A HIERARCHICAL SWARM SYSTEM FOR ROBUST REAL-TIME FEATURE DETECTION

A Thesis

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Master of Science

by

Christopher Middendorff, B.S.

Matthias J. Scheutz, Director

Graduate Program in Computer Science and Engineering
Notre Dame, Indiana
April 2006
A HIERARCHICAL SWARM SYSTEM FOR ROBUST REAL-TIME FEATURE DETECTION

Abstract

by

Christopher Middendorff

In this thesis, we discuss a simple extension to the standard particle swarm optimization algorithm, inspired by genetic algorithms that allow swarms to cope better with dynamically changing fitness evaluations for a given parameter space. We demonstrate the utility of the extension in an application system for dynamical facial feature detection and tracking, which uses the proposed “real-time evolving swarms” for a continuous dynamic search of the best locations in a two-dimensional parameter space to improve upon feature detection with static parameters. We show in several experimental evaluations that the proposed method is robust to lighting changes and does not require any calibration. Moreover, the method works in real time, is computationally tractable, and not limited to the employed static feature detector, but can be applied to any n-dimensional search space.

Further, this thesis introduces a novel 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. It 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.
To my parents, who inspired my pursuit of higher education.
CONTENTS

FIGURES .................................................................................. v

TABLES .................................................................................. vi

PREFACE .................................................................................. vii

CHAPTER 1: INTRODUCTION .......................................................... 1

CHAPTER 2: REAL-TIME EVOLVING SWARMS ................................. 3
  2.1 Particle Swarm Optimization(PSO) ......................................... 3
  2.2 Real-Time Swarm Systems .................................................. 4
  2.3 Hierarchical Swarm Systems ................................................. 6
  2.4 Summary ........................................................................... 10

CHAPTER 3: THE UTILITY OF SWARM SYSTEMS FOR FACIAL FEATURE DETECTION .............................................................. 11
  3.1 Swarm-based Adaptive Feature Detection and Tracking .......... 11
  3.2 Experiments and Results ..................................................... 20
  3.3 Discussion ......................................................................... 24
  3.4 Summary ........................................................................... 26

CHAPTER 4: A HIERARCHICAL EXTENSION FOR FEATURE DETECTION .............................................................. 27
  4.1 Hierarchical Swarm-based Adaptive Feature Detection and Tracking .... 28
  4.2 Experiments and Results ..................................................... 34
  4.3 Discussion ......................................................................... 36
  4.4 Summary ........................................................................... 38

CHAPTER 5: IMPLEMENTATION ON A ROBOTIC PLATFORM .................. 39
  5.1 Person Recognition Goals .................................................. 39
  5.2 Feature Extraction ........................................................... 41
  5.3 Overall Implementation ...................................................... 42
FIGURES

2.1 The hierarchical swarm system, showing the path of feedback. ........ 9

3.1 The algorithm for determining eyebrows. ........................ 14

3.2 The algorithm for locating the best features. ....................... 16

3.3 The algorithm for determining boundary and score. ............... 17

3.4 The algorithm for determining score. ............................... 18

3.5 **Top row:** full lighting, **Middle row:** dimmed lighting, **Bottom row:** ambient lighting only. .................... 19

3.6 **Top:** full lighting, **Center:** dimmed lighting, **Bottom:** ambient lighting only. ................................. 25

4.1 The algorithm for determining what features are present, and the functions to call based on this determination. ................. 29

4.2 The algorithm for scoring the relation between three detected features. ... 30

4.3 An illustration of the model used for facial feature geometric comparisons. 32

4.4 **Top block:** fair-skinned subject, **Bottom 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 .......................... 35

5.1 The robotic platform used for human-robot interaction. ............ 40

7.1 **Top row:** transition from surprise to neutral, **Bottom row:** transition from smile to neutral. ................................. 59
TABLE 4.1  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).
This work is an implementation of a concept by Dr. Matthias Scheutz for using swarm agents to quickly explore a space which is defined by the possible features that can be extracted using the coordinates of a point in that space. He was also responsible for the statistical calculations comparing the different systems.

This thesis describes this concept in detail. The definition of the function which defines the space is critical to the performance of the feature-detection system. Each function has to define a feature in a manner which is sufficiently vague to allow for variations across different people. Another space, which is defined by the positions of these features, evaluates the features (and their corresponding swarm agents, by extension) based on both their shape, and their relations to each other.

This is based on the work done in [27] and [28], which define the swarm system and the hierarchical swarm system, respectively. This work has also been used in [35, 34] for tracking faces for human-robot interactions.

I would like to thank my adviser, Dr. Matthias Scheutz, whose support made this work possible and whose analyses of the data demonstrate the viability of the system. I would also like to thank Paul Schermerhorn not only for his availability as a sounding board, but for his invaluable assistance as a test subject and proofreader. Finally, I thank David Anderson, whose work for the past two years has not only connected the feature recognition to the overall robotic system, but expanded the capabilities of the vision system as well.
CHAPTER 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 (e.g., [17, 18]). 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. Agents, as in a biological swarm, 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 thesis, we propose a simple mechanism inspired by genetic algorithms that will allow swarm agents to dynamically evolve, thus helping them to find and track fitness maxima in parameter spaces very quickly based on dynamically changing fitness evaluations. We will demonstrate the proposed method by applying evolving swarms to the real-time vision problem of finding and tracking facial features.

Because the facial features have particular relationships due to their geometric relationships, this information can be further used to ensure that agents in independent swarm systems are rewarded in a fashion that promotes a collective advancement instead of each
swarm’s *individual* advancement. To that end, the proposed swarm system is extended
to a hierarchical version of the 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 attract-
ing 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.
CHAPTER 2

REAL-TIME EVOLVING SWARMS

We start with a brief review of swarm agents (or particles) in PSO systems, review previous work on real-time swarm systems, and introduce the hierarchical swarm concept.

2.1 Particle Swarm Optimization (PSO)

We start with a brief review of swarm agents (or particles) in PSO systems. An agent or particle is characterized by a location $\mathbf{x}$ in an $n$-dimensional parameter space $P$. The parameter space has an associated (static) evaluation function $E : P \rightarrow \mathbb{R}$ (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

$$ v_{i,j}(t) = \omega \cdot v_{i,j}(t-1) + c_1 \cdot \Phi_1 \cdot (p_{i,j} - x_{i,j}(t-1)) + c_2 \cdot \Phi_2 \cdot (p_{g,j} - x_{i,j}(t-1)) \quad (2.1) $$
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 $E$) 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 [40].

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”.

[30] assert that most prior work would give task-specific values to the coefficients $c_1$, $c_2$ and $\omega$. However, van den Bergh [38, 39] showed that these must solve the inequality

$$\frac{c_1 + c_2}{2} < \omega$$

(2.2)

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 Swarm Systems

We now extend the notion of static evaluation to *dynamic evaluation*, i.e., to a sequence $\mathcal{E}_I = (\mathcal{E}_{i_1}, \mathcal{E}_{i_2}, \ldots)$ of functions $\mathcal{E}_k : P \mapsto \mathcal{R}, I := \{i_1, i_2, \ldots\}$. 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). We call such a sequence of functions 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 $\{i_k | \exists \mathcal{E}_{i_k} \in \mathcal{E}_I\}$. In other words, 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 states 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, crossover of their dimensions, or simply a random location drawn from a random distribution of the overall parameter space to reach relatively uniform coverage). 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 genetic algorithms with
mutations and cross-over, e.g. [12]) to quickly find and converge on peaks in a multidimensional space.

It is the dual nature of the swarms in the system that allows for rapid feature detection. The highest-performing swarms, due to their preservation between iterations and between frames, seek optimal solutions in their current location (which corresponds to the best values for the frame). However, between frames it is possible that their current location may no longer be the best location. In these cases, it is the randomly-placed swarms that can quickly find the best locations, due to the fact that they are already distributed throughout the space; they do not need to seek because they are already in place.

2.3 Hierarchical Swarm Systems

These solutions are fine for evaluating their individual spaces. However, there are qualities of the detected features that are not adequately represented in the individual systems, such as the geometry between them.

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 $x_{k,i}^l$, 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.\footnote{$x_{k,i}^l$ then denotes the position of $x_{k,i}$ in the dimension $j$.} Further...
thermore, 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:

$$RE^l_k : P^l_k \times \mathbb{R}^n \times T \mapsto \hat{P}^l_k$$

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

$$E^l_k : \hat{P}^l_k \mapsto \mathbb{R}$$

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 perform evaluation using additional information from other swarm systems $X^l_h$, $h \neq k$.

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 space at level $l + 1$:

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

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 $E^{l+1}_k$. 


\[ E^{l+1}_k(x^{l+1}_{k,i}) \]
\[ = E^{l+1}_k(RE^{l+1}_k(x^{l+1}_{k,j}, \ldots, T^{l+1})) \]
\[ = E^{l+1}_k(RE^{l+1}_k(C(x^{l}_{k,j}, x^{l}_{h,i}, \ldots)), \ldots, T^{l+1}) \]
\[ = E^{l+1}_k(RE^{l+1}_k(C(RE^{l}_k(x^{l}_{k,j}, \ldots, T^{l}), RE^{l}_k(x^{l}_{h,i}, \ldots, T^{l})), \ldots, T^{l+1}) \]

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 \( l \).

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

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 can be this is 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 $(RE_k^l)^{-1}$ 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.

Additionally, feedback from the upper level cannot contribute to the agent’s velocity. It is possible to evaluate the features relations to each other; however, even if the expected location for the image is calculated, and even if the feature is actually there, it is not possible to modify the velocity of an agent in such a way that the detected feature gravitates to the perfect feature. That feature has a corresponding location in the value space. If it was possible to know where that location was without having an agent search for it, then finding the features would be trivialized. An agent’s position in value space maps to a location in the image space, but the mapping does not go the other direction.

Figure 2.1: The hierarchical swarm system, showing the path of feedback.
2.4 Summary

For simple tasks, a single swarm in a single parameter space may be sufficient for an optimization problem. However, if the performance of that system is not actually isolated, but dependent on the performance of other swarm systems, then the swarm hierarchy can be used to combine results and evaluate agents in independent swarm systems and promote agents whose results contribute to a superior global result, instead of those agents whose results indicate a local best.
CHAPTER 3

THE UTILITY OF SWARM SYSTEMS FOR FACIAL FEATURE DETECTION

As shown in Section 2.2, swarm systems can be used to explore parameter spaces and retrieve maxima. In this chapter, a system is described where a parameter space is defined which describes the suitability of the parameters for extracting facial features from a face image.

3.1 Swarm-based Adaptive Feature Detection and Tracking

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

\[ features = F(img, k_1, \ldots, k_m, x_1, \ldots, x_n) \]

where \( img \) is the image to be processed, \( k_1, \ldots, k_m \) are non-variant parameters of the system, \( x_1, \ldots, x_n \) are parameters to this system, \( F \) is the extraction function, and \( features \) is the set of extracted features. Further, \( score = g(feature) \) can apply evaluation function \( g(\) \) to a feature that has been extracted, to provide feedback to the swarm system.

Determining values \( x_1, \ldots, x_n \) such that \( score \) is maximized for \( g \) requires searching an \( n \)-dimensional parameter space. While this can be done manually\(^1\), this is not only tedious but also subject to changes in \( img \).

\(^1\)Certainly, a parameter space may be so large that a “perfect solution” is impossible to pinpoint; however, it is possible to find a solution by trial-and-error that is sufficient for a specific feature-detection task.
Learning algorithms will be insufficient for the same reasons as the static parameters; conditions vary too widely to hope to encompass all possible cases. Using swarm agents, capable of continuously moving through the parameter space, it is possible to converge on a fitness maximum (not necessarily the global one, however).

In order to perform successfully in real-time, a vision system must be implemented that can quickly analyze each frame for useful data. In the case of feature detection, this can mean searching a parameter space to determine appropriate values (color ranges, edge detection thresholds) that will allow the feature to be cleanly extracted.

Using swarms for feature detection grants the adaptability necessary to perform 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). As the algorithm progresses, a fitness function is applied to the resulting detected features. These 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 [4]. Straightforward color detection systems (e.g. [14]) 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.

Unlike the edge detection system in [48, 47, 49] or the color segmentation system in [41], the swarm system described in this thesis will not allocate one agent per pixel. Rather, these will be used to explore a parameter space, whose dimensions define proper-
ties of a feature detector (similar to the [30] system for image segmentation). In this case, the parameter space is two-dimensional, representing the upper and lower thresholds of a Canny edge detector. As seen in [40] and [30], 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.

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 requires the re-use of the determined parameters. The parameter-space search is useful because it is not bound to the image space; in both a parameter-based and image-based search, it would be possible to reduce the search area to the region where a previous feature was found. However, if the feature moves, it would still have to be re-sought with an image-based method, where parameters can be used on the region to extract the feature in a parameter-based method.

By searching a parameter space for values to apply to an image, instead of the image itself, the search system can use these parameters on future images by preserving these parameters; the swarm agents do not need to exhaustively search the entire space for every frame, and in fact can re-use this value without searching parameter space at all if desired.

To adapt to changing lighting conditions, the system monitors the evaluation score of features found on subsequent frames; when a small number of frames are evaluated with a score less than a given correctness threshold, the swarms are re-initialized and repeat the evaluation process to determine the corrected values.

**DetermineEyebrow**, shown in Figure 3.1, controls the method used to detect an eyebrow in the feature, as a simple comparison to previous successes and failures. Values of *thresh*, *maxFC*, and *bottomFC* will depend on the evaluation function, the level of
FUNCTION DetermineEyebrow(face, nIters)

```plaintext
eyebox ← determine region for eye
if eyeFC ≥ 0 and eyeFC < maxFC then
    if eyeFC = 0 then
        reset eye swarm
    end if
    result ← LocateBestFeature(eyeSwarm, face, eyebox, nIters)
    eyeFC ← eyeFC + 1
else
    if eyeFC = maxFC OR eyeFC < bottomFC then
        eyeFC ← bottomFC
    end if
    result ← LocateBestFeature(eyeSwarm, face, eyebox, minIters)
    if result.score < thresh then
        eyeFC ← eyeFC + 1
    else
        eyeFC ← eyeFC − 1
    end if
end if
return result
```

Figure 3.1: The algorithm for determining eyebrows.
accuracy sought and the speed desired by the specific system; higher values of \textit{thresh} will enforce a higher threshold of quality, which has the tradeoff that if the swarms have difficulty finding one, performance will be hit. This can be mitigated by decreasing the magnitude of \textit{bottomFailCount}, but this increases the number of unacceptable frames before re-calculation is performed.

The value \textit{minIters} determines the number of iterations the swarm system will spend analyzing new frames; this value can range from 0 (always re-uses the best swarm agent) to \textit{nIters} (constantly performing thorough searches). While it is certainly possible to use a value of \textit{minIteration} such that \textit{minIteration} > \textit{nIters}, it would defeat the idea that after thoroughly searching the parameter space, re-usable values will be discovered.

Determining the eye region is done by using \(\frac{1}{3}\) of the detected face, starting from the top \(\frac{1}{6}\) down to the middle of the detected face. The region submitted for analysis will be dependent on the implemented method of feature detection.

In this specific method, the eye region is also halved by a vertical line through the center; this will also depend on the feature detector, as well as the face detector, and the task to be performed. With the face detector used (OpenCV\(^2\)), dividing the eye region further in half did not affect feature detection; in fact, it allowed an individual swarm to process each half of the face. There is never a guarantee that the face will be lit evenly, due to lighting situations in a room or by shadows the face may cause on itself due to uneven lighting. So the ability to process half of the face increases robustness against not only general lighting changes, but lighting configurations in a room as well.

\textit{LocateBestFeature}, shown in Figure 3.2, is the function responsible for determining the best agent to use for this frame, and subsequently using it to find the feature. Primarily, it submits agents to the function \textit{DetermineBoundAndScore}, which determines the best possible feature bound for that agent (as well as that bound’s score)

\(^2\)http://www.intel.com/research/mrl/research/opencv/
FUNCTION LocateBestFeature(agents, face, eyebox, nIters)
for K from 1 to nIters do
    for J from 1 to length(agents) do
        feature ← DetermineBoundAndScore(agents[J], face)
        if feature.score ≥ agents[J].previousBestScore then
            agents[J].previousBestScore = feature.score
            agents[J].bestLocalPosition = agents[J].curPosition
        end if
        agents[J].modifyCumulative(feature.score)
    end for
    sort(agents) by cumulative scores
    for J from 1 to length(agents) do
        agents[J].updateSwarmAgent(agents[0])
        agents[J].previousWin ← false
    end for
    agents[0].previousWin ← true
end for
reward agents[0]
return DetermineBoundAndScore(agents[0], face)

Figure 3.2: The algorithm for locating the best features.

and returns it. The LocateBestFeature function determines for each agent whether this new location is its personal best, and then accumulates the score. Scores accumulate within an agent but decay with time, giving each agent a memory which extends several frames into the past. After each iteration, the agent positions are updated and the process repeated. The lowest-performing replaceNum agents are replaced after every iteration. The value previousWin is used as a tiebreaker when sorting the swarm agents.

To help expand the search space, the lowest-performing replaceNum agents are removed after each iteration. Removing low-performing agents allows the resources that would otherwise be spent pursuing dead-ends (exploring spaces that evaluate poorly) to be spent exploring new spaces and potentially finding new peaks.

After all iterations, the highest-performing agent is used to determine the bound for
**FUNCTION** DetermineBoundAndScore(agent, image, boxratio)

params ← agent.currentPosition
edgemap ← canny(image, params[0], params[1])
contourlist ← findContours(edgemap)

for K from 1 to length(contourlist) do
    box ← boundingBoxOfContour
    if box.width > boxratio · box.height then
        potentialboxes.push(box)
    end if
end for

prevscore ← −∞
boxToReturn ← 0

for K from 1 to length(potentialboxes) do
    score ← DetermineScore(potentialboxes[K])
    if score ≥ prevscore then
        boxToReturn ← potentialboxes[K]
        prevscore ← score
    end if
end for

return boxToReturn

Figure 3.3: The algorithm for determining boundary and score.

the image feature. This agent is also given a reward (a small increase to score) to help stabilize agent selection. If one agent does not clearly dominate, feature detection can produce unstable results since the agents will have differing positions in the parameter space.

The state of the agent-system is preserved between calls, though it can be reset if necessary. This allows a swarm to use previously-determined parameters for operation on subsequent frames, which is the key to the speed of the system; performing an exhaustive search on every frame would be computationally intractible for a real-time system. This is important for real-time operation since it allows the vision system to continue operating on data (albeit a bit more slowly) and continue allowing the swarms to adapt. The system continues to read and process frames without negatively impacting the framerate.
FUNCTION score(image, pReg, eReg, minW, maxW, minH, maxH)

shapeFlag ← 0

posFlag ← 0

if pReg.width > minW · eReg.width and pReg.width < maxW · eReg.width then
    shapeFlag ← shapeFlag + \frac{pReg.width}{maxW · eReg.width}
else
    shapeFlag ← shapeFlag − 1
end if

if pReg.height > minH · eReg.height and pReg.height < maxH · eReg.height then
    shapeFlag ← shapeFlag + \frac{pReg.height}{maxH · eReg.height}
else
    shapeFlag ← shapeFlag − 1
end if

shapeFlag ← \frac{shapeFlag}{2}

posFlag ← 1 − \frac{2 · pReg.y + pReg.height}{eReg.height}

if posFlag > 0 then
    posFlag ← 1 − posFlag
end if

return posScale · posFlag + shapeScale · shapeFlag

Figure 3.4: The algorithm for determining score.

The DetermineBoundAndScore algorithm, shown in Figure 3.3, implements the particular method for this feature-detection system. Canny edge detection is performed to determine the edges, using the coordinates in parameter space to determine the upper and lower thresholds of the edge detector. From the detected edges, the contours (edge points associated into connected curves) defining the feature can be found and bounded. Each bounding box is analyzed, comparing the size of the bounding box against the size of the entire eye region. The highest-scoring box is returned with its score. The boxratio (the width/height ratio) is used to eliminate the detection of sideburns which, like eyebrows, tend to contrast against skin. To remove the detection of sideburns, the boxratio is tuned to eliminate boxes which are much taller than they are wide.

For the feature evaluation, shown in Figure 3.4, the box that has been found is com-
Figure 3.5. **Top row:** full lighting, **Middle row:** dimmed lighting, **Bottom row:** ambient lighting only.
pared against the maximum box. Because the determined face can be at any distance, the system is stronger when performing comparisons to the face size, instead of fixed pixel dimensions. The values \( \text{minW}, \text{maxW}, \text{minH} \) and \( \text{maxH} \) (minimum and maximum width and height) scale the maximum eye region. In this case, features are rewarded for having dimensions within a certain size range, which is relative to the size of the maximum box.

By this method, the position is rewarded based on the location of its median. If the median of the detected box is above the median of the eye region, the reward is from \([0,1]\), based on distance from the top (a reward of zero at the top, ranging to a reward of 1 at the center). In the bottom half, however, the reward ranges from \([0,-1]\), with zero at the center, down to -1 at the bottom.

This is a somewhat unusual reward system, though not without meaning. In the bottom half of the region, a box bounding the eye tends to meet the geometric criteria, so the score needs to be decreased for falling below the height-wise median. In the top half of the region, however, a shadow above the eyebrow (on the eyebrow ridge) should not be rewarded over the eyebrow itself.

The values \( \text{posScale} \) and \( \text{shapeScale} \) are determined experimentally, to scale the contribution of each factor to the final score.

### 3.2 Experiments and Results

To test the system, two subjects with differing facial attributes were used. One had dark skin and prominent facial features (for the purposes of this example, this means thick eyebrows). The other 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 recording, the light was switched off. Afterwards, the location of each eyebrow in all pictures was determined for the purpose of comparison against the features found by the swarm system.

These were important in simulating real-world scenarios for several reasons. First, it was important to test on subjects with different features which would affect the quality of the feature detection. Second, to test the robustness of the feature detection, lighting conditions had to vary during the experiment to replicate the uncertain lighting conditions the robot may encounter. While it is not likely that the robot would be in a situation where lights would suddenly turn on and off, it helped to illustrate the case of changing light, and adaptation to such.

To stress the accuracy of the feature detector, the features had to move. This is important for using the change in a feature over subsequent frames to ascertain information about the person. For example, eyebrow movement can signal emotions such as surprise or anger; changes in the mouth region can be used for determining a smile or frown, or to determine whether or not a person is talking. In order for a feature detector to be useful for emotion recognition (or action code detection), it has to successfully track a feature over time. The movement of features also demonstrated that even though the features are mobile, the same parameters can be used to consistently track the feature.

Tracking these motions is important especially in affective systems, for interpreting emotional cues and determining appropriate responses. A wide number of researchers, such as Lien and Cohn [21, 5], Yang, et al [44] and others have studied methods of emotion recognition by using Ekman’s [7] facial action codes and mapping combinations
of these to different emotions.

Two feature-detection systems were applied to each subject gallery, composed of 98 images of the dark-skinned subject and 93 of the fair-skinned subject. The first used the DetermineBoundAndScore algorithm for a fixed value (by submitting a single, immobile agent with a pre-determined location to the algorithm). The second used the agent system, replacing the lowest-performing 80% of agents on each iteration. Other parameters were chosen to optimize the accuracy of the detector against its speed to ensure real-time performance for the vision system: \( \text{length}(\text{agentList}) = 20, n\text{Iters} = \{\text{minIters}, 30\}, \text{minIters} = 1, \text{maxFC} = 5, \text{bottomFC} = 5, \text{thresh} = 0.7, \text{replaceNum} = 0.8 \cdot \text{length}(\text{agentList}), \text{boxRatio} = 0.8, \text{minWidth} = 0.05, \text{maxWidth} = 0.65, \text{minHeight} = 0.15, \text{maxHeight} = 0.4, \text{posScale} = 0.3, \text{shapeScale} = 0.7, \text{replaceNum} = 0.8 \cdot \text{length}(\text{agentList}) \).

- The number of agents \( \text{length}(\text{agentList}) \) and number of iterations \( n\text{Iters} \) performed determine the speed of the algorithm, and the swarm’s ability to explore the space. The maximum value of \( n\text{Iters} \) is the number of iterations the swarms will spend analyzing a frame when the optimal value is originally sought. For this implementation, it was determined that a small number of agents exploring the space for a small number of iterations can determine sufficient feature-tracking parameters. The value of \( \text{minIters} \) is the number of iterations performed over \( \text{length}(\text{agentList}) \) on each image when the values are being used; this value must be small for real-time operation.

- The values \( \text{maxFC} \) and \( \text{bottomFC} \) were defined to denote the number of consecutive frames to spend originally seeking the optimal, and the number of frames for which error (a returned score below \( \text{thresh} \) was allowed. When the number of erroneous frames exceeded \( \text{bottomFC} \), the swarm was re-initialized and the optimum re-sought.

- Inside the agent system, the replacement value was set at 80%, leaving the top 6 agents untouched on every iteration. To prevent these from completely dominating the other agents (due to their higher cumulative values), the cumulative values decrease by \( \frac{1}{3} \) on every iteration.

- The values of the ratios between the eyebrow size and the eye region size were determined experimentally; the values have to be restricted in some way to prevent the bounding box from expanding to cover the entire region. Larger boxes tend to be superior to smaller ones, since smaller boxes mean that the edge detection values are too high, and that the eyebrow has been eroded. However, very low threshold values will find contours over the entire area, which would result in the entire area being found as a feature.
• positionScale and shapeScale were decidedly weighed to emphasize shape more than position. Because eyebrow position in the image could vary so easily (head tilting, expression changes), it could not be the sole determinant. On the other hand, it needed enough weight to prevent an eye (with its similar geometry) from being found as an acceptable eyebrow.

Results can be seen in Figure 3.5. The top three rows illustrate the fair-skinned subject under changing lighting 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 bottom three rows are the dark-skinned subject under the same conditions. The first column is the original image, the second is that image after processing with the swarm-based system, and the third is the first image after being processed by a static detector using the same evaluation system as the swarm-based detector. The white box represents the actual eyebrow (as annotated by hand), and the black line is the eyebrow approximation detected by the swarm system.

The score was calculated for all possible agents on each image; this produced the surface the swarm agents explored on each image in the gallery. For an individual, this surface can change for two reasons: lighting, and motion. Lighting affects the surface because the peaks will shift, depending on the amount of light that is added/removed and the physical properties of the subjects. Motion will have a much more subtle effect. The results earn their scores based on their shapes and positions; as these change, the scores will change as well.

To evaluate the performance of each system, the metric defined was $d_{sum} = \sum_{k=1}^{n} (t_k + b_k)$, where $t_k$ is the distance from the top-left point of the determined eyebrow to the top-left 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. The sum is taken over the gallery, which contains $n$ images. By this metric, a lower $d_{sum}$ is desired because it represents a small difference from the actual eyebrow.
The static detector had a \( d_{sum} \) of 977 for the left eye and 2013 for the right. By comparison, the dynamic detector had a \( d_{sum} \) of 749, and 1321 for the right eye. The difference between the two detectors is significant \( (t = 4.411, df = 182.937, p < .0001) \).

3.3 Discussion

The score was calculated for all possible agents on each image. This produced the surface the swarm agents explored for each image in the gallery. For an individual, this surface can change for two reasons: lighting, and motion. Lighting affects this feature surface because the peaks will shift, depending on the amount of light that is added/removed and the physical properties of the subjects. Motion will have a much more subtle effect, due to the evaluation function. The results earn their scores based on their shapes and positions; as these change, the scores will change slightly as well.

It is apparent in Figure 3.6 that this search space is not completely continuous, but has several discontinuous jumps throughout the space. This is a side-effect of trying to make discrete evaluations continuous; evaluations are discrete in this case since, for example, the dimensions of this eyebrow have certain size ranges, and anything inside this size range will be equally acceptable. Similarly, the sizes are integral because they are measured in pixels. In order to provide gradients for the swarms to follow, this had to be “smoothed”. The result is a discrete approximation of a curve.

Because lighting changes will alter the parameter space, the system is augmented by the randomly-located swarms, located by the black points in Figure 3.6. Over the course of only a few frames, the shape of the graph changes dramatically, as the domain of the peaks decreases with the amount of light (as the difference between pixels will decrease with less lighting, the necessary threshold to separate them will decrease). The peak in the first frame, over the course of only a few frames (about 60-130 for the low threshold), became a low plateau. What was formerly a low plateau in the first frame (about 20-40
Figure 3.6: **Top:** full lighting, **Center:** dimmed lighting, **Bottom:** ambient lighting only.
for the low threshold) became the peak.

The swarms in the region with a low “lower threshold”, which are lower-ranked in the system when light is on, take priority almost immediately when the light is turned off. The result is that features are tracked consistently throughout the transition. In a traditional swarm system, however, there is no mechanism that will allow the swarms to easily move out of the valley they suddenly find themselves in. While these agents do have velocities and inertia to carry them through the space, they will lack the well-performing agents that can pull them towards a superior region.

3.4 Summary

While swarms are suitable for exploring multidimensional space, they were not suited to the demands of seeking peaks in a dynamic space in real time. For the problem of real-time feature detection and tracking, it is important to have a system that can quickly adapt to changes in the environment. Static methods are insufficient due to time complexities, but the evolving swarms provide adaptability; while the best agents converge on the maximum point in the parameter space, randomly-placed agents throughout the parameter space will rapidly respond to a sudden field change in their favor, resulting in the seamless track of a feature over the environmental change. Biodiversity increases the performance of the system in two ways: the convergence on a high feature is helpful in the case of minor changes to the shape of the field, while the scattered agents quickly take advantage of sudden peaks caused by major changes in the field.
CHAPTER 4

A HIERARCHICAL EXTENSION FOR FEATURE DETECTION

Chapter 3 discussed the utility of a single-level swarm system. On that level, a face was given and partitioned; each of the partitions was analyzed independently, with the highest-scoring result from each system submitted as the detected feature.

Using this method, the features are always known to be in the correct area, even if they are not on the precise location. They are known to be searching the correct area of a face, so a reinforcement step to confirm their relative positions would be largely redundant.

However, the face detection function is computationally expensive when compared to the feature detection system; the feature detection allows for a framerate of about 7.5 frames/second with the face detector, compared to about 24.6 frames/second without it. Reducing this expense would free up more resources for either the vision system, or another system running on the same processor. By decoupling the search regions from the face tracker, the individual features can be tracked in the image without the overhead of searching for a face on every frame.

The tradeoff is that the search regions become mobile. And when this happens, false positives can cause a search region to move from the actual position of the feature, to a point where the feature is no longer within the search region. The detected features, then, need to be restricted in their movement. This restriction comes from two sources. The first is the distance from the previously-detected location: given a “sufficient” framerate, the feature should not move far from its previously-detected location, between frames.
The second is the spatial relationship to other features. The utility of this evaluation will be covered in depth in this chapter; by defining the spatial relationships between the detected features, the agents which generated those features can be rewarded in turn for not only providing good features, but for providing features which work well in concert with other features. This level of abstraction essentially links the otherwise independent swarm systems by relating their results to each other.

To this end, the hierarchical swarm system defined in 2.3 is used for feature detection and tracking. This chapter demonstrates how the proposed swarm hierarchy can be used to provide feature tracking without needing the face detection on every frame.

4.1 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. The lower level \( l = 0 \) will be a two-dimensional space providing parameters to a Canny edge detector \([4]\), 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

\[
F : RE_k^0(s_1, ..., s_m, x_{k,1}^0, img) \\
F \in \hat{P}_k^0 : \{ (x,y,w,h) | x,y,w,h \in \mathbb{Z} \} \\
Img : \{ (x,y,r,g,b) | x,y \in \mathbb{Z}, r,g,b \in \mathbb{R} \geq 0 \} \\
RE_k^0 : \mathcal{P}(Img) \times \mathcal{R} \times \mathcal{R} \times ... \times \mathcal{R} \mapsto \mathcal{P}(\hat{P}_k^0)
\]

where \( img \) is the image to be processed, \( s_1, ..., s_m \) are static parameters of the system, and
FUNCTION scoreGeom(leftBrow, rightBrow, mouth)
face.leftBrow ← leftBrow
face.rightBrow ← rightBrow
face.mouth ← mouth
scoreFeat ← leScale · leftBrow.score + reScale · rightBrow.score + mScale · mouth.score
face.score ← scaleFeat · scoreFeat
face.nextMode ← RECOVER
lCentroid.x ← leftBrow.x + \frac{1}{2} leftBrow.width
lCentroid.y ← leftBrow.y + \frac{1}{2} leftBrow.height
rCentroid.x ← rightBrow.x + \frac{1}{2} rightBrow.width
rCentroid.y ← rightBrow.y + \frac{1}{2} rightBrow.height
mCentroid.x ← mouth.x + \frac{1}{2} mouth.width
mCentroid.y ← mouth.y + \frac{1}{2} mouth.height
lPolar ← Polar(lCentroid, mCentroid)
rPolar ← Polar(rCentroid, mCentroid)
if leftBrow.width > 0 and rightBrow.width > 0 and mouth.width > 0 then
  geomScore ← evalThreeFeats(lPolar, rPolar, lCentroid, rCentroid)
  face.score ← face.score + geomScale · geomScore
  if geomScore > geomThresh then
    face.nextMode ← FEATURES
  end if
else
  face.score ← face.score − 2
end if
return face

Figure 4.1: The algorithm for determining what features are present, and the functions to call based on this determination.
FUNCTION evalThreeFeats(lPolar, rPolar, lCentroid, rCentroid, mCentroid)
headTilt ← rCentroid.y - lCentroid.y
            lCentroid.x - rCentroid.x
headAngle ← tan⁻¹(headTilt)
lTheta ← lPolar.theta - headAngle
rTheta ← rPolar.theta - headAngle
if lTheta < 0 and rTheta > 0 then
    maxAngle ← max(−lTheta, rTheta)
    maxRadius ← max(lPolar.r, rPolar.r)
    angleS ← \frac{|lTheta + rTheta|}{maxAngle}
    if rTheta < angleTresh then
        angleS ← angleScore + 0.25
    end if
    if lTheta > −angleTresh then
        angleS ← angleScore + 0.25
    end if
    diffS ← abs(lPolar.r - rPolar.r)
    distS ← \frac{lPolar.r + rPolar.r}{maxRadius} + \frac{oldR - 2500}{2500}
    if distScore > 1.0 then
        distS ← max(2.0 − diffScore, −1.0)
    end if
    moveS ← moved(lCentroid) + moved(rCentroid) + moved(mCentroid)
    score ← scaleA · angleS + scaleDiff · diffS + scaleDist · distS + scaleM · moveS
else
    score ← −2.0
end if
return score

Figure 4.2: The algorithm for scoring the relation between three detected features.
\( x_{k,1}^0 \) 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 \( E_k^0: \mathcal{P}(\hat{\beta}_k^0) \mapsto \mathbb{R} \) to a feature that has been extracted. The space \( \hat{\beta}_k^0 \) 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.

Determining values \( s_1, \ldots, s_m \) such that \( score \) is maximized for \( g \) requires searching an \( m \)-dimensional parameter space. While this can be done manually\(^1\), this is not only tedious but also subject to changes in \( img \).

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 \( score_{Geom} \) function, given in Figure 4.1, determines what features are present and submits combinations to evaluation functions \( E_k^{l+1} \).

While this thesis 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. The polar coordinates for the features are calculated here instead of in the evaluation function, simply because it is conceivable for other evaluation functions to require the points, so this way the calculation is performed only once.

Depending on how well the feature set scores, this function will set one of two options for \( nextMode \), which will signal which method will be used to process the next frame. If the score does not exceed \( geomThresh \), \( nextMode \leftarrow \text{RECOVER} \). When this mode is activated, the system performs a face search over the image using a method such as the previously-used OpenCV Haar face detector, a system such as using 3D information to localize a head [43], multi-modal methods [10, 11, 2]. The features for the next frame are evaluated by their position inside the detected face. Otherwise, \( nextMode \leftarrow \text{REGION} \) and the next frame will be evaluated at the lower-swarm level solely on their shapes.

\(^1\)Certainly, a parameter space may be so large that a “perfect solution” is impossible to pinpoint; however, it is possible to find a solution by trial-and-error that is sufficient for a specific feature-detection task.
Figure 4.3: An illustration of the model used for facial feature geometric comparisons.
The evalThreeFeats function in Figure 4.2 will perform the $E^{l+1}$ evaluation of the three detected features to determine their fitness as a unit. Figure 4.3 illustrates the relationships evaluated for the fitness of the features as a face. 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 $a_2 - a_1$, the difference between the mouth-left eyebrow angle, and the mouth-right eyebrow angle; the perfect score is zero; because $angleS$ is more of a “punishment” than a “reward”, $scaleA < 0$. 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, $diffS = d_2 - d_1$ determines the difference between the two features’ distance to the mouth and scales it by the largest, to yield a score $[0,1]$. Again the perfect score would be a difference of zero, so $scaleDiff < 0$.

The distance from the mouth to the eyebrows, scored by $distS = d_2^2 + d_1^2$, 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 $RE^0_k$ (LocateBestFeature, DetermineBoundAndScore, DetermineScore) are based on those in 3. 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 \texttt{DetermineScore} evaluates the feature.

When using the higher-level swarms, it is possible to use a modified version of the evaluation function \texttt{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.

4.2 Experiments and Results

To test the system, the facial feature detection methods were applied to the gallery used in Section 3.2.

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 \texttt{DetermineBoundAndScore} algorithm for a fixed parameter in $P_k$. 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 top in Fig. 4.4 shows the fair-skinned subject under changing lighting conditions, while the block of 12 images on the bottom 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
Figure 4.4. **Top block:** fair-skinned subject, **Bottom 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
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 fea-
ture 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 point of the actual eyebrow, and \(b_k\) is the dis-
tance 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 \((F(1, 561) = 36.2, p < .001)\), in-
dicating 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 ef-
fect 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.

4.3 Discussion

In the hierarchical swarm system, a parameter space \(P^0_k\) is shaped by lighting and fea-
ture properties, and altered by other systems, through \(E^1_k\). Each agent’s position in the
parameter space will map to \(n \cdot m\) values, where \(n\) and \(m\) are the number of features re-
turned by the other two swarm systems. However, only one of these values will affect the
TABLE 4.1

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).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Static</th>
<th>Swarm</th>
<th>Hierarchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>2608.418</td>
<td>1774.5584</td>
<td>2214.4127</td>
</tr>
<tr>
<td>Right</td>
<td>3929.241</td>
<td>2082.952</td>
<td>2551.309</td>
</tr>
<tr>
<td>Mouth</td>
<td>4356.451</td>
<td>3379.492</td>
<td>2896.113</td>
</tr>
<tr>
<td>Total</td>
<td>10894.11</td>
<td>7237.002</td>
<td>7661.834</td>
</tr>
<tr>
<td>(\mu(\sigma))</td>
<td>57.6(36.7)</td>
<td>38.2(18.4)</td>
<td>40.5(21.3)</td>
</tr>
</tbody>
</table>

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 parameters.
features.

4.4 Summary

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 based on information about the presence of faces (e.g., determined by recognizing voices of people in the vicinity).
CHAPTER 5

IMPLEMENTATION ON A ROBOTIC PLATFORM

This system has been implemented and applied to a computer vision task for the Artificial Intelligence and Robotics Laboratory for the system described in [35]. The robotic platform for the experiment is a Pioneer ActivMedia Peoplebot (P2DXE) with a pan-tilt-zoom camera, a SICK laser range finder, two microphones, two speakers, three sonar rings, and an onboard 850 MHz Pentium III. In addition, it is equipped with two PC laptops with 1.3 GHz and 2.0 GHz Pentium M processors. All three run Linux with a 2.6.x kernel and are connected via an internal wired ethernet; a single wireless interface on the robot enables system access from outside the robot for the purpose of starting and stopping operation. Obstacle detection and avoidance is performed on the onboard computer, while speech recognition and production, action selection, and subject affect recognition are performed on the laptops.

5.1 Person Recognition Goals

The vision system provides information about detected faces using and histogram methods from [45] as well as skin color, color of clothes, and camera angle relative to the robot, which are used in conjunction with information about the distance of an object (as reported by sonar and laser sensors) to identify and track people in a room [36] and determine some of their salient features (such as using triangulation to take the subject’s distance from the robot, and the camera angle to yield the subject’s height [3]) for future

39
Figure 5.1: The robotic platform used for human-robot interaction.
identification. Identification of an individual is useful in human-robot interaction tasks, for convincing a human that the robot “remembers” them. If some set of physical properties can key an individual (say, using their height, their skin color and some average clothing color), the robot could theoretically associate that key with an emotional state (so the robot knows if the subject parted on friendly terms, or left in a fit of rage) as well as conversational state (knowing that the person has two kids and lives in central Ohio, or successfully answering a specific person’s request). This property of remembering a person smooths interaction by making the robot more than a box that responds to what was immediately said; it gives the robot a past with the person.

This system, of course, is not meant to be a robust solution in the general field of biometrics; the approximation of a subject’s height, and the fact that the system uses clothing color to identify a person ensures failure in a general sense. However, in the context of simply remembering a person one day at a conference, a reasonable assumption is that if nothing else, clothing, height and skin color will be enough to identify a person for a brief period of time.

5.2 Feature Extraction

Feature extraction of the eyes and mouth is performed using a swarm-based exploration of a parameter space to find values of a Canny edge detector. These parameters are used in edge detection, and follow an algorithm (such as that in Figure 3.3) that produces the best features within a face region given by the OpenCV Haar cascade. The features are tracked and represented in ways that can be mapped into emotional states; the eyebrows are represented as line segments so their slopes, heights and distances from each other can be easily followed from frame to frame. The mouth cornerpoints are the most important in tracking the shape of the mouth, but the bounding box of the mouth has a general importance as well.
In interactions with multiple people, it is important to be able to distinguish who is talking. Some systems [26] have been developed that use an array of microphones to determine the location of the speaker. Vision can be used for this task in a rather simple manner, if the mouth can be tracked closely: whose mouth is moving? This knowledge, regardless of how it is obtained, will allow attention to be paid to the speaker, and this focus of attention could trigger internal changes in the system such as loading different facts about the speaker for the context of the conversation, or holding a different emotional state (if the person is particularly liked or disliked).

Motion of the tracked features gives insight into emotional changes of the subject, as seen in [6]. For example, a sudden raise in the eyebrows can signal happiness or surprise, whereas downward movement can indicate frustration or confusion. The general shapes and positions of features are maintained in a hash table keyed by face and leg positions, allowing emotional states to be tracked simultaneously for several people.

5.3 Overall Implementation

The vision system, as implemented on the robot, detects and tracks faces. As a person moves across the robot’s field of vision, the camera will pan and tilt to keep the person in frame. A higher-level system is responsible for coordinating leg data (the distance angular location of a set of legs) and a face location, combining these to generate a key to identify an individual. This system is also responsible for storing the data about the person (height, and feature information). As a person is tracked, features are tracked (which can be done using either the method in Chapter 3 or 4), and the new information is sent to the higher-level system, which is responsible for the comparisons to determine a change in emotion.
5.4 Real-World Concerns

While it is possible to use the system in real-time, the processing requirements of vision processing can be demanding. The vision system worked well independently of other systems, but running other compute-intensive processes on the same platform hurt the performance due to the real-time demands of emotion detection. Emotion detection based on facial action points demands a framerate capable of discerning the motions. If an expression is quick enough to occur between frames (a smirk or a raised eyebrow) it won’t be caught and all the benefit that could have been gained from recognizing it will be lost.

However, a system that monopolizes resources is detrimental in a robotic perspective, as well. One problem, as stated, is that the vision system does not have the resources to perform in real-time. Another, however, is that while the vision system is performing, other robotic systems will be waiting. Because the robotic systems are intertwined, the system itself will be delayed in processing not only the visual stimulus from a second ago, but also auditory stimuli, or simple synchronization tasks. The real-time capability of the vision system may need to be diminished (by, say, reducing the framerate) so the entire system can operate in real-time, and coordinate subsystems appropriately. Any one system hoarding resources doesn’t leave the system with the capability to react in real-time to the stimulus.

Another troublesome factor was lighting. Although the swarm system was designed to adapt to different lighting conditions, it was a very basic implementation of the general concept which would benefit from further refinement. As such, feature extraction was less reliable than anticipated, which adversely affected performance in this area.

Future versions saw this implementation expanded upon, to the point where face detection was needed only for the initial detection of a face, and recovery when the detected features were geometrically unrepresentative of a face. This development is key, because
performing face detection on a frame is a computationally expensive process when compared to the feature detection in a region. This decrease in computational expense will allow more processes to run on the laptop in the wild while running feature detection.

5.5 Summary

The proposed feature detection system has been demonstrated to be viable for real-world, real-time tracking of facial features. Future work with respect to this system would be making the vision system less processor intensive, and simply stress-testing the system to see how well it handles different people under dynamic lighting conditions.
6.1 Feature Detection Systems

[45] provides a survey of face detection in images, as well as methods used and approaches taken to localize faces in images:

- top-down: using what we know about faces to determine their presence
- bottom-up: knowing that a face has certain features, and finding those to infer a face
- template: creation of face patterns to describe the face as a whole
- appearance-based/statistical: using training images to generate a statistical model of a face

The paper defines the challenges of face detection as:

- pose: the position of the face due to its relative angle to the camera (e.g. profile vs. frontal)
- presence/absence of structural components: the potential presence of beards, mustaches, glasses, etc. will add to the difficulty of determining facial features because they cause further variance in the prototypical face. Additionally, the different styles of these components will further complicate compensating for their presence
- expression: the expression on a face, of course, affect how that face looks
- occlusion: the obstruction of part of a face by an object, or even another face
- image orientation: the image rotation around the camera’s axis
- imaging conditions: lighting intensity and distribution, as well as camera properties (e.g. lens type/quality, automatic adjustments for light changes) will affect a face.

[32] provides a survey of expression analysis, listing 20 properties of an ideal analyzer, giving three general properties

- able to distinguish all possible expressions (e.g. through FACS combinations)
• bilateral/unilateral face change
• facial expressions with similar facial appearance

[13] works by finding the irises via edge-detection, using that information to calculate the iris circle, and from there finding the cornerpoints of the eyes, using a similar strategy for finding the mouth. However, their analysis is formed on high-resolution face images (512x512), well-lit images and frontal/non-tilted faces.

[14] discusses a method of finding faces, as well as eye and mouth regions, in color images, using YCrCb transformations. The first step is to apply lighting compensation to normalize the color appearance. The image is transformed, using $YC_rC_b$ information to transform to a space so the skin cluster (defining the $C_rC_b$ pairs which are acceptable as skin colors) is luma-dependent, enabling a more reliable detection of skin.

For the feature detection of eyes and mouths, empirical observations of those regions were used (high $C_b$ and low $C_r$ values for the eyes, high $C_r$ and low $C_b$ values for the mouth) were used as a basis for extracting these features, by performing pixel value manipulations to strengthen these components.

To confirm that these detections are within a face, an eye-mouth triangle is scored from the features, using the geometry of the triangle, comparison of the colors/gradient orientations, and detection of a face boundary around the triangle.

The problem with using this method for human interaction tasks is that the recognition process is too slow, taking up to 10 seconds on a frame for an image of dimensions $\sim 150 \times 220$; on a $640 \times 480$ image, times can be up to 40 seconds, with widely-varying speeds in general. The feature recognition is useful only for localizing features; the masking and dilation operations renders the actual detected areas too rough for any sort of detailed analysis$^1$.

$^1$The authors make no claim that the feature detection should be exact; however, given the real-time requirements of an interactive system, it bears mentioning that the time spent for feature detection following the methods used here does not result in precisely-discovered faces.
[46, 42] use fuzzy pattern matching to extract skin-color similarity maps and hair-color similarity maps. From the skin-color similarity map, projection is used to locate the face region, using geometric proportions of the face to determine the bound. Similar projection methods are used to locate the eyebrows, eyes, nose and mouth. The bounding boxes located by the projections are used as the initial positions of the feature snakes.

This method is reliant on the quality of the skin detection algorithm, for determining the area for facial features. Similarly, it will be skin color and lighting dependent due to its use of projections for localizing facial features. As with any edge-detecting system, the quality of the feature isolation will depend on the sharpness of the difference between the skin and the feature.

[21] is an important paper on the detection and tracking of facial action units. The authors propose a system that is sensitive to subtle changes in the face (i.e. the facial actions). They note that the facial actions are not intimately tied to emotions per se, but are analyzed constantly and unconsciously for enhancement of human interactions. Further, the authors assert that humans display Ekman’s prototypical emotions rather infrequently, and that emotion is communicated through more subtle means (change of a couple of features instead of all of them).

The authors use several different modules, evaluating them individually for their fitness in precisely tracking features.

- **Difference images** between successive frames had been shown (citation in paper) to be sufficient to recognize action units in the brow and eye regions. However, a potential problem is that if the pixel-wise correspondence between face images fails to line up, the intensity differences will be noisy.

- **Optical flow** can give more precise information, and studies showed that increased flow was associated with certain AU in the brow and cheek regions. This has been more at the level of the prototypic expressions than the level of action units.

- **Edge detection** is a popular method as well, detecting not only eyebrows, eyelids and mouths, but also naso-labial furrows. In this case, the type of edge detection (horizontal, vertical, diagonal) performed was dependent on the region being searched.

The results are stored as feature vectors and submitted to two classifiers:
• *Discriminant analysis* calculates the a posteriori probabilities of the vector being an action unit $AU_k$

• *hidden Markov model*, modeling the transitions between actions as state transitions over time.

[5] focuses on two action units: eyebrows raising and eyebrows lowering, in the context of spontaneous head motion, and the relation to the facial actions. The system performs 3D head stabilization by painting the face onto a cylindrical model, which allows the head motion to be represented with six degrees of freedom. Reference frames are stored intermittently to prevent error from accumulating. The image is then transformed to a canonical front which maintains a constant size and orientation, despite such changes in the actual face (for example, turning and twisting the head, or scale increase/decrease due to a change in distance from the camera). From this stabilized head, the action unit recognition is performed by using the Lucas-Kanade feature tracking algorithm combined with Gabor wavelets. In addition to tracking the brow actions, the paper explores the connection between the action units and the head motion by using the cylindrical model and comparing its movement to that of the eyebrows. Their paper shows a correlation between the action units, and a corresponding head movement (i.e. an eyebrow raise is often accompanied by a head lowering).

[20] uses a neural network to detect the activation of action units. It uses multistate face component models for the mouth (open/closed/tightly closed), eye (open/closed), eyebrow/cheek (present) and nasolabial furrows (present/absent), which are detected with template matching. The system also seeks transient features—features that are not always present, such as crows feet or wrinkles in the nose root, with edge detection. The system uses the template matching to detect the facial features on two consecutive frames and analyze the motions between frames. It then provides the presence of these motions, as well as the presence/absence of transient features, to a neural network with one hidden layer which is trained to respond to AUs whether isolated or in combination.
The system is certainly interesting in that it can track the action units alone or in combination, instead of taking a higher-level approach and learning the six basic emotions. However, the system is not suitable for real-time, taking up to a second for a pair of images.

Neural networks are used in finding faces and facial features as well, such as in [23, 9]. Because a face is an element in the set of all possible images, neural nets are used to determine the probability of a face being present in an image given subwindows of the image. Similarly, genetic algorithms such as [22, 19] have been used for finding faces and facial features as well.

6.2 Adaptive Systems

[30] uses several PSOs in an image-clustering algorithm, where each agent’s position represents a potential clustering. The parameter space explored here is defined by the median values for each cluster. Determining this mean is normally an iterative process, initializing \( N_c \) means and adjusting those means based on the pixel clusterings (exact adjustment depends on the particular method). This paper uses swarms in an \( N_c \)-dimensional space, where \( N_c \) is the number of clusters. Each swarm agent’s position encodes a mean for every group. The fitness function is defined to simultaneously minimize the distance between a pixel and its cluster mean, and to maximize the distance between clusters.

In this paper, as in our feature detection system, swarms are used to explore a higher-level parameter space to determine parameters for image processing, instead of exploring the image itself for solutions.

[48, 47] utilize ant colonies for the detection of edges and edge features, as well as for image segmentation. For this method of edge detection, ants are allowed to wander into neighboring pixels (with varying probability). This method should be lighting invariant, but since an ant is allocated to every pixel, it is computationally expensive. Additionally,
the calculation would have to be re-run on every frame.

[48] shows ant-colony optimization for edge feature extraction. Each pixel starts with an ant, which then moves over the image for a finite number of iterations. Movement cost has an inverse relationship with grayscale difference between consecutive pixels; as a result, the ants gravitate towards and sway in areas of high grayscale difference, i.e. edges. As a result, the pheromone trails in areas of small difference are rather uniform, while weight values in edge areas are large. This paper also mentions the image segmentation method discussed in [49].

This method probably would be prohibitive in real-time for any significant number of pixels, due to the one-ant-per-pixel requirement.

[49] uses ant-colony optimization for image segmentation. Each pixel has an ant designated to it, and the ant's movement is a function of the difference in grayscale between successive pixels. The value for pheromone updating is the reciprocal of the path cost. As a result, ants will move in the homogeneous regions they start in. The algorithm stops after some number of iterations. The pheromones left by the ants as they move define a perceptual graph; strong connections between pixels define an area where there has been a lot of concentrated ant-motion; weak areas indicate boundaries that were rarely crossed.

An experimentally-determined threshold is used to segment the regions. If the connection weight exceeds the threshold, the connection is kept, otherwise it is not. The result is that pixels in the same region are connected.

If the threshold could be made adaptive, this would have the light-invariance necessary for a real-world feature-detection system. However, the number of agents necessary would be computationally prohibitive for the real-time requirements. Once a feature was found, however, it may be possible to reduce the complexity by eliminating all ants except for the ones in the region, and track their movement.

Another image-based system is described in [29], where swarm agents move across
an image to locate features such as edges and corners. The resulting configuration is submitted to a classification system which categorizes the object.

[15] uses an agent-based search to find faces in an image. Skin-color filter is used, then combined with a search of the input image pyramid. Extracted window is pre-processed, then compared to a face pose classifier to determine if a frame contains a face, and the pose of that face if so.

Feature extraction follows two methods, depending on lighting and pose. Both use Gabor wavelets; one uses Eigenfaces while the other uses elastic graphs (both of these would require some template matching). Features also seem to be more than simple eye/mouth corners, but also points on other less-important face contours (nose, cheek corners, etc) used for person identification.

[41] performs image segmentation in the image space, using specialized agents (scouts and workers) to find color regions for segmentation. Scouts find pixels of similar color (following a Lorentzian distribution around the sought color) and mark them (if the ”interest level” crosses a threshold). The scouts wander randomly in the environment until they find a pixel of interest. That pixel is marked, and then the neighbors of that pixel are explored (in a “flood-fill” manner, 8-connected) until no more pixels of interest are found. At this point, the marker number is increased and the agent wanders randomly until it finds a new pixel of interest. Workers, associated with the same colors as the scouts, consolidate these marked pixels into connected segments.

6.3 Swarm Methodologies

[17] introduces the concept of optimization of nonlinear functions using particle swarms. Particle swarm optimization is born out of the insight that local processes may underlie otherwise unpredictable group dynamics. It introduces the autobiographical memory (the “local best” of an agent) and the group norm (the “global best” of the system) which will
affect an agent’s movement. The goal of the paper was to show the refinement process of the velocity equation of the particle swarm optimization function.

[40] uses swarm agents to search a multi-dimensional space, randomly-placed agents should converge on a solution within a few hundred to a few thousand iterations. Paper also discusses relation to Genetic Programming and Particle Swarm Optimization. In this paper, a new methodology is presented for creating Complex Adaptive Functional Networks based on the swarm social-psychological metaphor.

[38] states that in order to have convergence, the inequality $\omega > \frac{c_1+c_2}{2}$ must be satisfied. This, and [39] propose another particle swarm optimization, the GCPSO, which slightly modifies the velocity calculation of the global best to prevent it from stagnating (because if a particle is the best for a sufficient number of iterations, its velocity will decay to zero). The velocity will “reset” to the global best position, and basically perform a random search within some range $\rho$, depending on the “success” of the agent’s performance.

[24] introduces chaos into a swarm system. After a stopping criterion is met, the search region is reduced (to form a rectangular convex hull around the agents). Some fraction of the agents are reset, and the system repeats. This basically does what our system does with the randomized swarm agents and goes a step further: decreases the search area space (and the number of agents) periodically. Our system, however, uses the randomly-distributed agents in order to detect a rapid change between frames. To reduce our search space would be to remove ranges, which may be needed on the next image (because one iteration of search is done on every frame).

[16] organizes swarms into a hierarchy. Instead of the traditional swarm system, where a globally-best position for all agents and a locally-best position for each agent influences that agent’s movement, an agent’s movement is influenced by its local best, and the agent above it in the hierarchy. The best of all a node’s children trades position
with its parent node. Further, the degree of the tree can change dynamically. The wider
the branching factor, the more children that simultaneously compete for higher positions,
so it may be advantageous to use the different degrees at different stages. As described,
the branching factor decreases over the course of the algorithm so more agents can vie for
the global best simultaneously at first.

6.4 Dynamic Parameter Space Exploration

[33] presented a method using ant colonies to explore dynamic spaces, similar to the
dynamic field of the parameter space (described in the following section) 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 certainly
provides a very thorough search; however, this requires thousands of ants to explore the
search space so that pheromones can accumulate and be tracked before evaporating.

[12] 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.
Although much progress has been made with this system, it opens the door in many different directions. The system walks a line between computer vision and artificial intelligence, with room for improvement and expansion in both.

7.1 Perceptual Improvements

Currently, the system only has one type of bottom-level image-processing system: the Canny edge detector used for edge and contour extraction. While this edge detector is a very powerful one, it is also expensive relative to other types of edge detector (Sobel, for example) due to the operations it performs on the image.

As previously mentioned, some systems use different types of edge detection to provide input to the system; for example, the neural network in [21] use uses vertical and horizontal edge detectors depending on the very specific face region being checked. Instead of detecting all edges and contours at once, it may be possible to detect different kinds of edges using different edge detectors and combining their results at a higher level to yield the same results. Additionally, these different edge types may give additional information based on their shape.

The information can be more than just edge information, however. A significant amount of data is lost by choosing to ignore the color component of the image. Now, this is not completely without merit: by ignoring the color component of the image and
focusing solely on the difference between pixels emphasizes the fact that a feature must stand out from its surroundings. If used, however, the color component can be used as a detection system in itself; once the bound of the feature has been determined, the color can be learned and tracked itself. Depending on the nature of the component, it may provide even further information about the object (for example, finding a red eyebrow implies red hair).

Using color information can also help robustness against differing structural components such as glasses or beards. For example, the edge detection would be challenged by the edges presented by a beard. Using lip color, however, could help to reduce the search space so a mouth can be found. Or perhaps teeth / tongue / inside-of-mouth color can be used to detect an open mouth.

Color can also be sought in parallel to the edges and the results combined to yield sharper feature bounds; for example, the bound detected by an edge-detector and by a color-detector can be compared, and if they are similar then perhaps the feature has a stronger chance of being chosen.

Finally, 3D information could be combined as well. Facial features tend to be a bit more prominent when compared to the skin level, so this height could provide further feedback for the system, perhaps by confirming that a bounded area is indeed a peak. Additionally, a swarm system could be used to explore the 3D space itself to find peaks on the face and confirm their shape using evaluation functions.

7.2 Recognition Improvements

The current vision system also required a lot of attention to detail to build the hierarchy. The feature evaluation space $\hat{P}^l_k$ had to be carefully defined for each level, using common knowledge about each feature to define the reward system.

However, suppose a system could be used to extract its own areas of interest; instead
of pre-defining salient features (in the case of this thesis, the eyebrows and mouth of a human face), if they could somehow be pre-determined when learning an image, then it may be possible to use the relations between these points for an upper-level evaluation system.

[25] proposes the scale-invariant feature tracker (SIFT) system to determine “interesting” points by means of edge and vector processing of an image. Such points could be used to define regions and relations. It seems pointless to deconstruct the SIFT points back into individual analysis regions, related by the swarm. However, if the SIFT points are mapped to a point in the $E_k^{l+1}$ space, and an evaluation is defined, then the SIFT system itself becomes the lower-level of the hierarchy.

This method, then, becomes a statistical learning approach. If the points are viewed as agents, then the system is very similar to [29], where the positions of the agents are used to identify shapes. As the object is presented to the system, the relations between the SIFT points are adjusted (as they were by hand for the system in this thesis).

Using the SIFT points to determine the interesting features automatically has a drawback that must be taken into consideration. The primary disadvantage to using an automatic method would be that the system has no concept of the importance of a feature. The relations between any two feature points may be overemphasized in a situation where they are largely irrelevant and only related as virtues of being on the same object. As a result, false negatives can emerge when a strongly emphasized relation is not met (for example, on an object with movable parts). While mobile components pose a problem for a human operator programming the relations as well, that knowledge would be applied immediately to reduce or eliminate problematic relations.
7.3 Performance Improvements

Performance of the system is tied to the swarm in the number of agents that move through the parameter space, and the search termination condition. Currently, the number of agents is a constant and the search termination is after a set number of cycles.

However, a constant number of agents may not be necessary. If a space has the potential to be particularly feature-filled, additional agents or cycles may be necessary to determine the maxima.

Addition of swarm agents would need to be determined during the search process. If the global maximum bounces wildly between regions, then more swarm agents may be necessary to explore the space in more depth. As the evaluation functions in this thesis were defined, spaces were rather tame: peaks were unique; no other pair of parameters would perform as well as the best, and the field itself tended to converge towards a maximum. This, however, is a very charitable condition for the agents and may not apply in all cases. If a maximum is difficult to converge upon or isolate, more agents may help to determine the solution.

The number of cycles, additionally, should not need to be hardcoded. While a maximum number of cycles remains useful in real-time operations (to set a hard time limit on the parameter search), it is possible that convergence can be met much sooner. If a termination condition based on the proximity of the agents to each other and not the number of cycles is defined, the frame may be processed more quickly with no adverse effect on evaluation performance provided the termination condition can be checked quickly.

The replacement strategy, as well, is rather simple: a strict replacement of the worst-performing agents. However, if the agents have converged then the top agents may all be on the same point. This means even more agents could be replaced and allowed to explore the parameter space, with little effect on the system. Similarly, those agents could be removed altogether and only re-instantiated under certain conditions (for example, when
thoroughly searching the space on a recovery frame), which would allow the system to run a bit faster overall.

7.4 Emotion Recognition

The goal of the feature tracking system is to ultimately be capable of tracking facial features in changing lighting, with the intention of performing emotion recognition.

7.4.1 The Purpose Of Emotion Recognition in Interaction

Emotional feedback is useful for the interaction, allowing the robot to gauge its interaction in several ways. First, the robot would be able to tell how a participant felt about a particular topic and could then pursue it or change it, depending on the goal of the robot. Second, it would be useful to have a correlation between what the robot says and how it expects a person to react, to how the person actually reacts. If a comedian tells a joke and nobody laughs, he registers the disparity and attempts to correct in some way. If a robot could do this, it could tell more than just whether or not its joke fell flat. An indication that the robot responded poorly to a statement may be frustration (or amusement, depending on the charity of the listener), and may have misheard the statement to begin with. This in turn could trigger a request for the speaker to repeat himself, or perhaps the system can determine that the room is simply too loud or that the battery for the microphone is low. So the emotional state of the person not only feeds back into the goals of the robot, but into a diagnostic system as well.

7.4.2 Currently Designed Systems

Some simple steps have been made already with this goal in mind. Most recent developments in emotion recognition come from the observations made by [7] in his Facial

---

1Prototypically, this is meant in an opinion of having the robot put people at ease, for example at a social function. However, it is particularly interesting to imagine a robot that strives to upset and anger people, perhaps after having a long day at work!
Action Coding System (FACS), and are dedicated to tracking action units [21, 20, 5] or something similar [37] instead of learning prototypical emotions [1], or recognizing them in static images.

Our implemented system can track changes in the shape of the features by preserving characteristics of each feature from frame to frame. The cornerpoints of the mouth (uppermost and lowermost points shown in orange; leftmost and rightmost points shown in blue) and the endpoints of the eyebrows are preserved and, on the next frame, compared to the newly-detected positions of these points. By following very basic rules, these motions can be mapped to emotions. For example, the cornerpoints of the mouth moving upward and outward indicates a smile, or the eyebrows suddenly moving upward represents surprise. Figure 7.1 demonstrates the inverse motions; the top three images demonstrate how the shape of the eyebrows change as they go from “raised” to “furrowed” (eyebrows down and closer together). The bottom three images show the side mouth points move from the top of the bounding box to the bottom of the bound as the face goes from “happy” to

Figure 7.1. **Top row:** transition from surprise to neutral, **Bottom row:** transition from smile to neutral.
"sad". As stated previously, discerning these emotions will provide invaluable assistance in interacting with humans.

Because a tracker can be instantiated for every person around the robot (which practically is not more than one or two in the frame, but more can be detected with the laser), this emotional state can be preserved, so that person can be associated with that emotion later on (e.g. remembering that a person is angry). Additionally, the ability to track multiple emotional states simultaneously is an asset; it removes any necessity to forget the emotional state of an individual, and have to re-calculate it later, allowing for emotion tracking above simple facial feature tracking.

Aside from emotion recognition, however, some systems (i.e. [8]) strive to do more than infer emotional states; they use facial feature tracking as a tool for inferring mental states (agreement, concentration, disagreement, interest, thinking, unsure) from these visual clues. Such inference would be as useful, if not moreso, than emotion recognition. While a person’s emotional state would be a useful gauge for feedback (how well a conversation is going, how well a robotic agent is performing, etc), mental states are more useful in an actual conversation (i.e. to determine if someone is interested/bored in a particular topic).

[31] provides a survey of work done in multimodal methods for affect recognition in human-robot interactions. It also notes the following problem areas:

- What is an affective state?
- What kinds of evidence warrant conclusions about affective states?
- How can various kinds of evidence be combined to generate conclusions about affective states?

This paper states that the field of affect recognition has been explored using vision and speech independently, but only a handful (four at the time of the survey) of research had been done using the two methods in concert.
7.4.3 Emotion Recognition Improvements

The basis for a lot of emotion detection is present; the ability to track the eyebrows and mouth corners over time will be valuable for interpretation of facial expressions. However, currently the emotion system is a “digital” system: an emotion is on or off, with no determination of degree.

A future implementation would be able to use the amount of movement to gauge the intensity of the emotion. With stable feature detection, it would be possible to detect a smirk or slightly depressed facial expression; such detections are difficult due to noise, since noise may be mistaken for these subtle movements, or vice-versa. But distinguishing these from the neutral expression would allow the system to take less obvious hints during interaction.

The speed of movement can also be used to interpret the intensity of the emotions. When someone is exceptionally happy, they’ll smile widely, but their smile can only extend to a certain degree. Raising eyebrows suddenly can mean surprise, or more slowly it can be a facial indication that attention is being paid. The sincerity of a smile can be evaluated by how quickly the smile forms. All of this data is available through the movement of the points.

Finally, there is more to life than just “happy” and “sad”. It almost goes without saying that in order to provide more realistic feedback to a robotic system, the vision system ultimately needs to be able to recognize a wider range of emotions, coupled with the improvements above (tracking subtle emotions, using velocity to determine intensity of an emotion) to have truly meaningful affective interactions.

7.4.4 Summary

The ability to track emotional states is an important component to human-robot interactions. The state of the human can be used for gauging the success of the interaction
in several ways: first, in the sense of interaction, depending on the goals the robot has in
interacting with a person. The second is as a feedback system to determine whether or
not the robot is interacting successfully.

This system has the mechanisms for emotion recognition in rudimentary forms, inter-
preting facial motions as emotions, and is capable of simultaneously tracking emotions
of different people (provided they are in the frame simultaneously).

7.5 Conclusion

In this thesis, a case is made for the utility of a real-time vision system. Robotic
applications, as the primary example, demonstrate this utility: tracking a face is important
for interaction due to the clues provided by the face. Faces provide emotional feedback
and other visual interaction cues (such as mouth movement to indicate a speaker). As a
result, it is important to be able to precisely track these features movements across time.

If a face is detected, it is useful to extract the features. However, the success of the
feature extraction may be variable dependent on lighting or other conditions. Therefore,
it is important that a feature detection system be able to quickly adapt to different condi-
tions. By defining evaluations for the detected features, it is possible to explore a param-
eter space shaped by those evaluations. Using swarm mechanisms to explore results in a
gradient-ascending search, which results in finding the superior features.

However, face detection is an expensive operation. Performing this on every frame
is detrimental to the possible framerate of the vision system, which in turn negatively
affects feature tracking because features are likely to move further in fewer frames, which
gives the system the impression that a feature has moved faster than it actually had. A
low framerate is also detrimental because the calculations are essentially outdated by
the time they are complete. By using the positions of the detected features in a level
of reinforcement, it is possible to use these relations to reinforce the presence of a face,
instead of detecting a face on every frame. This will affect feature detection due to an added criteria: the relation of one detected feature to the others. The hierarchical swarm system allows the spatial relationship to be reinforced through several swarm systems, ensuring that though they practice local convergence, they are actually seeking positions that are beneficial to the overall system.
REFERENCES


