Empirical Validation of a New Visual Servoing Strategy

Noah J. Cowan Joel D. Weingarten Daniel E. Koditschek
The University of Michigan; Ann Arbor, MI 48105

Abstract— **The flexibility of computer vision is attractive when designing manipulation systems which must interact with otherwise unsensed objects. However, occlusions introduce significant challenges to the construction of practical vision-based control systems. This paper provides empirical validation of a vision based control strategy that affords guaranteed convergence to a visible goal from essentially any "safe" initial position while maintaining full view of all the feature points along the way. The method applies to first (quasi-static, or "kinematic") and second (Lagrangian or "mechanical") order plants that incorporate an independent actuator for each degree of freedom.**

I. Introduction

For many types of manipulation, sensing plays a crucial role. Computer vision provides flexible sensing for manipulation systems, and consequently has been used for many experimental platforms involving robotic dexterity. Because problems of manipulation are themselves quite challenging, little formal attention has been paid to the computer-vision side of manipulation. In visual servoing systems by contrast, the literature focuses on the geometry of computer vision, in the context of a somewhat trivial manipulation task: move a fully actuated rigid body kinematically from an initial condition to a goal location. To endow manipulation systems with more sensory intelligence, we seek to apply realistic sensor models in the context of dextrous manipulation tasks.

Increasingly, researchers employ computer vision systems as sensors that measure the projection, y, of features from a rigid body at each configuration, q, as it moves in some scene. Such an approach presupposes the availability of a reliable machine vision system that supplies a controller with the image plane coordinates, $y = c(q)$, of features of a rigid body being observed in the scene. The machine vision system must incorporate image processing, feature extraction and correspondence algorithms suitable to the scene in view [6], [7]. For visual servoing, these features are then used to close a control loop around some desired visual image.

Hutchinson *et. al.* [9] provide a general introduction and extensive bibliography to the large literature on visual servoing. Briefly, a vision-based controller imposes motion upon the actuated configuration space variables, q, so as to align the imaged features, y with a previously stored reference image, y^*. When the sensor map, c, is injective in some vicinity of the goal then this generally results in a closed loop system with an asymptotically stable equilibrium state at the unique pre-image $q^* = c^{-1}(y^*)$. Many algorithms of this nature have been proposed and implemented with the result of large basins of attraction around the equilibrium in the presence of large errors in sensor and robot calibration.

Typically visual servoing algorithms employ a simple, fully actuated, kinematic plant model, $\dot{q} = u$, the input to which is generally

$$u = -J(q)^{\dagger}(y^* - y) \quad \text{where} \quad J^{\dagger} := \left(J^T J\right)^{-1} J^T, \qquad (1)$$

and $J(q) := D_q c\,(q)$, is the Jacobian matrix. This control attempts to impose thereby straight-line motion of the feature points on the image plane. The main advantage to this approach, many argue, is that convergence is robust to the model parameters of the camera and the rigid body being servoed.

Despite their benefits, traditional visual servoing algorithms suffer from some or all of the following failings. First, they are generally *quasi-static.* Ignoring the mechanical system dynamics precludes the possibility of high performance control and hence imposes restrictions on the speed of operation. Second, their basin of attraction is *local.* For example the alignment of $(y^* - y)$ with the null space of $J^{\dagger}$ in (1) may incur spurious (attracting) critical points, and hence, not uncommonly, the local basin of attraction around q^* excludes seemingly reasonable initial conditions. Third, and perhaps most importantly, all of the visual servoing algorithms proposed to date are vulnerable to transient loss of features — either through self-occlusions or departure from the field of view (FOV). To the best of our knowledge, no prior work guarantees that these obstacles will be avoided. Usually these problems are ignored in analysis, and when encountered in practice simply cause the system to move into an emergency stop state, necessitating human intervention.

We have shown that, for some important special cases [2] [3] [4], the obstacles presented by self occlusion and finite FOV can be obviated by addressing in a methodical fashion the relationships between the domain and range of c from which they arise. Recently [2], we introduced a general framework for visual servoing yielding feedback controllers which are

1. *dynamic:* applicable to second order (Lagrangian) as well as first order (kinematic) actuation models;
2. *global:* guaranteeing a basin of attraction encompassing, essentially, the entire set of initial configurations that maintain feature visibility;
3. *visibility-obstacle free:* avoiding configurations that lose features due to either self occlusion or departure from the FOV.

In visual servoing, a few researchers have recently incorporated Lagrangian dynamics into the visual servo loop.

*This work was supported in part by the NSF under grant IRI-9510673, and the first author was supported by the Rackham Predoctoral Fellowship, University of Michigan.

For example, Zhang and Ostrowski [17] developed a controller for an Unpiloted Aerial Vehicle (UAV) equipped with a camera. Also, some researches have demonstrated provably large domains of attraction for kinematic servo systems (without accounting for occlusions) [13] [16]. Finally, there have been recent efforts to address the FOV problem [13] [1], albeit in a quasi-static setting.

II. Robot Control via Navigation Functions

We assume a holonomically constrained, fully actuated robot and denote the $n-$dimensional free configuration space as $\mathcal{Q}$. Lagrange's equations (see, for example, [5]) yield

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau$$

where τ is the vector of input torques. The camera plays the role of sensor, $c : \mathcal{Q} \to \mathcal{Y}$. Letting the input torque be $\tau = u + G(q)$, where u is our control input, the plant equations are, in state-space form,

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} x_2 \\ M(x_1)^{-1}\left(u - C(x_1, x_2)x_2\right) \end{bmatrix}, \\ y &= c(x_1), \end{aligned} \qquad (2)$$

where $x = [q^T, \dot{q}^T]^T$. Notice that our "generalized PD" approach to control, below, will not require the precise form of M or C.

A. Task specification

The state space is constrained by the presence of forbidden configurations, the *obstacle set* $\mathcal{O} \subset \mathcal{Q}$. The *free space* is defined as the obstacle-free configuration space $\mathcal{V} = \mathcal{Q} - \mathcal{O}$, and *safe configurations* $\mathcal{D} \subseteq \overline{\mathcal{V}}$ form a compact connected differentiable manifold with boundary. The positioning objective is described in terms of a *goal set* $\mathcal{G} \subset \overset{\circ}{\mathcal{D}}$. The task is to drive q to $\mathcal{G}$ asymptotically subject to (2) by an appropriate choice of u while avoiding obstacles. We restrict our attention to point attractors, $\mathcal{G} = \{q^*\}$. Moreover, the basin of attraction $\mathcal{E} \subset T\mathcal{D}$ must include a dense subset of the zero velocity section of $T\mathcal{D}$, so that we may guarantee convergence from the entire configuration space. Obstacle avoidance requires that the trajectories avoid the *boundary set* $\mathcal{B} = \partial\mathcal{D}$, *i.e.* $q(t \geq 0) \subset \mathcal{D}$.

B. Navigation functions

The task of moving to a goal while avoiding obstacles along the way can be achieved via a nonlinear generalization of proportional-derivative (PD) control deriving from Lord Kelvin's century old observation that total energy always decreases in damped mechanical systems [19]. Formally, this entails the introduction of a gradient vector field from a "navigation function," a refined notion of an artificial potential function, together with damping to flush out unwanted kinetic energy [11], [12], [14].

B.1 Second order, damped gradient systems

Let $\mathcal{D}$ be a compact manifold with boundary, and $\varphi : \mathcal{D} \to \mathbb{R}$ be a C^2 functional, which encodes a goal, q^*, as a unique minimum. Combining a potential function gradient with a linear damping term, yields a simple "PD" style feedback, in local coordinates,

$$u = -\,D_q\varphi(q)^T - K_d\,\dot{q}, \qquad (3)$$

that is appropriate for second order plants. From Lord Kelvin's observation it follows that the total energy,

$$\eta = \varphi + \kappa \quad \text{where} \quad \kappa = \tfrac{1}{2}\dot{q}^T M(q)\dot{q}, \qquad (4)$$

is non-increasing. A refined the class of potential functions will enable us to construct controllers for which the basin of attraction contains a dense subset of the zero velocity section of $\mathcal{D}$. The following definition has been adapted from [11].

Definition 1: Let $\mathcal{D}$ be a smooth compact connected manifold with boundary, and $q^* \in \overset{\circ}{\mathcal{D}}$ be a point in its interior. A Morse function (A smooth scalar valued function whose Hessian matrix is non-singular at every critical point [8]), $\varphi \in C^2[\mathcal{D}, [0,1]]$ is called a *navigation function* (NF) if
1. φ takes its unique minimum at $\varphi(q^*) = 0$;
2. φ achieves its maximum of unity uniformly on the boundary, *i.e.* $\partial\mathcal{D} = \varphi^{-1}(1)$.

This notion, together with Lord Kelvin's observation, now yield the desired convergence result for the Lagrangian system (2).

Proposition 1: (Koditschek [11]) Given the system described by (2) subject to the control (3), almost every initial condition q_0 within the set

$$\mathcal{E} = \{(q,\dot{q}) \in T\mathcal{D} : \eta(q,\dot{q}) \leq 1\} \qquad (5)$$

converges to q^* asymptotically. Furthermore, transients remain within $\mathcal{D}$ such that $q(t) \subset \mathcal{D}$ for all $t \geq 0$.
Note that $\mathcal{E}$ imposes a "speed limit" as well as a positional limit, since the total energy must be initially bounded.

B.2 First order gradient systems

If we assume our plant is given by $\dot{q} = u$, we may use a purely kinematic, or quasi-static, version of (3), namely

$$\dot{q} = -M^{-1}(q)\, D_q^T\varphi(q) \qquad (6)$$

where M is an arbitrary Riemannian metric. Under the same assumptions on φ, essentially all initial conditions with $\mathcal{D}$ converge to q^* [11].

B.3 Invariance under diffeomorphism

One last key ingredient in the mix of geometry and dynamics underlying the results we present revolves around the realization that a navigation function in one coordinate system is a navigation function in another coordinate system, if the two coordinate systems are diffeomorphic [11]. This affords the introduction of geometrically simple model spaces and their correspondingly simple model navigation functions.

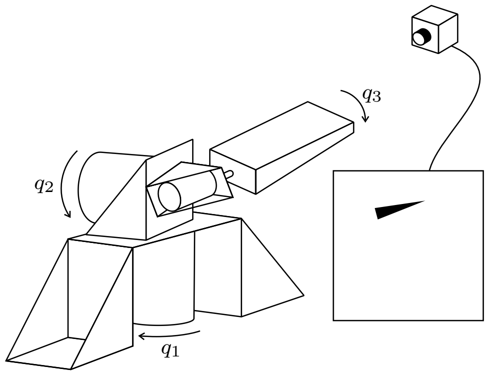

Fig. 1. The Buehgler arm has been modified with an "arrow" feature on the end of the paddle, which is observed by a perspective projection camera. The camera image is segmented to extract the arrow, as depicted.

III. Navigation Function Based Visual Servoing

We have created visual servoing algorithms that are high performance, global and yet safe with respect to occlusion obstacles $\mathcal{O}$ that arise due to the finite FOV and self-occlusions [2]. To achieve our objective we compute the *visible set* for a particular problem. This is the set of all configurations $\mathcal{V} := \mathcal{Q} - \mathcal{O}$ in which all features are visible to the camera and on which c is well defined. We then design a safe, possibly conservative, compact subset $\mathcal{D} \subseteq \mathcal{V}$ in which the body is permitted to move. The set $\mathcal{D}$ provides additional safety with respect to obstacles and possibly simplifies the topology. We then define the *image space* $\mathcal{I} = c(\mathcal{D}) \subset \mathcal{Y}$. The camera map must be a diffeomorphism $c : \mathcal{D} \approx \mathcal{I}$. For each problem, $\mathcal{D}$ is analyzed to construct a model space $\mathcal{Z}$ and a diffeomorphism $g : \mathcal{I} \approx \mathcal{Z}$. The camera measures the goal image $y^* = c(q^*)$.

This paper proposes a new framework for visual servoing that incorporates three ingredients:
1. a *model space*, $\mathcal{Z}$, for the "safe" configurations, $\mathcal{D}$;
2. a *navigation function* $\widetilde{\varphi} : \mathcal{Z} \to [0,1]$, for the model space;
3. a *diffeomorphism*, $g : \mathcal{I} \to \mathcal{Z}$, from the image space to the model space.

These three ingredients are assembled with the feedback control strategy (3), which guarantees that *all* initial configurations within $\mathcal{D}$ converge to the goal while ensuring occlusion-free transients.

By recourse to the general framework outlined above we have developed controllers for some specific configurations of a robot and monocular camera [2], two of which are reviewed in the subsections that follow. These two configurations are empirically validated in Section IV on two distinct experimental platforms.

A. Example 1: Buehgler arm, spatial camera

The Buehgler arm, depicted in Figure 1, has three actuated degrees of freedom denoted in local coordinates by angles $q = [\ q_1,\ q_2,\ q_3\ ]^T$ and its configuration space is $\mathcal{Q} = \mathrm{T}^3$. Denote the homogeneous transformation from the paddle frame to the world base frame, given in [15], as ${}^wH_b(q)$. The transformation from the world frame to camera frame is ${}^cH_w$, and hence $H = {}^cH_b = {}^cH_w {}^wH_b$. For a feature ${}^bp = ({}^br, {}^bv) \in T_1\mathbb{E}^3 \approx \mathbb{E}^3 \times \mathrm{S}^2 = \mathcal{FS}$, the total forward kinematic map is then defined $h : \mathcal{Q} \times \mathcal{FS} \to \mathcal{FS}$

$$h(q) := \begin{bmatrix} H(q)^b r \\ R(q)^b v \end{bmatrix} =: \begin{bmatrix} r(q) \\ v(q) \end{bmatrix}$$

where (r, v) are in the camera frame. We consider a feature point at the end of the paddle (which has length ϱ), and orientation in the y-direction of the paddle, namely $({}^bp, {}^bv) = ([\ 0,\ 0,\ \varrho,\ 1\ ]^T, [\ 0,\ 1,\ 0\ ]^T) \in \mathcal{FS}$.

A perspective projection camera is positioned to view the robot end effector as depicted in Figure 1. The transformation from the world frame to camera frame is ${}^cH_w$, and the kinematics expressed the camera frame are $r = {}^cH_w {}^wr$ and $v = {}^cH_w {}^wv$.

The camera map is given by the projection of the feature, and its orientation on the image plane, namely

$$c(q) := \begin{bmatrix} \pi \circ r(q) \\ N\left(D\pi|_{r(q)}\ v(q) \right) \end{bmatrix} \tag{7}$$

where N normalizes the vector on the image plane, and π is a perspective projection camera, $\pi : \mathbb{E}^3 \to \mathbb{E}^2$, $\pi(p) = [\ p_1/p_3,\ p_2/p_3,\ 1\ ]^T$.

The details of the visible set, $\mathcal{V}$, are given in [2], but intuitively it is the set of joint angles q, which keep the paddle facing the camera and in the field of view. Hence, the function c yields position and orientation on the image plane, *i.e.* $c : \mathcal{V} \to \mathcal{Y} = T_1\mathbb{E}^2$. Using the symbolic algebra package Mathematica, we have a relatively simple closed-form expression for c, and its Jacobian $D_q c(q)$.

The edge of the image plane is defined in terms of the "upper right corner" $y^+ \in \mathbb{R}^2$ and the "lower left corner" $y^- \in \mathbb{R}^2$. The image space is defined

$$\mathcal{J} := \left\{ (y, \theta) \in T_1\mathbb{E}^2 : y_i^+ < y_i < y_i^-,\ i = 1, 2 \right\}.$$

Following [2], under the right assumptions the map c (7) is shown to be a diffeomorphism $c : \mathcal{V} \approx \mathcal{J}$.

A compact manifold with boundary this is found by taking the inverse image under c of a compact subset of the image plane, namely

$$\mathcal{D} = c^{-1}(\mathcal{I}_\rho) \subset \mathcal{V} \tag{8}$$

which is a compact manifold with boundary.

By letting g be a simple affine scale and shift of the coordinates, the set $\mathcal{I}_\rho$ is diffeomorphic to $\mathcal{Z} = [-1, 1]^2 \times \mathrm{S}^1$. Let (z, ζ) denote coordinates on $\mathcal{Z}$. The function

$$\widetilde{\varphi} := \frac{\overline{\varphi}}{1 + \overline{\varphi}}, \quad \text{where}$$
$$\overline{\varphi}(z, \zeta) := \tfrac{1}{2} f(z)^T K f(z) + \kappa(1 - \cos(\zeta - \zeta^*)), \tag{9}$$
$$\text{and} \quad f(z) = \begin{bmatrix} \frac{z_1 - z_1^*}{(1 - z_1^2)^{\frac{1}{2}}} & \cdots & \frac{z_n - z_n^*}{(1 - z_n^2)^{\frac{1}{2}}} \end{bmatrix}^T.$$

for $n = 2$ is a navigation function on $\mathcal{I}_\rho$. Hence $\varphi = \widetilde{\varphi} \circ c$ is a navigation function on $\mathcal{D}$.

B. Example 2: Spatial body, spatial camera

In this example, we consider a convex polygonal rigid body, with a coplanar array of features on one face. A model space for the unoccluded scene is $\mathcal{Z} = [-1, 1]^5 \times \mathrm{S}^1$ (the details of the camera map and coordinate constructions are given in [2]). The navigation function for this case, is the same as that given in the previous example, with $n = 5$. Unlike the first example, the control law is performed directly in the task space – the only distinction being that the change of coordinates for this case includes a copy of the inverse camera map. In the other two cases, c^{-1} was unnecessary, because the image plane was a simple change of coordinates away from the model space. In this case, as outlined in [2], the transformation is more cumbersome (though straight forward).

IV. Empirical Validation

In order to test the framework proposed in Section III we experimented with two robotic systems that implement the latter two imaging models introduced in the previous section. The first system is the custom 3DOF direct drive Buehgler Arm described in Section III-A to test a fully dynamical controller (3) based on the NF given by (9). Our second set of experiments employ an industrial 6DOF RTX robot from Universal Machine Intelligence to test a kinematic controller (6) using the spatial 6DOF NF (9).

A. Calibration

Although the visual servoing methodolgy confers robustness against parameter mismatch in practice, all such methods, including the one presented in the paper, require at least coarse calibration of the robot and the camera. For the RTX, we used the manufacturer specified Denavit-Hartenberg parameters and a linear method ([6], Section 3) that requires a set of point correspondences between points in space and their respective image to simultaneously estimate both intrinsic and extrinsic camera parameters. To obtain the correspondences, a feature affixed to the robot end effector was moved to a grid of positions in view of the vision system which extracted an image plane location for each feature position in space. For the Buehgler setup we measured the paddle length ϱ and the shoulder offset δ by hand and proceeded as for the RTX to obtain a rough estimate of the camera parameters. A gradient algorithm based on a simple pixel disparity cost function refined our parameter estimates for the 11 camera and two robot parameters.

B. The Buehgler Arm

The Buehgler Arm is controlled by a network of two Pentium II class PCs running LynxOS (http://www.lynx.-com/), a commercial real-time operating system. The two nodes communicate on a private ethernet using the User Datagram Protocol (UDP). The first captures 8bit 528x512 pixel images at 100Hz using an Epix (http://www.epixinc.-com/) Pixci D frame grabber connected to a DALSA (http://www.dalsa.com/) CAD6 high-speed digital camera. The images are processed to extract the location (position and orientation) of the feature point at the end of the paddle. The second node implements servo control using the Trellis (http://www.trellissoftware.com/) motion controller with a servo rate of 1kHz, based on the dynamical controller in (3), wherein the damping term is computed from encoder data using finite differencing to estimate the joint velocities.

Two sets of experiments were conducted using the appropriate NF (9), implemented with two different gain settings (*i.e.*, assignments for the parameter array K in (9) and K_d in (3)) chosen to contrast performance resulting from a locally well tuned critically damped closed loop using relatively high gains, as against a "detuned" low gain and underdamped circumstance. Each trial consisted of driving the feature position and orientation to a goal (z^*, ζ^*) from some initial condition in joint space $(q_0, \dot{q}_0)$. For the "tuned" gain experiments, a set of 8 goal locations with and 40 initial conditions were chosen in an effort to "defeat" the controller. In particular, initial configurations were chosen near the edge of the FOV, with initial velocity vectors chosen so as to drive the robot out of the FOV. The initial conditions were prepared with a simple joint-space trajectory planner and joint-space PD controller that drove the robot to the starting state at which time the control switched to the NF based controller. In other words, we forced the robot to literally "fling" itself toward an obstacle before turning on our visual servoing controller. Both the goal positions and initial conditions where chosen to span the visible robot workspace. The control law gains were hand-tuned to provide nearly critically damped performance, and settling times on the order of a second.[*] For the "detuned" gain experiments, a smaller set of more aggressive initial conditions and goal locations was used, and the damping gain was reduced to provide "underdamped" performance. There were 4 goals and 8 initial conditions. Figure 2 shows the the error coordinates of a typical run for both "tuned" and "detuned" gains.

To quantify the results we examine similar measures as for the RTX. Table I summarizes the results of the our experiments. With well tuned gains the controller consistently drove the feature to the goal location with a rate of success of 97%. Of the 11 errors one was due to exceeding the robot's maximum velocity, one to a software driver error, and one to a feature leaving the FOV of the camera during initialization. The remaining 8 failures were caused by not allowing enough time for convergence as each experiment lasted 6 seconds. These errors generally arose when the robot was close to a saddle of the NF so the controller was slow to overcome the robot's unmodeled friction. However, with "detuned" gains and high initial velocity the feature left the FOV 25% of the time. These failures are due to the fact that the initial energy of the

[*]Of course, the allusion to linear notions of damping is merely an intuitive designer's convenience. We chose gains to ensure the local linearized system was critically damped at the eight equilibrium states, and then tuned up the "boundary" gains to force reasonably snappy descent into the domain wherein the linearized approximation was dominant.

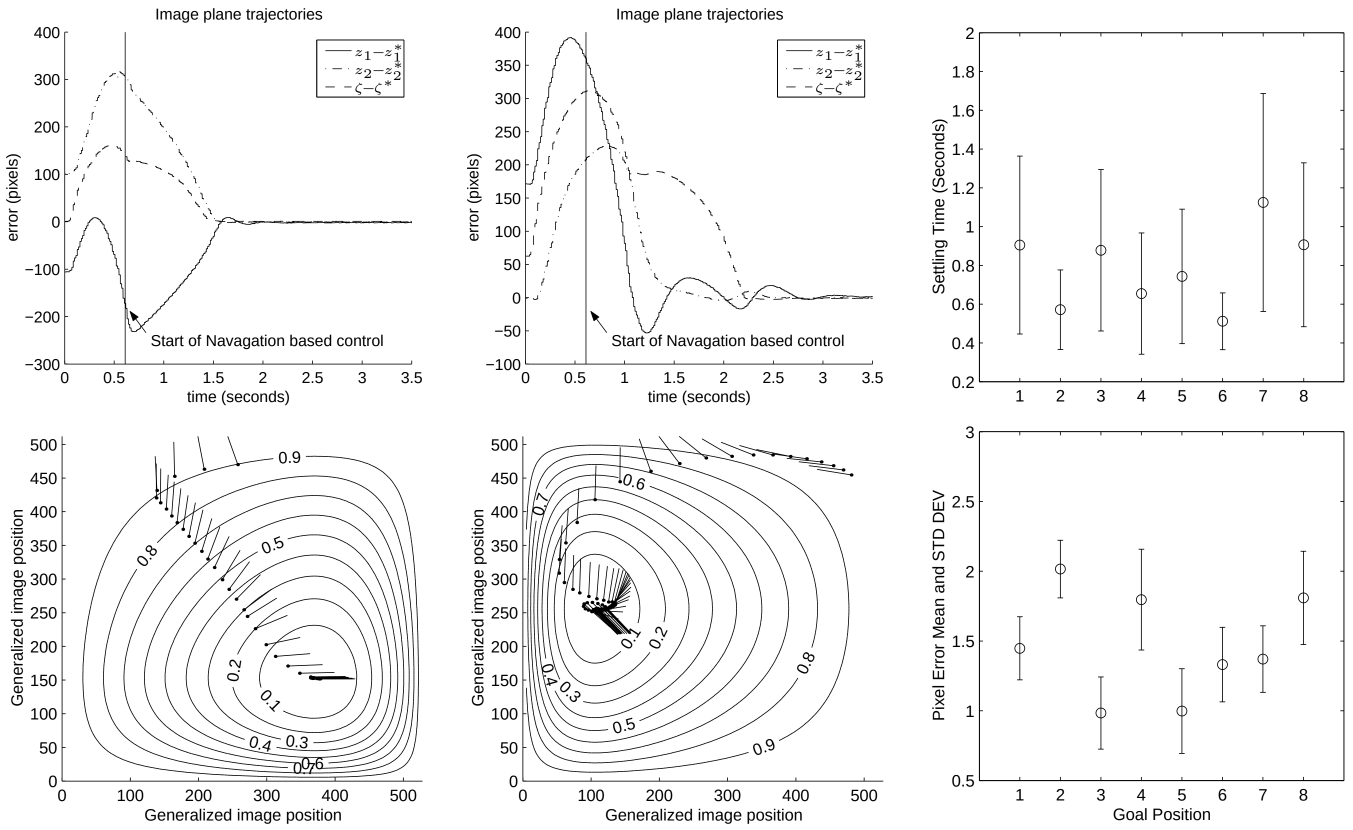

Fig. 2. **Left:** The pixel error and corresponding image plane feature trajectory for a typical high gain trial on the Buehgler robot. **Middle:** A typical low gain trial, with a different initial and goal locations. A two-dimensional cross section (with $\theta = \theta^*$) of the levels sets of the NF is superimposed on the image plane trajectory. **Right:** Buehgler convergence results. *Top:* Five percent settling time for each of the eight high-gain goal positions. *Bottom:* The mean pixel error for each of the eight goal positions.

TABLE I
Summary of results for Buehgler arm.

		Normalized Path Length		
Goal #	Succ. Rate	Jnt. Space Mean (dev)	Pix. Space Mean (dev)	Gains Tuned?
1	40/40	1.75 (0.12)	1.63 (0.19)	Yes
2	40/40	2.38 (0.34)	1.85 (0.35)	Yes
3	36/40	2.02 (0.87)	1.65 (0.22)	Yes
4	40/40	2.07 (0.36)	1.94 (0.50)	Yes
5	40/40	1.79 (0.36)	1.64 (0.23)	Yes
6	37/40	2.15 (0.41)	2.00 (0.62)	Yes
7	36/40	1.96 (0.85)	1.63 (0.19)	Yes
8	40/40	2.05 (0.65)	2.02 (0.75)	Yes
1	6/8	3.01 (1.93)	3.59 (1.94)	No
2	6/8	1.64 (0.40)	2.41 (0.53)	No
3	5/8	2.54 (1.86)	3.91 (2.40)	No
4	7/8	2.97 (0.46)	3.30 (0.65)	No

robot arm caused the arm to escape the potential well – by using a lower "detuned" gain on the potential energy feedback term, the potential barrier is reduced. (It would not be difficult to compute the invariant domain, as in [10] - these experiments give some sense of the relatively graceful performance degradation consequent upon imperfectly tuned gains.) Figure 2 shows image-based error plots and the image-plane trajectory for two typical runs.

To determine the accuracy with which the feature reached the goal, the mean pixel error is given by

$$256\sqrt{(z_{\text{final}} - z^*)^T(z_{\text{final}} - z^*) + (\tfrac{1}{\pi})^2(\zeta_{\text{final}} - \zeta^*)^2}$$

so the model coordinates are scaled to be commensurate with pixel error. As can by seen in Figure 2 (bottom right) the mean errors are in the neighborhood of 1 to 2 pixels over each of the eight goal positions.

Table I shows the path length taken by the robot compared to the straight line path length in both joint and pixel space. The results indicate that in joint space our navigation function produced results within a factor of 2.5 of the straight line distance. We attribute a large portion of this additional path length to the fact that the large initial velocities caused "curviness" to the trajectories.

Finally, we designed our controller to have a very rapid and dextrous response. The Buehgler arm has a mass in excess of 100Kg making precise, quick and efficient movement quite challenging. Figure 2 (top right) shows our navigation based controller produced a one second or less five percent settle time for seven of the eight primary goal positions.

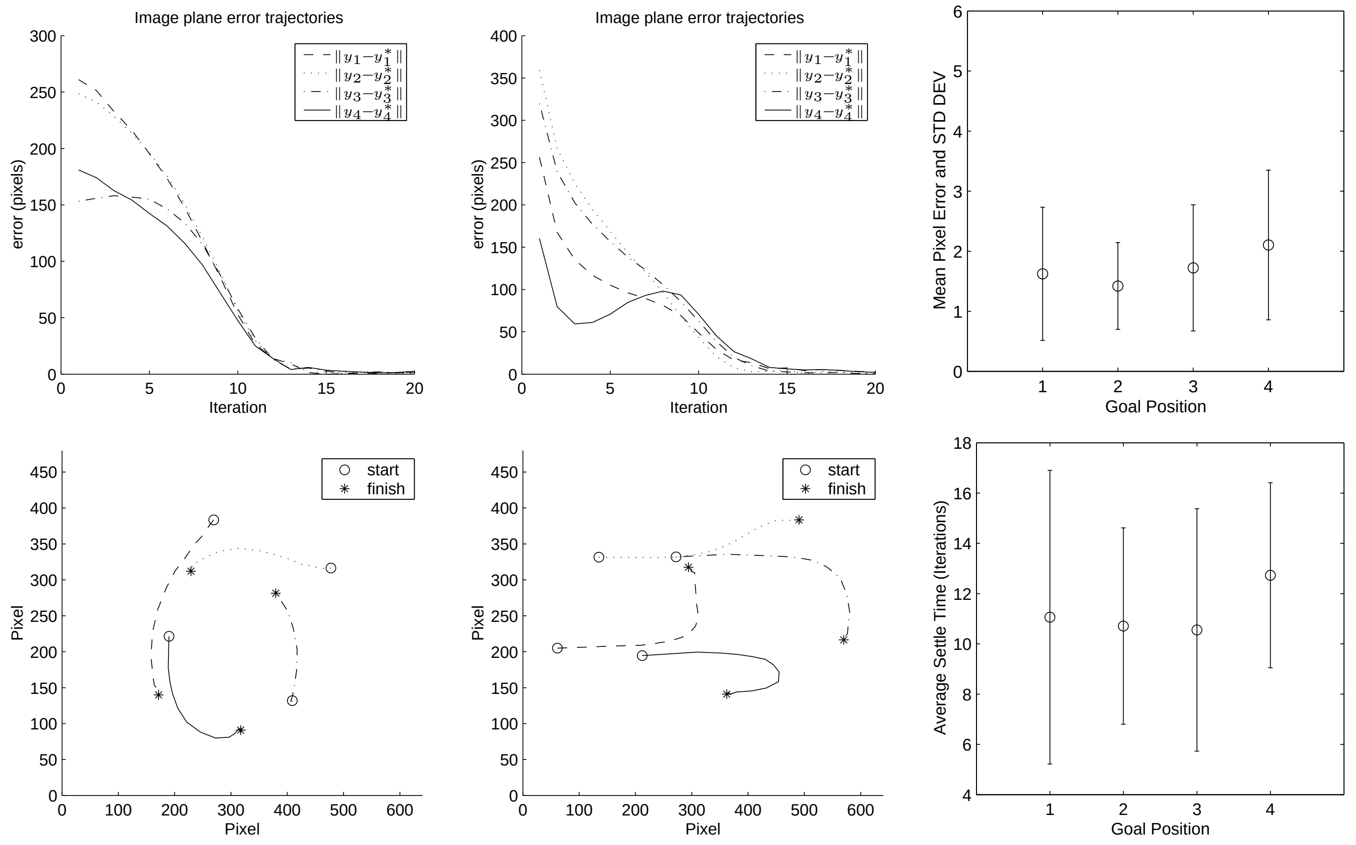

Fig. 3. **Left:** The pixel error and corresponding image plane feature trajectory for a typical trial. **Middle:** Another typical trial, with a different initial condition and goal location. **Right:** RTX convergence results. *Top:* The mean pixel error for each of the four goal positions. *Bottom:* Five percent settling time for each of the four goal positions.

C. The RTX Arm

The RTX is commanded through the serial port of a single Pentium PC running a Linux 2.0 kernel (hard real-time is not required, hence the standard Linux kernel was adequate). The PC is equipped with a Data Translations[†] DT3155 frame grabber connected to a standard 30Hz NTSC video camera. Using MATLAB's *C*-language API, we created a simple interface to the camera and robot accessible from within the MATLAB programming language.

The theory presented in Section III-B presumes the configuration space to be $\mathcal{Q} = \mathrm{SE}(3)$. However, $\mathcal{Q}$ is parameterized (locally) by the robot joint angles $q \in \mathbb{R}^6$ through the forward kinematics, namely $h : \mathbb{R}^6 \to \mathcal{Q}$. Of course, inevitably, all such kinematic parameterizations introduce singularities that may, in turn, inject spurious critical points to the gradient fields, necessarily actuated in the robot's joint space rather than in the task space, as our theory presumes. Similarly, since our formal theory "knows" only about visibility bounds, the robot's unmodeled joint space angles limits are not in principle protected against.[‡] However, the weight of experimental evidence we present below suggests that these discrepancies between presumed model and physical reality do not seriously imperil the practicability of this scheme. Regarding the first discrepancy, the absence of stalled initial conditions suggests that any critical points so introduced were not attractors. Regarding the second, we found that choosing initial and goal locations away from the joint space boundaries was sufficient to avoid running into the end-stops.

The RTX controller employs first order gradient descent on the navigation function in (9). Because the RTX arm accepts only position commands, given goal and current images with feature points extracted, the gradient update was implemented iteratively, as follows:

$$u_k \Leftarrow -D_q^T \varphi = -D_q^T(\overline{g} \circ h)(q_k)\, D_z^T \widetilde{\varphi}_{z^*}(z_k),$$
$$q_{k+1} \Leftarrow q_k + \alpha u_k \text{ (where } \alpha \text{ is the step size).}$$

Note that the Jacobian matrix $D_q(\overline{g} \circ h)$ can be decomposed into the product of $D\overline{g}$, which maps from the body screw axis to the model space, and the manipulator Jacobian Dh, which maps from the robot joint space to the body screw axis. Indeed, such a decomposition implies the extrinsic parameters are not needed and hence one may move the camera without recalibrating.

[†] `http://www.datax.com/`

[‡] Addressing the further practical realities of kinematic singularities and robot joint space limitations falls outside the scope of the present paper (and, indeed, is not even addressed at all in the traditional visual servoing literature). In principle, the NF framework would be relevant to these problems as well: joint space limits are analogous to the FOV obstacles, while the kinematic singularities are akin to self-occlusion. Again, we stress that a careful treatment of these ideas lies far beyond the scope of the present paper.

To explore our algorithm, we conducted a set of experiments in which 58 initial conditions were tested for four goal locations, giving 232 candidate experiments. Both the initial conditions and goal locations were chosen randomly from a grid of 4096 points in model space (configurations near kinematic singularities and not within the robot workspace were removed). We chose many initial and goal configurations near the boundaries of the workspace, hence of the 232 candidate experiments, an additional 29 experiments where removed as the features began outside of the robot's workspace due to "sloppiness" in the joints of the robot *i.e.* there is a lot of play in the joints so a specific initial condition in the joint space may not be the same place in SE(3) in subsequent trials, and therefore some initial conditions were out of the visible workspace at the start. Initially, the robot was moved to each goal location to capture an image of the robot, respecting which the vision system stored the desired location of feature points, y^*. Figure 3 shows the pixel errors feature trajectories of two typical runs. As shown, we used four coplanar feature points for the camera map, $c : \mathcal{Q} \rightarrow \mathcal{Y}$.

TABLE II
Summary of results for RTX arm

		Normalized Path Length	
Goal #	Success Rate	Jnt. Space Mean (dev)	Pix. Space Mean (dev)
1	49/51	1.35 (0.39)	1.33 (0.49)
2	47/47	1.36 (0.29)	1.27 (0.37)
3	55/57	1.32 (0.23)	1.27 (0.32)
4	47/48	1.42 (0.36)	1.32 (0.25)

To ascertain the performance of our controller, we employed several metrics described below: success and failures, efficiency of motion, mean pixel error and setting time.

Table II shows the success rate of the various goal positions. Of 203 trial runs, 5 were found to have failed. All 5 failures are due to the robot not converging in our limit of 30 iterations (though after inspecting the data by hand, it appears that the robot would have converged if given a few more iterations).

We also measured the "efficiency" of motion relative to the straight line distance in both image and cartesian space. The metric used for measuring distances in the configuration space was the sum of the angular displacement, scaled by body length, and the translational displacements. For all of the runs, both image and cartesian measures indicated that the path length was around 1.4 times that of straight line distance. See Table II.

Using the root mean squared average pixel error measurement given by $\sqrt{\frac{1}{4}\sum_{i=1}^{4}(y_i - y_i^*)^2}$ we found an average final pixel error on the order of 1-2 pixels upon convergence. Figure 3 (upper right) shows the mean pixel error and standard deviation for each of the four unique goal positions.

The average five percent setting time, shown in Figure 3 (lower right), was approximately 10-14 iterations for each of the four goal locations, averaged over all successful runs.

V. Conclusions

Using two different experimental platforms, we demonstrate the applicability of a new framework for visual servoing [2] designed to drive image plane features to some goal constellation while guaranteeing their visibility during transients. Both systems confirmed the practicability of theoretical framework: the custom 3DOF Buehgler Arm and the 6DOF commercial RTX arm. For the Buehgler, our experiments suggest that the navigation function based controller indeed drives the feature to within a few pixels of the goal in all cases where the initial energy did not overwhelm the potential energy well. The kinematic experiments with the RTX validated our 6DOF task-space servo architecture. In both cases our results show systems with large basins of attraction that both avoid self occlusion and respect FOV constraints.

References

[1] Peter I. Corke and Seth A. Hutchinson. A new hybrid image-based visual servo control scheme. In *Conference on Decision and Control*, volume 1, page 2521, Sydney, Australia, 2000. IEEE.

[2] Noah J. Cowan. *Vision-based Control via Navigation Functions.* PhD thesis, University of Michigan, 2001.

[3] Noah J. Cowan and Daniel E. Koditschek. Planar image based visual servoing as a navigation problem. In *International Conference on Robotics and Automation*, volume 1, pages 611–617, Detroit, MI, 1999. IEEE.

[4] Noah J. Cowan, Gabriel A. D. Lopes, and Daniel E. Koditschek. Rigid body visual servoing using navigation functions. In *Conference on Decision and Control*, pages 3920–3926, Sydney, Australia, 2000. IEEE.

[5] J. Craig. *Introduction to Robotics.* Addison-Wesley, Reading, Mass., 1986.

[6] Olivier Faugeras. *Three-Dimensional Computer Vision.* MIT Press, London, England, 1993.

[7] Robert M. Haralick and Linda G. Shapiro. *Computer and robot vision*, volume I and II. Addison-Wesley, 1992.

[8] Morris W. Hirsch. *Differential Topology.* Springer-Verlag, 1976.

[9] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, pages 651–670, October 1996.

[10] D. E. Koditschek. The control of natural motion in mechanical systems. *ASME Journal of Dynamic Systems, Measurement, and Control*, 113(4):547–551, Dec 1991.

[11] Daniel E. Koditschek. The application of total energy as a Lyapunov function for mechanical control systems. In *Dynamics and control of multibody systems (Brunswick, ME, 1988)*, pages 131–157. Amer. Math. Soc., Providence, RI, 1989.

[12] Daniel E. Koditschek and Elon Rimon. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics*, 11:412–442, 1990.

[13] Ezio Malis, Francois Chaumette, and Sylvie Boudet. 2-1/2-d visual servoing. *IEEE Transactions on Robotics and Automation*, pages 238–250, 1999.

[14] Elon Rimon and D. E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, Oct 1992.

[15] Alfred A. Rizzi, Louis L. Whitcomb, and Daniel E. Koditschek. Distributed real-time control of a spatial robot juggler. *IEEE Computer*, 25(5):12–24, May 1992.

[16] Camillo J. Taylor and James P. Ostrowski. Robust visual servoing based on relative orientation. In *International Conf. on Robotics and Automation*, 1998.

[17] Hong Zhang and James P. Ostrowski. Visual servoing with dynamics: Control of an unmanned blimp. In *International Conf. on Robotics and Automation*, 1999.