Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)

REAL-TIME HIERARCHICAL SWARMS FOR RAPID ADAPTIVE MULTI-LEVEL PATTERN DETECTION AND TRACKING

Matthias Scheutz

Artificial Intelligence and Robotics Laboratory
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
mscheutz@cse.nd.edu

ABSTRACT

In this paper, we introduce a hierarchical extension to the standard particle swarm optimization algorithm that allows swarms to cope better with dynamically changing fitness evaluations for a given parameter space. We present the formal framework and demonstrate the utility of the extension in an application system for dynamic face detection. Specifically, the feature detector/tracker uses the proposed "hierarchical real-time swarms" for a continuous concurrent dynamic search of the best locations in a two-dimensional parameter space and the image space to improve upon feature detection and tracking in changing environments. We show in several experimental evaluations on a robot interacting with people in real-time that the proposed method is robust to lighting changes and does not require any calibration. Moreover, the method is not limited to face detection, but can be applied to any n-dimensional search space.

1. INTRODUCTION

Particle swarm optimization (PSO) has been successfully employed in a variety of applications to quickly find optimal or close-to-optimal parameters in partly high-dimensional parameters spaces [5, 6]. The idea behind PSO is to use n particles or *agents* (that together form the *swarm*) to explore different regions of the parameter space in parallel. Swarm agents, like agents in a many biological swarms, attract each other to varying degrees dependent on the value of their location in the parameter space, thus causing agents in general to move towards better places in parameter space. If the value surface of the parameter space is smooth and there are pronounced peaks, agents will eventually gather around them. While this is desirable for static value evaluations, it can be problematic in the dynamic case where fitness surfaces change and peaks can turn into valleys, in which case swarm agents need to start their gradient ascent again.

In this paper, we propose a hierarchical version of standard swarm systems, where multiple swarm agents investigate different search spaces at different temporal and spatial levels of granularity in parallel. The different multi-scale, multi-level swarms are mutually linked via their evaluation functions that define the quality of a position in the search space. Groups of low-level swarm agents "define" high-level swarm agents by virtue of their locations (in their search space), which will get instantiated if they meet certain instantiation criteria. Once instantiated, they start to impose constraints on their low-level constituent agents. Specifically, the position of higher-level swarm agents in their search space is used in the evaluation of lower-level swarm agents to correct their update, thus possibly attracting lower-level agents to regions in the search space that they would have otherwise not visited. This top-down propagation of high-level structural constraints limits the search regions of lower-level swarms and leads to faster, often better dynamic low-level swarm configurations that find and track fitness maxima in parameter spaces very quickly, even with dynamically changing fitness evaluations (exactly because high-level constraints are part of the low-level evaluation). We will demonstrate the proposed method by applying hierarchical swarms to the real-time vision problem of finding and tracking facial features on an autonomous robot.

2. BACKGROUND AND RELATED WORK

We start with a brief review of swarm agents (or particles) in PSO systems, then review real-time swarms and introduce our hierarchical swarm concept, briefly comparing it to other previously used proposals.

2.1. Particle Swarm Optimization(PSO)

An *agent* (or *particle*) is characterized by a location $\bar{x}$ in an n-dimensional parameter space P. The parameter space has an associated (static) *evaluation* function $\mathcal{E} : P_D \mapsto \Re$

(from the parameter space into the real numbers) that determines the quality or *fitness* of a given location (i.e., set of parameter values) for each location in P_D based on an evaluation domain D. Each agent (located in the parameter search space) has a velocity in each dimension that varies with time, depending on its own current and past position and the current and past positions of other agents.

The velocity $v_{i,j}(t)$ of agent i in dimension j at time t is given by

$$v_{i,j}(t) = \omega \cdot v_{i,j}(t-1) + c_1 \cdot \phi_1 \cdot (p_{i,j} - x_{i,j}(t-1)) + c_2 \cdot \phi_2 \cdot (p_{g,j} - x_{i,j}(t-1)) \quad (1)$$

This equation has three components: an *inertia* component $\omega \cdot v_{i,j}(t-1)$, which pulls the agent in its current direction; a *cognitive* component $p_{i,j} - x_{i,j}(t-1)$, which represents the agent's memory of its highest-scoring location $p_{i,j}$ in that dimension (based on $\mathcal{E}$) and its comparison to the current location; and a *social* component $p_{g,j} - x_{i,j}(t-1)$, which is the distance from the global best location $p_{g,j}$ discovered among all agents [14].

The position of each agent in each dimension is then updated by the equation

$$x_{i,j}(t) = x_{i,j}(t-1) + v_{i,j}(t)$$

Typically, swarm agents in a PSO system are initially placed in random locations in P and then updated for a certain number of iterations, either until a stable position of all agents has been obtained (e.g., all agents end up in the same location or close-by with little to no movement) or a solution has been found by at least one agent that is "good enough".

[9] assert that most prior work would give task-specific values to the coefficients c_1, c_2 and ω. However, van den Bergh [12, 13] showed that these must satisfy the inequality

$$\frac{c_1 + c_2}{2} < \omega \quad (2)$$

to exhibit convergent behavior. Convergent behavior is desirable so swarms are attracted to the best determined value so that region can be more thoroughly examined for a maximum. However, in a non-smooth search space, this is not true; in a space composed of randomly-placed impulses (single points where the fitness function is high), a swarm requires more luck than ability in finding these areas.

2.2. Real-Time Swarms

[7] have extended the notion of static evaluation to *dynamic evaluation*, i.e., to a sequence $\mathcal{E}_I = \langle \mathcal{E}_{i_1}, \mathcal{E}_{i_2}, \ldots \rangle$ of functions $\mathcal{E}_{i_k} : \mathcal{P} \mapsto \Re$, $I := \{i_1, i_2, ...\}$. Dynamic evaluations reflect situations where the quality of locations in parameter spaces can change over time (e.g., the success of a particular set of parameters that determines the foraging strategy of a simulated animal might change based on the food distribution in the environment). A sequence of functions is called *real-time*, or *real-time evaluation* if a metric $\mathcal{M}$ is defined for the index set I, i.e., $\mathcal{M}$ is defined on $\{i_k | \exists \mathcal{E}_{i_k} \in \mathcal{E}_I\}$. Real-time sequences can be used to model the temporal characteristics of changes of parameter evaluations in an environment (e.g., the shift of shaded locations under a tree on a sunny day).

Depending on the degree of change in the evaluation between $\mathcal{E}_{i_k}$ and $\mathcal{E}_{i_{k+1}}$, swarm agents will be able to cope to varying degrees: minor smooth changes in the fitness surface will lead to quick adaptations, but large transitions (e.g., from a value peak to a value valley) could render the swarm system largely immobile or lead to very slow adaptations. This can be especially problematic in real-time evaluations, where the number of iterations that swarms can use for adaptation is constrained by a real-time interval. Consequently, it might be useful to add mechanisms to swarms that would allow them to react to changes more quickly.

A very simple, yet highly effective mechanism is to treat a swarm system of n agents as performing m-beam search (with $m < n$). In this case, as in genetic algorithms, n agents are initialized in randomly generated locations in the search space. The agents are ranked according to their performance based on the fitness evaluation $\mathcal{E}_{i_k}$. The best m agents are kept, while the other agents are eliminated and are replaced by *offspring* based on a *selection strategy* (e.g., mutation of the locations of existing chosen agents, cross-over of their dimensions, or simply a random location drawn from a random distribution of the overall parameter space to reach relatively uniform coverage). However, while the d lowest-performing agents are culled, the best-performing agents are not crossed-over. On the next iteration, d agents are instantiated in the system and compared against the surviving agents from the previous round. Surviving agents are competing against newly-instantiated agents while seeking maxima. This combines the maximum-seeking tendency of swarm-based search with the strengths of culling (i.e., faster convergence) and random searching (as implemented in GAs with mutations and cross-over, e.g. [2]) to quickly find and converge on peaks in a multidimensional space.

2.3. Hierarchical Swarm Systems

In a situation where a system's performance is defined by the performance of several swarm systems, a higher-level of abstraction can be used to link these systems. From this higher level, it is possible to evaluate agents in relation to each other and evaluate their contributions to a group performing independent tasks, instead of simply using the highest-scoring results from each independent system (e.g., evaluating the features of a face detection system like eyes, eyebrows, nose, mouth, etc. based on their relation to each other).

As in the single swarm system, the i-th swarm agent in a multi-level swarm system then can be identified by its position in some parameter space $x^l_{k,i}$, where the superscript l indicates the *level* of the space in the swarm hierarchy. Because each level may contain several independent swarm systems, the subscript k denotes one of these systems.[1] Furthermore, we let $X^l_k := \bigcup_i \{x^l_{k,i}\}$, so that X^l_k represents the k-th swarm system at level l, which uses parameter space P^l_k.

Each agent $x^l_{k,i} \in X^l_k$ evaluates an object T^l (e.g., a image) by decoding the position into a parameter set. The function RE^l_k uses the parameters from $x^l_{k,i}$ and some number of static parameters to perform some operations on T^l and yield some result. This result can be represented as a point in a result space $\hat{P}^l_k$ on the same level as P^l_k. This result extractor can be defined as:

$$RE^l_k : P^l_k \times \Re^{n_l} \times T \mapsto \hat{P}^l_k$$

Moreover, the resulting point $\hat{x}^l_{k,i} \in \hat{P}^l_k$ can be *evaluated* by

$$E^l_k : \hat{P}^l_k \mapsto \Re$$

which would serve as the value of the swarm agent in P^l_k.

Now suppose that the function E^l_k is sufficient for the local analysis of a feature, but not for the performance of the overall system. While E^l_k provides local evaluation, X^l_k would contribute better to the overall system if it could use additional information from X^l_h, $h \neq k$.

Hence, we define a combination function C^l to combine the results generated by X^l_k, X^l_h, and any other swarm system at level l, which results in new agents in a level $l+1$:

$$C^l : \mathcal{P}(\hat{P}^l_k) \times \mathcal{P}(\hat{P}^l_h) \times ... \mapsto \mathcal{P}(P^{l+1}_k)$$

The system X^{l+1}_k has a mapping function FE^{l+1}_k of its own, which maps to a point $\hat{x}^{l+1}_k \in \hat{P}^{l+1}_k$, which is evaluated by E^{l+1}_k.

$$\begin{aligned} & E^{l+1}_k(\hat{x}^{(l+1)}_{k,i}) \\ &= E^{l+1}_k(RE^{l+1}_k(x^{l+1}_{k,i}, ..., T^{l+1}) \\ &= E^{l+1}_k(RE^{l+1}_k(C^l(\hat{x}^l_{k,i}, \hat{x}^l_{h,i}, ...)), ..., T^{l+1}) \\ &= E^{l+1}_k(RE^{l+1}_k(C^l(RE^l_k(x^l_{k,i}, ..., T^l), RE^l_k(x^l_{h,i}, ..., T^l), ...), \\ &\quad ..., T^{l+1}) \end{aligned}$$

So, the evaluation of a single point at level $l+1$ is, by extension, an evaluation the swarm agents in the combinations of agents at level l.

[1] $x^l_{k,i,j}$ then denotes the position of $x^l_{k,i}$ in the dimension j.

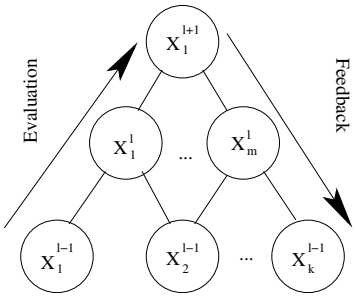

Figure 1. The hierarchical swarm system, showing the path of feedback.

For a swarm system, however, an evaluation is meaningless unless it has some effect on the swarms. Traditionally, the function E^l_k will provide feedback to the agent $x^l_{k,i}$. Because the function E^{l+1}_k has a fitness which can be related back to the swarms X^l responsible for x^{l+1}_k, the results can be used to affect the lower-level agents.

There are two methods to reward the lower-level swarms. The first is to promote the agents in X^l, $X^l = \bigcup_k \{X^l_k\}$, which contribute to the global maximum x^{l+1}_k to be the superior agents in their respective systems. This establishes that the relation between agents is of the utmost importance. The second is to provide a reward to the agents in X^l which contribute to x^{l+1}_k, variant on its evaluation E^{l+1}_k, establishing that the relation is only a component of evaluation and not an absolute. When doing this, it is possible for agents on level l to be artificially inflated due to their presence in several well-performing $x^{l+1}_{k,i}$ agents, even if none of these are the best. Therefore, if rewards are taken from a strictly-increasing set S and awarded in order from worst-performing to best-performing agents in X^l_k, the set S must be such that $\forall k : \sum_{i=1}^{k} S_i < S_{k+1}$.

Figure 1 illustrates the feedback cycle. Results $\hat{x}^l_k$ are combined by the function C^l and mapped to points $x^{l+1}_{k,i}$. These points are transformed by RE^{l+1} and evaluated, triggering a reward to the constituents from level l.

There are similarities, but also important differences between this hierarchical swarm system and others, for example, the multi-level system described in [8], where swarm agents move across an image to locate features such as edges and corners and submitting the resulting configuration to a classification system which categorizes the object, similar to the upper-level evaluation performed here to evaluate determined features. While their system performs image extraction and decodes agent positions to identify features, our proposed system decodes agent positions to serve as parameters for feature extraction.

Our proposal also differs from the system described in [4], where the swarm hierarchy modifies the velocity up-

date equation of the agents based on the parent-child relations. However, the disparity in that system lies between the agents in a single evaluation space, while the purpose of the hierarchy proposed in this paper is to use different levels of evaluation to bridge different evaluation functions $E_k^l, E_h^l, etc.$ on one level, using higher levels of knowledge at each level so the individual systems perform better in combination, not just independently. The proposed system modifies the space P_k^l itself, by changing the value of $x_{k,i}^l$. It does not alter the velocity of x in any direction. Because the agent score now depends on agents in a system h (and possibly others on the same level l), a gradient can change depending on these other systems. As such, agent $x_{k,i}^l$ may have two different values at t and $t+1$, although FE and $E_{k,i}^l$ will yield the same evaluations.

Because of the RE functions, however, it cannot be guaranteed there is a mapping backwards to P_k^l. The function E_k^{l+1} may reveal a way to "correct" x_k^{l+1} to produce a higher-utility result from RE_k^{l+1}. Even if this can be done, the change must be reflected at level l so the swarms in X_k^l can converge on the new point. In order to do that, the change would have to move backward through RE so the agents can move toward that point. If there is no RE^{-1} to retrieve a corresponding value P_k^l, there is no way to modify the velocity of any x in X_k^l to move toward the new point.

3. HIERARCHICAL SWARM-BASED ADAPTIVE FEATURE DETECTION AND TRACKING

In this section, we will demonstrate the utility of the hierarchical swarm system with a 2-level system for real-time face tracking.

3.1. Swarm Systems for Image Processing

Various swarm-based methods have been used to analyze images in the past. For example, the edge detection system in [16, 17, 10] or the color segmentation system in [15] featured a swarm system which allocate one agent per pixel in its method of edge detection. These methods, however, are image-specific as the agents move over the pixels. Unlike parameter-based methods where a search range can be applied to a series of images, the features would have to be calculated over every frame.

[11] presented a method using ant colonies to explore dynamic spaces, similar to the dynamic field of the parameter space explored by the swarm agents. All of the memory of an ant's position is left on each discrete space as pheromones, with no knowledge of the actual positions of the other ants. This provides a very thorough search; however, this requires thousands of ants to explore the search space so pheromones can accumulate and be tracked before evaporating. [2] offers a genetic algorithm approach to tracking maxima across dynamic fitness landscapes; the fixed hypermutation model is similar to the random replacement of the swarm agents in our system, and this model was shown to perform best over an abruptly-changing landscape.

Finally, [7] demonstrated the utility of evolving swarms for real-time evaluations in the traditionally-challenging domain of facial feature detection and tracking. That work is extended here by using the upper-level swarm system to relate the products of the lower-level swarm systems and provide feedback based on their collective strength. We show that this improves over the previous system by using the higher level to choose superior features and agents, while reducing the need to regularly reinforce the search regions against a given facial region.

3.2. A 2-Level Hierarchical Swarm for Feature Tracking

The lower level $l = 0$ will be a two-dimensional space providing parameters to a Canny edge detector [1], with $k = [0,2]$ as spaces for each of three features (left eyebrow, right eyebrow, mouth). The higher level $l = 1$ will be defined by triples of points, one from each system, which will be evaluated by their shape.

To make the domain-dependence of the evaluation functions more explicit, we consider the parameters of the visual feature detection system as variables to a function

$$
\begin{aligned}
&F : RE_k^0(s_1, ..., s_m, x_{k,1}^0, img)\\
&F \in \hat{P}_k^0 : \{\langle x, y, w, h\rangle | x, y, w, h \in \mathcal{Z}\}\\
&Img : \{\langle x, y, r, g, b\rangle | x, y \in \mathcal{Z}, r, g, b \in \Re \geq 0\}\\
&RE_k^0 : \mathcal{P}(Img) \times \Re \times \Re \times ... \times \Re \mapsto \mathcal{P}(\hat{P}_k^0)
\end{aligned}
$$

where img is the image to be processed, $s_1, ..., s_m$ are static parameters of the system, and $x_{k,1}^0$ provides parameters to this system in the form of thresholds to the edge detector RE_k^0 is the extraction function, and F is the set of extracted features. Further, $score = E_k^0(F)$ can apply evaluation function $E_k^0 : \mathcal{P}(\hat{P}_k^0) \mapsto \Re$ to a feature that has been extracted. The space $\hat{P}_k^0$ is defined by the (x, y) coordinate of the top-left point of the bounding box, and w and h are the width and height of that bounding box.

Using swarms for feature detection allows for the adaptation of the system necessary to perform well on an autonomous robot in a real-world environment. Variable parameters of a feature detection space can be represented as dimensions in this parameter space, and the position of each swarm agent represents values for each of these parameters; edge detection thresholds can be represented in 1-2 dimensions and color blob detection can be represented in 6 (a lower and upper bound for each of three color dimensions). A fitness function is applied to the detected features. These

FUNCTION evalThreeFeatures($lPolar, rPolar, lCentroid, rCentroid, mCentroid$)
$headTilt \leftarrow \frac{rCentroid.y - lCentroid.y}{lCentroid.x - rCentroid.x}$
$headAngle \leftarrow tan^{-1}(headTilt)$
$lTheta \leftarrow lPolar.theta - headAngle$
$rTheta \leftarrow rPolar.theta - headAngle$
if $lTheta < 0$ **and** $rTheta > 0$ **then**
  $maxAngle \leftarrow max(-lTheta, rTheta)$
  $maxRadius \leftarrow max(lPolar.r, rPolar.r)$
  $angleS \leftarrow |\frac{lTheta + rTheta}{maxAngle}|$
  if $rTheta < angleThresh$ **then**
    $angleS \leftarrow angleScore + 0.25$
  end if
  if $lTheta > -angleThresh$ **then**
    $angleS \leftarrow angleScore + 0.25$
  end if
  $diffS \leftarrow abs(\frac{lPolar.r - rPolar.r}{maxRadius})$
  $distS \leftarrow \frac{lPolar.r + rPolar.r}{2500} + \frac{oldR - 2500}{2500}$
  if $distScore > 1.0$ **then**
    $distS \leftarrow max(2.0 - diffScore, -1.0)$
  end if
  $moveS \leftarrow moved(lCentroid) + moved(rCentroid) + moved(mCentroid)$
  $score \leftarrow scaleA \cdot angleS + scaleDiff \cdot diffS + scaleDist \cdot distS + scaleM \cdot moveS$
else
  $score \leftarrow -2.0$
end if
return *score*

Figure 2. The algorithm for scoring the relation between three detected features.

are compared to determine the highest-performing swarm, which represents the best candidate for parameters.

An edge-detection algorithm was chosen over a color-based method to meet the speed constraints of a real-time system. Edge-detection can be performed on an intensity image (a black-and-white image obtained by color transformation) using two dimensions, where the dimensions in space represent the upper and lower thresholds of the Canny edge detector [1]. Straightforward color detection systems (e.g. [3]) which detect color regions, would require a six-dimensional search space. Intuitively, this will require more agents, more time, or both to determine the optimal value. A search of parameter space is preferable to directly searching a specific image. Moving between adjacent coordinates in parameter space should provide smooth transitions in the image processing (color blobs and edges should not suddenly vanish between adjacent points). This concept is critical for our real-time system, as it capitalizes on the re-use of the determined parameters.

Unlike the edge detection system in [17, 16, 18] or the color segmentation system in [15], the swarm system described in this paper does not allocate one agent per pixel. Rather, these are used to explore a two-dimensional space representing the upper and lower thresholds of a Canny edge detector. [14, 9] show how swarm agents can be used to quickly explore multi-dimensional spaces. Each agent is instantiated at a random point in this 2D space with a random velocity in each dimension, within the range of [0,1], and fixed attraction to global and local best values.

The features F_k^0 map to a point in P^{l+1}: a space defined by the positions of the centroids of the bounding boxes (from each $\hat{x}_{k,i}^l$). These positions generate a triangle feature in $\hat{P}^{l+1}$ which is evaluated by E_0^{l+1}. The `scoreGeom` function determines what features are present and submits combinations to evaluation functions E_k^{l+1}. While this paper discusses evaluation of only one case (i.e. if all features are present), it would be possible in general to define functions E_k^{l+1} for $k > 0$ to evaluate other feature combinations.

Depending on how well the feature set scores, this function will set one of two options for *nextMode*. If the score does not exceed *geomThresh*, $nextMode \leftarrow RECOVER$ and the features for the next frame are evaluated by their position inside a face. Otherwise, $nextMode \leftarrow REGION$ and the next frame will be evaluated at the lower-swarm level solely on their shapes.

The `evalThreeFeatures` function in Figure 2 per-

forms the E^{l+1} evaluation of the three detected features to determine their fitness as a unit. The polar coordinates represent the positions of the eyebrows with respect to the mouth, and the rectangular coordinates are used to calculate the slope between the eyebrows. This is used to determine the tilt of the head (assuming the eyebrows are level).

The value *angleS* is determined by the difference between the mouth-left eyebrow angle, and the mouth-right eyebrow angle ; the perfect score is zero, so $scaleA < 0$. An additional increase is made if the angle violates a bound $angleThresh$, to limit the sharpness of the feature triangle. Because the eyebrows are expected to be roughly the same distance from the mouth, $diffS$ determines the difference between the two features' distance to the mouth and scales it by the largest, to yield a score [0,1].

The distance from the mouth to the eyebrows, scored by *distS*, will reinforce that the features simply don't "wander" too far from each other. This is difficult to enforce, as the acceptable distance will vary with the subject's distance from the camera.The simple implementation was to use distance between the features from the last frame as the peak value in the current frame. To prevent the features from moving rapidly between frames, a score *moveS* is assigned, calculated by determining each feature's distance from the feature used on the previous frame.

The functions responsible for implementing the function RE_k^0 (`LocateBestFeature`, `DetermineBoundAndScore`, `DetermineScore`) are based on those in [7]. `LocateBestFeature` determines the global best swarm agent on each iteration and assigns rewards to agents based on the features they trigger. `DetermineBoundAndScore` uses image-processing techniques with parameters determined by the location of x_k^0 to extract a feature, and `DetermineScore` evaluates the feature. When using the higher-level swarms, it is possible to use a modified version of the evaluation function `DetermineScore` or provide a new one entirely, depending on the information given by the higher-level evaluation. Because the higher-level swarm provides face information (by its evaluation and tracking of the features which form the best faces), this information can be used in place of the positions given by the Haar cascade or the 3D region.

4. EXPERIMENTS AND RESULTS

To test the system, two subjects with differing facial attributes were placed in front of an ActivMedia Peoplebot equipped with a Unibrain Firewire camera and their faces were tracked with different methods. One subject had dark skin and prominent facial features (for the purposes of this example, this means thick eyebrows). The other subject had light skin and less prominent features. The different subjects served several purposes for testing the system:

1. The different colors of the subjects' hair and skin will change the effect of light on both subjects. A change in lighting resulted in stronger changes in the contrast between the illuminated and darkened faces of the fair-skinned subject, than in the dark-skinned subject. This stronger change results in a sharper change in the intensity difference between the skin and eyebrow, which in turn results in a change in the parameters necessary for edge detection.

2. Different facial-feature prominence has a twofold effect: the contrast between the feature and the skin, which affects the parameters of the edge detection; and the dimensions of the feature, which will affect the evaluation of eyebrows of differing shapes.

A gallery for each subject was recorded with a lamp shining to the left. Each subject raised and lowered his eyebrows throughout the recording. Approximately halfway through the recording, the light was switched off.

Three feature-detection systems were applied to each subject gallery, composed of 96 images of the dark-skinned subject and 93 of the fair-skinned subject. The first used the `DetermineBoundAndScore` algorithm for a fixed parameter in P_0^k. The second used the swarm system with annotated face regions on each image, replacing the lowest-performing 60% of agents on each iteration; these systems used the OpenCV Haar cascade to locate a face on every frame. The third used the hierarchical system with the same replacement rate as the agent system, but with no annotated images (except for the first frame), using the hierarchy to provide relational feedback. We also produce a set of images with hand-annotated features used as a baseline for the comparison of the three feature detection methods.

The block of 12 images on the left in Fig. 3 shows the fair-skinned subject under changing lighting conditions, while the block of 12 images on the right shows the dark-skinned subject, both under the same conditions: the first row is with a lamp on, the second row is just after the lamp was turned off, the third is 1-2 seconds after the lamp has been turned off. The first column in each block shows the hand-annotated features, the second column shows the static analysis, the third columns shows the images after processing with the swarm-based system, and the fourth shows the images after processing with the hierarchical system.

A two-way 3x2 ANOVA was conducted for the independent variables *method* (static, dynamic, hierarchical) and *person* (dark,light), and the dependent variable *summed feature error*, which are the summed differences between the hand-annotated bounding boxes along the diagonals $d_{sum} = \sum_{k=1}^{n}(t_k + b_k)$, where t_k is the distance from the top-left point of the determined eyebrow to the top-left

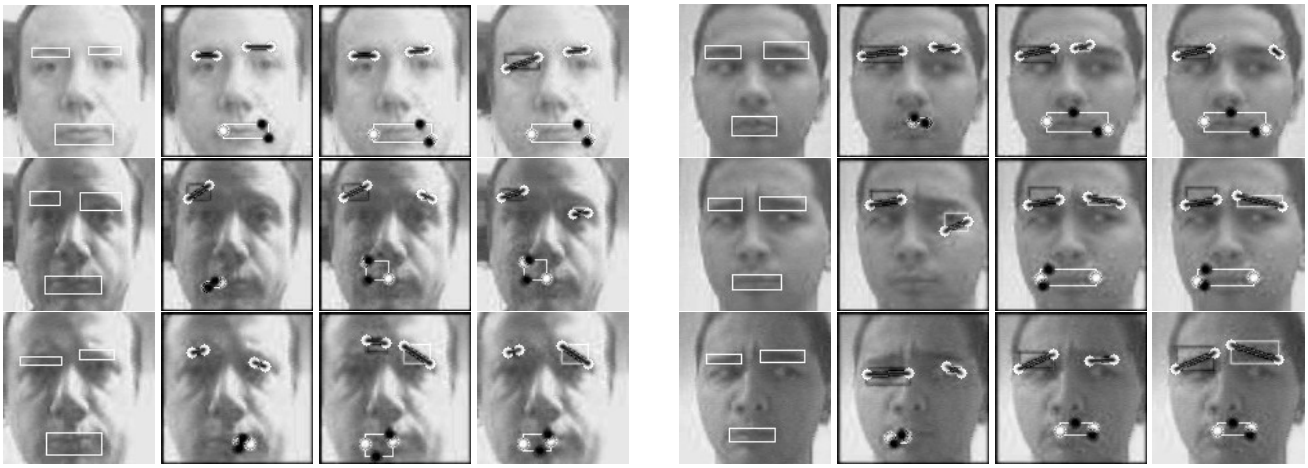

Figure 3. **Left block:** fair-skinned subject, **Right block:** dark-skinned subject. **Top row:** full lighting, **Middle row:** dimmed lighting, **Bottom row:** ambient lighting only. *For each subject:* **Column 1:** annotated image, **Column 2:** static detector, **Column 3:** swarm system with given face, **Column 4:** hierarchical system, face only given for first frame

point of the actual eyebrow, and b_k is the distance from the bottom-right point of the determined eyebrow to the bottom-right point of the actual eyebrow, summed for all three features. We obtained highly significant main effects on method ($F(2,561) = 31.4, p < .001$), indicating that the swarm-based methods are significantly better than the static method, and person ($F(1,561) = 36.2, p < .001$), indicating that the features of light-skinned person were significantly more difficult to track than those of the dark-skinned person. There was no significant interaction. Moreover, a reduced 2x2 ANOVA with method restricted to "swarm" and "hierarchical" again shows a highly significant effect on person ($F(1,374) = 49.8, p < .001$), but no significant effect on method ($F(1,374) = 1.4, p = .24$). Hence, the swarm system with pre-determined regions of interest (based on the Haar face detector) and the hierarchical swarm (without any such help) have about the same performance. Table 4 shows a summary of the evaluation results.

	static	swarm	hierarchy
left	2608.418	1774.5584	2214.4127
right	3929.241	2082.952	2551.309
mouth	4356.451	3379.492	2896.113
total	10894.11	7237.002	7661.834
$\mu(\sigma)$	57.6(36.7)	38.2(18.4)	40.5(21.3)

Table 1. Sum of pairwise diagonal length differences over 189 images, for each feature (left eyebrow, right eyebrow, mouth) and totals (including mean average error and standard deviation) using each method (static edge detection parameters, swarm-based parameters, and swarm-based parameters with hierarchical evaluation).

5. DISCUSSION

In the hierarchical swarm system, a parameter space P_k^0 is shaped by lighting and feature properties, and altered by other systems, through E_k^1. Each agent's position in the parameter space will map to $n \cdot m$ values, where n and m are the number of features returned by the other two swarm systems. However, only one of these values will affect the parameter space; the agent with the best-placed feature with relation to the others is promoted to the highest position, which will cause the other agents to slowly move towards it, regardless of the value from E_k^0. This analysis provided superior performance while performing fewer image searches for faces, because it incorporates important information, unavailable from the level of the feature extraction.

The significant improvement over the static system demonstrates the utility of the hierarchical swarm system; as with the swarm-based feature detection, the hierarchy adapts to changing lighting conditions. Further, the lack of statistical difference between the hierarchy and the non-hierarchical swarm with the "Haar" face detector shows that the hierarchical systems yields similar performance without the cost of the face detector, which can be viewed as a perfect higher-level swarm agent (note that the features are guaranteed to be within regions of the face reinforced by this higher-level agent). So the hierarchical system is doing a statistically-comparable job constraining the positions of the detected features.

6. CONCLUSION

The swarm systems are capable of exploring multidimensional search spaces and finding suitable features in real-

time. The particular hierarchical swarm system for face detection/tracking provided adaptability to changing lighting conditions, which caused static detection systems to fail. By exploiting higher-level knowledge about the expected configuration of the detected lower-level features, the position of a feature in relation to others was used as a reinforcing measure in the evaluation of lower-level swarms. The presence of facial features in appropriate configurations also provides further evidence that the region is indeed a face, in lieu of computationally expensive face-detection functions, which improves the performance of the system. The evaluation of the system on an autonomous robot shows that real-time face detection/tracking based on hierarchical swarm systems is feasible under changing lighting conditions in a dynamic environment. We expect further improvements of the system based on additional higher-level constraints such as 3D models of faces or context-dependent possibly non-spatial constraints (e.g., based on information about the presence of faces).

7. ACKNOWLEDGMENT

The author would like to thank Christopher Middendorff for implementing the algorithms described in this paper.

8. REFERENCES

[1] J. Canny. A computational approach for edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[2] John J. Grefenstette. Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In *Proceeding of the 1999 Congress on Evolutionary Computation*, pages 2031–2038. IEEE Press, 1999.

[3] Rien-Lien Hsu, Mohamed Abdel-Mottaleb, and Anil K. Jain. Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):696–706, 2002.

[4] Stefan Janson and Martin Middendorf. A heirarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(6):1272–1282, 2005.

[5] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. IEEE Int. Conf. Neural Networks*, volume 4, pages 1942–1948, 1995.

[6] J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm Intelligence*. Academic Press, 2001.

[7] Christopher Middendorff and Matthias Scheutz. Real-time evolving swarms for rapid pattern detection and tracking. In *ALIFE X*, 2006.

[8] Tanya Mirzayans, Nitin Parimi, Patrick Pilarski, Chris Backhouse, Loren Scott-Wyard, and Petr Musilek. A swarm-based system for object recognition. *Neural Network World*, 15(3):243–255, 2005.

[9] M. Omran, A. P. Engelbrecht, and A. Salman. Particle swarm optimization method for image clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(3):297–321, 2005.

[10] Vitorino Ramos and F. Almeida. Artificial ant colonies in digital image habitats: A mass behavior effect study on pattern recognition. In M. Dorigo, M. Middendorf, and T. Stuzle, editors, *From Ant Colonies to Artificial Ants - 2nd Int. Wkshp. on Ant Algorithms*, pages 113–116, 2000.

[11] Vitorino Ramos, Carlos Fernandez, and Agostinho C. Rosa. Social cognitive maps, swarm collective perception and distributed search on dynamic landscapes. *Brains, Minds and Media - Journal of New Media in Neural and Cognitive Science*, 2005.

[12] F. van den Bergh. *An analysis of particle swarm optimizers*. PhD thesis, University of Pretoria, 2002.

[13] F. van den Bergh and A. P. Engelbrecht. A new locally convergent particle swarm optimizer. In *Proc. IEEE Conf. Systems, Man, and Cybernetics*, 2002.

[14] Mark S. Voss. Social programming using functional swarm optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium 2003*, pages 103–109, 2003.

[15] Charles E. White, II, Gene A. Tagliarini, and Sridhar Narayan. An algorithm for swarm-based color image segmentation. In *Proceedings of the IEEE SOUTHEASTCON - 2004 "Engineering Connects"*, pages 84–89, 2004.

[16] X. Zhuang. Edge feature extraction in digital images with the ant colony system. In *IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pages 133–136, 2004.

[17] X. Zhuang. Image feature extraction with the perceptual graph based on the ant colony system. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, pages 6354–6359, 2004.

[18] X. Zhuang. Image segmentation by ant swarm - a case study of digital signal processing with biological mechanism of intelligence. In *2004 IEEE 11th Digital Signal Processing Workshop*, pages 143–146, 2004.