Encoding Self Movement via Kernel Projections

Vinutha Kallem [*] John P. Swensen [†] Maneesh Dewan [‡]
Gregory D. Hager [§] Noah J. Cowan [¶] [‖]

Abstract

1 Introduction

Animals incorporate complex visual information—depth, optic flow, texture, color, individual features, and more—to plan and execute actions in the physical world. Typically, these actions create movement of the animal relative to objects in its environment which transforms the visual image, resulting in a closed sensorimotor loop. In nature, visual information is pre-processed using a set of spatiotemporal filters [32] that can be used for downstream control. For example broad field motion stimuli provide self movement information [39], which has inspired autonomous vehicle controllers [21]. Here, we further explore closed-loop position control based on spatial tuning functions analogous to the Gabor filters that approximate visual responses in the early visual system of vertebrates [23].

[*]V. Kallem is with SRI International, Princeton, NJ, USA. vinutha.kallem@sri.com

[†]J. P. Swensen is with the Department of Mechanical Engineering, Yale University, New Haven, CT, USA. john.swensen@yale.edu

[‡]M. Dewan **add details**

[§]G. D. Hager is with the Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA. hager@cs.jhu.edu

[¶]N. J. Cowan is with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD, USA. ncowan@jhu.edu

[‖]Some results in this paper have been presented at the IEEE/RSJ International Conference on Intelligent RObots and Systems (IROS), 2007 [24] and in a book chapter [33].

In robotics, vision-based control entails moving relative to a visual target to achieve a perceptual goal. This might include visual alignment of a set of features such as points and lines in the current view of a scene and those of a previously stored "desired" view [3, 4, 22, 25]. In this approach, exact correspondence between these geometric features enables the generation of a variety of control algorithms [8–10, 17, 28–31, 36, 40–42]. Kragic and Christensen [26] use a combination of visual servoing methods to perform tasks motivated by service robotics problems where the robots are given a task to fetch a particular object. They use object recognition to recognize the object and use 2D visual servoing methods for the gross motions of the robots towards the object. In the final stage, the authors use 3D visual servoing techniques for precise motions of the robots and object manipulations.

In this paper, we propose a more abstract set of features to encode relative movement of the visual sensor and its environment. We define these features by a finite number of spatial filters of the image. We then use these abstract features or "kernel projections" to design a control law for performing a visual alignment task based on abstract features. In this new formulation, the control law is more closely rooted to visual information present at the level of individual pixels, much like the recent work of Dame and Marchand [11]. It also turns out that, with binary images and the simple kernels, our approach to KBVS can be reduced to visual servoing on image moments [34]. In fact, the crux of our convergence argument relies on the fact that KBVS is a sort of "grayscale generalization" of moment-based visual servoing.

There has been other important work on vision-based control without feature tracking. Deguchi [12] defines a high dimensional space whose dimensions encode all the image pixels and the camera/target motions. Since motion of the camera/target and the resulting image are related, they lie on a lower dimensional surface within this high dimensional space. The authors use this relationship and use numerical methods to find the equation defining the surface and its Jacobian to perform visual servoing. Tahri and Chaumette [34] perform visual servoing via moments which is related to the KBVS approach presented here as discussed extensively in this paper. In [2, 6, 6, 20] the authors present visual servoing methods without any tracking by considering each pixel as feature and developing a Jacobian that relates the motion of the camera to motion of the features. They minimize a cost function that is equal to the squared norm of the image intensity difference to develop the control law. These methods are similar to placing a kernel at every pixel location in our KBVS algorithm.

The inspiration of the present work arises from the idea of inverting the kernel-

based tracking problem. In kernel-based tracking, a spatially weighted average of the image is taken to obtain a kernel measurement (see Definition 2.1). Tracking is then cast as finding the optimal kernel placement in each frame of the image sequence to minimize the difference in the kernel measurements. Comaniciu *et al.* [7] bin each pixel location into one of the finite number of bins (clusters) with the binning function defined on the image features at that location. Then, a weighted spatial average is taken to obtain a histogram vector, which is the kernel measurement. Tracking the region of interest then becomes equivalent to moving the location of the kernel center in the subsequent frame so that a cost function based on the Bhattacharyya coefficient is minimized.

Hager *et al.* [19] extend the tracking algorithm to multiple kernels for tracking complex motions by defining kernels that depend on only degree of motion and invariant to other motions. Fan *et al.* [16] also use multiple kernels to track articulated objects like human hands. Dewan and Hager [13] develop optimal kernels for tracking purposes to improve the tracking performance in terms of lesser iterations needed for tracking convergence and lower computational cost.

In this paper, we take a "dual" view of the kernel-based tracking problem, and consider kernel projections/measurements as abstract features for visual servoing. In this method, tracking image features is unnecessary and the visual servoing problem is formulated as driving the kernel projections to desired values (obtained by computing the kernel-projected values at the goal location). Using these abstract features we prove (local) asymptotic convergence for motions on $\mathrm{SE}(3)$, and present extensive experimental trials and demonstrations.

Separation of tracking and control subproblems in traditional VS algorithms makes the selection of the features to be tracked, as well as the tracking method itself, detached from the controller choice. In contrast, KBVS involves designing kernel functions with the aim of increasing the basin of attraction as well as to obtain more robust, reliable, and rapid convergence. This design space ranges at one extreme from having a single kernel to the other extreme of placing a kernel at every pixel in the image, with the only constraint being that the kernels are smooth (C^2) functions from image coordinates to $\mathbb{R}$. The KBVS framework affords stability of the controller while providing an integrated design phase posed in terms of overall visual servoing performance.

2 Kernel-based visual servoing controllers

In this section, we develop KBVS controllers for $\mathrm{SE}(3)$ motions of a camera mounted on a robot. We consider the "eye-in-hand" configuration in which the camera is mounted on the robot end effector, and the target is planar and is stationary. Further, we assume a kinematic motion model for the robot, whose control inputs are its joint velocities (a requirement that can be relaxed readily for second order, fully actuated, holonomically constrained plants as discussed later). For simplicity, we treat image pixels as continuous variables over all of $\mathbb{R}^2$ (or S^2 in the case spherical projection) measured in continuous time, rather than discrete variables over a finite image measured in discrete time, as stability proofs allowing for spatiotemporal quantization would be challenging.

The major limitations of the algorithm we present is that, except for the $\mathrm{SO}(3)$ case (in which the location of the camera does not change) the visual target must be planar. In practice, some of these assumptions can clearly violated (see experimental results in Section 5) and we discuss these assumptions and their practicality in Section 4.

In the present work, we treat the image captured by the camera and its transformations are treated as signals that are directly measured. Given the signal, s, the kernel-projected measurement or kernel measurement is defined as below.

Definition 2.1 *Let $\mathbf{K} : \mathscr{I} \to \mathbb{R}^n$ be a vector-valued piecewise continuous function defined on the location space of the image. Given a signal, $s(w,t)$ (such as the image pixel intensities as a function of time) the kernel-projected value of the signal at time t, called the* **kernel-projected measurement** *or simply* **kernel measurement**, *$\xi \in \mathbb{R}^n$, is defined as the vector*

$$\xi(t) = \int_{\mathscr{I}} \mathbf{K}(w) s(w,t) dw, \tag{1}$$

where $w \in \mathscr{I}$ is the image spatial indexing variable. The function $\mathbf{K}$ is known as the **kernel**. *For perspective cameras $\mathscr{I} = \mathbb{R}^2$ and for spherical cameras $\mathscr{I} = S^2$.*

As the camera moves relative to the target, the signal, $s(\cdot,t)$, changes thus affecting the kernel-projected value. The aim of KBVS is to drive the robot/camera to the goal configuration by driving $\xi(t) \to \xi_0$.

Consider a robot with a camera mounted on its end-effector as shown in Figure 1. Assume that the camera is looking at a planar scene and for convenience, choose a world reference frame attached to the planar scene so that any point P in the scene can be represented as $P = [\lambda_1, \lambda_2, 0]^T \in \mathbb{R}^3$ in world coordinates; note

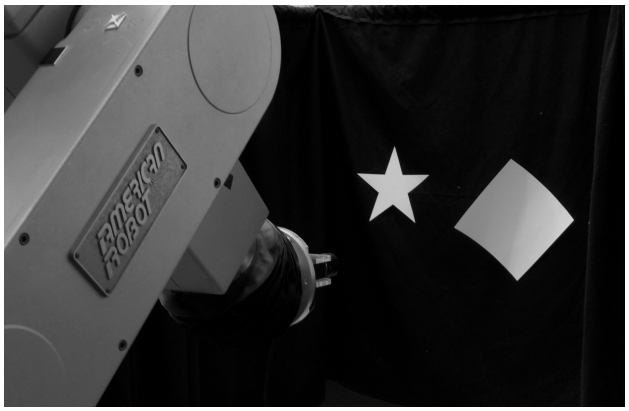


Figure 1: Experimental configuration. Camera is mounted on a 6-DOF industrial robot. The camera is looking at a planar scene in front of it.

that the third entry is zero, because of our choice of alignment of the world frame with the target plane. Here, P denotes the coordinates of a point in the world frame; we will use this convention for all the points. The location of the camera in this world reference frame is denoted by $g = (R,T) \in \mathrm{SE}(3)$, where $R \in \mathrm{SO}(3)$ and $T \in \mathbb{R}^3$. In the camera's reference frame, the coordinates of the point P are

$$P_c = H \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ 1 \end{bmatrix}, \quad H = \begin{bmatrix} Re_1 & Re_2 & T \end{bmatrix},$$

where e_1, e_2, e_3 denote the standard basis in three dimensions. (Note that H can be interpreted as homography matrix relating homogenous coordinates of the target plane and the image plane.) The image coordinates of the point P are given by

$$w = \frac{1}{e_3^T P_c} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} P_c = \frac{1}{e_3^T P_c} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} H \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ 1 \end{bmatrix}$$

Let function $h(\lambda, g)$ denote the mapping from target plane coordinates, $\lambda = (\lambda_1, \lambda_2)$, and camera pose, g, to the image coordinates. In other words, $w = h(\lambda, g)$. It will be conveinent to express coordinates on the planar scene in terms of coordinates on the image plane, so we write $\lambda = h^{-1}(w, g)$; this mapping can be readily calculated by using the inverse of the homography, H.

We assume that the image intensities are invariant to orientation, and in what follows, express the current image $s(w,t)$ in terms of a base image $s_0(\lambda)$, namely the visual intensity of the scene expressed in world coordinates. The image s_0 can be interpretted as what the image would look life it were frontoparallel. **need a figure to show this!** With this in mind, the signal at time t needed to compute the kernel project in (1) is given by

$$s(w,t) = s_0(h^{-1}(w, g(t))). \tag{2}$$

Here, the image value at each time and image coordinate, $s(w,t) \in \mathbb{R}$, is expressed in terms of the base image s_0.

Assume the camera is controlled according to the following kiematics:

$$g^{-1}\dot{g} = \begin{bmatrix} \omega \\ v \end{bmatrix}^{\wedge}$$

It is our goal to find a policy for choosing (ω, v) to ensure convergence to a desired pose. Upon differentating the image (2) as a function of time, we have

$$\dot{s} = \nabla s_0 \;\; D_2 h \;\; \begin{bmatrix} \omega \\ v \end{bmatrix}.$$

where $D_2 h(\lambda, g(t))$ represents the Jacobian matrix of $h(\lambda, g(t))$ with respect to its second argument, $g(t)$.

As a general proof of concept, choose the kernel function $\mathbf{K}(w)$ to a vector of impulses, say, one centered at each of n pixels. By the sifting property of impulses, the kernel measurement in (1) reduces to a measurement of the image, $\xi(t) = I(t) \in \mathbb{R}^n$ at a discrete set of pixel locations, i.e. the kernel projection just results in the discretely sampled image. Differentiating this kernel projection ξ, we obtain

$$\dot{\xi} = \nabla I \;\; D_2 h \;\; \begin{bmatrix} \omega \\ v \end{bmatrix}.$$

Therefore, if we choose

$$\begin{bmatrix} \omega \\ v \end{bmatrix} = B \;\; \dot{\xi}, \tag{3}$$

where B is the pseudo inverse of $\nabla I \;\; D_2 h$, the robot will converge to the goal. Here, we are assuming that $\nabla I \;\; D_2 h$ is full rank. In other words, we are assuming that

there are six independent optical flow equations that arise for the entire camera image.

While the control law given by (3) can be expected to be locally convergent unless the image intensity distribution is profoundly degenerate, the approach of placing an impulse at every pixel it is fraught with obvious problems, such as sensitivity to noise, and extremely small basin of attraction.

In the next few sections, we will show how for specific cases, the kernel function choices can be made to improve the control law performance.

3 Some Special Cases

In this section we develop KBVS controllers for a camera constrained to move in several subsets (of up to four dimensions) of SE(3). We first describe our overall method for a 2D translation parallel to the image plane (x,y), and then describe translation along the optical axis (z), and roll about the camera optical axis (θ) and then on the group of rotations (roll, pitch, yaw). In Sections 3.1 – 3.4 the visual targets are assumed to be planar and parallel to the image plane of the camera, thus having a uniform scaling of the target.

3.1 2D Translation

Consider a robot with a camera mounted on it as shown in Figure 1. The image intensity at each pixel is taken as the signal, under the image brightness constancy constraint (cf. [38], pp192–195). Let the kernel projection of the image at the goal be ξ_0 at the position $x = 0$ (without loss of generality). Our goal is to determine a control input u that will drive the kernel-projected measurement to ξ_0, thus driving $q(t) \to 0$.

Let the motion of the camera relative to the target be 2D translation parallel to the image plane of the camera. 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

$$\dot{q} = u, \tag{4}$$

where $u \in \mathbb{R}^2$ is the robot control input. Since the target scene is a unit distance away from the image plane $I(w,q(t)) = I_0(w-q(t))$. Through a change of variables, $\bar{w} = w+q$, and recalling that $\mathscr{I} = \mathbb{R}^2$, the kernel-projected measurement

$\xi \in \mathbb{R}^{n\times 1}$, with the image captured by the camera as the signal, can be rewritten as

$$\xi = \int_{\mathscr{I}} K(w) I_0(w-q) dw = \int_{\mathscr{I}} K(\bar{w}+q) I_0(\bar{w}) d\bar{w}, \tag{5}$$

where $K \in \mathbb{R}^{n\times 1}$ is the kernel function.

From (5), observe that even when the images or the signal are discontinuous and hence not differentiable, the kernel-projected measurement is analytically time 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:

$$V = \frac{1}{2} \|\xi - \xi_0\|^2. \tag{6}$$

Applying the chain rule, we have

$$\begin{aligned}
\dot{V} =& (\xi - \xi_0)^T \frac{\partial \xi}{\partial t} \\
& (\xi - \xi_0)^T \frac{\partial \xi}{\partial q} \dot{q} \\
=& (\xi - \xi_0)^T \Big(\int_{\mathscr{I}} K'(\bar{w}+q) I_0(\bar{w}) d\bar{w} \Big) \dot{q} \\
=& (\xi - \xi_0)^T \Big(\int_{\mathscr{I}} K'(w) I(w) dw \Big) u,
\end{aligned}$$

where $K'(w) = \frac{\partial K(w)}{\partial w} \in \mathbb{R}^{n\times 2}$. Note that in the last step, we revert the coordinates back to w. Now, choose the input, u, as

$$u = -\Big(\int_{\mathscr{I}} \nabla K(w) I(w,q) dw \Big) (\xi - \xi_0), \tag{7}$$

where we define $\nabla K = (\frac{\partial K}{\partial w})^T \in \mathbb{R}^{2\times n}$. This notation is motivated by the fact that the i^{th} column of ∇K is the gradient of i^{th} component of the kernel. With this choice, we have

$$\dot{V} = -\left\| \Big(\int_{\mathscr{I}} \nabla K(w) I(w,q) dw \Big) (\xi - \xi_0) \right\|^2. \tag{8}$$

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 5) to be a valid (locally) generic assumption and, in any case, can be numerically tested and optimized [13]. With these assumptions, this choice of input guarantees stability (in the Lyapunov sense) of the controller.

For practical applications, it is crucial to obtain at least local *asymptotic* stability. Let $J(q)$ be the Jacobian given by

$$J(q) = \frac{\partial \xi}{\partial q} = \int_{\mathscr{I}} K'(\bar{w}+q) I_0(\bar{w}) d\bar{w}. \tag{9}$$

For asymptotic stability we use Taylor series expansion of the system:

$$\xi = \xi_0 + J_0 q + O(\|q\|^2), \tag{10}$$

where $J_0 = J(\mathbf{0})$. Substituting this first order approximation into the expressions for V and $\dot{V}$ results in

$$V = \frac{1}{2} q^T J_0^T J_0 q + O(\|q\|^3), \tag{11}$$

and

$$\dot{V} = -q^T Q Q^T q + O(\|q\|^3), \quad Q = J_0^T J_0. \tag{12}$$

If the Jacobian matrix at the goal, $J_0 \in \mathbb{R}^{n\times 2}$, is full column rank, Q will be a full rank 2×2 matrix. From (11), (12), and a full rank assumption for Q, we can conclude that V is positive definite and $\dot{V}$ is negative definite in some neighborhood of the goal with $V = 0$ and $\dot{V} = 0$ at the goal. In general, if we choose the kernel function, $K \in \mathbb{R}^{n\times 1}$ for a p-dimensional system ($q \in \mathbb{R}^{p\times 1}$), then similar analysis shows that $J_0 \in \mathbb{R}^{n\times p}$ and $Q = J_0^T J_0 \in \mathbb{R}^{p\times p}$. For V to be positive-definite and $\dot{V}$ to be negative definite in q, it is necessary to have $n \geq p$, *i.e.* the dimensional of the kernel should be at least the number of degrees of freedom in the system for local asymptotic stability [33].

In the experiments presented in Section 5.1.1, we have chosen to decouple the 2D translational motions into two 1D translations by using two independent x and y directional kernels, similar to the idea presented in [13]. 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 everypixel in the y direction. In other words, this will make ∇K a diagonal matrix. In demonstrations presented in Section 5.2 we use three kernels and in Section 5.1.6 we use nine kernels to enlarge the domain of attraction and to improve the rate of convergence in more natural, complex scenes.

Remark: Note that by differentiating the Equation 1 we obtain

$$\frac{\partial \xi}{\partial t} = \Big(\int_{\mathscr{I}} K'(\bar{w}+q) I_0(\bar{w}) d\bar{w} \Big) \dot{q} = J(q)\dot{q}.$$

If we consider ξ as a set of abstract features, the above equation is exactly the Jacobian relationship between the image-feature locations and robot stated in the standard visual servoing equations. Since, in this work we choose the kernel functions to be smooth, the Jacobian $J(q)$ can be analytically and exactly calculated without tracking any features in the image.

3.2 Translation along the optical axis

Do we need to add a volume preserving element to the integration change of coordinate?

Here, we consider motions of a camera along its optical axis. Even though this corresponds to a translation as in the previous case, there is a fundamental difference between the two: 2D x–y translations simply translate the image, while motions in depth 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. Cideciyan [5] 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 the FT to translation to develop controllers for depth and rotation that can integrated with the previously developed 2D controllers in the x–y plane. First, we consider motions in depth only.

We assume that the goal corresponds to unity depth (without any loss of generality). Let I_0 denote the image at the goal, and F_0 the magnitude of its spatial FT. 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

$$\dot{z} = u, \tag{13}$$

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

$$I(w,z) = I_0(w/z).$$

One can show that the magnitudes of the spatial FT [1] of these images (F and F_0 respectively) are related by

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

We define the kernel-projected measurement, $\xi \in \mathbb{R}^{n\times 1}$, as

$$\xi = \int_{\mathscr{I}} K(v)F(v,z)dv = \int_{\mathscr{I}} K(\bar{v}/z)F_0(\bar{v})d\bar{v}, \tag{14}$$

where n is the dimension of the kernel function and $\bar{v} = zv$. At the goal we have $\xi_0 = \int_{\mathscr{I}} K(v)F_0(v)dv$. Our aim is to drive the robot to $z = 1$ by driving $\xi(t) \to \xi_0$. Consider the Lyapunov function candidate $V = \frac{1}{2}\|\xi - \xi_0\|^2$ and choose the input as

$$u = \left(\int_{\mathscr{I}} v^T \nabla K(v)F(v,z)dv\right)(\xi - \xi_0), \tag{15}$$

then

$$\dot{V} = -\frac{1}{z}\left\|(\xi - \xi_0)^T \int_{\mathscr{I}} K'(v)vF(v,z)dv\right\|^2.$$

If $z > 0$, $\dot{V}$ is negative semi-definite, which is a realistic assumption for objects seen by the camera.

3.3 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

$$\dot{\theta} = u, \tag{16}$$

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:

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

[1] In experiments, we have a discrete image, so we use the Discrete Fourier Transform (DFT) to compute the kernel measurement

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

$$F(v,\theta) = F_0(R_\theta v),\ v \in \mathbb{R}^2. \tag{17}$$

We define the kernel-projected measurement as

$$\xi = \int_{\mathscr{I}} K(v)F(v,\theta)dv = \int_{\mathscr{I}} K(R_\theta^T \bar{v})F_0(\bar{v})d\bar{v}, \tag{18}$$

where $\bar{v} = R_\theta v$ and n is the dimension of the kernel function. At the goal, the kernel-projected measurement is $\xi_0 = \int_{\mathscr{I}} K(v)F_0(v)dv$. As before, our aim is to drive the robot to $\theta = 0$ by driving $\xi(t) \to \xi_0$. Consider the Lyapunov function candidate $V = \frac{1}{2}\|\xi - \xi_0\|^2$ and choose the control input as

$$u = -\left(\int_{\mathscr{I}} v^T J \nabla K(v)F(v,\theta)dv\right)(\xi - \xi_0) \tag{19}$$

where $J = R_{\frac{\pi}{2}} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$. With this choice of u, $\dot{V}$ can be calculated as

$$\dot{V} = -\left\|(\xi - \xi_0)^T \int_{\mathscr{I}} K'(v)J^T v F(v,\theta)dv\right\|^2,$$

which is negative semi-definite.

3.4 Extensions to $\mathrm{SE}(2)$ + depth motions

In the above controllers, we used the image as the signal for the x–y translations and we use the magnitude of the FT of the image as the signal for depth and roll. As discussed above, the magnitude of the FT of the image removes any translation effects while controlling depth and roll and for our coupled $\mathrm{SE}(2)$ + depth experiments (Section 5.1.5) we use kernels that partially decouple 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 $\mathrm{SE}(2)$ (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 by executing the depth and roll controllers and then the planar 2D controller. In the experiments discussed in Section 5, we control all the four degrees of freedom ($\mathrm{SE}(2)$ and depth) simultaneously.

3.5 Kernel-based visual servoing on the group of rotations

In this section we develop KBVS controllers for motions on the group of rigid body rotations. Assume that the camera is a (unit) spherical camera and that the robot is rotating in place (about the camera optical center) with no translations. We consider a fully actuated robot whose kinematic model is given by

$$R^{-1}\dot{R} = \hat{\omega} \tag{20}$$

where $R \in \mathrm{SO}(3)$ denotes the state of the robot and $\omega \in \mathbb{R}^3$ represents the control input to the robot. The symbol $\widehat{\cdot}$ is the usual isomorphism between $\mathbb{R}^3$ and $\mathfrak{so}(3)$ defined as

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}.$$

At the goal, let the signal be $I_0 = s(\cdot, 0)$ and the kernel-projected measurement at the goal be denoted by $\xi_0 = \xi(0)$. Since the dimension of $\mathrm{SO}(3)$ manifold is 3, for the current purpose we need to have $n \geq 3$. At any rotation, the image captured by the camera is a rotated version of the image at the goal and the relationship between the image at time t, $I(t)$ and image at the goal, I_0 is given by

$$I(p,t) = I_0(R(t)p). \tag{21}$$

Here, kernels must be defined on S^2, the unit sphere. To do this in our experiments (Section 5.1.6) we generated Gaussian kernels in $\mathbb{R}^3$ and found their intersection with the 2D sphere. With that in mind, define the kernel measurement as

$$\xi(t) = \int_{\mathscr{I}} K(p)I(p)dp = \int_{\mathscr{I}} K(R^{-1}\bar{p})I_0(\bar{p})d\bar{p}, \tag{22}$$

where $\bar{p} = Rp$. Given the kernel measurement at the goal as ξ_0, let the Lyapunov function be

$$V = \frac{1}{2}\left\|\xi - \xi_0\right\|^2.$$

The time derivative of the Lyapunov function is given by

$$\dot{V} = -(\xi - \xi_0)^T \int_{\mathscr{I}} K'(p)\ \hat{\omega}\ p\ I(p)\ dp,$$

where $K'(p) = \frac{\partial K}{\partial p} \in \mathbb{R}^{n\times 3}$. $\dot{V}$ can be rewritten as

$$\begin{aligned}\dot{V} =&(\xi - \xi_0)^T \int_{\mathscr{I}} K'(p)\ \hat{p}\ \omega\, I(p)\, dp \\ =&(\xi - \xi_0)^T \underbrace{\left(\int_{\mathscr{I}} K'(p)\ \hat{p}\, I(p)\, dp\right)}_{J(R)}\ \omega\end{aligned}$$

where $J(R)$ is the mapping rotational velocities and kernel-measurement "velocities".

$$\omega = \left(\int_{\mathscr{I}} \hat{p}\ \nabla K(p)\ \ I(p)\, dp\right)(\xi - \xi_0), \tag{23}$$

where we define $\nabla K = (\frac{\partial K}{\partial p})^T \in \mathbb{R}^{3\times n}$. With this choice of the input, the time derivative of the Lyapunov function is

$$\dot{V} = -\left|\left|(\xi - \xi_0)^T \int_{\mathscr{I}} K'(p)\ \hat{p}\, I(p)\, dp\right|\right|^2$$

which is negative semi-definite function. Here we used the fact that $(K'(p)\ \hat{p})^T = -\hat{p}\ \nabla K(p)$. Thus, the system is Lyapunov stable. Following the previous arguments in Section 3.1, one can show that the $n \times 3$ matrix $J(R_0)$, where R_0 is the rotation at the goal, must be rank 3 to ensure that both $V > 0$ and $\dot{V} < 0$ in a neighborhood of the origin.

4 Assumptions and their practicality

In the design of the above KBVS controllers, we make several simplifying assumptions which might not hold in real experiments. We assume that the spatial indexing of the image, w or p is a continuous variable while in practice it is discrete (pixels locations are discrete). For modern high resolution cameras (480×640 or higher) this assumption seems mild and is handled in practice by treating the image intensities as a piecewise constant function with intensities being constant along a each pixel length. To improve computation speed, this can also handled by evaluating the kernel function and its derivatives at discrete pixel locations and storing them ahead of the experiment. The kernel measurements can now be calculated as the dot product of two matrices—image intensities and

kernel matrix. Similarly the control inputs can be calculated as dot products of relevant matrices.

We also make the image brightness constancy assumption (cf. [38], pp192–195), namely that as the camera moves, the image brightness of each point in space is time invariant as the point moves through the field of view. This is a common assumption in visual servoing and tracking literature and is well justified for a fixed, Lambertian surface under constant lighting. One could relax this assumption using various methods such as normalizing the image intensities by subtracting the mean intensity and scaling by the standard deviation, or considering "basis" images to compensate for illumination, or using the magnitudes of the gradients of the image instead of the image intensities [1, 18, 35]. Ultimately, this constraint could be relaxed, and in fact changes in brightness induced by changing viewing angle (for non-Lambertian surfaces) could be utilized for control.

The other major assumption we make is that we assume that the image captured by the camera is infinite. This assumption in the 2D cases is handled by using finite support kernels (e.g. a truncated Gaussian with small standard deviation). In the depth and the roll cases, we handle this by using images that have constant intensity at the boundaries and then performing discrete FT; for the simple scenes in the experiments we simply perform intensity thresh-holding before the discrete Fourier transform. In future, this class of images used for depth and roll cases will be expanded. One way to expand the applicability is to perform segmentation to extract a region of interest in the image and set the rest of the image to an intensity of zero. This approach, however, will require region tracking frame to frame (and with close correspondence to the goal segmentation), for example by using background and foreground segmentation [15, 37]. Binning functions similar to the ones used in kernel-based tracking methods [7, 19] can be used for these purposes.

We also assume that the target scene of the camera is planar in Sections 3.1–3.4. This assumption made the controller design simpler but we have seen in our laboratory experiments that this assumption can safely be violated to some extent. It has been observed that the controllers work when the scene largely consists of planar patches even though the scene is not completely planar. The formal bounds on how much this assumption can be violated is left as future research.

5 Experiments and Demonstrations

In this section we present the experimental results of the KBVS controllers developed in this paper. Experiments are performed on an American Robot Merlin 6200 series robot arm. This robot has six degrees of freedom located at the waist, shoulder, elbow and 3-DOF wrist. The camera attached to the robot's end effector is a gray scale Basler 602fc camera with IEEE1394 (firewire) interface. The resolution of the camera is 480×640 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 the capture of images directly from the GNU Octave mathematical software [14].

Below are six sets of experiments that show the convergence of the KBVS controllers:

1. Translation in the 2D plane parallel the image plane of the camera ($\mathbb{R}^2$, Section 5.1.1).
2. Translation along the optical axis of the camera ($\mathbb{R}$, Section 5.1.2).
3. Rotation about the camera's optical axis ($\mathrm{SO}(2)$, Section 5.1.3).
4. Simultaneous translations parallel to the camera's image plane and translational along the camera's optical axis ($\mathbb{R}^3$, Section 5.1.4).
5. The three translational degrees of freedom and roll about the camera's optical axis ($\mathbb{R}^3 \times \mathrm{SO}(2)$, Section 5.1.5).
6. Pure rotation about the optical center of an omnidirectional camera ($\mathrm{SO}(3)$, Section 5.1.6).

Finally, in Section 5.2, we present demonstrations where the controllers are applied in more natural, complex images.

5.1 Experiments

5.1.1 2D Translational motion

In these experiments the robot is allowed to translate parallel to the image plane of a camera. The first image in the Figure 2 (*Third column*) shows the image of the scene at the goal location. From the goal location, the camera is moved away in

the two translational directions (approximately) parallel to the image plane. The kernels for the experiments are:

$$K_x(w) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(w_1-\mu_x).^2}{2\sigma_x^2}},$$
$$K_y(w) = \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{(w_2-\mu_y).^2}{2\sigma_y^2}},$$

where $\mu_x = \mu_y = -100$ and $\sigma_x = \sigma_y = 70$ and K_x and K_y are kernels in the x and y directions. w represents the pixel indexing of the image.

We conducted 10 trials with random initial positions. In these experiments, the mean initial position error was 6.98 cm with a standard deviation of 0.72cm. The trial was stopped when the difference of between the current kernel measurement and that at the goal location was below a certain threshold; in these experiments this threshold was chosen as 0.1 which is $0.02 - 0.03\%$ of the goal kernel measurement in each direction. In all the trials, the position converged to the goal position with a mean position error of 0.31cm and standard deviation of 0.15cm. In Figure 2 (*Top row*) we plot these 10 trials and highlight the one with maximum initial displacement from the goal location (red solid line). In all the experiments, the robot converges to the goal state and the Lyapunov energy goes to zero.

5.1.2 Depth motion

In these experiments the robot is allowed to translate along the camera's optical axis. Recall that the signal for the depth DOF controller is the magnitude of the FT of the image. As discussed in Section 4, since the field of view is finite, the depth controller needs the coarse image segmentation to make the intensity of the image background zero. In our experiments, we perform a simple color segmentation and assign a value of zero to the background and one to the rest of the image. The kernel function we use is:

$$K_z(v) = e^{-\frac{1}{8}||v||^2},$$

where v represents the spatial indexing for the FT.

We conducted 10 trials with random initial positions. In these experiments, the mean initial position error was 8.02cm with a standard deviation of 4.52cm. The trial was stopped when the difference of between the current kernel measurement and that at the goal location was below a threshold; in these experiments this was

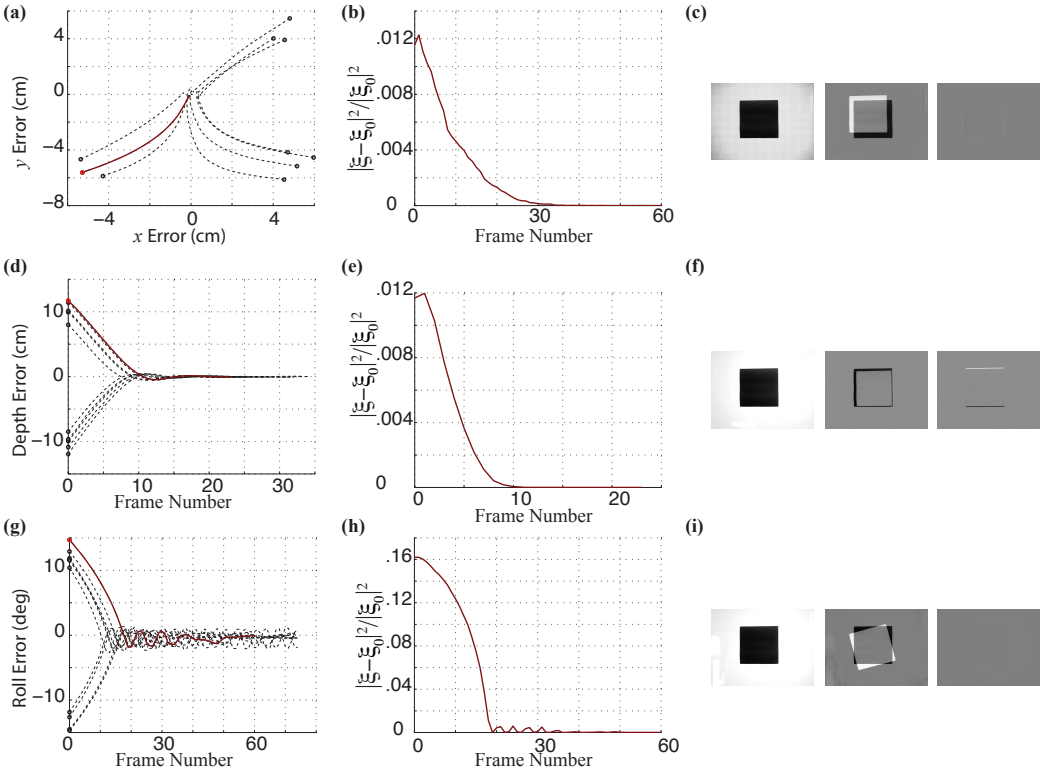


Figure 2: (a)-(c): 2D KBVS experiments. (d)-(f): Depth KBVS experiments. (g)-(i) Roll KBVS experiments. **First column:** convergence in the states for each set of 10 trials. The trajectory with largest initial displacement (red solid line) is explored in more detail in the second and third columns. **Second column:** Lyapunov function over time for trial with largest initial displacement. **Third column:** For the same trial displayed in the second column, three images are shown: the image at the goal location, the difference between the goal and the initial displacement images, and difference between the goal and the final images.

chosen as 5 which is 0.02% of the goal kernel measurement. In all the trials, the position converged to the goal position with a mean position error of 0.06cm and standard deviation of 0.065cm. In Figure 2 (*Middle row*) we show these 10 trials and highlight the trial representing the maximum initial depth displacement of the robot from its goal location (red solid line).

5.1.3 Roll motion

In these experiments the robot is allowed to rotate about the camera's optical axis. As in the depth case, the signal to the controller is the FT of the background segmented case. The kernel function we use is:

$$K_\theta(v) = e^{-\frac{1}{8}v_1^2} + e^{-\frac{1}{8}v_2^2},$$

where v represents the spatial indexing for the FT.

We conducted 10 trials with random initial positions. In these experiments, the mean initial position error was 12.67° with a standard deviation of 1.51°. The trial was stopped when the difference of between the current kernel measurement and that at the goal location was below 5 which is 0.02% of the goal kernel measurement. In all the trials, the position converged to the goal position with a mean position error of 0.52° and standard deviation of 0.53°. In Figure 2 (*Bottom row*) we show the trials and the highlighted trial (red solid line) represent the maximum initial roll displacement of the robot from its goal location.

5.1.4 3D Translational motion

In these experiments the robot is allowed in move in all the three translational degrees of motion. Recall that the signal for the depth DOF controller is the magnitude of the FT of the image and that for the other two translational DOFs is the image itself. Since the magnitude of the FT does not depend on the translations motions parallel to the image plane, the vector field given by (15) is independent of the other two translational directions. Hence, as discussed in Section 3.4, we can first control the depth independently and once this controller converges, the 2D controller takes over. But in experimental settings, one might have to iterate multiple times to obtain adequate convergence in all the three DOFs, since the target scene is not exactly parallel to the image plane of the camera. (In anecdotal trials of this iterative approach, two iterations usually suffice.)

We present a simpler approach here that also proves effective. Specifically, we simply add the perpendicular vector fields given by (15) and (7). With this

approach, we (theoretically) maintain local convergence guarantees because the depth controller is independent of the other two directions of translation. Consequently, the local system dynamics have an "upper triangular" structure, guaranteeing local stability. In the experiments, we ensure that depth convergence is faster by choosing higher gains for the depth controller compared to the 2D controller.

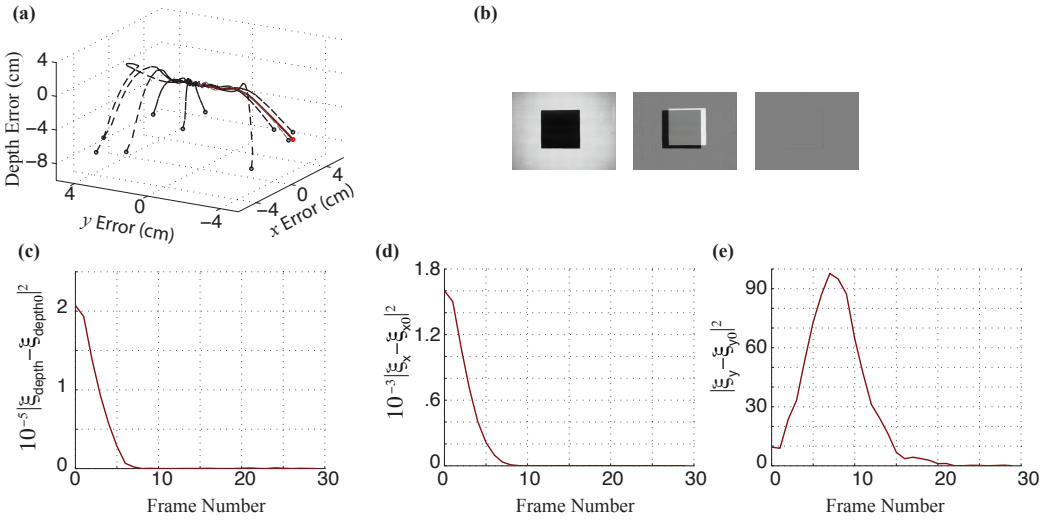


Figure 3: Coupled 3D translation experiments. (a) Convergence in x, y, and depth degrees of freedom for all 10 trials. The trial with the maximum initial displacement from the goal location is represented by a solid red line. (b) For the trial with the maximum initial displacement from the goal, three images are shown from left to right: the image at the goal location, the difference between the goal and the initial displacement images, and difference between the goal and the final images. (c)-(e) Convergence in kernel measurements in all three degrees of freedom. Note that, as expected, the depth convergence is invariant to $x-y$ translation; however, incorrect depth early in the trial imperils the initial performance of the 2D controller, particularly in the y direction. Once depth has nearly converged (by approximately frame 15) the kernel error in the y degree of freedom begins to decrease after its initial rise, and ultimately all three degrees of freedom converge.

From the goal location, the camera is moved away in the three translational directions. The controller acts to bring the robot/camera to the goal. The kernels

for the experiments are:

$$K_x(w) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(w_1-\mu_x).^2}{2\sigma_x^2}},$$
$$K_y(w) = \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{(w_2-\mu_y).^2}{2\sigma_y^2}},$$
$$K_z(v) = e^{-\frac{1}{8}||v||^2},$$

where $\mu_x = \mu_y = -100$ and $\sigma_x = \sigma_y = 70$ and K_x, K_y and K_z are kernels in the x, y and z directions. w represents the pixel indexing of the image and v is the spatial indexing for the FT.

We conducted 10 trials and in all of them the robot converged to the goal with a mean position error of 0.46cm and standard deviation of 0.2cm. The initial position difference from the goal location in these 10 trials has a mean of 8.74cm with a standard deviation of 0.75cm. In Figure 3 we show the 10 trials and highlight the one with maximum initial displacement from the goal (in red). In the highlighted trial, we can see that the kernel measurement corresponding to the y direction increases in the beginning and then comes down. This happens because the controller in the y direction is dependent on the depth DOF and once the depth has converged to its value at the goal, the defined on the y coordinate converges to zero and the goal location is reached in all the three DOFs. In general, the depth DOF converges faster than the other two.

5.1.5 SE(2) and depth motion

Here we discuss the experiments that are conducted when the robot moves in all the three translational DOFs and roll DOF about the camera's optical axis. As in the previous case, we conduct the experiments by adding the vector fields in the four DOFs given by (7), (15) and (19). From the goal location, the camera is moved away in the three translational and roll directions. The controller acts to

bring the robot/camera to the goal. The kernels used for the experiments are:

$$K_x(w) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(w_1-\mu_x).^2}{2\sigma_x^2}},$$
$$K_y(w) = \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{(w_2-\mu_y).^2}{2\sigma_y^2}},$$
$$K_z(v) = e^{-\frac{1}{8}||v||^2},$$
$$K_\theta(v) = e^{-\frac{1}{8}v_1^2} + e^{-\frac{1}{8}v_2^2},$$

where $\mu_x = \mu_y = -100$ and $\sigma_x = \sigma_y = 70$ and K_x, K_y, K_z and K_θ are kernels in the x, y, z and roll (θ) directions. w represents the pixel indexing of the image and v is the spatial indexing for the FT. Note that with the chosen depth kernel, the depth kernel measurement and control is invariant to not just 2D translations but also to roll. Roll kernel measurement and control is invariant to just 2D translations. Hence, the depth DOF must converge before the roll DOF can converge to the goal.

We conducted 10 trials and in all of the trials the robot converged to the goal with a mean position error of 0.44cm with standard deviation of 0.21cm and mean roll error 0.56° with standard deviation of 0.39° in roll DOF. The initial position difference from the goal location in these 10 trials has a mean of 8.94cm with a standard deviation of 0.48cm in the translational DOFs and a mean of 12.01° with standard deviation of 3.15° in the roll DOF. In Figure 4(a) and highlight one of the trials. As in the last experiment set, the depth DOF converges first and the rest of the DOFs converge later.

5.1.6 $\mathrm{SO}(3)$ motion

These experiments use a Basler A620fc camera, fitted with a RemoteReality omnidirectional lens, attached to the end-effector of the robot. The camera resolution is 480×640 and the omnidirectional lens consists of an orthographic lens and a paraboloid mirror. Here the robot rotates in all the three directions about the focus of the paraboloid mirror (the center of the equivalent spherical camera). From the goal location, the camera is rotated about the center of the spherical camera. For the image used in this set of experiments, we found that using a larger number of kernels than the required 3 improves the domain of attraction [33]. Figure 5 also shows the kernels used for these experiments overlaid on top of the goal image; level-sets of the kernels are shown as dotted-circles.

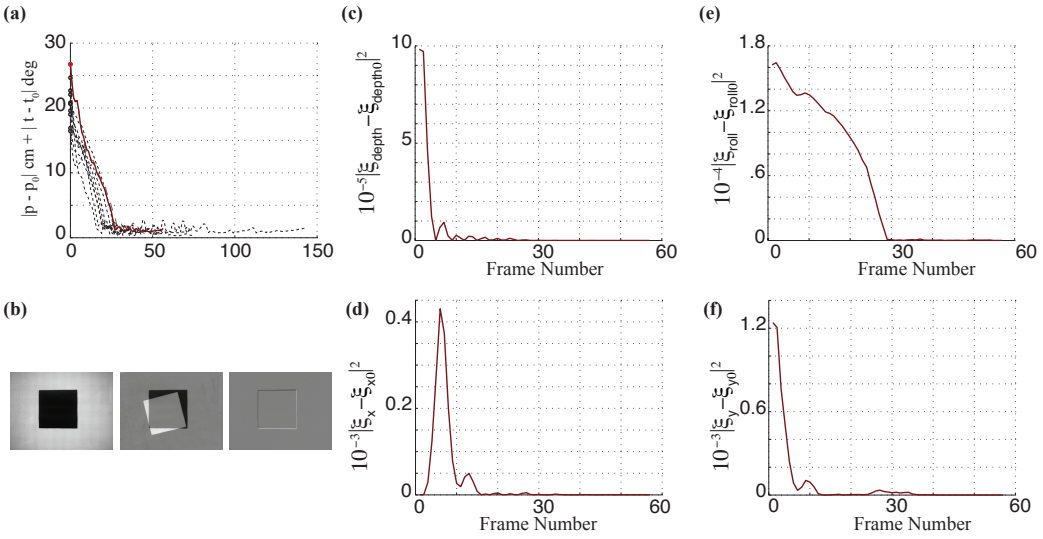


Figure 4: SE(2) and depth experiments. (a) Ten trials showing the convergence in the states. Again, the trial with maximum initial displacement (red solid line) is highlighted in (b–f). (b) From left to right: image at the goal location, difference between the goal and the initial displacement images and difference between the goal and the final images, all for the highlighted trial. (c–f) Convergence in kernel measurements in all four degrees of freedom.

We conducted 10 trials with random initial positions. In Figure 6(a) show the convergence (presented in exponential coordinates) for all these trials with the trial with maximum initial orientation displacement from the goal location highlighted (solid red line). In Figure 6(b) we show the convergence in the Lyapunov function for this highlighted trial and show the image at the goal, difference between the initial and goal images and the difference between the final and goal images in Figure 6(c). In these experiments, the mean of the norm of the initial displacement in all three degrees of rotation is 9.28° with a standard deviation of 1.321°. The trial was stopped when the difference of between the current kernel measurement and that at the goal location was below a threshold of 5% of the goal kernel measurement. In all the trials, the position converged to the goal position with a mean position error of 0.14° and standard deviation of 0.1074°.

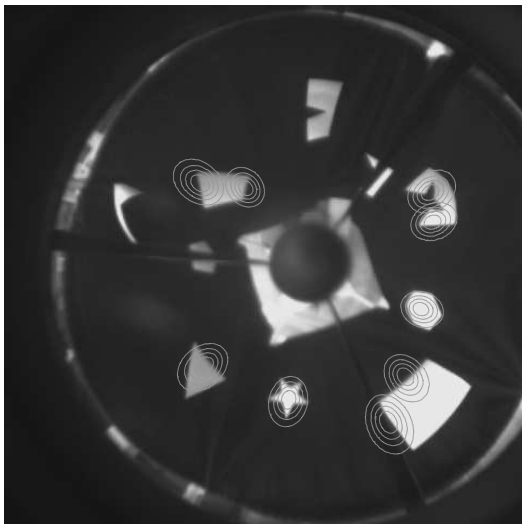


Figure 5: Nine kernels are overlaid on top of the image at the goal location. The level-sets of the kernels are shown as dotted-circles.

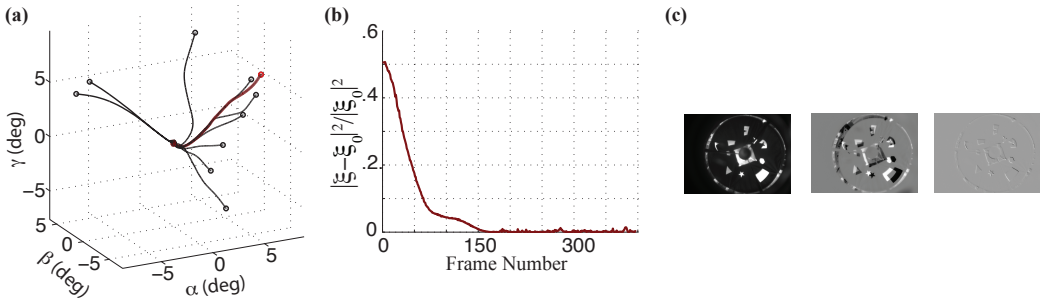
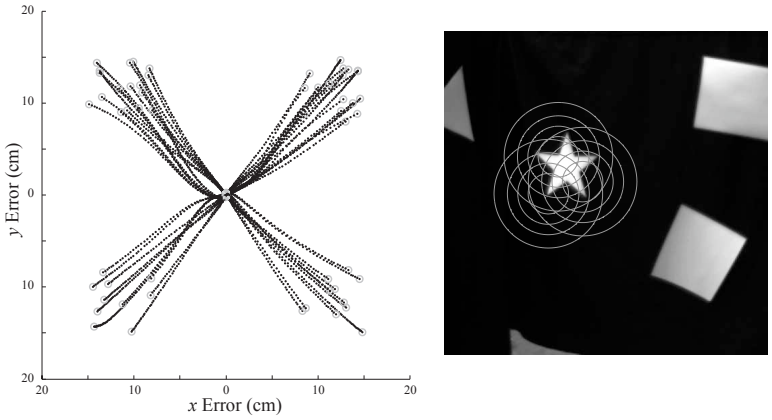


Figure 6: SO(3) KBVS experiments. (a) Convergence in the exponential coordinates of the rotation matrix. The trial with the maximum initial displacement from the goal orientation is highlighted (solid red line). (b) Convergence in Lyapunov function for the highlighted trial. (c) From left to right: goal image, difference between the goal and the initial displacement images, and difference between the goal and the final images, all for the highlighted trial.

5.2 Demonstrations

In the above sets of experiments, we use simple images to illustrate the working of KBVS controllers. These simple images simplify kernel placement (which we did "by hand"), leaving issues such as the kernel optimization to future work; as a first step toward such optimization, we experimentally characterized the domain of attraction of KBVS controllers in [33]. So, rather than an exhaustive experimental account of KBVS for "natural" images, in this section, we present a few anecdotal experiments using KBVS on more complex natural scenes to demonstrate the effectiveness of the approach. In Figure 7 we present 2D translation KBVS experiments where we use three kernels to enlarge the region of convergence. In Figures 8, 9 we present KBVS experiments for depth and roll KBVS controllers respectively. In Figure 10 we show successful experimental trials in a more natural scene in the laboratory with the robot allowed to translate in a plane parallel to the image plane of the camera.



(a) (b)

Figure 7: A demonstration of 2D KBVS with three kernels. (a)Convergence results for the 50 trials of 2D translations parallel to the image plane. (b) Image and three kernels used for the experiment. The level sets (circles) of the kernel are shown on the top of the images, to show the kernel placement.

In Figure 5.2 we show the effect of kernel function choice on convergence of 2D KBVS controller. In the left column we see the region of convergence is nearly the extent of the sampled robot motion. In contrast, the right column shows a case for which a very similar distribution of kernels results in a much smaller

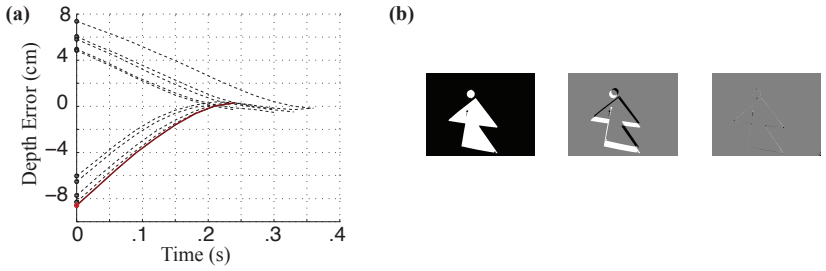


Figure 8: A demonstration of depth KBVS. (a) Convergence in the depth DOF with one highlighted trial (solid red line). (b) Image at the goal, difference between initial and goal images, and difference between final and goal images for the highlighted trial.

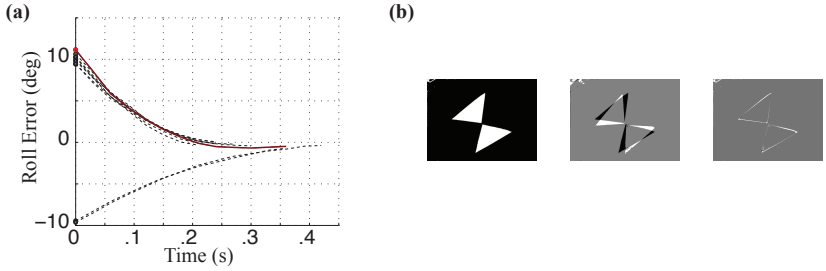


Figure 9: A demonstration of roll KBVS. (a) Convergence in the roll DOF with one highlighted trial (solid red line). (b) Image at the goal, difference between initial and goal images, and difference between final and goal images for the highlighted trial.

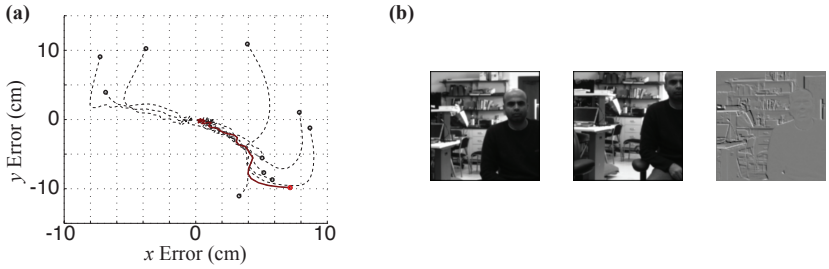
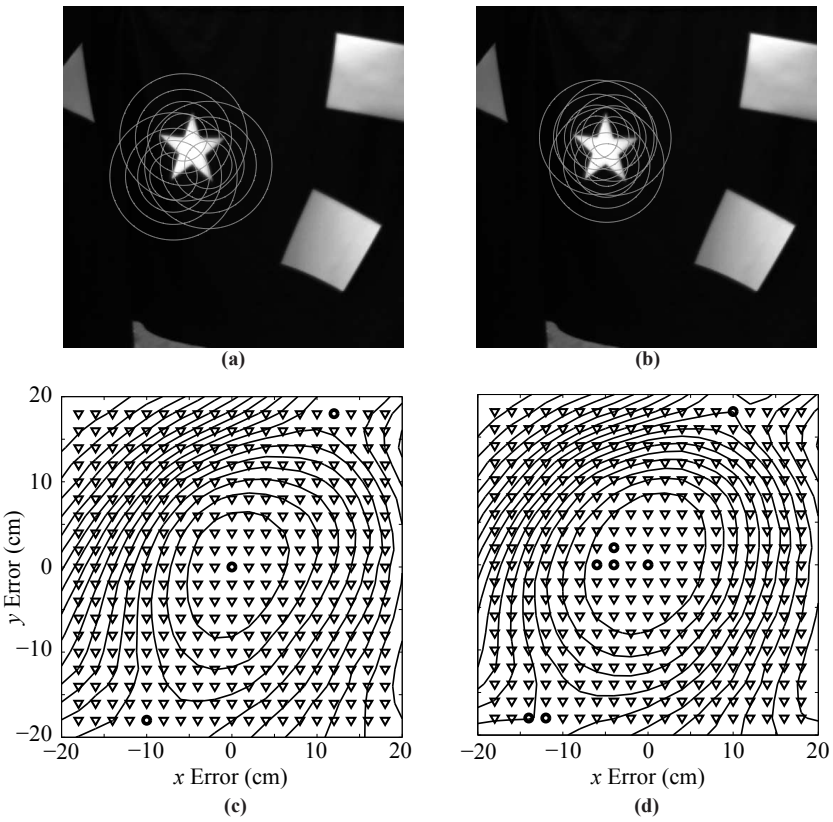


Figure 10: A demonstration of 2D KBVS in natural scene. (a) Convergence in the x and y DOFs with one highlighted trial (solid red line). (b) Image at the goal, difference between initial and goal images, and difference between final and goal images for the highlighted trial.



region of convergence. This sensitivity of a successful solution to the kernel selection process further illustrates the need for a principled method of selecting kernels, and is an important area for future research. In laboratory experiments, we observed that placing Gaussian kernels centered at SIFT (Scale-Invariant Feature Transform) features [27] of the scene resulted in Lyapunov functions with a large region around the goal in which the Lyapunov function was "flat". In hindsight, this could have been easily predicted: the invariance of these feature points to motions in near the goal results in little variation in the kernel-projected measurement! We observed that randomly placed Gaussian kernels (numbering two to fifty) was typically at least as good as our handpicked kernels, and was often better. While these investigations concerning kernel placement are anecdotal, they suggest that KBVS controllers will work with a wide variety of more complex and natural scenes when we pick suitable kernels; techniques to choose such kernels automatically is work in progress.

References

[1] P. Belhumeur and D. Kriegman. What is the set of images of an object under all possible lighting conditions? *Int. J. Computer Vision*, 28:270–277, 1996.

[2] A. Censi, S. Han, S. B. Fuller, and R. M. Murray. A bio-plausible design for visual attitude stabilization. In *Proc. IEEE Int. Conf. on Decision Control*, 2009.

[3] F. Chaumette and S. Hutchinson. Visual servo control, part i: Basic approaches. *IEEE Trans. Robot.*, 13(4):82–90, 2006.

[4] F. Chaumette and S. Hutchinson. Visual servo control, part ii: Advanced approaches. *IEEE Trans. Robot.*, 14(1):109–118, 2007.

[5] 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.

[6] C. Collewet, E. Marchand, and F. Chaumette. Visual servoing set free from image processing. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 81–86, May 2008.

[7] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intel.*, 25(5):564–575, 2003.

[8] P. I. Corke and S. A. Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Trans. Robot. Automat.*, 17(4):507–515, 2001.

[9] N. J. Cowan and D. E. Chang. Geometric visual servoing. *IEEE Trans. Robot.*, 21(6):1128–1138, Dec. 2005.

[10] N. J. Cowan, J. D. Weingarten, and D. E. Koditschek. Visual servoing via navigation functions. *IEEE Trans. Robot. Automat.*, 18(4):521–533, 2002.

[11] A. Dame and E. Marchand. Mutual information-based visual servoing. *IEEE Trans. Robot.*, 27(5):958–969, 2011.

[12] K. Deguchi. A direct interpretation of dynamic images with camera and object motions for vision guided robot control. *Int. J. Computer Vision*, 37(1):7–20, 2000.

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

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

[15] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. In *Proc. IEEE*, volume 90, pages 1151–1163, 2002.

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

[17] G. D. Hager. A modular system for robust hand-eye coordination. *IEEE Trans. Robot. Automat.*, 13(4):582–595, 1997.

[18] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. Pattern Anal. Mach. Intel.*, 20(10):1125–1139, 1998.

[19] G. D. Hager, M. Dewan, and C. V. Stewart. Multiple kernel tracking with SSD. In *Proc. Comp. Vision and Pattern Recognition*, volume 1, pages 790–97, 2004.

[20] S. Han, A. Censi, A. D. Straw, and R. M. Murray. A bootstrappable bioplausible design for visual pose stabilization. In *Proc. IEEE Int. Conf. Robot. Autom.*, 2010. Submitted.

[21] J. S. Humbert and A. M. Hyslop. Bioinspired visuomotor convergence. *IEEE Trans. Robot.*, 26(1):121–130, 2010.

[22] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Trans. Robot. Automat.*, 12(5):651–670, 1996.

[23] J. P. Jones and L. A. Palmer. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *J. Neurophysiol.*, 58, 1987.

[24] V. Kallem, M. Dewan, J. P. Swensen, G. D. Hager, and N. J. Cowan. Kernel-based visual servoing. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1975–1980, San Diego, CA, USA, Oct. 2007.

[25] D. Kragic and H. I. Christensen. Survey on visual servoing for manipulation. Technical Report, Jan. 2002.

[26] D. Kragic and H. I. Christensen. Robust visual servoing. *Int. J. Robot. Res.*, 22(10-11):923–939, 2003.

[27] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Computer Vision*, 60(2):91–110, 2004.

[28] C. P. Lu, G. D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, 2000.

[29] E. Malis and F. Chaumette. Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods. *IEEE Trans. Robot. Automat.*, 18(2):176–186, 2002.

[30] E. Malis, F. Chaumette, and S. Boudet. 2-1/2-d visual servoing. *IEEE Trans. Robot. Automat.*, 15(2):238–250, Apr. 1999.

[31] Y. Mezouar and F. Chaumette. Path planning for robust image-based control. *IEEE Trans. Robot. Automat.*, 18(4):534–549, 2002.

[32] W. Reichardt. Autocorrelation and the central nervous system. In A. Rosenblith, editor, *Sensory Communication*, pages 303–318. MIT Press Cambridge, 1959.

[33] J. P. Swensen, V. Kallem, and N. J. Cowan. Empirical characterization of convergence properties for kernel-based visual servoing. In *Visual Servoing via Advanced Numerical Methods*, volume 401, pages 23–38. Springer, 2010.

[34] O. Tahri and F. Chaumette. Point-based and region-based image moments for visual servoing of planar objects. *IEEE Trans. Robot.*, 21(6):1116–1127, Dec. 2005.

[35] I. M. Takahiro, T. Ishikawa, and S. Baker. The template update problem. *IEEE Trans. Pattern Anal. Mach. Intel.*, 26:810–815, 2003.

[36] C. J. Taylor and J. P. Ostrowski. Robust vision-based pose control. In *Proc. IEEE Int. Conf. Robot. Autom.*, volume 3, pages 2734–2740, San Francisco, CA, 2000.

[37] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *Proc. Comp. Vision*, volume 1, pages 255–261, 1999.

[38] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice-Hall, 1998.

[39] A. Wertz, B. Gaub, J. Plett, J. Haag, and A. Borst. Robust coding of ego-motion in descending neurons of the fly. *J. Neurosci.*, 29(47):14993–15000, 2009.

[40] W. J. Wilson. Visual servo control of robots using kalman filter estimates of robot pose relative to work-pieces. *In K. Hashimoto, editor, Visual Servoing, World Scientific*, pages 71–104, 1994.

[41] P. Zanne, G. Morel, and F. Plestan. Robust 3d vision based control and planning. In *Proc. IEEE Int. Conf. Robot. Autom.*, volume 5, pages 4423–4428, Apr. 2004.

[42] H. Zhang and J. P. Ostrowski. Visual motion planning for mobile robots. *IEEE Trans. Robot. Automat.*, 18(2):199–208, 2002.