterio, "The feature cmac: a neural-network-based vision system for robotic control," *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 2959–2964, 16-20 1998.

[10] J. Qian and J. Su, "Online estimation of image Jacobian matrix by Kalman-Bucy filter for uncalibrated stereo vision feedback," *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 562–567, 2002.

[11] J. Sebastin, L. Pari, L. Angel, and A. Traslosheros, "Uncalibrated visual servoing using the fundamental matrix," *Robotics and Autonomous Systems*, vol. 57, no. 1, pp. 1–10, 2009.

[12] J. Piepmeier, G. McMurray, and H. Lipkin, "Uncalibrated dynamic visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 143–147, feb. 2004.

[13] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1996.

[14] (2011, July) Open source computer vision library. http://opencv.willowgarage.com/wiki/.

[15] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, May 2006.

[16] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features - demonstration," *Computer Vision ECCV*, 2006.

[17] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, pp. 698–700, September 1987.