International Journal of Computer Vision 35(1), 45–63 (1999)

Incremental Focus of Attention for Robust Vision-Based Tracking

KENTARO TOYAMA
Microsoft Research, One Microsoft Way, Redmond, WA 98052-6399
kentoy@microsoft.com

GREGORY D. HAGER
Department of Computer Science, The Johns Hopkins University, Baltimore, Maryland 21218-2694
hager@cs.jhu.edu

Abstract. We present the *Incremental Focus of Attention* (IFA) architecture for robust, adaptive, real-time motion tracking. IFA systems combine several visual search and vision-based tracking algorithms into a layered hierarchy. The architecture controls the transitions between layers and executes algorithms appropriate to the visual environment at hand: When conditions are good, tracking is accurate and precise; as conditions deteriorate, more robust, yet less accurate algorithms take over; when tracking is lost altogether, layers cooperate to perform a rapid search for the target and continue tracking.

Implemented IFA systems are extremely robust to most common types of temporary visual disturbances. They resist minor visual perturbances and recover quickly after full occlusions, illumination changes, major distractions, and target disappearances. Analysis of the algorithm's recovery times are supported by simulation results and experiments on real data. In particular, examples show that recovery times after lost tracking depend primarily on the number of objects visually similar to the target in the field of view.

Keywords: visual tracking, real-time tracking, robust tracking, face tracking

1. Introduction

Biological visual systems exhibit an amazing robustness to complex visual events. The human visual system, for example, is able to adapt to or recover from many unexpected visual circumstances. On one hand, it can acquire partial information about an object if it is at all visible; on the other hand, it can reacquire an object that is temporarily lost from view. Thus, athletes can still catch, hit, or kick a ball by knowing its approximate position even when it is spinning rapidly, and motorists can quickly recover track of the vehicle ahead of them, even after a glance in the mirrors.

As we move vision systems from the structured laboratory to the "real world," we must endow them with this ability to cope with unexpected or unmodeled situations. Robotic hand-eye systems must be able to track grasped objects even if they are dropped, vision-based human-computer interfaces should not be bothered by a sneezing subject, and automated driving vehicles must remain aware of the road even if it becomes momentarily obscured by snow or dirt. In biological systems, this kind of robustness is taken for granted, yet in computer vision it is an issue that has received relatively little attention.

Incremental Focus of Attention (henceforth, IFA) begins to fill this gap by providing a design methodology for developing robust motion tracking systems. Conceptually, IFA is a framework for organizing multiple tracking algorithms and search heuristics into robust systems. During execution, IFA efficiently focuses the "attention" of the tracking system onto relevant parts of the image. The result is robustness in two senses. First, if an IFA system is composed of several trackers

of varying accuracy, failure of any specific tracking algorithm usually means that another, less precise algorithm takes over. Second, when visual perturbations temporarily cause a tracking system to lose its target, IFA provides the possibility of reinitialization and recovery. In this way, IFA tracking systems gracefully degrade as visual conditions deteriorate and recover when conditions improve.

This article describes the development of the IFA architecture for efficient and robust vision-based tracking. In the next section, we formulate the tracking problem in more precise terms and develop necessary terminology. In Section 3, we give an overview of related work. Section 4 describes the construction of IFA systems. Their correctness and robustness properties are discussed in Section 5. Finally, Section 6 discusses the implementation of several IFA tracking systems and offers experimental results which justify IFA's claims to robustness.

2. The Robust Tracking Problem

In *vision-based object tracking*, the goal is to determine specific measurable attributes of a *target* given a sequence of temporally ordered images. Tracking problems can be as simple as determining the vertical position in image coordinates of a bouncing ball or as complex as computing the instantaneous 3D kinematic configuration of a walking person. Throughout this article, we assume that the value of these quantities at a given time point t is given by a d-dimensional *state* vector, $\mathbf{x}(t) \in \mathcal{X} \subset \Re^d$. We further assume that $\mathcal{X}$, the *state space* of the system, is bounded and therefore of finite volume. Then, the *vision-based tracking problem* is defined to be the calculation of a series of estimates $\hat{\mathbf{x}}(t)$ of the actual state, $\mathbf{x}^*(t)$, of the target at time t. For the remainder of the article, we assume the dependence of $\mathbf{x}^*$ and $\hat{\mathbf{x}}$ on t without explicitly noting it.

We view vision-based tracking as a repeated search and estimation process, where the search occurs *in the state space of the target, not in the image*. The input to a tracking algorithm is an *input configuration set*, or a set of candidate target states, $\mathbf{X}^{\text{in}} \subseteq \mathcal{X}$, together with an image, I. The output at each time step consists of an *output configuration set*, $\mathbf{X}^{\text{out}} \subset \mathcal{X}$, such that $\mathbf{X}^{\text{out}} \subset \mathbf{X}^{\text{in}}$, where $\mathbf{X}^{\text{out}}$ should include $\hat{\mathbf{x}}$. The output set, $\mathbf{X}^{\text{out}}$, of an algorithm is also a representation of the algorithm's *margin of error*. Thus, we have chosen the margin of error to be a set of states rather than, for example, a statistical distribution over states. Note that for actual implementation, neither input sets nor output sets are necessarily explicit. An algorithm may return a single state, for example, but such output together with the known margin of error can be interpreted as a set of states.

By introducing an explicit margin of error, we can precisely define several relevant terms. We say that tracking is *accurate* if and only if $\mathbf{x}^* \in \mathbf{X}^{\text{out}}$. *Mistracking*, or tracking *failure* occurs when $\mathbf{x}^*$ is not an element of $\mathbf{X}^{\text{out}}$. *Precision* is related to a measure of the size of the error margin, denoted $|\mathbf{X}^{\text{out}}|$. The simplest formulation of precision is that it relates inversely with the volume of states occupied by the error margin. Under this formulation, algorithms which return large margins of error are less precise than algorithms with smaller margins of error. In addition, the dimensionality of the state information computed by an algorithm affects its precision. For example, a tracking algorithm which can determine the position *and* orientation of a face is more precise than an algorithm which determines only the position: the former has a smaller output set than the latter, which includes sets ranging over all possible orientations.

As noted above, we concentrate on the problem of *robustness*. Robustness is the ability of a vision-based tracking system to track accurately and precisely during or after visual circumstances that are less than ideal.

3. Previous Work in Robust Tracking

The existing literature on robust tracking can be broadly categorized into research that contributes to either *ante-failure* or *post-failure* robustness (Toyama and Hager, 1998). Ante-failure robust systems seek to avoid tracking failure altogether through specialized algorithms that anticipate visual disturbances and attempt to track despite them. Post-failure systems, on the other hand, accept the inevitability of mistracking and are designed to recover from failure once it happens. It seems clear that both ante-failure and post-failure means are necessary for reliable tracking, but research thus far has been heavily weighted towards the former.

Advances in ante-failure robustness are usually achieved through robust statistics, temporal filtering, or *ad hoc* methods for handling specific types of visual perturbations.

For instance, distractions—objects which are close to the target both in appearance and state—can be handled in several ways. One method is to consider only a small set of states surrounding the target (Hager and

Toyama, 1998; Vincze, 1996). This requires some predictability in the target trajectory but eliminates the need to examine the entire image. Another way to handle distraction is through foveation, effectively blurring the image region around the target (Burt and van der Wal, 1990; Terzopoulos and Rabie, 1995). Finally, a tracking system may filter output estimates based on expected motion and noise (Blake et al., 1993). Some of this work is based on a well-established literature in target tracking which handles similar problems by taking advantage of known dynamics and noise models (Bar-Shalom and Fortmann; 1988; Bar-Shalom and Li, 1993; Reid, 1979). All of these techniques avoid distraction by ignoring or filtering out competing object states which are unlikely to be the target.

Other visual disturbances have other solutions. Changes in ambient lighting have been handled by concentrating on color cues (Rasmussen et al., 1996; Wren et al., 1995), by tracking based on edges (Gennery, 1992; Kass et al., 1987; Lowe, 1992), or by explicitly modeling illumination parameters (Hager and Belhumeur, 1998). Fast or unpredictable motion requires combinations of faster hardware, full-frame processing (Burt, 1988; Nishihara, 1996), or probabilistic dynamics (Isard and Blake, 1998). Occlusions, where opaque objects intercept the camera's line of sight to the target, can be handled by robust matching (Gennery, 1992; Hager and Belhumeur, 1998; Lowe, 1992; Kosaka and Nakazawa, 1995) or by dynamic state prediction (Blake et al., 1993; Burridge et al., 1995; Terzopoulos and Rabie, 1995).

Ante-failure work generally seeks to handle problems one at a time, but some researchers have sought to design systems that are ante-failure robust to many types of visual perturbations simultaneously. Notable among these are probabilistic methods, which sample and interpolate likelihoods over the state space (Isard and Blake, 1996), and sensor fusion techniques, which track based on multiple cues (Crowley and Berard, 1997; Kahn et al., 1996; Oliver et al., 1997; Prokopowicz et al., 1994).

None of these efforts are error-free. As in other domains, ante-failure methods cannot eliminate failure entirely. Most often, the limitations are a reflection not of poor algorithmic design, but rather of the impossibility of perfect vision-based tracking in complex circumstances.

To deal with such situations, some researchers have incorporated post-failure schemes to recover tracking when mistracking occurs. Typically, post-failure techniques execute different algorithms depending on tracking success. Much of this work is inspired by *focus of attention* in biological systems. Cognitive science research in focus of attention suggests that biological vision systems are broadly organized into *pre-attentive* and *post-attentive* stages: The pre-attentive stage rapidly finds image subregions of interest on which to focus the attention of a post-attentive stage, which examines the attended region more closely (Neisser, 1967; Treisman, 1985; Tsotsos, 1993; Wolfe, 1995).

Many resesarchers have incorporated knowledge gained from cognitive science and built tracking systems that manage two separate algorithms, where the first algorithm rapidly finds relevant candidate regions in an image and the second performs tracking (e.g., (Isard and Blake, 1998)). Most of this work focuses on a single means to find and track a target, using various cues: intensity (Pahlavan and Eklundh, 1992), color (Rasmussen et al., 1996), or motion (Huber and Kortenkamp, 1995; Murray and Basu, 1994). These methods are efficient, but do not take advantage of the multiple cues which identify real-world objects.

Some recent work extends this notion to using more than two stages or multiple cues for tracking. In face tracking, for example, some systems are able to robustly determine such things as the facial expression of the target in real-time (Crowley and Berard, 1997; Oliver et al., 1997). Another system is able to detect and track a limited set of vehicles using a three-stage focus of attention scheme (Concepcion and Wechsler, 1996). Others, focusing on the information contained in multiple cues focus on fusing different types of image information to identify a unique target (Maki et al., 1996; Uhlin et al., 1995).

IFA differs from prior work in several ways. First, rather than trying to build computational models of biological visual systems (Culhane and Tsotsos, 1992; Tsotsos, 1995), we are primarily concerned with synthesizing methods for vision-based tracking which are computationally efficient and robust. Second, IFA is not restricted to computing positional or directional information about a target, as are many active tracking systems which emphasize the control aspects of keeping an object in view (Bradshaw et al., 1994; Coombs and Brown, 1993; Nordlund and Uhlin, 1996; Pahlavan and Eklundh, 1992; Uhlin et al., 1995). Rather, IFA search may recover even affine 2D or fully articulated 3D pose information. Third, we pose tracking itself as a flexible multi-stage focus of attention rather than

remaining within the standard two-stage paradigm (Maki et al., 1996; Nordlund and Uhlin, 1996) or even a fixed k-stage architecture (Concepcion and Wechsler, 1996; Crowley and Berard, 1997; Oliver et al., 1997). Lastly, our framework is not restricted to particular targets or specific cue types. Techniques such as eye-blink detection (Crowley and Berard, 1997) or mouth localization (Oliver et al., 1997) are specifically designed for frontal views of faces only and do not generalize for other target types. IFA, in contrast, allows utilization of a variety of algorithms and thus confers robustness to many different target types.

Finally, we note that recent work in control of robotic systems lays a possible theoretical foundation for IFA. "Dynamical pick and place" tasks, in contrast to static pick and place tasks, require a robot to control the dynamics of an object in addition to controlling its position (Burridge et al., 1995). An example is the problem of using a flat, gripper-less, 3-DOF arm to catch a ball thrown into the robot's workspace, where "catching" means bringing the ball to eventual standstill on the bat. By *sequential composition* of locally convergent controllers, one for each subregion in the state space of the ball, the ball can be caught from almost any initial state (Burridge et al., 1995, 1998): When the ball is falling at high speeds, the controller switches to bouncing control to slow the ball down. At lower speeds, the controller may switch to "palming" behavior, where the bat is able to maintain contact with the ball continuously without bouncing. Ultimately, the ball is caught and held still. A kind of sequential composition occurs in IFA, where different tracking algorithms take control depending on the dynamics of the target.

There are two reasons, however, why sequential composition is not adequate for the tracking problem. First, in tracking, the state of the tracked object is also the quantity to be determined, whereas in robot control, sensing is taken for granted and the result of computation is a command to take action. A circularity arises here—the very state that needs to be computed is what determines the method of computation. Secondly, as explained above, tracking can fail. The state of the target can at times be unknown, requiring a system that recognizes such situations and takes steps to recover from them.

4. System Construction

The design of an IFA system begins with an inventory of the tracking algorithms that are available for tracking a particular target. Algorithms are broadly classified as attention-focusing mechanisms (*selectors*) or as tracking mechanisms (*trackers*). The former can be thought of as heuristics which select regions of the state space to search while the latter track the target once it is found. These algorithms, called *layers* in the framework, are ordered by decreasing precision, with more precise algorithms at the top. At any given moment, processing occurs in a single layer. Successful tracking at a layer pushes control upward while unsuccessful tracking pulls control downward. Because one layer's output set is used as the next layer's input set, ascending layers gradually increase the precision of the system's state estimate. Finally, the system operates in either of two modes, indicating its awareness of tracking success. During `search` mode, the system has no definite state information about the target but actively searches the state space to find it. During `track` mode, the system asserts that it is tracking the target and can estimate part or all of the desired state information.

4.1. Layer Elements

The raw materials for IFA layers are tracking algorithms and visual search heuristics. In general, the more algorithms are available for finding and tracking a target, the better the final system, so it is advantageous to explore and exploit every visual uniqueness of the target.

Given a set of visual searching and tracking algorithms, they must first be classified as either potential trackers or potential selectors. This distinction idealizes the algorithms, simplifying design and analysis. Section 5 will discuss the practical implications when these assumptions are not satisfied.

Trackers are algorithms which always generate an output set that contains the target state if given an input set which contains the target. If a tracking algorithm often tracks non-target objects, it may be more suitable as a selector. Conversely, if an attentional heuristic is based on a cue that is known to be unique within its input sets, it may be better suited as a tracker. An attention algorithm focusing on flesh-colored "blobs" could be a successful tracking algorithm if guaranteed that its input will only contain one flesh-colored object. The sensitivity of the surrounding task to mistracking and accuracy will also come into consideration.

In the following, let $\bar{\mathcal{X}}$ denote the set of all state subsets of $\mathcal{X}$ and let $\mathcal{I}$ denote the set of all images. We model trackers and selectors as functions from state

sets and images to state sets. For a given function f, $\mathrm{Dom}(f) \subseteq \bar{\mathcal{X}}$ and $\mathrm{Rng}(f) \subseteq \bar{\mathcal{X}}$ denote the state set part of the domain of input state sets and range of output state sets of f, respectively. In all cases, we assume that $\mathrm{Dom}(f)$ covers $\mathcal{X}$.

4.1.1. Trackers. Formally, a tracker should fit the following definition as closely as possible (in the sections to follow, we discuss the effect of non-ideal trackers):

Definition 1. An *idealized tracker* is a function, $f : \bar{\mathcal{X}} \times \mathcal{I} \mapsto \bar{\mathcal{X}}$, such that for all $\mathbf{X} \in \mathrm{Dom}(f)$ and $I \in \mathcal{I}$,

1. $f(\mathbf{X}, I) \subset \mathbf{X}$.
2. If $\mathbf{x}^* \in \mathbf{X}$, then either $\mathbf{x}^* \in f(\mathbf{X}, I)$ or $f(\mathbf{X}, I) = \emptyset$.
3. If $\mathbf{x}^* \notin \mathbf{X}$, then $f(\mathbf{X}, I) = \emptyset$.

Together, Properties 2 and 3, which we refer to as the *filter criterion*, are a formalization of partial correctness. They state that trackers may return occasional false negatives, where a target which is present goes undetected, but never produces false positives, where a non-target is hallucinated although none exists. Thus, trackers must monitor their performance and report tracking failure. Usually, geometric constraints or thresholds can be set so that a tracker will report failure when appropriate. A tracker based on a wire-frame object model, for example, might report failure when a certain percentage of its model edges lack correspondences in the image. Although precise self-assessment is optimal, for the correctness of the algorithm, it is better for trackers to err on the side of conservativeness in their own estimation of success.

4.1.2. Selectors. Selectors are attention-focusing algorithms which are heuristic in nature and hence prone to returning sets not containing the target state:

Definition 2. An *idealized selector* is a randomized function, $g : \bar{\mathcal{X}} \times \mathcal{I} \mapsto \bar{\mathcal{X}}$, such that for all $\mathbf{X} \in \mathrm{Dom}(f)$ and $I \in \mathcal{I}$,

1. $g(\mathbf{X}, I) \subset \mathbf{X}$.
2. If $\mathbf{x} \in \mathbf{X}$, there is some finite probability $\epsilon_g > 0$ that $\mathbf{x} \in g(\mathbf{X}, I)$.

The purpose of selectors is to output manageable state sets for higher layers with a possible bias toward certain geometric or image-based constraints. For instance, a selector for face tracking will prefer *but not insist on* returning output sets which include states consistent with detected motion or flesh color. Thus, selectors return different output sets over time such that different portions of the state space are passed up the hierarchy with each call.

Finally, associated with each selector, g_i, are an *iteration index*, σ_i, and an iteration threshold, σ_i^{MAX}. The iteration index counts the number of times a selector is executed and elicits selector *failure* when it surpasses the iteration threshold (Section 4.4 describes when indices are incremented or reset). Monitoring of the iteration index prevents repeated, unsuccessful attempts to find the target in a region of the state space not containing the target. Actual σ_i^{MAX} values will depend on the reliability of the attention heuristic.

4.1.3. Set Dilation. One last adjustment is made to trackers. Since targets are moving even as computation in layers takes place, the state sets output by layers must be adjusted in order to accommodate target movement. The union of potential adjusted output sets must be guaranteed to include the target state if the input set included it. In practice, this seemingly strict requirement is satisfied by *set dilation*, in which the input sets to a tracker are simply expanded to include neighboring states as well. For the remainder of this article, $\mathbf{X}^{\mathrm{out}'}$, will be taken to mean the output set after set dilation.

4.2. Layer Composition

In order to construct the system, all layers are sorted by output precision. Because greater precision often requires more processing, this means that faster algorithms will tend to occur at the bottom layers. Algorithms which are superseded by others in both speed and precision should be discarded.

Next, additional selectors are inserted wherever one layer's output is not precise enough to satisfy the input constraints of the layer above it. These selectors may be specially designed for the system and the environment, or they may be brute-force search algorithms which systematically partition a state space into subsets of the appropriate size. If the bottommost layer does not take the full configuration set as its input, a selector is added at the bottom. Any selectors that are more precise than the most precise tracker are discarded or converted into trackers.

Layers are then labeled from 0 to $n-1$ such that the topmost layer is $n-1$, and the bottommost is 0.

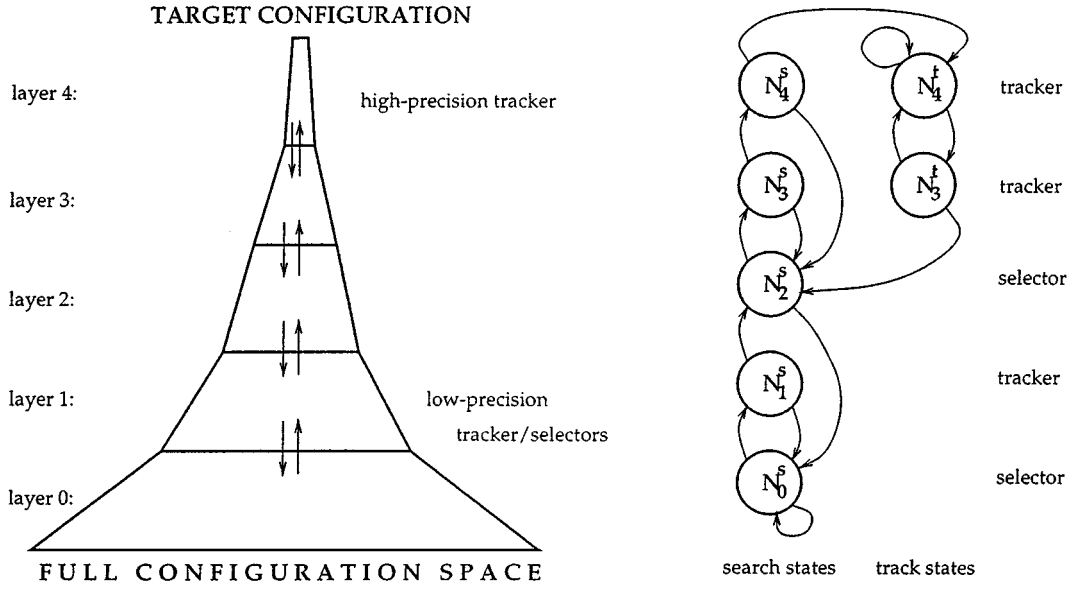

Figure 1. A five-layer Incremental Focus of Attention (IFA) system. On the left, a schematic representation of five algorithmic layers. On the right, the corresponding finite state transition graph for an example with three trackers and two selectors. Subscripts indicate corresponding layer number; superscripts indicate whether the state is in `search` or `track` mode. Outbound edges at the top left of a state are taken when a layer is successful; outbound edges at the bottom right are taken when a layer fails.

4.3. State Transition Graph

At the heart of an IFA system lies a deterministic finite state automaton (Fig. 1(right)), in which transitions occur based on the success of layers. This automaton is constructed as follows:

- For each layer k ($0 \leq k < n$), we create a node N_k^s.
- Let m be the index of any selector that occurs immediately above a tracker. We create nodes N_j^t, for each layer j ($m+1 \leq j < n$). In Fig. 1, $n = 5$, $m = 2$, layers 0 and 2 are selectors, layers 1, 3, and 4 are trackers, `search` nodes are on the left, and `track` nodes are on the right. The superscripts indicate the obvious correspondence of states with the two modes.
- For each k, $0 < k < n$, we create a *success link* (a directed edge) from node N_{k-1}^s to N_k^s and a *failure link* from node N_k^s to node N_l^s, where l is the first layer below layer k that is a selector. One more failure link is created from node N_0^s to itself.
- From N_{n-1}^s, the `search` node for the topmost tracker, we add a success link to N_{n-1}^t.
- For each j, $m < j < n$, we add a failure link from node N_j^t to N_{j-1}^t; and similarly, we add a success link from node N_{j-1}^t to N_j^t. One more success link is created from node N_{n-1}^t to itself.
- Finally, we add a failure link from node N_{m+1}^t to N_m^s.

Since each state represents a unique combination of tracking layer and operating mode (i.e., `search` or `track`), the internal knowledge of the system in terms of tracking mode and precision is represented in its entirety by the current state of the automaton.

4.4. Algorithm

In describing the IFA algorithm (Fig. 2), we use the following variables: N represents the current node in the state transition graph, $i = L(N)$ represents the layer associated with N, l_i represents the algorithm at layer i, and $\mathbf{X}_i^{\text{in}}$ and $\mathbf{X}_i^{\text{out}}$ denote the input and output sets of layer i. The functions $S(N)$ and $F(N)$ return the destination nodes of transition edges out of node N for success and failure, respectively.

Once initialized, the algorithm spends its time in the main loop, repeatedly executing the currently active layer and evaluating the results. Layers cause a straightforward transition based on their success, moving control up a layer when successful and down to the next tracker or selector, if unsuccessful.

Some additional bookkeeping happens for these transitions, both to keep iteration indices updated and to send the proper state sets to the next executing layer.

Initialization: Set all iteration indices, σ_i, to 0, set N to N_0^{s} $(i = 0)$, and set $\mathbf{X}_0^{\mathrm{in}} \leftarrow \mathcal{X}$, the entire state space.

Loop: Compute $\mathbf{X}_i^{\mathrm{out}} = l_i(\mathbf{X}_i^{\mathrm{in}}, I(t))$, and evaluate the following conditional:

1. If layer i is a tracker and $\mathbf{X}_i^{\mathrm{out}} = \emptyset$, perform set dilation on the input set (resulting in $\mathbf{X}_i^{\mathrm{out}'}$), update $N \leftarrow F(N)$, update $i \leftarrow L(N)$, and increment σ_i if Layer i is a selector. Go to Loop.
2. If layer i is a selector and $\sigma_i > \sigma_i^{\mathrm{MAX}}$, then reset σ_i to 0, set $N \leftarrow F(N)$, and set $i \leftarrow L(N)$. Go to Loop.
3. Otherwise, set $N \leftarrow S(N)$, $i \leftarrow L(N)$, and set $\mathbf{X}_i^{\mathrm{in}} \leftarrow \mathbf{X}_{i-1}^{\mathrm{out}'}$. If $N = N_{n-1}^{\mathrm{t}}$, ‘¡set all iteration indices to 0, and set $\mathbf{X}_i^{\mathrm{in}} \leftarrow \mathbf{X}_i^{\mathrm{out}'}$. Go to Loop.

Figure 2. The IFA algorithm.

A selector's iteration index is incremented each time its corresponding node is visited and reset to 0 either when the selector fails or when top layer tracking is achieved. Output state sets at a particular layer are only used as input sets of the next processing layer when the layer in question is successful. If a layer is unsuccessful, then the selector to which control falls uses the same input set as it did the previous time it was called. In Fig. 2, Steps 1 and 2 are concerned with tracker and selector failure, respectively, and Step 3 handles success for both trackers and selectors.

The construction of the layers according to Sections 4.1 through 4.3 together with the algorithm described above assures that the following properties hold (straightforward proofs are given in (Toyama, 1998)):

Property 1. (*correctness*) *The return of* $\mathbf{X}^{\mathrm{out}} \neq \emptyset$ *by Node* N_{n-1}^{t} *implies that* $\mathbf{x}^* \in \mathbf{X}^{\mathrm{out}}$.

Property 2. (*robustness*) *Assume that a system is begun at some node,* N_k^{s} $(0 \leq k < n-1$, *where n is the number of layers*), *and that* $\mathbf{x}^* \in \mathbf{X}^{\mathrm{in}}$. *Then, under ideal visual conditions,* $P(N = N_{n-1}^{\mathrm{t}}) \rightarrow 1$ *as* $t \rightarrow \infty$.

A special case of Property 2 affirms that when the system starts at the bottom layer, high precision tracking will eventually take place.

5. Non-Ideal IFA Systems

In reality, the assumptions made in the previous section do not hold perfectly. Since we are making claims about the robustness of systems, it is vital to our analysis to examine the effects of violated assumptions. We show that even though system performance deteriorates with weakened assumptions, the system as a whole nevertheless retains robustness and partial correctness.

5.1. *IFA in Practice*

5.1.1. Permanent Visual Disturbances. Any visual event that causes a tracker to fail will prevent recovery from any layer below that tracker, so a permanent perturbation will prevent recovery forever. On the other hand, the intent is to recover quickly, so ideal conditions should not need to remain for very long. In Section 5.2, we will consider the issue of expected recovery times.

5.1.2. Imperfect Trackers. Real trackers may return false positives. This can happen even with excellent visual criteria for judging whether an image contains the target object, because it is always possible that there are multiple *distractor* objects visually identical to the target.

Distractors can exist for any single tracker. Thus, a tracker which uses only color to determine its target will be distracted by objects with the same color as the target. Distractors may violate Definition 1(Item 2) because the output set of a tracker may contain the state of a distractor only, even if the input set contains the actual target. Distractors can also violate Definition 1(Item 3) of trackers, since they will cause a tracker to output a

non-empty set, even when only a distractor, and not the actual target state, is in their input state set. Therefore, the correctness and robustness properties in Section 4.4 depend on the fact that at the very least, the top layer tracker does not admit distractors.

Distractors can cause one other form of tracking error, which we call errors due to *sleight of hand*. Sleight of hand occurs when the system is in one of the non-top-layer track nodes (N_i^t, where $i \neq n-1$), and an object that is a distractor for that layer occludes and then moves away from the target object. An example of this occurs when the system is tracking a target's partial state based solely on color, and another object of similar color passes in front of the target. It is possible in this instance that the system will begin tracking the second object instead of the target, without recognizing its error. This situation can be completely avoided by excluding all but the top layer from `track` mode. The cost of choosing this conservative option, however, is to lose the possibility of tracking at lower precisions under perturbed conditions, i.e., to lose the quality of graceful degradation.

5.2. *Recovery Times*

The key issue of IFA systems in practice is a quantitative one: How long is the recovery time after a failure event?

It will be useful to consider a compact representation of an IFA system, in which selectors are always followed by trackers and trackers (except the top layer tracker) are followed by selectors. Formally this is not a problem since the composition of any number of trackers is a tracker and the composition of any number of selectors is a selector. Label the $2m$ new layers g_0, f_0, g_1, f_1, $\ldots$, g_{m-1}, f_{m-1}, where g_0 represents the bottommost selector and f_{m-1} is the topmost tracker.

In the following sections, we will let κ_i represent the number of times that selector g_i will be called before its output set contains the target (and $\mathbf{E}[\kappa_i]$ the expected number of calls). Let T_i be the total time to execute both selector g_i and tracker f_i. We will additionally make some independence assumptions which will make the analyses tractable:

1. State-image pairs are static between visits to the bottom layer.
2. Each time the bottom-layer is visited, the state component of the input is generated statistically independently from previous visits.
3. The operation and associated probabilities of each layer is independent of the operation of layers preceding it.

These assumptions are intended to be consistent with intuitive notions of object motion, particularly for those targets which require post-failure robustness: While most objects exhibit apparent continuity over small time scales, they are more likely to appear to move randomly when sampled over larger time scales.

5.2.1. Ideal IFA. Under ideal conditions, the filter criterion for tracker f_i ensures that g_{i+1}'s input set always includes the target state. Thus, we can set $\sigma_{g_{i+1}}^{\mathrm{MAX}}$ to ∞ and still expect recovery.

The expected value of the sum of random variables is simply the sum of their expected values, so the total expected time to complete recovery from selector layer i, is given by

$$\mathbf{E}\left[T_i^{m-1}\right] = \sum_{j=1}^{m-1} \mathbf{E}[\kappa_j] T_j, \tag{1}$$

where T_i^{m-1} represents the total time to recover from Layer i to Layer $m-1$. In this case, better selector heuristics at any Layer i will decrease $\mathbf{E}[\kappa_i]$ and contribute directly to shorter expected recovery times. We note that the impact of the improvement at any layer is roughly equal to improvement of any other layer, because most branches in the implicit search tree are immediate dead ends (see Fig. 3(a)).

5.2.2. IFA in Practice. The existence of false positives for trackers complicates the recovery time analysis. With false positives, iteration thresholds must be finite (except at the bottom layer), in order to ensure exit from upper layers that are given input sets not containing the target.

To make the analysis concrete, we make the following modeling decisions: (1) there is a fixed probability, p_{f_i}, of a false positive at Layer f_i for each potential branch of the search at Layer g_i; (2) a false positive at one layer does not affect the probability of false positives at higher layers; (3) there are fixed probabilities, $\rho_{ij} < (1 - p_{f_i})$, which represent the probability that a given selector, g_i, receives the target state in its input state, it will return an output set containing the target on the jth iteration.

We now consider the time required to fall out of a layer, should that layer be called with an input set not containing the target. We merely add up the number of

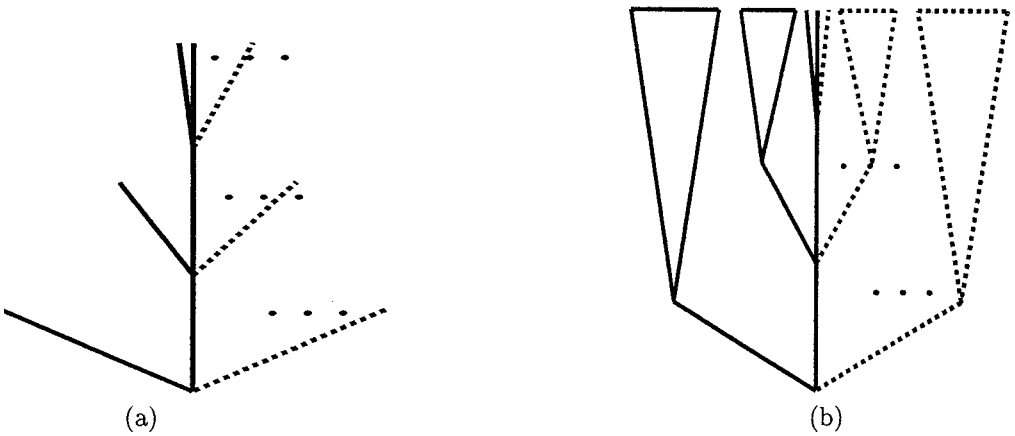

Figure 3. Schematics for the IFA search process. Solid lines represent branches or subtrees which actually were explored until target state localization. Dotted lines represent branches or subtrees which were not explored before state determination. Triangles represent entire subtrees. (a) IFA systems with trackers which fulfill the filter criterion prune all but one subtree at each depth; (b) Systems whose trackers do not fulfill the filter criterion may have to search every subtree to maximum depth.

visits to each layer (multiplied by the respective times it takes to do so), given that all iteration thresholds must be reached before the algorithm reverts to layer $i-1$ below. Below, M_i represents the time it takes to revert to Layer $i-1$, starting at Layer i, when Layer i receives a set without the target:

$$\mathbf{E}[M_i] = \sigma_i^{\mathrm{MAX}} T_i + p_{f_i} \sigma_i^{\mathrm{MAX}} \mathbf{E}[M_{i+1}] \qquad (2)$$

Note that only false positives generated at higher layers cause additional search deeper in the subtree. Setting the base case, $\mathbf{E}[M_{m-1}]$, of this recurrence to T_{m-1}, we arrive at the explicit form,

$$\mathbf{E}[M_i] = \sum_{j=i}^{m-1} \left(\prod_{k=i}^{j-1} p_{f_k} \sigma_k^{\mathrm{MAX}} \right) \sigma_j^{\mathrm{MAX}} T_j. \qquad (3)$$

Next, we compute H_i^{m-1} the time required to find a target from Layer i, if the system does in fact find the target. First, we compute $\mathbf{E}[\kappa_i']$, the expected number of times Layer i itself is called, by normalizing the probabilities of κ_i for values less than or equal to σ_i^{MAX} and then computing their expected values. We then sum the expected time it takes to execute Layer i, the expected time spent exploring false positives, and the expected time spent ascending the remaining layers:

$$\begin{aligned} &\mathbf{E}\left[H_i^{m-1}\right] \\ &\quad = T_i \mathbf{E}[\kappa_i'] + p_{f_i}(\mathbf{E}[\kappa_i'] - 1)\mathbf{E}[M_{i+1}] + \mathbf{E}\left[H_{i+1}^{m-1}\right] \\ &\quad = \sum_{j=i}^{m-1} T_j \mathbf{E}[\kappa_j'] + p_{f_j}(\mathbf{E}[\kappa_j'] - 1)\mathbf{E}[M_{j+1}], \end{aligned} \qquad (4)$$

where M_{j+1} is as in Eq. (3) and $\mathbf{E}[H_{m-1}^{m-1}] = T_{m-1}$. For purposes of recovery after failure, we are interested primarily in the case when $i = 0$.

Letting $\mathbf{E}[\kappa_{\mathrm{all}}]$ be the expected number of times the bottom layer will be called before the algorithm converges to the top layer, we can expect the total recovery time to be

$$\mathbf{E}\left[T_0^{m-1}\right] = \mathbf{E}[\kappa_{\mathrm{all}}]\mathbf{E}[M_0] + \mathbf{E}\left[H_0^{m-1}\right]. \qquad (5)$$

The value of $\mathbf{E}[\kappa_{\mathrm{all}}]$ can be estimated as follows. The probability that selector-tracker pair i will succeed, given it receives the target in its input set is given by the sum of a geometric series:

$$P_i^{\mathrm{hit}} = \sum_{j=0}^{\sigma_i^{\mathrm{MAX}}} \rho_{ij} \prod_{k=0}^{j-1} (1 - \rho_{ik}). \qquad (6)$$

Taking advantage of our independence assumptions, the probability of reaching the top layer from the bottom layer is always

$$P_{0,m-1}^{\mathrm{hit}} = \prod_{i=0}^{m-1} P_i^{\mathrm{hit}}. \qquad (7)$$

The number of times the bottom layer must be visited to reach the top layer is geometrically distributed, with initial probability $P_{0,m-1}^{\mathrm{hit}}$. The expected value of this

distribution is simply $1/P^{\text{hit}}_{0,m-1}$, so $\mathbf{E}[\kappa_{\text{all}}]$ is $1/P^{\text{hit}}_{0,m-1} - 1$.

The complex interaction of several parameters (selector heuristic effectiveness, number of distractors, layer execution times, and iteration indices) makes these expressions difficult to grasp, and in any case, most of the probabilities are unknown a priori. In the next section, we offer a concrete example.

5.2.3. Simulation. Using the assumptions made in Section 5.2.2 and approximate probability models based on our face tracking implementation (presented in Section 6), we analyze expected recovery times.

The face tracker can be compactly represented as four alternating selector and tracker layers. Empirical data show that the bottom two layers take approximately 0.13 s to find and determine whether a blob of color is of sufficient size to be a face; the top two layers take mean time 0.07 s to confirm or reject a candidate state set.

Under good visual conditions, the hit rate and the rate of false positives at the bottom layers are determined entirely by the number of distractors present in the image. Letting d be the number of flesh-colored distractors in the image, we approximate p_{f_0}, the rate of false positives, with $d/(d+1)$. So, for example, with 9 distractors, $p_{f_0} = 0.9$. We eliminate the possibility of false positives at the top layer and set $p_{f_1} = 0.0$. The top layer iteration threshold is set to 1 (for reasons to follow), so that fixes $\mathbf{E}[\kappa'_1]$ for the top selector to 1, as well.

This particular set of values induces the following terms for Eq. (5):

$$\mathbf{E}[M_0] = 0.13 + \frac{d}{d+1}0.07$$
$$\mathbf{E}\left[H_0^1\right] = 0.2$$
$$\mathbf{E}[\kappa_{\text{all}}] = \frac{1}{1/(d+1)} - 1 = d.$$

Thus,

$$\mathbf{E}[T_{\text{face}}] = 0.13d + 0.07\frac{d^2}{d+1} + 0.2.$$

$\mathbf{E}[T_{\text{face}}]$ versus d becomes asymptotically linear as d grows larger, as seen in Fig. 10 (the dashed line).

Figure 4 shows the effect of the top layer's iteration threshold on expected recovery time. The graph shows two different cases. The solid line indicates the case when there is one distractor and the dotted line indicates the case for two distractors. For these particular cases, clearly a smaller iteration threshold is better (and thus, for all of the experiments on real data, we used iteration indices of 1, except for the bottom layer).

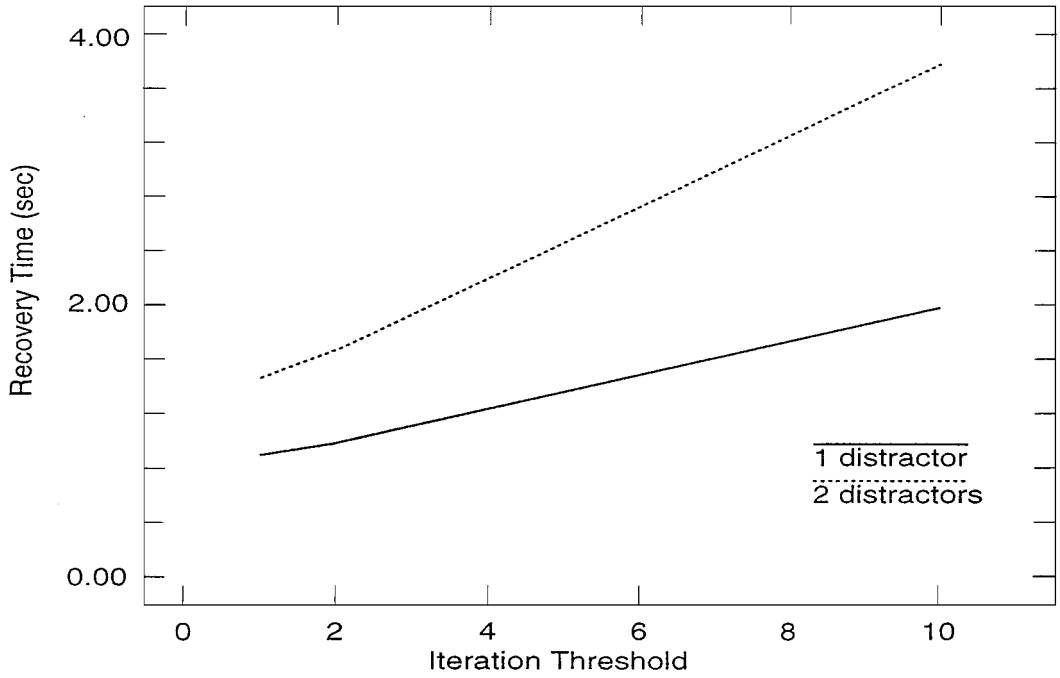

Figure 4. Recovery time as a function of the bottom-layer iteration threshold: with one distractor (solid), and two distractors (dotted).

We caution that analysis with different parameters show that the trends noted here do not necessarily hold in general. Some systems with 3 layers, for example, display best performance at iteration thresholds of 2 or 3. We have found that small iteration thresholds appear optimal in most cases. Exceptions occur when low layers take significantly longer to execute than higher layers. These cases, however, represent those instances where the low layers are computationally slow compared with the verification and pose-estimation layers above. In such instances, the use of the lower layers is usually unjustified, as they fail as efficient focus of attention heuristics.

6. Implementation and Results

In this section, we demonstrate several implemented IFA systems which robustly track various objects.

6.1. IFA Face Tracking

Our current real-time implementation runs on a single-processor 266 MHz Pentium II PC with a Matrox Meteor framegrabber.

The system uses seven layers (see Fig. 5 for four of the layers in action). In the following, all variables, k, indicate a constant threshold set empirically.

Layer 0 randomly selects one of nine regions of the search space, roughly corresponding to those states which would project a face to one of nine rectangles in the image.

Layer 1 selects regions of the search space corresponding to pixels exhibiting skin color. Skin color is defined to be those RGB values where

$$k_{rg}^{-} < r/g < k_{rg}^{+}$$
$$k_{rb}^{-} < r/b < k_{rb}^{+}.$$

This color model describes a thin pyramidal wedge in RGB space which accepts skin colors of all races under approximately white light. Although probabilistic or adaptive color models are conceivable (Oliver et al., 1997; Raja et al., 1998), this layer is meant as an initial attention-focusing scheme only; a fast, liberal color model is preferred to a slower, more precise one which might miss actual skin under varying illumination. Any pixel not classified as skin-color immediately eliminates a range of configurations from the search space: facial configurations that would project skin-color to image pixels not classified as skin-color are thrown out.

Layer 2 is a color- and motion-based selector that is biased toward regions nearest the last observed position of the target. The search spirals outward and selects state sets consistent with those pixels which exhibit both skin color (as in Layer 1) and large motion,

$$\frac{dI(\mathbf{x})}{dt} > k_m.$$

$I(\mathbf{x})$ represents the image intensity at pixel $\mathbf{x}$. Search space reduction proceeds in a manner similar to Layer 1.

Layer 3 uses *radial spanning* to find the approximate size and shape of a single cluster of skin-colored pixels. Radial spanning is a fast cluster detection algorithm: Pixel probes are initialized at the center of the predicted position of the face and then extended radially outward until they find non-skin-colored pixels (Toyama, 1998). Probes are affected by forces as follows:

$$F_i = F_i^{\text{out}} + F_i^{\text{in}} + F_i^{\text{int}}, \quad \text{where}$$

$$F_i^{\text{out}} = k_{\text{out}}, \quad \text{if pixel at } \mathbf{x}_i \text{ is skin color,}$$

$$F_i^{\text{in}} = k_{\text{in}}, \quad \text{if pixel at } \mathbf{x}_i \text{ is not skin color,}$$

$$F_i^{\text{int}} = k_{\text{int}} * \frac{(2\mathbf{x}_i - \mathbf{x}_{\mathbf{p}(i)} - \mathbf{x}_{\mathbf{s}(i)}) \cdot \mathbf{v}_i}{|2\mathbf{x}_i - \mathbf{x}_{\mathbf{p}(i)} - \mathbf{x}_{\mathbf{s}(i)}|}$$

where i indexes a predetermined number of probes (16

Figure 5. Face tracking at different layers. Layer 1: skin-colored pixels are marked black. Layer 3: skin-color blob tracking. Layer 5: a rectangle marks SSD tracking using a stored template. Layer 6: Full pose estimation; facial features are tracked using small templates.

are used for face tracking), $\mathbf{x}_i$ is probe i's pixel location, $\mathbf{v}_i$ is probe i's expansion direction, and $\mathbf{p}$ and $\mathbf{s}$ return the predecessor and successor indices of i. The algorithm is similar to *draping* (Turk, 1996), but without the static background restriction, and to balloons (Cohen, 1991), but with greater efficiency and resistance against spurious edges. The size and centroid of the resulting blob gives an approximate estimate of the center of the face in 3D. Thus, the search space is reduced considerably.

Layer 4 is the same as Layer 3, with an additional computation of the principal axis of the cluster; approximate position and in-plane orientation are tracked. This is the first step in which configurations corresponding to a range of orientations are eliminated from the search space.

Layer 5 performs linearized sum-of-squared-differences (SSD) tracking (Hager and Belhumeur, 1998), which iteratively matches live images to a stored template acquired during a manual initialization step:

$$\mathbf{X}_{t+1} = \mathbf{X}_t - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T(\mathbf{I}(\mathbf{X}_t, t+1) - \mathbf{I}(\mathbf{0}, t_0)),$$

where $\mathbf{X}$ is a vector $[x\ y\ \theta]^T$, $\mathbf{I}(\mathbf{X}, t)$ is a vector representing the image at time t translated and rotated according to $\mathbf{X}$, $\mathbf{J}$ is the is the Jacobian of $\mathbf{I}$ with respect to $\mathbf{X}$, and $\mathbf{I}(\mathbf{0}, t_0)$ is the original template. Fine position and in-plane orientation are tracked. This layer works only for approximately frontal poses. In those poses, by accepting only those matches which have a low SSD residual value, it ensures that the remaining configurations correspond to a particular individual, and not just any skin-colored object—face or otherwise. In addition, both in-plane position and rotation are more precisely localized, further thinning the search space.

Layer 6 tracks 5 point features of the face including the eyes, the nostrils, and the mouth. Features are initialized based on their expected relative locations with respect to the Layer 5 template. The eyes, nostrils, and upper lip are tracked by small templates (7×7 at 1/2 resolution), which are matched by sum of square differences. These features are then used to determine the 3D transformation from a face reference frame to the camera reference frame by finding a least-squares fit between an approximate geometric model of facial features and the tracked features under weak perspective:

$$\mathbf{T} = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{Q},$$

where $\mathbf{T}$ is the recovered transformation matrix, $\mathbf{P}$ is a 3×5 matrix of the concatenated image feature positions with the optical axis coordinate held constant, and $\mathbf{Q}$ is the 4×5 concatenation of the five model points in homogeneous coordinates. The assumption that the face is oriented toward the camera allows full recovery of 6-DOF pose. At this point, the configuration space is shrunk to a small set of configurations corresponding to faces which appear very similar to the target face and which are in poses within a margin of error of ground truth.

6.1.1. Experiments. We have tested the system extensively under various types of visual perturbations. Four sets of experiments were performed for each type of perturbation event: (TT) a temporary event during `track` mode, (TS) a temporary event during `search` mode, (PT) a permanent event in `track` mode, and (PS) a permanent event in `search` mode. All experiments for `track` mode were performed with the tracking system initially at the top layer. All experiments for `search` mode were begun at the bottommost layer. Fig. 6 shows the results of the experiments.

Briefly, the results can be summarized as follows: With a temporary perturbation, the system, if it is affected at all, recovers to full-precision tracking of the target no matter what type of disturbance occurs and regardless of the layer then executing. Recovery never occurs for permanent perturbations where the disturbance prevents target confirmation in any tracking layer. Overall, the system is highly robust to any type of temporary disturbance. The same conclusions were informally drawn from repeated demonstrations to skeptical, well-informed audiences (Hager and Toyama, 1996).

6.1.2. False Positives. The worst error a tracking system can make is to track the wrong object, believing it to be the correct target. As mentioned in Section 5.1.2 such false positives should only occur in an IFA framework when there are distractors for the top layer or when sleight of hand occurs.

The case of top-layer distractors is all but ruled out in face tracking by the exacting nature of the SSD tracking algorithm which expects a strict match with the stored template. Over the course of hundreds of experiments, the face tracking system has never locked on to a face other than the one on which it was initialized.

Sleight of hand is a rare, but real problem. An example of this is when the system is tracking the face based solely on color (Layer 3, `track` mode), and another flesh-toned object passes in front of the face. It is possible in this instance that the system will begin

Mode:	TT	TS	PT	PS
Change in hue	none	slows recovery	none	prevents recov
Change in intensity	none	none	none	none
Lights out	loss & recovery	slows recovery	loss	prevents recov
Medium speed	drop(1) & recov	slows recovery	drop(1)	prevents recov
Fast speed	drop(2) & recov	slows recovery	drop(2)	prevents recov
Very fast speed	loss & recovery	slows recovery	loss	prevents recov
Flesh-toned distract	none	slows recovery	none	slows recovery
Other faces distract	none	slows recovery	none	slows recovery
Twin distract	none	distracts & recov	none	confuses 50%
Partial Occlusion	varies	none	varies	none
Full Occlusion	loss & recovery	slows recovery	loss	prevents recov
Target Distortion	varies	slows recovery	varies	prevents recov
Noise	varies	slows recovery	varies	prevents recov
Scale Changes	drop(3) & recov	slows recovery	drop(3)	prevents recov
Out-of-plane Rotation	drop(3) & recov	slows recovery	drop(3)	prevents recov
Disappearing Target	loss & recovery	slows recovery	loss	prevents recov

Figure 6. The effect of various visual disturbances on face tracking performance. Numbers after "drop" entries indicate the number of layers descended. Some disturbances affect the system differently based on the extent of the disturbance. In these cases, the effect varies from no effect at all to mistraking.

tracking the second object instead of the face, without recognizing its error. This situation can be contrived, and the system does in fact mistrack, recovering only if the second object itself undergoes a significant visual disturbance.

In practice, this scenario has never occurred even during hour-long demonstrations, primarily because it is rare for anything flesh-toned and larger than the target face to occlude the target completely, and then remain in the image. Occlusions with smaller flesh-toned objects, such as the subject's hands, are not distracting enough to the color tracking algorithm to pull tracking away, and other occlusions simply cause complete mistracking (during which time the system acknowledges its temporary inability to track).

6.1.3. Actual Recovery Times. Figure 7 shows where the tracking systems spends it time during some example runs. The horizontal axis represents time and the vertical axis represents the IFA layer executed by the system. The graph displays a concatenation of several time intervals during a sample run of the face tracking system, rearranged so that disturbances occur every 5 s. The sample run itself began with a fully initialized system at Layer 6.

At the 10 s mark, a finger was passed quickly in front of the subject's eye. At 15 s, the subject briefly turned his head approximately 15 degrees to the right. In both instances, top-layer tracking was disturbed, causing the system to fall to an SSD tracker (Layer 5) momentarily, but as the occlusion ceased or as the head turned to face

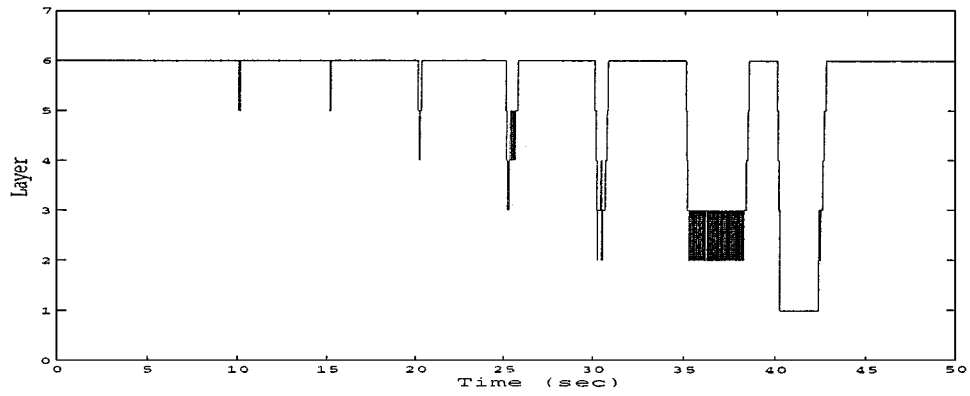

Figure 7. Face tracking layers during execution. Tracking occurs at Layer 6 for the most part. Every five seconds, a visual disturbance occurs. See text (Section 6.1.3) for details.

forward, the system immediately climbed back up to the top-layer tracker. At 20 s, the subject scratched his nose, causing tracking to descend to coarse SSD tracking, Layer 4. Tracking fell to Layer 3, skin-color blob tracking, after a sudden jerk of the face at the 25 s mark. At 30 s, a hand was passed over the entire face. The system descends all the way to Layer 2, where the algorithm searches for a moving object near the last known location of the face. Since the occlusion ends quickly, the second candidate output set of Layer 2 (a selector) contains the target and tracking recovers quickly. In the disturbance at 35 s, tracking bounces between Layers 2 and 3 while the face moves rapidly back and forth. During this time, the frame-to-frame velocity of the head is great enough that the system does not attempt to match the face with the SSD template. Finally, at the 40 s mark, the subject exits the field of view, forcing a full search of the image for skin-colored blobs. None are found, so the system remains at Layer 1 until the subject returns 2.4 s later, at which point the face is found and tracking resumes at full precision.

Recovery when the subject is in view and background clutter is minimal takes no more than 0.66 s from Layer 1.

The last experiment (with the subject temporarily leaving the field of view) was repeated under different levels of clutter (Fig. 8). As more clutter is added to the background, the tracking system must examine more disjoint subsets of the state space before finding a subset that contains the actual target, requiring longer recovery times (Fig. 9).

The data correspond closely with the analysis of expected recovery times from Section 5.2.2 and Section 5.2.3. The expected recovery time should increase linearly with the number of flesh-toned objects in the scene. Repeated experiments (40 trials) confirm these figures (see Fig. 10). This trend suggests analogies with pop-out and camouflage effects in biological vision systems.

6.1.4. Tradeoff Management. Part of the robustness of an IFA system arises from its ability to dynamically manage tradeoffs between different tracking needs. For example, consider the tradeoff between allowable tracking speed and precision. IFA systems are constructed so that more precise algorithms occupy higher layers. Since no one would choose to use a slow, imprecise algorithm when a faster, more precise one is available, a consequence of this organization is that layers tend to be sorted in order of decreasing speed. In particular, those layers which have corresponding `track` nodes in the state transition graph show an increase in precision together with a decrease in speed.

In Fig. 11, we plot results from a continuous sample run of the face tracking system over 20 s, during which time the subject moved from left to right in the image, being careful not to cause the system to fall below Layer 2. The values plotted are ground truth as measured by a Polhemus Fastrak device, tracked estimates, layer number (multiplied by 10), and tracking error as a function of time. Only values relevant to the horizontal position of the subject are plotted. As comparisons between the layer number and the tracking error show, error in the estimate is inversely related to the layer being executed, with the least error present when tracking occurs at Layer 6.

Similar tradeoffs between the extent of other visual perturbations (extent of occlusion, deviation from object description, etc.) and precision are also handled by IFA in the same manner.

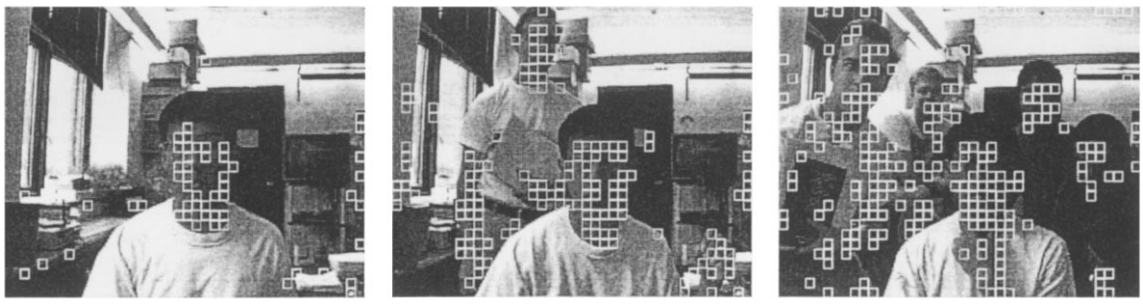

Figure 8. Experimental setup for various levels of clutter. Regions considered flesh-toned are marked. Note that as clutter increases, more regions are marked, and consequently, the number of candidate states increases.

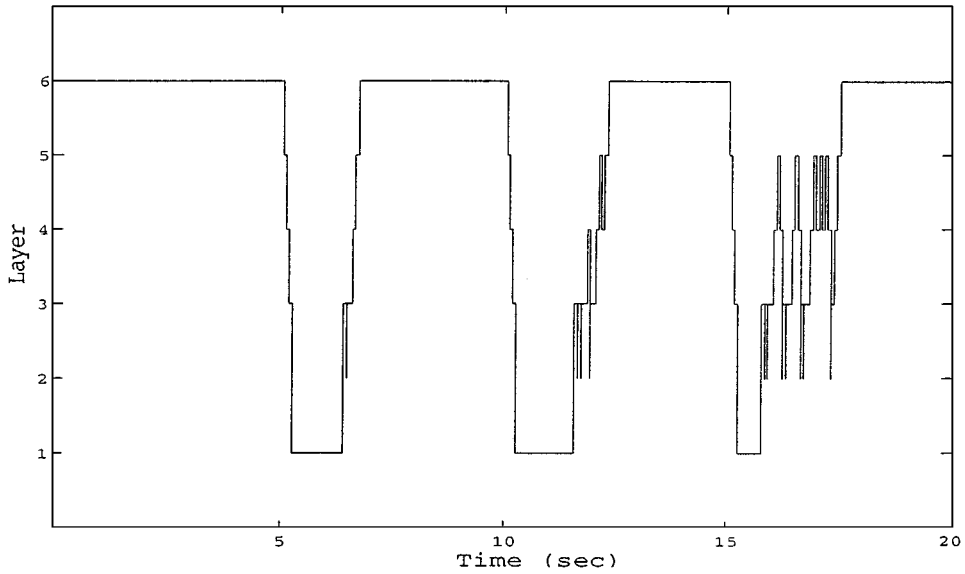

Figure 9. Layers with different degrees of clutter. At 5 s, low clutter; 10, medium clutter; 15, high clutter. The recovery time (time it takes to climb from Layer 1 to 6) increases with greater amounts of clutter.

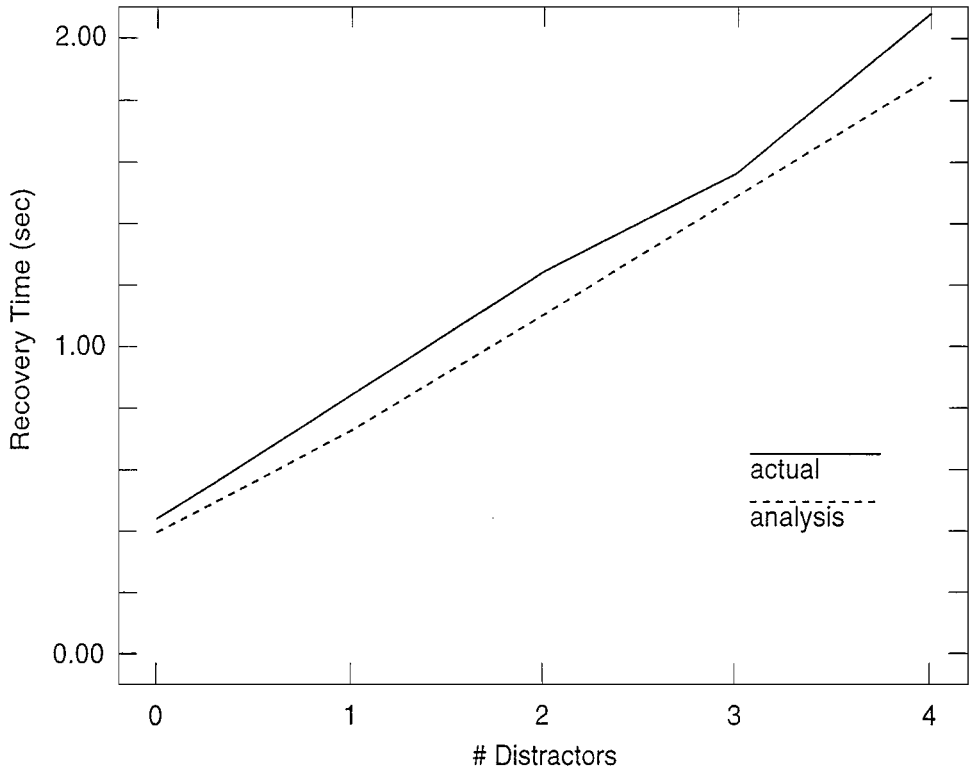

Figure 10. Actual (solid, 40 trials) and computed (dashed) mean recovery times for zero to four distractors.

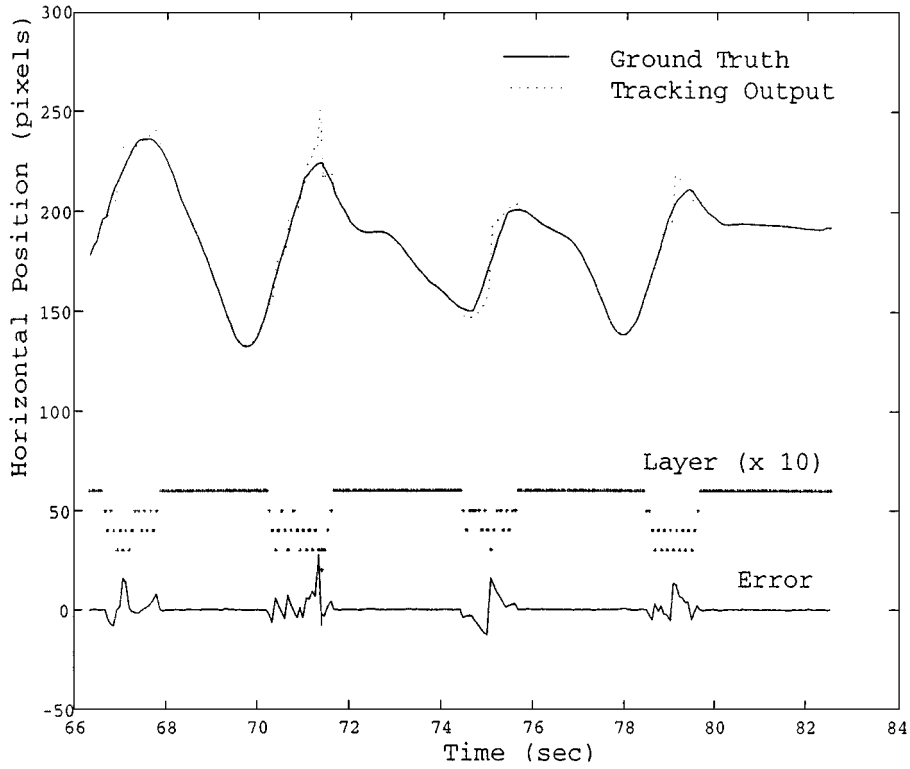

Figure 11. Layers exhibit a tradeoff between speed and precision. Ground truth (as measured by a Polhemus tracking device) and tracking estimates of the horizontal position of a tracked subject are shown at top. The middle dots indicate the layer of operation. At the bottom, the difference between ground truth and tracking estimates. Note that increases in the error correspond to dips in the layer of execution.

6.2. Other IFA Systems

All of the systems described below were implemented either on a Sun Sparc 20 equipped with a standard framegrabber connected to a single-chip color CCD camera or on a Silicon Graphics Indigo with VINO color camera.

In the past, we have often used 3.5 inch diskettes as a convenient target for various experiments with visual servoing (Toyama et al., 1996). These experiments, while successful in demonstrating hand-eye coordination, were nevertheless extremely sensitive to disturbances—a consequence of the local nature of the edge tracking used. To make this system more robust to visual disturbances, we 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 feature-based rectangle tracker (layer 3).

The results for disk tracking are qualitatively similar to the results for face tracking. The major difference in performance was that more disturbances caused tracking to fail completely. The poorer performance can be explained by the lack of robustness in the high-layer tracking for rectangles. But, as with face tracking, the system was able to rapidly recover from all temporary perturbations.

In another illustration of IFA applied to robotics, we developed a robust doorknob tracking module for use with mobile robots (Feiten et al., 1997). This particular implementation used an *intermediate object* (Wixson and Ballard, 1994), where a search for the door takes place to facilitate finding the doorknob itself (see Fig. 12). Restriction of attention to doors helps both to speed up the search as well as to constrain the search in an environment full of edge-rich objects.

Finally, we constructed a two-sided object tracker which tracks an object by matching to multiple views.

The implementation is identical to that for face tracking, except that the facial feature trackers are

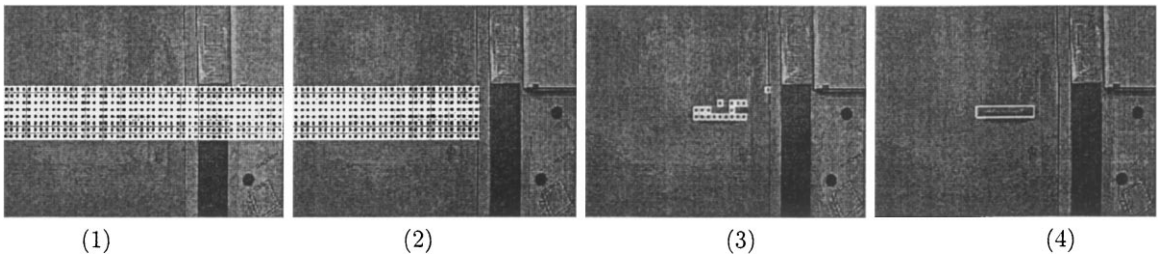

Figure 12. Layers in doorknob search. Regions marked by white indicate remaining regions of the search space projected onto the image. (1) position constraint, (2) color search, (3) horizontal edge search, (4) polygon tracking.

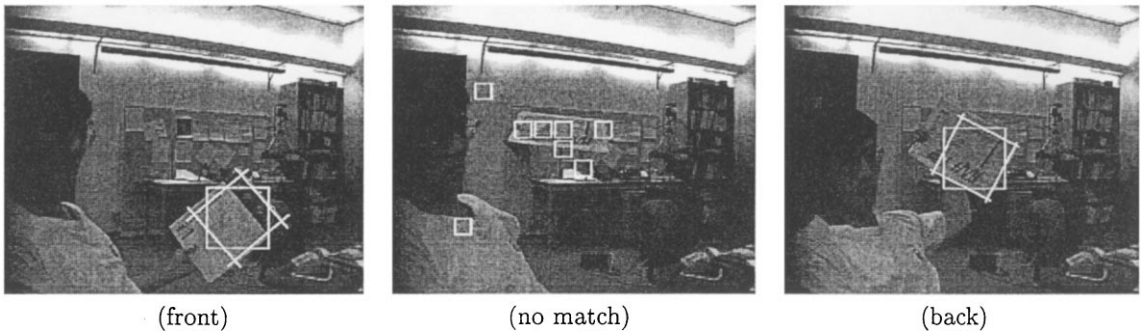

Figure 13. Tracking an envelope with two different sides. In the center image, the envelope is tilted too much to match with either stored image, so tracking has reverted to color tracking.

replaced by SSD-based multi-pattern trackers (Hager and Toyama, 1998), 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 Fig. 13, we see how two images were used to track an envelope. The front and back of the envelope were initialized as the images to track. In the left and right figures, tracking proceeded at the top layer, 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.

Performance of two-sided tracking was similar to that of face tracking, although recovery times were greater due to the need to compare the image with two stored templates.

7. Conclusion

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 the environment resettles, tracking will have resumed. In a word, tracking is tenacious.

One of the advantages of the IFA framework from a programmer's point of view is the relative ease with which existing trackers can be incorporated into it. The bottom layers with some variation suit almost all tracking applications. Then, given existing trackers, it is usually easy to decide when they mistrack 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. Placing the most precise tracker in the innermost loop fashions the top of the framework and added robustness is the result.

Further work with IFA proceeds along several avenues. One interesting problem is the automatic construction of IFA systems, given some a priori knowledge

about the desired target. It should be possible to design and initialize an IFA system given statistical models for what faces look like, how often they appear in an image, and so on. Another direction for future work lies in creating dynamic IFA systems which swap layers in and out depending upon visual conditions. Finally, we are attempting to further formalize the notions of "robustness" and "graceful degradation" with the goal of providing a theoretical framework for evaluating the robustness of IFA and other real-time systems.

Acknowledgments

This research was supported by Siemens AG, 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

Bar-Shalom, Y. and Fortmann, T.E. 1988. *Tracking and Data Association*. Academic Press.

Bar-Shalom, Y. and Li, X.-R. 1993. *Estimation and Tracking: Principles, Techniques, and Software*. Artech House.

Blake, A., Curwen, R., and Zisserman. A. 1993. Affine-invariant contour tracking with automatic control of spatiotemporal scale. In *Proc. Int'l Conf. on Computer Vision*, Berlin, Germany, pp. 421–430.

Bradshaw, K.J., McLauchlan, P.F., Reid, I.D., and Murray, D.W. 1994. Saccade and pursuit on an active head/eye platform. *Image and Vision Computing*, 12(3):155–163.

Burridge, R. Rizzi, A., and Koditschek, D. 1995. Towards a dynamical pick and place. In *Proc. Int'l Conf. Intel. Rob. and Sys.*, Vol. 2, pp. 292–297.

Burridge, R., Rizzi, A., and Koditschek, D. 1999. Sequential composition of dynamically dexterous robot behaviors. *Int'l Journal Robot. Res.*, 18(6):534–555.

Burt, P.J. 1988. Attention mechanisms for vision in a dynamic world. In *Proc. Int'l Conf. on Patt. Recog.*, pp. 977–987.

Burt, P.J. and van der Wal, G.S. 1990. An architecture for multiresolution, focal, image analysis. In *Proc. Int'l Conf. on Patt. Recog.*, pp. 305–311.

Cohen, L.D. 1991. On active contour models and balloons. *CVGIP: Image Understanding*, 53(2):211–218.

Concepcion, V. and Wechsler, H. 1996. Detection and localization of objects in time-varying imagery using attention, representation and memory pyramids. *Patt. Recog.*, 29(9):1543–1557.

Coombs, D. and Brown, C.M. 1993. Real-time binocular smooth-pursuit. *Int'l J. of Computer Vision*, 11(2):147–165.

Crowley, J. and Berard. F. 1997. Multi-modal tracking of faces for video communication. In *Proc. Computer Vision and Patt. Recog.*, pp. 640–645.

Culhane, S.M. and Tsotsos, J.K. 1992. An attentional prototype for early vision. In *Proc. European Conf. on Computer Vision*, Italy, pp. 551–560.

Feiten, W., Hager, G.D., Bauer, J., Magnussen, B., and Toyama, K. 1997. Modeling and control for mobile manipulation in everyday environments. In *Proceedings of the 8th International Symposium on Robotics Research.*

Gennery, D.B. 1992. Visual tracking of known three-dimensional objects. *Int'l J. of Computer Vision*, 7(3):243–270.

Hager, G. and Toyama, K. 1998. XVision: A portable substrate for real-time vision applications. *Computer Vision and Image Understanding*, 69(1):23–37.

Hager, G.D. and Toyama, K. 1996. XVision: Interaction through real-time visual tracking. In *CVPR Demo Program*. San Francisco.

Hager, G.D. and Belhumeur, P.N. 1998. Efficient region tracking with parametric models of geometry and illumination. *PAMI*, 20(10):1025–1039.

Huber, E. and Kortenkamp, D. 1995. Using stereo vision to pursue moving agents with a mobile robot. In *Proc. Int'l Conf. on Robot. and Autom.*, Nagoya, Japan, pp. 2340–2346.

Isard, M. and Blake, A. 1996. Contour tracking by stochastic propagation of conditional density. In *Proc. European Conf. on Computer Vision*, Vol. I, pp. 343–356.

Isard, M. and Blake, A. 1998. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. In *ECCV98*, pp. 893–908.

Kahn, R.E., Swain, M.J., Prokopowicz, P.N., and Firby, R.J. 1996. Gesture recognition using Perseus architecture. In *Proc. Computer Vision and Patt. Recog.*, pp. 734–741.

Kass, H., Witkin, A., and Terzopoulos, D. 1987. Snakes: Active contour models. *Int'l J. of Computer Vision*, 1:321–331.

Kosaka, A. and Nakazawa, G. 1995. Vision-based motion tracking of rigid objects using prediction of uncertainties. In *Proc. Int'l Conf. on Robot. and Autom.*, Nagoya, Japan, pp. 2637–2644.

Lowe, D.G. 1992. Robust model-based motion tracking through the integration of search and estimation. *Int'l J. of Computer Vision*, 8(2):113–122.

Maki, A., Nordlund, P., and Eklundh, J. 1996. A computational model of depth-based attention. In *Proc. Int'l Conf. on Patt. Recog.*, p. D9E.1.

Murray, D. and Basu, A. 1994. Motion tracking with an active camera. *IEEE Trans. Patt. Anal. and Mach. Intel.*, 16(5):449–459.

Neisser, U. 1967. *Cognitive Psychology*. Appleton-Century-Crofts, New York.

Nishihara, H.K. 1996. Real-time vision. In *CVPR Demo Program*.

Nordlund, P. and Uhlin, T. 1996. Closing the loop: Detection and pursuit of a moving object by a moving observer. *Image and Vision Computing*, 14:265–275.

Oliver, N., Pentland, A., and Berard, F. 1997. LAFTER: Lips and face real time tracker. In *Proc. Computer Vision and Patt. Recog.*

Pahlavan, K. and Eklundh, J.-O. 1992. A head-eye system – analysis and design. *CVGIP: Image Understanding*, 56:41–56.

Prokopowicz, P., Swain, M., and Kahn, R. 1994. Task and environment-sensitive tracking. Technical Report 94-05, University of Chicago.

Raja, Y., McKenna, S.J., and Gong, S. 1998. Tracking and segmenting people in varying lighting conditions using colour. In *Proc. Int'l Conf. on Autom. Face and Gesture Recog.*, pp. 228–233.

Rasmussen, C., Toyama, K., and Hager, G. 1996. Tracking objects by color alone. DCS RR-1114, Yale University.

Reid, D.B. 1979. An algorithm for tracking multiple targets. *IEEE Trans. Autom. Control*, 24:843–854.

Terzopoulos, D. and Rabie, T.F. 1995. Animat vision: Active vision in artificial animals. In *Proc. Int'l Conf. on Computer Vision*, pp. 801–808.

Terzopoulos, D. and Szeliski, R. 1992. Tracking with Kalman snakes. In *Active Vision*, A. Blake and A. Yuille (Eds.), MIT Press: Cambridge, MA.

Toyama, K. 1998. Radial spanning for fast blob detection. In *Proc. Int'l Conf. on Comp. Vision, Patt. Recog., and Image Proc.*

Toyama, K. 1998. *Robust Vision-Based Object Tracking*. PhD Thesis. Yale University.

Toyama, K. and Hager, G. 1997. If at first you don't succeed... In *Proc. AAAI*, Providence, RI, pp. 3–9.

Toyama, K., Wang, J., and Hager, G. 1996. SERVOMATIC: a modular system for robust positioning using stereo visual servoing. In *Proc. Int'l Conf. Rob. and Autom.*, pp. 2636–2643.

Treisman, A. 1985. Preattentive processing in vision. *CVGIP: Image Understanding*, 31:156–177.

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

Tsotsos, J.K. 1995. Towards a computational model of visual attention. In *Early Vision and Beyond*, T. Papathomas, C. Chubb, A. Gorea, and E. Kowler (Eds.), MIT Press, pp. 207–218.

Turk, M. 1996. Visual interaction with lifelike characters. In *Proc. Automatic Face and Gesture Recognition*.

Uhlin, T., Nordlund, P., Maki, A., and Eklundh, J.-O. 1995. Towards an active visual observer. In *Proc. Int'l Conf. on Computer Vision*, Cambridge, MA, pp. 679–686.

Vincze, M. 1996. Optimal window size for visual tracking. In *Proc. Int'l Conf. on Patt. Recog.*, p. A91.1.

Wixson, L.E. and Ballard, D.H. 1994. Using intermediate objects to improve the efficiency of visual-search. *Int'l J. of Computer Vision*, 12(2-3):209–230.

Wolfe, J. 1995. Guided search 2.0: A revised model of visual search. *Psychonomic Bulletin and Review*, 1(2):202–238.

Wren, C.R., Azarbayejani, A., Darrell, T., and Pentland, A. 1997. Pfinder: Real-time tracking of the human body. *IEEE Tran. Patt. Anal. and Mach. Intel.*, 19(7):780–785.