Proceedings of the 2000 IEEE
International Conference on Robotics & Automation
San Francisco, CA • April 2000

# On Specifying and Performing Visual Tasks with Qualitative Object Models

Gregory D. Hager
Department of Computer Science
The Johns Hopkins University
3400 N. Charles St.
Baltimore, MD, 21218
E-mail: hager@cs.jhu.edu

Zachary Dodds
Department of Computer Science
Harvey Mudd College
1250 N. Dartmouth Avenue
Claremont, CA 91711-5901
E-mail: dodds@cs.hmc.edu

## Abstract

Since its genesis, vision-based control has aimed to develop general-purpose, high accuracy systems for manipulating objects. While much of the scientific and technological infrastructure needed to accomplish this aim is now in place, several stumbling blocks still remain. One continuing issue is accuracy, and its relationship to system calibration. We describe a generative task structure for vision-based control of motion that admits a simple, geometric approach to task specification. At the same time, this approach allows us to state precisely what types of miscalibration lead to errors in task performance. A second hurdle has been the programmability of hand-eye systems. We however, argue that a structured object representation sufficient for flexible hand-eye coordination is a possibility. The result is a high-level, object-centered language for expressing hand-eye tasks.

## 1 Introduction

Since their infancy, one of the goals of vision, robotics and AI has been to create an "intelligent connection from perception to action" [2]. Over the last several decades, a great deal of research has been devoted to this topic. Specifically, in the area of vision and robotics, great strides have been made in understanding both the theory and practice of developing vision-based control algorithms.

However, in the process of developing this understanding, the focus in the field has shifted away from the question of how to specify (e.g. program) vision-based tasks toward more circumscribed, though technically challenging, problems in vision and control. For example, in several recent workshops on vision and control [21, 17, 32], there is not a single paper on system integration issues or programming of vision-based systems. Although many systems have demonstrated the ability of image-based feedback control to accomplish complex positioning tasks, these system are usually devoted to a single task and human intervention is often needed to specify visual goals.

The goal of this paper is to place our recent work in the area of vision-based control within the larger perspective of building programmable vision-based control systems. In particular, we argue for an approach to vision-based robot programming that, on the one hand moves from expressing vision-based control of a single task using simple image features to a general task specification language in terms of *object relative actions*; and on the other hand chooses to do so at a level which is technologically and scientifically achievable. As such, this work builds on previous results on the feasibility of performing broad families of vision-based control tasks [6], how the geometric structure of visual features [9, 10] enables robotic tasks without painstaking calibration of the imaging system [28, 12], and on the visual tracking capabilities needed to implement such tasks [14].

It is worth noting that there has been prior work on languages for specifying vision-based tasks from simple primitives. Schnackertz and Grupen, for instance, employ three control primitives and corresponding image constraints as a basis for visual homing tasks [29]. Morrow and Khosla start from a complete taxonomy of the constrainable degrees of freedom between two rigid bodies and progress to a partial list of vision- and force-based primitives for accomplishing those constraints [25]. Chaumette, Rives, and Espiau develop complete control details for several such DOF constraints, termed "virtual linkages," to form a library of visual servoing tasks [5]. Their framework handles points, lines, spheres, cylinders, and their combinations. The ViSP system [22] implements this library modularly, enabling composition through any desired conjunction of available primitives. It also permits the definition of secondary tasks which control degrees of freedom left unspecified by a primary visual specification [23]. However, ViSP commits at the control

0-7803-5886-4/00/$10.00© 2000 IEEE

level to an eye-in-hand configuration, and few guarantees can be made regarding task performance [4]. In addition, all of these systems lack completeness, e.g., disjunctions of primitives are unavailable, and each implicitly assumes the stereo calibration necessary to perform its primitive tasks. In contrast, the languages presented here do suffice to specify any performable task, and they do so under a variety of precisely-defined levels of certainty about the observing vision system. Finally, [20] describes sensor-based specifications (clicking on desired goals in an image). Likewise, other authors describe geometrically-based specifications (derived from known properties of visible objects) and even linguistic specifications. In this paper we are interested in the relationship between task-level geometric specifications, which are the most natural given our pose-based notion of a task, and image-based ones, on which the robot control is based.

The remainder of this paper is organized as follows. In Section 2 we present a high-level overview of a system for programming object-centered manipulation tasks. Section 3 describes a class of task specification languages. Section 4 proposes an object representation, and Section 5 grounds the examples of section 2 using the task language and object representation. We close with suggestions of open research problems.

## 2 System Structure and Motivational Example

The primary role of vision in manipulation is to govern the free-space motions of an object — either to bring surfaces into contact, or to avoid collision during motion. In either case, any high-level approach to programming must somehow indicate what surfaces are to participate in the task, and how. Thus, a programming system must have some way of relating task-specific object information to low-level control loops and vision processing.

To illustrate the general idea, consider mating operations such as inserting a plug [8] or placing a tool onto a fastener [13]. Assuming a screwdriver is already in the end-effector, a skeleton program for applying it to a screw would be

```
locate(screw);
move_above(sdriver.tip,screw.head,screw.axis,1 cm);
while move_above(sdriver.tip,screw.head)
{   align(sdriver.axis,screw.axis);  }
while align(sdriver.axis,screw.axis)
{   guarded_move(vel,sdriver.z);     }
```

Here, statements terminated with semi-colons indicate sequencing. The `while` command sets up a primary motion control task and a secondary motion control task along the lines of [8]. The precise definitions of `align` and `move_above` will be given in Section 5; for the moment we appeal to the reader's intuition as to their function.

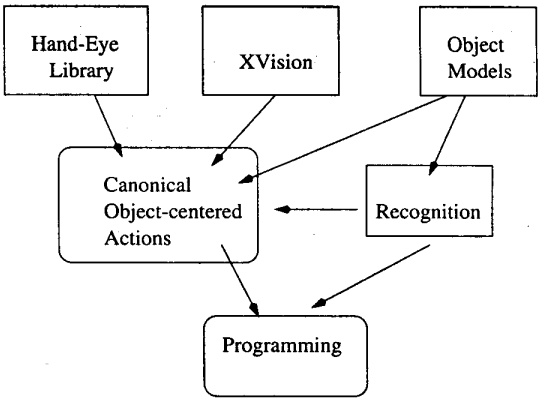


Figure 1: A schematic diagram of an object-level vision-based programming system.

In order to carry out such a program, the system must

1. Contain models of the objects involved — specifically models that ground out the various labels used in the program.
2. Register objects to a model using visual information.
3. Use object geometry and part labels to locate specific visual features of interest for the operation.
4. Initiate visual tracking and feedback control appropriate to the operation.
5. Monitor the operation to ensure the computed visual information remains sufficient to perform the task.

Figure 1 shows a how an object-centered hand-eye system capable of such operations could be constructed. The primary components are: a programming interface to a set of object-level actions; a recognition module that grounds object labels for the programming language and actions; a set of object models; a library of low-level control actions, and a visual tracking system. The two components shown in gray, the library of hand-eye primitives and XVision, are components we have already constructed in our research [14, 13]. An initial proposal for a programming interface appears in [30]. The remaining boxes have yet to be developed.

In conception, the structure and use of the system resembles a classical AI/planning system [24]. That is, there are a set of actions (the vision-based control primitives), and a set of sensing routines (XVision)

and object recognition to tie them together. However, the most important differences are the following:

- The sensing control actions are very low-level primitives — essentially tight feedback loops. Furthermore, these low-level primitives are based on a generational structure as discussed in Section 3. As such, they have a clear, well-defined semantics in terms of what they accomplish, and when they can accomplish it.
- The model database is a hybrid design which includes only enough geometric and object appearance information to make it possible to pick the object out of an image and find certain task-relevant features. There is no intention of "reasoning" about objects — the information available to the system (part labels, appearance, geometry) is pre-programmed for the set of objects in question. This representation is further discussion in Section 4.
- Object-centered actions are based on using recognition to find specific object features which can be tracked, and then defining tasks (kinematic constraints between objects) on those objects. Again, we view this as an extension of existing work on object recognition [11, 19, 27], visual search [31], visual tracking [14], and vision-based control [18].

In the remainder of this article, we expand upon each of these points in turn.

# 3 Towards a Robust Vision-Based Control Library

Two of the principle issues in any control system are speed and accuracy. However, both speed and accuracy can be limited by the degree which fundamental system parameters are known. In the case of vision-based control, one might argue that if object models were perfect, vision were able to precisely compute scene geometry, and the calibration between the actuator and the vision sensors was exact, then issues of accuracy would be moot. However, reality is far from such idealizations. Hence, an important issue is how to design a system that has high accuracy, even when calibration information is imprecise or unavailable.

We and others have shown that, in many cases, the static, set-point accuracy of a vision-based control system is independent of some or all calibration information, provided the appropriate system design is used. We have further shown that complete families of tasks that have this property can be generated using a simple "task algebra.". Here, we review those results, and, in particular, highlight relationship between task complexity and calibration requirements. We note this material is a summary of [3, 15, 6].

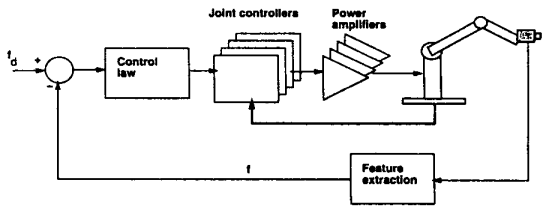


Figure 2: Dynamic image-based look-and-move structure (from [18]).

## 3.1 Feedback Systems and Uncertainty

In the sequel, by a positioning task or simply a "task" is meant, roughly speaking, the objective of bringing the pose of a robot to a "target" in its workspace. For the purposes of this article, we will assume these tasks are carried out by systems with a feedback structure as shown in Figure 2. In particular, we assume local feedback from encoders is used to stabilize the system, so that the predominant factors affecting the dynamic response of the system are time delay and system model accuracy. The task is given by a choice of error signal to the system. Given such an error signal, the system chooses a control signal (usually using some model of the relationship between the sensed information and endpoint geometry) to drive this error to zero.

The *accuracy* of a configuration is its distance to the goal set of configurations, defined under a suitable (and perhaps task-specific) metric. Of interest is the relationship between the steady state behavior at the set-point (zero error point) of this system and the accuracy that is achievable for a given geometric task. For example, the error could be based on a geometric reconstruction (e.g. position the end-effector at a given set of visual coordinates corresponding to geometric coordinates). This is an extreme case where *any* error in system calibration could impact the endpoint accuracy of the system. Conversely, it is possible to show that if the task is to move a directly observable point on the end-effector to a directly observable point in the world, then the only geometric information needed is that which leads to closed-loop stability [15]. In short, camera geometry (and hence calibration) has no effect on the accuracy of the system. Furthermore, in many cases this level of geometry can be estimated adaptively [3], and closed-loop stability is usually insensitive to minor errors in modeling.

In previous work, we have developed a framework for formally studying the relationship between uncertainty in calibration and the accuracy of task performance [15]. We quantify uncertainty in calibration by modeling the observing camera system by a *set* of camera models, $\mathcal{C}$. By varying the size and structure of $\mathcal{C}$, we are able to model different levels of knowledge about the underlying camera system. More specifi-

cally, the camera models we consider are:

1. A perfectly calibrated camera system.
2. An internally calibrated camera system.
3. A weakly calibrated projective camera system.
4. An uncalibrated projective camera system.
5. A camera system which is injective on the robot workspace.

These camera models have been arranged in order from lesser to greater levels of calibration uncertainty. Not surprisingly, the tasks that can be performed with camera systems varies inversely with the size of the uncertainty set. In fact, we have shown that the set of tasks which, in the absence of noise, can be performed with absolute accuracy for the sets above are:

1. All visually specifiable tasks ($\mathcal{T}_{all}$).
2. All tasks that are invariant to similarity transformations ($\mathcal{T}_{sim}$).
3. All tasks that are invariant to projective transformations ($\mathcal{T}_{proj}$).
4. Unknown ($\mathcal{T}_{uncal}$).
5. All tasks that are invariant to the class of all bijections ($\mathcal{T}_{bij}$).

In symbols, the following relationship holds:

$$\mathcal{T}_{\text{inj}} \subset \mathcal{T}_{\text{uncal}} \subset \mathcal{T}_{\text{wk}} \subset \mathcal{T}_{\text{sim}} \subset \mathcal{T}_{\text{all}}.$$

We note that in the case of $\mathcal{T}_{uncal}$, it is not possible at this time to give a precise characterization of the set of tasks that can be performed in terms of invariance. Although we do not have a complete characterization of $\mathcal{T}_{\text{uncal}}$, we do have, in effect, "bounds" on the capabilities of systems with uncalibrated projective cameras. We also note that a more general characterization of task relationships is given in terms of group invariance and can be found in [16].

## 3.2 A Generative Structure for Tasks

From a programming perspective, tasks are specified terms of the geometry of objects and the robot workspace. However, to carry out a task, it must be expressed in terms of an error on quantities measurable in an image. Furthermore, this translation should preserve correctness — driving the error signal to zero should achieve the desired endpoint configuration. Thus, given a level of knowledge about the camera, we would like to automatically determine if and how a task may be performed.

The invariance properties of the task families of the previous section means that there is an underlying group structure to this problem. Thus, a principled way of generating task descriptions is to develop a set of primitive operations and a set of operators on primitives that can generate the entire group. To make this idea concrete, consider the following set of "primitive" tasks.

**Point-to-point task** Let $T_{\text{pp}}$ represent the task of making two points in space coincident.

**Collinearity Task** Let $T_{\text{3pt}}$ represent the task of making three points collinear in space.

**Co-planarity Task** Let $T_{\text{4pt}}$ represent the task of making four points co-planar in space.

**Cross-ratio Task** Let $T_{\text{cr}\alpha}$ be the task of arranging four points to be collinear, and also spaced to achieve a given cross-ratio [26]. In fact, the set of tasks specifying point configurations with each possible cross-ratio $\alpha \in \mathbb{R}$ constitute a general ability to position metrically with respect to three collinear points; we term this set the *cross-ratio primitive.*

**Euclidean-ratio Task** Similar to $T_{\text{cr}\alpha}$, let $T_{\text{Er}\alpha}$ represent the task function of making three points collinear such that the ratio of the two lengths formed is $\alpha$. Again, this is formally a collection of tasks, the *Euclidean-ratio primitive.* For example, a midpoint-positioning task would be $T_{\text{cr1.0}}$.

Some or all of these example tasks are a part of most implemented hand/eye systems. Based on the previously mentioned invariance structure, it is not hard to show that $T_{\text{pp}}$ is in all task families (it is invariant to the group of bijections), $T_{\text{3pt}}$, $T_{\text{4pt}}$ and $T_{\text{cr}\alpha}$ are in $\mathcal{T}_{\text{proj}}$ but nothing larger, and $T_{\text{cr}\alpha}$ is in $\mathcal{T}_{sim}$, but nothing larger. Furthermore, each of these tasks has an equivalent image-based error signal for carrying it out, provided the camera model uncertainty falls within the specified set.

We can now characterize the complete class of decidable tasks in two ways: the first is a set of operations that generate, from the above primitives, all possible tasks. The second is an operation that, given a (larger) class of primitives, can generate all tasks with a single combinator. Here we detail the former and refer to [6] for the latter.

Let $T, T_1, T_2$ be task functions which take $n$-lists of feature points, i.e., $f = \{f_1, \ldots f_n\}$. We can define the following five operators on tasks:

**Complement** ($\neg T(f)$) : produce the "negation" of the original task.

**Permutation** ($\pi T(f)$) : permute (by $\pi$)the role of features in a task

**Disjunction** ($(T_1 \vee T_2)(f)$) : perform task $T_1$ or $T_2$.

**Expansion** ($\varepsilon T(\{g_1 \ldots g_m\})$) : include new features into a task.

**Contraction** ($cT(\{f_1 \ldots f_l\})$) : discard features from a task.

We can then show the following:

- The set $\mathcal{T}_{inj}$ is the closure of the point-to-point task under these operations.
- The set $\mathcal{T}_{wk}$ is $\mathcal{T}_{inj}$ plus the closure of the cross ratio primitive, the collinearity primitive, and the coplanarity primitive under these operations.
- The set $\mathcal{T}_{sim}$ is $\mathcal{T}_{wk}$ plus the closure of the Euclidean-ratio primitive under these operations.

Furthermore, we can show how to build error signals for each construction based on the error signals of the component tasks.

In short, we can build a few simple primitives, and by suitable composition, program a complete family of tasks relative to a given level of knowledge about camera calibration . It is worth noting that the complement operator creates special problems as it is not hard to show there is no continuous encoding that precisely specifies it in most cases. For example, the complement of the point-to-point task is the task of making sure two points are not coincident. There can be no continuous function that is zero everywhere except at a single point. Thus, we more commonly remove the complement task from consideration and add instead:

**Conjunction** ($(T_1 \wedge T_2)(f)$) : perform task $T_1$ and $T_2$.

The closure under this new set of operator generates what we refer to as the *positive* tasks — intuitively the tasks that achieve contact as opposed to ensuring free-space motion. In this case, it is not hard to show that each task has a continuous (and even differentiable) error function.

In summary, we can now write programs which combine a small set of control primitives using a small set of operators, and generate vision-based control algorithms that have provable performance, even when there is uncertainty about the camera calibration.

# 4 Object Models, Recognition, and Tracking

The task library of the previous section operates at the level of small collections of feature vectors. However, our goal is to flexibly and accurately manipulate objects. At the simplest level, this may involve direct operator intervention to, for example, indicate control points for the manipulation task. However, it seems safe to assume that, in the case of human assistance, vision-based manipulation systems would be largely dealing with a small set of well known tools. Thus, while the problem of representing objects (indeed the question of what constitutes an object) is a subtle one which has a long history in vision and AI, we focus more specifically on developing a representation which meets the needs of vision-based control. We see those needs as the following:

- Geometry: A representation needs to contain the essential geometric aspects of an object — specifically those aspects that are instrumental to the function or mode of employment of the object. The language for representing objects should be able to specify precisely-known metrical information as well as weaker geometric relationships between features, e.g. the collinearity or parallelism. The representation also defines a set of canonical coordinate directions associated with the object — e.g. the $z$ axis which in turn defines top and bottom, and the $x$ axis which defines front and back.
- Appearance: The representation needs to include enough detail about the appearance of the object that it can be selected in an image. In addition, the representation must contain descriptions of potentially observable features, which we take to be points and lines formed by markings or object structure. These features form the bridge from object recognition (identifying image information) to task specification (identifying task-specific object geometry).
- Categorization: We would like the representation to represent general aspects of objects, not specific objects. For example, when describing how a screwdriver is used for a task, the description should apply to *all* screwdrivers, not a specific instantiation of one.

In particular, we believe that, with sufficient structure, the problem of locating task-specific features, what we refer to as the *image search* problem, is tractable and solvable. Ultimately, the value of a representation is that it lets us assign a label (e.g. the business end of a screwdriver) to a visual feature useful for tasks.

## 4.1 Object Representation

Our suggestion for a representation harks back, in many ways, to the attempts to building generic volumetric models using generalized cylinders [1]. However, we attack the problem as one of specifying vari-

able and fixed geometric relationships, along with appearance information that could be used to register these parameters. For example, the one-dimensional essence of a tool like a screwdriver is not altered if the relative dimensions of the handle and the shaft change; likewise the generic appearance of a screwdriver is the same across a large family.

These ideas can be captured a simple hierarchical collection of objects composed of parts. A part has the following general form:

part:
    appearance: <ap1>, <ap2>, ....
    geometry: <geo type>
    frame: <coord trans>
    labels: <(name1,coord1)>, <(name2, coord2)> ...

The `appearance` field is simply a list of images giving canonical views of the part. These views are registered to a canonical part coordinate frame. The `geometry` field allows a list of known geometry types which include cylinders, planes, polygons of specific types, and so forth. The primary purpose of this field is to infer how a part might be tracked, and possibly how it may be recognized. The location field is a coordinate in the frame of the parent object. This will be a Euclidean coordinate with variables expressing relationships among unknown values. This approach has the advantage of making the uncertainty in the model explicit. In particular, it allows objects to be categorical by introducing uniform and nonuniform scaling. Finally, the `labels` field allows more specific feature locations to be tagged.

As a simple example, a flat bladed screwdriver would consist of three parts:

handle:
    appearance: ...
    geometry: quasi-cylindrical
    frame: (0,0,0), (0,0,$\alpha$)
    labels:

shaft:
    appearance: ...
    geometry: cylindrical
    frame: (0,0,a), (0,0,$\alpha$)
    labels:

blade:
    appearance: ...
    geometry: segment
    frame: (0,0,b), (0,$\pi/2$,$\alpha$)
    labels: (tip, (0,0,0))

Here, for simplicity, we have represented coordinate transforms by three-vectors of position and angles. The values $a$, $b$, and $\alpha$ are variable. Cylindrical and quasi-cylindrical objects have an axis along $z$, so this part representation says that there are two cylinders that share a common $z$ axis, and an edge perpendicular to that axis which determines the orientation about $z$. In short, this also captures the one-dimensional line of action of the tool, as well as the unknown scaling of its different parts. The label `tip` is added as a way of identifying the exact center of the blade (as determined by the central axis of the screwdriver).

### 4.2 Matching Models in Images and Tracking Features

As suggested in the representation, we have two sources of information about objects: appearance and (approximate) geometry. Furthermore, we can safely assume that the object in question is in view and may even be in an approximately known location and pose. Thus, the problem of matching parts can be attacked using both feature-based methods such as interpretation trees [11] and/or image-based matching methods [27]. The problem of using variable coordinate information can be viewed either as a problem in geometric inference, or, as we advocate, a source of information for an active visual search process [31]. Once registered, task-specific features can be tracked using modalities appropriate to the appearance and structure of the object. Tools such as the XVision system [14] provide a proof of concept that real-time visual tracking capable of addressing such problems is available.

We note that the tracking problem may itself involve the use of object information, particularly if the task involves a construction that depends on underlying object geometry. For example, placing a screwdriver onto a screw involves determining the location of the central axis of both screwdriver and screw. These involve image constructions that in turn depend on knowing that both have cylindrical sections from which an image construction for the central axis can be derived [12]. Such constructions would be contained within a tracking/geometry library, and automatically produced when needed.

## 5 Specifying Visual Tasks

Combining the ideas in the previous two sections, we are now in a position to ground the program originally expressed in Section 2. We assume that, within a program, each label is grounded as a stereo pair of features. Thus, for example, the label `screw.tip` implicitly refers to a pair of image locations. Likewise, `screw.axis` implicitly refers to lines in two images. A line in an image is represented as a center point and

a direction. The operator `trans` translates a stereo pair of points along a stereo pair of lines by an image distance estimated (using calibration) from the supplied metric coordinates. With this, we can define the vision-based control operators as follows:

$$\texttt{move\_above}(pt_1, pt_2, v, d) = T_{pp}(pt_1, trans(pt_2, dv))$$

and

$$\texttt{align}(l_1, l_2) = T_{p2l}(l_1.center, l_2) \wedge T_{p2l}(l_2.center, l_1)$$

where $T_{p2l}$ places a point on a line — it can be expressed as a slight variation on $T_{3pt}$ [13].

From this decomposition, we can state the following claims:

- Given a weakly calibrated camera pair, `move_above` will place the screwdriver tip *somewhere* along the axis of the screw, although we cannot guarantee the exact distance. If the camera is not projective, then we cannot guarantee the effect of `trans`. If we had an internally calibrated camera, we could guarantee the distance by using either the length of the screwdriver or the length of the screw as a metric. Given three points on the screwdriver, we could also guarantee the distance between screwdriver and screw with a projective camera.
- Given at least a weakly calibrated camera pair, the `align` operation will force the screwdriver and screw to be physically aligned. Moreover, the use of align in the original program (Section 2) in the primary/secondary task configuration guarantees this continues to be the case during motion.

# 6 Conclusion

Although clearly speculative at this point, we feel the ideas expressed above lead in the direction of an algebraically interesting and practical notion of task specification. However, several issues remain before practical and reliable systems of this form can be implemented. Some open issues include

**Inexact constraints** A task specification may, in fact, be impossible to satisfy given the actual geometry of the objects involved. For example, the task of aligning the prongs of a plug with the ends of an outlet receptacle is not realizable, since the prongs are necessarily slightly smaller than the outlet designed to receive them. In such cases, however, the approximation is a benign one, since the performance of the task will effectively average the weight of each of the unrealizable constraints on wither end of the prong and place the plug in a position for insertion. Not all tasks involve this kind of benign averaging, however: inserting bottles into a crate, gripping a chess piece, aligning a screwdriver, and cutting a wire are tasks in which the visual specifications can, in fact, be achieved.

**Visual Search** The framework above presumes that image features can be quickly and reliably located from a coarse object description. Given the wide variation in object appearance, the problem of making effective use of model and image information to accomplish this in a general fashion is not completely solved [31].

**Monitoring** The change in appearance and self-occlusion properties of an object mean that monitoring and reacting to changes in the visual information available are imperative. In [6, 7] we describe an techniques or doing so, however much work remains to be done in this area.

**Obstacle Avoidance** We have concentrated on describing how positive tasks could be expressed and carried out. However, just as important are *negative tasks* — that is, task involving avoidance of contact. Some ideas in this direction can be found in [23].

In summary, we feel the framework outlined in this paper offers a preliminary pass at an architecture for the accurate programming of vision-based robotic tasks. The approach grounds task specification in the fundamental capabilities of the sensing system. In turn, it simplifies the difficult issues of object representation and visual search by only maintaining task-applicable information. Such a strategy, in addition to being both flexible and feasible, has the potential to lead to a larger and more integrated picture of vision and control.

**Acknowledgements** This work was supported by National Science Foundation grant IRI-9420982, and by funds provided by Yale University.

# References

[1] T.O. Binford. Generalized cylinder representation. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 321–323. Wiley, New York, 1987.

[2] M Brady. Artificial intelligence and robotics. *Artificial Intelligence*, 26:79–121, 1985.

[3] W-C. Chang, J. P. Hespanha, A.S. Morse, and G. D. Hager. Task re-encoding in vision-based control systems. to appear in the Proceedings of CDC '97.

[4] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. *Lecture Notes in Control and Information Sciences*, 237:66–78, 1998.

[5] 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.

[6] Z. Dodds, G. Hager, A.S. Morse, and J.P. Hespanha. Task specification and monitoring for uncalibrated

hand/eye coordination. In *Proc. IEEE Int. Conference Rob. Automat.*, pages 1607–1613, May 1999.

[7] Z. Dodds, M. Jägersand, and G. Hager. A hierarchical architecture for vision-based robotic manipulation tasks. In H. I. Christensen, editor, *First Int. Conf. on Computer Vision Systems*, volume 1542 of *Lecture Notes in Computer Science*, pages 312–330, January 1999.

[8] B. Espiau, F. Chaumette, and P. Rives. A New Approach to Visual Servoing in Robotics. *IEEE Trans. Robot. Automat.*, 8:313–326, 1992.

[9] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proc. European Conf. Computer Vision*, pages 563–578, 1992.

[10] Olivier Faugeras. Stratification of 3-d vision: Projective, affine, and metric representations. *J. Opt. Soc. Am. A*, 12(7):465–484, 1995.

[11] W. Eric L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints.* MIT Press, 1990.

[12] G. D. Hager. Real-time feature tracking and projective invariance as a basis for hand-eye coordination. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 533–539. IEEE Computer Society Press, 1994.

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

[14] G. D. Hager and K. Toyama. XVision: A portable substrate for real-time vision applications. *Computer Vision and Image Understanding*, 69(1), 1998.

[15] J. Hespanha, Z. Dodds, G. D. Hager, and A. S. Morse. What tasks can be performed with an uncalibrated stereo vision system? *International Journal of Computer Vision*, 35(1):65–85, 1999.

[16] João Hespanha, Zachary Dodds, Gregory D. Hager, and A. Stephen Morse. Decidability of robot positioning tasks using stereo vision systems. In *Proc. 37th Conf. on Decision and Control*, December 1998.

[17] R. Horaud, editor. *Proceedings of the IEEE/RSJ/INRIA Workshop On New Trends in Image-Based Robot Servoing.* IEEE Press, 1997.

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

[19] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *IJCV*, 5(2):195–212, November 1990.

[20] M. Jagersand and R. Nelson. Visual task specification, planning, and control. In *Proc., IEEE International Symposium on Computer Vision*, pages 521–526, 1995.

[21] D. Kriegman, G. Hager, and A.S. Morse, editors. *The Confluence of Vision and Control.* Number 237 in Lecture Notes in Control and Information Sciences. Springer-Verlag, 1998.

[22] E. Marchand. ViSP: A software environment for eye-in-hand visual servoing. In *Proc. IEEE International Conference on Robot. Automat.*, pages 3224–3229, 1999.

[23] E. Marchand and G. D. Hager. Dynamic sensor planning in visual servoing. In *Proc. IEEE International Conference on Robot. Automat.*, pages 1988–1993, 1998.

[24] D. McDermott. Planning reactive behavior: A progress report. In J. Allen, J. Hendler, and A. Tate, editors, *Innovative Approaches to Planning, Scheduling and Control*, pages 450–458. Morgan Kaufmann, San Mateo, CA, 1990.

[25] J. D. Morrow and P. K. Khosla. Manipulation task primitives for composing robot skills. In *Proc. IEEE International Conference on Robot. Automat.*, pages 3354–3359, 1997.

[26] J. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision.* MIT Press, Cambridge, Mass., 1992.

[27] H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearence. *Int. J. Computer Vision*, 14(5–24), 1995.

[28] L. Robert, C. Zeller, and O. D. Faugeras. Applications of non-metric vision to some visually guided robotics tasks. Technical Report 2584, INRIA, Sophia-Antipolis, June 1995.

[29] T. J. Schnackertz and R. A. Grupen. A control basis for visual servoing tasks. In *Proc. IEEE International Conference on Robot. Automat.*, pages 445–451, 1995.

[30] K. Toyama, J. Wang, and G. D. Hager. SERVOMATIC: a modular system for robust positioning using stereo visual servoing. In *Proc. IEEE Int'l Conf. Robot. and Automat.*, pages 2636–2643, 1996.

[31] M. Vincze, M. Ayromlou, and W. Kubinger. An integrated framework for robust real-time 3d object tracking. *Lecture Notes in Computer Science: First Int. Conf. of Computer Vision Systems*, 1542:135–150, 1999.

[32] M. Vincze and G. Hager, editors. *Robust Vision for Vision-Based Control of Motion.* IEEE Press, 1999.