Real-time Camera-based Face Detection using a Modified LAMSTAR Neural Network System

Javier I. Girado*, Daniel J. Sandin, Thomas A. DeFanti, Laura K. Wolf
Electronic Visualization Laboratory, University of Illinois at Chicago, Chicago, IL USA 60607

ABSTRACT

This paper describes a cost-effective, real-time (640x480 at 30Hz) upright frontal face detector as part of an ongoing project to develop a video-based, tetherless 3D head position and orientation tracking system. The work is specifically targeted for auto-stereoscopic displays and projection-based virtual reality systems. The proposed face detector is based on a modified LAMSTAR neural network system. At the input stage, after achieving image normalization and equalization, a sub-window analyzes facial features using a neural network. The sub-window is segmented, and each part is fed to a neural network layer consisting of a Kohonen Self-Organizing Map (SOM). The output of the SOM neural networks are interconnected and related by *correlation-links*, and can hence determine the presence of a face with enough redundancy to provide a high detection rate. To avoid tracking multiple faces simultaneously, the system is initially trained to track only the face centered in a box superimposed on the display. The system is also rotationally and size invariant to a certain degree.

Keywords: face tracking, face detection, artificial neural networks, LAMSTAR, head tracking

1. INTRODUCTION

The Electronic Visualization Laboratory (EVL) is one of several research groups working on producing PC-driven, projection-based virtual reality (VR) displays. The use of high-end systems, such as EVL's CAVE and ImmersaDesk are well established in application areas such as computational science, automotive engineering and chemical exploration. The next-generation of VR displays, both tiled LCD displays and projection-based, aim to eliminate encumbrances on the user. The trend is towards higher resolution displays where the user is not required to wear special glasses to view stereoscopic scenes. Analogously, the trend for interaction with these displays is towards lightweight and tetherless input-devices.

EVL and its collaborators are exploring the use of other modalities (such as vision, speech and gesture) as human-computer interfaces for this new generation of VR systems. Gesture recognition can come from either tracking the user's movements or processing them using video camera input. Gaze direction, or eye tracking, using camera input is also possible. Audio support can be used for voice recognition and generation, as well as used in conjunction with recording tele-immersive sessions. Used together, these systems enable tetherless tracking and unencumbered hand movements for improved interaction and collaboration within the virtual scene.

The research described here is a cost-effective real-time (640x480 at 30Hz) face detector that will serve as the core of a video-based, tetherless 3D head position and orientation tracking system targeted for either auto-stereoscopic displays or projection-based virtual reality systems. It will be tested first using EVL's Access Grid Augmented/Autostereo Virtual Environment (AGAVE), a high-resolution autostereoscopic display consisting of tiled LCD displays driven by a PC cluster and fitted with a highly sensitive tracking system to track user's gaze and gestures without the use of head mounted or hand held tracking devices.

The complete tracking system (Fig. 1) will consist of two neural network-based face detectors working in parallel, each running through four distinct phases. They are: *pre-processing*, which includes input masking, image normalization, histogram equalization, and image sub-sampling; *detection*, where a modified LAMSTAR neural network takes the pre-

* E-mail: jgirad1@uic.edu; URL: www.evl.uic.edu; Telephone: 312-996-3002; Fax: 312-413-7585

processed image, scans for a face and outputs the coordinates of the corresponding box surrounding the face; *post-processing*, where facial feature extraction and stereo correspondence matching occurs to extract the 3D information; and the implementation of a *prediction module*, which is based on a neural network linear filter with a Tap Delay Line (TDL). The function of the prediction module is to inform the face detection modules where in the scene the face will likely be found, so as to avoid scanning the next whole frame for a face. If the face is not detected in the predicted position, the system will rescan the entire scene. Phases one and three use computer vision techniques. This paper addresses the pre-processing and detection phases.

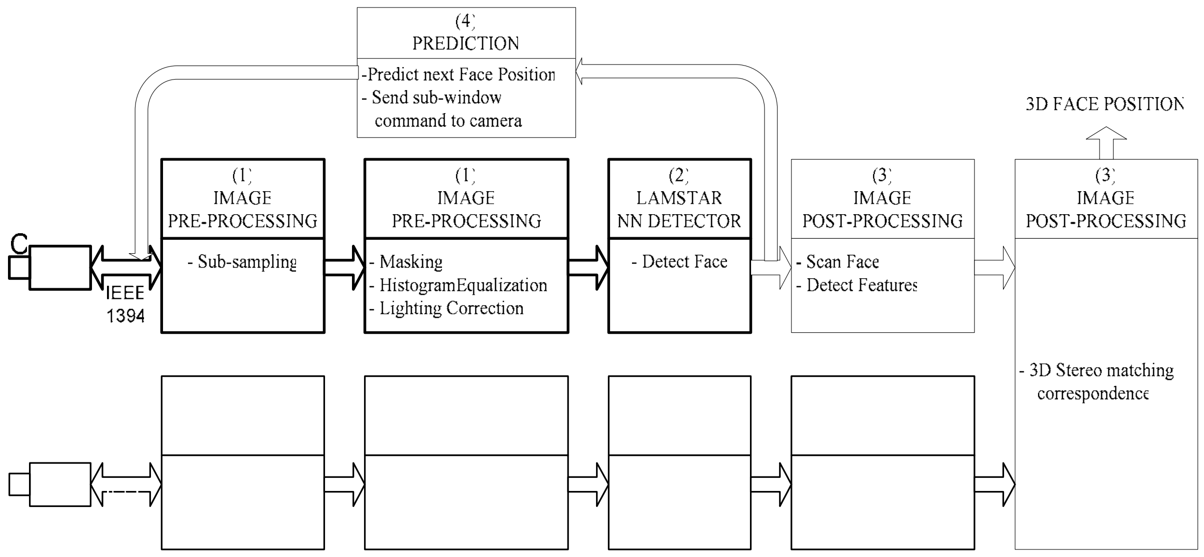


Figure 1: The complete tracking system

Based on recent surveys, face detection approaches rely upon one or a combination of the following techniques: Feature-based, Image/View-based and Knowledge-based. The proposed face detector is based on a modified LAMSTAR neural network system along with a novel combination of the three techniques mentioned above.

At the input stage, after image normalization and equalization are achieved, the information is divided into sub-pattern categories, each representing a part of the raw image. Each sub-pattern is then fed to a neural network layer that is a Kohonen SOM (Self-Organizing Map) module. The output of all the neural network modules are interconnected by correlation-links, and can hence determine the presence of a face with enough redundancy to provide a high detection rate. The face detector is also rotationally and size invariant for front-view faces to a certain degree. The full paper will detail several aspects of the technique.

To meet real-time constraints (low latency and high frame rates), the algorithms are highly tuned for the new Intel 4 Family Processor micro-architecture, specifically its vector processor.

2. BACKGROUND

The core of the face detector is a system of artificial neural networks based on Graupe and Kordylewski's LArge Scale Memory STorage And Retrieval (LAMSTAR) network research, which targets large-scale memory storage and retrieval problems[1,2,3,4]. This model attempts to imitate, in a gross manner, processes of the human central nervous system (CNS) concerning storage and retrieval of patterns, impressions, and sensed observations including processes of forgetting and recollection. It attempts to achieve this without contradicting findings from physiological and psychological observations, at least in an input/output manner. Furthermore, it attempts to do so in a computationally efficient manner using tools of neural networks, especially Self-Organizing-Map based (SOM) network modules, combined with statistical decision tools. Its design was guided by trying to find a mechanistic neural network-based model for very general storage and retrieval processes involved in, say, recalling the face of a previously encountered person. LAMSTAR follows a top-down approach based on neurophysiological observations that involve many parts of the CNS for the reconstruction of information. This general approach[1] is related to Minsky's idea[5] that the human brain consists of many agents, and a *knowledge link* is formed among them whenever the human memorizes an experience. When the knowledge link is subsequently activated, it reactivates the mental agents needed to recreate a mental state similar to the original. The LAMSTAR network employs this general philosophy of linkages between a large number of physically separate modules that represent concepts, such as time, location, patterns, etc., in an explicit algorithmic network. (Note: For consistency, this paper uses the term *correlation links* instead of *knowledge links*.)

The LAMSTAR network has been successfully applied in fields of medicine (diagnosis)[2,3,4], engineering (automotive fault detection) and multimedia information systems[6]. Whereas the LAMSTAR design addresses large-scale memory retrieval problems, the face detection system is a comparatively small-scale retrieval problem. We are most interested in modeling our system design based on the LAMSTAR concept of using correlation links to retrieve stored information.

Our real-time system design has two major constraints: the detection and subsequent tracking of a face must be performed in under 33 ms to achieve 30 fps, and we only require the memory capacity needed to store one face. Hence, we limit our use of LAMSTAR concepts to processes of storage and retrieval, interpolation and extrapolation of input data, and the use of reward-based correlation-links between modules to retrieve stored information. Since our coding approach deals only with pre-processed raw images, we use *input vector* or *input pattern* when referring to the sub-window image, *sub-vectors* or *sub-patterns* when referring to the post-processed segmented image, and *correlation-links (C-links)* when referring to the connection between modules.

3. SYSTEM OVERVIEW

The modified LAMSTAR network is a self-trained/self-organized system of neural networks composed of Self-Organizing-Map (SOM)[7] modules (Fig. 2 and Fig. 3), with added statistical decision tools. Interpolation takes place at these SOM levels, where the incomplete sub-pattern (minor variation or lack of data) is reconstructed as a result of the generalization property of the SOM network. The extrapolation, which is the primary function of associated memory models, is a result of correlation-based retrieval among SOM modules where one or more (but not all) sub-patterns are needed to reconstruct the entire pattern.

3.1 The Kohonen Self-Organizing-Map

Our system needs to map, in an unsupervised mode, whatever cluster of input data it is presented. Kohonen's Self-Organizing-Map (SOM)[7,8] is best suited for this task because it is a self-organizing neural network that quickly integrates new input data into existing clusters by analyzing, then classifying the data using connection weights modified through different iterations.

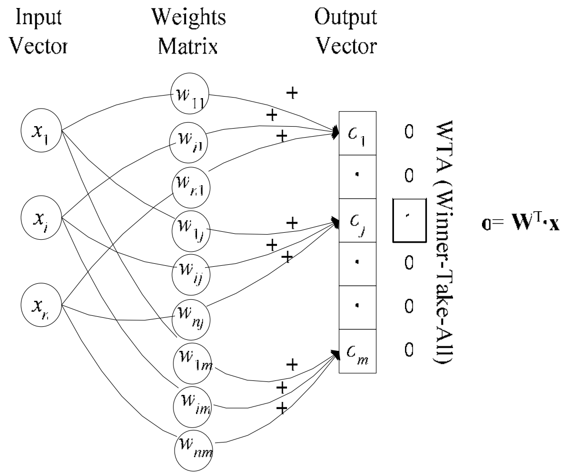


Figure 2: Kohonen Self-Organize-Map (SOM) network

Using the Lateral Inhibition technique, where distant neighbors are inhibited and closer ones reinforced, the Kohonen SOM is able to infer relationships and, as more input data is presented (in our case pre-processed video images), it creates the closest possible set of outputs for the given inputs. The output of a Kohonen SOM network (Fig. 2) is a weighted sum of its inputs:

$$k_j = \sum_{i=0}^{m} w_{ij} x_i = \mathbf{w}_j^{\mathrm{T}} \mathbf{x}, \text{ where: } \mathbf{w}_j = \left[w_{1j} ... w_{mj} \right]^{\mathrm{T}}, \mathbf{x} = \left[x_1 ... x_m \right]^{\mathrm{T}} \tag{1.1}$$

After the equation weights (w_{ij}) are computed during the training phase, an unknown case is presented to the network. All outputs are found and the maximum output neuron is declared the winner, thus determining its class. The Kohonen neural network implements a “Winner-Take-All” (WTA) strategy[8], i.e., for any given input vector only one Kohonen neuron output is 1, whereas all others are 0:

$$k_{winner} > k_{j \neq winner} \Rightarrow k_{winner} = \sum_{i=0}^{m} w_{i\,winner} x_i = 1, k_{j \neq winner} = 0 \tag{1.2}$$

The neighborhood size parameter is used to model the effect of the Mexican hat or Gaussian hat function. Only input patterns whose neurons fall within the neighborhood size participate in training (i.e., learning) and weight changes (i.e., updates); all others are excluded. A good reference for these techniques and Kohonen SOM network implementations can be found at [9,10].

3.2 SOM Modules: description, input and training

In this modified LAMSTAR network, each Kohonen SOM module represents a class of sub-patterns. The model assumes that the input patterns have been separated into sub-patterns before entering the SOM module (Fig. 3). The network is thus organized to assign each neuron to a class of neurons (i.e., one SOM module) that best corresponds to the input sub-pattern. This SOM configuration yields very rapid matching with good error tolerance, and is capable of generalization.

Arrays of correlation links (C-links) horizontally connect the modules using coefficients determined by the statistical correlations between the various patterns considered. A coordinated activation of neurons between the various modules allows the network to recreate (interpolate) complex patterns and make associations (i.e., detect a face).

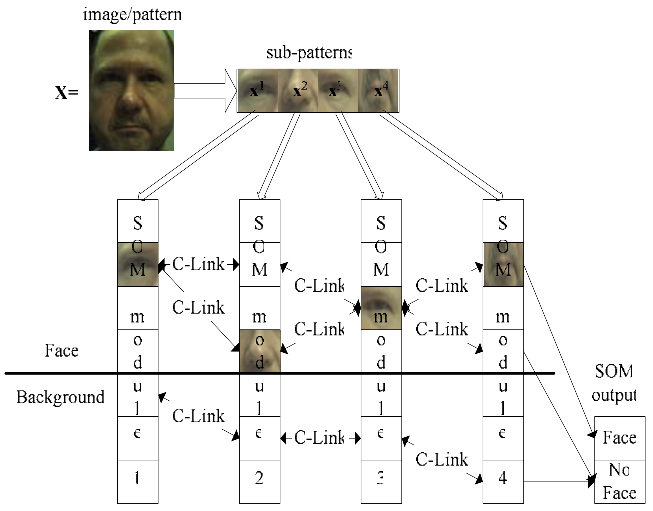


Figure 3: Simplified LAMSTAR diagram including the relationship between SOM models, SOM output layer and C-links

The input pattern/image is coded in terms of a real vector x given by:

$$\mathbf{x} = [\mathbf{x}^{1^{\mathrm{T}}}, ..., \mathbf{x}^{i^{\mathrm{T}}}, ..., \mathbf{x}^{m^{\mathrm{T}}}]^{\mathrm{T}} \qquad (1.3)$$

The dimension of each *i'th* sub-vector varies according to the number of elements needed to describe a sub-pattern feature. To store data concerning the *i'th* category of the input pattern, each sub-pattern $\mathbf{x}^i$ is then channeled to the corresponding *i'th* SOM module. Thus, the dimension of a sub-pattern is equal to the amount of pixels representing a specific feature.

A winning neuron is determined for each input based on the similarity between the input vector $\mathbf{x}^i$ and weight vectors $\mathbf{w}^i$ (stored information). For a sub-pattern $\mathbf{x}^i$, the winning neuron is determined by the minimum Euclidean distance between $\mathbf{x}^i$ and $\mathbf{w}^i$:

$$\left\|\mathbf{x}^i - \mathbf{w}^i_{winner}\right\| = \min\left\|\mathbf{x}^i - \mathbf{w}^i_k\right\| \quad \forall\, k$$

where: $\mathbf{x}^i$	-	input vector in $i'th$ SOM module	
$winner$	-	index of the winning neuron	
$\mathbf{w}^i_{winner}$	-	winner weight vector in $i'th$ SOM module	
k	-	a number of neurons (stored patterns) in $i'th$ SOM module	(1.4)
$\left\|\bullet\right\|$	-	Vector Euclidean distance: $\left\|\mathbf{x}-\mathbf{w}\right\| = \sum_{i=1}^{i=n}(w_i - x_i)^2$	
where: n	-	dimension of sub-vectors $\mathbf{x}$ and $\mathbf{w}$	

The SOM module is a Winner-Take-All (WTA) network where only the neuron with the highest correlation between its input vector and its correspondence weight vector will have a non-zero output. The WTA feature also involves lateral inhibition such that each neuron has a single positive feedback onto itself and negative feedback connections to all other units.

$$\mathbf{o}^i_j = \begin{cases} 1, & for \left\|\mathbf{x}^i - \mathbf{w}^i_{winner}\right\| < \left\|\mathbf{x}^i - \mathbf{w}^i_j\right\| \ \forall\, winner \neq j \\ 0, & otherwise \end{cases}$$

where: $\mathbf{o}^i_j$	-	output of neuron j in $i'th$ SOM module	(1.5)
$\mathbf{w}^i_{winner}$	-	winning weight vector in $i'th$ SOM module	
$winner$	-	index of winning neuron in $i'th$ SOM module	

The neuron with the smallest error determined by equations 1.4 and 1.5 is declared the *winner* and its weights $\mathbf{w}_{\text{winner}}$ are adjusted using the Hebbian learning law, which leads to an approximate solution:

$$\mathbf{w}^i_{winner}(t+1) = \mathbf{w}^i_{winner}(t) + \alpha \cdot (\mathbf{x}^i(t) - \mathbf{w}^i_{winner}(t))$$

where: $\mathbf{w}^i_{winner}(t+1)$	-	the new value of winning weight vector $\mathbf{w}_{winner}$ in the $i'th$ SOM module	
α	-	learning coefficient	(1.6)
$\mathbf{x}^i(t)$	-	current input $i'th$ sub-vector	
$\mathbf{w}^i_{winner}(t)$	-	current wining weight vector	

The adjustment in the LAMSTAR SOM module is weighted according to a pre-assigned Gaussian hat neighborhood function Δ(*winner,j)*:

$$\mathbf{w}_j^i(t+1) = \mathbf{w}_j^i(t) + \Delta(winner, j)\cdot\alpha\cdot(\mathbf{x}^i(t) - \mathbf{w}_j^i(t))$$

where: $\mathbf{w}_j^i(t+1)$ - new weight of neighbor neuron j from winning neuron *winner*

$\Delta(j, winner)$ - neighborhood define as gaussian hat: $e^{\frac{|j-winner|^2}{2\sigma^2}}$ (1.7)

3.3 Training/Storage phase

The training of the SOM modules and the training of the C-links are performed in separate phases (Fig. 4):

(1) *Storage of new sub-patterns in SOM modules:* In the first case, given an input pattern **x** and for each $\mathbf{x}^i$ sub-pattern to be stored, the network inspects all weight vectors $\mathbf{w}^i$ in the *i'th* SOM module. If any previously stored pattern matches the input sub-pattern within a preset tolerance (*error* ε), the system updates the proper weights or creates a new pattern in the SOM module. It stores the input sub-pattern $\mathbf{x}^i$ as a new pattern, $\mathbf{x}^i = \mathbf{w}_j^i$, where index *j* is the first unused k_j^i neuron in *i'th* SOM module. If there are no more 'free' neurons, the system will fail, which means either the preset tolerance has to be increased to include more patterns in the same cluster of already stored patterns, or more neurons have to be added on the *i'th* SOM module.

(2) *Creation of C-links among SOM modules:* Individual neurons represent only a limited portion of the information input. For efficient storage and retrieval, only individual sub-patterns are stored in SOM modules, and correlations between these sub-patterns are stored in terms of creating/adjusting the C-links connecting the neurons in different SOM modules (Fig. 3, Fig. 4 and Fig. 6). This linkage distributes storage information horizontally between SOM modules. Consequently, in case of failure in one cell, one loses little information if a pattern is composed of many sub-patterns. The neurons in one module of the network are connected to those in another by C-links. Correlation-link coefficient values C-link are determined by evaluation distance minimization as in equation 1.4 and 1.5 to determine winning neurons, where a win (successful match) activates a count-up element associated with each neuron and with it respective input-side link. During training/storage sessions, the values of C-links are modified according to the following simple rule (reward):

$$C_{k,l}^{i,j}(new) = \begin{cases} C_{k,l}^{i,j}(old) - \beta_{reward}(C_{k,l}^{i,j}(old) - C_{Max}), & \text{for } C_{k,l}^{i,j}(old) \neq 0 \\ 1, & \text{otherwise} \end{cases}$$

where: $C_{k,l}^{i,j}$ – correlation link between $k'th$ neuron in $i'th$ SOM module and $l'th$ neuron in $j'th$ SOM module (1.8)

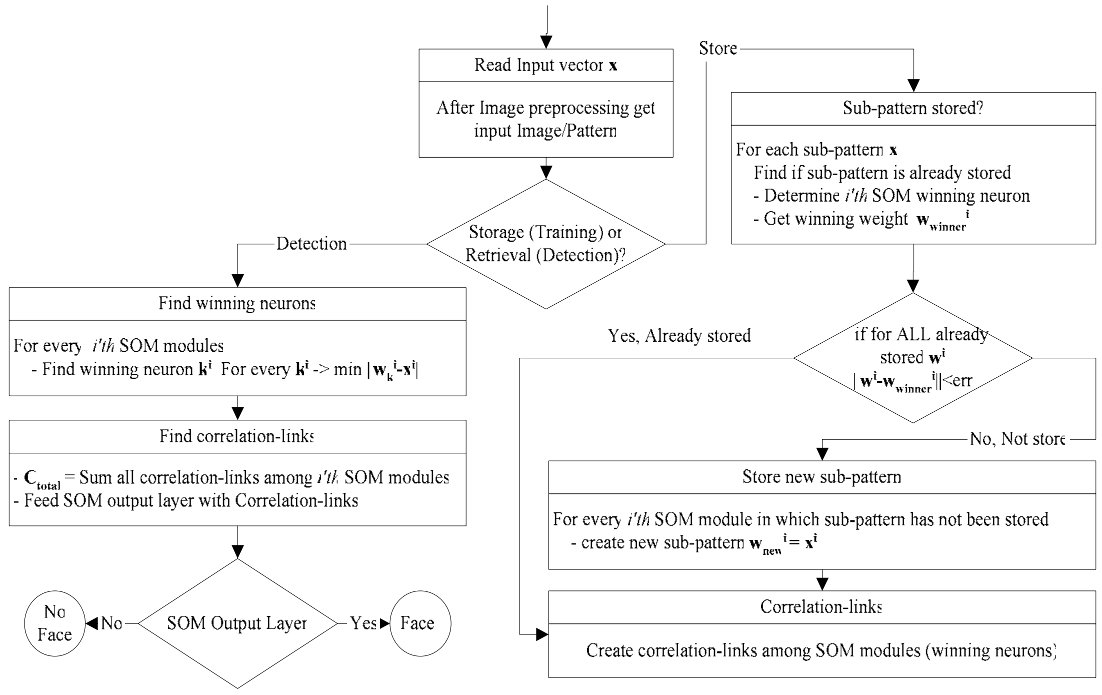
β_{reward} – *reward* coefficient

If the correlation link between two sub-patterns already exists, namely, $C^{i,j}_{k,l} > 0$ (a result from previous training), the formula of equation 1.9 updates (increases) the analyzed C-link. If there are no correlations ($C^{i,j}_{k,l} = 0$), the system creates new C-link with initial value $C^{i,j}_{k,l} = 1$.

3.4 Detection/Retrieval phase

The retrieval phase of the model (Fig. 4) selects sub-patterns from the input pattern one at a time, than examines correlations with stored sub-patterns of a given class of sub-patterns in each SOM module. For example, one *i'th* SOM module could have previously stored noses, and will correlate any given input *i'th* sub-pattern and determine if there is a match or not (e.g., a nose or not).

The face is detected (or retrieved) by means of it C-links. Once all the winning neurons are determined, the system obtained all correlation-links coefficient values among all SOM modules. The output SOM layer (Fig. 2 and Fig. 6), which all C-links are inter-connected, will determine the presence or not of a face.



Read Input vector x
After Image preprocessing get input Image/Pattern
Storage (Training) or Retrieval (Detection)?
Detection
Store
Sub-pattern stored?
For each sub-pattern x Find if sub-pattern is already stored - Determine i'th SOM winning neuron - Get winning weight w_{winner}^i
Find winning neurons
For every i'th SOM modules - Find winning neuron k^i For every k^i -> min $|w_k^i-x^i|$
Yes, Already stored
if for ALL already stored w^i $|w^i-w_{winner}^i|<err$
No, Not store
Find correlation-links
- C_{total} = Sum all correlation-links among i'th SOM modules - Feed SOM output layer with Correlation-links
Store new sub-pattern
For every i'th SOM module in which sub-pattern has not been stored - create new sub-pattern $w_{new}^i = x^i$
Correlation-links
Create correlation-links among SOM modules (winning neurons)
No Face
No
SOM Output Layer
Yes
Face

Figure 4: Flow diagram of the face training/storage and detection/retrieval

4. METHODOLOGY

Functioning as the head tracker in a virtual reality system, the face detector need only detect the face of the person intended for tracking. The system, therefore, accommodates each new user by retraining itself. Training is achieved through the following steps:

- With the face detector system set in training mode, the user faces a video camera and centers his face in a surrounding ellipse (Fig. 6) shown on the face detector workstation screen. This is the training sub-window.
- A command initiates the training and the user slowly runs through a series of subtle head poses (chin left/right, chin up/down, tilting left/right) over the course of 10-15 seconds. This process allows the system to accommodate for the natural changes in the user's face orientation while using the virtual reality system.
- A command terminates the training stage. The user moves away from the camera's field of view. The user may opt to save the internal training parameters under a filename to later retrieve in lieu of retraining the system.
- A command initiates background training. The video camera takes a snapshot of the background scene and trains the neural network as non-face cases. Like in detection (Fig. 5), it scans the whole image using a sub-sampling technique applied to every location of the image-scene to account for every possible face size[11]. This takes approximately five minutes. If the environment does not change, the user can save the background training information for later use.

This face detector first determines whether a given image sub-window belongs to the face category. Variability in the face images may increase the complexity of the decision boundary to distinguish faces from non-faces. To reduce this variability, the input image is pre-processed (Fig. 5). During training and detecting phases, a pre-processing step[11] masks the background pixels from the face pixels with an ellipse mask (Fig. 6). It then attempts to equalize the intensity values across the face pixels. A linear function is factored into the intensity values contained in the window, then subtracted out and corrected for extreme lighting conditions. Next, histogram equalization is applied to correct for different camera gains and to improve contrast. For each of these steps, the pre-processing is computed based on pixels inside the ellipse mask.

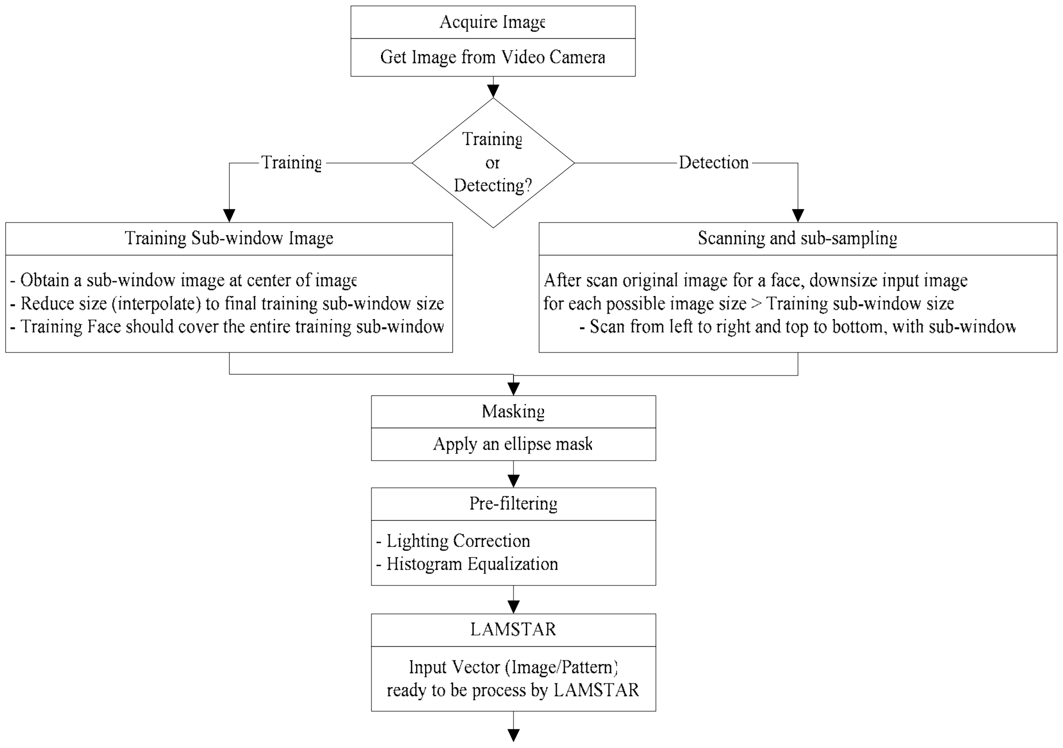

Acquire Image
Get Image from Video Camera
Training or Detecting?
Training
Detection
Training Sub-window Image
- Obtain a sub-window image at center of image
- Reduce size (interpolate) to final training sub-window size
- Training Face should cover the entire training sub-window
Scanning and sub-sampling
After scan original image for a face, downsize input image for each possible image size > Training sub-window size
- Scan from left to right and top to bottom, with sub-window
Masking
Apply an ellipse mask
Pre-filtering
- Lighting Correction
- Histogram Equalization
LAMSTAR
Input Vector (Image/Pattern) ready to be process by LAMSTAR

Figure 5: Flow diagram of the pre-processing stage

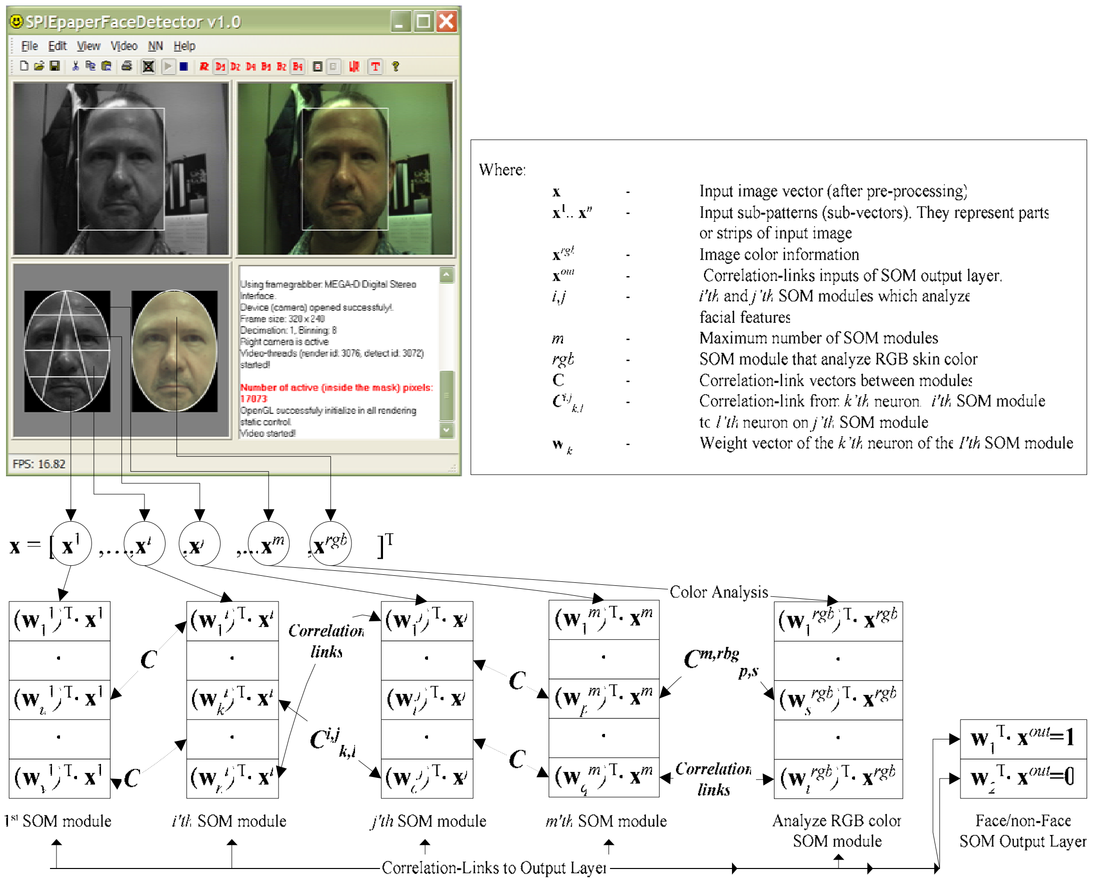
During the training process the system will train whatever face appears in the training sub-window in the image screen (Fig. 6). Using an interpolation technique, the training sub-window is first reduced in size to match the internal face detection sub-window size, currently 64 x 85. This dimension is the minimum face width and height the system is able to detect and track.

During the detection process, it is still able to detect a trained face that appears larger than the trained sub-windows size, using a sub-sampling technique applied to every location of the image-scene. The input image is repeatedly sub-sampled by a factor of 1.2.

Once the image sub-window is pre-processed and reduced to a determined input vector size, the image is divided in non-parallel strips that cover local features that might important for face detection (Fig. 6). Each non-parallel strip is a sub-vector (or sub-pattern) that is raw-fed into each SOM module. Currently there are 10 SOM modules, each with a number of outputs ranging from 5 to 30. There is also an output SOM layer inter-connected with all the other SOM modules. It only has two output neurons for the face and non-face case (Fig. 3 and Fig. 6).

In order to properly use a Kohonen SOM network, it is recommended that all weights and input vectors are normalized, therefore during training and detection phases, sub-patterns and SOM weights are mapped to [-1, 1] and their lengths set to 1.

During the face and background training, sub-patterns are present and each SOM module is trained individually. All weights are initialized with random low values (< 0.2) and C-links coefficient with 0, which means no correlation at all. Once the SOM output is obtained using the WTA technique, the C-links between SOM modules and the output layer are established. During detection, the system selects the winners of each SOM module and through the C-links and weights the output SOM layer should be able to determine whether or not a face is present.


SPIEpaperFaceDetector v1.0
File Edit View Video NN Help
FPS: 16.82
$\mathbf{x} = [\,\mathbf{x}^1, \ldots, \mathbf{x}^i, \mathbf{x}^j, \ldots, \mathbf{x}^m, \mathbf{x}^{rgb}\,]^T$
Color Analysis
$(\mathbf{w}_1^1)^T \cdot \mathbf{x}^1$
$(\mathbf{w}_k^1)^T \cdot \mathbf{x}^1$
$(\mathbf{w}_s^1)^T \cdot \mathbf{x}^1$
$(\mathbf{w}_1^i)^T \cdot \mathbf{x}^i$
$(\mathbf{w}_k^i)^T \cdot \mathbf{x}^i$
$(\mathbf{w}_r^i)^T \cdot \mathbf{x}^i$
$(\mathbf{w}_1^j)^T \cdot \mathbf{x}^j$
$(\mathbf{w}_l^j)^T \cdot \mathbf{x}^j$
$(\mathbf{w}_o^j)^T \cdot \mathbf{x}^j$
$(\mathbf{w}_1^m)^T \cdot \mathbf{x}^m$
$(\mathbf{w}_p^m)^T \cdot \mathbf{x}^m$
$(\mathbf{w}_q^m)^T \cdot \mathbf{x}^m$
$(\mathbf{w}_1^{rgb})^T \cdot \mathbf{x}^{rgb}$
$(\mathbf{w}_s^{rgb})^T \cdot \mathbf{x}^{rgb}$
$(\mathbf{w}_t^{rgb})^T \cdot \mathbf{x}^{rgb}$
$\mathbf{w}_1^T \cdot \mathbf{x}^{out}=1$
$\mathbf{w}_2^T \cdot \mathbf{x}^{out}=0$
C
Correlation links
$C^{i,j}_{k,l}$
$C^{m,rbg}_{p,s}$
1st SOM module
i'th SOM module
j'th SOM module
m'th SOM module
Analyze RGB color SOM module
Face/non-Face SOM Output Layer
Correlation-Links to Output Layer

Using framegrabber: MEGA-D Digital Stereo Interface.
Device (camera) opened successfuly!.
Frame size: 320 x 240
Decimation: 1, Binning: 8
Right camera is active
Video-threads (render id: 3076, detect id: 3072) started!

Number of active (inside the mask) pixels: 17073
OpenGL successfuly initialize in all rendering static control.
Video started!

Where:

$\mathbf{x}$	-	Input image vector (after pre-processing)
$\mathbf{x}^1 .. \mathbf{x}^n$	-	Input sub-patterns (sub-vectors). They represent parts or strips of input image
$\mathbf{x}^{rgb}$	-	Image color information
$\mathbf{x}^{out}$	-	Correlation-links inputs of SOM output layer.
i,j	-	*i'th* and *j'th* SOM modules which analyze facial features
m	-	Maximum number of SOM modules
rgb	-	SOM module that analyze RGB skin color
$\mathbf{C}$	-	Correlation-link vectors between modules
$\mathbf{C}^{i,j}_{k,l}$	-	Correlation-link from *k'th* neuron *i'th* SOM module to *l'th* neuron on *j'th* SOM module
$\mathbf{w}_k$	-	Weight vector of the *k'th* neuron of the *I'th* SOM module

Figure 6: Snapshot of the face detector system including the relationship between SOM models

5. IMPLENTATION DETAILS

For video input, the face tracking system uses a Videre Design MEGA-D video camera. Its native CCD resolution (ZR32112) is 1288 x 1032 and provides sub-sampling by binning (averaging) or decimation (removing) through its library developed by SRI International. It connects to the host using an IEEE1394 at 400Mbps. An important and vital feature of this digital camera is its sub-window capabilities. Once a face is found in the scene, the face detector prediction module sends a sub-window command to the video camera to confine the next scan to the portion of scene where the face is found. This substantially reduces the bandwidth used between the camera and the host computer and increases the throughput of the face detector.

The system runs on a Dell 530MP with an *Intel® Dual Pentium IV Xeon* @1.7GHz and 768MB RAMBUS.

The pre-processing is developing using the *Intel® Integrated Performance Primitive* (IPP) library and the vector products involve in the internal SOM programming the *Intel® C++ Class Libraries for SIMD Operations*. The compiler is the *Intel® C++ Compiler* v7. The analyzer tool is *Intel® VTune™ Performance Analyzer 6.1*. All libraries, classes and tools are optimizing for Pentium 4 and Xeon™ processor.

6. CONCLUSIONS AND FUTURE RESEARCH

This research is intended to serve as the core of a video-based, tetherless 3D head position and orientation tracking system targeted for either auto-stereoscopic or projection-based virtual-reality systems.

Although still in early stages of development, this face-detection approach is showing promising results. We have successfully trained the system to recognize a single face in less than 30 seconds and a background image in under five minutes. Before the prediction model is enabled, even in a controlled environment where lighting and background remain constant, it can take from 0.5-16 fps to detect a face. The slow rate of 0.5 fps is due to the location of the face in the scene and the necessary successive image sub-sampling at each pixel position in order to detect arbitrary face sizes. Once the prediction model is enabled, the system achieves a detection rate of 16 fps.

This system is also able to locate and track a face at varying distances from the video camera, with the size of the face ranging from 640 x 480 pixels (maximum full frame) to 64 x 85 pixels (minimum due to internal face detection sub-window size).

A simple linear extrapolation prediction module localizes the scanning area to where the face is likely to appear in the scene. It takes the position velocity between the last two frames and uses it to predict the subsequent frame. This scheme improves system performance from 0.5 fps to 16 fps. However, when the system fails to detect a face, the system has to rescan the entire scene.

The performance of face detection and tracking is fairly robust for up-right frontal faces, achieving 90% of *true positive* detection rate (i.e., the face is a face). Once the system detects and locks onto a face, it only fails if the person's head rotates or tilts too much (lack of proper face training) or makes sudden movements (lack of good prediction and/or distorted face cause by video smearing). Sometimes, the system also fails with slight combinations of rotation and tilt; this could be due to either a lack of thorough training on the part of the user or a problem in the system's network design.

During the early stages of system development, when the system was only trained for face detection in controlled environments, the system had an average of 50% *false positive* (i.e., the face was not a face). After adding background training (non-face cases), the system improved and false positive detection went to 0%.

Tests are inconclusive in a uncontrolled environment, where more people can be present and the system might lock in a face that is not trained user's face.

Future work includes:

- Create a user information database. Each time the system learns to identify a new user face, a new record with weights and coefficients can be added and saved for future use.
- Refine the sub-pattern and SOM module arrangements to achieve better detection and faster system response.
- Add additional detectors, each one that can be independently trained, to create redundancy and therefore better reliability and accuracy.
- Stress the system in an uncontrolled environment and try to reach 30 fps (with prediction enabled).
- Continue developing the 3D head position and orientation tracking system.
- Currently, when the prediction module fails, the system has to re-scan the entire image, lowering performance to 0.5 fps. New techniques are being developed that use the coordinates of the last box surrounding the face (the face sub-window), increase the box size and then re-scan that area. The box size is incrementally increased and re-scanned until the face is found.
- Implement a better prediction module to avoid decreasing the frame rate. Currently under development is a more sophisticated prediction module based on a neural network linear filter with a Tap Delay Line (TDL).
- Decrease the bottleneck caused by image sub-sampling at each pixel position by developing a better algorithm for detecting arbitrary face sizes.
- Try different video camera with higher sensitivity, faster shutter speed and therefore less smearing.

This face-detection system is being designed to detect specific faces, not arbitrary faces; therefore, it differs from most of the general face-detection research in the literature. It is designed for use in a virtual-reality system with known users as a tetherless replacement for a magnetic head tracker. It avoids generalization, and doesn't lock on unknown persons who enter a scene, enabling the user to reliably control the virtual environment. Future research will, however, compare this specific algorithm with more general efforts, to determine the benefits and bottlenecks of this approach.

ACKNOWLEDGMENTS

The virtual reality and advanced networking research, collaborations, and outreach programs at the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago are made possible by major funding from the National Science Foundation (NSF), specifically NSF Major Research Instrumentation awards EIA-0115809 and EIA-9871058.

The CAVE and ImmersaDesk are registered trademarks of the Board of Trustees of the University of Illinois. PARIS, AGAVE and Varrier are trademarks of the Board of Trustees of the University of Illinois.

REFERENCES

1. D. Graupe and W. J. Lynn, "Some Aspects Regarding Mechanistic Modeling of Recognition and Memory," *Cybernetica*, **vol. 12**, pp.119-141, 1969.

2. D. Graupe, *Principles of Artificial Neural Networks*, pp. 191-222, World Scientific Publishing Co. Pte. Ltd., Singapore, 1997.

3. H. Kordylewski, "A Large Memory Storage and Retrieval Neural Network for Medical and Engineering Diagnosis/Fault Detection," Doctor of Philosophy's Thesis, University of Illinois at Chicago, TK-99999-K629, 1998.

4. D. Graupe and H. Kordylewski, "A Large Memory Storage and Retrieval Neural Network for Adaptive Retrieval and Diagnosis," *International Journal of Software Engineering and Knowledge Engineering*, **vol. 8**, pp.115-138, 1998.

5. M. L. Minsky, "K-Lines: A Theory of Memory," *Cognitive Science*, No. 4, pp. 117-133, 1980.

6. S. Chang, D. Graupe and K. Hasegawa, "An Active Multimedia Information System for Information Retrieval, Discovery and Fusion," *International Journal of Software Engineering and Knowledge Engineering*, **vol. 8**, pp. 139-160, 1998.

7. T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, New York, Third Ed., January 2001.

8. S. Kaski and T. Kohonen, Winner-Take-All Networks for Physiological Models of Competitive Learning, Neural Networks, Vol. 7, Nos. 6/7, pp. 973-984, 1994.

9. V. Rao and H. Rao, *C++ Neural Networks and Fuzzy Logic*, MIT Press, Second Ed., 1995.

10. T. Masters, *Practical Neural Network Recipes in C++*, Academic Press, 1993.

11. H. A. Rowley, "Neural Network-Based Face Detection," Doctor of Philosophy's Thesis, Carnegie Mellon University, CMU-CS-99-117, May 1999.