Situated Vision in a Dynamic World: Chasing Objects

Ian Douglas Horswill and Rodney Allen Brooks

MIT Artificial Intelligence Lab
545 Technology Square
Cambridge, MA 02139

Abstract

We describe a system that approaches and follows arbitrary moving objects in real time using vision as its only sense. The system uses multiple simple vision computations which, although individually unreliable, complement each other in a manner mediated by a situated control network. The objects can move over a wide variety of backgrounds including those with strong secondary reflections from light sources. Previously unseen objects can be tracked against backgrounds that include other moving objects. Computations are carried out in image coordinates at roughly 5 frames per second on a Lisp machine. The camera need not be calibrated or aligned well, and the system can tolerate a wide range of dynamically changing actuator response characteristics.

1 Introduction

In this paper we describe a robot that follows arbitrary moving objects in real time using vision as its only sense. The system is constructed from simple components using simple hardware and runs at approximately five frames per second. It has been tested by the authors and several others with a variety of objects and lighting conditions. Testing was performed in normal environments inhabited by people. Neither the environments nor the behavior of the inhabitants were altered for the experiments.

Approach-and-follow is an example of a simple navigation behavior. Having identified an object as being in some sense interesting, the agent moves so as to keep the object centered in its visual field. The agent effectively computes and follows a smoothed approximation to object's path. Alternatively, the object may be viewed as "dragging" the agent as it moves. Approach-and-follow is simple enough to be implemented very efficiently, yet can operate in a chaotic and unpredictable world. The short path from sensors to effectors allows the system to be highly responsive in dynamic situations.

The system is inspired by the way a kitten plays with a ball. It is largely passive at first, but reacts to an object's motion by following at a respectful distance. People experimenting with the robot during demos sometimes "play" with the robot as if it were a young animal, sometimes lead it around, and sometimes herd it by causing it to back away from the object. Thus approach-and-follow allows complex and interesting dynamic behavior in spite of its its simplicity of implementation.

2 Implementation

To implement an agent which exhibits approach-and-follow behavior, it is necessary to divide the scene into objects (segmentation). The agent must also choose a specific object to follow (triggering). Having done this, the agent must maintain the identity of the object from frame to frame (tracking). Finally, the motors must be driven in such a way as to follow the object in question (motor control).

In our system, each of these tasks is implemented by one or more asynchronous process. The system is implemented on a lisp machine with a standard frame-grabber and the MIT AI Lab Allen robot. Processes are simulated as lisp functions which are called in a round-robin fashion. Wires connecting them are simulated with global variables. Visual input is obtained by subsampling the grabbed image to 32×28. The camera is uncalibrated[1], but must face roughly forward. The lisp machine sends motor commands to the robot via an RS-232 serial line.

Figure 1 shows the processes and connectivity of the system. Fat lines represent visual pathways between processes, while thin lines represent simpler control signals. Visual information enters on the left side, while motor commands exit the right side and feed back through the environment.

2.1 Segmentation

The system must divide the visual world into visual objects. We make the distinction between visual objects and physical objects because they need not lie in one-to-one correspondence. It is sufficient that the visual object being tracked correspond to a single physical object over time[2]. In our system, we take visual objects to be connected regions of the image. Segmentation is the process of dividing the image into such regions.

There exists a large literature on segmentation (see [BB82] for a survey). For our purposes a very simple technique is sufficient. We first apply a local predicate to all points in the image and then label connected components of ones in the resulting binary image using a blob coloring algorithm[BB82, page 151].

Many possible predicates could be used. For example, we could use a simple threshold on the grey level. This is only useful for following objects of known homogeneous in-

[1]The original camera mounting consisted of a wedge of newspaper and some duct-tape.

[2]In this sense, it is not even necessary to divide the world into objects, only "the-object-I-am-tracking" and "everything-else".

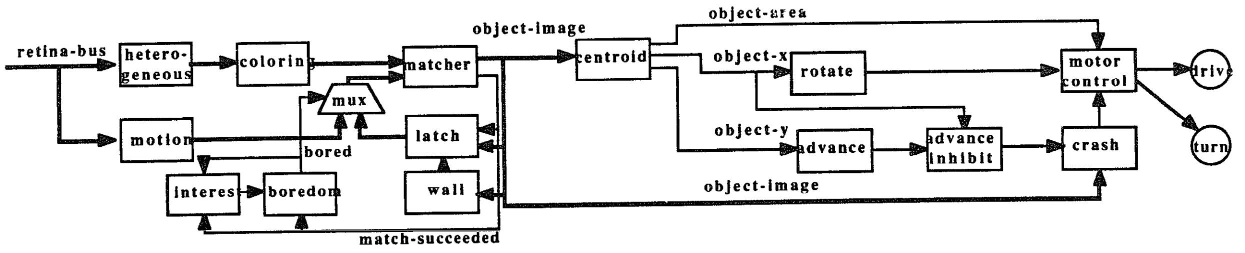

Figure 1: Structure of the approach-and-follow system

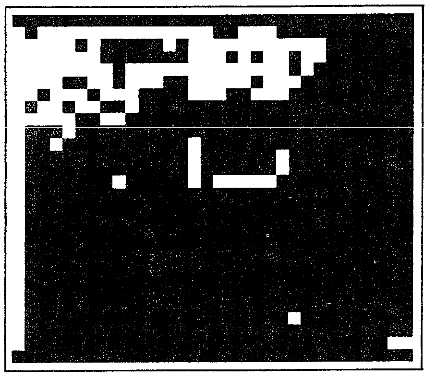

Figure 2: Leaking of homogeneous regions

tensity however. An alternative would be local homogeneity; regions of homogeneous intensity would be grouped into connected regions. Local homogeneity is relatively stable with respect to camera noise in the sense that most pixels do not change. The connectivity of the binary image formed by applying the local homogeneity test to each pixel is highly unstable however. This is due to that fact that images of objects consist of large homogeneous regions separated by relatively thin edges. When camera noise perturbs the values at edge points, changes in connectivity can result, perhaps causing the target object to be considered a part of the floor. Figure 2 shows the binary image of a white sheet of paper on an untextured floor with clutter in the background. Black pixels correspond to homogeneous regions. The top edge of the paper and one pixel of the bottom edge have been blurred out, causing the interior to be joined to the floor. The problem is solved by using a local heterogeneity constraint, the inverse of the previous constraint. Since connected regions of change tend to be separated by larger distances than connected regions of homogeneity, the topology of the the resulting image is stable.

The performance of the system is relatively insensitive to the actual operator used to detect heterogeneity[3]. Similarly, the system is relatively insensitive to the choice of threshold. The segmentation threshold was set once, and has not needed to be modified since.

The segmentation is implemented by two processes, the **heterogeneous** box continuously computes the binary image of heterogeneous points, and the **coloring** box computes the connected components.

2.2 Tracking

Tracking is implemented using a **matcher** box, and a latch. The matcher continuously outputs the silhouette of the blob in the coloring which has maximal overlap with the other input (usually the matcher output from a previous iteration), subject to the constraint that the input and output blobs cannot vary in area by more than a constant factor. This implements the tracking of a blob. The matcher also asserts a control line whenever it finds a match. The latch is a state-element which holds the silhouette of the object being tracked from a previous match. When a new match is found, the latch updates itself. When no match is made, the latch retains its state. The two processes, matching and latching, track the current object wherever it moves in the visual field. If the object is briefly lost, it can be found again provided that it has not moved too far. The latched silhouette of the truck from figure 3 is shown in figure 3.

Tracking is bootstrapped by inserting a multiplexer between the output of the latch and the input of the matcher. To attend to a new object, the latch is bypassed and the silhouette of the new object is injected into the matcher.

More complicated matching algorithms are imaginable. The current algorithm has the disadvantage that it requires the object's silhouette to overlap from frame to frame, thus constraining the velocity of tracked object. It is however, fast. Since the current system runs at approximately five frames per second, a one-foot object is only constrainted to move slower than five feet per second. This speed is considerably faster than the motors can drive the robot.

Figure 3 shows a snapshot of the states of the **retina-bus**, the output of the **heterogeneous** box, and the latch, during the course of tracking a toy truck.

[3]Our current implementation computes the sum of the absolute values of the differences in the x and y direction for efficiency reasons. It applies this operator not to the image itself, but to its pixel-wise logarithm, so as to normalize for local intensity.

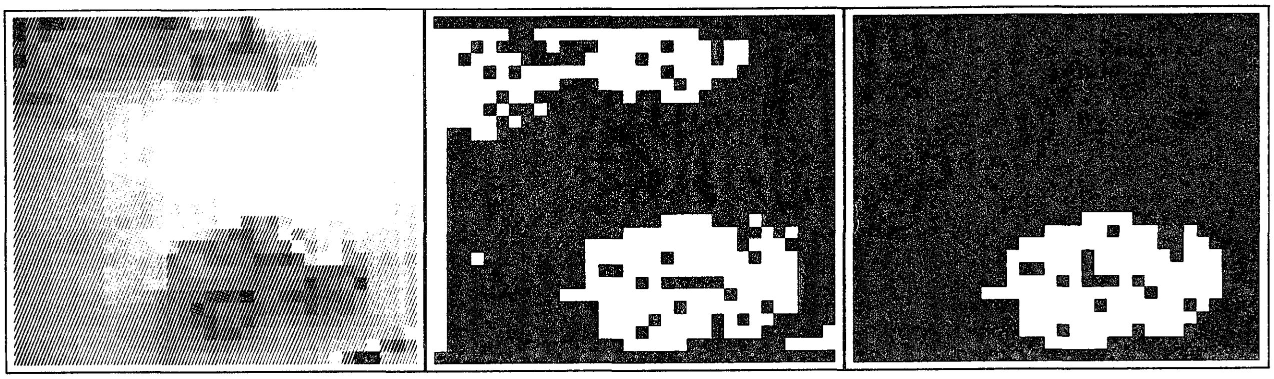

Figure 3: Grey-scale image, heterogeneous pixels, and silhouette of a toy truck

2.3 Motor Control

Having tracked the object from frame to frame, the system can now drive its motors so as to follow the object. To do this the system must have some understanding of the relative position of the object. This need not mean a precise set of coordinates however. The information will be used only to determine whether the robot should turn left or right, move forward or backward, etc. If we constrain the camera to point roughly forward and down at a flat floor, then the robot need only keep the object centered in its visual field[4]. In effect, the object drags the robot as it moves.

Motor control is accomplished by six asynchronous processes. The **centroid** box computes the centroid of the object being tracked and asserts its x and y image coordinates on the **object-x** and **object-y** wires, and its area on the **object-area** wire. The **rotate** box drives the turning motor left or right so as to reduce the difference between the value on the **object-x** wire and center of the screen. The **advance** box drives the drive motor so as to reduce the difference between the **object-y** wire and an arbitrary screen line (scan line 18 is presently used within a 28 line image). Two extra boxes were added after the fact to insure that the **advance** box did not get into trouble. The **advance-hold** box inhibits the output of the **advance** box until the **rotate** box has had a chance to move the object from the left or right edge of the screen. This prevents the combined advance and rotate motions from running the object off the edge of the screen. The **advance-hold** box and the **rotate** box do not communicate directly. Instead, the **advance-hold** box samples the **object-x** wire. In effect, they may be said to communicate *through the world.* The **crash** box samples the output of the matcher and inhibits the **advance** box when the object being tracked runs off of the bottom of the screen. This saves the robot when the object it is tracking overlaps a much larger object such as a baseboard, and the segmenter joins them. When the robot tries to center the baseboard in its field of view, the **advance** box would sometimes drive the robot into the wall. Finally, the **motor-control** box actually drives the motors. It also stops the robot when the area of the object is zero (i.e. when no object is being tracked).

[4]It is also necessary to assume here that the object is resting on the floor.

2.4 Triggering

We have described all of the machinery necessary to follow a visual object, but we also need machinery to control that machinery. That is, we need machinery to initiate following behavior when an "interesting" object is found, and to terminate following when it is lost.

Rather than have a separate mechanism which detects interesting objects, we reuse the existing components as follows. We add a set of processes to detect "interesting" pixels, and use the matcher already in the tracker to find the interesting object. Using this system, it is not even necessary to check whether there is an interesting object to be tracked. The system has two states: *tracking* and *bored.* When the system is *tracking*, it has an object and the multiplexer feeds the matcher with the output of the latch. When the system is *bored*, the multiplexer feeds the matcher with the set of "interesting" pixels. If there are no interesting pixels, then no match will occur and the system will stay bored. If there are interesting pixels, they will be matched to the interesting object and the system will switch to *tracking* state. Boredom is determined by another set of processes monitoring the success of the matcher. Triggering is thus decomposed into *candidate detection* and *attention control.*

Candidate detection is accomplished by looking for pixels whose grey levels change significantly over time. These pixels are assumed to correspond to moving objects. Changing pixels are detected by comparing two versions of the camera input, one of which is fed through a delay line. The comparison is made using a thresholded difference operator by the **motion** box.

Attention is controlled by a **boredom** box which forces the tracker into *bored* mode by switching the multiplexor whenever the matcher has failed for two consecutive frames. This allows it to tolerate occasional matching failures, but prevents it from locking on to camera noise. However, if someone walks in front of the object being tracked, the system will get bored and retarget before the object becomes unoccluded. For this reason, an **attention** box was added later. The **attention** box waits until it has seen seven successive matches and then raises the **boredom** box's threshold to four frames, or slightly less than a second. The **attention** box is reset when then boredom wire is asserted, thus resetting the threshold of the

boredom box.

The system has one extra component, the **wall** box, which further filters the candidates by suppressing any object which runs off the top of the screen. The suppression is accomplished by clearing the tracker's latch. The **wall** box has the effect of preventing the system from trying to follow objects on walls (see below).

3 Experimentation

The system was tested in two areas of the MIT AI Lab which are regularly inhabited by people—a small laboratory crowded with chairs and equipment, and a larger lobby area. The areas were not modified, other than to clear paths for the robot. The small laboratory is carpeted, while the lobby has a shiny Linoleum floor which reflects images of the overhead lights. It is necessary that the floor not have significant texture at the same scale as the objects, although the images of the lights on the Linoleum floor have not proven to be a problem. The system was tested under conditions ranging from early morning when the areas were deserted to mid-afternoon when members of other research groups were rushing about preparing for a demonstration. Workpeople and researchers unfamiliar with the system would regularly walk in front of the robot or have conversations in front of it during the course of the experiments. The system has been tested with and without sunlight, and with varying amounts of overhead lighting.

The system has tracked a black videotape box, a pink plastic flamingo, a dark grey notebook, and a white plastic food container as they were dragged along by a bright white string. Later, a radio-controlled toy truck was used.

The system was able to follow all of the objects. It was relatively insensitive to the relative contrast of the object and the background. For example, the blue toy truck is easily tracked on the blue Linoleum floor. The grey notebook is also easily found on the carpet which is a close shade of grey. The major constraint is simply that the object have significant visual angle so that it may be found in the low resolution image.

Several things can confuse the system however. The biggest problem is the overlap of objects. The present segmentation algorithm joins adjacent or overlapping objects into a single object, which is usually rejected by the matcher as being too large. Thus the tracked object is lost when it moves too close to furniture or the baseboards of walls to be distinguished at low resolution. Sometimes the joined objects do get through the matcher however, and this can led to disastrous results. Since the introduction of the **crash** box however, there have been no such mishaps. A particularly annoying example of the problem is that the string dragging the objects is usually bright enough to be seen against the relatively dark background, in spite of its thinness. Sometimes it is seen as a part of the object and the system will track along the string to the person dragging it. Sometimes it will just pan back and forth along the string. The string is thin enough however that it is lost eventually, and the system retargets.

The system as yet has no notion of collision avoidance. This is usually not a problem because it approximates the path of the object being followed, a path assumed to be clear. There are three cases where this has lead to trouble. The first is when it follows an object which does not rest on the floor, such as a wall poster. This is partially alieviated by requiring that no object run off of the top of the screen (the **wall** box). The second case is when the object "pushes" the robot backward into a wall. The final case is when the robot tries to cut a corner through a doorway. The farther the robot is from the object being tracked, the more it smooths the path of the object. When the object is far away and takes a tight corner, the robot can smooth its path into the door frame. Thus while collision avoidance is simplified by the robot's pattern of interaction with the world, a backup system is called for.

In another set of experiments, static objects were followed either by forcibly gating their silhouettes into the matcher[5] or by waving a hand in front of them to attract the system's attention. Books and magazines with different amounts of texture on their covers could be approached this way.

When several objects were placed along a path and successively gated into the matcher, the robot could be made to follow a static path. At one point, one of the authors (IDH) gated the the silhouettes of the legs of successive people talking in the lobby. The robot drove up to each and waited quietly. Thus while the current triggering components only facilitate patterns of interaction such as leading, herding, and play, other triggering mechanisms could implement other patterns of interaction. Such triggering mechanisms and the patterns of interaction to which they lead are presently being explored.

4 Related Work

There are a number of projects which use situated control networks. These include our own subsumption networks [Bro86] which consist of graphs of finite state machines augmented with timers, the REX based networks of simulated synchronous gates of [RK88] which have also been interfaced with a real time vision system [III87], and the Pengi nets of simulated synchronous gates of [AC87], which have relied on simulated vision processors.

The most relevant of these is the work of Wells [III87]. He, though, concentrated on the more traditional role assumed of vision; to produce a three dimensional model of the world. His work does share similarities with ours however; he bootstraps from one simple process to another, and uses the results from one time step to focus the attention of his vision algorithms in the next step.

Agre and Chapman [AC87] treat vision as a set of subroutines that can be invoked by a "central system" (funnily enough the only state in their system is buried in which subroutine calls to vision are currently active). An abstraction barrier between the central and vision systems provides a clean interface.

Wallace, et. al. have also developed a navigation system (for road following) which uses only computations in image coordinates [WST*85].

[5]This feature was added for purposes of debugging. The output of the segmenter is updated on the screen as it is computed. Objects can then be gated into the matcher by pointing with the mouse.

A case can be made that the approach we have taken to linking sensing to action is reminiscent of the cybernetic approaches of the fifties and sixties. This is true to the extent that our system opts for short paths from sensors to effectors and uses feedback through the world to quickly verify its decisions. These are desirable traits, particularly for a system operating in an unpredictable environment, and a behavior such as approach-and-follow is simple enough to be easily implemented in this manner.

The major difference is that our system exhibits a modularity which aids its understanding and evolution. Cybernetics, and often connectionism, tend to treat the brain as an amorphous whole with little structure. The components of the system described in this paper can be modified individually without hurting the structure as a whole. New triggering components can be added, the matching algorithm can be modified, etc. In addition, we consider the system we have described here as just one component of a larger system controlling a mobile robot. Other modules with similar importance in a complete system might implement behaviors such as collision avoidance, exploration of a static world, etc. We have in fact extended the system described here to follow corridors. This work, and the approach-and-follow system, are described in detail in [Hor88].

5 Conclusions

We have presented a robot that follows moving objects in a variety of environments using vision as its only sense. The system is built from relatively simple and unreliable components. Some unusual aspects of its structure bear mention.

The system performs segmentation and motion analysis, tasks which are quite difficult to do well. We have not solved either problem. Our system solves only limited forms of the problems, and still does a poor job of each. Nonetheless, the system performs its task *as a whole* quite well. Two lessons can be learned from this. First, while the segmenter does a poor job of determining the set of pixels that corresponds to a given object, or even of finding all of the pixels in a given set of connected heterogeneous pixels, it does a perfectly adequate job of finding stable blobs attached to isolated objects, which is all it *needs* to do. That is to say, it is adequate for the task. The same is true of the motion detector. It does a lousy job of finding all of the moving pixels, but it finds enough for the matcher to uniquely determine the moving object. Second, while the tracker may lose an object occasionally, the motion detector will usually find it again. The motion detector sometimes hallucinates motion, but it is usually in homogeneous areas. Since there are no objects in homogeneous areas, the matcher simply rejects the motion. Thus individually unreliable components can combine synergistically to form a more reliable whole.

It is also interesting that the system performs no planning or collision avoidance. While an account of these capabilities would be necessary for a full theory of navigation, our system avoids much of the problem through the structure of its interactions with the world[6]. In effect, it lets its target perform these tasks.

Finally, we have argued [Bro86] that it is possible to build systems that appear intelligent to observers yet have no central representations or loci of control. We have also argued [Bro87] that things such as central representations of the world are purely in the mind of the observer. A system situated and acting within the world does not need them. Ours is an example of such a system. Instead of representing the visual world in a single, central database, our system distributes the representation in a principled manner.

[6]As was mentioned above, some sort of extra collision mechanism is called for, perhaps a set of whiskers.

Acknowledgements

Support for this research is provided in part by the University Research Initiative under Office of Naval Research contract N00014-86-K-0685, and in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014-86-K-0124.

References

[AC87] Philip E. Agre and David Chapman. Pengi: an implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272, 1987.

[BB82] Dana H. Ballard and Christopher M. Brown. *Computer Vision.* Prentice Hall, 1982.

[Bro86] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automoation*, 2(1):14–23, March 1986.

[Bro87] Rodney Brooks. Intelligence without representation. In *Proceedings of the Workshop on the Foundations of Artificial Intelligence*, MIT, June 1987.

[Hor88] Ian D. Horswill. *Reactive Navigation for Mobile Robots.* Master's thesis, Massachusetts Institute of Technology, March 1988. A revised version to appear as a technical report of the MIT Artificial Intelligence Laboratory.

[III87] William M. Wells III. Visual estimation of 3-d line segments from motion - a mobile robot vision system. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 772–776, AAAI, July 1987.

[RK88] Stanley J. Rosenschein and Leslie Pack Kaelbling. The synthesis of digital machines with provable epistemic properties. In Joseph Halpern, editor, *Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 83–98, Morgan Kauffman, 1988.

[WST*85] R. Wallace, A. Stenz, C. Thorpe, H. Moravec, W. Whittaker, and T. Kanade. First results in robot road-following. In *IJCAI-85*, 1985.