Kernel-Based Visual Servoing

Vinutha Kallem Maneesh Dewan John P. Swensen Gregory D. Hager Noah J. Cowan

***Abstract*— Traditionally, visual servoing is separated into tracking and control subsystems. This separation, though convenient, is not necessarily well justified. When tracking and control strategies are designed independently, it is not clear how to optimize them to achieve a certain task. In this work, we propose a framework in which spatial sampling kernels – borrowed from the tracking and registration literature – are used to design feedback controllers for visual servoing. The use of spatial sampling kernels provides natural hooks for Lyapunov theory, thus unifying tracking and control and providing a framework for optimizing a particular servoing task.**

As a first step, we develop kernel-based visual servos for a subset of relative motions between camera and target scene. The subset of motions we consider are 2D translation, scale, and roll of the target relative to the camera. Our approach provides formal guarantees on the convergence/stability of visual servoing algorithms under putatively generic conditions.

I. Introduction

Visual servoing (VS) entails moving either a camera or the camera's visual target such that the image of the target asymptotically converges to a desired image. Traditionally, visual servoing assumes that there is an image processing unit tracking the feature points in the image. This information is used by the visual servoing controller to drive the feature trajectories to some desired constellation. This technique is sometimes convenient because the problem can be decoupled into "feature tracking" and "control" sub-problems.

However, the classical division between vision and control, may be ill-equipped for the reality of a complex, unstructured world. By decoupling vision and control, the vision design problem includes little or no direct information related to the underlying control task that it serves, rendering it difficult or impossible to make intelligent choices as to what to observe, or how to observe it. Conversely, control cannot adapt to a changing visual environment. Thus, neither the vision nor controller design can be tuned (much less optimized) for the properties of its counterpart. Consequently, visual tracking algorithms tend to be hand-tailored (often along with the environment) to provide adequate information needed for a specific control algorithm.

In the present work, we propose a method to perform visual servoing without separating the tracking and control tasks. We build on spatial kernel-based tracking algorithms [3], [6], [8], [9] to design feedback controllers and use Lyapunov stability theory to show stability of the same. We present preliminary results on the implementation of this kernel-based visual servoing approach on a class of simple motions that can be extended to more complex motions.

V. Kallem, J. P. Swensen, and N. J. Cowan are with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: vkallem@jhu.edu).

M. Dewan and G. D. Hager are with the Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218 USA.

Related Work

a) Visual Servoing: Visual servoing traditionally requires that a vision subsystem provide feature point correspondences between current and desired views as the robot or target moves. These geometric measurements are then used directly for vision-based control. While nearly all of these methods rely on a vision front-end to provide geometric visual features, researchers have begun to tackle the sensing problem more directly. Rivlin and Rotstein [13] investigate the tradeoffs between window size and resolution, analogous to adjusting the kernel width, as we propose. Kragic et al. [10], [11] directly address the issue of tracking for control. They use appearance-based and geometric models of a scene to match the current view with an expected projection that is computed from an estimate of the current pose. Using a detect, match and update scheme, their tracker feeds into a very classical 2D or 3D visual servo controller. Deguchi [5] encodes the entire target image using the Karhunen-Loève expansion. The coefficients in this expansion are related to the underlying object motion, leading to a form similar to the image-Jacobian for feature-based methods. Tahri and Chaumette [14] perform visual servoing via moments which is related to the present paper as described below.

b) Kernel-Based Tracking Methods: Kernel-based tracking methods have recently gained popularity primarily due to their broad range of convergence and their robustness to unmodelled spatial deformations. In these methods a kernel $K : \mathbb{R}^2 \to \mathbb{R}$ is a real-valued piecewise continuous function defined on the location space of the image. The kernel acts as a sampling function that weighs the feature space of the image. These weighted values are usually summed over all locations to create a measurement. Tracking then reduces to optimally shifting the location of the kernel, the objective function being a metric between the kernel-based measurement at the current location and a fixed reference measurement [3], [8], [9]. Interestingly, spatial kernels can be viewed as moment-generators, an interpretation that may connect this paper to prior work [14]: since kernels are usually polynomial functions of image location, kernel measurement can be thought of as a collection of moments of feature space around the kernel center.

One can extend the kernel-based tracking framework by defining multiple kernels and/or multiple image projections. Comaniciu et. al [3] define multiple image projections through a binning function into mutually orthogonal binary

projections. Their work was extended by Hager et al. [9] to multiple kernels for tracking complex motions. The idea of using motion specific kernels invariant to other motion parameters was introduced. The idea of multiple kernels was further extended to articulated motions in [8]. In all these approaches, the kernel parameters are chosen in an ad-hoc manner often leading to sub-optimal performance. More recently, Dewan and Hager [6] introduced a scheme for optimizing the kernel parameters to the target being tracked.

Our approach differs fundamentally from the aforementioned approaches in that we place tracking and control in the same framework: spatial sampling kernels unify geometry and sensing, which will enable us to develop an integrated strategy for tracking and control. We note that our work is preliminary, and as such we have not directly compared it with other work; we leave such comparisons for future work.

II. Kernel-Based Visual Servoing (KBVS)

For the present exposition, we make a number of simplifying assumptions. First, we consider the "eye-in-hand" configuration in which the camera is mounted on the robot end effector, and the target is stationary. Further, we assume a kinematic motion model for the robot, whose control inputs are its joint velocities. We treat image pixels as continuous variables over all of $\mathbb{R}^2$ measured in continuous time, rather than discrete variables over a finite image measured in discrete time (in practice these assumptions are clearly violated; see experimental results, section III). In all the analysis below, the image or its transformations are treated as signals that are directly measured.

Given a signal, $s(\boldsymbol{w}, t)$ (such as the intensity of the image at each pixel as time progresses), the kernel-projected value of the signal at time t may be defined as the scalar

$$\xi(t) = \int_{\mathcal{I}} K(\boldsymbol{w}) s(\boldsymbol{w}, t) d\boldsymbol{w}, \tag{1}$$

where $\boldsymbol{w} \in \mathcal{I} = \mathbb{R}^2$ is the image spatial indexing variable. As the camera moves relative to the target, the signal, $s(\cdot, t)$, changes, thus affecting the kernel-projected value. At the goal, let the signal be $s_0 = s(\cdot, 0)$ and the kernel-projected measurement at the goal be denoted by $\xi_0 = \xi(0)$. The aim of KBVS is to drive the robot/camera to goal configuration by driving $\xi(t) \to \xi_0$.

Below, we develop KBVS controllers for a subset of generic camera motions in $\mathrm{SE}(3)$. We first describe our overall method for a 2D translation (x-y) parallel to the optical plane and then for translation along the optical axis (z), and roll about the camera optical axis (θ).

A. 2D Translation

Consider a robot with a camera mounted on it as in Figure 1. Image intensity at each pixel is taken as the signal, ignoring illumination changes. The pixel location is given by $\boldsymbol{w} \in \mathbb{R}^2$. Let the kernel projection of the image at the goal be ξ_0 at the position $x = 0$ and $y = 0$ (without loss of generality). Our goal is to determine a control input that will drive the kernel-projected measurement to ξ_0, thus driving $[x(t), y(t)] \to 0$.

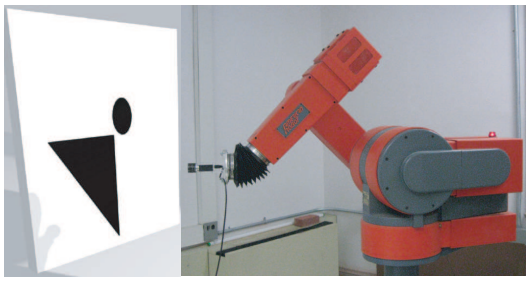

Fig. 1. Experimental configuration.

Let the configuration of the robot be denoted by $q = [x, y]^T \in \mathbb{R}^2$. Assume that the camera moves parallel to the optical axis according to the simplified dynamics[1]

$$\dot{\boldsymbol{q}} = \boldsymbol{u}, \tag{2}$$

where $\boldsymbol{u} \in \mathbb{R}^2$ is the robot control input. For the remainder of the paper, we assume the signal only depends on time via the camera motion, which in this case implies (in an abuse of notation) $s(\boldsymbol{w}, t) = s(\boldsymbol{w}, \boldsymbol{q}(t))$.

For simplicity of presentation, we assume that the scene is a unit distance away from the image plane, so that $s(\boldsymbol{w}, \boldsymbol{q}(t)) = s_0(\boldsymbol{w} - \boldsymbol{q}(t))$. Through a change of variables, $\bar{\boldsymbol{w}} = \boldsymbol{w} + \boldsymbol{q}$, and recalling that $\mathcal{I} = \mathbb{R}^2$, the kernel-projected measurement ξ can be rewritten as

$$\xi = \int_{\mathcal{I}} K(\boldsymbol{w}) s_0(\boldsymbol{w} - \boldsymbol{q}) d\boldsymbol{w} = \int_{\mathcal{I}} K(\bar{\boldsymbol{w}} + \boldsymbol{q}) s_0(\bar{\boldsymbol{w}}) d\bar{\boldsymbol{w}}. \tag{3}$$

From (3), observe that even when the images or the signal are discontinuous and hence not differentiable, the kernel-projected measurement is analytically differentiable as long as the kernel is smooth. As we show below, we exploit the differentiability of $\xi(t)$ in the design of KBVS controllers.

Consider a Lyapunov function candidate $V = \frac{1}{2}(\xi - \xi_0)^2$. Applying the chain rule, we have

$$\begin{aligned}
\dot{V} =& (\xi - \xi_0) \frac{\partial \xi}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} \\
=& (\xi - \xi_0) \Big[\int_{\mathcal{I}} K'(\bar{\boldsymbol{w}} + \boldsymbol{q}) s_0(\bar{\boldsymbol{w}}) d\bar{\boldsymbol{w}} \Big] \dot{\boldsymbol{q}} \\
=& (\xi - \xi_0) \Big[\int_{\mathcal{I}} K'(\boldsymbol{w}) s_0(\boldsymbol{w} - \boldsymbol{q}) d\boldsymbol{w} \Big] \boldsymbol{u},
\end{aligned}$$

where $K'(\boldsymbol{w}) = \frac{\partial K(\boldsymbol{w})}{\partial \boldsymbol{w}}$. Note that in the last step, we revert the coordinates back to $\boldsymbol{w}$. Now, choose the input, u, as

$$\boldsymbol{u} = -(\xi - \xi_0) \int_{\mathcal{I}} \nabla K(\boldsymbol{w}) s(\boldsymbol{w}, \boldsymbol{q}) d\boldsymbol{w}, \tag{4}$$

where $\nabla K = (\frac{\partial K}{\partial \boldsymbol{w}})^T \in \mathbb{R}^2$. This requires only the current signal projection, ξ, the signal projection at the goal, ξ_0, the kernel function derivative, K', and the current signal, $s(w, t)$,

[1] We believe that lifting these control laws to second order mechanical systems should be straight forward.

which depends on t only through $\boldsymbol{q}(t)$ (see above). With this choice $\dot{V}$ becomes

$$\dot{V} = -(\xi - \xi_0)^2 \left\| \int_{\mathcal{I}} \nabla K(\boldsymbol{w}) s(\boldsymbol{w}, \boldsymbol{q}) d\boldsymbol{w} \right\|^2 .$$

Assuming the candidate Lyapunov function, V, is positive definite in the configuration variable, then $\dot{V}$ is negative semi-definite. The assumption that $V > 0$ admittedly depends on the signal and kernel properties, although it appears from our experiments (Section III) to be a locally valid generic assumption and, in any case, can be numerically tested and optimized [6]. This choice of input guarantees stability (in the Lyapunov sense) of the controller with mild assumptions on the image and kernel – KBVS is appealing exactly for this reason.

For practical applications, it is crucial to obtain at least local *asymptotic* stability. If the kernel-image pair is such that in a neighborhood around the goal, $\int_{\mathcal{I}} \nabla K(\boldsymbol{w}) s(\boldsymbol{w}, \boldsymbol{q}) d\boldsymbol{w} \neq 0$ (again, which appears to be true generically), local asymptotic stability is guaranteed. This quantity becoming zero is analogous to image error lying in the null space of the Jacobian in tracking literature [1]. For good practical performance of the controller, the Lyapunov function in the configuration space of the robot should be quadratic near the goal, the Hessian at the goal should have positive eigenvalues, and condition number as close to one as possible. This provides us with an objective function that likely can be optimized for larger regions of attraction and better performance, which we leave for future work.

Similar to the ideas presented in [6], an alternate way of doing 2D translation is to decouple it into two 1D translations using two independent x and y directional kernels. Each kernel is invariant to the motion in the other direction, thereby providing independent controllers. For example, a kernel oriented in the x direction can be formed by stacking a gaussian kernel along every pixel in the y direction. As discussed in [6], we also found that using the two independent kernels provides better results than using a single kernel. The experiments presented for the 2D translation case in Section III use the two-kernel approach.

B. Translation along optical axis

Cideciyan [2] uses a spatial Fourier transform (FT) of images for tracking and registration to decouple translation and scaling. We seek to capitalize on this invariance of the magnitude of FT to translation to develop controllers for depth and rotation that can integrated with the previously developed 2D controllers in the x-y plane. As a first step, we consider motions in depth only.

Here, we consider motions of a camera along its optical axis. Even though this corresponds to a translation as in the previous two cases, there is a fundamental difference between the two: 2D x-y translations simply translate the image, while motions in depth inversely scale the image. Thus, we seek an appropriately transformed signal and control strategy. Specifically, we use the magnitude of the FT of the image as the signal.

Let I_0 denote the image at the goal, and F_0 the magnitude of its spatial FT. We assume that the goal corresponds to unity depth (without any loss of generality). Let the inertial world reference frame be such that the z-axis is parallel to the camera's optical axis. In this frame, the camera is moving along the z-axis according to

$$\dot{z}(t) = u, \tag{5}$$

with goal $z_0 = 1$. At any generic position of the camera, the image I is a scaled version of I_0, *i.e.*

$$I(\boldsymbol{w}, z) = I_0(\boldsymbol{w}/z).$$

One can show that the magnitudes of the spatial FT of these images (F and F_0 respectively) are related by

$$F(\boldsymbol{v}, z) = z^2 F_0(z\boldsymbol{v}),\ \boldsymbol{v} \in \mathbb{R}^2.$$

We define the kernel-projected measurement as

$$\xi = \int_{\mathcal{I}} K(\boldsymbol{v}) F(\boldsymbol{v}, z) d\boldsymbol{v} = \int_{\mathcal{I}} K(\bar{\boldsymbol{v}}/z) F_0(\bar{\boldsymbol{v}}) d\bar{\boldsymbol{v}}, \tag{6}$$

where $\bar{\boldsymbol{v}} = z\boldsymbol{v}$. At the goal we have $\xi_0 = \int_{\mathcal{I}} K(\boldsymbol{v}) F_0(\boldsymbol{v}) d\boldsymbol{v}$. Our aim is to drive the robot to $z = 1$ by driving $\xi(t) \to \xi_0$. Consider a Lyapunov function candidate: $V = \frac{1}{2}(\xi - \xi_0)^2$ and choosing the input as

$$u = (\xi - \xi_0) \int_{\mathcal{I}} K'(\boldsymbol{v}) \boldsymbol{v} F(\boldsymbol{v}, z) d\boldsymbol{v}, \tag{7}$$

then $\dot{V} = -\frac{1}{z}(\xi - \xi_0)^2 \left\| \int_{\mathcal{I}} K'(\boldsymbol{v}) \boldsymbol{v} F(\boldsymbol{v}, z) d\boldsymbol{v} \right\|^2$. If $z > 0$, $\dot{V}$ is negative semi-definite, which is a realistic assumption for objects seen by the camera.

C. Rotation about the optical axis

In this section we develop KBVS for rotation of the camera relative to the target about its optical axis. Let the robot dynamics be

$$\dot{\theta} = u, \tag{8}$$

where u is the control input. As in the case of scaling, we use the magnitude of the spatial FT of the image as the signal. Let I_0 and F_0 denote the image and signal at the goal, where $\theta = 0$ (without any loss of generality). At any generic roll position of the camera, the image I is a rotated version of I_0:

$$I(\boldsymbol{w}, \theta) = I_0(R_\theta \boldsymbol{w}), \text{where } R_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \in \mathrm{SO}(2).$$

The magnitudes of the spatial FT of these images are related by

$$F(\boldsymbol{v}, \theta) = F_0(R_\theta \boldsymbol{v}),\ \boldsymbol{v} \in \mathbb{R}^2. \tag{9}$$

We define the kernel-projected measurement as

$$\xi = \int_{\mathcal{I}} K(\boldsymbol{v}) F(\boldsymbol{v}, \theta) d\boldsymbol{v} = \int_{\mathcal{I}} K(R_\theta^T \bar{\boldsymbol{v}}) F_0(\bar{\boldsymbol{v}}) d\bar{\boldsymbol{v}}, \tag{10}$$

where $\bar{\boldsymbol{v}} = R_\theta \boldsymbol{v}$. At the goal, the kernel-projected measurement is $\xi_0 = \int_{\mathcal{I}} K(\boldsymbol{v}) F_0(\boldsymbol{v}) d\boldsymbol{v}$. As before, our aim is to drive the robot to $\theta = 0$ by driving $\xi(t) \to \xi_0$. Consider a

Lyapunov function candidate: $V = \frac{1}{2}(\xi - \xi_0)^2$. Choose the control input as

$$u = -(\xi - \xi_0) \int_{\mathcal{I}} K'(\boldsymbol{v}) J\boldsymbol{v} F(\boldsymbol{v}, \theta) d\boldsymbol{v}, \qquad (11)$$

where $J = R_{-\frac{\pi}{2}}$. With this choice of u, $\dot{V} = -(\xi - \xi_0)^2 \left\| \int_{\mathcal{I}} K'(\boldsymbol{v}) J\boldsymbol{v} F(\boldsymbol{v}, \theta) d\boldsymbol{v} \right\|^2$, which is negative semi-definite.

D. Extensions to SE(2) + Depth Motions

In the above controllers, we used the image as the signal for the x-y translations and the magnitude of the FT of the image as the signal for depth and roll. As discussed before, the FT of the image removes any translation effects while controlling depth and roll. For 3D translational control, one can execute the depth controller first, since it is invariant to translation, and then run the 2D x-y controller. Similarly, to control all of SE(2) (identified with x, y, and roll), one can control for roll first, since it is again invariant to translation, and then control in the 2D plane. Furthermore, all four degrees of freedom (x, y, z, and roll) can be controlled in a similar manner. We have yet to verify these coupled controllers experimentally.

III. Experimental Results

In the design of the our above controllers, we made several simplifying assumptions which deserve attention for use in practical applications on real images. The issue of discrete spatial sampling is addressed by pre-computing the kernel and its partial derivatives, evaluating them at the pixel locations, storing them as matrices, and then approximating the kernel projections with dot products between the kernel matrix and the signal. This pre-computation speeds up the control loop. Moreover, while the images are not of infinite extent as assumed, we use kernels whose support is, for all practical purposes, compact (e.g. a Gaussian with small standard deviation).

In this section, we present the experimental results for controlling an eye-in-hand configuration according to the control laws described in Section II. Experiments were run using an American Robot Merlin 6200 series robot arm. This robot arm provides six degrees of freedom via a waist, shoulder, elbow, and spherical wrist configuration. Attached to the end of the arm is a Basler 602fc firewire camera, providing gray scale images with a resolution of 640x480 pixels. The robot is controlled via a dedicated workstation running Linux with real-time extensions. In order to facilitate algorithm development and implementation, the software infrastructure allows for direct control of the robot and capture of images directly from the GNU Octave mathematical software [7].

Our experiments consisted of 10 tests for each of the prescribed scenarios. We ran experiments on both contrived and natural images. In each set of trials, the robot was placed in an initial position and then moved a random amount away from the goal along the degree(s) of freedom being tested. In Figures 3, 5, 7, and 9, three of the ten trials are shown. They represent the maximum (red dashed line) displacement, minimum (black dash-dot line) displacement, and median (blue solid line) displacement from the goal location out of the random set of runs.

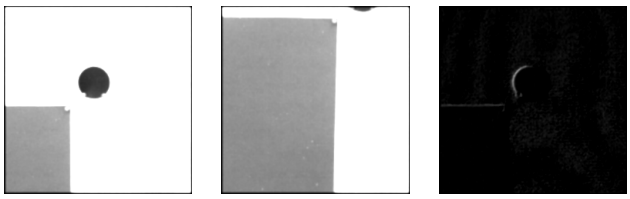

Fig. 2. Example images of a 2D trial with contrived image. *Left*- Goal image. *Center*- Initial displacement image. *Right*- Difference between the goal and the final images.

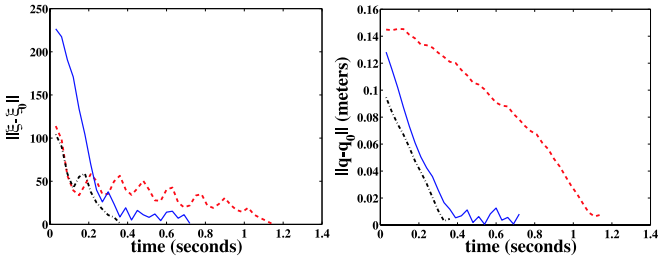

Fig. 3. 2D Translation: Three trials showing control to the goal image shown in Figure 2 as discussed in the text. *Left*- Convergence in kernel-projected value. *Right*- Convergence in x and y translation.

A. 2D Translation

In the two-dimensional case, our contrived image required a certain amount of structure to avoid the well known aperture problem in motion estimation. However, we found that natural images from our laboratory environment contain the needed information to avoid the aperture problem. Our natural images consist of the second author sitting in a chair in the foreground with the background inherent to the lab environment as shown in Figure 4. For all the experiments we used two Gaussian kernels, one for x motions and another for y motions.

The following parameters are tuned for the experiment: the sigma of the Gaussian kernels, the controller gains, and the convergence threshold. The width of the Gaussian kernel plays an important role in determining the size of the region of convergence and the accuracy of convergence. With a wide kernel, the domain of attraction is large, but may not converge exactly to the goal. A narrow kernel results in a small domain of attraction, yet provides tight convergence when starting near the goal. In all cases, the width of the kernel, the lighting, and brightness play an important role in determining the value and dynamic range of the kernel-projected value, ξ, thus driving the selection of the controller gain and the convergence threshold. Future work will entail determining how to reduce the number of parameters, as well as adapting the kernels during control to provide both a large domain of attraction and accurate convergence.

Figure 2 shows the images from a typical 2D translation using our contrived image. The translation performed

Fig. 4. Example images of a 2D trial in a real environment. *Left*- Goal image. *Center*- Initial displacement image. *Right*- Difference between the goal and the final images.

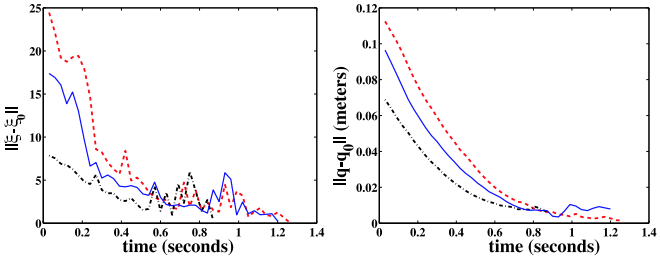

Fig. 5. 2D Translation: Three trials showing control to the natural goal image shown in Figure 4 as discussed in the text. *Left*- Convergence in kernel-projected value. *Right*- Convergence in x and y translation.

between the goal image and the initial image in Figure 2 was 6 cm in x motion and 12 cm in y motion. This typical example shows that even though the majority of the circular object in the image had been translated out of the image, the algorithm still converges to the goal. The convergence in both the Cartesian distance $|x - x_0|$ and the kernel-projected value $|\xi - \xi_0|$ is shown for the contrived and natural image in Figures 3 and 5 respectively.

B. Depth and Roll Motion

As the image signal is a finite window representation of a continuous underlying signal from the real scene, its FFT will have truncation ringing effects popularly known as the Gibbs phenomenon [12]. These truncation effects in the FFT will always be present and renders the relation given in (9) invalid. Different windowing functions [12] can be used to attenuate the Gibbs phenomenon at the cost of losing the image signal. In order to avoid this issue for the present paper, we use a simple scene with a constant background and threshold-based segmentation to make the background signal zero. This is similar to the binary projections used by [3], [9] for kernel-based tracking. Taking the FFT of this segmented image almost completely removes the truncation effects.

For both of these experiments, we picked an almost constant background so it could be easily segmented as shown in Figures 6 and 8. In order to remove new background pixels, we only use a cropped circular region in the image. For depth, we used a Gaussian kernel and for roll we used a rotationally asymmetric kernel,

$$K(\boldsymbol{w}) = (r_{max} - (w_i^2 + w_j^2)) \sin^2(\theta),$$

where r_{max} is the maximum radius picked by the user,

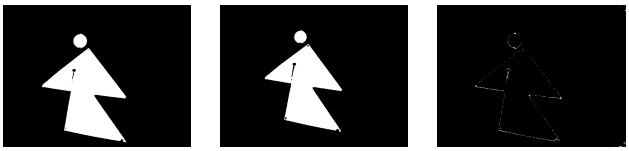

Fig. 6. Example images of a typical depth trial. *Left*- Goal image. *Center*- Initial displacement image. *Right*- Difference between the goal and the final images.

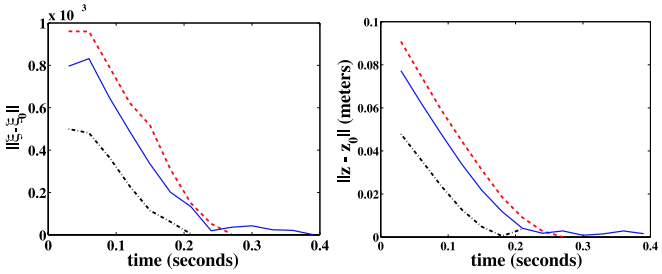

Fig. 7. Depth: Three trials showing control to the goal image shown in Figure 6 as discussed in the text. *Left*- Convergence in kernel-projected value. *Right*- Convergence in depth DOF.

$\boldsymbol{w} = (w_i, w_j)$ is the pixel location in the image and $\theta = \text{atan2}(w_j, w_i)$ is the angle of the pixel location $\boldsymbol{w}$.

Convergence in the degree of freedom and the kernel-projected value for depth and roll can be found in Figures 7 and 9, respectively. These figures show the convergence in the kernel-projected values and in the actuated degree of freedom: Cartesian translation for depth and angle of rotation about the optical axis for roll.

In the future, we plan to explore other methods of reducing the truncation due to the Gibbs phenomenon through various windowing methods [12]. Another alternative is to first transform the image to polar coordinates before computing the FFT. We suspect that this will recover the invariance, without the need to pre-segment the image.

C. Convergence Error Results

To quantify the performance of the KBVS control, we measured three differences between the initial conditions and the converged conditions. We first calculated the position error of the robot arm along the pertinent degree(s) of freedom using the forward kinematics of the robot arm. This provides an absolute measurement, in the world frame of the robot, of the convergence to the goal. Second, we

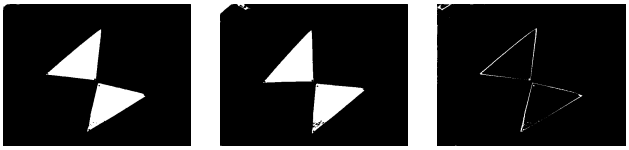

Fig. 8. Example images of a typical roll trial. *Left*- Goal image. *Center*- Initial displacement image. *Right*- Difference between the goal and the final images.

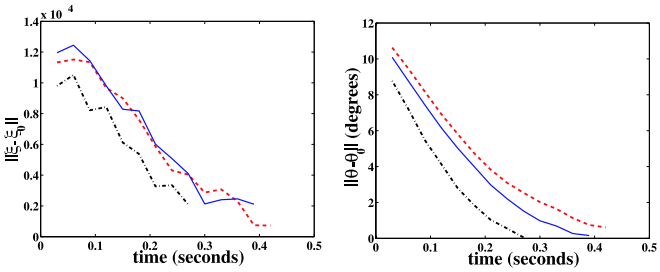

Fig. 9. Roll: Three trials showing control to the natural goal image shown in Figure 8 as discussed in the text. *Left*- Convergence in kernel-projected value. *Right*- Convergence in roll angle.

calculated a normalized difference of the kernel-projected value, giving an error in convergence to the goal in terms of the selected Lyapunov function. Third, we computed a summed square difference between the goal image and the image after convergence, normalized by the norm of the goal image. Table I lists all these three average errors and standard deviations over all the runs for the 2D translation, depth, and roll motion cases. For all the trials, we neither saw divergence or convergence to a local minimum for the chosen set of kernel parameters, gains and convergence thresholds. The repeatability of convergence to the goal is clearly demonstrated by the low error values in the table.

TABLE I

ANALYSIS OF FINAL ERRORS AFTER CONVERGENCE TO THE GOAL.

Motion	Error Type	Avg	Std
2D	Position (meters)	0.004875	0.002365
2D	kernel-projected Value (%)	0.1302	0.0632
2D	Image (%)	4.2009	1.4337
Depth	Position (meters)	0.002322	0.001563
Depth	kernel-projected Value (%)	0.0411	0.0238
Depth	Image (%)	4.8740	1.6543
Roll	Position (degrees)	0.28875	0.217241
Roll	kernel-projected Value (%)	0.7943	0.4077
Roll	Image (%)	12.2651	0.9038

IV. CONCLUSIONS

We presented a visual servoing controller that uses kernel-based sampling to directly exploit the underlying high dimensional visual signal for control, thereby bridging the gap between tracking and control. In particular, we have demonstrated the applicability of this approach separately on four degrees of freedom of motion – three translational directions and roll about the optical axis– on a range of visual signals. In addition, the approach allows combinations of these basic motion controllers to be extended to more complex tasks such as '$\mathrm{SE}(2)$ and depth'. We believe this provides an important first step towards a theoretical framework for analyzing stability and convergence issues of various vision-based control tasks.

As an exploratory paper, there was no attempt made to characterize the performance of our algorithm with respect to the image-kernel pair. The optimization of KBVS methods for a given servoing task will be an obvious next step, thus allowing us to investigate the benefit that KBVS may provide over existing visual servoing approaches for those specific tasks. The optimization of KBVS would entail exploring issues such as kernel parameter tuning, incorporating multiple kernels/image projections [4], [9] and designing kernels sensitive only to particular motions [6].

V. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. CMS-0625708.

REFERENCES

[1] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In D. Kriegman, G. . Hager, and A. Morse, editors, *The Confluence of Vision and Control*, pages 66–78. LNCIS Series, No 237, Springer-Verlag, 1998.

[2] A. V. Cideciyan. Registration of ocular fundus images: an algorithm usingcross-correlation of triple invariant image descriptors. *Engineering in Medicine and Biology Magazine, IEEE*, 14:52–58, 1995.

[3] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–575, 2003.

[4] J. J. Corso and G. D. Hager. Coherent Regions for Concise and Stable Image Description. *Computer Vision and Pattern Recognition*, 2:184–190, 2005.

[5] K. Deguchi. Interpretation of dynamic images with camera and object motions for vision guided robot control. *International Journal of Computer Vision*, 37(1):7–20, 2000.

[6] M. Dewan and G. Hager. Towards optimal kernel-based tracking. In *Computer Vision and Pattern Recognition*, volume 1, pages 618–625, June 2006.

[7] J. W. Eaton. *GNU Octave Manual*. Network Theory Limited, 2002.

[8] Z. Fan, Y. Wu, and M. Yang. Multiple collaborative kernel tracking. In *Computer Vision and Pattern Recognition*, 2005.

[9] G. D. Hager, M. Dewan, and C. V.Stewart. Multiple kernel tracking with ssd. *Computer Vision and Pattern Recognition*, 1:790–797, 2004.

[10] D. Kragic and H. I. Christensen. Cue integration for visual servoing. *IEEE Transactions on Robotics and Automation*, 1999.

[11] D. Kragic and H. I. Christensen. Robust visual servoing. *International Journal of Robotics Research*, 22(10-11):923–939, 2003.

[12] J. Proakis and D. Manolakis. *Digital Signal Processing: Principles, Algorithms and Applications*. Prentice Hall, 1996.

[13] E. Rivlin and H. Rotstein. Control of a camera for active vision: Foveal vision, smooth tracking and saccade. *International Journal of Computer Vision*, 39(2):81–96, 2000.

[14] O. Tahri and F. Chaumette. Point-based and region-based image moments for visual servoing of planar objects. *Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, 21(6):1116–1127, 2005.