CAR-TR-721 DACA76-92-C-0009
CS-TR-3304 June 1994

Human Emotion Recognition from Motion Using a Radial Basis Function Network Architecture

Mark Rosenblum
Yaser Yacoob
Larry S. Davis

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3275

Abstract

In this paper a radial basis function network architecture is developed that learns the correlation between facial feature motion patterns and human emotions. We describe a hierarchical approach which at the highest level identifies emotions, at the mid level determines motions of facial features, and at the low level recovers motion directions. Individual emotion networks were trained to recognize the "smile" and "surprise" emotions. Each network was trained by viewing a set of sequences of one emotion for many subjects. The trained neural network was then tested for retention, extrapolation and rejection ability. Success rates were about 88% for retention, 73% for extrapolation, and 79% for rejection.

The support of the Advanced Research Projects Agency (ARPA Order No. 8459) and the U.S. Army Topographic Engineering Center under Contract DACA76-92-C-0009 is gratefully acknowledged, as is the help of Sandy German in preparing this paper.

1 Introduction

Visual communication plays a central role in human communication and interaction. This paper explores methods by which a computer can recognize visually communicated facial actions—facial expressions. Developing such methods would contribute to human-computer interaction and to other applications, such as low-bandwidth transmission of facial data and face recognition from dynamic imagery.

The study of emotion in human facial expressions was pioneered by Darwin's work [4] and has been extensively studied in psychology during the last thirty years [25]. This research has indicated that at least six emotions are universally associated with distinct facial expressions. Several other emotions, and many combinations of emotions, have been studied but remain unconfirmed as universally distinguishable. The six principle emotions are: happiness, sadness, surprise, fear, anger, and disgust (see Figure 1), and these are the emotions we focus on in this paper.

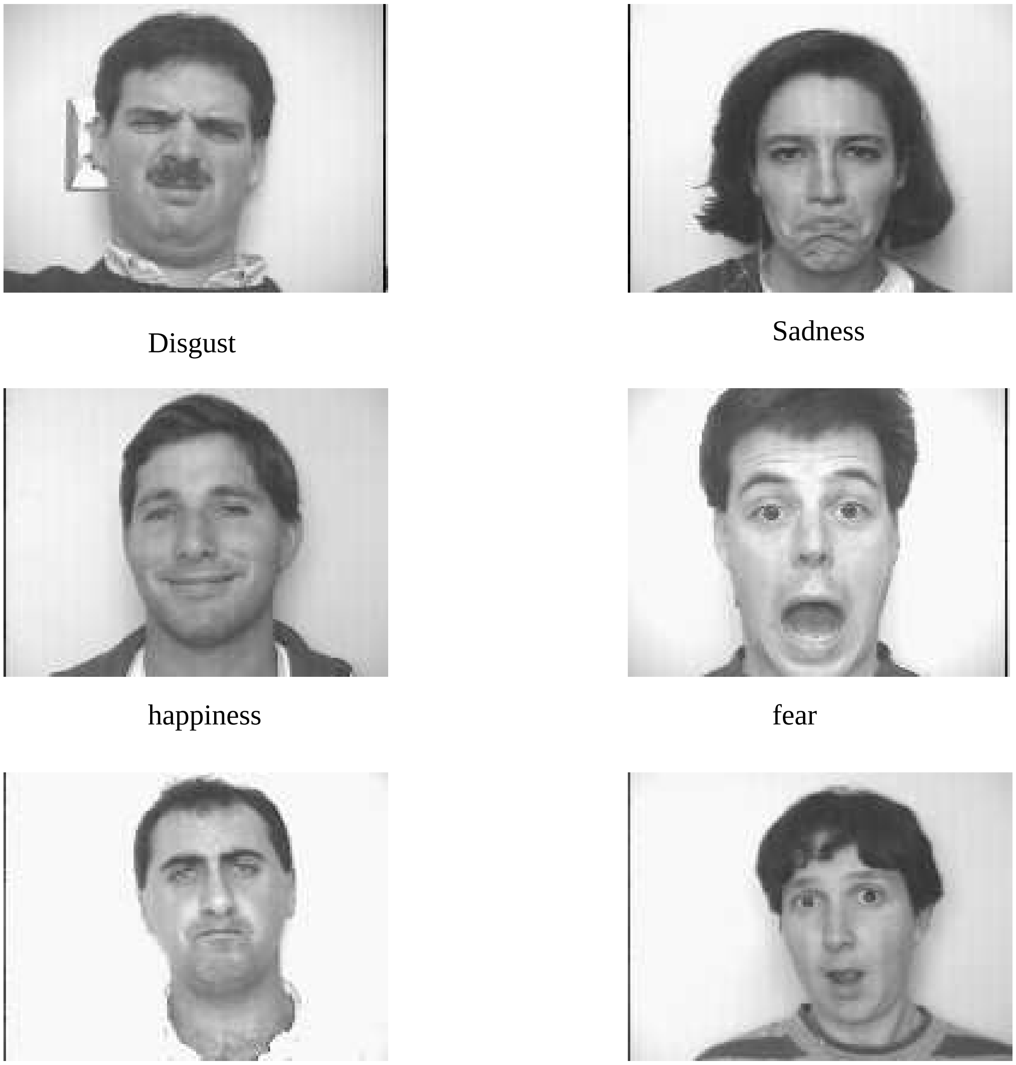
Most psychological research on facial expression has been conducted on "mug-shot" pictures that capture the subject's expression at its peak [25]. These pictures allow subjects to detect the presence of static cues (such as wrinkles) as well as the position and shape of the facial features. Few studies have directly investigated the influence of the motion and deformation of facial features on the interpretation of facial expressions. Bassili [2] suggested that motion in the image of a face would allow emotions to be identified even with minimal information about the spatial arrangement of features. The subjects of his experiments viewed image sequences in which only white dots on the dark surface of the person displaying the emotion were visible. The reported results indicated that facial expressions were more accurately recognized from dynamic images than from a single static image. Whereas all expressions were recognized at above chance levels in dynamic images, only happiness and sadness were recognized at above chance level in static images. Building on the results of Bassili [2] we explore the potential of motion analysis in an autonomous system for recognition of emotions in facial images.

We focus our attention on the appearance of the face from its near frontal image projection, without considering the underlying anatomic and musculature models and actions. In doing so we depart from the current trend in which facial expression analysis is based on recovering muscle actions [15, 16, 21].

Before proceeding, we introduce some terminology. Face region *motion* refers to the changes in images of facial features caused by facial *actions* corresponding to physical feature deformations on the 3-D surface of the face. Our goal is to develop computational methods that use such motions as *cues* for action recovery.

Recent research in computer vision

The problem of recognizing facial expressions has recently attracted attention in the computer vision community [15–17, 21, 23]. With the increase of interest in human-computer interaction and related applications, the need arises to analyze human messages that are communicated through body part gestures and facial expressions.



Disgust

Sadness

happiness

fear

Anger

Surprise

Figure 1: Six expressions expressed by different faces

Yacoob and Davis [23] proposed an approach to analyzing and representing the dynamics of facial expressions from image sequences. This approach is divided into three stages: locating and tracking prominent facial features (i.e., mouth, nose, eyes, and brows), using optical flow at these features to construct a mid-level representation that describes spatio-temporal actions, and applying rules for the classification of this mid-level representation into one of the six universal facial expressions. On a sample of 46 image sequences of 32 subjects displaying a total of 105 emotions, the system achieved a recognition rate of 86% for "smile", 94% for "surprise", 92% for "anger", 86% for "fear", 80% for "sadness", and 92% for "disgust". Blinking detection success rate was 65%.

The work reported here explores the use of a connectionist learning architecture for identifying the motion patterns characteristic of facial expressions. This approach could replace the expert rules developed in [23], and may allow the development of person-specific learning capabilities. Such systems could be used in applications in which interactions are limited to a specific individual.

Matsuno et al. [17] proposed an approach for recognizing facial expressions from static images based on a pre-computed parameterization of facial expressions. Their approach lays a grid over the face and warps it based on the gradient magnitude using a physical model. The amount of warping is represented in a multi-variate vector that is compared to learned vectors of four facial expressions (happiness, sadness, anger, and surprise).

Mase [16] used optical flow computation for recognizing and analyzing facial expressions in both a top-down and bottom-up approach. In both cases, the focus was on computing the motion of facial *muscles* rather than of facial *features*. Four facial expressions were studied: surprise, anger, happiness, and disgust.

The top-down approach assumed that the face's image can be divided into muscle units that correspond to the Action Units (AUs) suggested by Ekman and Friesen [6]. Optical flow is computed within rectangles that include these muscle units, which in turn can be related to facial expression. However, Mase did not report any results on mapping the optical motion results into facial expressions. This approach relies heavily on locating rectangles containing the appropriate muscles—a difficult image analysis problem, since the muscle units correspond to smooth, featureless surfaces of the face. Furthermore, the approach builds on a model that is suitable for synthesizing facial expressions but remains untested in analysis of facial expressions (for more details see [3]).

The bottom-up approach covered the area of the face with evenly divided rectangular regions over which feature vectors derived from an optical flow computation were computed. The feature vectors are defined over a 15-dimensional space that is computed based on the means and variances of the optical flow. The recognition of expressions is based on a k-nearest-neighbor voting rule. The optical flow calculation was averaged within each window to smooth the results over edges. Furthermore, the optical flow was treated on a per-frame basis without considering the time-sequence of frames. The experiments considered the expressions of just one face and the results were compared with the performance of human subjects that were asked to classify the displayed emotions.

Terzopoulos and Waters [21] proposed an approach to synthesis and analysis of facial expressions based on physical modeling of the muscles of the face. They devised a six level representation of the face that consists of: expression level (which includes the six primary expressions); control level (implements a subset of the FACS—Facial Action Coding System—for controlling muscles), muscle level (models the muscles' contraction and expansion as springs), physics level (models the facial tissue's deformations), geometry level (provides a geometric representation of the face as a mesh of polyhedral elements that depend on the curvature of surface), and image level (visualizes the data).

The analysis part assumes that eleven principal contours are initially located manually on the face. These contours are tracked throughout the sequence by applying an image force field that is computed from the gradient of the intensity image. The estimation of the muscle contractions takes place after the contours' reference points were determined. In addition to assuming a frontal view, it was assumed that the projection is orthographic. Once the muscle contractions have been estimated, they were resynthesized onto the 3D range data model of the subject to recreate the muscle contractions. It remains to be determined whether the computation of muscle contractions would be useful for facial expression recognition (the authors did not explore this aspect in [21]).

Li et al. [15] proposed an approach that focuses on analyzing facial images for the purpose of resynthesizing these images. Their approach does not attempt to classify or reason about facial expressions and actions, although some aspects of their work might potentially achieve that. The approach is model-based; it assumes that a 3-D mesh has been placed on the face in the image, and that the depths of points on the face have been recovered. The contribution of the research is an algorithm for recovering the rigid and non-rigid motions of the face from the sequence of images, and reapplying these motions to create an approximation to the initial sequence.

The motion recovery employs a facial modeling approach that uses six AUs to represent possible facial expressions, thereby arriving at an affine non-rigid motion model. Two methods for computing the motion between two images were proposed for small and large motions. In addition, a closed loop feedback architecture was proposed for facial tracking over long image sequences. This architecture assumes motion continuity and employs a prediction-and-correction strategy to handle the motion tracking problem; both linear and adaptive predictors were proposed.

2 Overview of our approach

The following constitute the framework within which our approach for analysis and recognition of facial expressions is developed:

- The face is viewed from a near frontal view throughout the sequence. This allows us to avoid, initially, the increase in ambiguity of expression interpretation as the face moves from the frontal view.
- The overall rigid motion of the head is small between any two consecutive frames.
- The non-rigid motions that are the result of face deformations are spatially bounded, in practice, by an $n \times n$ window between any two consecutive frames. The image sequence is densely sampled in time.
- The subjects wear no eyeglasses.

We chose not to model or analyze facial muscle actions, setting our work apart from [15, 16, 21], as well as not to use models for muscle actions [6]. Instead, we focus on the motions associated with the edges of the mouth, eyes, and eyebrows. These edges allow us to refer to the face features using natural linguistic terminology (for a detailed list of considerations see [24]).

Figure 2 describes the flow of computation of our facial expression system. The system is similar to [23] in the tracking and optical flow computation but differs in the analysis and interpretation of motion patterns. The system is composed of the following components:

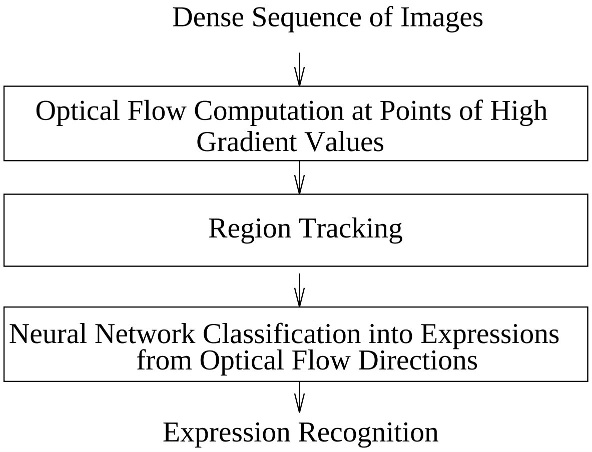

Dense Sequence of Images
Optical Flow Computation at Points of High Gradient Values
Region Tracking
Neural Network Classification into Expressions from Optical Flow Directions
Expression Recognition

Figure 2: The flow of the facial analysis algorithm

- Optical flow computation: Optical flow is computed at the points with high gradient at each frame. Our algorithm for flow computation is based on a correlation approach proposed by Abdel-Mottaleb et al. [1]. It computes subpixel flow assuming that the motion between two consecutive images is bounded within an $n \times n$ window.
- Region tracking: Accurate localization of facial features is both difficult and computationally expensive if performed for each frame. We assume that, for each feature, we can initially compute a rectangular region that encloses it. Such an algorithm has been recently proposed for range data by Yacoob and Davis [22] and a similar algorithm could be developed for intensity images (or stereo images), a problem we are currently working on. Our algorithm tracks these regions through the remainder of the sequence. The tracking is based on the localization of points with high gradient and the optical flow fields computed at these points.
- Connectionist classification: We propose a connectionist approach for learning the facial motion information and relations that are important to the determination of emotion. This system learns using a training set which consists of images from a diverse set of human subjects displaying the same emotion.

In this paper we focus on the connectionist approach to solving this problem.

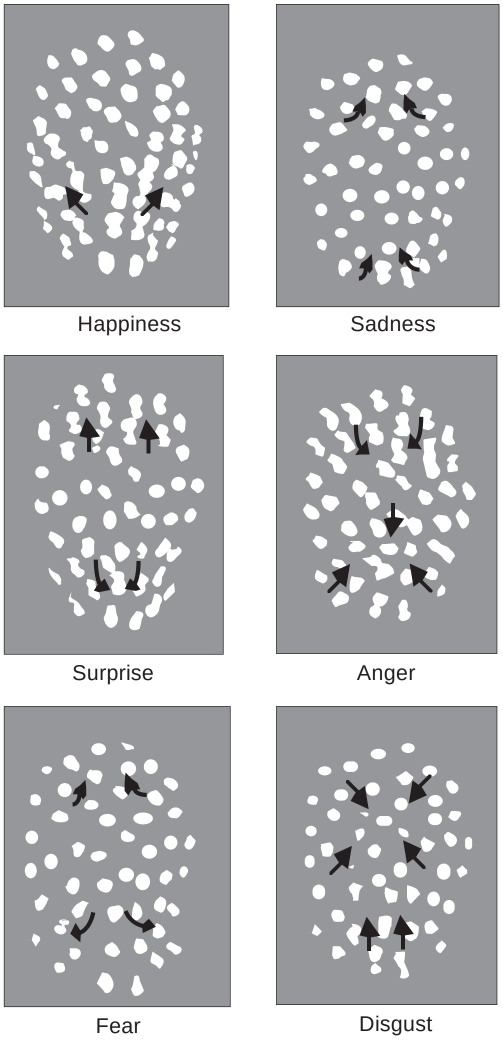
3 Psychological basis for recognizing facial expressions

Table 1 summarizes the results of Ekman and Friesen [5] on the universal cues for recognizing the six principal emotions. These cues describe the peak of each expression and thus they provide a human interpretation of the static appearance of the facial feature. For example, a description such as "brows are raised" means that the viewer's interpretation of the location of the brows relative to other facial features indicates they are not in a neutral state but higher than usual. The viewer uses many cues to deduce such information from the image, among these are: the appearance of wrinkles in certain parts of the face, effect of the hypothesis of a high brow on the shape of the eyes (i.e., state of eyelids), etc. Unfortunately, the performance of humans in arriving at such descriptions is far better than what can be currently achieved by computers if only static images are considered. Some of the linguistic expressions used to describe these cues appear very hard to model computationally—“upper lid is tense, tension or stress in the mouth, lip trembling”, etc. These descriptions seem rather instinctive to humans but are quite difficult to translate into computational procedures.

Table 1: The cues for facial expression as suggested by Ekman and Friesen

Emotion	Observed facial cues
Surprise	brows raised (curved and high) skin below brow stretched horizontal wrinkles across forehead eyelids opened and more of the white of the eye is visible jaw drops open without tension or stretching of the mouth
Fear	brows raised and drawn together forehead wrinkles drawn to the center upper eyelid is raised and lower eyelid is drawn up mouth is open lips are slightly tense or stretched and drawn back
Disgust	upper lip is raised lower lip is raised and pushed up to upper lip or is lowered and slightly protruding nose is wrinkled cheeks are raised lines below the lower lid, lid is pushed up but not tense brows are lowered, lowering the upper lid
Anger	brows lowered and drawn together vertical lines appear between brows lower lid is tense and may or may not be raised upper lid is tense and may or may not be lowered due to brows’ action eyes have a hard stare and may have a bulging appearance lips are either pressed firmly together with corners straight or down, or are open, tensed in a squarish shape nostrils may be dilated (could occur in sadness too) unambiguous only if registered in all three facial areas
Happiness	corners of lips are drawn back and up mouth may or may not be parted with teeth exposed or not a wrinkle runs down from the nose to the outer edge beyond lip corners cheeks are raised lower eyelid shows wrinkles below it, and may be raised but not tense crow’s-feet wrinkles go outward from the outer corners of the eyes
Sadness	inner corners of eyebrows are drawn up skin below the eyebrow is triangulated, with inner corner up upper lid inner corner is raised corners of the lips are drawn or lip is trembling

Figure 3 summarizes the observations of Bassili [2] on motion based cues for facial expressions. Recall that the experiments of Bassili were intended to explore only the role of motion in facial expressions; therefore the face features, texture and complexion were unavailable to the experiment subjects. As illustrated in Figure 3, Bassili identified principal facial motions that provide effective cues to the subjects to recognize facial expressions.



Happiness

Sadness

Surprise

Anger

Fear

Disgust

Figure 3: The cues for facial expression as suggested by Bassili

Bassili's results do not explicitly associate the motion patterns with specific face features or muscles since such information was unavailable to the subjects. For example, a "surprise" motion is recognized by an upward motion in the upper part the face and a downward motion in the lower part of the face. Our work unifies the face features described by Ekman and Friesen and the general motion patterns suggested by Bassili into one spatio-temporal system that allows facial expression recognition from dynamic images.

4 Optical Flow Computation

The approach we use for optical flow computation is one recently proposed by Abdel-Mottaleb et al. [1]. Assume that a pixel's displacement $(\Delta x, \Delta y)$ between frame t and frame $t+1$ is at most n pixels and is expressed in terms of an integer and a fraction part, i.e., $(\Delta x, \Delta y) = (i_x + f_x, i_y + f_y)$. A pixel in frame t, in general, will be a combination of four pixels in frame $t+1$, explicitly written as:

$$I_t(x,y) = (1-|f_x|)(1-|f_y|)I_{t+1}(x,y) + |f_x|(1-|f_y|)I_{t+1}(x,y+sy) + \\ |f_x||f_y|I_{t+1}(x+sx,y+sy) + (1-|f_x|)|f_y|I_{t+1}(x+sx,y) \quad (1)$$

where $sx = 1 + i_x$ if Δx is positive and $sx = (-1 - i_x)$ if Δx is negative, and sy is defined similarly. Figure 4 shows the graphic explanation for Eq. 1.

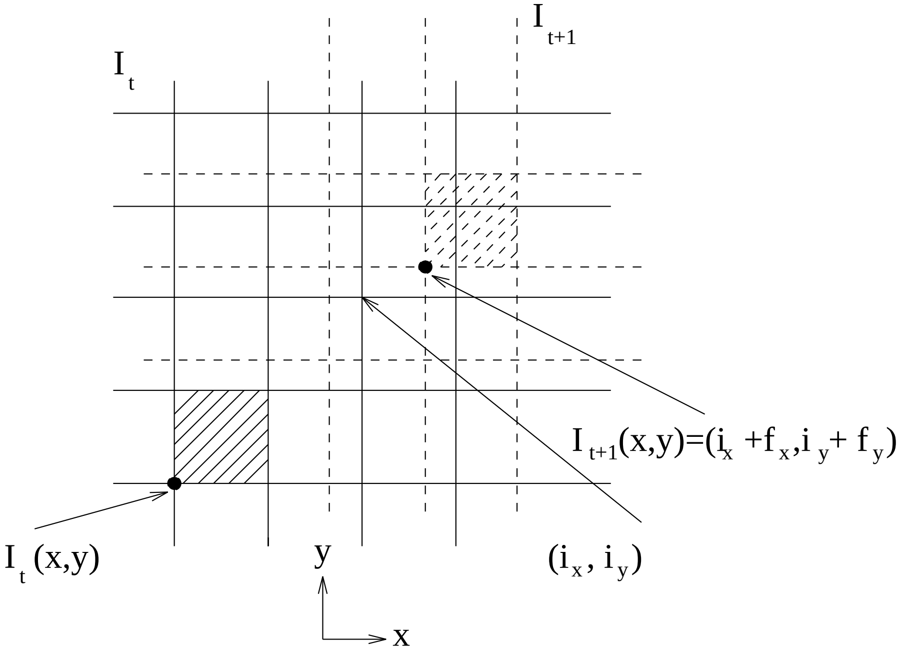


Figure 4: The correlation based optical flow computation

The difference, e, between the right and left sides of Eq. 1 may not be zero due to noise. This difference is to be minimized over all possible motions of the pixel within the defined window. Let $d = (\Delta x, \Delta y)$; then using Bayes' rule we can write

$$p(d|e) = p(e|d)p(d)/p(e) \quad (2)$$

In the case of Gaussian noise with mean zero and standard deviation σ, we have

$$p(e|d) = \frac{1}{\sqrt{2\pi\sigma_d}} \exp^{\frac{e^T e}{2\pi\sigma_d}} \quad (3)$$

where ${\sigma_d}^2 = 2\sigma^2$. Since $P(e)$ is constant with respect to d it can be dropped, and the MAP estimate of the displacement is

$$\arg\max_d p(d|e_w) = \arg\max_d p(e_w|d)p(d) \quad (4)$$

where $p(e_w|d) = \pi^n{}_{j=1}p(e_j|d)$ and e_j is the error term at pixel j in frame i.

5 Tracking face regions

Yuille et al. [26] used deformable templates and an energy minimization approach to localize and approximate the eyes and mouth in intensity face images. This approach requires an accurate initial estimate of the location of the features, and might not be easily applied to facial features such as the eye when it is being closed or opened, since the deformations of the template could become quite complex.

Yacoob and Davis [22] present an approach for qualitatively localizing and labeling natural facial components from range data. They used a multi-stage diffusion process that classifies range points into relative convexities and concavities. The convexities are isolated and contextual reasoning is employed to determine possible labels for each component and an overall consistent interpretation of those labels. Their approach does not quantitatively recover the facial features but it designates rectangles that enclose the features.

The only dynamic tracking algorithm for facial features we are aware of is that of Terzopoulos and Waters [21]. Eleven contours were manually located in the first image of the sequence and then tracked using deformable contour models that are attracted by the local minima of the intensity image. The performance of the tracking algorithm depends on the image gradients since the gradient is used as an external attractive force on the deformable contours. In the experiments reported there was a requirement to enhance the edges in each tracked contour, accomplished by having the subject wear makeup.

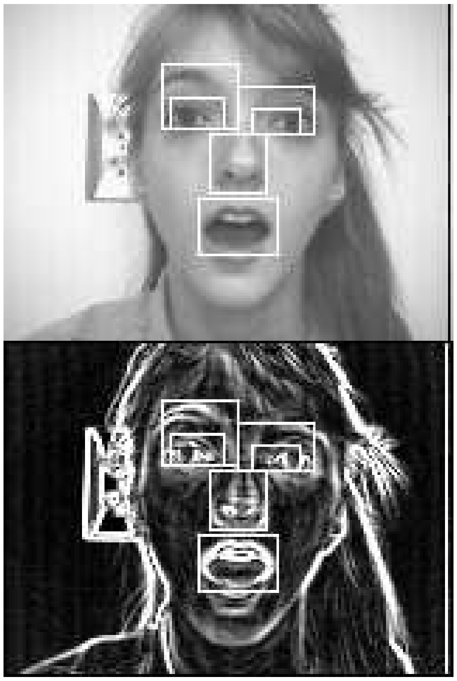


Figure 5: An example of the rectangles surrounding the face regions of interest for the intensity and gradient images

We assume that rectangles enclosing the regions of interest are given for the first frame.

Each rectangle is assumed to enclose just the feature of interest, so the flow computation within the region is not "contaminated" by the motions of other facial features. Figure 5 shows the initial rectangles enclosing the regions of interest of the face for one of our subjects. To simplify the modeling of the eyebrows, we define the rectangles to include the eyes, and then subtract the rectangles of the corresponding eye from the combined rectangle.

Our approach to tracking the face regions is based on computing two sets of parameters at the points with high gradient magnitude within the rectangle that encloses each feature. The following are computed from frame i after placing the rectangles from frame $i-1$ over the image in frame i:

- The centroid $({C^i}_x, {C^i}_y)$ of the points having a high gradient value within each rectangle in frame i.
- The window $W = ({WX^i}_{\text{min}} - 2, {WY^i}_{\text{min}} - 2, {WX^i}_{\text{max}} + 2, {WY^i}_{\text{max}} + 2)$ which encloses those high gradient values and leaves a buffer, two pixels deep, that allows the detection of window expansion during subsequent iterations.

The centroid's location determines the translation of the rectangle from the previous frame. The window W determines the scaling of the rectangle.

The translation and scaling of the rectangles are limited between consecutive images. Limits on scaling are set a priori. For example, when tracking the eyes we assume that the size of the rectangle that encloses the eyes cannot scale more than ± 0.1 from the "neutral" state (i.e., neither wide open in "surprise" nor closed). These scaling bounds were determined, empirically, from our image data sets.

The temporal tracking incorporated the optical flow results in refining the scaling and translations computed by the gradient magnitude change. The statistics of the motion directions within a rectangle are used to verify translation of the rectangles upward and downward (by measuring significant similar optical flow) and to verify scaling of the rectangles (by measuring motions that imply scaling).

This tracking algorithm is admittedly quite simple and has the following shortcomings:

- When the edges of different features move too close together, the rectangles surrounding the features may sometimes be subject to incorrect scaling or translation. We developed heuristic constraints that minimize such artifacts. For example, the rectangle surrounding a feature is not allowed to move too far from the other rectangles over the sequence (which is motivated by the fact that the deformation of the face is constrained by anatomic and muscle factors).
- Due to specularity or change in feature location the gradient at some points may become lower than the threshold. This results in exclusion of these points from the computation of the parameters of the rectangle. For example, for some subjects the lower part of the lower lip tended to produce values below the threshold due to highlights. We are studying ways to overcome this in our current research.

In spite of these shortcomings, the algorithm was quite robust when applied to thousands of images.

6 The Inputs and Outputs of the Neural Network

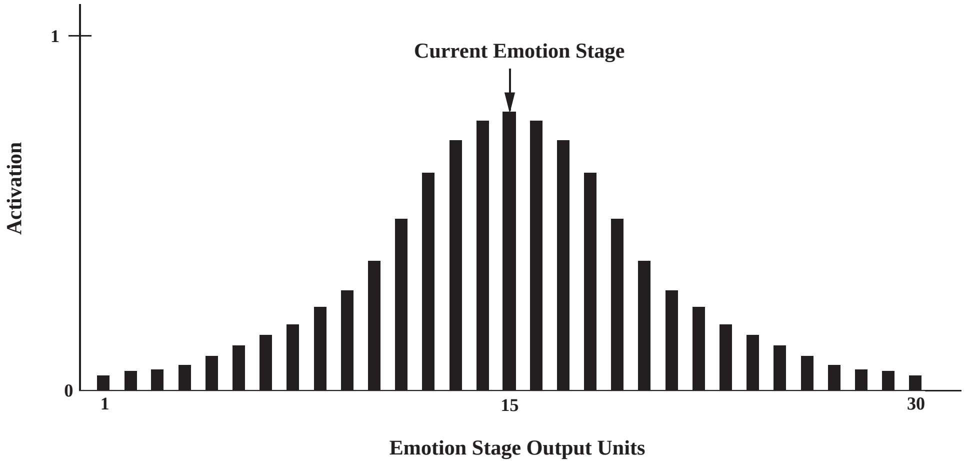
We perform three stages of preprocessing on the input sequence before providing it to the neural network. The first stage generates a sequence of images which represents the instantaneous optical flow of the image sequence. The second stage extracts, using tracking techniques, the facial features from the optical flow sequence (i.e., the right and left eyebrows and the mouth). The third stage of preprocessing performs a log-polar transformation on the feature motion images of the sequence. This transformation compresses the outer extremities of the feature images for the purpose of reducing the effects of size variance. Size variance occurs because of the different subjects' distance from the camera, subject's motion during the image sequences, and the natural diversity in the sizes of subjects' features.

The output nodes of the system could be structured so an output is associated with a particular emotion. In this case we would have six output nodes, one each for happiness, anger, sadness, disgust, fear, and surprise. The activations of the output nodes would represent the network's confidence that the subject experienced the corresponding emotion. This representation, however, does not provide enough spread for the neural network to learn effectively. An intermediate output representation is required to provide this spread. The intermediate output representation could then be connected to the ideal final output layer or to a post-process that determines which emotion occurred based on the intermediate output representation. The intermediate output representation we have chosen represents the stage of an emotion to which an input image of a sequence corresponds. Each output unit is associated with a stage of the emotion sequence. The activation of an intermediate output unit in this representation corresponds to the network's confidence that the emotion of the current subject in the sequence is in the stage corresponding to the particular output unit.

Pomerleau [19] found that when there exists a proximal relation between output units, the supervised learning unit activations should reflect this relation. In our application, an output unit represents a stage of an emotion. The stages are related by a temporal proximal relation, in that one stage occurs before and after others in time. If the current training vector is to reflect that the emotion is currently in stage N, then output unit N should be set with the greatest activation, while output units $N+1$ and $N-1$ should be set with a slightly lower activation and so on until the boundaries of the output vector are reached. Pomerleau used a Gaussian function to set the training activations. We also used a Gaussian, placing its peak on the current stage in the output training vector and setting the output units corresponding to the value of the Gaussian at that position in the vector (see Figure 6).

7 The Network Architecture for Emotion Decomposition

We divide the emotion detection architecture into three layers. The first layer is a decomposition by emotion (see Figure 7), and occurs at the network level (i.e., we train a separate network for each emotion). During training, a network in this layer is only exposed to one emotion for multiple subjects. The second layer of decomposition is at the facial component level. This decomposition is internal to each of the emotion tuned networks. Each emotion network is broken into three subnetworks, where each subnetwork specializes in a particular facial component. A component-tuned subnetwork only uses the portion of the input vector



The Gaussian weighting function applied to desired-output vector corresponding to the fifteenth stage of an emotion sequence

Figure 6: The Gaussian weighted output vector

that corresponds to its component specialization (see Figure 8). The third layer of decomposition is by direction sensitivity, and further decomposes the component subnetworks. These "subsubnetworks" are sensitive to one direction of motion for a specific pre-assigned facial component for a specific emotion. In order to capture all resultant motions, we use the four direction sensitivities of "up", "down", "right", and "left".

The fusion of information from each of the six emotion-tuned networks is performed by a process external to these networks. We developed a heuristic scheme (discussed in a later section) which combines the outputs of all six emotion networks. The fusion of information from the internal subnetworks is done internally in each of the emotion networks through the coupling of these component subnetworks and the output units of the individual emotion network.

8 The Basic Connectionist Building Block

The overall architecture for this application is a hierarchy of neural networks. We chose to use as a building block a modified version of the radial basis function network (RBFN), based on its ability to represent prototypical visual templates from the application domain [20]. In the rest of this section we discuss the basic RBFN architecture, and the enhancements we made to the basic RBFN architecture to handle the temporal relations associated with this problem.

8.1 The General RBFN Architecture

Neural networks are used to approximate multivariate nonlinear functions using sparsely sampled data. The networks are trained using sample data from the function to be approximated, achieving (essentially) a form of function fitting in a multi-dimensional space. If the training set is representative of the function being approximated, the network can

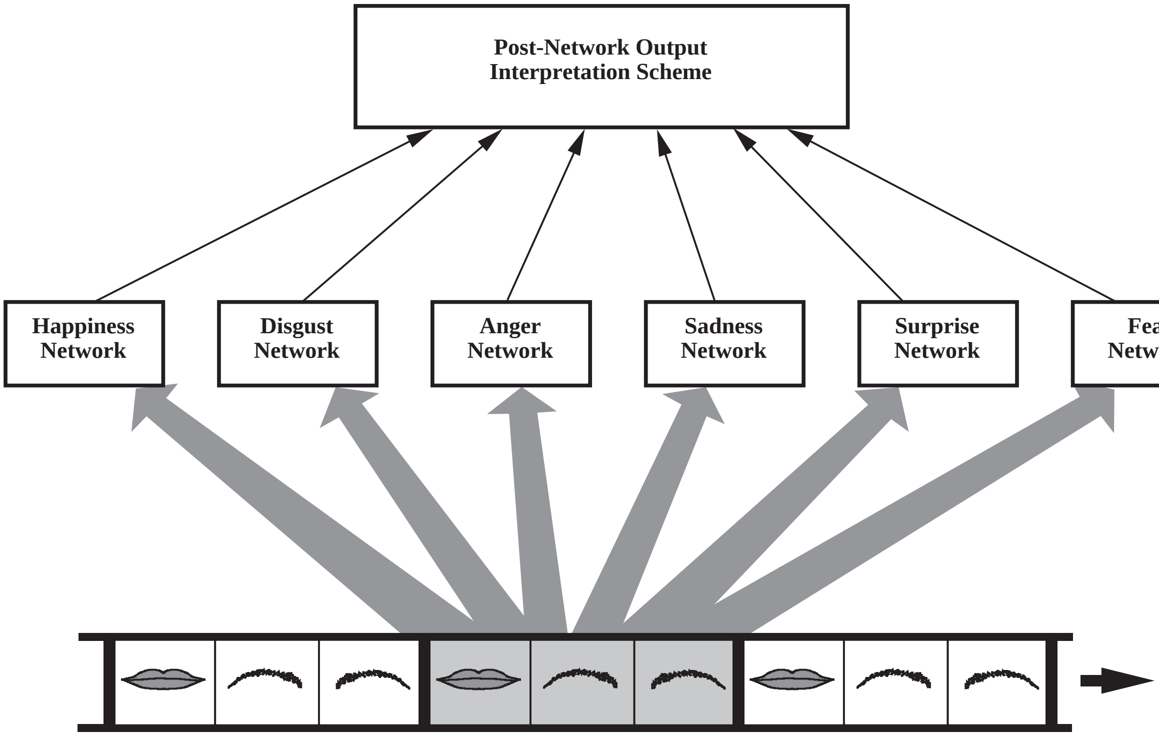


Figure 7: The hierarchy of networks based on emotion decomposition

approximate it with a high degree of accuracy; otherwise the network may learn the function incorrectly. In most cases a well trained network can successfully generalize within the input space subtended by the training set (interpolation), and in some cases, the network can successfully generalize to inputs that fall outside the space subtended by the training set (extrapolation).

The radial basis function network performs a form of template matching in which a template contributes to the generation of the output proportional to the degree that the template matches the input, and by the significance of that template to the output. This relates to the *across-fiber-pattern theory* used to describe sensory coding in humans and animals [7]. Goldstein remarks "This theory states that different qualities are signaled to the brain by the pattern of activity across a large number of neurons." The simple RBFN of Moody and Darken [18] consists of an input layer, a hidden layer, and an output layer (see Figure 9 for the simple RBFN architecture). The input layer consists of a number of units clamped to the input vector. The hidden layer is composed of units which are driven by radial basis activation functions; we will refer to these units as *receptive fields*. The output layer consists of units clamped to the output vector. The input units are fully connected by unit weighted links to the receptive fields in the hidden layer, and the receptive fields are fully connected by weighted links to the output units.

The radial basis function used for the receptive field activation functions can take many forms. A common radial activation function is an N dimensional Gaussian

$$\rho_i(\vec{x}) = \exp\left(-\beta_i \, (\vec{x} - \vec{a}_i)^T(\vec{x} - \vec{a}_i)\right) \tag{5}$$

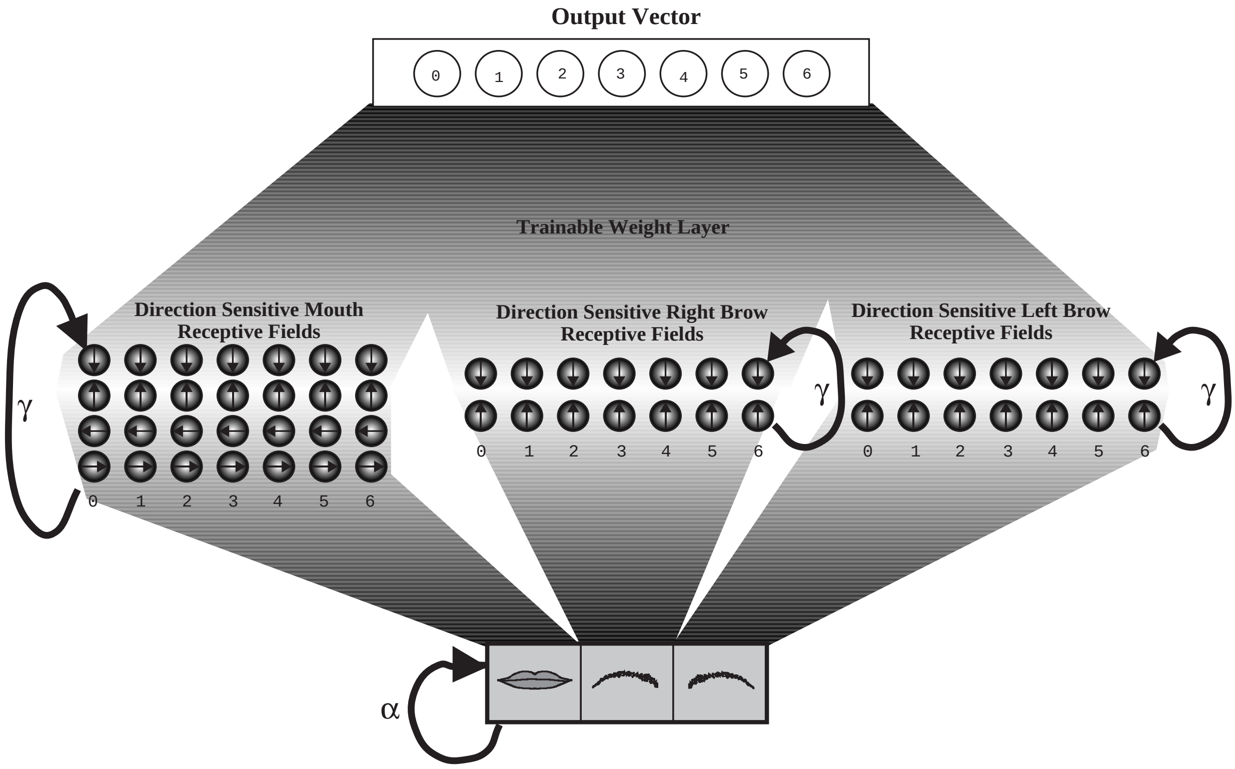

Output Vector
0
1
2
3
4
5
6
Trainable Weight Layer
Direction Sensitive Mouth Receptive Fields
Direction Sensitive Right Brow Receptive Fields
Direction Sensitive Left Brow Receptive Fields
γ
γ
γ
0 1 2 3 4 5 6
0 1 2 3 4 5 6
0 1 2 3 4 5 6
α

Figure 8: An individual emotion tuned network

where $\rho_i(\vec{x})$ is the response of the i^{th} basis function to input vector $\vec{x}$, $\vec{a}_i$ represents the center of the i^{th} basis function, and β_i is the reciprocal of the standard deviation squared. We refer to the standard deviation as the "width" of the basis function. The activation function for an output unit is

$$\phi_i(\vec{x}) = \sum_j f_{ij}\ \rho_j(\vec{x}) \tag{6}$$

where $\phi_i(\vec{x})$ is the activation level for the i^{th} output unit with input vector $\vec{x}$, and f_{ij} is the weight of the link from the j^{th} receptive field to the i^{th} output unit [18].

Several improved versions of the simple RBFN have been made; one of these is called the Connectionist Normalized Linear Spline Network(CNLS) [11]. The CNLS differs from the simple RBFN in two ways:

1. In the CNLS network a linear term has been added to the activation function for an output unit.

2. The responses of the receptive fields have been normalized.

These changes modify equations (5) and (6) to

$$\rho_i(\vec{x}) = \frac{\exp\left(-\beta_i\ (\vec{x}-\vec{a}_i)^T(\vec{x}-\vec{a}_i)\right)}{\sum_j \exp\left(-\beta_j\ (\vec{x}-\vec{a}_j)^T(\vec{x}-\vec{a}_j)\right)} \tag{7}$$

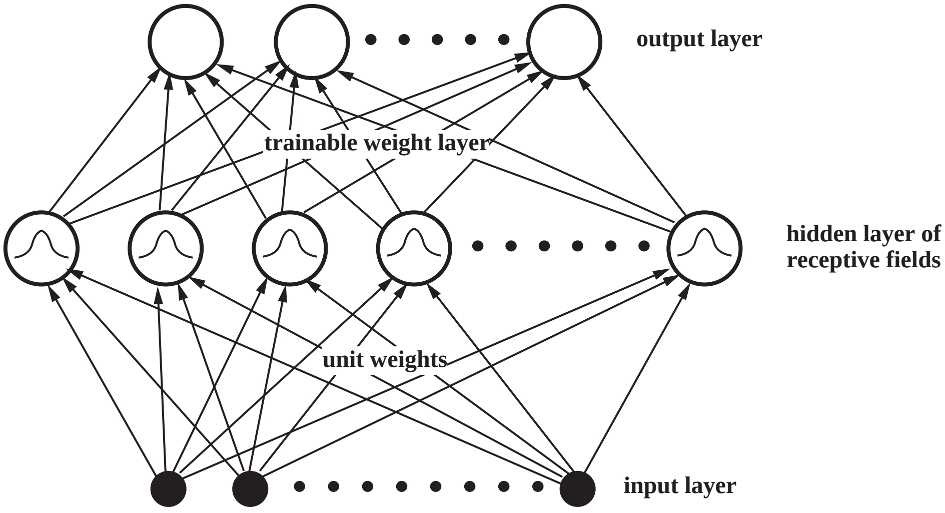


Figure 9: Radial basis function network architecture

$$\phi_i(\vec{x}) = \sum_j (f_{ij} + \vec{d_{ij}} \cdot (\vec{x} - \vec{a}_j))\ \rho_j(\vec{x}), \tag{8}$$

These enhancements to the original RBFN improve the interpolation ability of the network and reduce the amount of training necessary for accurate learning [10, 12–14].

In the RBFN there are three phases of learning. Learning can be performed on: 1) the center locations of the receptive fields in input space, 2) the widths of the receptive fields, and 3) the layer of weights between the hidden layer and the output layer. The first two phases of learning are optional, but the last phase of learning is mandatory. We will explain the algorithm for each phase of learning and how it improves the network's performance.

The training of the receptive field centers is done using task-dependent clustering techniques [8, 18]. An input pattern to the network corresponds to an N-component vector in the N-dimensional input space, and the receptive fields in the hidden layer act as N-dimensional Gaussian response regions in the input space. We would like to distribute the receptive fields of the hidden layer in regions of input space where input patterns occur; this is done using clustering techniques. A simple algorithm for performing this clustering is to expose the network to a set of typical input patterns, and for each input pattern move the closest receptive field closer to that input vector.

$$\frac{\partial}{\partial t}\vec{x}_{\text{closest}}[t] = \eta(\vec{x}_{\text{center}}[t] - \vec{x}_{\text{closest}}[t]) \tag{9}$$

Clustering of receptive fields around nominal input vectors makes more efficient use of the receptive fields in the network, and results in a crisper response from receptive fields near an input vector, and negligible response from receptive fields with little correspondence to the input vector.

The second phase of learning is training on the receptive field widths. Moody and Darken train receptive field widths by minimizing the following objective function with respect to the β's:

$$E = \frac{1}{2}\sum_i \left[\sum_j \rho_i(\vec{a}_j)(\vec{a}_i - \vec{a}_j)^2\beta_i\right]^2 \tag{10}$$

The adjustment of receptive field widths using this technique is only dependent on the locations of receptive field centers and widths, and does not depend directly on input training vectors [18]. Also note that this approach considers the widths along each dimension to be equal, resulting in a Gaussian hyper-sphere response from each receptive field. The training of the receptive field widths optimizes the specialization of receptive fields. A receptive field with a relatively large width in N-space will respond to a large set of input vectors, and a receptive field with a small width will respond to a small set of input vectors.

Training the weight terms between the receptive fields and the output units is the third and mandatory phase of training. This is accomplished by minimizing the sum squared error of the output layer:

$$E = \frac{1}{2}\sum_{i,\mu}(\varsigma_i^\mu - O_i^\mu)^2 \tag{11}$$

Any optimization technique can be used to adjust the weight settings in the network. In our work on RBFN's we have used Newton's Method. The weight change equation for the constant weight term from the j^{th} receptive field to the i^{th} output unit is

$$f_{ij}^{t+1} = f_{ij}^{t} + \eta_f\ (y_i(\vec{x}) - \phi_i(\vec{x}))\frac{\rho_j(\vec{x})\sum_k \rho_k(\vec{x})}{\sum_k \rho_k^2(\vec{x})}. \tag{12}$$

where η_f is the learning rate for the constant weight term. The weight change equation for the linear weight term from the j^{th} receptive field to the i^{th} output unit is

$$\vec{d}_{ij}^{t+1} = \vec{d}_{ij}^{t} + \eta_d\ (y_i(\vec{x}) - \phi_i(\vec{x}))\frac{(\vec{x} - \vec{a}_j)\ \rho_j(\vec{x})\sum_k \rho_k(\vec{x})}{\sum_k (\vec{x} - \vec{a}_k)^T(\vec{x} - \vec{a}_k)\ \rho_k^2(\vec{x})}. \tag{13}$$

where η_d is the learning rate for the linear weight term. Each weight represents the contribution a receptive field makes to the activation level of an output unit. Thus, a large positive weight means that a receptive field plays a large role in an output unit's activation, a small weight indicates that a receptive field plays little role in an output unit's activation, and a large negative weight indicates a receptive field has a large inhibitory effect on an output unit's activation.

An RBF receptive field is a response region in N-dimensional input space, with an N-component center coordinate. The input space can be considered as an image space, since the input units are clamped directly to the values of the pixels of an image retina. Since each coordinate in image space corresponds to a unique image on the image retina, the receptive field centers also correspond to unique images on the image retina, and these function as the application templates. The maximum response of a receptive field occurs when an input image is situated at the same location as the center of the receptive field, and the response degrades in a Gaussian fashion as the Euclidean distance of the input image to the receptive field center increases.

8.2 The Spatio-Temporal Building Blocks

The RBFN is well suited to handle static spatial image information, since the receptive field centers capture prototypical images from the application, which in turn directly capture two-dimensional spatial information. The RBFN architecture, however, is not well suited to handle temporal relations. A significant part of the task of analyzing sequences of images is being able to relate information in consecutive frames and over time so that past information contributes to the current response. For example, there are several emotions where the eyebrows of a subject move downward. In the "surprise" emotion, the eyebrows move downward at the end of the emotion, and in "anger" the eyebrows move downward at the beginning of the emotion. In order to decipher whether the eyebrows are moving downward in the "surprise" or "anger" emotion, it is necessary to determine what happened to the eyebrows before they moved downward. In "surprise", the eyebrows move upward before they move downward, and in "anger", the eyebrows are not moving before they move downward. If this information is incorporated into the current state, the upward motions of the eyebrows for "surprise" and "anger" can be distinguished. Past information will contribute to the current input vector during the training and usage modes, will be used in the placement of the receptive field centers during the center training mode, and will also contribute to the current receptive field activations also during the training and usage modes.

Past information is incorporated into the input vector by using feedback from the previous state of the input vector multiplied by a decay constant. Input units that use self feedback are called "context units" [9]. The activation function for each input unit in our architecture is

$$C_i(t) = \begin{cases} 1 & \text{if } \alpha C_i(t-1) + I_i(t) > 1 \\ \alpha C_i(t-1) + I_i(t) & \text{otherwise} \end{cases} \tag{14}$$

where $C_i(t)$ is the activation of input unit i at time t, $C_i(t-1)$ is the activation of input unit i at time $t-1$, α is the decay constant, and $I_i(t)$ is the current input to unit i at time t. The decay constant is set so that remnants of previous motions linger for a portion of the sequence. If motion occurs for several iterations at the same pixel location in the input image, the input unit activation that corresponds to that pixel location becomes saturated and is set to the maximum activation level of one. The effects of using past information in the current input vector can be seen in Figure 10. For visual simplicity Figure 10 shows the effects of decay for a sequence of images showing a ball moving across a retina. The brightest portions of the image are a result of the most recent frames in the sequence, and the darker ghost images are a result of older remnants from previous frames. In our network, retinas will contain motion images of facial features after a log-polar transformation; decay, however, will have the same effect on these images as the sequence of the ball.

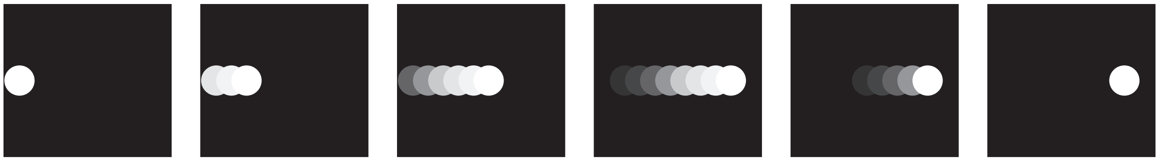


Figure 10: The effect of decay on the input vector

As discussed earlier, the emotion network is divided into three subnetworks, each specializing in a particular facial component and motion direction. Each subnetwork consists of receptive fields tuned to the particular component feature, and the weights fully connecting those receptive fields to the output units. The set of receptive fields corresponding to a particular component and for each motion direction are further tuned to become sensitive to only portions or subsequences of the input sequence. In other words, the component receptive fields become sensitized to stages of the emotion sequence for the component and direction they are assigned to. A receptive field center image or template is set by adding up the motion images for a subsequence of the receptive field's assigned facial component and motion direction sensitivity. Any position in the summed image that has a value greater than zero is set to one. Figure 11 shows how a subsequence is used to set the center of a receptive field for a simple sequence of a ball moving across the retina. This creates a blurring effect of the template image, similar to the blurring effect created by using a slow shutter speed in photography, but with no decay of the "ghost" images. It is important to note that an input vector can never perfectly match a receptive field center template unless the decay constant for calculating the input vector is set to one.

In order to minimize the problem of overloading and under-utilizing receptive fields, we defined a parameter which represented the minimum number of pixels that must be turned on during the image summing stage to set a receptive field center with the summed image. If the number of "on" pixels during the summation crossed over this minimum threshold, no additional images were incorporated into the summed image and a receptive field center was set with the current accumulated image. The result was that portions of the sequence where significant motion occurred were spread over more center templates for higher temporal resolution, and portions of the sequence where little motion occurred were fit into fewer center templates for lower temporal resolution (see Figure 12).

In addition to using past information to set the center positions of the receptive fields, past information is also used to determine the activations of the receptive fields during the training and usage modes. The determination of a receptive field activation is similar to the determination of the activation of an input unit, in that the activation from the previous time step is factored into the activation at the current time step using a decay constant. The activation of a receptive field in our architecture is determined by the following equation:

$$\rho_i(\vec{x}_{t+1}, t+1) = \begin{cases} 1 & \text{if } \gamma\rho_i(\vec{x}_t, t) + V_i(t) > 1 \\ \gamma\rho_i(\vec{x}_t, t) + V_i(t) & \text{otherwise} \end{cases} \tag{15}$$

where γ is the decay constant and

$$V_i(t) = \frac{\exp\left(-\beta_i\ (\vec{x}_{t+1} - \vec{a}_i)^T(\vec{x}_{t+1} - \vec{a}_i)\right)}{\sum_j \exp\left(-\beta_j\ (\vec{x}_{t+1} - \vec{a}_j)^T(\vec{x}_{t+1} - \vec{a}_j)\right)} \tag{16}$$

Like the input vector determination, the receptive field response can become saturated, in which case the activation is set to the maximum value of one.

There is an ambiguity in using a decay constant to determine receptive field activation. The ambiguity can lead to a blurring between temporal and spatial effects. The ambiguity arises because a receptive field response can arrive at a particular activation through temporal decay or by spatial similarity. The temporal decay can give the faulty appearance of

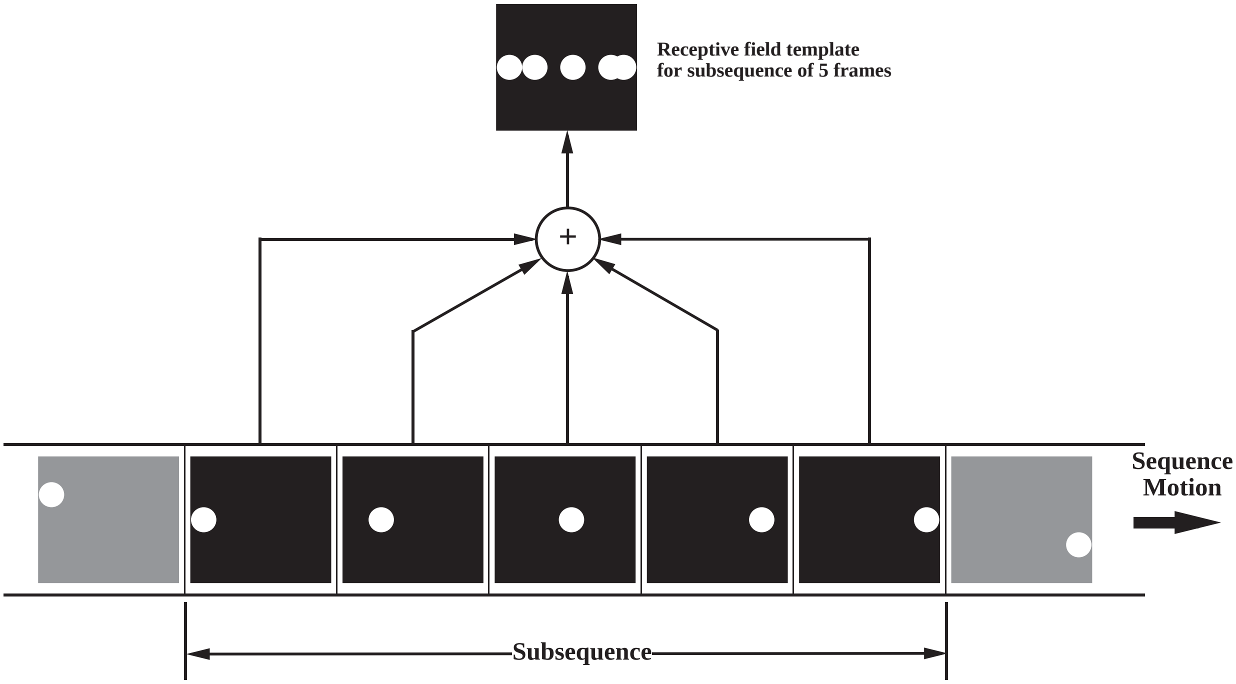


Figure 11: The approach to setting the receptive field centers from subsequences

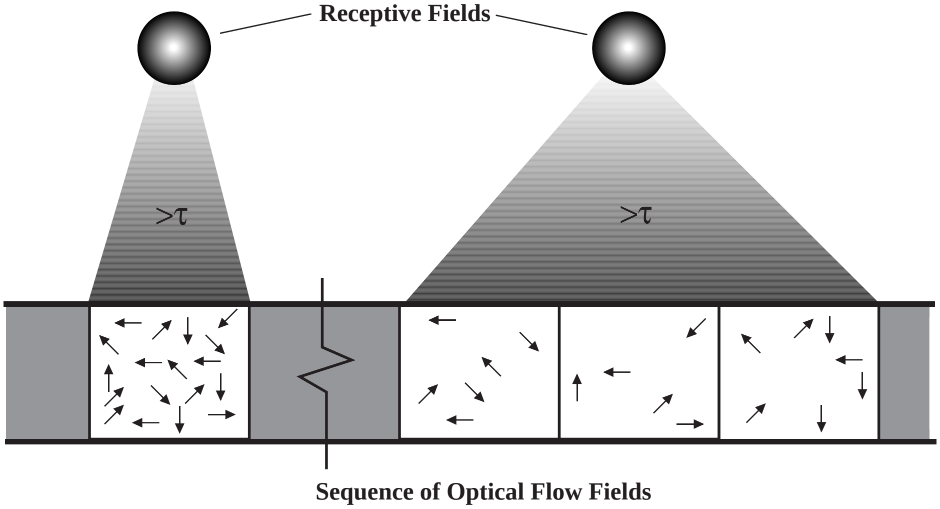


Figure 12: The temporal resolution of the receptive field centers

a receptive field responding poorly to an input vector. In the implementation of our architecture, this ambiguity did not present a problem. Since an ensemble of receptive field activations was used to collectively generate an output response, the ambiguities at specific receptive fields were absorbed; thus, the network was able to learn the correspondence between receptive field responses and desired outputs no matter how the receptive fields arrived at their activation state. This is similar to the theory in psychology and neuroscience, *across-pattern-fiber-theory*, used to explain how ensembles of neurons in the brain are used to collectively generate a response while overcoming the ambiguities that occur at individual neurons [7].

9 Implementation of Training

In order to ensure randomness in our pattern presentation during training, we present the network with all of the images from all of the sequences in a random order, removing the notion of an individual sequence. An individual training pattern would then consist of the pre-processed image from a sequence tagged with the learning vector representing the stage of the emotion that the image corresponds to. A difficulty arises in dealing with temporal information using this highly randomized training set in this network architecture. Temporal information is a key element of this architecture, which is embedded in the fixed sequential ordering of the images in a sequence, and is lost when this ordering is altered. In this architecture, the temporal relations of an emotion are captured by the receptive field activation patterns over the entire set of receptive fields, which are a result of the combination of remnant activations from previous inputs and current responses from the current input. The weights between the receptive field layer and the output layer are trained to learn the correspondence between these receptive field patterns and the stage of the emotion coded into the learning vector. If we capture the receptive field activation patterns for each image in a sequential sequence presentation to the network, and tag these activation patterns with the learning vector that corresponds to the current stage of the input image, we can use the resulting pairs as the training exemplars for the training set. This training set can now be presented to the network in a completely random order without losing the temporal relations between images in a sequence.

The weight training phase thus required an extra pre-processing step that involved building the training set of receptive field activation pattern/learning vector pairs. We built the training set by exposing the network to each subject sequence for a particular emotion in sequential order, capturing the receptive field activation pattern and current emotion stage for each image, and placing the pair in a training buffer. Once the training buffer was filled with the pairs, a training exemplar was randomly selected from the training buffer for each iteration of weight training. This technique solved the retention problems of the network during training.

10 The Analysis of Network Output

10.1 The Absolute Analysis of the Emotion Tuned Network

In this subsection we focus on the interpretation of the output vector from an emotion tuned network. This analysis level will be called the *absolute analysis.* Neural networks will always produce an output result, which in some cases is beneficial because this allows for interpolation. It is not desired, however, for the network to interpolate in regions of the input space where it did not specialize; outputs as a result of inputs from these regions can be unreliable. In order to discard unreliable results we need a measure of network confidence for the generation of an output vector. This confidence measure can then be used to determine whether to use the information provided by the network or not, or in a more continuous approach, how much to weigh the results of the network. It also provides us with a relative means of comparison between networks to determine which network is the most confident. We made use of a technique called Output Appearance Reliability Estimation (OARE) to provide this confidence measure [19]. This technique of determining network reliability is based on comparing the shape of the network generated output distribution with the ideal shape of the training vector distribution; the farther the actual distribution is from the ideal distribution the less reliable the network is. As previously discussed, the output vector in this application was trained to generate a Gaussian output distribution.

Hypothetically, if a network has learned to recognize the stages of an emotion with 100% accuracy, then over the sequence of input images, every stage of the emotion will be turned on in sequential order; thus, we are not only concerned with how many and which stages turned on but also with the order in which they turned on. We use three scoring measures to interpret the neural network output:

- *Stage count*: This measure determines the number of stages that turn on during an emotion sequence. Each time a stage is turned on, the Gaussian distribution should be centered on the output unit corresponding to that stage. Figure 13 shows the ideal output activation distribution over the output units for each iteration of a perfectly learned input sequence. Every stage of the emotion will be the center of the Gaussian distribution. We can maintain a counter for each stage that is incremented whenever that stage occurs at the peak of the Gaussian. Ideally, at the end of an emotion sequence every stage counter should be at least one, and the sum of the stage counters should equal the number of images in the input sequence. Note that in this hypothetical situation, the output distributions for each stage are perfect Gaussians; this will not be true for an actual network. We want to determine passage through a stage, but we do not want to increment a stage counter if the network confidence is low. First, we determine at which stage the approximate peak occurs. We find the approximate peak by scanning the output vector for the maximum activation. In order to determine if we increment a stage counter, a confidence measure is determined for every output generation, and if the iteration confidence is greater than a threshold, we increment the stage counter.

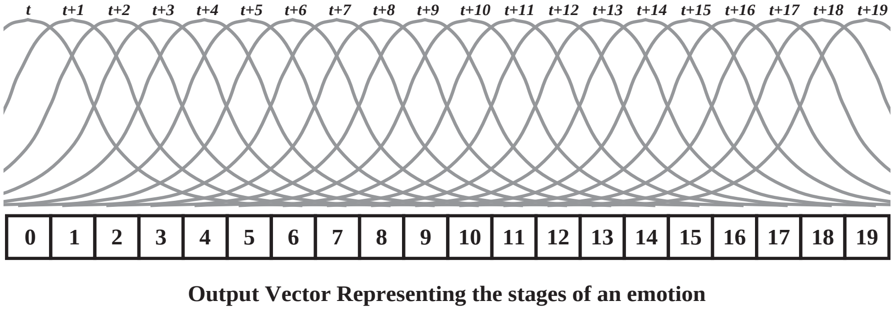


Figure 13: The Gaussian weighted output vector

- *Transition ratio*: This measure relates to the sequential ordering of the stage activations. We previously discussed the use of decay constants in an emotion tuned network to help the network learn these temporal relations. The decay constants should minimize high activations for motions that do not occur in the correct order for a specialized emotion network. A network with decay constants, however, can be fooled. An incorrect lengthy motion sequence can register a large activation, and will therefore turn on several stages. In observing the output patterns over time for emotion sequences that did not match the specialization of the network, we noticed that several stages would often turn on, but the ordering was incorrect; later stages turned on before earlier stages. This was a sign that the network was not viewing an emotion that it was tuned for. In this second measure we rewarded forward stage transitions, penalized reverse stage transitions, and minimally penalized no-stage transitions. The reward or penalty was added to a running sum, called the transition score, throughout the emotion sequence. The transition score was then normalized by the number of iterations in the sequence to get the average transitions per iteration, and this was called the transition ratio which was used in the interpretation rules.

- *Average stage confidence*: The first metrics only make use of high confidence and weakness and do not utilize occurrences of low confidences. It may be possible for several stages to turn on with very high confidence, and the rest to have extremely low confidence—so low, in fact, that it is probable that the input emotion is not the emotion the network is tuned for. In order to utilize low confidence a second confidence measure is calculated. This score is determined by summing all of the iteration confidences determined throughout the sequence, and dividing by the number of iterations to get the average confidence per iteration. For trials on subjects not in the training set, the stage score and transition ratio are often not as revealing as for trials on subjects in the training set for the same emotion. At this point we can look at the average confidence to gain some insight as to whether the network is really seeing the emotion it is tuned for.

There are various ways to use the above measures to interpret the network response. One approach is to use a rule set, and a second approach is to use a neural network to interpret the results. We found it necessary to tune the output interpretation rules by varying thresholds in the rules. There are four possible outcomes of the rule set for the interpretation of the network output: 1) a hit, 2) a miss, 3) a false alarm, and 4) a correct negative. By tuning the rules it was possible to vary the conservativeness of the system. By making the thresholds high, the rules became strict, reducing the number of false alarms, but increasing the number of misses. By making the thresholds low, thus, making the rules less strict, the number of hits increased, but so did the number of false alarms. We tuned the interpretation rules until the responses were close to optimal for a model of unit cost and rewards for incorrect and correct decisions, respectively.

10.2 The Relative Analysis of the Emotion Tuned Networks

In the previous subsection we identified several measures to interpret the individual emotion-tuned network response. These same measures can be used in a relative comparison between different emotion tuned networks to determine the emotion of a subject in an emotion sequence. In the relative analysis, a network that experienced ambiguous results in the absolute analysis may score relatively higher than the other emotion networks, resolving the ambiguity. It is important to recognize that it is possible that no emotion is being conveyed through facial feature motion, and the relative analysis should be able to detect this. If no emotion network responds beyond a minimum threshold of response, then the relative analysis response will be for a null emotion.

11 Experiments and Results

For our preliminary experiments, we only trained two emotion networks; one for the "smile" emotion, and another for the "surprise" emotion. The test stage included image sequences of "smile", "surprise", and "anger" emotions.

Our database of image sequences includes 32 different faces (see Figure 14). For each face several expressions were recorded each lasting between 15 and 120 frames of 120 × 160 pixels (at 30 frames per second), some expressions recurring. We requested each subject to display the emotions in front of the video camera while minimizing his/her head motion. Nevertheless, subjects inevitably moved their heads during a facial expression. As a result, the optical flow at facial regions was sometimes overwhelmed by the overall head rigid motion. Our facial expression system detects such rigid motion easily (*all* face regions move in one direction, an event unlikely to be found in a facial expression) and marks the respective frames as unusable for analysis.

Before we discuss the methodology and results we define the terminology used. The term *familiar-face* indicates that the face used is that of a person that the system has seen in the *training* session. For such a face there can be two types of sequences, *familiar-* and *unfamiliar-sequences*. The former denotes those image sequences that were used in the training, and the latter indicates these sequences of the *familiar-face* that are new to the system.

11.1 Absolute Analysis

In order to evaluate the performance of the neural network architecture, we conducted experiments that measure the network's *retention, extrapolation,* and *rejection* ability. Retention refers to the ability of the network to perform successfully on familiar sequences. Extrapolation refers to the ability of the network to perform successfully on sequences of unfamiliar faces. Rejection refers to the ability of the network to reject a sequence that did not express the emotion that the network was tuned for.

To measure the performance of the system relative to the above criteria we divided the experiments into four categories. The first category encompassed familiar sequences, and it measured the network's retention ability. In the second category, unfamiliar faces were tested in order to measure the extrapolation ability. The third category included unfamiliar sequences of familiar faces and it measured a smaller increment of extrapolation than the second category. The fourth category included sequences of emotions that the tuned network

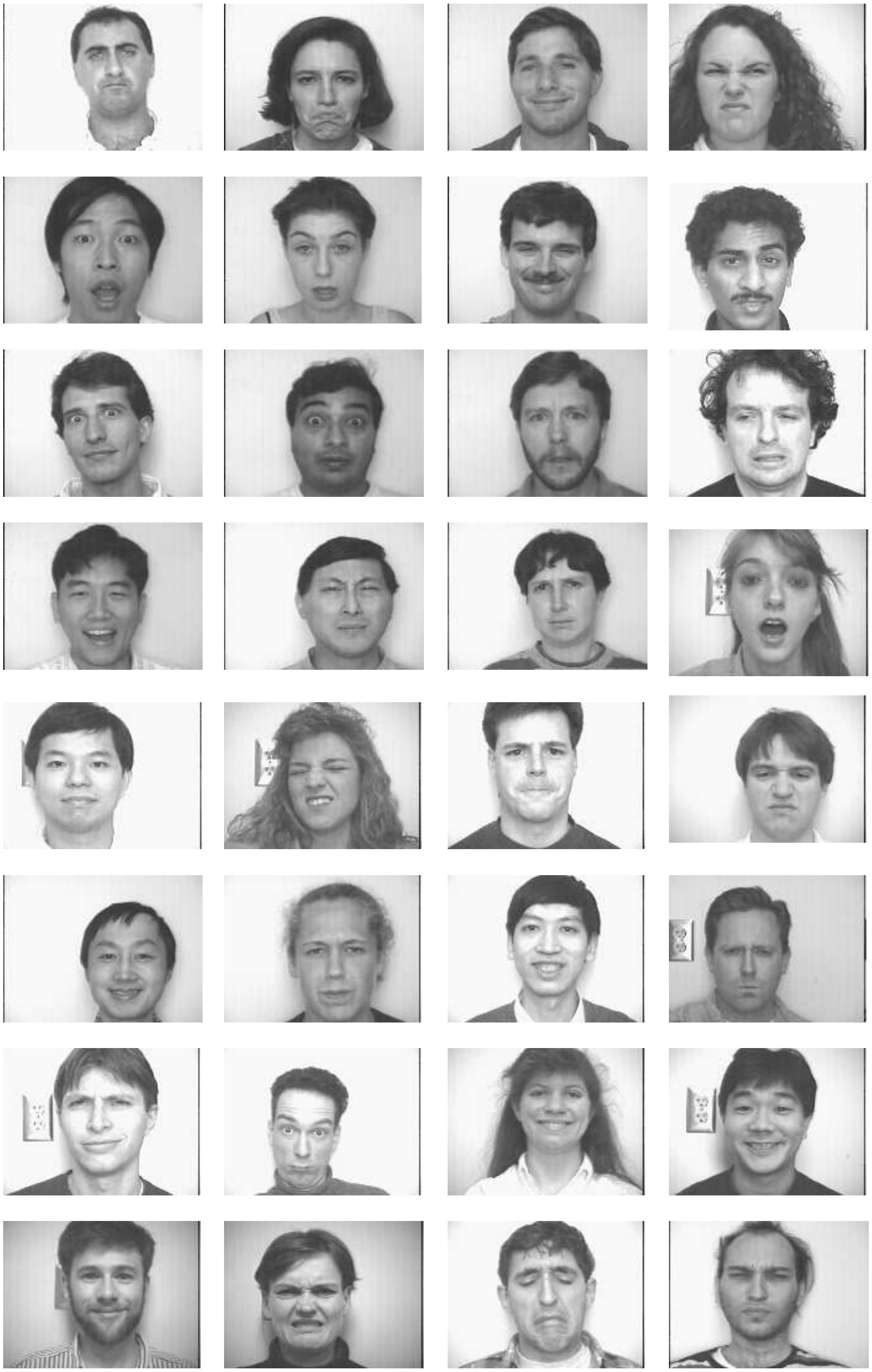


Figure 14: Thirty-two faces used in experiments

did not specialize in (these can be for any type of emotion and face) and it measures the rejection rate of the network.

For the “smile” and “surprise” emotions, we trained two networks that only differed in receptive field width, and we tested each network using the four test categories. Each network was trained for 100,000 iterations, and the receptive field widths for SMILENET 1 and SURPNET 1 were larger than the receptive field widths for SMILENET 2 and SURPNET 2 (see Table 2). The “smile” and “surprise” networks were trained with 20 and 14 subjects, respectively. The output vector for each network represented 40 stages of an emotion. We used the criterion of at least seven stages being turned on to signify that the network recognized the emotion of a sequence, and we used an iteration confidence threshold of 0.155 to increment a stage counter for a frame of the sequence. Table 3 shows the results from the absolute analysis.

Table 2: The *relative* receptive field width settings for the experimental analysis

network	mouth rf widths	eyebrow rf widths
SMILENET 1	1	1
SURPNET 1	1	1
SMILENET 2	.694	.563
SURPNET 2	.694	.563

Table 3: The results of the absolute analysis

network	familiar seq.	unfamiliar face	unfamiliar seq.	foreign expression
SMILENET 1	16/20=80%	2/4=50%	7/7=100%	29/41=71%
SURPNET 1	13/14=93%	5/6=83%	3/3=100%	39/52=75%
SMILENET 2	16/20=80%	2/4=50%	4/7=57%	32/41=78%
SURPNET 2	13/14=93%	2/6=33%	3/3=100%	46/53=87%

In Table 4 we further break down category 4 to show that “anger” sequences were rejected differently than “smile” and “surprise” sequences. The “surprise” and “smile” rejection rates only apply to a network if it was trained for the alternate emotion from the test sequence.

The results indicate that the retention rates are higher than the extrapolation rates. Category 3 success rates are higher than category 2 rates, since category 3 represents a smaller increment of extrapolation from the training set than does category 2. In the absolute analysis we found that the “surprise” networks performed better overall than the “smile” networks because of larger detectable motion.

In Table 4 the rejection rates for “surprise” were better than those for “smile” for the three emotions. For the “smile” and “surprise” networks with the same receptive field widths, the “surprise” network had a much higher rejection rate of the “smile” emotion than the “smile” network had of the “surprise” emotion. The larger detectable motion of

Table 4: The further breakdown of category 4

network	anger	surprise	smile
SMILENET 1	16/18=89%	13/23=57%	–
SURPNET 1	17/18=94%	–	22/31=71%
SMILENET 2	16/18=89%	16/23=70%	–
SURPNET 2	18/18=100%	–	28/31=90%

the "surprise" emotion improved performance for all four test categories, thus improving retention, extrapolation and rejection of the "surprise" networks over the corresponding width size "smile" networks.

Also from Table 3 and Table 4 we can see that larger receptive field widths enhanced extrapolation abilities of the networks (categories 2 and 3), but at the same time reduced the retention and rejection rates (categories 1 and 4). Since one of the main goals of this research was to determine if a network could learn the commonalities of an emotion over a wide population from a small sample set, wider receptive field widths are better suited for our application. Wider receptive fields respond to larger regions of the input space surrounding the receptive field center. On one hand, if the receptive fields widths for a network are too large, thus over-generalizing, then all the receptive fields will respond with equally large activations, and the categorizing ability of the network is lost. On the other hand, if the receptive field widths are too small, the receptive fields will respond crisply to training patterns, but will have negligible responses to test patterns that only vary slightly from the training patterns, thus possessing no generalization ability. Therefore a retention/extrapolation trade-off exists between large and small receptive field widths.

11.2 Relative Analysis

Since it was our intention to teach a network extrapolation instead of retention, we focused our relative analysis on networks SMILENET 1 and SURPNET 1, which had better extrapolation performance because of their relatively larger receptive field widths. The relative analysis is dependent on the results of the absolute analysis. Similarly, in the relative analysis we defined four test categories to measure retention, extrapolation and rejection. The first category tests familiar sequences of "smile" or "surprise". The second category tests sequences of unfamiliar faces. The third category tests unfamiliar sequences of familiar faces in at least one of the two training sets. The fourth category tests expression sequences foreign to both networks. Since we trained on the "smile" and "surprise" emotions, the only emotion sequences in the fourth category were those of "anger".

In the relative analysis, we compare the responses of the two networks; the thresholding is done in the earlier absolute stage of analysis. In the case of two networks, we have four possible combinations of outputs: Yes/Yes, No/Yes, Yes/No, and No/No (where a "Yes" signifies that a network recognizes a sequence as its specialization emotion, and a "No" signifies the network did not recognize the emotion). The Yes/No and No/Yes responses are straightforward, in that the relative emotion response is taken as the emotion of the network that responded with a “Yes”. The No/No relative response also represents a clear answer that neither network recognizes the emotion of the sequence. The Yes/Yes response is ambiguous, however, and is resolved by the relative analysis. To resolve the Yes/Yes ambiguity, the absolute output statistics of each network for the ambiguous sequence are compared. We used the number of stages turned on as the comparison statistic. The network that had the highest number of stages turned on was declared the winning network, and the resultant emotion was determined to be the specialization emotion of that network. The Yes/Yes ambiguous response was possible in test categories 1, 2, and 3; thus, the relative ambiguity resolution was expected to improve the performance for these three categories. Table 5 shows the resulting accuracies from the relative analysis after the ambiguity resolution for categories 1, 2 and 3.

Table 5: The results of the relative analysis

familiar seq.	unfamiliar face	unfamiliar seq.	foreign expression
30/34=88%	11/15=73%	11/12=92%	14/18=77%

In order to compare the absolute and relative analyses, the absolute performances for the SMILENET 1 and the SURPNET 1 are combined into one performance measure based on a weighted average of the number of test cases for each network in each test set category, except category 4, since it does not apply. Table 6 shows the combined results from the absolute analysis compared with the results from the relative analysis for each category. The results show an expected slight performance improvement for categories 1 and 2, and an unexpected slight reduction in performance for category 3 between the absolute and relative analysis. The reduction in performance for category 3 was caused by the incorrect network having a higher score than the correct network.

Table 6: Comparison of the absolute results with the relative results

analysis	familiar seq.	unfamiliar face	unfamiliar seq.
absolute	85%	70%	100%
relative	88%	73%	92%

12 Conclusion

In this paper, we developed a radial basis function network based human emotion detection system. By training the network, it was able to learn the correlations between facial feature motion patterns and specific emotions. In order to capture the temporal relations that are important to emotion detection, several enhancements were made to the underlying network architecture. In order to make the problem more tractable, the emotion detection problem was divided into several levels, which mapped directly back to the overall network architecture. The high level decomposition was by emotion, the mid level decomposition was by facial feature, and the low level decomposition was by motion direction sensitivity. For our preliminary experiments, of the six universal human emotions, we trained networks to recognize the "smile" and "surprise" emotions. Our experiments tested each network's retention, extrapolation, and rejection abilities. The analysis of the experimental results were conducted by absolute and and relative terms. The purpose of the relative mode was to improve overall emotion detection over the absolute mode by comparing all network outputs and picking a winner.

From our experiments, we found, as one might expect, that networks tuned better on emotions that involved more motion. The larger the motions and the more sources of motion in the image, the more sources of discrimination between emotions. We also found that a trade-off existed between large and small receptive field widths. Large widths improved extrapolation, while degrading retention and rejection, while small widths had the opposite effect. Since our focus was to learn generalities, we used relatively large receptive field widths.

We encountered several problems in these preliminary experiments due to subject rigid motions and subject orientation. The neural network learns to correlate motions in certain parts of the image retina with specific emotions. When these motion patterns are offset by rigid motion, they do not register properly with the neural network. The problems due to varying subject orientations can be handled algorithmically or through neural network training. Algorithmically, the image can be adjusted based on facial feature locations and the assumption of facial symmetry. A second approach is to include varying orientations of subjects in the training set, and allow the network to learn these variations.

References

[1] M. Abdel-Mottaleb, R. Chellappa, and A. Rosenfeld, "Binocular motion stereo using MAP estimation", *Procceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 321–327, 1993.

[2] J.N. Bassili, "Emotion recognition: The role of facial movement and the relative importance of upper and lower areas of the face", *Journal of Personality and Social Psychology*, Vol. 37, 2049–2059, 1979.

[3] V. Bruce, *Recognizing Faces*, Lawrence Erlbaum Assoc., London, 1988.

[4] C. Darwin, *The Expression of Emotions in Man and Animals*, John Murray, 1872, reprinted by University of Chicago Press, 1965.

[5] P. Ekman and W. Friesen, *Unmasking the Face*, Prentice-Hall, 1975.

[6] P. Ekman and W. Friesen, *The Facial Action Coding System,* Consulting Psychologists Press, San Francisco, CA, 1978.

[7] E.B. Goldstein, *Sensation and Perception*, Wadsworth Publishing Co., 1989.

[8] W.E.L. Grimson, B.K.P. Horn and T. Poggio, "Progress in image understanding at MIT", *Proceedings of the DARPA Image Understanding Workshop*, 1992.

[9] J. Hertz, A. Krogh and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.

[10] J.A. Howell, C.W. Barnes, S.K. Brown, G.W. Flake, R.D. Jones, Y.C. Lee, S. Qian and R.M. Wright, "Control of a negative-ion accelerator source using neural networks", *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, Vancouver, BC, Canada, 1989.

[11] R.D. Jones, C.W. Barnes, G.W. Flake, Y.C. Lee, P.S. Lewis, M.K. O'Rourke and S. Qian, "Prediction and control of chaotic processes using nonlinear adaptive networks", *Proceedings for Nonlinear and Chaotic Processes in Plasmas, Fluids, and Solids*, Edmonton, Alberta, Canada, 1990.

[12] R.D. Jones, "Nonlinear adaptive networks: A little theory, a few applications", *Proceedings of the First Los Alamos Workshop on Cognitive Modeling in System Control: Theoretical Foundations and Prospects for Applications*, Santa Fe, NM, 1990.

[13] R.D. Jones, Y.C. Lee, C.W. Barnes, G.W. Flake, K. Lee, P.S. Lewis and S. Qian, "Function approximation and time series prediction with neural networks", *Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA, 1990.

[14] Y.C. Lee, "Neural networks with memory for intelligent computations", *Proceedings of the 13th Conference on the Numerical Simulation of Plasmas*, Santa Fe, NM, 1989.

[15] H. Li, P. Roivainen, and R. Forcheimer, "3-D motion estimation in model-based facial image coding", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, 545–555, 1993.

[16] K. Mase, "Recognition of facial expression from optical flow", *IEICE Transactions,* Vol. E 74, 3474–3483, 1991.

[17] K. Matsuno, C. Lee, and S. Tsuji, "Recognition of human facial expressions without feature extraction", *Proceedings of the European Conference on Computer Vision*, 513–520, 1994.

[18] J. Moody and C. Darken, "Learning with localized receptive fields", *Proceedings of the 1988 Connectionist Models Summer School*, eds. Touretzky, Hinton, and Sejnowski, Morgan Kaufmann, 1988.

[19] D.A. Pomerleau, "Neural Network Perception for Mobile Robot Guidance", Ph.D. thesis, Carnegie Mellon University, Department of Computer Science, February 1992.

[20] M. Rosenblum and L. Davis, "The Use of a Radial Basis Function Network for Visual Autonomous Road Following", Technical Report CAR-TR-666, Center for Automation Research, University of Maryland, College Park, MD, May 1993.

[21] D. Terzopoulos and K. Waters, "Analysis and synthesis of facial image sequences using physical and anatomical models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, 569–579, 1993.

[22] Y. Yacoob and L.S. Davis, "Labeling of human face components from range data", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 592–593, 1993.

[23] Y. Yacoob and L.S. Davis, "Computing spatio-temporal representations of human faces" *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[24] Y. Yacoob and L.S. Davis, "Recognizing human facial expression", Technical Report CAR-TR-706, Center for Automation Research, University of Maryland, College Park, MD, May 1994.

[25] A.W. Young and H.D. Ellis (eds.), *Handbook of Research on Face Processing*, Elsevier Science Publishers, 1989.

[26] A.L. Yuille, D.S. Cohen, and P.W. Hallinan, "Feature extraction from faces using deformable templates", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 104–109, 1989.