ELSEVIER

Image and Vision Computing 16 (1998) 275–292

image AND vision COMPUTING

PLAYBOT
A visually-guided robot for physically disabled children

J.K. Tsotsos[a,*], G. Verghese[a], S. Dickinson[b], M. Jenkin[c], A. Jepson[a], E. Milios[c], F. Nuflo[a] S. Stevenson[b], M. Black[d], D. Metaxas[e], S. Culhane[a], Y. Ye[f], R. Mann[g]

[a]*Department of Computer Science, University of Toronto, 6 King's College Road, Toronto, Ontario, Canada M5S 3G4*
[b]*Department of Computer Science, Hill Center for Mathematical Sciences, Rutgers, The State University of New Jersey, Piscataway, New Jersey 08855*
[c]*Department of Computer Science, York University, 4700 Keele Street, Toronto, Ontario M3J 1P3, Canada.*
[d]*Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA, USA*
[e]*GRASP Laboratory, 3401 Walnut Street, Suite 300C, University of Pennsylvania, Philadelphia, PA 19104-6228, USA*
[f]*IBM Research, T.J. Watson Labs, Yorktown Heights, New York, NY, USA*
[g]*NEC Research Institute, 4 Independence Way, Princeton, NJ, USA 08540*

Received 1 November 1996; revised 1 September 1997; accepted 22 September 1997

Abstract

This paper overviews the PLAYBOT project, a long-term, large-scale research program whose goal is to provide a directable robot which may enable physically disabled children to access and manipulate toys. This domain is the first test domain, but there is nothing inherent in the design of PLAYBOT that prohibits its extension to other tasks. The research is guided by several important goals: vision is the primary sensor; vision is task directed; the robot must be able to visually search its environment; object and event recognition are basic capabilities; environments must be natural and dynamic; users and environments are assumed to be unpredictable; task direction and reactivity must be smoothly integrated; and safety is of high importance. The emphasis of the research has been on vision for the robot this is the most challenging research aspect and the major bottleneck to the development of intelligent robots. Since the control framework is behavior-based, the visual capabilities of PLAYBOT are described in terms of visual behaviors. Many of the components of PLAYBOT are briefly described and several examples of implemented sub-systems are shown. The paper concludes with a description of the current overall system implementation, and a complete example of PLAYBOT performing a simple task. © 1998 Elsevier Science B.V.

Keywords: PLAYBOT; Visually-guided robot; Physical disability; Computer vision

1. Introduction

The experimental goal of the project is the development of a prototype environment which will assist disabled children in play, thereby enhancing their innate capabilities. Imagine a child seated in a robotic wheelchair. This robot possesses a robotic arm and hand, a stereo-colour vision robot head, and a communication panel (an artist's view of the target PLAYBOT system is shown in Fig. 1, which includes the actual robot arm and head). The communication panel design is motivated by the BLISS symbolic language, invented by Charles Bliss in 1941 and successfully used by cerebral palsy patients [1]. It is assumed that the child can point to a picture of a toy (or toys) on the panel and point to a sequence of pictorially represented actions that he/she wishes the robot to perform with that toy (or toys). In effect, a play sentence is specified in terms of a grammatically formed sequence of actions. The play sequence could involve bringing toys to the child's table for close inspection and manipulation, it could also involve the starting or stopping of various automated aspects of the environment (i.e. motorized trains). The robot would visually locate the toys, plan the execution of the play sequence, and then together with the child, move to the proper position and carry out the actions.

Current robots in use for the disabled all rely on the user's intact visual system to be an integral part of a closed-loop control system. For example, in one class of robotic aids, specialized sensors are developed for a controllable muscle the user might have (a finger, eye brow, eye movement, etc.). The user decides what the robot manipulator should grasp, and then through a long series of micro-activations of the robot, visually guides the manipulator to the target

* Correspondence to: John K. Tsotsos, Dept. of Computer Science, 6 King's College Rd., University of Toronto, Toronto, Ontario, Canada M5S 1A4. Tel: 416 978 3619; fax: 416 978 1455; e-mail: tsotsos@vis.toronto.edu

PII S0262-8856(97)00088-7

Fig. 1. Artist's conception of PLAYBOT (actual child in a wheelchair on which is superimposed PLAYBOT specific components, robot arm, user interface and TRISH stereo head).

object. Each micro-activation might move a particular joint of a robot arm a small amount. The key to success here is that the user's visual system senses the progress of the arm in comparison to the target, and the user's planning system determines which micro-activation to perform next in order to move closer to the target. This can be very tedious; the user may tire easily and the amount of work that might be done in a given unit of time is very small. Nevertheless, the user does have some independence, and this is important. In the long run, however, improvements are needed. PLAYBOT's goal is to provide a further step along the path of enhanced mobility, functionality and independence for physically disabled children.

2. PLAYBOT overview

PLAYBOT is designed with the goal of 'short-circuiting' the control loop described above. The user's visual system is still needed, of course, to determine what the goal of a manipulation is, and to communicate with the robot. But the robot's visual system then takes its place in the closed-loop control of the robot in the execution of the task. The big research questions then are: what kind of interface is best so that the user's goals are easily and effectively communicated? and, how exactly is the vision problem to be solved? The remainder of this paper describes our progress in answering these two questions.

As must be obvious, this is a long-term, large-scale project. This overview paper will briefly address many of the components of PLAYBOT. The bulk of our research concerns vision, since this is central to the motivations for the project. In the following sections, overviews of the user interface, stereo vision head, object recognition, visual attention, and other aspects will be given (with references to sources for further details). It should not be too difficult to see how, in the context of PLAYBOT, each component is necessary. The paper concludes with a short example showing how the pieces fit together for a simple task, and how PLAYBOT actually executes a simple task.

Neither the research nor the implementation of the PLAYBOT system are complete at this point; it is anticipated that perhaps three or four years of further development are required to achieve a system with a broad set of competencies. Section 4 provides a description of the current implementation.

2.1. System goals

There are several guiding goals which have united the research efforts over the past six years:

1. *Vision as primary sensor.* Humans rely on vision as their major source of world input and we wished to explore vision as the primary sensor for PLAYBOT. One might consider the overall project as not only a development task with a particular application domain, but also as an experiment in furthering our understanding of how far computational vision can advance. There is no attempt in the basic research of the project to find 'tricks' or clever short-cuts to solving the vision problem. Each of the vision behaviors described represents a genuine attempt at furthering the state-of-the-art for that particular visual sub-task.
2. *Task direction.* Previous research has shown the importance of task direction as a major complexity reduction strategy for visual perception and intelligent behavior ([2]). For this reason, PLAYBOT is task-driven, the task being specified by the user. Effort has been devoted to the development of a simple, yet powerful, task direction language and user interface through which an unsophisticated user may provide natural commands to the system. It is clear that it would be highly cumbersome for a user to provide all details of task execution. Each task that may be specified may be considered as a 'macro' instruction, which is expanded into a default sequence of internal instructions.
3. *Visual search.* A PLAYBOT-like robot is useless without the ability to search the world for particular objects or events. It will not always be the case that the toys requested by the child will be in the current camera view. This search function is a component of many of the other behaviors the system; efficient search strategies are critical.

4. *Recognition.* A key component for successful execution of a PLAYBOT task is the ability to recognize objects and perceive events in the world. Much of our research is devoted to these tasks, coupled tightly to the search capability referred to above.
5. *Minimal re-engineering of environments.* In general, it is highly undesirable that environments in which intelligent robots are placed be re-engineered to simplify the robot tasks. In the case of the physically disabled, any additional environment engineering would only further emphasize the disability for the user. Thus, it would be desirable that PLAYBOT function in a 'normal' indoor environment such as one available in a home. However, even PLAYBOT must make some concessions. It is thus further assumed that the PLAYBOT system requires specific visual targets in the environment. For example, the room may be re-painted in bright colours with strong colour boundaries and wall edges may be emphasized with contrasting colours. Objects with strong contrast may be added, such as a white play table with black edges and contours. Paintings of toys or wall paper may be added, again including strong contrast. In any case, these changes may be made on system installation. They would appear to the users as natural and 'fun' decorations in their new play room; however, they also serve the critical function of providing many visual calibration targets throughout the room which PLAYBOT uses as needed.
6. *Unpredictability of environment and user.* Careful design dictates that any user of a computer system must always be considered unpredictable. Further, any real world environment must be considered to have some inherent unpredictability. PLAYBOT must be able to deal sensibly with user unpredictability: incomplete instructions; illogical instructions; improper system use; and so on. PLAYBOT should have a reasonable set of default actions and behaviors, and must be able to respond to the user in a helpful manner. The play environment will also have characteristics which require special actions: a sibling or parent entering a room; new toys being added to the room; playmates cooperating with the user; other PLAYBOT-enhanced children wishing to cooperate in play; and so forth. Again, PLAYBOT must react in the appropriate ways. One particularly interesting supplementary visual behavior of PLAYBOT is dynamic environment mapping. A typical user may take some time to express an instruction and there might be sizable gaps in time between successive instructions. Any 'idle time' of this type can be used to advantage. PLAYBOT would detect such delays and re-awaken an always active visual reconstruction behavior. This behavior would be the same object recognition behavior invoked for recognizing particular toys, but now would be used to update the 3-D map of the play environment.
7. *Reactivity.* Task direction is helpful for visual search, object recognition and event perception; however, it does not suffice for an intelligently functioning robot. PLAYBOT must be able to interrupt task execution in order to react to events in its environment such as; new persons in the room; events which have unknown consequences on the robot and child's safety; and so on. These constraints mean that if vision is the primary sensor, the sensing system must have good peripheral vision, and a robust attention system which can re-direct the system's gaze if needed. Further, it may be that in order to satisfy fully safety constraints, other sensor schemes must be added to act as the 'eyes in the back of the head'. PLAYBOT currently does not have these extra eyes. The future addition of 'ears' to the PLAYBOT head will assist, because auditory stimuli may be localized and, if out of camera view, can direct the head to fixate the source.
8. *Safety.* Little need be added here; it is clear that a PLAYBOT system must satisfy the most stringent guidelines.

To the above, one must add robustness and reliability of the overall system under changes in lighting, playroom characteristics, and other such variations.

Due to the size of the project, we acknowledge at the outset that we do not do justice to related research in this presentation. The reader is encouraged to look at the literature cited in order to see proper comparisons to other work. Previous descriptions of PLAYBOT have appeared in [3,4].

2.2. New hardware components

The user interface for the child is critical and for this a new hardware and software design is proposed. The ActiveDeskTop is a large-scale, touch-sensitive video display [5]. It is the table-like object in front of the child in Fig. 1. The video surface will be constructed by adjoining a large number of flat screen display devices of a given shape and size, tiling some arbitrary surface. Each screen on this surface is controlled by its own computer processor, and these processors are networked together. Superimposed over this surface is a tiling of transparent touch-sensitive screens, which allow for user input. Each display unit contains its own processing sub-system networked to all the others. In this way, each processor is responsible for both generating the graphics for its own tile as well as dealing with any local touch interactions. Each processor is also responsible for communications with the remainder of the network as required in order to achieve the global results desired. The resulting display device is thin, has a very high resolution, appears to be a single large screen to the user, and is capable of supporting many different types of human–machine interaction.

On this hardware substrate, a user interface is designed. For PLAYBOT, there are a number of specific requirements that must be met to ensure that the child is able to communicate with the robot in a natural manner. The child must be

told what toys might be on the table and what actions the robot can perform on those toys. The child must be able to easily specify the toys that she wants the robot to manipulate, and the actions that she wants the robot to perform on them. The interface must show the child what the robot sees, so that she understands what objects the robot knows about within its environment. How these requirements are met is highly restricted by the fact that not only is the intended user not computer-literate, she is also unable to use the typical means of communicating with a computer through a keyboard and a mouse.

At this time, the ActiveDeskTop has not been constructed. We use a standard workstation display with mouse interaction as the interface. However, the details of the interface are well-defined and implemented, as can be seen in Fig. 2. There are several windows within this interface:

1. *Actions*. The 'verbs' of the command language are depicted using animated buttons (real hand and object performing the action) in individual icons. A user may touch this button in order to request the action. The actions *grasp*, *place* and *push* are shown in Fig. 2.
2. *Objects*. The 'nouns' of the command language are depicted using actual images in individual icons. A touch of this button requests the toy. A touch on an object in the camera or scene views will also suffice to specify an object. In these instances, however, the object recognition behavior must be invoked to recognize the object at the specified location so that its type is known. Blocks of different colours are shown in the figure.
3. *Locations*. Objects may be placed in specific spatial relationships with other objects, such as on top of, to the left of, or to the right of, some other object. These three relationships are shown in Fig. 2 as red blocks related to black blocks with an arrow.
4. *Scene View*. A graphical depiction of the actual environment in which PLAYBOT and the play table exists is shown here from the user's current point of view, i.e. this view dynamically changes with user movement. Users may touch objects or locations within the window, including room locations to which they may wish to be moved. The table and room are pre-specified; the table contents are recovered by PLAYBOT as it operates.
5. *Left and Right Camera Views*. The actual images seen by the stereo head are shown. Users may touch objects or locations within the windows.
6. *Message Window*. Feedback from the system to the user may appear here.
7. *Parameters*. Additional buttons are provided here for

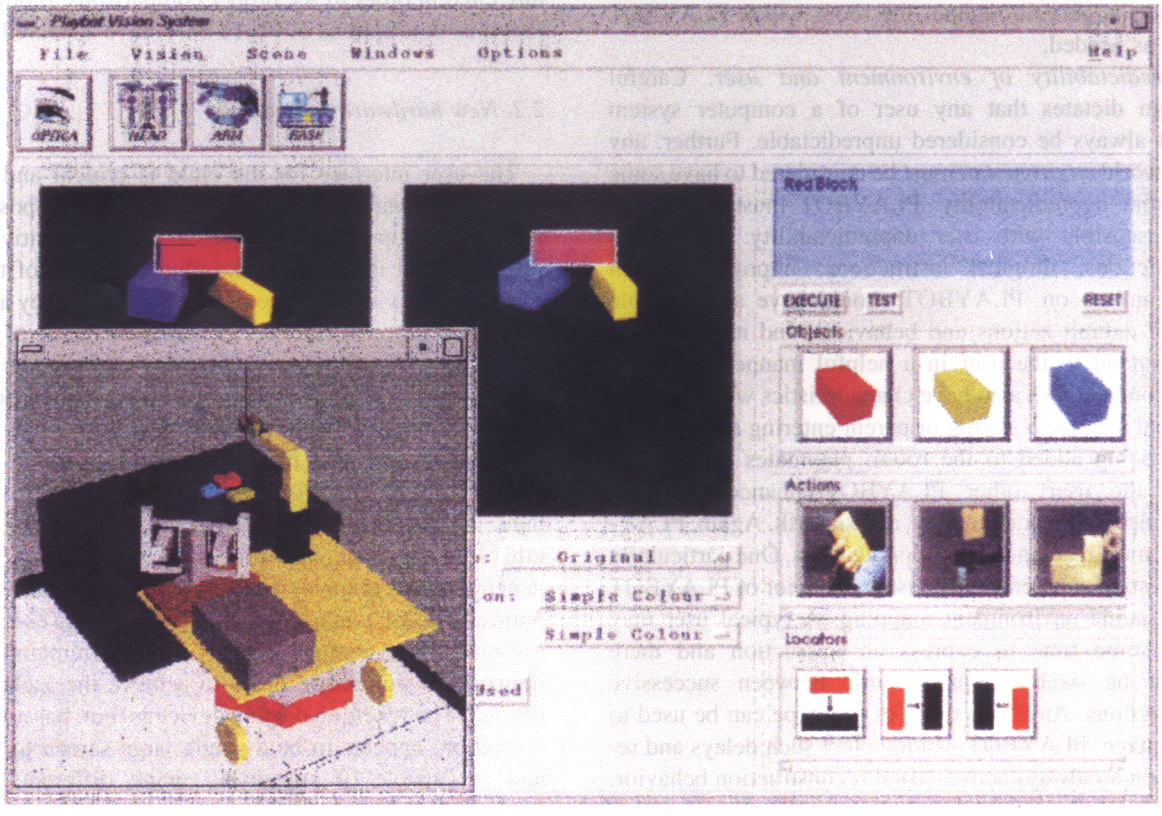


Fig. 2. The current interface. The correspondence with the window and icon descriptions in the text is: *Actions*—right hand side, third row of icons from the top; *Objects*—right hand side, second row of icons; *Locations*—right hand side, bottom row of icons; *Scene View*—window in bottom left corner; *Left and Right Camera Views*—left hand side, middle two windows; *Message Window*—right hand side, top window; *Parameters*—four icons in top left corner; *Commands*—hidden. The bottom right hand corner is just screen background and may be ignored.

users who may wish to 'play' with the robot itself; that is, to move the arm, platform, etc., independently of a play command.

8. *Commands*. System debugging windows and related information.

Note that: (1) the windows permit users to specify objects and locations in many different ways, but actions in only one way; and (2) there are scroll buttons for each of the object, action and location icons since all cannot be displayed simultaneously. The grammar which ties these concepts together is presented below.

The current objects are geons (see Section 3.1.3): Block; Cone; Cylinder; Pyramid; Truncated Cone; Truncated Pyramid; Ellipsoid; Bent Block; Bent Cylinder; Truncated Ellipsoid. Although these are not really toys, a large number of objects may be constructed out of combinations of geons. Thus, we expect that the representation of many simple toys will not prove difficult, once recognition of geons is a basic competence of the system. The current actions are: Pickup; Drop; Push; Bring; Inspect; Separate; Place; Locate; Assemble; Disassemble; Use; and Move-Me-To.

A second new hardware addition is that of the finger sensors. We use the CRS Plus robot arm, a five degree of freedom robot with a gripper that has a two-finger force sensitive manipulator. It can move the parallel fingers together or apart, and is able to limit the force it exerts on the object between its fingers. We have constructed infra-red sensors, and embedded them within new finger attachments to allow the new fingers to sense the presence or absence of solid objects in their vicinity [6]. The specific algorithm employed to perform this function is presented later.

The final major new hardware component in PLAYBOT is the vision sensor. TRISH (**T**oronto **IRIS** Stereo **H**ead) is a robotically controlled binocular head, consisting of two, fixed focal length, colour cameras with automatic gain control forming a verging stereo pair [7]. TRISH is capable of version (rotation of the eyes about the vertical axis so as to maintain a constant disparity), vergence (rotation of the eyes about the vertical axis so as to change the disparity), pan (rotation of the entire head about the vertical axis), and independent tilt (rotation of each eye about the horizontal axis).

One novel characteristic of the design is that each camera can rotate about its own optical axes (torsion). Torsion movement makes it possible to minimize the vertical component of the 2-D search which is associated with stereo processing in verging stereo systems. Thus, TRISH is a seven degree of freedom robot. The utility of these degrees of freedom is such that not only can TRISH point to different locations in the world, but also actively control the geometry of the head to reflect the best horopter arrangement for stereo processing given a task. The control and use of TRISH is beyond the scope of this paper (see [8,9,7,10]). TRISH is seen in Fig. 1 in the upper center-left region, and plays a prominent role in the overall examples later in the paper.

2.3. An example scenario

Suppose a user wishes to find a cube in order to put it onto an existing stack of blocks. The play sentence might be something like: touch pickup icon, touch cube icon, touch place icon, touch location in the scene view. Exactly how would PLAYBOT execute this task? One might imagine that a wide variety of visual, reasoning, and robot motion behaviors must be invoked in the proper sequence, behaviors such as:

- look for the cube
- move robot so that cube is within reach of robot arm while fixating cube and stabilizing for robot's motion
- determine robot manipulator path and grasp parameters for cube
- grasp cube and retract arm from play table area

Now suppose the task was changed slightly to be: pick up cube, place in boxcar of moving train (which could simply be pointed to in the scene view window). How is the above solution modified? The following must be inserted:

- locate train boxcar
- track boxcar model
- plan robot manipulator path and cube pose parameters so that cube might be dropped into the boxcar when it passes under cube position
- determine timing of drop given tracked boxcar motion parameters and knowledge of train track path

Our research has as a primary goal the development of solutions for the visual behaviors required by tasks of this complexity. A secondary, but no less important, goal has been the development of behaviors for the non-visual tasks which are necessary to execute such tasks. The next section will briefly overview the behaviors investigated in the project.

3. Behaviors to support PLAYBOT

The approach chosen to imbue PLAYBOT with useful intelligence is based on a set of interacting behaviors. Although behavior-based, the limitations of previous behavior-based robot architectures are not found in PLAYBOT. Those limitations are documented and discussed in [2]. We take a radically different view, and intend to extend the applicability of a behavior-based framework to include behaviors which reason, perceive, manipulate, and so on. The controlling framework is S*, described in [11], and a brief overview will appear below.

As a result, all actions of the system are defined as behaviors which take some set of representations as input and act on other representations. PLAYBOT thus contains a

broad collection of behaviors: visual; kinematic; language understanding; and so on. The following sections describe the behaviors within these broad classes. No claims are made about the completeness of this behavior set at this point.

3.1. Visual behaviors

The list of visual behaviors which the PLAYBOT team has investigated is substantial, and includes:

- visual attention
- gaze stabilization
- object recognition
- active object recognition
- object tracking
- object search
- event perception
- calibration
- hand–eye coordination

The list of behaviors is by no means complete nor claimed sufficient. For example, visual collision detection, visual floor anomaly detection, and recognition of flexible objects have not yet been addressed. The behaviors will be integrated using the control framework described in Section 3.3 and, for the purposes of this paper, may be considered as autonomously and continuously operating robot capabilities that may be tuned by task demands. The remainder of Section 3.1 reports progress on each of these problems, each sub-section providing a brief description plus pointers to available literature for further details. The best motivation for these behaviors is to simply consider how humans perform everyday manipulation tasks; all of the above are part of our normal cognitive repertoire. The goal of PLAYBOT is to perform using similar capabilities.

3.1.1. Visual attention

Vision (or perception systems in general) requires attention mechanisms, because the search space is too large [3]. The model is based on the concept of selective tuning. It provides for a solution to the problems of selection in an image, information routing through the visual processing hierarchy, and task-specific attentional bias. The central thesis is that attention acts to optimize the search procedure inherent in a solution to vision. Attention does so by selectively tuning the visual processing network, which is otherwise an approximate process for recognition. The selective tuning is accomplished by a top-down hierarchy of winner-take-all processes embedded within the visual processing pyramid.

The use of attention is to locate the strongest instances of object features which are relevant to objects which are being searched for in the world. The instances are found in order from strongest to weakest, and may point to matching candidates for the object recognition system. Several different features may be searched for simultaneously depending on the specification of the target. An example sequence using only blue colour saliency is shown in Fig. 3.

We have tested the attention system implementation using separate representations of luminance, edges, colour, abrupt onsets and offsets, peripheral visual field, and optic flow pattern correlations [12]. Two of these representations provide input to the TRISH control system; both the abrupt onset/offset detector and the peripheral event detector may direct the head to fixate particular positions. In the former case, the head would fixate on potentially unexpected changes in the visual environment, while in the latter, the head has an input that assists in its exploration of the visual world outside the current image. The onset/offset mechanism is particularly important for detecting potentially hazardous situations affecting the safety of the robot (and, more importantly, the child).

3.1.2. Gaze stabilization

Suppose PLAYBOT had only one eye, that is, its camera system was monocular. When PLAYBOT moves, how is the object of interest stabilized in the images it sees? This is important, since after the robot moves, it would be desirable for it to not have to search again for the object of interest. A method based on mixture models of motion which can compensate for purposeful or accidental motion of the robot's camera system has been developed [13]. A point or object must be identified as the one to be stabilized.

The computation of optical flow relies on merging information available over an image patch to form an estimate of 2-D image velocity at a point. This merging process raises a host of issues, which includes the treatment of outliers in component velocity measurements and the modeling of multiple motions within a patch which arise from occlusion

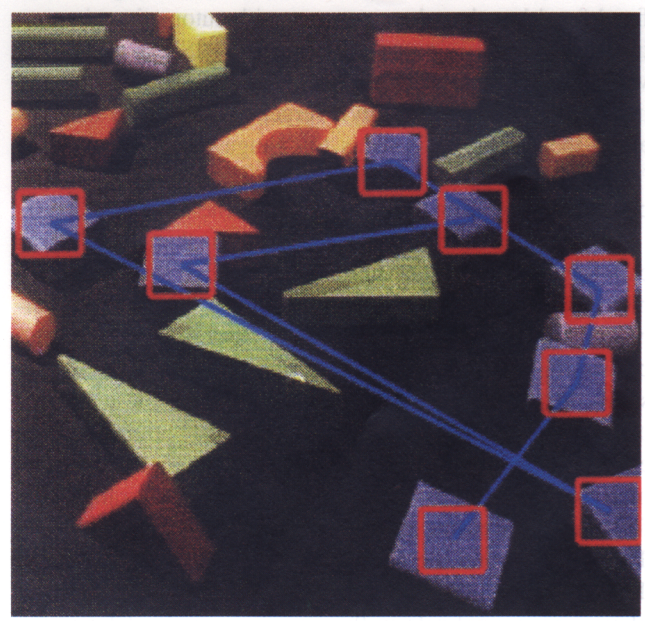


Fig. 3. Selectively fixating on blue objects in a typical image of objects; candidate regions for the object recognition system are localized.

boundaries or transparency. Our approach is based on the use of a probabilistic mixture model to explicitly represent multiple motions within an image patch. We use a simple extension of the EM-algorithm to compute a maximum likelihood estimate for the various motion parameters. Preliminary experiments indicate that this approach is computationally efficient and can provide robust estimates of the optical flow values in the presence of outliers and multiple motions. The basic approach can also be applied to other problems in computational vision, such as the computation of 3-D relative motion, which require the integration of several partial constraints to obtain a desired quantity. An example is shown in Fig. 4.

Using the TRISH stereo camera head, however, a different solution is available for gaze stabilization for PLAYBOT. The object tracking algorithm described in Section 3.1.5 plus stereo position triangulation suffice to solve the problem. This is the method used in the example shown in Section 4. However these two methods are complementary. The monocular method does not require an object model for tracking, and can operate in cluttered and textured scenes. The stereo method does require a model, and is best suited for objects where line and point features are readily detectable. It is clear that if each of the methods is used in their appropriate contexts, the overall robustness of PLAYBOT with respect to gaze stabilization is greatly enhanced.

3.1.3. Object recognition

A new approach to shape recovery for 3-D object recognition has been developed, that uses qualitative shape recovery and recognition techniques to provide strong fitting constraints on physics-based deformable model recovery techniques. Deformable models are fit to occluding image contours in image data captured under general orthographic, perspective, and stereo projections. On one hand, the integration of qualitative knowledge of the object being fit to the data with knowledge of occlusion supports a much more robust and accurate quantitative fitting. On the other hand, recovering object pose and quantitative surface shape not only provides a richer description for indexing, but supports interaction with the world when object manipulation is required [14,15].

The object recognition system must support the following four recognition behaviors (in addition to being used by the object search algorithm):

- Unconstrained bottom-up recognition: the system must be able to identify as well as locate in space all objects within the field of view. In the system, this behavior would be invoked as a background behavior when the child is not executing a task.
- Constrained bottom-up recognition: the system must be able to identify as well as locate in space the object at a particular location within the field of view. This behavior would be invoked when the child points to some object in the interface windows.
- Unconstrained top-down recognition: the system must be able to search the image for a particular object.
- Constrained top-down recognition: the system must be able to search a particular location for a particular object. This might entail movement of the sensors so that the location specified is seen.

Objects are modelled as object-centered constructions of qualitatively-defined volumetric parts chosen from some arbitrary, finite set. The part classes are qualitative, in the sense that they are invariant to degree of curvature, relative dimensions, degree of tapering, etc. *Aspects* are used to

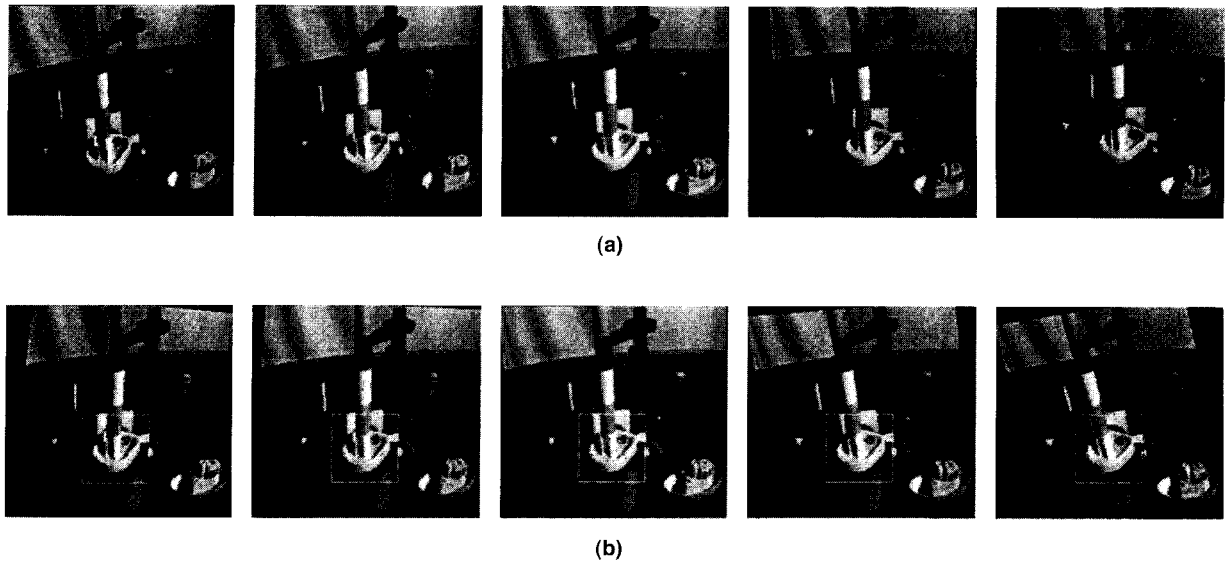


Fig. 4. The top sequence shows five images of an image sequence, taken by a camera moving roughly upwards and to the right along an unknown path. The bottom sequence shows the result after applying the stabilization algorithm fixating on the base of the toy tower. Although the rest of the image is severely warped, the fixation region holds the tower base firmly in position throughout the camera motion.

represent these volumetric parts. Consequently, the goal is to use aspects to recover the 3-D volumetric parts that make up the object, in order to carry out a recognition-by-parts procedure, rather than attempting to use aspects to recognize entire objects. The advantage of this approach is that since the number of qualitatively different volumes is generally small, the number of possible aspects is limited and, more important, independent of the number of objects in the database. The disadvantage is that if a volumetric part is occluded from a given 3-D viewpoint, its projected aspect in the image will also be occluded. Thus we must accommodate the matching of occluded aspects, which we accomplish by use of a hierarchical representation we call the aspect hierarchy.

The aspect hierarchy consists of three levels, consisting of the set of aspects that model the chosen volumes, the set of component faces of the aspects, and the set of boundary groups representing all subsets of contours bounding the faces. The ambiguous mappings between the levels of the aspect hierarchy are captured in a set of upward and downward conditional probabilities, mapping boundary groups to faces, faces to aspects, and aspects to volumes. The probabilities are estimated from a frequency analysis of features viewed over a sampled viewing sphere centered on each of the ten volume classes. Fig. 5 shows two recovered parts, which together make up the volume which is a cup.

To find instances of a target object in the image, a Bayesian approach is employed which exploits the probabilities in the aspect hierarchy. Given a target object, the utility of searching for its various volumetric subparts is computed. Similarly, the utility of searching for the various aspects of the target volume was determined, as was the utility of searching for the various faces of a target aspect. Since the preprocessed face topology graph contained at each node a number of face hypotheses ranked in decreasing probability, a set of ranked search locations was defined from which the target aspect, volume, and object could be recovered.

3.1.4. Active recognition

In general, single-view object recognition is subject to many difficulties, mainly due to viewpoint-related ambiguities, occlusions and coincidences. Recognition which is active, that is, that has the ability to vary viewpoint according to the interpretation status, overcomes many of these difficulties. We have investigated two approaches to this problem.

In the first approach, the aspect hierarchy is used [16]. If, for example, in a top-down recognition task, a particular volume is recovered from an aspect that is ambiguous, i.e. the conditional probability mapping the aspect to the volume is less than 1.0, then the system should be able to guide the camera to an unambiguous viewpoint. By combining a single-part aspect graph with the probabilities associated with the hierarchy, we derived a representation, called the aspect prediction graph, that could support the following queries: 1) Is there a less ambiguous view of a given recovered volume? 2) If so, in what direction relative to the aspect-centered coordinate system defined by the recovered volume's aspect should the camera move? 3) Finally, what visual events (appearing/disappearing faces) will be encountered as the camera is moved in that direction? In preliminary experiments, the approach was successfully demonstrated on single-part objects drawn from our shape vocabulary, such as is shown in Fig. 6.

Wilkes and Tsotsos [17] have developed a different active recognition strategy. The system has been demonstrated on images of jumbled piles of origami objects. Each object in the model base has associated with it a set of special views, viewpoints in the sphere around an object from where its features may be most easily detected unambiguously. A

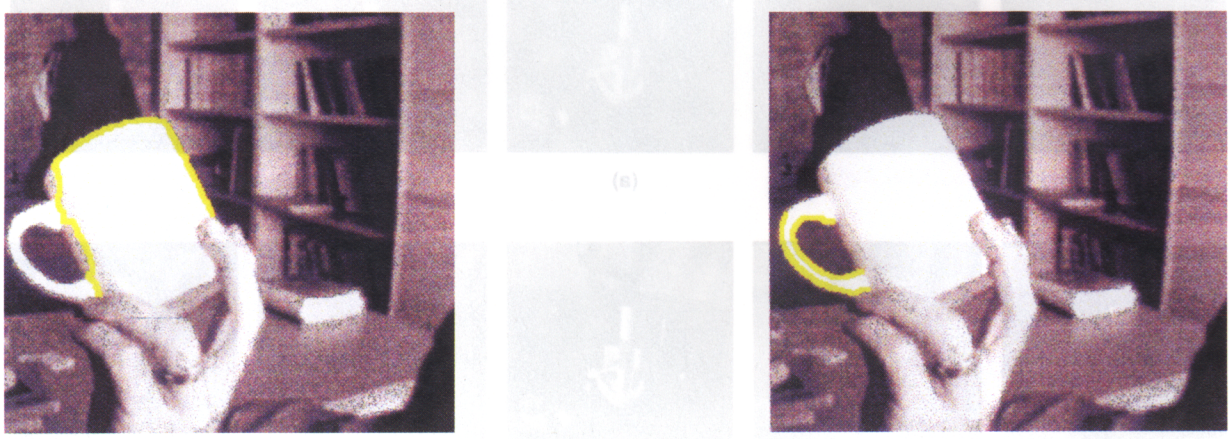
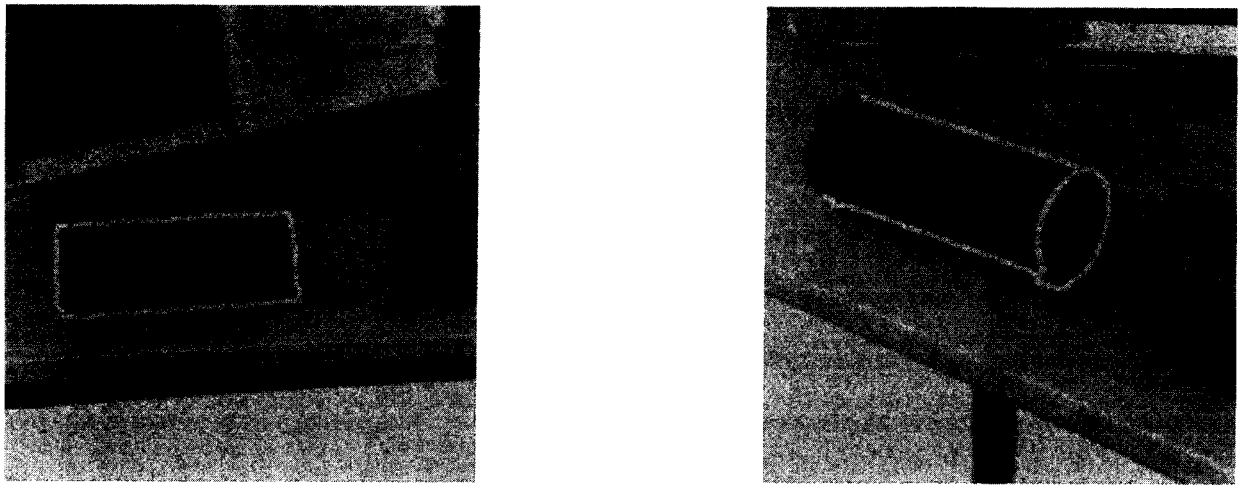


Fig. 5. Aspect-based object recognition by parts. The cup is composed of two parts, the handle and the body. Although the cup grasp and pose are perhaps not those usually encountered in day-to-day use, the example is illustrative of the recognition process. The parts must be visible in an image in order to be recognized and this is possible in a wide variety of poses.



a b

Fig. 6. An example of active object recognition. The first view of the cylinder is ambiguous; the object might be a cylinder or it may be a cube. Shifting the imaging system geometry reveals the circular base of the object, confirming that it is a cylinder.

simple behavior-based viewpoint control is used in order to achieve the robustness necessary to reach a special view reliably. The behaviors are interesting in that they are driven primarily by the current image data, making little use of inference concerning the 3-D structure of the scene. Qualitatively, the three behaviors perform *image-line-centering, image-line-following* and *camera-distance-correcting*. Together, these form the motion control component in Fig. 7.

Probabilistic algorithms are used for efficient storage and retrieval of sets of feature vectors. An error model is used to prune the search for the best model for a given query. The use of an error model also allows quantification of the degree of ambiguity in object identification. A method is provided for selecting additional special views in the case in which there remains uncertainty in the identity of the object of interest from the first special view acquired. Examples can be found in Refs [17,18].

The first method is object face (aspect) based, while the second uses line features. The first method is more appropriate to situations where the object has a simple model, where faces are mostly un-occluded, and the part decomposition is simple. The second method is preferred where there is significant occlusion initially, and for objects where the volumetric modelling approach becomes less obvious. PLAYBOT will have both active recognition behaviors, thus enabling more robust object recognition in difficult settings.

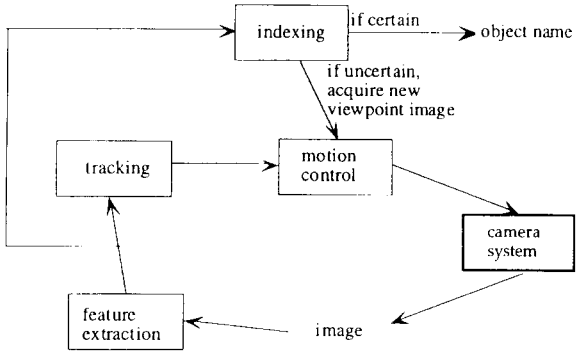


Fig. 7. Control of an active recognition strategy.

3.1.5. Object tracking

For objects in motion in the environment, or objects with induced image motion due to PLAYBOT's motion, a method is required for reliable tracking even under severe occlusion situations (such as a human hand picking up a fallen or new toy and placing it on the table). Perspective Alignment is a new analytic method for performing (non-linear) back-projection from 2-D to 3-D in real-time monocular model-based tracking [19,20]. It avoids the instabilities and false positives associated with iterative locally-linear approximation methods. More importantly, it provides maximally constrained pose solutions in under-constrained situations (caused by occlusion, noise, etc.), and thereby avoids the need for combinatorial re-recognition in many of these situations.

The overall tracking algorithm has a pipeline organization with four stages: edge detection; feature tracking; validation; and perspective alignment. The edge detection stage uses a standard magnitude of gradient operator. The feature tracking stage maintains a set of 2-D line-segments originally projected from the object model. The goal of this stage is to maintain each segment's correspondence with image edge peaks. The validation stage maintains and monitors the persistence and reliability of all line-segments. The validation stage produces an ordered list of image line-segments and corresponding paired model edges. The perspective alignment back-projection stage considers features in the

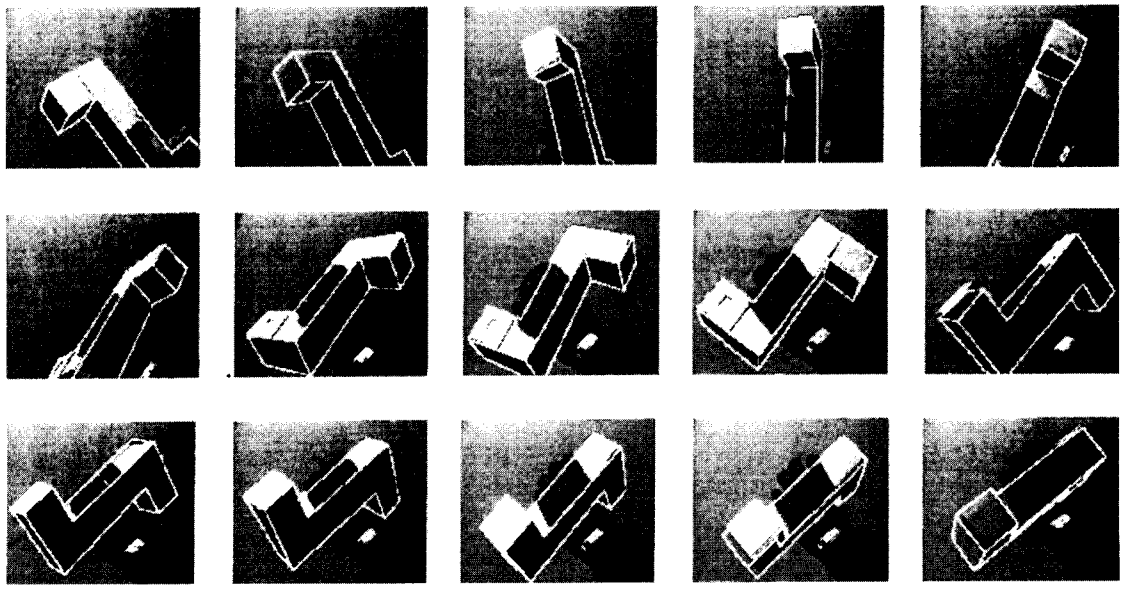


Fig. 8. The perspective alignment method tracking an object moved by hand.

order provided by the previous stage. Analytic relationships between real-world features and their perspective projections are used to allow each feature in the sequence to contribute the maximal additional geometric constraint to the previous constraints on object pose. Thus, as many degrees of freedom as possible are determined given the available feature information. An example is shown in Fig. 8.

3.1.6. Object search

Although an attention system may locate good candidates for matching to object models in a single image, and may help with some forms of eye movements as described above, it does not by itself suffice. If the object sought is not within the current image, then a strategy for locating it efficiently is needed. Ye and Tsotsos [21–23] have developed an approach to visual search for 3-D objects in the real world.

Object search is the task of finding a given 3-D object in a given 3-D environment. The searcher is assumed to be a mobile platform equipped with a camera and a method of calculating depth, like stereo or a laser range finder. The search space is characterized by the probability distribution of the presence of the target. A priori knowledge may be incorporated into these distributions. In the example of Fig. 9, suppose that a priori knowledge is available specifying that a target (a baseball) is more likely to be found on a table, so those regions have initially higher probability. The table position and characteristics are given in advance; in general, this method requires that the environment be partially known. The goal is to find a sequence of sensing parameters that will maximize the expected probability of detecting the target for a given time constraint. In this way, the problem of search is transformed into an optimization problem, where the best sequence of actions is sought that will reliably find the object. An action here is a sensing action which may involve robot motion and setting of sensing parameters such as camera pan-tilt angles, distance to candidate, vergence angles, etc. In other words, the acquisition of the next image to be searched is entirely determined by the robot, using current probability distributions. The

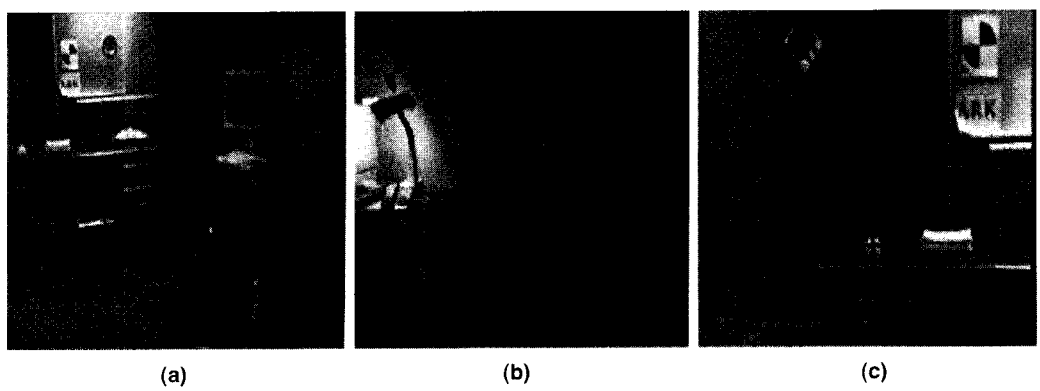


Fig. 9. Object search example. (a), (b) and (c) show the sequence of images acquired in the search for a baseball; when the ball is found in the third image, it is marked by a cross. Initial probability distributions biased search for table tops.

detecting ability of a recognition algorithm is characterized by the probability of successful recognition when the sensing parameters and the target center is given. The probability of detecting the target by applying a given operation can be calculated by combining the current probability distribution of the search space and the detection probability. An objective function is then used to locate the highest probability actions to be tried next, and these actions determine camera viewing angle size and viewing direction. The result of the sensing operation is used to update the status of the search space using Bayes' law, and this updated status is used to direct the next operation. An example is shown in Fig. 9.

3.1.7. Event perception

Understanding observations of image sequences requires one to reason about qualitative scene dynamics, that is, in terms of the forces acting on objects and the nature of forces between interacting objects. For example, on observing a hand lifting a cup, we may infer that an 'active' hand is applying an upwards force (by grasping) on a 'passive' cup. In order to perform such reasoning we require an *ontology* that describes object properties and the generation and transfer of forces in the scene. Such an ontology could include, for example: the presence of gravity; the presence of a ground plane; whether objects are active or passive; whether objects are contacting and/or attached to other objects; and so on (see Fig. 10). In this work we make these ideas precise by presenting an implemented computational system that derives symbolic force-dynamic descriptions directly from camera input.

The approach to scene dynamics is based on an analysis of the Newtonian mechanics of a simplified scene model. The critical requirement is that, given image sequences, one can obtain estimates for the shape and motion of the objects in the scene. It is assumed that the scene can be described by a collection of rigid bodies in continuous motion. Furthermore, it is assumed that the objects can be approximated by a 2-D 'layered' scene model. Given such a representation a system that extracts force-dynamic descriptions directly from camera input can be developed. Several computational examples demonstrate that the ontology is sufficiently rich to describe a wide variety of image sequences [24].

This work makes three central contributions. First, an ontology suitable for describing object properties and the generation and transfer of forces in the scene is provided. Second, a computational procedure is defined to test the feasibility of such interpretations by reducing the problem to a feasibility test in linear programming. Finally, a theory of preference ordering between multiple interpretations along with an efficient computational procedure to determine maximal elements in such orderings is employed [25].

An example is given in Fig. 11, showing the results for a complicated image sequence. In this sequence three boxes are arranged in the form of an arch. A hand approaches and pulls the left box out from the arch. When the left box is removed, the top block tips, falls, slides and finally comes to rest on the table top. In Frame 45 there are two preferred interpretations: either the hand is 'pulling' the left block or the left block is 'carrying' the hand. In both cases, the hand is attached to the block. At Frame 52, the top block is tipping while the hand continues to pull the left block away. Note that even though there is significant acceleration of the top block, the system sees it as a passive object since the motion can be explained by gravity. As with Frame 45, however, there is still ambiguity as to whether the hand or the left block is causing the remaining motion in the image. Additional details and other examples are provided in Ref. [24].

3.1.8. Calibration

Robotic tasks are plagued by the problem of calibration, and PLAYBOT is no different. In order to maintain a true representation of the world, and in order to be able to execute user requests, the robot must ensure that errors inherent with robot motions, or with unpredicted (but natural) changes in the environment, do not interfere.

PLAYBOT uses natural objects of the environment for calibration. A calibration behavior is available whose results may be accessed by any other part of the system. This behavior is based on the object tracking algorithm described earlier. If natural objects, with known 3-D models, are available, then as the robot moves about, they can be tracked and can be used as calibration targets. Of course, several such objects must be available in several different areas of the room, to ensure that at least one calibration object is in the scene at all times. For the current implementation of PLAYBOT, a set of small platforms on the play table, with distinctive white edges are used. The walls of the room itself,

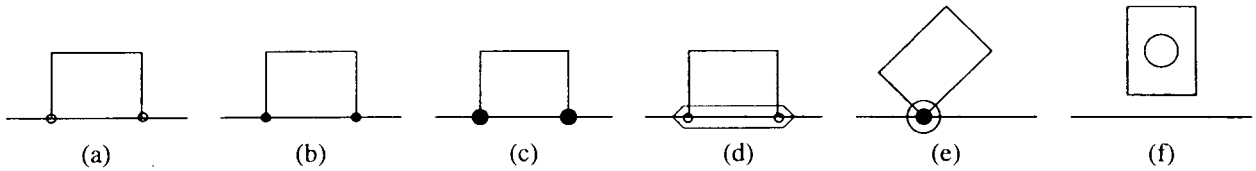


Fig. 10. In the presentation of results to follow, we use: (a) small circles to depict sliding contact; (b) small disks for non-sliding contact; (c) large disks for attachment; while (d), (e) and (f) depict a linear motor, angular motor, and body motor, respectively. For the first two motors, the closed curve surrounds the contact region over which the motors operate, while for body motors the large circle is placed at the object center.

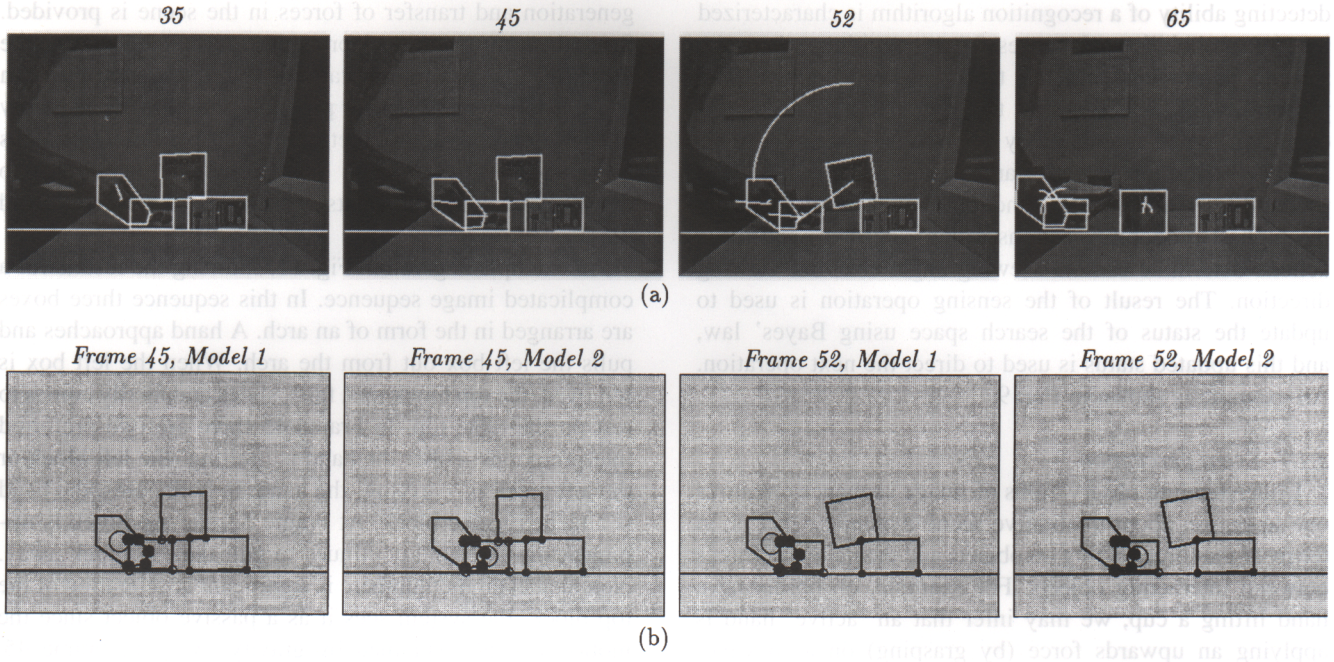


Fig. 11. Results for the arch sequence. (a) Frames with overlayed objects and acceleration vectors. Angular accelerations are denoted by the arcs. (b) Preferred interpretations for selected frames of the sequence. The numbers on the top of each image in part (a) are the frame numbers of the image sequence.

distinct drawings on the walls or floors or ceiling can be added (colourful wall drawings would only enhance the room's appearance in a real setting). Fig. 12 shows how these calibration objects are tracked as the robot moves to grasp an object. The perspective alignment method is invaluable in such examples, since it does not lose tracking with occlusion, due to its incremental constraint strategy.

3.1.9. Hand–eye coordination

Although not a specific separate behavior, all actions performed by PLAYBOT require the cameras to track the robot arm during the execution of the action. This is desirable from the perspective of ensuring the action is properly completed. Tracking of the arm and/or hand is accomplished in the same way that object tracking is done in general. Visual servoing is employed to achieve camera

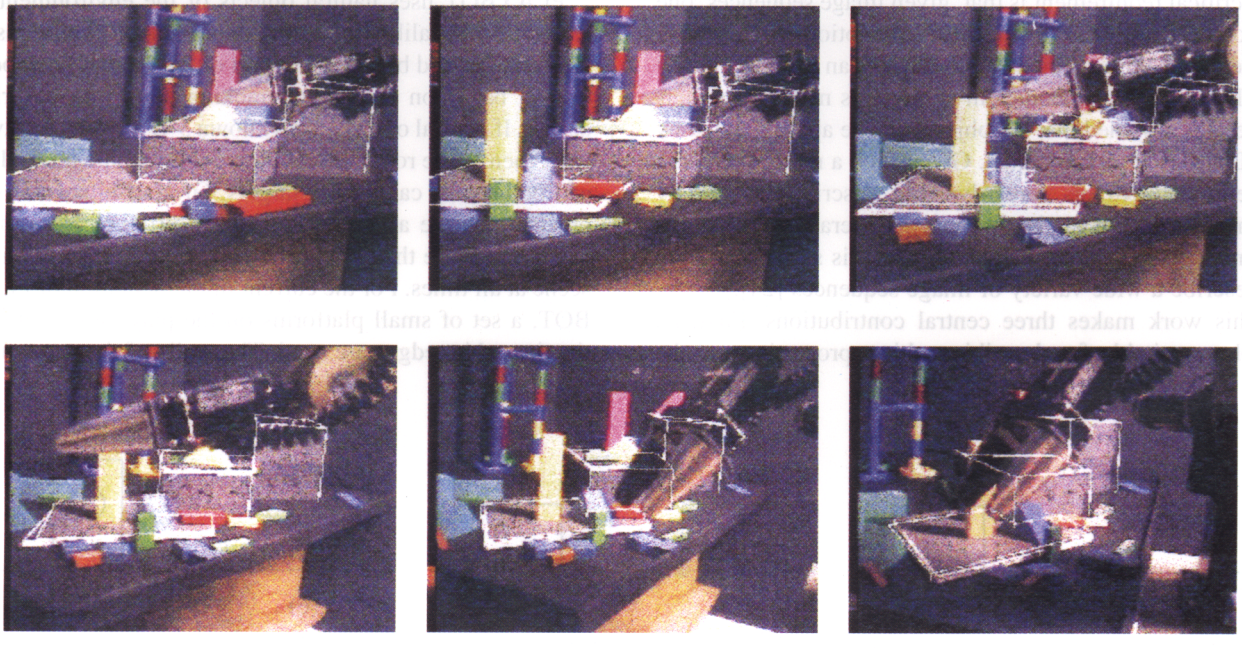


Fig. 12. Tracking of calibration target, a set of natural platforms on the play table. Tracking is not lost even though the robot arm and fingers present large amounts of occlusion.

viewpoints such that both the object to be manipulated as well as the robot are in the same view. The complete example presented in Section 4 shows these functions as well as details of the algorithm.

3.2. Non-visual behaviors

There are two major types of non-visual behaviors that are currently part of PLAYBOT. First, there is the language parsing and semantic analysis component, which reads the sequence of touches on the ActiveDeskTop and translates them into well-formed commands for the robot. The second is the object grasping behavior. It is likely that other non-visual behaviors will soon become part of the system, such as sonar-based collision avoidance, auditory attentional cues, sensors on toys themselves which provide input for behaviors that control their functions, and so on.

3.2.1. PLAYBOT command language

In order to provide an efficient command interface that a child can quickly master, the command language consists of a highly restricted subset of English imperatives. The possible commands are described by a simple semantic grammar, whose terminals are expanded by a simple syntactic grammar. The sentences of the language have a flat structure, which ensures that there are no ambiguous constructions.

The semantic grammar defines four types of commands:

Command ← Action
| Action Object
| Action Location
| Action Object Location

Each action in the language can appear in exactly one of these rules. The appropriate rule is associated with each action, and describes the arguments the action must occur with in a command specification. The first rule captures the semantics of actions that take no arguments (e.g. Quit). The second rule is for actions that manipulate a toy (e.g. Pick up). The third rule describes actions that take a single location argument (for example, Inspect). The fourth rule covers actions that manipulate a toy in terms of a location (e.g. Push). The syntactic grammar is as follows:

Action ← verb

Object ← {adjective} noun

Location ← preposition Object

Adjectives may be colours or sizes, whereas prepositions specify locations such as *on* or *beside*. By expanding the semantic and syntactic rules, we can derive a general description of commands in the language as:

Command ← verb [{adjective} noun] [preposition {adjective} noun]

Thus the linguistic knowledge of the interface can be divided into a single general syntactic rule and specific semantic knowledge associated with individual verbs. Commands may be concatenated in a list, each beginning with an action as long as each is well-formed.

3.2.2. Object grasp behavior

The CRS robot arm's gripper can rotate about the central axis parallel to the fingers. A full rotation takes about 1 s, during which time we can reliably sample the forward finger sensor about 800 times. With the fingers separated fully, these samples trace out a circle (of radius 1 inch). We analyze the samples to determine which sectors of the circle the object occupies and which represent free space into which the fingers could be placed for grasping. This information is used for grasp planning in the absence of object model or pose information.

The object grasp behavior begins with the assumption that the gripper is pointing downward, and the forward sensor senses the object of interest. The goal is to move the gripper to a position and orientation where part of the object of interest is between the fingers.

The first step is to open the fingers (to the maximum 2 inches). Then we perform the following loop:

1. Rotate gripper to obtain circular occupancy map from finger sensor.
2. If all free space, check for small object, end loop.
3. If no free space, object too large to pick up, end loop.
4. If free space in only one sector of sensor circle try to find two disjoint free space sectors move arm maximum amount in direction that reduces size of free space sector and repeat from Step 1.
5. If free space in two disjoint sectors of sensor circle, end loop.

After the above loop terminates, we know whether: (1) the object is too large to pick up, in which case we return with failure; (2) the object is very small, in which case the gripper orientation for pick-up is not critical; or (3) we can pick the object up provided the fingers are n free space, in which case we calculate the amount by which to rotate the gripper. If we have not returned with failure, we perform the final steps to place the fingers around part of the object of interest:

1. Rotate gripper to place fingers in free space.
2. Lower gripper until object sensed between fingers.

3.3. Integration of behaviors

In subsumption-style behaviour definitions, a behaviour

acts directly on the physical world. In S*, the overall control framework for PLAYBOT, this notion is generalized: the 'world' on which a behaviour may act may be an internal (logical) representation or an external (physical) representation. The world is added to the sense-model-plan-act (SMPA) cycle. A behavior is taken to mean any process which uses input available in one or more representations and causes an effect to one or more (may be the same) representations. Each behavior is represented as such an SMPA-W cycle. Behaviors may thus act on the external world, by manipulating physical objects or causing the robot to move, or may act on the internal world of the robot. That is, a behavior may manipulate internal representations, taking input from an internal representation and making changes to or creating another internal representation for use by subsequent processes. Intermediate representations, hierarchical organizations, attentive selection, and explicit goals are thus all facilitated.

Whereas the sense, model, plan and act portions of a control cycle are commonly used and understood, the world node employed here as the fifth node of the cycle requires further elaboration. The world node contains two representations, an event window and an action window. The event window opens up (or makes accessible) a relevant portion of some set of representations within the system. Similarly, the action window opens up onto some set of representations (may be overlapping with the event window). The world node also contains a set of demons which monitor the contents of the event window. These demons detect changes to the event representations of relevant types, which act as triggers for the activation of the behavior. The behavior is quiet until the demons awaken it. In this way, the representations are not limited to be those of the external world only, and more sophisticated forms of intelligent reasoning are permitted.

A number of representations would be required in any typical robot control application, and would include the following: state; actuator commands; mission/task direction; environment; recognized aspects of the environment; sensed data; and so on. Three representations are special:

1. Exception record (ER). Failures during the execution of a behavior must be detected; exception records encode this information. Each exception contains a specification of what must be sensed in order to confirm that the exception occurred.
2. Event windows (EW). The parameters of all event windows are included in the EW representation, that is, the subsets of the representations to be considered are defined.
3. Perception systems internal parameters (PP). The PP representation is a database of parameters used by the perception systems and their values. This includes all thresholds, filter tuning values, any constants, etc. PP is partitioned, one partition for each sensor system. Behaviors which read PP need only consider the partition relevant to their own sensor. These facilitate attention, goal-direction and hierarchical processing strategies and are further described in Ref. [11]. S* is currently being implemented.

4. Current implementation

PLAYBOT is controlled by a network of computers in a client–server architecture. Each robot component (arm, head, cameras, platform) has a hardware connection to one computer on the network, and this computer implements the server for the robot components. The above behaviors are mapped onto client computers, and clients send requests to the servers for robot components to perform actions or return status information. The network currently runs at 10 Mbits, and has a hardware TCP bridge to the Internet. Hardware servers include:

1. *Datacube server.* SUN SPARC II running SunOS 4.1.2 and Imageflow 2.5 as host to a Datacube with two DigiColor and two MV200 boards.
2. *User interface, Geometry server, Planner server, and Arm server.* Silicon Graphics Power Series 4D/380VGX with eight processors running Irix 4.3, with a serial interface to a CRS Plus five degree of freedom robot arm, and serial interfaces to two custom-made infrared proximity sensors in the arm's fingers.
3. *Mobile platform server.* Silicon Graphics Indigo running Irix 5.3, with serial interface to a Cybermotion K2A mobile platform.
4. *Head server.* Intel Pentium running Linux 1.2.8 with a Digital Motion Control DCX ISA interface card and eight MC-110 servo modules controlling a custom-made robot head.

Each of these machines runs a socket-based TCP/IP server daemon for the attached device. TCP/IP client programs connect to servers required for the behaviour the client implements. The User interface starts client connections to all the servers.

In addition to C and C++ application programmer interfaces, we have compiled all client functions into the CLIPS programming language. CLIPS is an expert system tool developed by the Software Technology Branch (STB), NASA/Lyndon B. Johnson Space Center. It is an object-oriented LISP-like interpreted language with inference, pattern matching, and tracing capability. PLAYBOT's Planner server is written in CLIPS, and has client connections of its own to all other servers. The Planner receives commands and goals directly from the user interface, to which it reports its results. It also has the ability to pre-empt any goal or robot action in progress.

The Head server has a client connection to the Datacube server to perform the object tracking behaviour. It instructs the Datacube server to track the target object in both left and right images, and uses the centroids returned by the

Datacube server in real time to adjust the eyes' vergence and tilt degrees of freedom, as well as the neck's pan degree of freedom, to stabilize gaze upon the tracked object. The Datacube produces updates at the rate of 30 Hz. When calibrated, the head motor encoder values can be used to triangulate the positions of tracked objects to within one cubic inch for objects up to 5 feet away. The head can fixate objects moving up to about 3 feet per second at that distance. This is approximately the maximum speed of the mobile platform. We are thus able to fixate an object of interest while moving the platform, avoiding the need for object search after most platform motions.

The Geometry server is used for two main purposes. One is to display a simulated world view of PLAYBOT and its environment to the user from any chosen viewpoint. The other is to maintain positions of modeled objects and robots for planning purposes. The Planner server instructs the Geometry server to update simulated robot and object positions to reflect their real counterparts, and queries the Geometry server for current robot and object position and orientation for planning purposes. We intend to make use of the Geometry server for collision avoidance and simulation of proposed plans.

Finally the Arm server performs object manipulation behaviours (such as pickup and put down) under control of clients with connections to the visual servers.

A typical plan for the task 'pickup object' follows:

1. Track object 1
2. Pickup object 1
    - 2.1. object 1 reachable
        - 2.1.1. verify object 1 visible to both left and right cameras; verify tracking status of object 1 from datacube server
        - 2.1.2. obtain 3-D location of object 1; instruct head server to triangulate object 1
        - 2.1.3. move platform to bring object 1 within arm's reach
            1. raise arm to prevent collision while moving
            2. plan platform motion that avoids table and wall obstacles
            3. fixate object 1 to keep it in view during planned platform motion instruct head server to fixate object 1 (continually move eyes to keep object 1's centroid in center of left and right image)
            4. instruct platform server to roll to desired location
            5. stop fixating object 1
    - 2.2. object 1 and fingers visible
        - 2.2.1. raise eyes to look above object but keep object in view; tilt eyes so bottom of object is 30 pixels from bottom of images and/or at least 200 pixels are visible above the object
        - 2.2.2. obtain 3-D location of object 1; instruct head server to triangulate object 1
        - 2.2.3. obtain gripper position from arm server
        - 2.2.4. move fingers to 1.5 inches above object 1; move incrementally[1] toward target 1.5 inches above top of object
        - 2.2.5. instruct datacube server to find and track fingers at target location
    - 2.3. object 1 within finger grasp
        - 2.3.1. obtain finger sensor status from arm server
        - 2.3.2. move fingers in spiral shaped path, keeping bottom of visible finger closely above top of object 1 in both left and right images, until finger senses object 1
        - 2.3.3. stop tracking object 1 and fingers
    - 2.4. object 1 between fingers
        - 2.4.1. open fingers (to maximum of 2 inches)
        - 2.4.2. position arm such that finger senses free space around object 1
            - 2.4.2.1. rotate gripper to obtain circular occupancy map from finger sensor
            - 2.4.2.2. if all free space, check for small object
            - 2.4.2.3. if no free space, object too large to pick up
            - 2.4.2.4. if free space in only one sector of sensor circle try to find two disjoint free space sectors move arm maximum amount in direction that reduces size of free space sector and repeat from 2.4.2.1.
            - 2.4.2.5. if free space in two disjoint sectors of sensor circle rotate gripper to place fingers in free space, lower gripper until object 1 sensed between fingers
    - 2.5. close fingers
    - 2.6. lift object 1

Using this plan, the images in Fig. 13 show the sequence of actual actions PLAYBOT executes. It is interesting to note the wide variation in images PLAYBOT sees during the execution of this plan; these are shown, for both left and right camera views, in Fig. 14. Each of the image pairs corresponds to the images PLAYBOT sees at the stages of the plan captured as snapshots in Fig. 13.

5. The future

The complete example shown should suffice to convince the reader that the basic foundation for the system is real and successfully performs some simple tasks. However, much

[1] Collision avoidance behavior is not yet implemented.

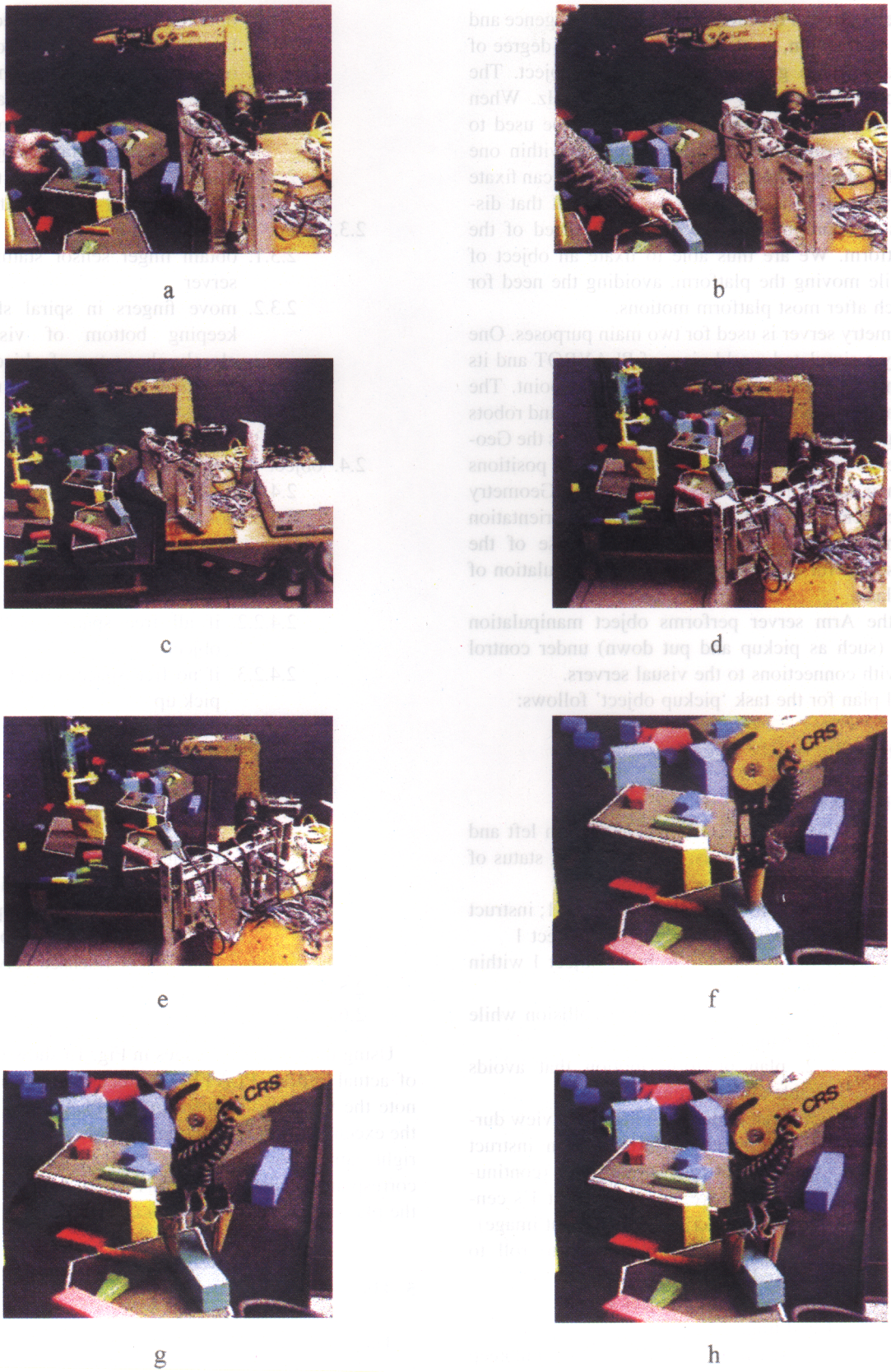
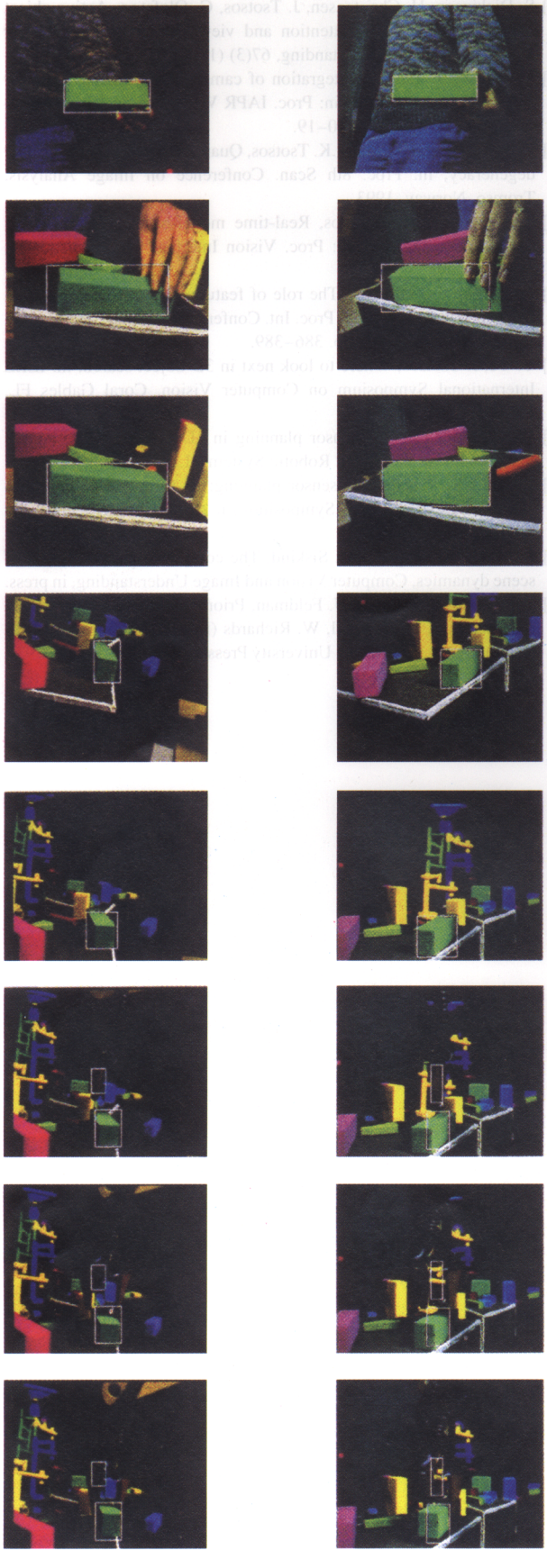


Fig. 13. Eight successive views of executing a 'pickup object' task. The correspondence of images to task steps is: (a) 1; (b) 1; (c) 2.1.1; (d) 2.1.3; (e) 2.2.1; (f) 2.2.4; (g) 2.4.1; (h) 2.6. Note that the object is placed in the field of view by a human, and may be placed anywhere in the camera's current field of view constrained by positions that are actually reachable by the robot, for this implementation.

Fig. 14. The left and right camera images seen by PLAYBOT during execution of the 'pickup object' task. Each corresponds to the similarly labelled view on Figure 13.



remains to be done. Among the major outstanding tasks are: develop the hardware environment to reflect that of Fig. 1 (build the ActiveDeskTop, acquire a computer controlled wheelchair, acquire a robot arm with a longer reach and with an articulated hand); complete the implementation of the S* control framework; enlarge and enhance the network of computers on which the behaviors run; add the complete visual behaviors as described; use the geon representation to represent real toys; expand the database of objects and actions that PLAYBOT understands. Of course, after all this is complete, serious testing with the intended users will reveal how well the initial vision fits their needs.

Acknowledgements

We thank the many students and staff at the University of Toronto and York University who have contributed: Eugene Amdur, Jonathan Appavoo, Rimon Barr, Robert Day, Brian Down, Raymond Ghaly, Matthias Goebel, Raoul Jarvis, Imola Kerekes, Allen Lau, Katy Ly, James Maclean, Bernie Maillard, Andrew Prior, Dale Singh, Dave Suydam. Development of PLAYBOT has been funded by IRIS (Institute for Robotics and Intelligent Systems, a Government of Canada Network of Centers of Excellence), ITRC (the Information and Technology Research Center, one of the Province of Ontario's Centers of Excellence) and NSERC (the Natural Science and Engineering Research Council of Canada). Parts of this development have benefited from the collaboration with Lars Olsson and Goran Olofsson at the Computer Vision and Active Perception Laboratory at the Royal Institute of Technology, Stockholm, Michael Chan at the GRASP Lab at the University of Pennsylvania, and Henrik Christensen at the Laboratory of Image Analysis at the University of Aalborg, Denmark. We thank the staff of the Hugh MacMillan Rehabilitation Center, Toronto, for discussions and comments on the design of PLAYBOT. Tsotsos and Jepson acknowledge the support of the Canadian Institute for Advanced Research.

References

[1] E. Helfman, Blissymbolics: Speaking without Speech. Elsevier/Nelson Books, New York, 1981.

[2] J.K. Tsotsos, Behaviorist intelligence and the scaling problem, Artificial Intelligence 75 (1995) 135–160.

[3] J.K. Tsotsos et al., The PLAYBOT project, in: Proc. IJCAI Workshop on AI Applications for Disabled People, Montreal, 1995.

[4] S. Dickinson, S. Stevenson, E. Amdur, J. Tsotsos, L. Olsson, Integrating task-directed planning with reactive object recognition, in: Proc. SPIE Intelligent Robotics and Computer Vision XII, Boston, 1993, pp. 212–224.

[5] M. Jenkin, J.K. Tsotsos, Large-scale, touch-sensitive video displays, British Informal Patent, 9201949.6, Jan. 30, 1992. US Patent Pending, 011,453, 1993.

[6] G. Verghese, J.K. Tsotsos, Robotic Fingers which determine object grasp pose, University of Toronto Intellectual Property Disclosure, 1996.

[7] E. Milios, M. Jenkin, J. Tsotsos, Design and performance of TRISH, a binocular robot head with torsional eye movements, International Journal of Pattern Recognition and Artificial Intelligence 7 (1) (1993) 51–68.

[8] M. Jenkin, A. Jepson, J. Tsotsos, Techniques for disparity measurement, CVGIP: Image Understanding 53 (1) (1991) 14–30.

[9] M. Jenkin, E. Milios, J. Tsotsos, B. Down, A binocular robotic head system with torsional eye movements, in: IEEE Int. Conf. on Robotics and Automation, Atlanta, 1993, pp. 776–781.

[10] M. Jenkin, J., Tsotsos, Active stereo vision and cyclotorsion, in: Proc. Computer Vision and Pattern Recognition, Seattle, 1994.

[11] Tsotsos, J.K., Intelligent control for perceptually attentive agents: The S* proposal, Robotics and Autonomous Systems, 21(1) p 5–21, 1997.

[12] J.K. Tsotsos, S. Culhane, W. Wai, Y. Lai, N. Davis, F. Nuflo, Modeling visual attention via selective tuning, Artificial Intelligence 78 (1-2) (1995) 507–547.

[13] A. Jepson, M. Black, Mixture models for optical flow computation, in: I. Cox, P. Hansen, B. Julesz (Eds.), Proc. of the DIMACS Workshop on Partitioning Data Sets: With Applications to Psychology, Vision and Target Tracking, AMS, Providence RI, 1995, pp. 271–286.

[14] S. Dickinson, A. Pentland, A. Rosenfeld, From volumes to views: an Approach to 3-D object recognition, Computer Vision, Graphics, and Image Processing: Image Understanding 55 (2) (1992) 130–154.

[15] S. Dickinson, D. Metaxas, Integrating qualitative and quantitative shape recovery, International Journal of Computer Vision 13 (3) (1994) 1–20.

[16] S. Dickinson, H. Christensen, J. Tsotsos, G. Olofsson, Active object recognition integrating attention and viewpoint control, Computer Vision and Image Understanding, 67(3) (1997) 239–260.

[17] D. Wilkes, J. Tsotsos, Integration of camera motion behaviours for activeobject recognition, in: Proc. IAPR Workshop on Visual Behaviors, Seattle, 1994, pp. 10–19.

[18] D. Wilkes, S. Dickinson, J.K. Tsotsos, Quantitative modelling of view degeneracy, in: Proc. 8th Scan. Conference on Image Analysis, Tromso, Norway, 1993.

[19] G. Verghese, J.K. Tsotsos, Real-time model-based tracking using perspective alignment, in: Proc. Vision Interface '94, Banff, 1994, pp. 202–209.

[20] G. Verghese, J. Tsotsos, The role of feature visibility constraints in perspective alignment, in: Proc. Int. Conference on Image Processing, Washington DC, 1995, pp. 386–389.

[21] Y. Ye, J. Tsotsos, Where to look next in 3D object search, in: IEEE International Symposium on Computer Vision, Coral Gables FL, 1995, pp. 539–544.

[22] Y. Ye., J.K. Tsotsos, Sensor planning in 3D object search, in: Int. Symposium on Intelligent Robotic Systems, Lisbon, 1996.

[23] Y. Ye, J.K. Tsotsos, 3D sensor planning: its formulation and complexity, in: International Symposium on Artificial Intelligence and Mathematics, 1996.

[24] R. Mann, A. Jepson, J.M. Siskind, The computational perception of scene dynamics, Computer Vision and Image Understanding, in press.

[25] W. Richards, A. Jepson, J. Feldman, Priors, preferences and categorical percepts, in: D. Knill, W. Richards (Eds.), Perception as Bayesian Inference, Cambridge University Press, Cambridge, 1996 pp. 93–122.