A Control Basis for Visual Servoing Tasks*

T. J. Schnackertz Jr.   R. A. Grupen

Laboratory for Perceptual Robotics
Department of Computer Science
University of Massachusetts, Amherst, 01003

Abstract

This article discusses a set of three controllers that can be used as a basis for visual servoing tasks. These controllers map image plane feature errors to actuator commands through visuomotor Jacobians. It is shown how unary and binary image plane constraints can be used to specify visual servoing tasks. A specific visual servoing task is described and the experimental results are presented.

1 Introduction

In the recent literature, many authors have presented work using visual information as feedback for controllers [7, 4, 1, 2, 5, 6]. Several of these systems are designed around a specific task and may be difficult to generalize. Weiss *et al* [7] and Feddema *et al* [4] predefine absolute feature goal states and precompute feature dependent Jacobians. Castano *et al* [3] constrain the motion of an end effector by a feature reference in what they call visual compliance.

This work develops robust vision-based sensory feedback controllers for a general purpose visual servoing system. The controllers use image plane information to derive feedback signals, mapping these error signals directing to actuator commands. It suggests two image plane feature constraints that can be used to solve complex visual servoing tasks. It also explores strategies for composing these visuomotor controllers to solve complex visual and motor tasks.

2 Visual Servoing

The essence of visual servoing is the use of visual information to derive control signals for articulated devices. The degrees of freedom under the control of the visual servoing system can result in egomotion of the camera, motion of an independent manipulator [3], or both [6]. The visual information used to derive the control inputs is extracted from the image sequence. Any uniquely definable feature can be used, from pixel intensities to abstract features like corners and doors.

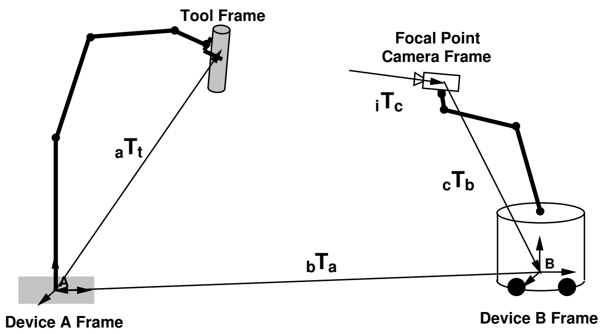

Figure 1: General purpose visual servoing system.

Figure 1 depicts a general purpose visual servoing system. Both the camera and the object of interest are capable of Cartesian translations and rotations. The kinematic relationship between the camera and the world geometry is defined by the configuration of the intervening mechanical degrees of freedom. To control this type of system the Jacobian that maps object velocities to image feature velocities must be known or learn-able. This system Jacobian is computed from the transformation from the image plane to the tool frame.

$$_iT_t = {_iT_c} \cdot {_cT_b} \cdot {_bT_a} \cdot {_aT_t}$$

A specific visual servoing task is defined by selecting a set of relationships between the projection of 3D features onto the image plane and establishing a mapping of these relationships to actuators through the system Jacobian. For more complex tasks, an activation policy can be used to dynamically change the feature to actuator bindings.

*This work is supported in part by NSF CDA-8922572, IRI-9116297, and IRI-9208920.

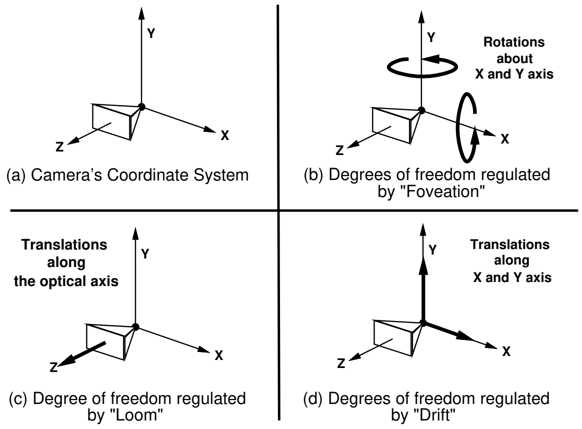

Figure 2: Camera frame and degrees of freedom controlled by each visuomotor controller.

3 Visuomotor Controllers

Controllers that take image plane data and map it directly to actuator commands are referred to as visuomotor controllers. In this section three visuomotor controllers are developed. Each derives its feedback signal from image plane features and maps this error to differential motions in the system. Rotations about the camera's optical axis, torsional motions, have been shown to be useful for changing the zero disparity surface in a stereo system, but they will not be considered in this work.

Figure 2 illustrates the degrees of freedom supervised by each of the controllers. The first controller termed "foveate", controls the rotations about the $\hat{x}$ and $\hat{y}$ axes of the camera. The second, entitled "loom", controls translations along the optical axis. The third, "drift", controls translations in the plane parallel to the image plane that passes through the focal point.

This section describes two parametric image plane references that can be used to derive the feedback signals for the visuomotor controllers. Each feature defines a constraint on the pose of the camera, or the location of 3D object. The combination of these features defines a new constraint which is equivalent to the intersection of the individual feature constraints.

3.1 Unary Image Plane Constraint

With a single image feature, the orientation of an active camera can be constrained by requiring that the feature remain at a fixed location on the image plane. Under this unary feature constraint, every camera position has a unique orientation that places the feature at the desired image location. If the camera maintains a unary feature constraint at the center of the image plane, lateral movements will generate a path along a constant radius sphere. Therefore, the unary constraint in conjunction with lateral movements constrains the camera to one of the concentric spheres centered at the object. Whereas movements along the optical axis change the constraint sphere. Figure 3 shows a 2D example. Here the active camera is shown at several positions with unique orientations defined by the unary constraint.

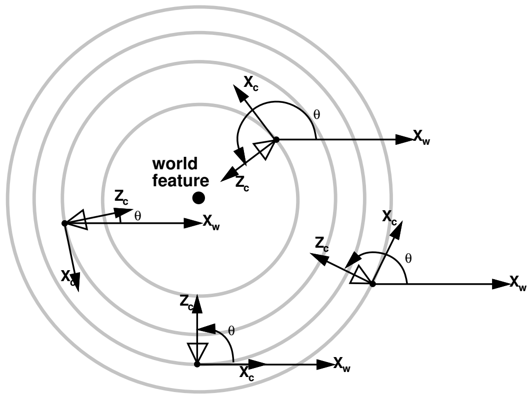

Figure 3: 2D examples of unary feature constraint. Every camera location (c_x, c_z) has unique orientation θ.

This unary feature constraint is used to derive the error signal for the foveation controller. This controller is designed to stabilize the location of an image plane feature through rotations about the camera's x and y axes. Foveation receives image plane errors, $\Delta\vec{i}$, which are weighted by a gain G and then transformed by the system Jacobian to actuator commands. G is dynamic, and dependent on $\vec{i}$. It's value is at a minimum when the foveation error is above a set threshold. As the error approaches zero, G scales linearly to a maximum.

The system Jacobian, J_s, is the composition of the inverse manipulator Jacobian, J_b^{-1} and foveation's image Jacobian, J_i, which maps the image plane errors to rotational errors about the $\hat{x}$ and $\hat{y}$ axes of the camera. The computations for the foveation error, the value of G and the rotation commands are as follows:

$$
\begin{aligned}
e &= \sqrt{i_x^2 + i_y^2} \\
G &= \begin{cases} g_{min} & e \geq e_{max} \\ g_{max} - ((g_{max} - g_{min})\frac{e}{e_{max}}) & e < e_{max} \end{cases} \\
\Delta\vec{\theta} &= J_s(G\Delta\vec{i})
\end{aligned}
$$

where:

$\Delta\vec{i}$	(i_x, i_y), image plane error vector,
$g_{min,max}$	min and max gain values,
e_{max}	max distance threshold,
$\Delta\vec{\theta}$	manipulator command,
J_s	system Jacobian.

3.2 Binary Image Plane Constraint

With the addition of a second image plane feature, we can create a binary feature constraint by requiring that the image plane distance between the two features is k. Figure 4 shows the set of points that satisfy this constraint in 2D and 3D. The 3D surface is generated by rotating the 2D arc around the line passing through the 3D features. The camera is free assume any position on the 2D surface in Figure 4(b).

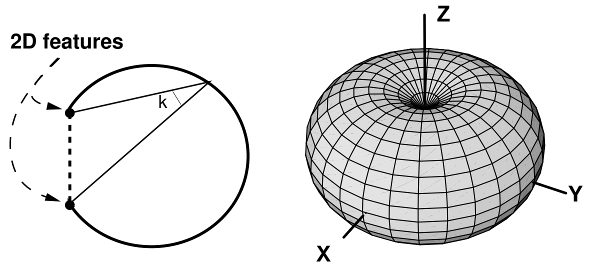

Figure 4: Camera constraint surfaces produced by the binary constraint.

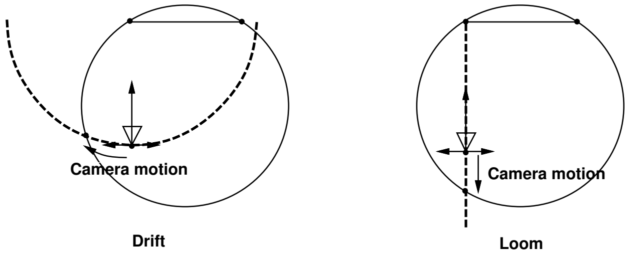

Figure 5: 2D camera motion for lateral drift or loom when satisfying a binary constraint.

Figure 5 demonstrates lateral drift and loom controllers converging to a binary constraint. The control error signal for the lateral drift controller is the difference between the actual and reference distances on the image plane between two image features, Δk. With continuous foveation, motions generated by the drift controller to satisfy the Δk reference error move th camera along a circular arc until it lies on the corresponding binary constraint curve. The particular circle that it moves along depends on the camera's position when the controller begins. It is important that foveation is more responsive then either lateral drift or loom. If a system is designed in which foveation can not eliminate the disturbances of the other controllers, the features can not be guaranteed to remain in view.

The loom controller will move a foveated camera along a radial line extending from the foveated feature through the camera's focal point. The particular radial line depends on the camera's position when the controller begins execution.

The loom controller's error signal is the difference between the reference distance k and the actual image plane distance between the features, $\vec{i}_1$ and $\vec{i}_2$. The loom error is weighted by a gain and then transformed to actuator commands through the system Jacobian.

$$\begin{aligned} \Delta k &= k - \mathrm{dist}(\vec{i}_1, \vec{i}_2) \\ \Delta\vec{\theta} &= J_s(G\Delta k) \end{aligned}$$

where:

Δk	loom error - binary feature constraint error,
$\Delta\vec{\theta}$	manipulator command,
G	dynamic gain,
J_s	system Jacobian.

The image Jacobian for loom is a scalar constant, which maps loom error to a desired motion along the camera's optical axis. This approach does not require the knowledge of camera parameters or the distance to the 3D features. A large J_i can cause a qualitatively underdamped response in the camera motion. A small J_i will exhibit a qualitatively overdamped response and take a long time to converge.

Since the lateral drift controller supervises two orthogonal degrees of freedom, the desired feature distance k is broken into corresponding orthogonal components creating two new binary constraints, k_x and k_y. The lateral drift controller is really a composite controller. It will try to satisfy the binary constraint derived from k_x with translations along the camera's $\hat{x}$ axis. While simultaneously trying to satisfy the k_y constraint by $\hat{y}$ axis translations. The implementation of lateral drift is similar to that of loom. Drift's J_i is a constant scalar mapping of Δk's to translations and has the same selection problem.

$$\Delta\vec{k} = \vec{k} - (\vec{i}_2 - \vec{i}_1)$$

$$\Delta\vec{\theta} = J_s(G\Delta k)$$

where:

$\Delta\vec{k}$ drift error vector,

$\Delta\vec{\theta}$ manipulator command,

G dynamic gain,

J_s system Jacobian.

3.3 Visual Servoing System

Figure 6 shows a block diagram of the control architecture for the visual servoing paradigm used in this work. The system has four main components; the plant, the model reference, the PD controller, and the visuomotor controllers. The plant consists of a manipulator and a vision sensor. The model reference component contains the feed forward dynamic model of the plant which uses state information about the plant and the desired accelerations from the PD controller to generate torque commands for the motors. The PD controller is a standard position-derivative controller transforming desired position vectors into acceleration commands. The front end component, which is the focus of this work, contains the vision processing, the application, the visuomotor controllers, and the Jacobian that maps Cartesian commands from camera Cartesian space to the configuration space of the manipulator.

4 Example: Homing

This section develops a procedure for generating a sequence of visuomotor control activations that always yield unique postures of an endpoint vision system. Provided that this posture has a corresponding unique robot configuration, servoing the camera to the posture, uniquely determines the robot "home" configuration.

It will be demonstrated that three features, (f_1, f_2, f_3), can be used to constrain the camera to a unique pose. Feature f_1 is used for the unary feature constraint and defines a unique orientation for every camera location. Features f_1 and f_2 are used to create a binary feature constraint that is satisfied by the lateral drift controller. f_1 and f_3 are used to create a second binary constraint, which is used to establish a reference for the loom visuomotor controller.

To explore the responsiveness of the controllers, the system Jacobian that maps errors on the image plane, $\Delta\vec{i}$, to motion commands $\Delta\vec{\theta}$ is used to determine how a single pixel error maps to a motion command at different locations along a constraint curve. Both the lateral drift controller and the loom controller have a greater pose refinement ability as the camera approaches the world features. The loom controller is much more sensitive to the distance between the world features than to its position on the curve. The lateral drift controller however is just the opposite. Therefore, when choosing the features from which to generate the binary constraints it is important that the features for loom be close together and that the intersection of the constraint curves place the camera in a well conditioned part of drift's constraint curve. Figure 8 has an example of such a feature placement.

With this feature placement, the three features will define a unique pose for the camera. The two binary constraint curves intersect at a point which is a unique solution for the (x, z) position of the camera. The camera's orientation is unique due to the single feature constraint being satisfied by foveation. In order to move the camera to the unique pose defined by these constraints, all three controllers are executed concurrently.

In theory, as the interval between control actions approaches zero, $(\Delta t \to 0)$, the camera moves in a smooth continuous path through configuration space to the desired pose. In practice, to guarantee the camera reaches the pose, it is necessary to guarantee that foveation is more responsive then the other controllers with its error remaining close to zero. To this end, a controller activation policy has been developed. This policy is shown in Figure 7. The activation policy provides are convenient technique for handling the interactions of the controllers. It is conceivable that appropriate gains could be selected for the visuomotor controllers so that this policy could be approximated be executing the controllers concurrently, but this is not currently implemented.

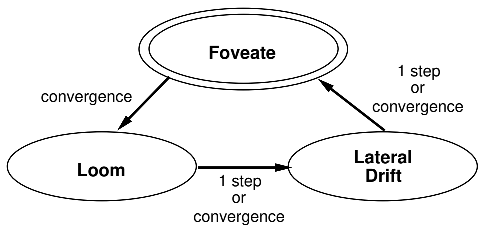

Figure 7: Activation policy for the visuomotor controllers.

This policy starts with the activation of the foveation controller that executes until the foveation error has been eliminated. Then, there is a transition to the loom controller. If the binary constraint established by f_1 and f_3 is not satisfied, the controller exe-

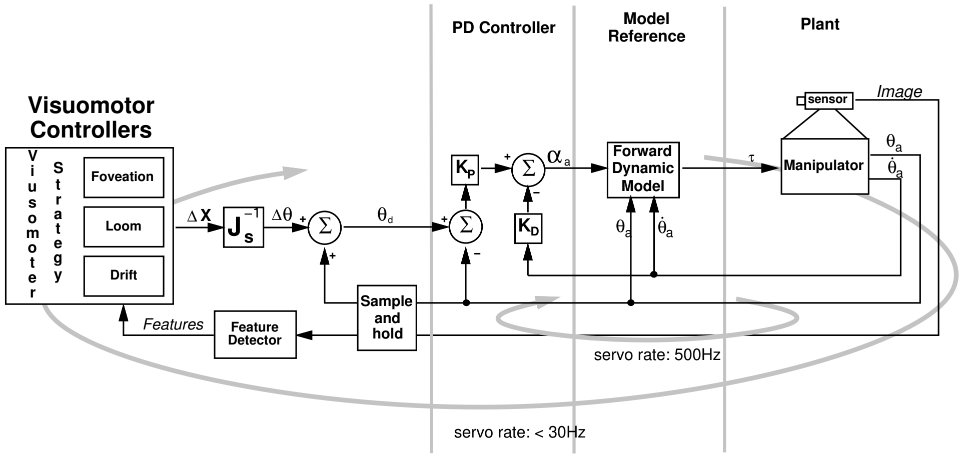

Figure 6: Block diagram of Animate Vision system under the control of the visuomotor controllers.

cutes a single control action. Since the loom controller only generates motion along the optical axis, the system remains foveated, (not always true in practice). Next, if the binary constraint established by features f_1 and f_2 is not satisfied, lateral drift executes a single control action. Lateral drift will perturb the foveation controller's equilibrium so it is activated next until it converges. The cycle continues until all three reference states are satisfied simultaneously. Figure 8 shows a possible sequence of moves.

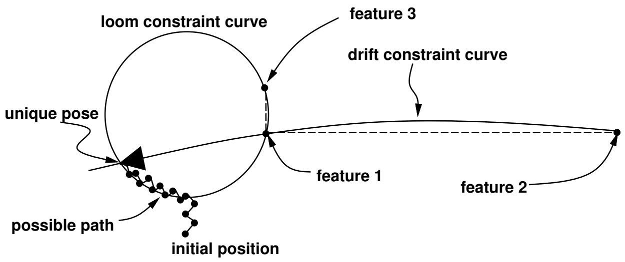

Figure 8: The constraint curves and a possible sequence of camera motions.

4.1 Experimental Results

The first experiment tests robustness of the controllers with respect to small kinematic errors of the manipulator. These types of errors are common, they are usually due to inaccurate link lengths or absolute positioning. The experiment was performed by first moving the camera to a position referred to as "home". Three features are then selected as discussed earlier and templates for feature detection are created. The templates, their current image plane positions and the current manipulator configuration are written to a file for later use. Figure 9 shows the camera's view at the home position along with the location of the three features.

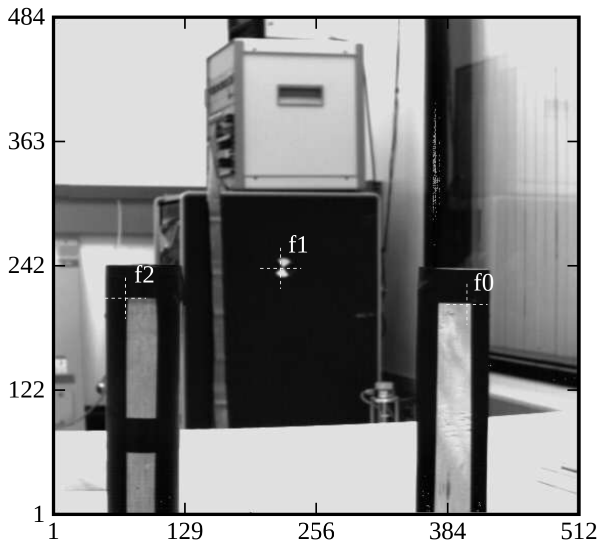

Figure 9: Camera view at "home" position and the selected features.

Once a "home" position has been defined, the manipulator is moved to a known position where the absolute joint angles are perturbed causing small kinematic errors similar to those that will occur if the

Joint Angles 0-4				
-1.8083	-0.2656	0.2702	-1.5780	0.0642
-1.7814	-0.2513	0.2776	-1.5478	0.1271
-1.8084	-0.2655	0.2707	-1.5774	0.0642
0.0001	-0.0001	-0.0005	-0.0006	0.000

Table 1: Results from homing experiment.

robot is incorrectly zeroed. The homing procedure was then initiated. After it's completion, the manipulator's absolute joint angles are reset to the joint angles previously recorded that correspond to this position. The manipulator was then sent to the known position and compared to the initial zeroing pose. Figure 10 shows the square root of the sum of the squares of the image plane errors for the controllers,

$$\sqrt{\Delta \text{foveation}^2 + \Delta(loom)^2 + \Delta(drift)^2}$$

for each complete loop of the activation policy. Some error remained due to the manipulators inability to remove small position errors.

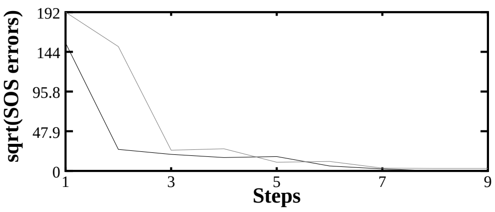

Figure 10: Square root of the sum of squared controller errors for two trials.

The second experiment tested for system's repeatability. In this experiment the home position and templates are established as before, the home position is then disturbed slightly and the homing procedure was executed. The results of this experiment are summarized in Table 1. The rows show the home configuration, the starting configuration, the configuration after executing the homing algorithm, and the resulting errors. The results of this experiment demonstrate that the system is consistently generating the same camera pose.

5 Conclusions and Future Work

The visuomotor controllers developed in this work provided a useful control basis for an active vision system. By combining the controllers with activation policies or another composition technique, a variety of vision tasks can be accomplished. The three controllers are capable of moving a disembodied camera to any Cartesian pose.

The homing application demonstrates how very simple controllers can be combined to accomplish a complex task. It also reinforces the view that vision can be understood in the context of the task the system is engaged in, and that generic vision systems that produce elaborate representations of the 3D world are not necessary.

The following is a list of possible extensions and modifications to be considered for future implementations of the system.

- Apply the set of visiomotor controllers to other vision tasks.
- Eliminate the need for independent system calibration. Use an auto-calibration technique to generate one transform to represent the manipulator kinematics and the camera parameters.

References

[1] Allen, P. K., Timcenko, A., Yoshimi, B., and Michelman, P. Hand-eye coordination for robotic tracking and grasping. In *Visual Servoing*. World Scientific, 1993, pp. 33–69.

[2] Allotta, B., Buttazzo, G. C., and Fanizza, F. P. Mousebuster: a robot system for catching fast moving objects by vision. In *International Conference on Robotics and Automation* (1993), pp. 932–937.

[3] Costano, A., and Hutchinson, S. Visual compliance: Task-directed visual servo control. *IEEE Transactions on Robotics and Automation 10*, 3 (1994), 334–342.

[4] Feddema, J. T., Lee, C. G., and Mitchell, O. R. Model-based visual feedback control for a hand-eye coordinated robotic system. *Computer 25* (1992).

[5] Nelson, B., Papanikolopoulos, N. P., and Khosla, P. K. Visual servoing for robotic assembly. In *Visual Servoing*. World Scientific, 1993, pp. 33–69.

[6] Papanikolopoulos, N., and snd P.K. Khosla Fanizza, B. N. Six degree-of-freedom hand/eye visual tracking with uncertain paramters. In *International Conference on Robotics and Automation* (1994), pp. 174–179.

[7] Weiss, L. E., Sanderson, A. C., and Neuman, C. P. Dynamic sensor-based control of robots with visual feedback. *Journal of Robotics and Automation* (1987), 404–417.