Incremental Focus of Attention for Robust Visual Tracking

Kentaro Toyama and Gregory D. Hager
Department of Computer Science, Yale University, New Haven, CT 06520-8285

Abstract

We present the Incremental Focus of Attention *(IFA) architecture for adding robustness to software-based, real-time, motion trackers. The framework provides a structure which, when given the entire camera image to search, efficiently focuses the attention of the system into a narrow set of possible states that includes the target state. IFA offers a means for automatic tracking initialization and reinitialization when environmental conditions momentarily deteriorate and cause the system to lose track of its target. Systems based on the framework degrade gracefully as various assumptions about the environment are violated. In particular, multiple tracking algorithms are layered so that the failure of a single algorithm causes another algorithm of less precision to take over, thereby allowing the system to return approximate feature state information.*

1 Introduction

Robustness is a hallmark of intelligent behavior and a desirable property of any system. Yet, the abundance of black velvet in vision laboratories testifies to the brittleness of current visual tracking technology. Although there have been attempts to make trackers more robust with respect to certain disturbances such as changes in the background [8, 9] or foreground occlusion [5], little has been done toward developing an effective, generalizable computational framework for dealing with these problems.

In an effort to fill this void, we present the Incremental Focus of Attention (IFA) architecture. IFA shares abstract design concepts with a variety of existing architectures and algorithms. We agree with designers of the subsumption architecture of robotics [1] that artificial constraints on a system's environment provide a crutch on which we may never stop leaning. From vision research, IFA uses ideas similar to *pop-out* and *foveated vision*. Pop-out is an effect where certain locally identifiable cues, such as motion or color, rapidly focus visual attention [6]. Single components

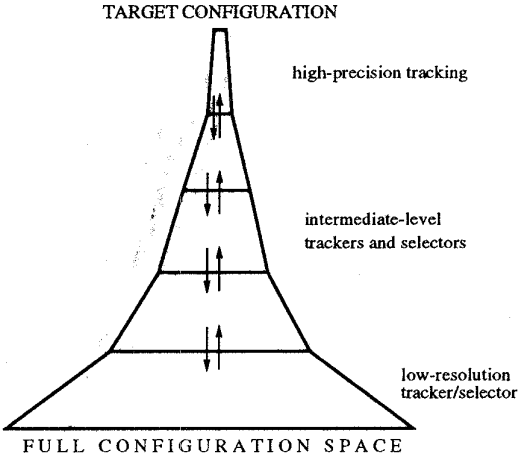


Figure 1. The IFA framework.

of the IFA framework are often based on pop-out algorithms. In foveated vision, the resolution of an image is inversely proportional to its distance from the center, or fovea [7, 9]. As with foveation, our framework takes advantage of speed gained by processing images at varying resolution. If we view tracking as a large search problem, IFA can even claim precursors in classic artificial intelligence applications such as STRIPS and Hearsay which speed up search by examining preconditions. A layer of an IFA tracker prunes the search space for the next layer by checking if regions marked by certain visual cues warrant further attention.

The IFA architecture is closest in conception to *focus of attention* algorithms [2, 10, 11]. These algorithms use layered processing to implement both parallel and inherently serial procedures for narrowing attention. The intention is to create fast, robust vision systems. There are, however, a few conceptual differences between our approach and that of most focus of attention research. First, we do not restrict ourselves to only positional information about features in an image. Instead, the search ranges over configuration spaces that may include rotation, scaling, shear, or even full 3D pose information. Second, instead of a network model with neural components arranged in layers, our layers are heterogeneous input-output

devices. Third, during each tracking cycle, IFA processing occurs only in a single layer of the framework rather than in all layers. Finally, and what is perhaps the root cause of the previous differences, much focus of attention work has been concerned with computational models of biological visual systems, whereas we are primarily concerned with performing efficient, robust tracking on common computing hardware. As a result we have been able to implement complete systems which are able to track moving objects robustly in real time with real-world scenes.

2 Motivation and Theory

We define the goal in visual motion tracking to be to return the *state* of a target object[1] from the data contained in a stream of images. The images themselves may be monchrome or color, monocular or stereo. Possible components of state include the target's position, orientation, shape, velocity, and other geometric or photometric qualities. Ideally, a motion tracker returns complete and accurate state information, but ideal conditions rarely hold in the real world. Even our well-developed human visual systems are unable to track objects with precision under adverse conditions, such as when the objects are moving too quickly. In these circumstances, human beings are nevertheless able to do two things: gain partial state information about an object that cannot be precisely tracked and recover an object that is temporarily lost from view. Thus, athletes can still catch, hit, or kick a ball even as it rapidly revolves.

Robustness is a difficult concept to formalize, but we assume, for tracking, that it consists of several components. First, a robust system should not unknowingly track anything other than the target object. Secondly, there should be a means for reinitialization when tracking fails. And third, there should be substitute tracking mechanisms while reinitialization occurs so that some information can be output.

These constraints suggest a layered fallback scheme (see Figure 1). After a tracking cycle, control moves up or down the framework, depending upon the success of tracking during the previous cycle. The topmost layer performs high-precision tracking. Under ideal conditions, processing should take place solely in this layer. As environmental conditions deteriorate, however, control is relinquished to lower layers which perform less accurate, but more robust, tracking. Lower layers continuously attempt to pass control to the higher layers so that better tracking returns with better conditions. The crucial questions involve the transitions between layers: how layers detect their own failure and give up tracking to a lower layer, and how tracking becomes more precise as favorable conditions return.

[1] We use the term "object" to generically denote anything from simple features such as edge segments to complex three-dimensional objects.

2.1 Trackers and Selectors

First, we define $\mathcal{X}$ as the space of all possible configurations assumable by a target object. Configurations may be actual physical states of the object (such as 3D pose) or image-based states of the projection of the object onto the camera image. We also define the *visible configuration set*, $\mathcal{X}_v$, to be the set of all configurations which would cause part or all of the object to project onto the camera image.

Layers in the IFA framework are either *selectors* or *trackers*. Both narrow an *input configuration set*, $\mathbf{x}^{in} \subseteq \mathcal{X}$, to a smaller *output configuration set*, $\mathbf{x}^{out} \subset \mathcal{X}$, such that $\mathbf{x}^{out} \subset \mathbf{x}^{in}$. Both trackers and selectors can be characterized by their *output accuracy* which depends on the size of their output sets. Less accurate trackers or selectors return larger output sets. As described in the next section, layers of the IFA framework are arranged in order of increasing output accuracy, with the least accurate layer at the bottom taking the entire visible configuration set as its input and the most accurate layer at the top returning precise object state information.

What distinguishes trackers from selectors is that trackers must be designed so that if the output set of a tracker is non-empty, then it contains the actual configuration of the target object. That is, trackers do not produce false positives. We will call this the *filter criterion*.

Selectors, on the other hand, may return subsets of their input set which do not contain the target configuration. Instead, the primary role of selectors is to partition the output of a tracker below it into sets with size and shape appropriate as input to the tracker immediately above. For example, the initial input set for any IFA system is the entire configuration space. Most tracking algorithms, however, expect only a portion of the configuration space as input. A selector, therefore, is necessary to partition the large initial set into smaller subsets that can be used as input to the tracking algorithm.

Selectors also focus the attention of the system by favoring certain output sets over others. In particular, heuristics may be used to determine which part of a selector's input set is likely to contain the target

state. Still, because selectors do not guarantee that their output sets will contain a target configuration even when their input sets do, selectors must necessarily fulfill another requirement: given the same input set, a selector must return a different output set each time it is called. The union of output sets over time equals the input set (this is similar to the reasoning behind inhibition of input areas in focus of attention work [2]). This restriction ensures that the system as a whole will not fall into a loop where a selector insistently produces a set that does not include the target state. Lastly, a selector's input set may become exhausted, in which case it must be *reset* so that other lower-level algorithms can take over.

Examples of trackers and selectors are given in §3.

2.2 Framework Construction

Given several tracking algorithms which do not conform to the standards prescribed in §2.1, we must *adapt* them before we can construct an IFA system. Some tracking algorithms are more easily adapted into selectors and others are better converted into IFA trackers. The decision is based partly on whether or not the tracker satisfies the filter criterion and partly on what assumptions can be made about the visual environment.

Tracking algorithms which almost always satisfy the filter criterion under a minimal set of assumptions about the environment should become trackers in the IFA framework. A few modifications are necessary: first, the trackers must be altered to detect mistracking, *i.e.*, they must return the empty set if tracking fails. There are usually geometric constraints or thresholds which can be set so that a tracker will report failure when appropriate. For example, a tracker based on an object model might report failure when a certain percentage of its model components don't find correspondences in the image. For the robustness of the system as a whole, it is better for trackers to be *conservative* in their own estimation of success: a tracker which sometimes signals failure even during successful tracking is preferred over one which occasionally mistracks without knowing.

Tracking algorithms which are likely to miss a target are converted into selectors. For example, an algorithm which tracks an object by searching the whole image for its predominant intensity value may find itself tracking the wrong parts of the image when ambient lighting changes. Were this tracking algorithm incorporated as a low-level tracker in an IFA system, altered lighting conditions would cause the entire system to enter a loop where high-level trackers vainly seek the target in input sets that do not include it. However, as a selector, the same tracking algorithm becomes a heuristic to efficiently narrow the search space when lighting conditions are as expected. In order to ensure reasonable behavior, the system must also be able to disregard the heuristic. For this reason, a candidate selector algorithm is adapted to the framework by incorporating it into a generic selector procedure which initially attends to the areas suggested by the candidate. Over time, however, the selector switches to exhaustively returning its input set independent of the nested tracker. In the case mentioned above, when ambient illumination changes, the selector initially tries its heuristic, but as higher layer tracking algorithms continue to fail, the selector gives up on its heuristic and switches to an exhaustive search.

After all tracking algorithms have been appropriately adapted, they are sorted by output accuracy. Additional selectors are inserted wherever a layer's output is not appropriate as input to the layer immediately above it. These selectors may be specially designed for the system and the environment, or they may be brute force search algorithms which systematically return different output sets. In the event that the bottommost layer does not take the full visible configuration set as its input, a final global selector is added at the bottom.

Given a collection of n configured and sorted trackers and selectors, they are labeled from 0 to $n-1$ such that the topmost layer is $n-1$, and the bottommost is 0. We then associate a *frustration index*, $frus_i$ and a frustration threshold $frus_i^{thresh}$ to each layer i from 0 to $n-1$. The purpose of these values is to put a limit on how many times a layer may be called before reverting to a lower layer (see §2.3). Monitoring of the frustration index at any layer allows for detection of repeated, unsuccessful attempts to track in an empty region of the configuration space.

2.3 Algorithm

With this framework, we now proceed to describe the algorithm (see Figure 2) used to control computation with the IFA framework. We employ the following nomenclature: i represents the currently active layer, $\mathbf{x}_i^{in}$ and $\mathbf{x}_i^{out}$ denote the input and output sets of layer i, respectively, and $frus_i$ and $frus_i^{thresh}$ represent the frustration index and frustration threshold for layer i as described above.

Once initialized, the algorithm spends its time in the main loop, repeatedly executing the currently active layer and evaluating the results. If a layer mis-

Initialization: Reset all selectors, set all frustration indices to 0, set i to 0 and set $\mathbf{x}_0^{in} := \mathcal{X}_v$, the entire visible configuration set.

Loop: Compute $\mathbf{x}_i^{out}$ and evaluate the following conditional:

1. If layer i is a tracker and $\mathbf{x}_i^{out} = \emptyset$ or $frus_i > frus_i^{thresh}$ (signaling mistracking), set $frus_i$ to 0 and decrement i if it is greater than 0.
2. Else, if layer i is a selector and $frus_i > frus_i^{thresh}$, then reset $frus_i$ to 0, reset the selector, and decrement i if it is greater than 0.
3. Else, if layer i is a selector or if layer i is a tracker with $i < n-1$, increment $frus_i$, set $\mathbf{x}_{i+1}^{in} := \mathbf{x}_i^{out}$, and increment i.
4. Else, $i = n-1$ and tracking must be successful. Reset all selectors, set all frustration indices to 0, and set $\mathbf{x}_i^{in} := \mathbf{x}_i^{out}$.

Figure 2. The IFA algorithm.

tracks or is frustrated (steps 1 and 2), it is reset and the system moves down a layer. Step 3 moves the system up a layer either because tracking at that level was successful and there are higher layers, or because a selector is not yet frustrated. Step 4 represents the case of successful top-layer tracking. In this case, all selectors and frustration indices are reset. The net result is that the algorithm moves the tracking system down layers when tracking is uncertain, thereby broadening the search region, and ascends layers as trackers succeed, narrowing the set of configurations until the top layer is reached.

3 Implementation

We demonstrate the IFA framework with several implemented systems which robustly track various objects. These systems were implemented on a Sun Sparc 20 equipped with a standard framegrabber and a single chip color camera.

Since there is significant overlap in the set of trackers and selectors used by the experiments, we briefly describe them separately:

Random Selector: This selector takes any set of configurations as its input and randomly returns a set that is appropriate for the tracker above it. Repeated application of this selector returns different output sets such that in the limit of infinite applications, the entire visible configuration space is covered. It can be a "last resort" mechanism for any heuristic-based selector and is the basis for heuristic selectors.

Heuristic Selectors: Given any input set, output sets can be chosen on the basis of detected motion, predicted state, color or intensity matching, or any combination of these and similar heuristics. These selectors return output sets which are subsets of all configurations that are consistent with what is detected by the camera. For example, a color and motion-based selector would initially return output sets whose configurations correspond to image projections which match a given color and which exhibit motion. These selectors often operate on subsampled, low-resolution images to gain speed.

Homogeneous Region Tracker: If the target object has pixel values (color or intensity) which are predominantly within a certain range, then the homogeneous region tracker will produce a set of configurations which narrow the target object's position. Tracking is performed by finding the first and second moments of pixels classified as being part of the region.

Edge-of-Polygon Tracker: This tracker tracks a single edge of a polygon. It expects a set of target configurations which would project the target object onto a small region of the image, but for which no orientation information is known. By tracking a single edge, the output configuration set is culled to include only those configurations of the polygon which have orientations consistent with the tracked edge.

Rectangle Tracker: This tracker tracks approximately rectangular objects by tracking four corners, which are in turn tracked as two intersecting line segments (see [4]). The tracker will narrow a "compact" set of input configurations, into an smaller set of configurations which corresponds to the image projection of a rectangle together with some allowance for noise. Mistracking is detected when the line segments of adjacent corners are not collinear.

SSD-based Pattern Tracker: The SSD-based pattern tracker matches a stored template to a region of the current image and computes a residue value. It returns an output set whose elements are configurations that correspond to affine transformations of the template which match with the current camera image. It fails if the computed residue, the sum-of-squared-differences (SSD), is above a given threshold (see [4]).

SSD-based Multi-Pattern Tracker: The same tracker as above, but which tracks successfully if the current image contains a match to any of several stored images.

3.1 Robust Disk Tracking

In the past, we have often used 3.5 inch diskettes as a convenient target for various experiments with visual servoing [3]. These experiments, while successful in demonstrating hand-eye coordination, were nevertheless extremely sensitive to disturbances. Distractions in the background, whole or partial occlusion, sudden motion of the camera, and various other "unexpected" perturbations of the visual environment often resulted in undetected mistracking, which led to incorrect and even dangerous motions by the robot arm as the visual servoing algorithm computed robot velocities based on unreliable tracking information.

To make this system more robust to visual disturbances, we have developed a robust disk tracker which consists of the following components (in order of increasing output accuracy): a color, motion, intensity, and position-based combination selector (layer 0), a homogeneous region tracker (layer 1), an edge-of-polygon tracker (layer 2), and a rectangle tracker (layer 3). The selector at the bottom rapidly focuses the attention of the system to areas which match the color of the disk, which exhibit motion, and which are positionally near the predicted location of the disk. The color and intensity heuristic is valid because ambient lighting conditions normally don't change significantly (although if they did, the selector would not insist on particular colors indefinitely). The use of motion and positional prediction are justified by the fact that if the system reverts to this layer, it is most likely due to overly fast motion of the disk relative to the camera. The homogeneous region tracker helps to quickly discard regions of the input set in which the disk is unlikely to be by conservatively signaling a mistrack if it is unable to grow beyond a certain diameter. In other words, it checks if the output configuration set of the selector projects onto a reasonably-sized patch of the camera image. Finally, the edge-of-polygon tracker finds an edge of the disk, and the rectangle tracker tracks the four corners.

The robustness of the system is evident through its behavior under various conditions. Tracking the disk proceeds perfectly as long as the disk moves within a certain speed, there are no distractions, and the disk remains wholly in view. When one of the corners is occluded or if there are significant distractions in the background, the top-layer tracker signals a mistrack, and tracking oscillates between homogeneous region tracking and rectangle reinitialization. Homogeneous region tracking remains stable if only the corners are occluded, but the rectangle tracker mistracks as long as the occlusion remains. During this time, even though the tracker is unable to recover corner coordinates of the rectangle, it still maintains a rough approximation of the position of the disk because of the region tracker. If the occlusion is temporary, tracking resumes as the rectangle tracker recovers tracking. If the occlusion persists and the region tracker becomes frustrated, if occlusion becomes complete, or if sudden fast motion occurs and the region tracker signals a mistrack, then the system must start again at the bottom-layer selector. At this point, if the disk is completely occluded, tracking fails. If there is another disk of the same color in the field of view, tracking may continue on the imposter, unless the user has explicitly incorporated a way to distinguish between disks. If the disk reappears, IFA will reinitialize the disk at its new location, and the system will continue tracking.

3.2 Robust Surface Tracking

Approximately planar surfaces of various kinds can be tracked robustly using an IFA system similar to the disk tracker but substituting a sequence of SSD trackers where the disk tracker uses homogeneous region tracking and rectangle tracking. For concreteness, we describe the actual technique we use to robustly track a face, although the same system can be used with minimal alteration to track almost any roughly planar surface.

We again list the trackers and selectors, in order of increasing output accuracy: a color, motion, and position-based combination selector (layer 0), an SSD tracker with high residue (layer 1), and an SSD tracker with low residue (layer 2). The selector narrows the visual field to those areas which exhibit motion and are of the target person's skin tone. The lower SSD tracker with the high residue threshold tries to make a rough match to its template given a region to search. If during its tracking phase, it is unable to reduce the residue below its threshold, it signals a mistrack and reverts to the selector. If the residue after a tracking phase is below its threshold, then tracking proceeds to the top-layer tracker which is the same SSD tracker with a lower threshold. The lower threshold is chosen so that the tracker will be successful only when the examined region of the image matches, up to ambient image noise and minor illumination changes.

Repeated experiments show that this three-layered

(layer 2) (layer 0)

Figure 3. Face tracking at different layers. On the right, tracking has temporarily reverted to layer 0 tracking, since the stored image does not match the rotated head.

IFA tracker quickly recovers track of a target face after perturbations like those described for disk tracking (§3.1) and continues tracking accurately under the assumptions made when configuring the bottom-layer tracker.

Figure 3 shows face tracking at different layers.

3.3 Robust Object Tracking

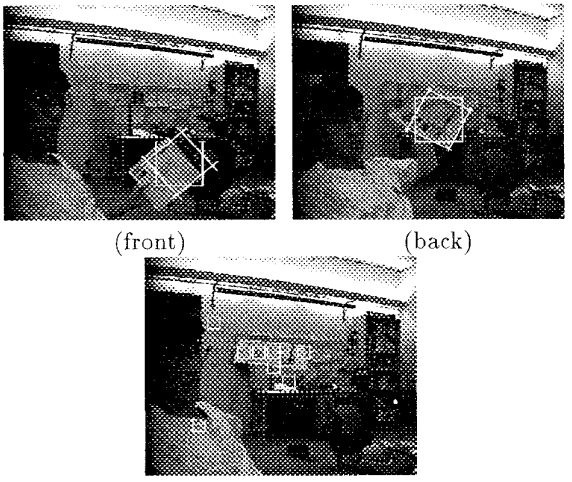
Finally, we describe a generic object tracker which can be seen as a precursor to a robust pose-estimating 3-D object tracker. The generic object tracker is based on the idea of tracking an object by matching to multiple views.

The implementation is identical to that for robust surface tracking, except that the SSD trackers are replaced by SSD-based multi-pattern trackers, and the color-based heuristic for the bottom layer selector seeks to find color matches based on the union of predominantly occurring colors in all stored figures.

In Figure 4, we see how two images were used to track an envelope. The faces of the envelope were initialized as the images to track. In the two top figures, tracking proceeded at layer 0, since one of the envelope's faces was completely visible. However, during the time the envelope was being flipped over, no match was made to either image; thus, tracking temporarily reverts to a lower layer where color-based selection takes place.

3.4 Analysis

Figure 5 shows where the tracking systems spends it time during some example runs for disk tracking (similar results were obtained for SSD tracking). The horizontal axis represents time and the vertical axis represents the IFA layer executed by the system. Time instants marked by an '**a**' indicate where temporary partial occlusions or slight taps to the camera occurred. In these instances, top-layer tracking was disturbed,



(front) (back)

(no match)

Figure 4. Tracking an envelope as two different sides.

causing the system to revert to an intermediate tracker momentarily, but as the occlusion ceased or as the camera resettled, the system quickly climbed back up to the top-layer tracker. At times marked by '**b**,' the camera was jerked away so that tracking was completely lost. While the camera was readjusted with the target object in a different location in the field of view, the system oscillated back and forth among the low layers as it tried to find the target object among various candidates. Eventually, the selector chose an output set containing the target object and tracking reverted back up to the high layer of the framework. Finally, at times marked '**c**,' temporary full occlusion occurred. In all such instances, tracking dropped to the bottommost selector since intermediate trackers could not find their target. However, unlike the case where the tracking was lost without occlusion, the bouncing between low layers is caused by the fact that the system sought for a target object that was not visible. As the occlusion ended, precise tracking resumed.

We note that with more clutter, recovery from a significant jolt or occlusion takes longer. Figure 6 shows the backgrounds used for the different levels of clutter. Because our top level selectors use heuristics similar to pop-out for color and motion, the more cluttered the background, the greater the time required for tracking to recover. The maximum recovery time after over 100 trials was 15.3 seconds in a heavily cluttered environment, during which time both foreground and background objects were in motion.

One of the advantages of the IFA framework is the relative ease with which existing trackers can be in-

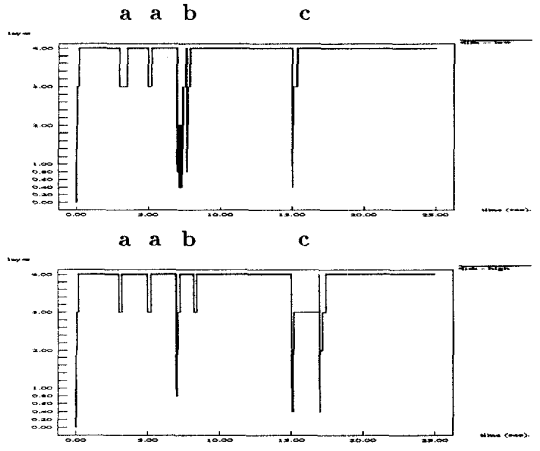


Figure 5. Disk tracking layers and recovery time. Top, low clutter; bottom, high clutter.

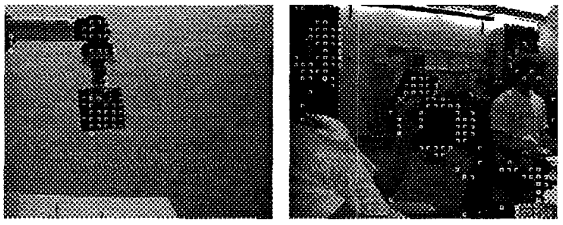


Figure 6. Experimental setup for low and high clutter.

corporated into it. The bottom layers suit almost all tracking applications. Then, given an existing tracking algorithm, it is usually a simple matter to decide when it mistracks based on geometric and visual constraints, especially because there is room to err on the side of being overly conservative. Putting them together is a matter of nesting tracking loops with entrance and exit conditions that depend on the layers. In this way, we have been able to easily adapt the framework to provide robustness for trackers of the X Vision system [4], which offers a set of modular tools for feature-based motion tracking.

Perhaps the best argument for IFA systems is the subjective experience of watching them in action. Where previously, entering the camera field of view was taboo during experiments which use tracking, now, it is almost a pleasure to deliberately cause mistracking, just to watch the system recover its target. Tracking applications can be left running even as people walk in front of the camera, lights are turned off, or target equipment is moved, because soon after favorable environmental conditions return, tracking will have resumed. In a word, tracking is tenacious.

4 Conclusion

By explicitly constructing trackers which recognize their own failure and layering fallback mechanisms, an IFA tracking system performs robustly and degrades gracefully with respect to environmental assumptions. We do not claim to solve the tracking problem under complete occlusion or when lights are off, but IFA provides a framework such that whenever reasonable conditions return, so, too, will tracking.

Further work with IFA proceeds along several avenues — improving selectors and trackers, creating dynamic IFA systems which swap layers in and out depending upon environmental conditions, and formalizing notions of "robustness" and "graceful degradation" as applied to visual tracking.

Acknowledgments

This research was supported by ARPA grant N00014-93-1-1235, Army DURIP grant DAAH04-95-1-0058, by National Science Foundation grant IRI-9420982, and by funds provided by Yale University.

References

[1] R. A. Brooks. A robust layered control system for a mobile robot. *J. of Rob. Autom.*, 1(1):24–30, 1986.

[2] S. M. Culhane and J. K. Tsotsos. An attentional prototype for early vision. In *ECCV '92*, Italy, 1992.

[3] G. D. Hager. Calibration-free visual control using projective invariance. In *Proc. ICCV*, 1995.

[4] G. D. Hager. The "X-vision" system: A general purpose substrate for real-time vision-based robotics. In *Proc. Wksp. Vis. Rob.*, 1995.

[5] A. Kosaka and G. Nakazawa. Vision-based motion tracking of rigid objects using prediction of uncertainties. In *Proc. IEEE Conf. Rob. Autom.*, Japan, 1995.

[6] H. Tagare and D. McDermott. Model-based edge selection for 2-D object recognition. DCS RR-1044, Yale University, August 1994.

[7] D. Terzopoulos and T. F. Rabie. Animat vision: Active vision in artificial animals. In *Int'l Conf. on Comp. Vision*, 1995.

[8] K. Toyama and G. D. Hager. Keeping your eye on the ball: Tracking occluding contours of unfamiliar objects without distraction. In *Proc. 1995 Int'l Conf. on Intel. Rob. and Sys.*, Pittsburgh, 1995.

[9] K. Toyama and G. D. Hager. Tracker fusion for robustness in visual feature tracking. In *SPIE Int'l Sym. Intel. Sys. and Adv. Manufacturing*, vol. 2589, Philadelphia, 1995.

[10] J. K. Tsotsos. An inhibitory beam for attentional selection. In *Spatial Vision for Humans and Robots*. Cambridge Univ. Press, 1993.

[11] J. K. Tsotsos. Towards a computational model of visual attention. In T. Papathomas, C. Chubb, A. Gorea, and E. Kowler, editors, *Early Vision and Beyond*. MIT Press, 1995.