In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

A Hierarchical Vision Architecture for Robotic Manipulation Tasks

Zachary Dodds[1], Martin Jägersand[1], Greg Hager[1], and Kentaro Toyama[2]

[1]Department of Computer Science, Yale University, New Haven, CT 06520
[2]Microsoft Research, Redmond, WA 98052

Abstract. Real world manipulation tasks vary in their demands for precision and freedoms controlled. In particular, during any one task the complexity may vary with time. For a robotic hand-eye system, precise tracking and control of full pose is computationally expensive and less robust than rough tracking of a subset of the pose parameters (e.g. just translation). We present an integrated vision and control system in which the vision component provides (1) the continuous, local feedback at the required complexity for robot manipulation and (2) the discrete state information needed to switch between control modes of differing complexity.

1 Introduction

In *robotic hand-eye tasks* (which we will also refer to as *visual servoing* or *vision-based manipulation*), the rote motions of classical industrial robotics are replaced by a flexible control strategy which is robust to deviations in camera positioning, robot calibration, and placement of objects in the robot workspace. In large part, this robustness is due to feedback from vision systems which observe a robot action in progress, allowing for continual adaptation of robot motion.

Despite the importance of vision in these tasks, vision systems which support hand-eye robot coordination have generally been developed *ad hoc*, designed specifically for particular tasks. In this paper, we present a principled approach to vision and control system design which supports complex vision-based manipulation.

Robotic systems need to integrate a broad spectrum of sensor information to accomplish their tasks. Often, tasks decompose into manipulation skills of varying levels of specificity. In particular, there is a continuum of control motions ranging from simple, coarse motions to refined, precise movements. Likewise, there is an associated range of vision algorithms from observations that are low-dimensional and approximate to those which are multi-dimensional and precise. A key to solving robotic hand-eye tasks efficiently and robustly is to identify how precise control is needed at a particular time during task execution, and then match that with appropriately precise visual tracking, Fig. 1. For instance during the reach motions of a basic pick-and-place task simple vision and control

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

suffices, while for the pickup and place typically precise fine manipulation and visual estimation of full 6D pose are needed.

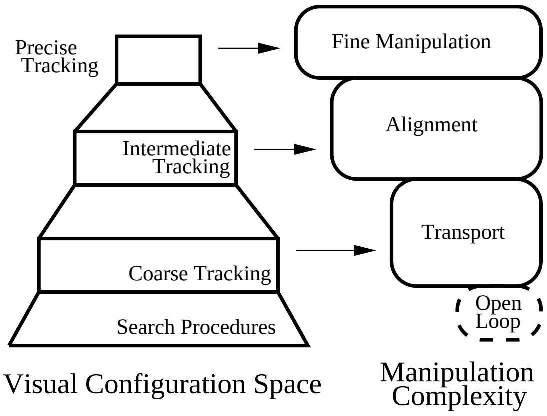

Fig. 1. The vision hierarchy and corresponding supported manipulation skills. As the set of possible target configurations narrows, the control system can perform more complex actions.

In this paper, we claim that an effective vision-system architecture for hand-eye coordination can be distilled into three key ideas:

- A control framework for hand-eye manipulation which manages the complexity of tasks through the composition of a few simple primitives that can be intuitively manipulated.
- A complementary set of fast image-processing primitives which can be combined to support the requirements of hand-eye tasks.
- Integration of the control and vision frameworks in such a way that both planned and exogenous events are handled by discrete transitions among the servoing and vision modules. Because hand-eye tasks are dependent on vision, failure of the vision system to deliver certain observations will affect the choice of robot action.

In Section 2, we briefly review related work and present background for visual servoing and vision-based tracking. Section 3 extends these low-level capabilities through temporal composition to construct more sophisticated tasks which adapt to visual events. Finally, an implementation of the system and some experimental results are presented in Section 4.

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

2 Background

Rapid strides in technology and a great deal of research interest have yielded progress on both theoretical and applied aspects of robotic hand-eye coordination [1, 6, 8, 13, 15, 22, 9, 24, 31, 37, 33, 41, 40, 45, 47, 46]). For a recent survey see [11]. Notable recent advances include the manipulation of automotive parts under realistic conditions [42], visual learning of robotic behaviors [3], and techniques for fusing 2D and 3D information [7]. Work on visual tracking for real-time applications has also proliferated [2, 4, 10, 12, 27, 28, 32, 34, 39, 43]. Some of this work considers task composition and discrete event systems to link active-vision modules, *e.g.*, [29, 30, 14].

In our own work, we have begun the development of modular systems for hand-eye coordination and visual tracking [19, 21, 25, 26]. Our work differs from other work in several ways. First, we emphasize a modular approach where task primitives are defined for simple subtasks that occur repeatedly in the domain. These primitives are composed to perform complex task via consistent design principles, where composition can occur both in time and "in space." Throughout, we strive to maintain an intuitive representation of primitives so that implementations follow simply from task specifications. Finally, the novel contribution of this paper considers the interaction between analogous components of control and vision.

In this section, we review our approach to visual servoing and visual tracking. We will consider task primitives for each domain and explain non-temporal compositions of primitives.

2.1 Visual Servoing

An Example Task Consider a gluing task: a robot must manipulate a glue-filled syringe such that the nozzle reaches a designated application point and the syringe is at a prescribed orientation. Both constraints must be satisfied within a given error tolerance before the actual gluing can begin. Because the exact geometric relationships between the robot, tool, and surface may be unknown, rote actions by the robot are not acceptable. One option is to continually reconstruct the 3D scene using vision, and proceed until the reconstruction fits the goal configuration. For computational efficiency, however, we consider another approach whereby the task is described as a set of sensor constraints which *imply* the desired goal. In our example, the goal configuration may be described as one in which the nozzle tip appears to coincide with the glue-application point simultaneously in two cameras.

As the example illustrates, the fundamental goal of visual servoing is to achieve a set of constraints on the configuration of an actuated system. Constraints can be described in terms of a set of salient features in the robot workspace – that is, by an equation

$$T(p) = 0,$$

where p is a list of observable features on the robot or in its environment and T is a *task function* [38]. In the example above, p could contain the 3D coordinates

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

of the nozzle and the target point. T measures a configuration's error, vanishing when the task is complete.

The information available to perform a task are the observations of the features, p, by one or more cameras. Let $\mathbf{y} \in \Re^m$ denote the available observations of the features p. Assuming $\mathbf{y}$ is the only information available for performing a task $T(p) = 0$, then if the task can be performed at all, it must be the case that there is an equation of the form

$$\mathbf{E}(\mathbf{y}) = 0$$

such that

$$\mathbf{E}(\mathbf{y}) = 0 \Leftrightarrow T(p) = 0. \tag{1}$$

We refer to $\mathbf{E}(\mathbf{y}) = 0$ as an *image-based encoding* of the task $T(p) = 0$ and we say that $\mathbf{E}(\mathbf{y}) = 0$ *verifies* the task $T(p) = 0$ if Equation 1 holds [23].

Carrying out a given visual servoing task involves finding a robot configuration which achieves the encoded task. Let $\mathbf{x} \in \Re^n$ represent the robot state, and let $\mathbf{f}$ be the *visual-motor model*, which expresses image-feature values as a function of the robot's state: $\mathbf{y} = \mathbf{f}(\mathbf{x})$. In a precisely calibrated system one can solve the equation

$$\mathbf{E}(\mathbf{f}(\mathbf{x}^*)) = \mathbf{E}(\mathbf{y}^*) = 0$$

for a goal position $\mathbf{x}^*$. In an uncalibrated system, $\mathbf{f}$ is not known, but a rough estimate of the Jacobian, $\mathbf{J}(\mathbf{x})$, of $\mathbf{E}(\mathbf{f}(\cdot))$ is sufficient to perform a set of static positioning skills accurately. Intuitively, $\mathbf{J}(\mathbf{x})$ maps small changes in robot state about the point $\mathbf{x}$ to changes in the error measurement. Thus, the following control law can be used to determine the robot's image velocity, $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = -\mathbf{W}\mathbf{J}^{+}(\mathbf{x})\mathbf{E}. \tag{2}$$

This drives the observed error, $\mathbf{E}$, to zero under certain assumptions, even in the presence of calibration error [21]. Here, $\mathbf{W}$ is a positive definite gain matrix and $^+$ denotes the generalized matrix inverse (simple matrix inverse, when $\mathbf{J}$ is invertible, and the pseudo-inverse, $(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T$, when it is not).

A crucial aspect of this framework is that task specifications exist in *task space* while the command encoding exists in *image space*. Thus, by developing a set of primitive task-space constraints with corresponding image-space encodings, we can support intuitive specification of tasks in the "natural" geometry of the robot workspace and still issue image-space commands directly from observations *without having to perform an explicit scene reconstruction.*

In addition, if the primitive tasks are chosen carefully, they can be combined to form new tasks which meet each of the constituent contraints. For example, Equation 2 expresses a primitive task defined by a single image constraint. A common method for building more complex tasks is to "stack" such constraints:

$$\dot{\mathbf{x}} = -\mathbf{W}\begin{pmatrix}\mathbf{J}_1\\ \vdots \\ \mathbf{J}_k\end{pmatrix}^{+}\begin{pmatrix}\mathbf{E}_1\\ \vdots \\ \mathbf{E}_k\end{pmatrix}. \tag{3}$$

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

We refer to this composition of servoing tasks as *parallel composition*.

In general, one can define any number of primitive servoing operations using one or more cameras. In this article, we consider two cameras and only two primitive skills: point-to-point and point-to-line servoing. In the former, a point rigidly attached to the robot's end effector is positioned to a fixed reference point; in the latter, the manipulated point is brought to lie on a fixed reference line. Detailed explanations of these and related primitives are provided in the appendix and in [16, 18].

As suggested, even these simple primitives are sufficient for performing relatively complex tasks. Returning to the gluing task, for example, assume that the syringe must be aligned vertically at the point where gluing begins. A point-to-point constraint will bring the nozzle tip to the target point. A point-to-line constraint which moves the top of the syringe to a vertical line through the target point will ensure the desired orientation. These constraints can be met simultaneously by composing them as in Equation 3.

Notice that typically there are multiple ways to choose $\mathbf{E}$ for a particular task. The choice we make can affect the convergence and convergence rate of the servoing (related to how well the lineararization $\mathbf{J}(\mathbf{x})$ approximates $\mathbf{E}(\mathbf{f}(\mathbf{x}))$), the accuracy with which the goal can be achieved, and how robust the servoing will be with respect to disturbances and loss of data. We will take advantage of this in Section 3.

2.2 Vision System Primitives

In order to support the vision-based control architecture described above, we have developed a corresponding set of visual tracking primitives. This system, called "XVision," also incorporates a notion of primitive tracking composition similar in spirit to parallel composition. Here, we highlight XVision's fundamental features; [17, 19, 36, 35] contain more detailed descriptions.

Every tracking primitive returns a parameter vector – the vector of parameters it observes. Tracking algorithms follow a cyclic process consisting of three phases: In the *predict* phase, an initial guess is made for the parameter vector of the primitive. The default prediction is that features remain at their last observed location, but other prediction schemes (*e.g.*, Kalman filters) have been incorporated. Although in practice, the prediction is a single parameter vector, implicitly, the prediction represents a set of parameter vectors over which the *search* phase searches. Using the notation introduced for visual servoing, if $\mathbf{y} \in \mathcal{Y}$ represents the space of possible parameter vectors, then a prediction that an observation is expected at $\hat{\mathbf{y}}$ can be thought to represent a compact region, $\mathbf{Y}$, in parameter space, where $\hat{\mathbf{y}}$ and, hopefully, the ground-truth observation are in $\mathbf{Y}$. A final *update* phase adjusts the optimal result from search based on higher-level constraints.

Currently, XVision supports the following primitive tracking types:

– "Blobs" of classified pixels which provide the centroid and area of a set of pixels classified according to intensity, color, motion, or texture.

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

– Contrast edges, which are fixed in length and return center position and orientation parameters.
– Textured image regions based on iterative sum-of-squared-differences (SSD) minimization, which return up to six parameters corresponding to the affine deformations an image region can undergo.
– Arbitrarily-shaped contours based on image gradients, which can return either the positions of contour control points or the centroid of the circuscribed region.

In addition to the tracked parameters, each tracking primitive returns a confidence value in its tracking estimate. For example, for the edge tracker, this value is greater for higher-contrast edges; for the SSD region tracker, the value is proportional to the negative of the sum-of-squared-difference residual between the template and the image.

Tracking primitives are organized according to a type system; each feature type provides a state vector with characteristic component values. Analogous to the servoing primitives, the elemental types are `PointFeature` and `LineFeature`.

Primitive features are composed by placing constraints on their joint state vector. In addition, a composition function C may be used to compute the state of a composite tracker in terms of its constituent parts. For example, tracking two edges, l_1 and l_2, as a corner, y_{cor}, exemplifies the composition of a `PointFeature` from two `LineFeatures`: $y_{\mathrm{cor}} = C_{l \to p}(l_1, l_2)$.

Similar geometric constructions, such as affine transformations $C_{\mathrm{tr}}(l)$ and midpoint constructions $C_{\mathrm{mp}}(y_1, y_2)$, can likewise be represented as compositions or state transformations of visible image primitives. In addition to features of a geometric type, more general scalar and vector states are possible, *e.g.*, cross ratios $C_{\mathrm{cr}}(y_1, y_2, y_3, y_4)$, componentwise differences $C_{\mathrm{pp}}(y_1, y_2)$, and scalar products $C_{\mathrm{pl}}(y, l)$. The latter two provide exactly the error information required for the point-to-point and point-to-line servoing primitives. In this way, the vision system can provide a single, composite tracker whose state is the encoded task error, $\mathbf{E}$, needed for robot control in Equation 3.

3 Task Decomposition

Parallel composition alone cannot solve whole manipulation tasks. A principal challenge is the creation of temporal manipulation sequences from simpler servoing actions. In order to distinguish this type of task composition in time from the parallel composition of tasks, we will use the term *task chain* to denote a temporal sequence of servoing tasks, and *task link* to denote a single task in time. We note that a task link may itself be a parallel composition of primitive tasks. There are obvious extensions to tasks in tree structures for multi-robot or multi-threaded tasks, but they will not be discussed here.

3.1 Task Chaining

In image space, a single task link can be described by an action specification as follows:

$$A \;=\; (\, \mathbf{E}^{\mathrm{init}}(\cdot),\ \mathbf{M},\ \mathbf{E}^{\mathrm{final}}(\cdot)\,)\ . \tag{4}$$

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

Here, $\mathbf{E}^{\text{init}}(\cdot)$ expresses the perceptual precondition for the task: $\mathcal{Y}^{\text{init}} = \{\mathbf{y} : \mathbf{E}^{\text{init}}(\mathbf{y}) = 0\}$ represents the set of observations, of which one must be observed before the task can begin. Similarly, $\mathbf{E}^{\text{final}}(\cdot)$ expresses the desired final configuration, and the set of allowable observations is $\mathcal{Y}^{\text{final}} = \{\mathbf{y} : \mathbf{E}^{\text{final}}(\mathbf{y}) = 0\}$. For explicit trajectory control $\mathbf{E}^{\text{init}}$ and $\mathbf{E}^{\text{final}}$ can be extended to a one parameter family of encodings: $\mathbf{E}^{\text{traj}}(y,t) = 0$, for $t \in [0,1]$, $\mathbf{E}^{\text{traj}}(\cdot,0) = \mathbf{E}^{\text{init}}(\cdot)$ and $\mathbf{E}^{\text{traj}}(\cdot,1) = \mathbf{E}^{\text{final}}(\cdot)$. For t between 0 and 1, $\mathbf{E}^{\text{traj}}(y,t) = 0$ defines the intermediate points (or sets thereof) on the trajectory.

$\mathbf{M}$ describes any (visual or motor) constraints that must maintained during the manipulation, *e.g.*, restricting the movement of a redundant arm, avoiding joint limits and singularities, and maintaining grasp [25]. Here we simply use M to maintain the pose of the not explicitly controlled freedoms e.g. for a 3 DOF translatory movement we maintain $(x_4 \ldots x_6)$ constant in motor space.

A task chain is then a sequence of actions, $\{A_0 \;\ldots\; A_{K-1}\}$, such that the potential final observations of task $k-1$ satisfy the required initial observations for task k. More precisely, $\mathcal{Y}_{k-1}^{\text{final}} \subseteq \mathcal{Y}_k^{\text{init}}$ for $0 < k < K$. $\mathbf{E}_{K-1}^{\text{final}}(\mathbf{y}^{\text{f}inal}) = 0$ verifies task completion.

In many cases the purpose of a task is to go from a loosely-defined initial state to a highly constrained final state; typically, $\mathcal{Y}^{\text{init}}$ is large whereas $\mathcal{Y}^{\text{final}}$ contains a smaller set of observations. For example, in the gluing task, the robot may start almost anywhere in the workspace, allowing $\mathcal{Y}^{\text{init}}$ to be the set of all of the possible observations that might occur with the robot holding the syringe. On the other hand, there is only a single observation vector (together with some small margin of error) that forms the set $\mathcal{Y}^{\text{final}}$ of goal observations. The purpose of the sequence $\{A_0 \;\ldots\; A_{K-1}\}$ is to break down this overall task into manageable and robust task links, where each subgoal is defined by smaller and smaller sets of possible final states.

As long as each subtask succeeds, the system will achieve its overall goal. In Section 3.4, we consider the possibility of subtask failure due to failures in vision.

3.2 Manipulation Hierarchy

How should large tasks be broken into task links? One natural division establishes task links based on changes in the features, dimensionality, or precision of the constraints that occur during a complex task. This happens, for instance, (i) when more precise movement is needed and more features need to be incorporated, (ii) when the salient part of an object changes (*e.g.*, from the head to the claw of a hammer) and (iii) whenever an object is grasped or regrasped, changing its geometry with respect to the robot end effector.

A general principle for system design is to use the simplest possible servoing movement which suffices for each step in a manipulation task. In other words, we use the set of primitives which accomplish the task with contraints on the fewest degrees of freedom (DOF). Our experimental evaluations of control level visual servoing in [26] show that smaller and simpler manipulation models result in

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

more robust model estimation and servoing. Also, controlling fewer DOF requires fewer visual measurements and a smaller task description. Simpler and more robust tracking can be used for coarse movements, and sensitive, computationally expensive tracking is only needed during fine manipulation.

Consider again the gluing task. The system begins with a movement to bring the syringe from some arbitrary initial configuration to some point near the target. This is naturally solved as a 3-DOF translational movement, accomplished by a coarse point-to-point primitive ("coarse" means that $\mathbf{E}^{\text{final}}$ accepts a fairly large volume of observations as end goals). To align the syringe, orientation must be controlled, *e.g.*, through parallel composition of two point-to-line primitives. Finally, to bring the nozzle to the application point, an additional point-to-point constraint must be added. One possible decomposition of this task, therefore, involves three separate task links. Discontinuities in control complexity such as these provide natural breakpoints for many tasks.

To solve typical coarse-to-fine tasks, we define three classes of servoing movements: *transportation*, *alignment* and *fine manipulation*, summarized in Figure 2. The transportation movement is a coarse primitive for long motions, such as reaching to bring a tool near the site of a task. The movement is typically 3 DOF, but can be lower (it is 2 DOF when pushing something on a surface). This means that for m visual measurements the visual-motor Jacobian is of size $m \times 3$. When using stereo vision, simple centroid tracking of the moving object giving a 4×3 Jacobian is sufficient to complete a transportation subtask.

Fine manipulation movements adjust translation and orientation precisely in up to 6 DOF for a rigid object. A visual-motor Jacobian of size $m \times 6$, $m \geq 6$ is required, based on tracking m image feature values. For example, centroid tracking is, in general, insufficient for controlling both position and orientation. More sensitive estimation of several object features must be used.

When switching between transportation and fine manipulation modes, the appropriate preconditions $\mathbf{E}^{\text{init}}(\mathbf{y})$ of the higher DOF fine manipulation must be satisfied, and to perform accurate fine manipulation a reasonably accurate model is needed. The alignment movement is a high DOF movement whose goal $\mathbf{E}^{\text{final}}(\mathbf{y})$ is the precondition of the subsequent fine manipulation, but whose precondition $\mathbf{E}^{\text{init}}(\mathbf{y})$ specifies fewer DOF. 3-DOF transportation as defined above can deliver an object to a specific location, but typically, at an unspecified, arbitrary orientation. The alignment move accepts this arbitrary orientation in $\mathbf{E}^{\text{init}}(\mathbf{y})$, but delivers the object at a specific position and orientation for fine-manipulation.

Other variations are possible. For instance, a guarded move (useful in grasping tasks) can be created by servoing along some visually defined direction until an external contact condition is met. Open-loop movements can be performed using the visual-motor model, but without continuous visual feedback.

Composition Example We now describe the construction of a mid-level skill from the chaining of low-level task links. The example in Fig. 3 shows a system in which the task is similar to the gluing application: insertion of a piece of

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

Basic visual servoing movements

1. **Transportation**
   - A coarse primitive for large movements.
   - ≤ 3 DOF control of object centroid.
   - Robust to disturbances.
2. **Alignment**
   - For switching modes between coarse and fine manipulation.
   - Moves from 3D control coarse final pose to satisfy the 6D defined initial pose of the fine manipulation.
3. **Fine manipulation**
   - For high precision control of both translational and angular position.
   - 6 DOF control based on several object features.
   - Adaptive to environment changes if on-line model estimation is used, but not as robust as the 3 DOF control.

Fig. 2. Visual space task planning and decomposition. Typical tasks have coarse and fine movements. We decompose movements into transportation, alignment and fine manipulation.

packing foam into a box of similar size using direct joint control on a PUMA robot arm. The visual goal is to align the white dots $p = p_1 \dots p_6$ on the foam with the corresponding white dots p^* on the box, so $T(p) = p - p^*$. The insertion naturally decomposes into a coarse transportation to the vicinity of the box and an alignment to make the foam plane parallel with that of the box. Lastly, a 6-DOF fine manipulation movement is used to bring the foam into the box.

To switch between the different movement types, intermediate goals are constructed in free space. In this example the cameras are uncalibrated, but by using object structure in each image separately, consistent (stereo) goal points can be constructed as follows. The vertical lines around the box are extracted and labeled l_1, l_2, l_3. The average of their lengths is $||l||$. Using l_1, l_2, l_3 the two visible sides of the box are reconstructed projectively at $||l||$ and $2||l||$ above the real target in each frame.

For the 3-DOF long range transportation, only one of the feature points is needed in each image. A point-to-point constraint, $\mathbf{E}^{\text{final}} = [y_1\ y_2\ y_3\ y_4]^T - [y_1^*\ y_2^*\ y_3^*\ y_4^*]^T$ is used to position this feature on the corresponding reconstructed point at height $2||l||$. For the fine manipulation a redundant description based on 14 feature values are used $\mathbf{y} = [y_1 \dots y_{14}]^T$ by tracking the row and column pixel location of 7 points between the two cameras (Figure 3). To satisfy initial constraint of the fine manipulation, $\mathbf{y} = [y_1 \dots y_{14}]^T$ are aligned with the corresponding reconstructed points at $||l||$; at the end of the fine manipulation, they coincide with the real feature points on the box. This alignment forces the foam to be directly above the box in a plane parallel to it.

The box-packing example demonstrates the varying demands on a visual tracking system during the execution of a manipulation task. Even as task-specific demands for information increase, exogenous events can degrade sensing and inhibit progress. A hierarchical vision system can tailor its processing to

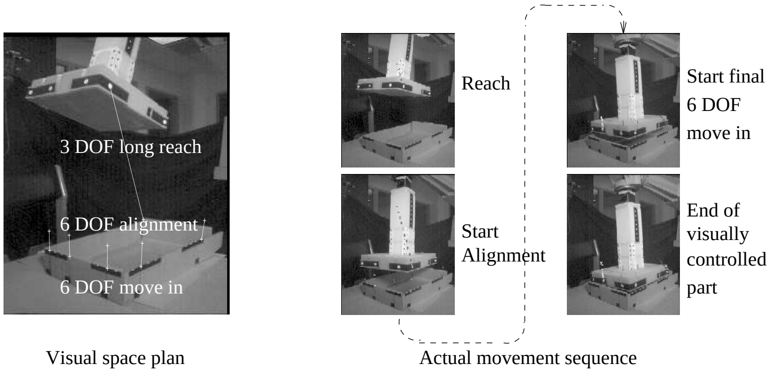

Fig. 3. Left: Planning the different phases of an insert type movement consisting of reaching and fine manipulation movements. Right: Performing the planned insertion.

the changing requirements of a manipulation task and can inform a supervisory controller as to whether sensing requirements are being met.

3.3 Vision Hierarchy

Consider a vector of visual observations, $\mathbf{y}$. A key question is how much of the vector $\mathbf{y}$ must be known for a particular task. Often, a partial observation vector suffices for transport, while more observations are needed for alignment or manipulation. Due to non-optimal tracking conditions, the system may obtain only a partial estimate of $\mathbf{y}$ at any given moment. That is, elements of $\mathbf{y}$ may be only approximately estimated or altogether unreliable.

In general, it is difficult to determine exactly what part of an observation vector is reliable. Nor is it clear what should happen when an observation is inaccurate. In practice, one can imagine different strategies to construct encodings, $\mathbf{E}$, with varying tolerance to data loss:

- If $\mathbf{E}$ is a minimal encoding the loss of any visual data will render the task T impossible.
- If $\mathbf{E}$ is a paralell composition $\mathbf{E} = (\mathbf{E}_0 \ \ldots \ \mathbf{E}_{I-1})^T$ of several "blocks" $\mathbf{E}_i$, each independently verifying T, then the dropping of features could be tolerated as long as at least one block $\mathbf{E}_i$ can be computed reliably from the observations. This situation arises when the vision system provides a great deal of redundant information, *e.g.*, many tracked points.
- An on-line strategy involves analyzing $\mathbf{J}$, the linear estimate of the visual-motor model. If a n-DOF task is being performed, but the decimated $\mathbf{J}$ has an effective rank less than n, the task cannot be completed.

Ultimately, however, it is necessary that the vision system be aware of its ability to support a particular task. We, therefore, choose a vision system architecture which parallels that of the manipulation hierarchy. The *Incremental Focus of Attention* framework [43] (IFA), developed to make object tracking

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

robust to unmodeled disturbances, provides exactly the mechanism for switching between different compositions of tracking primitives and for maintaining knowledge about observation reliability.

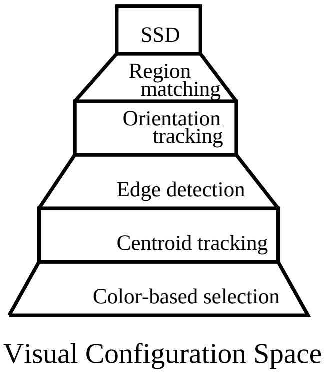

Fig. 4. The IFA vision hierarchy used for the task in Section 4.

As suggested in Figure 4, layers in the hierarchy represent different sets of tracking primitives (singly or in composition, as described in Section 2.2), and each layer fills in the observation vector, $\mathbf{y}$, to varying degrees, with higher layers offering more complete observations. At any moment, execution occurs only in a single layer. Tracking takes place at the highest layer possible, given good visual circumstances, but if tracking is lost for whatever reason (fast object motion, lighting changes, clutter, occlusions, etc.), the system falls to lower layers to perform partial observation and recovery.

Integration of the trackers is managed through explicit handling of failures and tracking re-initialization. Layers of tracking (rectangular layers in Figure 4) are interspersed with search layers (trapezoids) which narrow the set of possible configurations of objects of interest.

Layers can be interpreted as operating on the space of observations, where each layer consumes and produces sets of observations vectors. A color-based blob tracking layer, for example, would be given an initial set of possible observations, $\mathcal{Y}^{\text{in}}$, corresponding to a prediction range about where the blob might be, and return an observation set, $\mathcal{Y}^{\text{out}}$, representing its estimate together with a small margin of error. In this particular example, the $\mathcal{Y}$'s might specify the position of the target feature, but leave the orientation observations undefined. A color-based search layer, on the other hand, might take the full observation space

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

as its input set $\mathcal{Y}^{\text{in}}$, and return a smaller output set, $\mathcal{Y}^{\text{out}}$, that could serve as the input to color-based tracking. In this case, $\mathcal{Y}^{\text{out}}$ specifies observation vector elements that are unspecified in $\mathcal{Y}^{\text{in}}$. This difference between a tracking layer and a search layer is typical (though not necessary) – tracking layers will tend to preserve the dimensionality of the observations, whereas search layers will tend to provide additional dimensions of observation.

Transition to another layer occurs based on the success of the current layer. Tracking failures, as determined by the failure of the tracking primitives or failure to maintain a geometrical constraint among primitives cause transitions downward. In all other cases, execution proceeds to the next higher layer (when one exists). For example, in Figure 4, consider the case where the orientation tracking layer is invoked. This layer may be using two parallel edge tracking primitives to determine the orientation of a robot tool. Should one of the tracking primitives return low confidence values, or should the two edges appear sufficiently non-parallel, the layer would signal failure and drop down to the edge detection layer in an attempt to re-initialize the edges. On the other hand, if tracking is successful, exeuction would continue with region matching and SSD tracking, where observations are made with greater precision.

Finally, we emphasize the natural correspondence between the task decomposition paradigm and IFA. In particular, where tasks are decomposed according to the types of observations required to perform them, the vision system is organized based on the observations it can provide. Moreover, although the transitions between layers in the IFA hierarchy are controlled by visual events, the layers themselves encode the degree of observational information provided by the vision system. In short, the layer of execution immediately determines which visual servoing tasks can execute.

3.4 Integrating Vision and Manipulation

If vision-based tracking operated perfectly (and barring physical obstacles), task chains as defined in Section 3.1 would run one link after the other without difficulty. Section 3.3's hierarchy for the vision component would be entirely unnecessary, since tracking would remain at the top layer throughout the task.

In reality, tracking is imperfect, so movement up and down IFA layers is inevitable. When vision runs at a lower layer, some observations are unavailable, and so certain task links are effectively prohibited. This model for vision-action interaction is sufficiently flexible that it can be integrated with a number of robot control models – robots are free to plan, react, backtrack, learn, and do otherwise, just as long as they do not execute prohibited actions.

The simplest approach for a system which is prohibited from proceeding with a subtask due to inadequate vision is to wait for vision to recover. In this case, the task chain is followed in order, with occasional pauses in robot execution.

If actions have a strict ordering of preference, reactive control through action prohibition enables a more proactive approach. Assume that A_i is always preferred over A_j for $i < j$. Then, given an observation, $\mathbf{y}$, the action to execute is the one indexed by $\arg\min_k \left(A_k : \mathbf{E}_k^{\text{init}}(\mathbf{y}) = 0\right)$. That is, the most preferred

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

task allowed by sensing should be performed. For example, if the manipulated object remains partly visible even if some features are lost, the robot may still continue with a coarse positioning task. This can be seen as a type of reactive behavior [5], where sensory input deterministically triggers one (or more) motor actions.

One can also incorporate open-loop actions which trigger upon entry to a particular layer of the vision system. Open-loop movements are a detour from the task chain. In order to continue, the sensors must observe the results of the open-loop action, $\mathbf{y}_{ol}$ and then find a task link for which $\mathbf{E}^{\text{init}}(\mathbf{y}_{ol}) = 0$. At that point, the system can merge onto the original task chain. Disturbances to the system can be handled similarly.

Finally, classical planning and robot control can be integrated with this paradigm by adding run-time checks and additional preconditions into the planning framework. Run-time checks simply require accurate sensing of relevant portions of the world prior to operation execution. The additional preconditions state that particular perceptual inputs must be available for certain operations to be legal. Initially, planning would proceed as normal, with assumptions that all perceptual conditions are met. Then, at run-time, certain actions may fail because of faulty sensing and enforced prohibitions. At that point the agent may replan, first eliminating those operations which have the failed sensing as preconditions. Under these paradigms, task chains are constantly revised and reconstructed, based on the observables and the planning algorithm.

4 Implementation

This section presents an implementation of the gluing task. Information from the vision system provides the only sensing used in these experiments. Briefly, the workcell setup consists of an IMI Zebra manipulator, and a variety of cameras which supply images to Sun Sparc 10 or Sparc 20 and SGI Indigo workstations running Solaris and Irix.

The task chain for the gluing task consists of three links: (i) a transport motion which brings the syringe from its starting location to within a neighborhood of the target surface, (ii) an alignment to achieve a desired orientation, and (iii) a movement to bring the tip of the applicator to the surface while maintaining vertical alignment.

The vision modules are arranged in layers paralleling these three links. At the bottom of the attentional hierarchy lies a centroid-tracker, based on the color and intensity of the syringe, sufficient for coarse 3 DOF control. Above this, a pair of edge trackers extract the occluding contours of the applicator. With the edge information, two corner trackers can lock onto the top of the syringe. Finally, an SSD region tracker is used to monitor the location of the lower tip of the syringe. Figure 4(right) outlines this hierarchy. Search procedures build up the vision system's set of tracked image fatures from lower-level information. For example, when tracking at the lowest level, the system only searches for parallel edges which are approximately equidistant from the syringe's centroid. The goal point, an arm of a small fastener, is also tracked using an SSD tracker.

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

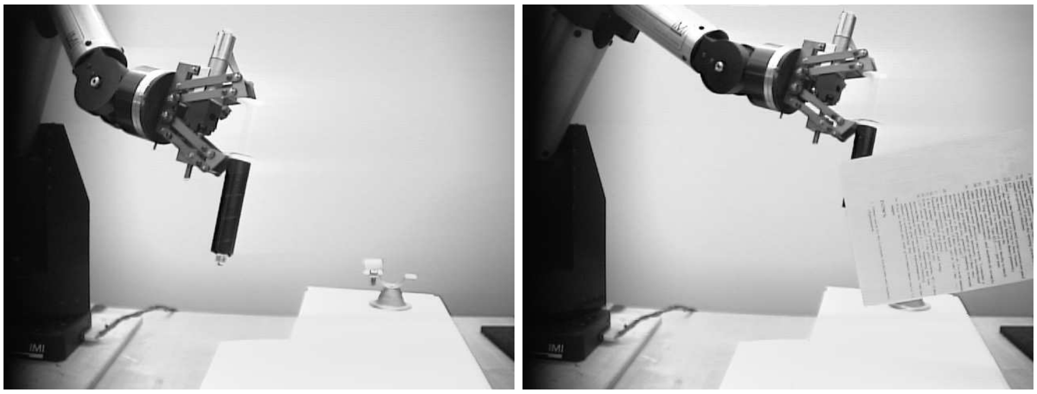

Fig. 5. Left: The starting configuration of the robot and the syringe in an adhesive-application task. **Right:** A 3 DOF reaching motion continues, even without complete state information (the applicator tip and edges are occluded), because partial information (the centroid of the syringe) is available.

Initially, the syringe, grasped by the robot end-effector, is held steady while tracking bootstraps itself through IFA's layers. Figure 5(left) shows the starting position of the robot. Once both cameras supply consistent estimates of the syringe's centroid, the prerequisites for the first level in the manipulation hierarchy are satisfied, and transport motion begins. A setpoint constructed above the goal point (using pixel columns, rather than an external cue to indicate vertical alignment) with a coarse tolerance serves as the goal of the reaching motion. As the syringe approaches the fastener, the vision system continues to try to improve its estimation of its state, *i.e.*, searching to initialize the edge and SSD trackers. Even if the edges and tip are found and subsequently lost through temporary occlusion (Figure 5(right)), the motion proceeds because the vision system is providing sufficient information (a centroid estimate) to perform the current subtask.

When the syringe is above the target surface, the manipulation system checks to be sure that the central axis of the applicator is being estimated. If not, motion stops and visual search proceeds until the edges are found. When the central axis (as the average of the occluding contours [21]) is found, the manipulation enters the alignment phase, in which the syringe is servoed to an upright position. The final approach to the fastener is prohibited until both images provide the additional visual information of the target point (via SSD trackers) and applicator tip (with corners). Figure 6(left) shows the locations of the blob tracker and the two SSD trackers before descent begins; the tracker of the applicator tip has slid slightly to the left. If one of the trackers fails entirely, the action ceases until visual recovery occurs. When the final configuration is reached (Figure 6 (right)), the manipulation system switches state and waits. At this point an open-loop motion to apply adhesive to the surface would be executed.

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

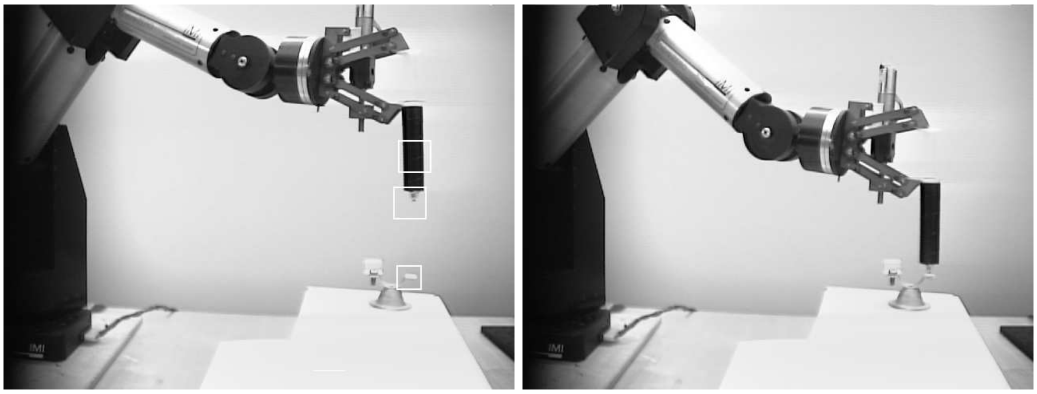

Fig. 6. Left: The states of the SSD trackers of the syringe tip and goal point are shown after the alignment phase of the task. The upper square is centered at the current centroid estimate provided by the syringe's blob tracker. **Right:** The final configuration.

Nonmanipulative Applications One of the benefits of designing a component-based real-time vision system is its ability to transfer to disparate applications. XVision and the IFA framework have been successfully applied to such pure tracking tasks as robust face localization [43], robot navigation [20], and six DOF pose estimation for human-computer interaction through a known calibration object (the "surfball") [44]. These examples testify to the flexibility and ease-of-use of the vision system architecture.

5 Conclusions

We have presented a vision system which enables robust hand-eye manipulation tasks. Both the vision and control frameworks abstract low-level processing away from algorithmic details. The vision hierarchy explicitly incorporates on-line performance assessment, and the control hierarchy allows efficient planning and replanning of tasks based on available observation. The compositional structure balances the need for performance, gained through task-specialization, with the generality of a tool kit based on primitive components that support reuse and modularity. The result is a powerful tool with built-in handling of unmodeled disturbances which can be applied to a variety of tasks.

The XVision system and IFA framework are part of an ongoing project to create a lean, portable vision system for a variety of real-time applications. This research intereacts closely with continuing work on manipulator control, robot navigation, and task specification. Additional functionality currently being investigated includes a probabilistic model for combining trackers based on different modalities (color and contour, for example). Broader problems include autonomous learning of tracking hierarchies, possibly in conjunction with learning a particular manipulation task, and incorporation of algorithms for image-to-model feature matching to link 2D and 3D *a priori* information.

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

As demands for flexible and powerful actuation systems grow, the demands on the sensing systems which inform them will likewise increase. In order to handle the spectrum of system concerns – from low-level servoing to control in the face of unmodeled events – vision architectures must balance task-specificity and domain-generality. The principles of composition of primitive components and continuous performance evaluation and adjustment lay a necessary foundation for a vision system which supports practical and general hand-eye tasks.

Acknowledgements This work was supported by National Science Foundation grant IRI-9420982 and EIA-9805817, ARPA grant DAAE07-98-C-L031, and by funds provided by Yale University.

References

1. P.K. Allen, B. Yoshimi, and A. Timcenko. Hand-eye coordination for robotics tracking and grasping. In K. Hashimoto, editor, *Visual Servoing*, pages 33–70. World Scientific, 1994.
2. R. L. Anderson. Dynamic sensing in a ping-pong playing robot. *IEEE Trans. Robot. Automat*, 5(6):723–739, December 1989.
3. M. Asada, K. Hosoda, and S. Suzuki. Vision-based learning and development for emergence of robot behaviors. In *Proc. of Symp. on Emergent Systems*, pages 16–22, 1997.
4. A. Blake, R. Curwen, and A. Zisserman. Affine-invariant contour tracking with automatic control of spatiotemporal scale. In *Proc. Internal Conf. on Computer Vision*, pages 421–430. IEEE Computer Society Press, 1993.
5. R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.
6. A. Castano and S. A. Hutchinson. Visual compliance: Task-directed visual servo control. *IEEE Trans. Robot. Automat*, 10(3):334–342, June 1994.
7. F. Chaumette, E. Malis, and S. Boudet. 2-D 1/2 visual servoing with respect to a planar object. In *Workshop on New Trends in Image-Based Robot Servoing*, pages 45–52, 1997.
8. F. Chaumette, P. Rives, and B. Espiau. Classification and realization of the different vision-based tasks. In K. Hashimoto, editor, *Visual Servoing*, pages 199–228. World Scientific, 1994.
9. W.Z. Chen, U.A. Korde, and S.B. Skaar. Position control experiments using vision. *Int. J. of Robot Res.*, 13(3):199–208, June 1994.
10. Jiyoon Chung and Hyun S. Yang. Fast and effective multiple moving targets tracking method for mobile robots. In *IEEE Int. Conf. Robotics Automat.*, pages 2645–2650, Nagoya, Japan, May 1995.
11. P. I. Corke. Visual control of robot manipulators—a review. In K. Hashimoto, editor, *Visual Servoing*, pages 1–32. World Scientific, 1994.
12. M. W. Eklund, G. Ravichandran, M. M. Trivedi, and S. B. Marapane. Adaptive visual tracking algorithm and real-time implemenation. In *IEEE Int. Conf. Robotics Automat.*, pages 2657–2662, Nagoya, Japan, May 1995.
13. B. Espiau, F. Chaumette, and P. Rives. A New Approach to Visual Servoing in Robotics. *IEEE Trans. on Robotics and Automation*, 8:313–326, 1992.
14. J. Faymann, E. Rivlin, and H. I. Christensen. A system for active vision driven robotics. pages 1986–1992, April 1996.
15. J.T. Feddema, C.S.G. Lee, and O.R. Mitchell. Weighted selection of image features for resolved rate visual feedback control. *IEEE Trans. Robot. Automat*, 7(1):31–47, February 1991.

16. G. D. Hager. A modular system for robust hand-eye coordination. DCS RR-1074, Yale University, New Haven, CT, June 1995. Accepted to appear in IEEE Trans. on Robotics and Automation.
17. G. D. Hager and P. N. Belhumeur. Efficient region tracking of with parametric models of illumination and geometry. To appear in IEEE PAMI., April 1997.
18. G. D. Hager and Z. Dodds. A projective framework for constructing accurate hand-eye systems. In *Proc. IROS Workshop on New Trends in Image-based Robot Servoing*, pages 71–82, 1997.
19. G. D. Hager and K. Toyama. The "XVision" system: A general purpose substrate for real-time vision applications. *Comp. Vision, Image Understanding.*, 69(1):23–27, January 1998.
20. G. D. Hager, K. Toyama, W. Feiten, and B. Magnussen. Modeling and control for mobile manipulation in everyday environments. Technical Report TR-1137, Yale University, New Haven, CT, 1997.
21. G. D. Hager, J. Wang, and K. Toyama. Servomatic: A modular approach to robust positioning using stereo visual servoing. In *Proc. Conf. on Robotics and Automation*, pages 2636–2641, 1996.
22. K. Hashimoto. LQ optimal and nonlinear approaches to visual servoing. In K. Hashimoto, editor, *Visual Servoing*, pages 165–198. World Scientific, 1994.
23. J. Hespanha, Z. Dodds, G. D. Hager, and A. S. Morse. What tasks can be performed with an uncalibrated stereo vision system? Submitted for review to IJCV, November 1997.
24. K. Hosoda and M. Asada. Versatile visual servoing without knowledge of true jacobian. In *IEEE Int. Workshop on Intelligent Robots and Systems*, pages 186–191. IEEE Computer Society Press, 1994.
25. M. Jagersand, O. Fuentes, and R. Nelson. Acquiring visual-motor models for precision manipulation with robot hands. In *Proc., ECCV*, 1996.
26. M. Jagersand, O. Fuentes, and R. Nelson. Experimental evaluation of uncalibrated visual servoing for precision manipulation. In *Proc., ICRA*, pages 2874–2880, 1997.
27. H. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. Journal of Computer Vision*, 1:321–331, 1987.
28. D. Koller, J.W. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In *Proc. European Conf. on Computer Vision*, pages A:189–196, 1994.
29. J. Kosecka, H. Christensen, and R. Bajcsy. Discrete event modeling and visually guided behaviors. *IJCV*, 14:179–191, 1995.
30. Demian M. Lyons. Representing and analyzing action plans as networks of concurrent processes. *IEEE Transactions on Robotics and Automation*, 9(7):241–256, 1993.
31. N. Maru, H. Kase, A. Nishikawa, and F. Miyazaki. Manipulator control by visual servoing with the stereo vision. In *IEEE Int. Workshop on Intelligent Robots and Systems*, pages 1866–1870. IEEE Computer Society Press, 1993.
32. D. Murray and A. Basu. Motion tracking with an active camera. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 16(5):449–459, May 1994.
33. B. Nelson and P. K. Khosla. Increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance. In *Proc. IEEE Int. Conf. Robot. and Automat.*, pages 418–423. IEEE Computer Society Press, 1993.
34. P. Prokopowicz, M. Swain, and R. Kahn. Task and environment-sensitive tracking. Technical Report 94-05, University of Chicago, March 1994.
35. C. Rasmussen and G. D. Hager. Joint probabilistic techniques for tracking multi-part objects. to appear in CVPR 98, November 1998.

36. C. Rasmussen, K. Toyama, and G. D. Hager. Tracking objects by color alone. In *Proceedings of the Workshop on Applications of Computer Vision*, 1996. Submitted.
37. A.A. Rizzi and D. E. Koditschek. Further progress in robot juggling: The spatial two-juggle. In *Proc. IEEE Int. Conf. Robot. and Automat.*, pages 919–924. IEEE Computer Society Press, 1993.
38. C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: The Task Function Approach.* Clarendon Press, Oxford, England, 1992.
39. J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conf. Comp. Vision and Patt. Recog.*, pages 593–600, 1994.
40. S. B. Skaar, W. H. Brockman, and W. S. Jang. Three-dimensional camera space manipulation. *Int. J. of Robot Res.*, 9(4):22–39, 1990.
41. C.E. Smith, S.A. Brandt, and N.P. Papnikolopoulos. Controlled active exploration of uncalibrated environments. In *Proc. IEEE Conf. Comp. Vision and Patt. Recog.*, pages 792–795. IEEE Computer Society Press, 1994.
42. M. Tonko, K. Schafer, F. Heimes, and H.-H. Nagel. Towards visually servoed manipulation of car engine parts. In *IEEE Proc. Int. Conf. on Robotics and Automation*, 1997.
43. K. Toyama. *Robust Vision-based Object Tracking.* PhD thesis, Yale University, 1997.
44. K. Toyama. The surfball. Technical report, Yale University, http:// www.cs.yale.edu/ HTML/ YALE/ CS/ HyPlans/ toyama/ surfball/ surfball.html, 1997.
45. L.E. Weiss, A.C. Sanderson, and C.P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE J. Robot. Automat.*, RA-3(5):404–417, Oct. 1987.
46. S.W. Wijesoma, D.F.H Wolfe, and R.J. Richards. Eye-to-hand coordination for vision-guided robot control applications. *Int. J. of Robot Res.*, 12(1):65–78, 1993.
47. 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*, pages 71–104. World Scientific, 1994.

Appendix

Servoing Primitives

Point-to-Point Positioning Given a fixed reference point p^* and a point p rigidly attached to the end-effector, we want to develop a regulator that positions the end-effector so that $p = p^*$. The corresponding feature-based task error function is

$$T_{pp}(p; p^*) = p - p^*. \tag{5}$$

Because two points not on the camera baseline are coincident in space if and only if their stereo projections are coincident, an image-based encoding which verifies the point-to-point task function (outside of singular configurations) is

$$\mathbf{E}_{pp}(\mathbf{y}) = \mathbf{y} - \mathbf{y}^*, \tag{6}$$

where $\mathbf{y} = (y_{lx}, y_{ly}, y_{rx}, y_{ry})$, the row and column coordinates in the left and right camera and $\mathbf{y}^*$ is the similarly-parametrized observation of the designated reference point.

In Proceedings Int. Conf. on Computer Vision Systems 1999, pp312-330.

Point-to-Line Positioning Suppose we are given a reference line in Cartesian space, $\mathbb{L}$, specified by a point on the line, $\mathbb{L}_d$, and a vector in the direction of the line, $\mathbb{L}_v$. With a point p rigidly on the robot end-effector, we want to develop a regulator that positions the arm so that $p \in \mathbb{L}$. The corresponding task error function is given by

$$T_{pl}(p; \mathbb{L}) = (p - \mathbb{L}_d) \times \mathbb{L}_v. \tag{7}$$

We let $\mathbf{y} = (l; y)^T$, $l = (l_l \; l_r)^T$, and $y = (y_l \; y_r)^T$ represent the image projections of $\mathbb{L}$ and p, respectively, where l_l and l_r represents the line by the normal vector to a plane defined by the line and the camera center, and the points in y are in homogeneous coordinates. Then, an image-based encoding verifying T_{pl} results from the observation that for arbitrary $\mathbb{L}$ (not in the epipolar plane containing p), $l_l \cdot y_l = l_r \cdot y_r = 0$ if and only if $p \in \mathbb{L}$. Thus, we define $\mathbf{E}_{pl}$ as:

$$\mathbf{E}_{pl}(y; l) = \begin{bmatrix} y_l \cdot l_l \\ y_r \cdot l_r \end{bmatrix}. \tag{8}$$

An equivalent image encoding can be defined by representing the line as 2 points and expressing Eq. 7 directly in left and right image coordinates.

A composition example Now, consider the gluing example from Section 2.1. Let the observation of the nozzle tip be denoted by y_{tip}, the target point by y^*, and the top of the syringe by y_{top}. A line in a known relative pose (perhaps observed on a nearby object), l_{vis}, can be used to hallucinate a target line, l^*, should no vertical cue exist at y^*. The resulting image encoding of the composite task is

$$\mathbf{E}(y_{\text{tip}}; y_{\text{top}}; y^*; l_{\text{vis}}) = \begin{bmatrix} y_{\text{tip}} - y^* \\ y_{\text{top}} \cdot tr(l_{\text{vis}}) \end{bmatrix} \tag{9}$$

Here, tr is an affine translation of its argument in image space; its use is justified so long as an affine visual approximation is valid.