Abstract

Computerized human face processing (detection, recognition, synthesis) has known an intense research activity during the last few years. Applications involving human face recognition are very broad with important commercial impacts. Human face processing is a difficult and challenging task: the space of different facial patterns is huge. The variability of human faces as well as their similarity and the influence of other features like beards, glasses, hair, illuminations, background etc., make face recognition or face detection difficult to tackle.
The main task during the internship was to study and implement a neural-network based face detection algorithm for general scenes, which has previously been developed within the IDIAP Computer Vision group. It also included the study and design of a multi-scale face detection method.
A face database and a camera were available to make tests and perform some benchmarking. The main constraint was to have a real-time or almost real-time face detection system. This has been achieved. Evaluation of the face detection capability of the employed neural networks were demonstrated on a variety of still images. In addition, we introduced an efficient preprocessing step and a new post-processing strategy to eliminate false detection significantly. This allowed to deploy a single neural network for face detection running in a sequential manner on a standard workstation.

Résumé

Pendant les dernières années, le traitement des visages humains assisté par ordinateur (detection, reconnaissance, synthèse) a connu une activté intense en recherche. Les applications qui impliquent la reconnaissance des visages humains sont très répandues et importantes d'un point vue commercial. Le traitement des visages humains est une tâche difficile et en même temps un défi: l'espace des formes de visages est enorme. La variabilité des visages humains et aussi leur similitude et l'influence d'autres caractéristiques comme la barbe, les lunettes, cheveux, l'illumination, l'arrière-plan, etc. rend la reconnaissance et la détection des visages difficile à réaliser. La tâche principal a accomplir pendant le stage était l'étude et l'implementation d'un algorithme de détection des visages qui était développé dans le groupe de vision de l'IDIAP. Il est basé sur un réseau de neurones et appliqué dans des scènes générales. De plus, l'étude et l'implémentation d'une methode de détection à plusieurs niveau faisaient partie du projet.
Une base de donné des visages et une caméra etaient disponibles pour faire des tests et des mesures. La contrainte principale était d'avoir un system de détection des visages qui tourne en temps-réel. Ce but était accompli. L'évaluation de la capacité de détection des visages des réseaux de neurones employé fut demontré sur une variété d'images. De plus, nous avons introduit une étape de pré-traitement éfficace et une nouvelle stratégie pour effectivement éliminer des détections erronés. Ceci a permis d'employer un seul réseau de neurones pour faire la détection des visages s'effectuant de façon séquentielle sur une station de travail standard.

Project Summary

The project evolved over several stages. First, a simple neural network has been developed for recognizing faces of some people recorded in a small database. Then, a simple face detection neural network has been implemented, based on a classical approach using sliding windows to parse a given test image for faces. The next step consisted of building a *Fast Still Image Face Detector* based on a new approach using the Fast Fourier Transform in order to speed up the execution time. Tests on a variety of images were performed to compare the usability of several neural network pre- and postprocessing strategies. The code of the still image face detector has been almost completely rewritten and much of the computation intensive routines could be precalculated off-line. The resulting *Moving Picture Face Detector* runs in real time and this in a purely sequential form on a single workstation. During a demonstration, where several people presented themselves before the camera, the system detected all of the faces correctly.
Both the still and the dynamic face detector evolved further into advanced versions running finally almost three times faster (full image search) and producing more stable results than the initial versions. Several methods and techniques found their way into the final implementations and contributed to increased reliability of the whole system.

At the center of face detection is the classification of faces and non-faces. A neural network can handle this task after being trained on a variety of face and non-face images. The successful run of a face detector system depends on several points. First there is the neural network training which determines to what extent a neural network is capable of detecting faces and under which conditions. The question arises how one should train the neural network optimally so that it stays generic in face detection but still can distinguish faces from everything else in a reliable fashion.This is one of the main problems encountered with face detection, how does a non-face look like? Further more how should the training sets be set up, what should be their optimal size and content, how long should one train the system, in what order should the training material be presented to the neural network, how should it be preprocessed?

Of paramount importance is also the postprocessing step taking place at the output of the neural network. Its aim is to reduce the number of false detections made by the neural network. It is based on the fact that faces normally produce a variety of multiple detections located just a few pixels away from each other and that in general other objects don't show this behaviour when the neural network was trained on faces beforehand. A method to address this problem has been developed.

The goal of the project, namely creating a system that can detect faces in real-time, has been achieved. Depending on the scales, resolutions, preprocessing, and neural network configuration, the face detection system deliverers frame rates of 1 to up to 5 frames per second running in a purely sequential version on a workstation equipped with a camera and a frame grabber.

Testing of the reliability of the Moving Picture Face Detector was done in an office environment with stable lighting conditions. Almost all faces were detected immediately and independent of gender, beards, and glasses. The distance between the camera and the object depends on several parameters such as the sliding-window size, but detection is in general possible at a distance between 0.3 m and about 3m from the camera.

Table of Contents

Fast Multi-Scale Face Detection

1 Introduction

Face detection is the fundamental step before the face recognition or identification procedure. Its reliability and time-response have a major influence on the performance and usability of the whole face recognition system [1].

In [1] a new approach has been introduced, that can speed up processing time by considering a MLP (Multi Layer Perceptron) as a bank of filters and also by reformulating the processing steps in terms of cross-correlations.

- The characteristic of this approach is that it reduces considerably the computation time (speed-ups of a factor of 8 to 14, depending on the image sizes) while maintaining identical performances to classical MLP based detectors.
- Further on, a speed-up can be gained when considering multiple scales face detection at different resolutions, due to a property of the Fourier transform that implies that sub-sampled images can directly be obtained using the original FT.

1.1 Goal of the Project

The aim of this project was the implementation of the proposed algorithm, introducing it into a real-world application, where a scene is searched for faces in real-time. Face detection has to be done in a complex background without using colour or motion information.

1.2 Applications

Applications of face recognition arise in various situations. But before the actual identification can take place, a face must first be located in a given scene. Following applications are of interest:

- **Image/Movie database search, (Video Indexing):** Every year, improved technology provides cheaper and more efficient ways of storing information. However, automatic high-level classification of the information content is very limited; this is a bottleneck that prevents media technology from reaching its full potential [2]. A system using face detection would allow a user to make queries of the form: “Which scenes in this video contain human faces?” and in combination with a face recognition system even questions like “In what scene is a certain person present?” could be answered by the system.
- **Personal Identification, (Recognition):** Credit cards, Driver’s License, Passports
- **Bank / Store Security(Recognition):** Automatic surveillance cameras
- **Advanced Human-Machine Interface**
- **Human Expression Analysis**

1.3 Demo Set-Up

Figure 1 shows a demonstration set-up of the moving picture face detector where a person's face is being captured by a camera. A system connected to it detects in real-time the face and marks its position with a square, as can be seen on the monitor.

FIGURE 1. The Face Detection System

1.4 Report Overview

The next chapter gives an overview of the *state-of-the-art* in face detection. Several approaches can be found in the literature, basically probabilistic and connectionist ones. The latter ones deploy neural networks for classifying faces and non-faces.

Further we present the *classical neural network for object detection* using a sliding window to parse a given input image. It will be compared to the new *fast neural network for object detection* based on cross-correlations computed in the frequency domain using the Fast Fourier Transform.

The *still image face detector* incorporates the fast neural network for object detection and served as a bases for the development of the *moving picture face detector*. A difference must be made though between those two: the latter one uses a camera to detect objects, so we do have real-time constraints. On the other hand, the still image face detector version handles - as its name suggests - still images, here we have plenty of time at disposition and therefore we can employ for example sophisticated combinations of several networks to increase the performance on detections.

Both approaches evolved in parallel and during development profited from each other. The still image face detector used for example the main execution kernel of the neural network while the most recent post-processing techniques developed in the still image face detector could be implemented into the moving picture detector.

Both the still image and the moving picture detector depend on properly trained neural networks. In the *Neural network training* chapter the training process is being described in detail. Several methods can be applied in order to decrease neural network errors.

A *discussion* of the obtained results in previous chapters as well as a comparisons to those obtained by another research group follows. Finally the whole project is reviewed and an outlook chapter presents on how the current project could evolve further.

2 State-of-the-Art in Face Detection

An abundance of approaches addressing face detection and face recognition (often coupled) can be found in the literature. Each has its advantages and disadvantages, making it difficult to choose a superior technique.

Among a multitude of approaches, there are two important ones, namely the probabilistic and the connectionist one. Also there can be found some falling in between, namely having basically a probabilistic nature, but using neural networks for later classification purposes.

2.1 Probabilistic Approaches

2.1.1 Discrete Markov Processes

1. A. J. Colmenarez et al. [8] presented a visual learning technique that maximizes the discrimination between positive and negative examples in a training set. It was applied in the context of face detection using a family of *discrete Markov processes* to model the face and background patterns and estimate the probability models using data statistics. The detection process was carried out by computing the likelihood ratio using the probability model obtained from the learning procedure (the one that optimizes the discrimination between the two classes in the training set).

2.1.2 Eigenfaces

2. M. Turk and A. Pentland [7] describe a detection system, that uses similarly to the one mentioned before “eigenfaces”. It functions by *projecting face images onto a feature space* that spans the significant variations among known face images. The significant features “eigenfaces” of faces are the eigenvectors (principal components) of the set of faces; these do not necessary correspond to features as eyes, ears and noses. The projection operation characterizes an individual face by a weighted sum of the eigenface features, and so a particular face can be recognized by *comparing those weights* to those of known individuals. A non-face is assumed at the input, if the weighted sum is too far away from the “face space.” The method is based on an information-processing model, it can be implemented using three layer neural networks. The symmetric weights between the hidden layer and the input, respectively the output, are the eigenfaces. The hidden units themselves reveal the projection of the input image onto the eigenfaces.The output is the face space projection of the input image.

3. Another probabilistic approach presented by B.Moghaddam and A. Pentland [12] is based on *density estimation* in high dimensional spaces using the eigenface decomposition. Two types of density estimators are derived for modelling the training data using either a multivariate Gaussian or a mixture-of-Gaussian models. With these probabilities one can then formulate a *maximum-likelihood estimation* framework for visual search and target detection for automatic object recognition and coding. Computationally this is achieved by sliding an m-by-n observation window throughout the image and at each location computing the likelihood that the local subimage X is an instance of the target class Ω - i.e. $P\langle x|\Omega\rangle$, After this probability map is computed, the location corresponding to the highest likelihood is being selected as the ML estimate of the target location.
   The described approach exploits the intrinsic low-dimensionality of the training imagery to form a computationally simple estimator for the complete likelihood function of the object. It can be applied to probabilistic visual modelling, detection, recognition and coding of human faces and non-rigid objects such as hands.

2.2 Semi-Probabilistic / Connectionist approaches

4. K. S. Sung and T. Poggio [5] came up with a technique modelling the distribution of human face patterns by means of a few view-based “face” and “non-face” *prototype clusters* (six face and six non-face prototypes delivered good results). At each image locations, a difference feature vector is computed between the local image pattern and the distribution-based model. A trained classifier (a neural network has been applied) determines, based on the difference feature vector, whether or not a human face exists at the current image location. This means that an image pattern is classified as “a face” if its distance from the sub-space of faces is below a certain threshold, according to an appropriate distance metric.

2.3 Connectionist Approaches

2.3.1 Feed-Forward Neural Networks

5. H. A. Rowley et al. [2], [3] presented face detection systems based on *retinally connected neural networks*. Through a sliding window, the input images are being examined after going through and extensive preprocessing stage. The systems arbitrates between multiple networks to improve performance over a single network. The bootstrap algorithm was employed to add false detections into the training set. Simple heuristics, such as using the fact that faces rarely overlap in images improved the accuracy of their detection systems. In one version, preprocessing of the image delivered by the sliding window is done by a so called router network which is able to determine the orientation of the input image and to turn it into an up-right position, so that the following-up detector networks have a consistent input. Post-processing of the neural networks are performed by either adding / oring the outputs or using an additional neural network to arbitrate between the outputs.
   A faster version of the above described detector used differently trained networks that parsed a given input image not pixel by pixel but in larger steps, using slightly enlarged and overlapping sliding-windows. In a second scan, interesting positions (containing faces with a high probability) were then scanned by another network on a pixel by pixel base.

2.3.2 Recurrent Neural Networks

6. S. Oka et al. [9] employed a preprocessing system that extracted face parts such as eyes and mouths from an input image. These were then fed into a recurrent neural network composed of a non-linear oscillation system. The system is able to detect several faces one after another by composing face parts extracted by the preprocessing step. It works well even when some face parts are lost.

2.4 Comparisons

When comparing probabilistic and connectionist approaches, one must be very careful, because the authors used different databases for both the training processes and the detection evaluations later on. There are other differences that can have an important influence on the final results, such as for example different sliding window sizes deployed during the tests. The comparison below can therefore only be regarded as a prediction of the author.

2.4.1 Key Features

The above described approaches are based on following key features:

Approach	Sliding Window	Preprocessing Methods	Type	Scale Ratio
Colmena-rez, 1997	11 x 11 pixel	Histogram equalization	Holistic	sq(2)
Turk, 1991	Sliding Win.	No preprocessing	Holistic	-
Moghad dam, 1995	Sliding Win.	No preprocessing	Holistic	-
Sung, 1994	19 x 19 pixel	Background mask, histogram equalization, gradient	Holistic	-
Rowley, 1997	20 x 20 pixel	Linear fitting, histogram equalization, background mask	Holistic	1.2

TABLE 1. Methods applied in the literature and their Key Features

2.4.2 Strong / Weak points

Approach 1: With the discrete nature of the models applied, the detection process is by almost two orders of magnitude less computationally expensive than neural network approaches, however, the presented method showed no improvements in terms of correct and false detections. The system produced more false detections than were observed with the purely connectionist approaches, see [8].

Approach 2: Using spatial temporal filtering, two to three images per second could be processed, the detection and recognition processing was done on two computers connected with each other and running in parallel. Although the results reported are quite good, it is unlikely that this system is as robust as approach 4, because Pentland's classifier is a special case of Sung and Poggio's system, using a single positive cluster rather than six positive and six negative clusters [2]. The computational requirements of the current approach is high since every tested sub-window has to be projected onto the eigenspace.

Approach 3: In comparison to the similar approach 4, there where no neural networks applied for classification, but the maximum likelihood function was evaluated directly on the input data.

Approach 5: The detector consisting of both extensive preprocessing (histogram equalization, best fit linear functions, masks for ignoring the background around faces) and post-processing (arbitration between multiple neural networks) and heuristics (no overlapping) showed to perform very well. The question is if it is not too specialized just for face detection (already by its retinally connections that favour the typical face feature locations)

Approach 6: This approach is quit different from the others, since it can detect partly obscured faces using a not clearly described preprocessing strategy. Also there are no detection results at disposition, so it is difficult to compare it’s performance.

3 Neural Networks

Work on artificial neural networks, commonly referred to as “neural networks”, has been motivated right from its inception by the recognition that the brain computes in an entirely different way from the conventional digital computer [20].

Both the *Still Image Face Detector* and the Moving *Picture Face Detector* that will be presented in further chapters are based on fully connected three layer feed-forward neural networks that employ a sigmoid activation function, also called squash function. See Figure 2.

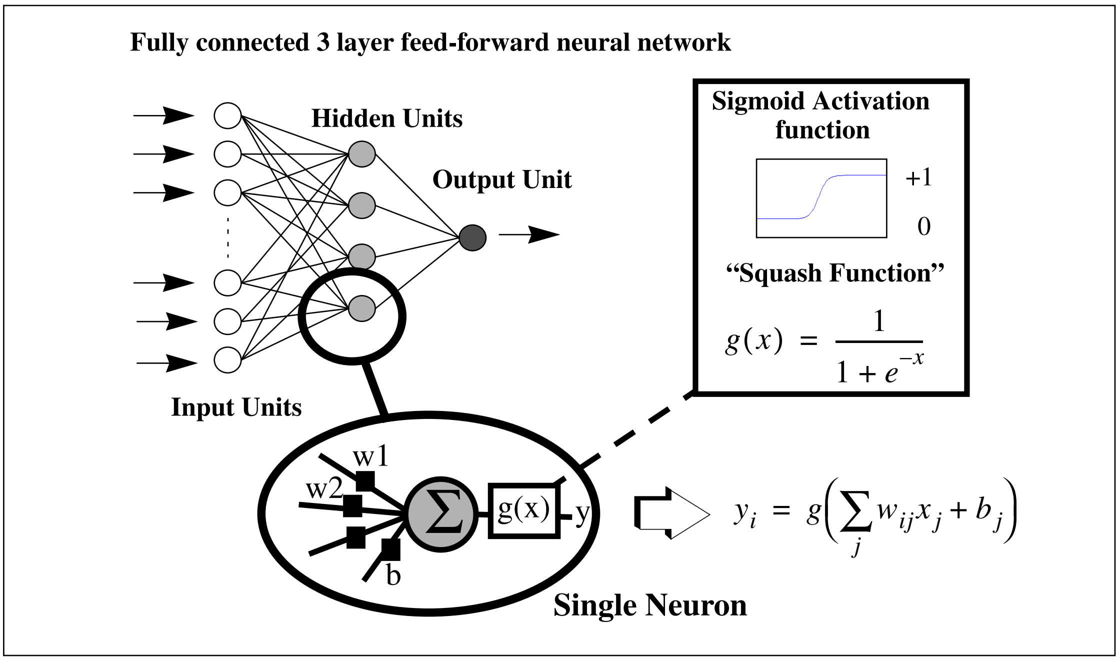


FIGURE 2. Neural Network, Neuron and Activation function

A neural network is built-up on neurons, which are the basic information treating units. These do a weighted linear combination of their inputs and pass the sum through an activation function of sigmoid type, which acts as a switch and propagates a given input activation further or suppresses it.

In order to be able to detect faces, a neural network must first be trained to handle this task. Figure 3 shows two examples of images, a neural network must learn to distinguish. On the right hand side we see a sample face and on the left hand side a sample non-face.

FIGURE 3. Training Images of Size 25 x 25

Clearly, it is difficult to define good representatives of the so called non-face space, as the latter one is huge. It consists of all possible combinations of pixels at different lighting conditions found in a given image. In the real world, this non-face space would be infinitely large, which is not the case here, as we do use grayscale pictures of 25 x 25 pixels size featuring discrete luminance values (integers). But nonetheless, all combinations possible represent an incredible large number. As we cannot have a training databases of that size and have neither the time at disposition needed for a complete training on all variations. Instead we do the following: using an initial non-face training base that represents important variations and introducing additional non-face images yielded by the so called *bootstrap* algorithm.

3.1 Training Techniques

Following techniques were applied for the training of our neural networks:

3.1.1 Back-Propagation Algorithm

Among the algorithms used to perform supervised learning, the back-propagation algorithm has emerged as the most widely used and successful algorithm for the design of multilayer feedforward networks. There are two distinct phases to the operation of back-propagation learning: the forward phase and the backward phase. In the forward phase the input signals propagate through the network layer, eventually producing some response at the output of the network. The actual response so produced is compared with a desired (target) response, generating error signals that are then propagated in a backward direction through the network. In this backward phase of operation, the free parameters of the network are adjusted so as to minimize the sum of squared errors [20].

3.1.2 Boot-Strap Technique

A now well established technique (first applied in [5]) is the so called bootstrap technique. It allows to add non-face images to the training database that are of importance with regard to improve the neural networks performance at a given time during the training period. Additional non-face images are being gained by scanning images containing no faces, automatically clipping false detections and inserting these into the current training set, then resuming the training at the point where it stopped beforehand. As expected, the networks false detection rate diminished greatly in our experiments.

3.1.3 Multiple Similarly Networks

The performance of a trained neural network depends not only of the number and kind of images it was trained on, but also in which order these were presented during the training (best is to choose in a randomly order in which images to train on). Further on, a neural network with a different initial distribution of its weights will learn differently on exactly the same training set. This is due to the fact the back-propagation algorithm finds sub-optimal solutions. Therefore with a given initial representation of weights, a neural network finds a local minimum. Another network would find different minimum.

So in order to obtain networks that perform differently on a given input, one must alter the initial seed distribution of their weights as well as randomly present images during the training period.

3.1.4 Neural Network Training

Our *Neural Network Trainer* uses three different kind of data bases: The *Training Base*, the *Clipper Base* and the *Test Set*. The first one is of variable size and contains both face and non-face images (non-face images are being added during the training process by the bootstrap technique). The Clipper Base consists of images that don't contain faces and which serve as a source of new non-face images that get clipped out to be added to the Training Base during the neural network training process. The Test Set contains images that contain faces, needed to compare the momentary performance of the neural network.

The neural networks employed in the face detectors were trained by applying the back-propagation algorithm using both a fixed learning step and momentum. Fixed values may be not optimal, but simplified the implementation of the neural network trainer considerably.

Figure 4 shows the data flow during the training of the neural networks. Basically there are four parts: the actual training is based on the back-propagation algorithm, the test set evaluation and the train set evaluation part where the test images are scanned for faces and evaluated on the number of correctly detected faces $\lambda_{Correct}$, missed faces λ_{Missed} and false detections λ_{False}. The evaluation tuple $e(\lambda_1, \lambda_2)$ must be maximized in order to get networks that behave optimally:

$$e(\lambda_1, \lambda_2) = \left(\frac{\lambda_{Correct}}{\lambda_{Missed}}, \frac{\lambda_{Correct}}{\lambda_{False}}\right) \qquad \text{(EQ 1)}$$

Note that false detections are in general more frequent than missed faces, that's why Equation 1 contains two separate ratios to be evaluated.

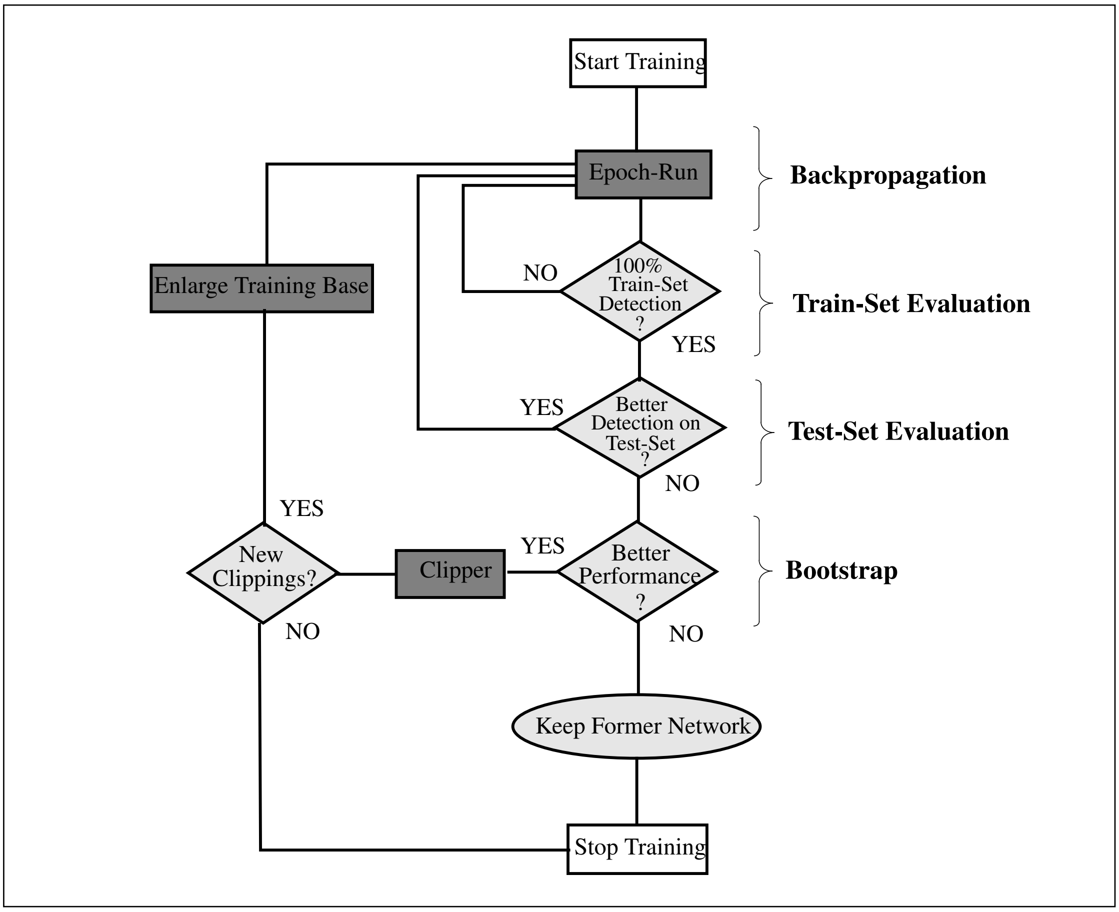


FIGURE 4. Neural Network-Training

Training of the neural network is done in *Epochs*. An Epoch represents the complete run of the network trainer through all images (faces and non-faces) contained in the training data base. The latter ones are structured in *Training Interval*s, meaning the number of faces or non-faces that follow up in a training sequence. An intervals of one is good, that means presenting a face, then a non-face, then again a face and so on. Even better is to choose randomly faces and non-faces. This way we prevent the neural network from concentrating too much on certain features.

3.2 Training Database

The face images contained in the training set were all of up-right frontal type. The training database featured also slightly rotated faces as well as some taken at closer distances. This improves the robustness of face detection process.

Having highly representative images in the training database is of paramount importance. Figure 5 depicts some bad face examples. These either don't show all face features (missing eyes for example) or then again show to much background which belongs to the non-face space and shouldn't be present in the face training set in order to have a maximum separation of the two classes.

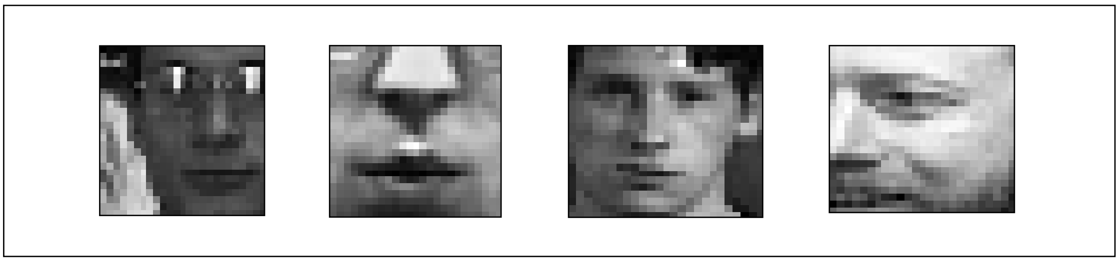


FIGURE 5. Bad Training Examples

Training images have an influence on the performance of face detection. Figure 6 shows an example of a false detection that occurred because the image on the left hand side (a face taken at a too close distance) was present in the training set. After removing it, this kind of false detections disappeared.

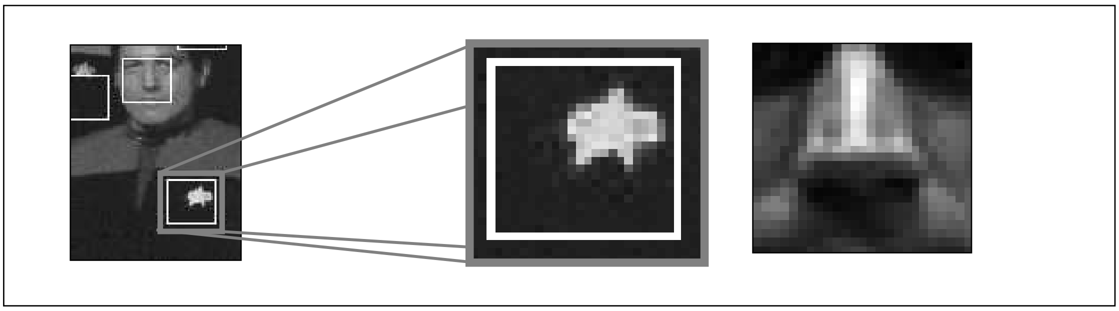


FIGURE 6. Impact of Bad Training Images

Our main training database consisted of about 1500 face examples and 4000 non-face examples. Additional non-faces were introduced by applying the bootstrap algorithm.

On the left hand side of Figure 7 is shown the sum of errors of the hidden and output units during network training. One can see in the center of the figure the effect of a bootstrap operation that introduced new clipping images (non-faces) into the training data base. It was started when 100% of the training set was correctly detected. New clippings in the training base lead to increased errors, because the network must adapt to the new data and in doing so produces also more errors on images it was previously trained on.

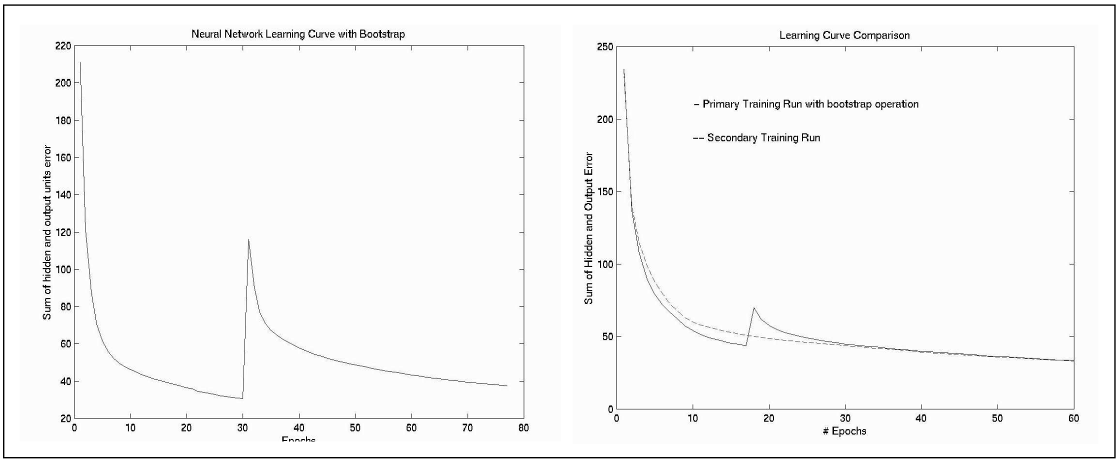


FIGURE 7. Hidden Units Error

Error recovering is rather fast though, but later on, errors drop slower than before. This is because the network has lost some of its flexibility due to an adaptation on the former training base. On the right hand side we see the error rates of two networks using the same training base. The first one was trained on an initial training base which got enlarged by the bootstrap algorithm. Training a second neural network on this combined training base leads to a similar adaptation on the training data as suggests the asymptotic convergence of both curves.

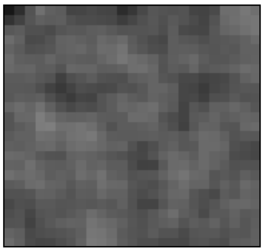
3.3 Training Risks

There are some risk one must try to avoid when training neural networks, among these are:

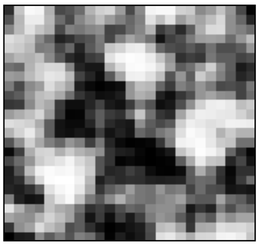
- **Over-training**: If one trains a network too long, it starts to learn by heart a given training set and behaves worse than a less trained network when being encountered with some unknown input data.
- Another issue is how to **schedule** training data, that means, how training data should be presented to the neural network. Best results can be obtained by randomly choosing faces and non-faces.
- **Forgetting** happens with large databases where quite some time passed by before a certain feature is represented to the network, so, what has been learned before is forgotten in the meantime. One may avoid this by trying to keep in the training base only examples that are relevant at a given time. Especially when applying the bootstrap algorithm, the training base grows during following-up iterations, introducing more and more non-faces. This threatens the balance between the number of face and non-face examples contained in the training base. The so called boosting algorithm can provide a solution to that problem, see also Section 10.5 on page 63.

3.4 Hidden Weights Visualization

Two weight matrices of a trained neural network are depicted in Figure 8. These can look faintly like faces, featuring parts that represent eye and mouth locations as can be seen on the left hand side, or round forms representing head contours, as can be seen on the right hand side.



Hidden Weight Nb 14
Neural Net with 15 Hidden Units



Hidden Weight Nb 6
Neural Net with 15 Hidden Units

FIGURE 8. Weight Visualization

4 Neural Network for Object Detection

In this section we focus on the classic solution for face detection based on the parsing of a given input image with a sliding window.

Such a still image face detector, is depicted in Figure 9, and works as follows: a given gray-scale input image is scanned pixel by pixel with a sliding window passing through it and which represents also the input that is being fed to a neural network after pre-processing in order to remove much of the variations due to various lighting conditions and camera characteristics.

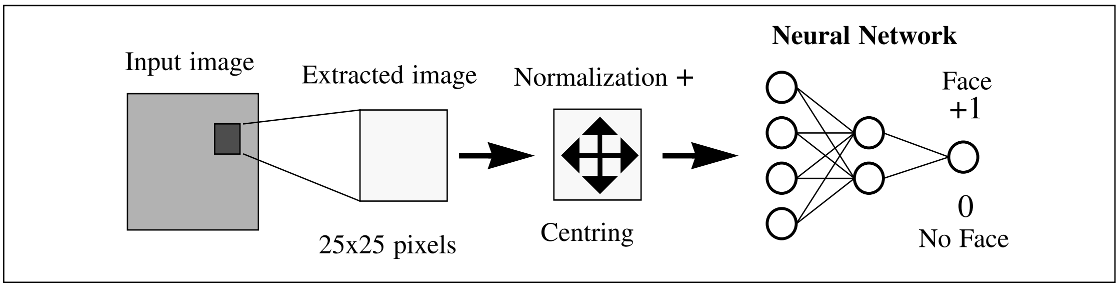


FIGURE 9. System for the detection of faces in a still image

Further on, contrast enhancement methods such as histogram equalization may be applied in addition. Finally, the neural network decides whether there is a face present at the momentary position of the sliding window or not.

4.1 Theory

Below is shown a neural network with an image as its input. Every pixel of the input image connects to an input unit. The rectangles depict the weight matrices W1 of the input and W2 of the hidden units respectively, see Figure 10.

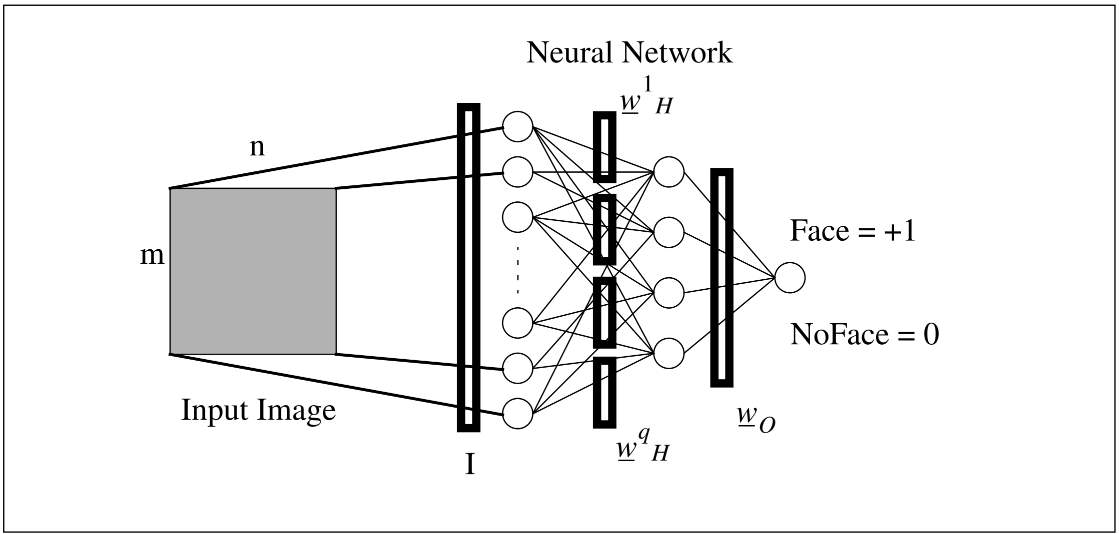


FIGURE 10. Classic MLP architecture

The activity of the hidden units and the output unit are given by:

$$\underline{h} = g(W_H I + \underline{b}_H) \quad \text{(EQ 2)}$$

$$o = g(\underline{w}_O \underline{h} + b_O) \quad \text{(EQ 3)}$$

$$\text{where } [W_H(i, j) = \underline{w}^i_H(j)]_{i \in 1.q,\, j \in 1.mn} \quad \text{(EQ 4)}$$

Here, g represents the squash function, q the number of hidden units and I the input image in vector form. As the outputs of the neural network are in [0,1] we apply a sigmoid function:

$$g(x) = \frac{1}{1 + e^{-x}} \quad \text{(EQ 5)}$$

The activity of a particular neuron i in the hidden layer H can be written as:

$$\underline{h}(i) = g\left(\sum_{j=1}^{mn} W_H(i, j) I(j) + \underline{b}_H(i)\right) \quad \text{(EQ 6)}$$

Similarly, the output activity is:

$$o = g\left(\sum_{j=1}^{q} \underline{w}_O(j) \underline{h}(j) + b_O\right) \quad \text{(EQ 7)}$$

Where q denotes the number of hidden units.

4.2 Fast Classic Neural Network

The above described method performs an exhaustive search, passing the sliding window pixel by pixel through the input image. An alternative consists in scanning the test image only at certain intervals, let's say for example every three pixels along the rows and likewise for the columns. A considerable speed-up can be gained this way, see Figure 11. Comparison is made to the FFT approach that will be introduced further below.

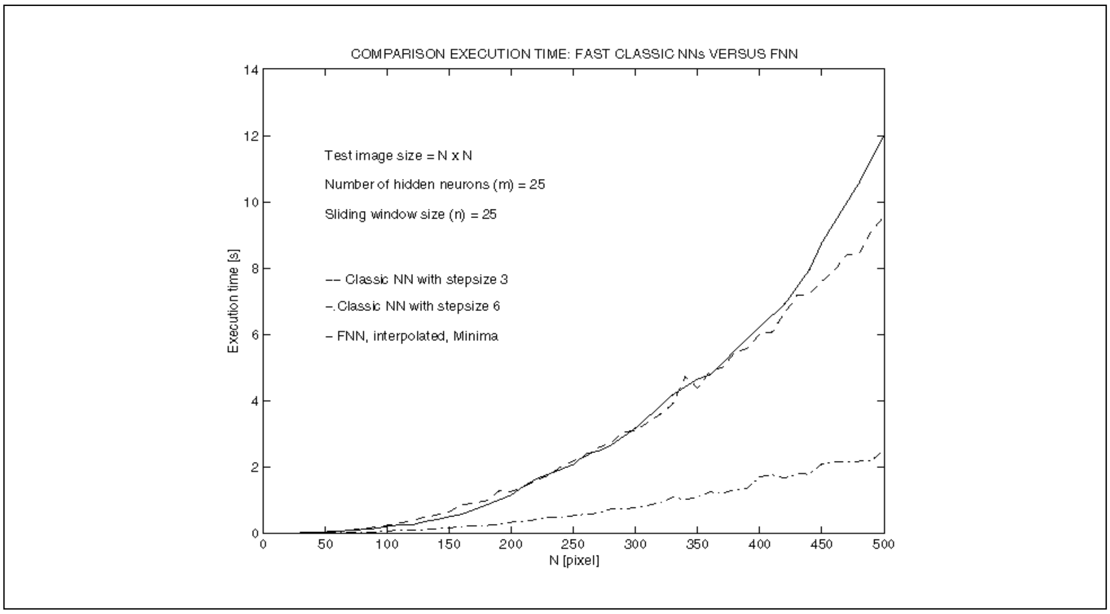


FIGURE 11. Comparison FNN / CNN with stepsize 3 and 6

There is a mayor drawback though with this strategy: scanning is not performed in every possible image location, therefore faces may get overlooked. In addition, multiple detections, that can help greatly reducing false detections, as will be shown, occur less often.

5 Fast Neural Network for Object Detection

In contrast to the classic neural network presented above, this approach is based on two dimensional cross-correlations that take place between the test image and the sliding window, itself represented by the neural network weights situated between the input units and the hidden layer. These cross-correlations can be represented by a product in the frequency domain. By using the Fast Fourier Transform, speed-ups in the order of a magnitude can be gained.

In the next few sub-sections we present several approaches that differ in the way the input images are being preprocessed. First there's only a rescaling of the input values, followed by a centred version and finally we address vector normalized / standard deviation normalized versions. The theory is presented in sections 5.1, 5.2, 5.5 and is based on the work done in [1]. For an explanation of the symbols used, have a look at the "Table of Symbols and Abbreviations" on page 65.

5.1 Basic Algorithm

During the detection step, a sub-image of size *m x n* (sliding window) is extracted from the test image of size *S x T* and fed to the neural network.

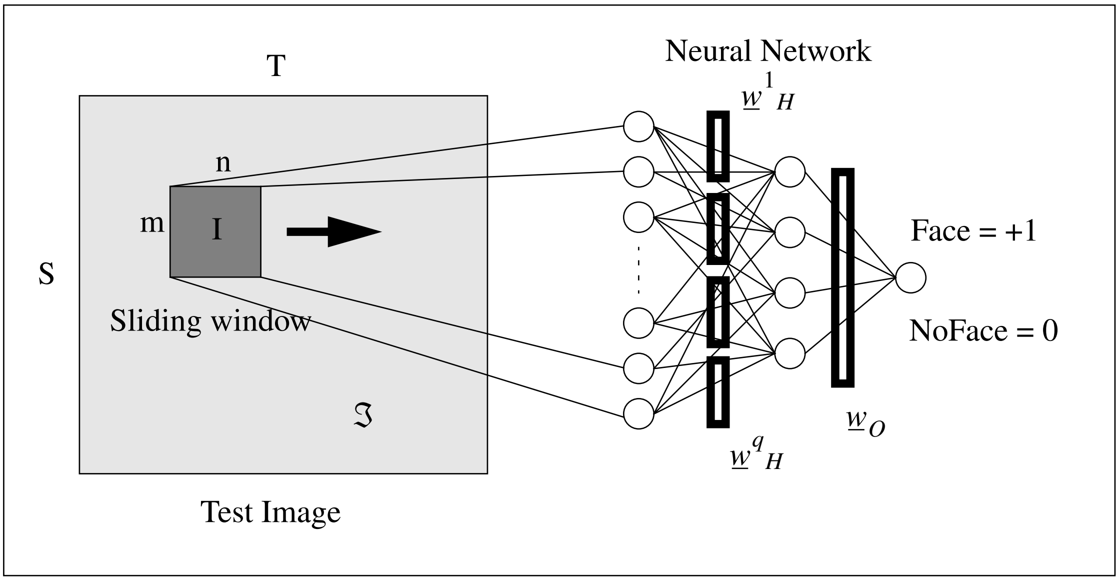


FIGURE 12. MLP architecture for object detection

Let $\underline{w}^i_H$ be the vector of weights needed to compute the activity of the hidden neuron *i*. This vector is of size mn an can be represented as a $m \times n$ matrix Φ_i. Equation 6 in a 2D space will look like:

$$\underline{h}(i) = g\left(\sum_{j=1}^{m}\sum_{k=1}^{n}\Phi_i(j,k)I(j,k) + \underline{b}_H(i)\right) \qquad \text{(EQ 8)}$$

The expression is obtained for a particular sub-image *I*. We can extend Equation 8 to the global input image $\Im$.

$$H_i(r,c) = g\left(\sum_{j=-m/2}^{m/2}\sum_{k=-n/2}^{n/2}\Phi_i(\dot{j},k)\Im(r+j,c+k)+B^i{}_H\right) \quad \text{(EQ 9)}$$

Note that the origin of Φ_i is in the center of the matrix. By posing $a = -j$ and $b = -k$ we obtain

$$H_i(r,c) = g\left(\sum_{\zeta=m/2}^{-m/2}\sum_{\vartheta=n/2}^{-n/2}\Phi_i(-\dot{a},-b)\Im(r-a,c-b)+B^i{}_H\right) \quad \text{(EQ 10)}$$

which contains a 2-dimensional convolution and can be written as:

$$H_i(r,c) = g(\Phi_i((-\dot{a},-b)\lozenge(\Im(\dot{a},b)+B^i{}_H))) \quad \text{(EQ 11)}$$

In general the following property holds:

$$f \lozenge g(-t) = f \otimes g \quad \text{(EQ 12)}$$

where the first operation denotes a cross-correlation[1]operation, and the second one a convolution. Therefore we can reformulate Equation 11 as

$$H_i = g(\Phi_i \otimes \Im + B^i{}_H) \quad \text{(EQ 13)}$$

$$\text{where } B^i{}_H(r,c) = b_H, \forall(r,c) \in [S-m+1, T-n+1] \quad \text{(EQ 14)}$$

H_i is the activity of the hidden unit *i* and $H_i(r,c)$ is the activity (or output) of the hidden unit *i* when the observation window (sliding window) is located at position (r, c).

Similarly, the final output of the neural network can be expressed by a linear combination of the hidden units activity:

$$O = g\left(\sum_{i=1}^{q}\underline{w}_O(i)H_i + B_O\right) \quad \text{(EQ 15)}$$

$O(r,c)$ is the output of the observation window located at the position (r, c) in the input image $\Im$.

---

1. Cross correlation: $f(x,y) \otimes g(x,y) = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} f^*(m,n)g(x+m,y+n)$, here *m* an *n* are limited to the sliding window size, outside these are zero. Note that both the input image and the weight matrices have real values.

Now, the above given cross-correlation can be calculated in terms of a Fourier Transform[1]:

$$\Im \otimes \Phi_i = F^{-1}(F(\Im) \bullet F^*(\Phi_i)) \qquad \text{(EQ 16)}$$

Evaluating this cross-correlation using the Fast Fourier Transform, an important speed-up can be gained in comparison to a classic neural network based on sliding windows, see Figure 18.

5.2 Vector Length Normalized Input Version

In order to get meaningful and stable results using a face detector in various lighting conditions, normalization of the input images must be done. This has to be done locally over the sliding window and not globally over the whole input image, as otherwise values may not be normalized / centred in the sub-windows. As the Fourier transform is a global process, the local centring must be reformulated, adapted to the whole Fourier Transform framework.

Let $\overline{[X]}_{rc}$ be the zero-mean normalized sub-image located at *(r,c)* in the input image $\Im$:

$$\overline{[X]}_{rc} = \frac{[X]_{rc} - \bar{x}_{rc}\mathbf{1}_{mn}}{\left\|[X]_{rc} - \bar{x}_{rc}\mathbf{1}_{mn}\right\|} \qquad \text{(EQ 17)}$$

where $\bar{x}_{rc}$ is the mean value of the sub-image located at *(r,c)* and $\mathbf{1}_{mn}$ a *m x n* matrix where every element is 1. We are interested in computing the cross-correlation between the sub-image $\overline{[X]}_{rc}$ and the filter Φ_i that is:

$$\overline{[X]}_{rc} \otimes \Phi_i = \frac{[X]_{rc} \otimes \Phi_i - \bar{x}_{rc}(\mathbf{1}_{mn} \otimes \Phi_i)}{\left\|[X]_{rc} - \bar{x}_{rc}\mathbf{1}_{mn}\right\|} \qquad \text{(EQ 18)}$$

$$\text{where } \bar{x}_{rc} = \frac{1}{mn}\sum_{j=1}^{m}\sum_{k=1}^{n}[X(j,k)]_{rc} = \frac{[X]_{rc} \otimes \mathbf{1}_{mn}}{mn} \qquad \text{(EQ 19)}$$

$$\text{and } \left\|[X]_{rc} - \bar{x}_{rc}\mathbf{1}_{mn}\right\| = \sqrt{\sum_{i=1}^{mn}(x_{rc}(i) - \bar{x}_{rc})^2} = \sqrt{\left(\sum_{i=1}^{mn} x_{rc}^2(i)\right) - mn\bar{x}_{rc}^{\ 2}} \qquad \text{(EQ 20)}$$

1. Fundamental property: $A \otimes B = F^{-1}(F(A) \bullet F^*(B))$ A and B are matrices, the multiplication on the right hand side is not a matrix multiplication but is done element-wise.

$$\text{also} \sum_{i=1}^{mn} x_{rc}^2(i) = [X^2]_{rc} \otimes \mathbf{1}_{mn} \tag{EQ 21}$$

Combining (18), (19), (20), (21) we get the expression

$$\overline{[X]}_{rc} \otimes \Phi_i = \frac{[X]_{rc} \otimes (\Phi_i - \sigma_i \mathbf{1}_{mn})}{\sqrt{[X^2]_{rc} \otimes \mathbf{1}_{mn} - \frac{([X]_{rc} \otimes \mathbf{1}_{mn})^2}{mn}}} \tag{EQ 22}$$

$$\text{with} \qquad \sigma_i = \frac{\mathbf{1}_{mn} \otimes \Phi_i}{mn} \tag{EQ 23}$$

Equation 22 gives the activity of the hidden unit i when the sub-image is located at position *(r,c)* in the image $\Im$.

One can observe from Equation 22 (compare the nominator) that cross-correlating a centred image with a filter is equal to the cross-correlation of the non-centred image with the centred filter, i.e. we have the following property that holds in general:

$$\overline{[X]}_{rc} \otimes \Phi_i = [X]_{rc} \otimes \overline{\Phi}_i \tag{EQ 24}$$

This is an interesting property, as we can now calculate the normalization of the weights off-line, instead of normalizing the sub-images during the detection process.

In order to get the activity of the hidden unit i when considering all possible sub-images, we get the activity matrix N_i which corresponds to a normalized version of H_i in Equation 13.

$$N_i = g\left(\frac{\Im \otimes (\Phi_i - \sigma_i \mathbf{1}_{mn})}{\sqrt{\Im^2 \otimes \mathbf{1}_{mn} - \frac{(\Im \otimes \mathbf{1}_{mn})^2}{mn}}} + B_H{}^i \right) \tag{EQ 25}$$

The final output of the neural network is a simple combination of the different activity matrices (q represents the number of hidden units):

$$O = \sum_{i=1}^{q} \underline{w}_O(i) N_i + B_O \tag{EQ 26}$$

Note that the output here is not a scalar but a matrix of dimension [S-m+1, T-n+1], where S and T are respectively the rows and columns of the input image, m and n the dimension of the sub-image (sliding window). Note further that since the central part of a face is almost square, we will use square sliding window sizes (m = n) when doing face detections later on.
Figure 13 compares the classic approach with the fast one where the whole input image gets treated at once. In the latter one, part of the preprocessing stage takes place off-line: centring of the input can be integrated right into the weights as has been shown

above. The output of the face detector neural network is a matrix of values situated between zero and one, representing estimated face and non-face locations.

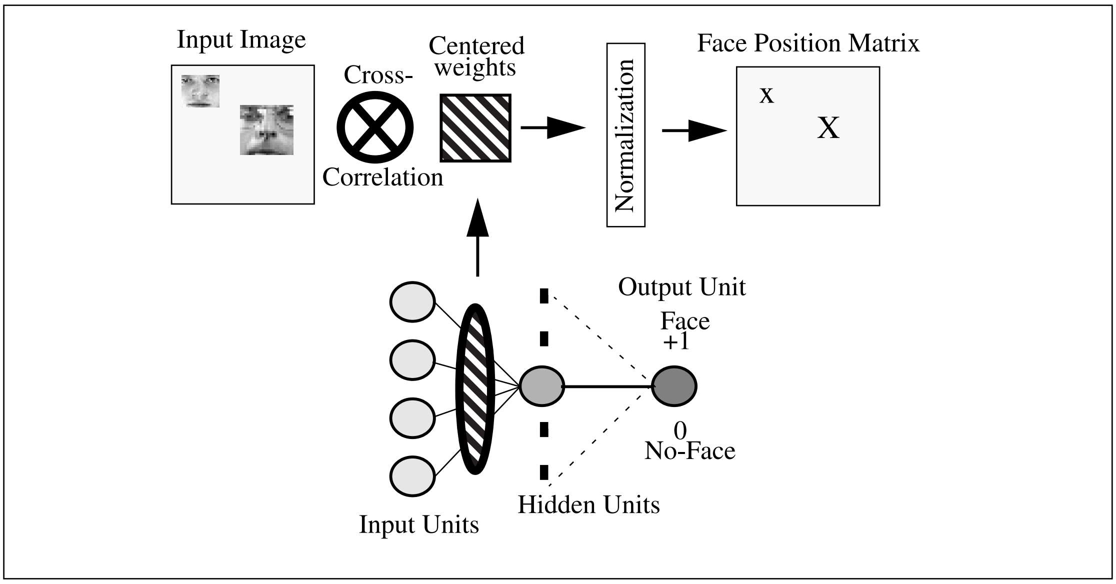


FIGURE 13. Cross-Correlation Operation

Given below is a graphical representation of Equation 25. F^* is the conjugated complex of the Fourier Transform.

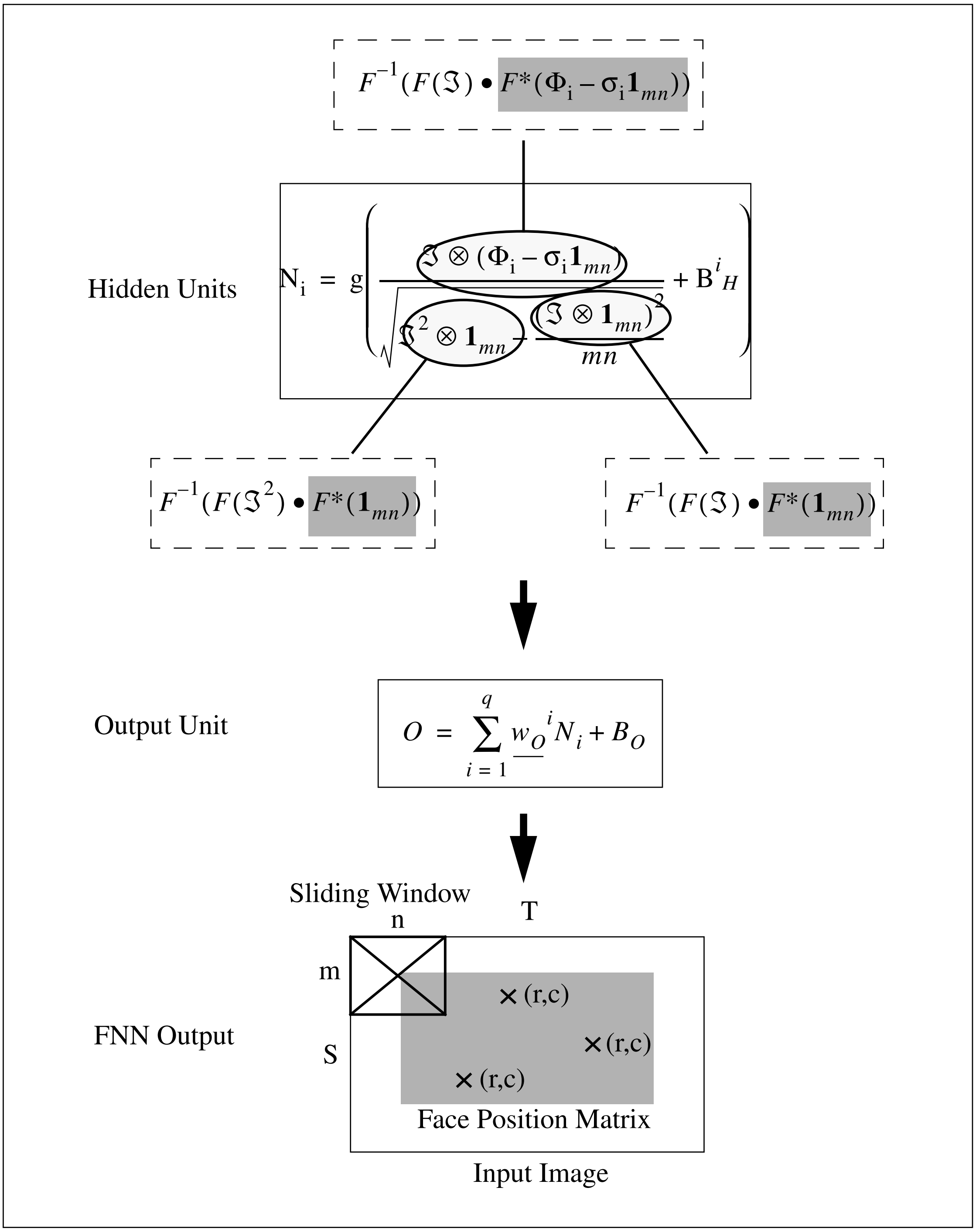


FIGURE 14. Graphical Representation of the Equations

The upper part of Figure 14 shows the central equation that must be computed for all hidden units. It contains three cross-correlations that may also be expressed using Fourier transforms. The hatched parts in the dashed formulas can be precalculated (off-line).

The activities of the hidden units are then fed into the output unit which finally delivers the Face Position Matrix (faces are detected, if an output network output value exceeds the decision level. In the above case, the network found three faces, their positions are marked with X's).

5.3 Standard Deviation Normalized Input

The normalization procedure described in the previous section, based on the work done in [1] caused problems during the training of the neural network - slowly decreasing error rates, especially errors reported by the hidden units. Therefore epoch rates of 1000 and more were needed for a complete training cycle.

We found that the problem was caused by the division of large values in the denominator aggregated by summing over all values in the sliding window:

Equation 17 can also be written as

$$\overline{[X]}_{rc} = \frac{([X]_{rc} - \bar{x}_{rc}\mathbf{1}_{mn})}{\sqrt{\sum_{i=1}^{mn}(x_{rc}(i) - \bar{x}_{rc})^2}} \quad \text{(EQ 27)}$$

where n is the sliding window size, $\overline{[X]}_{rc}$ the sub-image located at rc in the test image and $\bar{x}_{rc}$ its mean value.

The weight of the denominator was therefore be reduced according to the number of components contained in the sliding window. Dividing by the standard deviation σ_i is virtually the same (one component less in the latter case):

$$\overline{[X]}_{rc} = \frac{([X]_{rc} - \bar{x}_{rc}\mathbf{1}_{mn})}{\sigma_i} = \frac{([X]_{rc} - \bar{x}_{rc}\mathbf{1}_{mn})}{\sqrt{\frac{1}{mn-1}\sum_{i=1}^{mn}(x_{rc}(i) - \bar{x}_{rc})^2}} \quad \text{(EQ 28)}$$

There is still a problem though, namely with the behaviour of the denominator in Equation 28. In homogenous areas of the input image, the average of a sliding window may cancel out with each of its components, leading to very small denominator and in turn to extreme peaks at the momentary position of the sliding window, yielding more false detections, as the neural network sees its input getting overflowed by huge values.

The solution is to divide by a small additional constant α and Equation 28 becomes:

$$\overline{[X]}_{rc} = \frac{([X]_{rc} - \bar{x}_{rc}\mathbf{1}_{mn})}{\sigma_i + \alpha} \quad \text{(EQ 29)}$$

5.4 Centered[1]Input Version

As the normalized version of the face detector delivered too many false detections (see also obtained test results), the zero-mean centred only approach had been investigated.

1. Centred image stands here for an image having zero-mean values.

Here we divide by Λ, the half of the maximum pixel value that can be found in the input image.

$$\overline{[X]}_{rc} = \frac{([X]_{rc} - \bar{x}_{rc}\mathbf{1}_{mn})}{\Lambda} \tag{EQ 30}$$

Our test sets contained exclusively grayscale image encoded with 8 bits. In other words, 255 levels were present, therefore Λ was set to 128. In this way the neural network input values are between -1 and +1. Not dividing by this constant resulted in poor learning and detection rates.

5.5 Execution Speed-Ups

5.5.1 Theoretic Speed-Ups

The 2D Fourier Transform of a N x N test image $\Im$ requires $O(N^2 Log_2{}^2 N)$ computation steps[1]. The 2D FT of the filters $(\Phi_i)_{i \in 1.q}$ can be computed off-line since these are constant parameters of the network independent of the test image. A 2D FT of the test image has to be computed, therefore the total number of FT to compute is *q+1* (one forward and *q* backward transforms). In addition, we have to multiply the transforms of the weights and the input image in the frequency domain adding a further $O(mqN^2)$ computation steps.

This yields a total of $O((q+1)N^2 Log_2{}^2 N + qN^2)$ computation steps for the fast neural network.

For a classic neural network, $O((N-n+1)n^2 q)$ computation steps are required, when one considers the activity of q neurons in the hidden layer and a square test image of size NxN and a sliding window of size nxn. Therefore the theoretical speed-up factor is

$$O\left(\frac{q(N-n+1)^2 n^2}{(q+1)N^2 Log_2{}^2 N + qN^2}\right) \tag{EQ 31}$$

Equation 31 describes the speed-up gained with the input centred version of the face detector. The input normalized version features three more FFT transforms, which leads to a complexity of

1. http://theory.lcs.mit.edu/~fftw/

$$O\left(\frac{q(N-n+1)^2 n^2}{(q+4)N^2 Log_2^{\ 2} N + qN^2}\right) \quad \text{(EQ 32)}$$

Figure 15 displays the expected speed-up with the centred input centring only, respectively with normalized input.

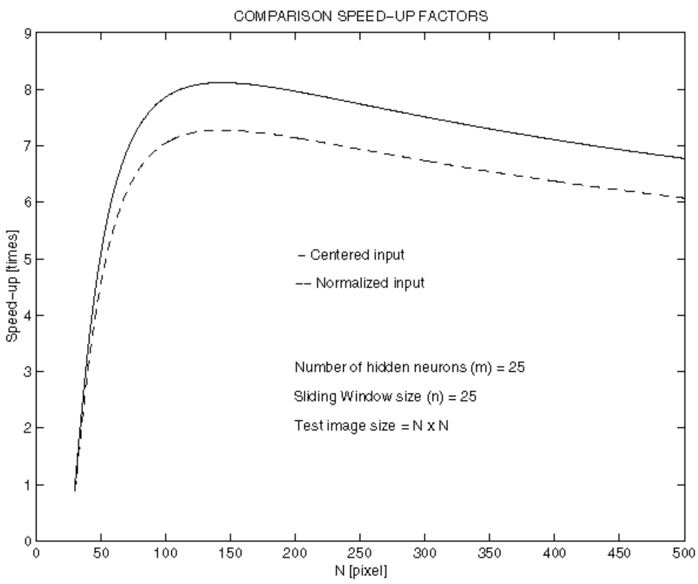


FIGURE 15. Theoretic Speed-Up curves FNN versus Classic NN

The multiscale centred version on the other hand needs one less FFT transformation on upper scales, (two less in the case of the multiscale normalized version) so there would be a speed-up of order

$$O\left(\frac{(N-n+1)^2 n^2}{N^2 Log_2^{\ 2} N + N^2}\right) \quad \text{(EQ 33)}$$

with sub-sampled scales.

Figure 16 shows the aligned theoretical values (using a scaling factor) in comparison with the measured ones. Prediction is quite satisfying and underlines the correct modelling of the curves by above formulas.

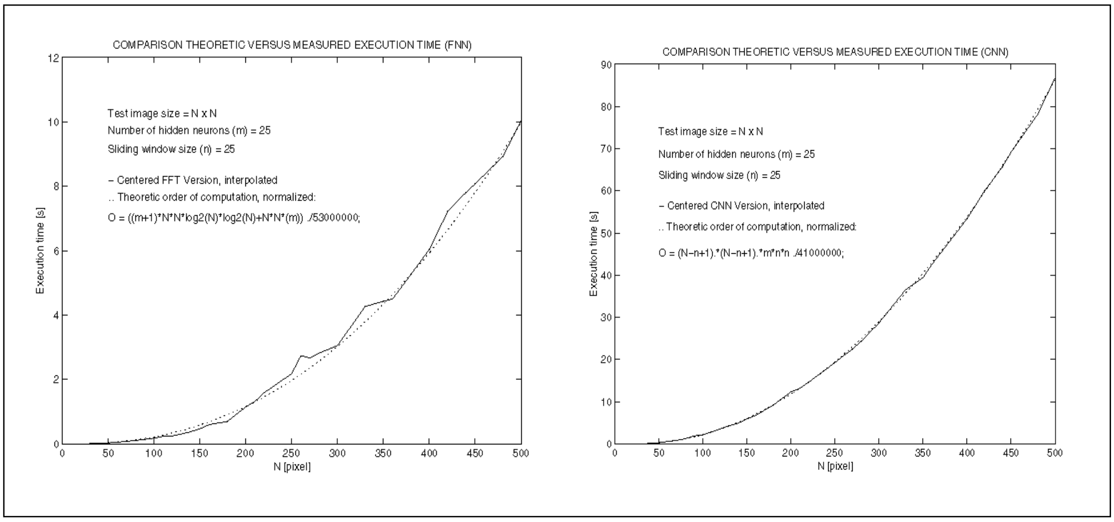


FIGURE 16. Comparison CNN / FNN measured and theoretical values

Figure 17 (see also Figure 11): The difference in execution time introduced by the fast squash function can be clearly seen with large input image sizes as we must call it proportionally to the seize of the input image.

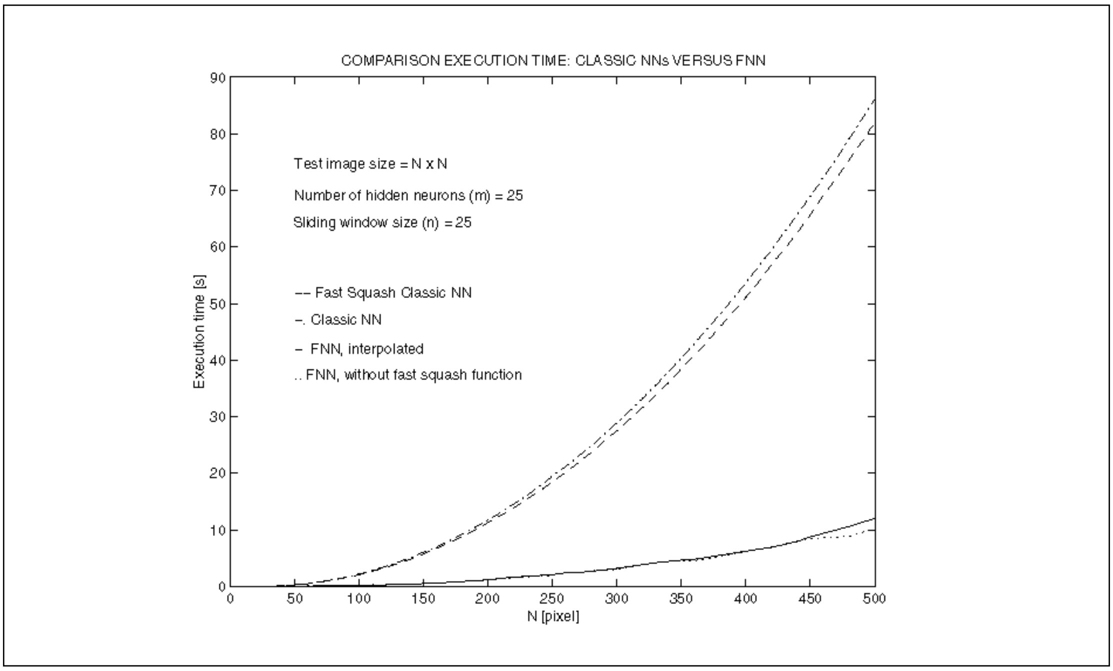


FIGURE 17. Comparison with/without the Fast Squash function

5.5.2 Measured Speed-Up

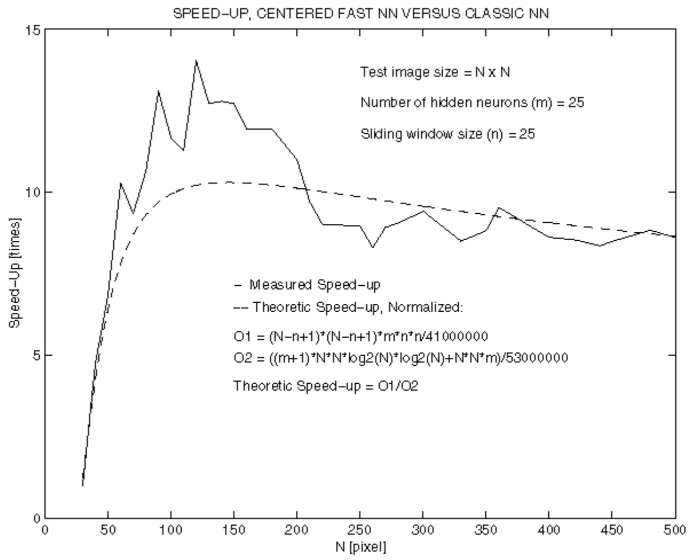


FIGURE 18. Measured Speed-Ups curve Centered Input FNN versus Classic NN

Interesting is the comparison between the measured speed-up and the predicted one in Figure 18, using the aligned curves defined in Figure 16. The theoretic curve models in a quite precise fashion the general progression of the measured curve. But with smaller input image sizes, the measured speed-up exceeds the predicted one significantly. In addition the curve is rather jagged, reflecting the internal behaviour of the FFT implementation. The FFT library used during this project [18] is heavily optimized and we experienced execution times that go well below the theoretically predicted order of $O(N^2 Log_2{}^2 N)$.

5.6 Neural Network Visualization

Each hidden unit in a neural network performs a cross-correlation operation between its weights and its input. Here, the input consisted of the image shown on the left hand side of Figure 19. Its intensity plot is given in the middle, the viewing angle being from top of the input image.

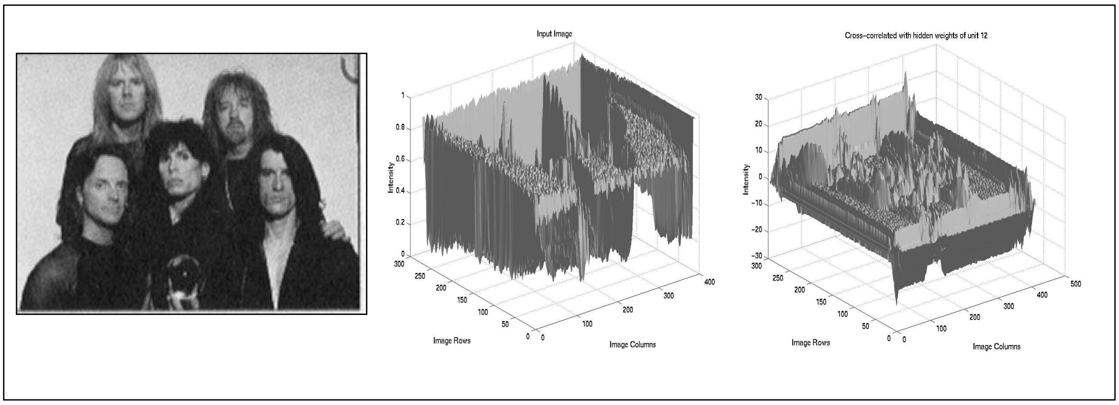


FIGURE 19. Cross-Correlation Operation

On the right hand side is shown the same image cross-correlated with the weights of a hidden neuron. Clearly all uniform areas result with low intensity. On the other hand, the group and especially their faces get accentuated.

Figure 20 depicts the effects of the cross-correlation operations on a Neural Network scale. Note that images having passed the squash function contain values between zero and one.

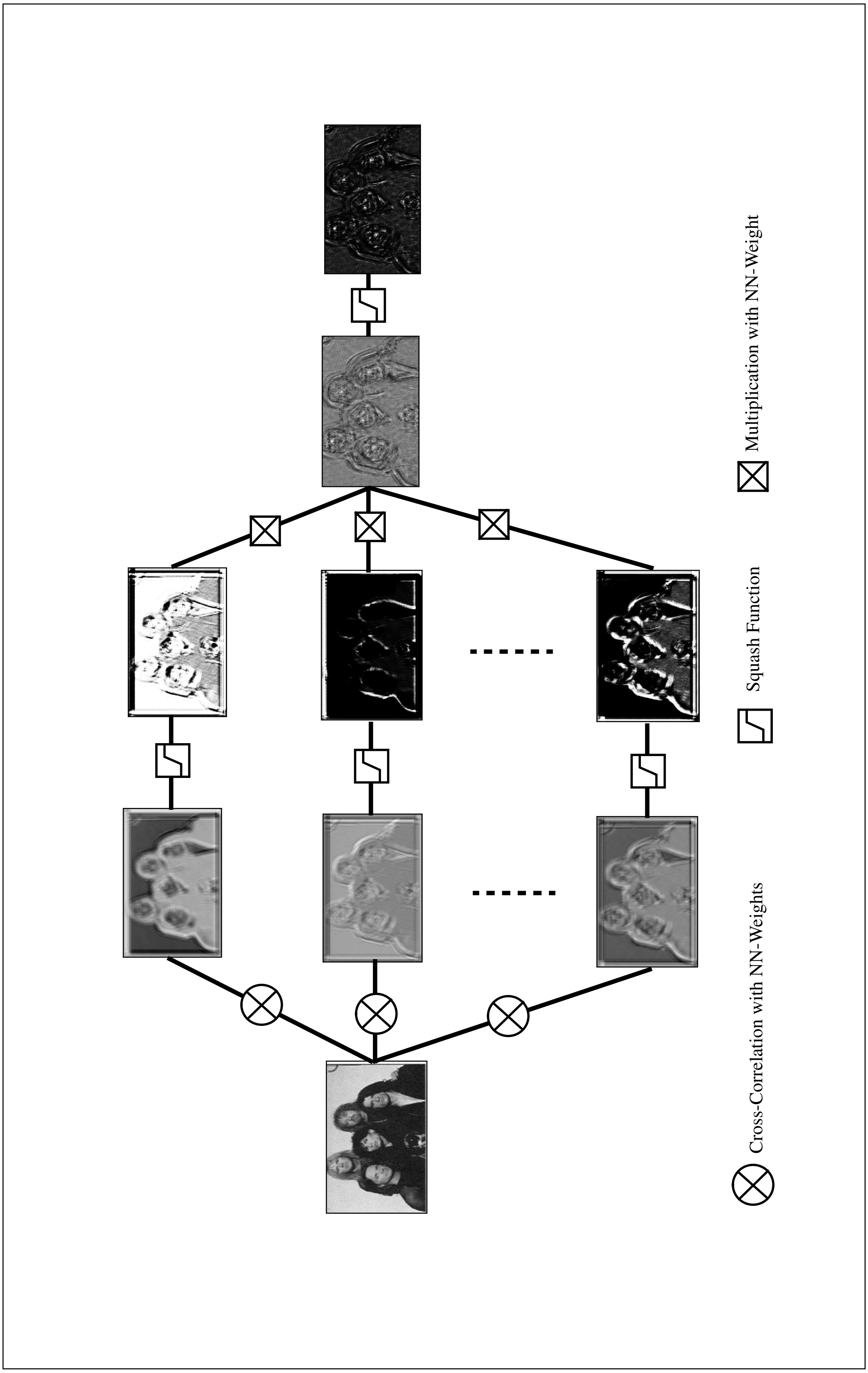


FIGURE 20. NN Cross-Correlation Visualization

5.7 Summary

The fast neural network computes exactly what the classical approach which employs a sliding window. But it runs in a much faster way, thanks to FFT.

The fast neural network approach can be summarized as follows:

Cross-correlating the input image with the hidden weights is the same as passing a sliding window through the input image and afterwards multiply the input with the weight matrices. This cross-correlations can be expressed in terms of a multiplication in the frequency domain. The mapping into the frequency domain and backwards afterwards is accomplished by using the Fast Fourier Transform. Parts of the cross-correlation can be calculated off-line, before the actual operation. This leads to a significant speed-up in calculation that have to be performed on-line.

Note that with the fast neural network, the input image is treated as a whole and there is no sliding window employed to feed a local sub-window into the neural network.

6 Still Image Face Detector

The fast neural network described above can be applied for the purpose of still image face detection. Speed of execution is not an important issue here and the still image face detector serves as a test bed for the Moving Picture Face Detector presented in the next chapter.

The architecture of the Still Image Face Detector will be explained in the implementation section followed by a discussion of postprocessing methods as well as still image scanning results.

6.1 Theory

We found that blurring the input image and therefore reducing its high frequency components can lead to dramatically reduces false detections, especially with high contrast images, the theoretic layout is given in the next section

Scanning an input image at different resolutions allows to detect objects (faces), that are close to the camera, or further away and couldn't therefore be detected otherwise. Scanning at close distances has another advantage, because there occur 3D-Multiple detections, that can help to distinguish faces from non-faces in the post-processing step. This subject will be addressed further below.

6.1.1 Gaussian Blurred Input

The idea is to filter the input image with a low-pass filter, in order to reduce high frequency components that have shown to be prone to excite the neural network in an uncontrolled way, see also Figure 21. This can be achieved by cross-correlating the input image with a gaussian mask *G:*

$$\Im_{Blurred} = \Im \otimes G = F^{-1}(F(\Im) \bullet F^*(G)) \qquad \text{(EQ 34)}$$

$$\text{where G may be chosen as: } G = \begin{bmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{bmatrix} \qquad \text{(EQ 35)}$$

Using a gaussian mask *G* of size *3 x 3* results in a reasonable blurring of the input image.

The blurring operation can be integrated right into the FFT framework, the cross-correlation part of Equation 13 becomes then

$$\Im_{Blurred} \otimes \Phi_i = (\Im \otimes G) \otimes \Phi_i = \Im \otimes (G \otimes \Phi_i) = \Im \otimes P(G, \Phi_i) \qquad \text{(EQ 36)}$$

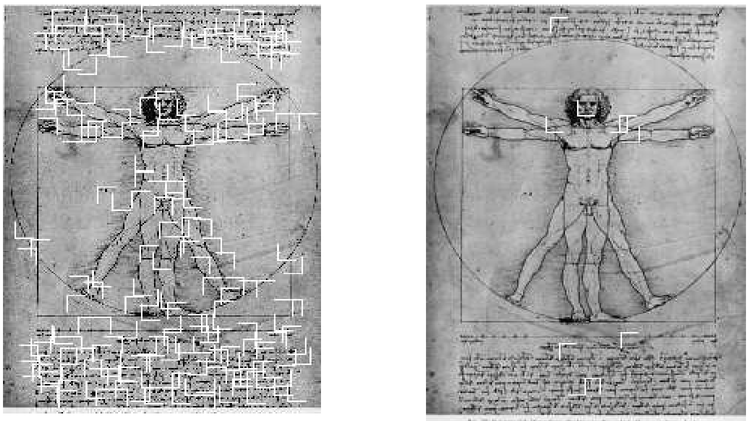
$$\text{where } P(G, \Phi_i) = G \otimes \Phi_i = F^{-1}(F(G) \bullet F^*(\Phi_i)) \qquad \text{(EQ 37)}$$

Finally the main cross-correlation operation (see Equation 13) can be reformulated as

$$\Im_{Blurred} \otimes \Phi_i = F^{-1}(F(\Im) \bullet F^*(P(G, \Phi_i))) \tag{EQ 38}$$

The term $P(G, \Phi_i)$ can be precalculated in the off-line section, so there won't be additional computational expenses during the actual detection process in the on-line section.

Blurring the input image using a gaussian filter results in a substantial reduction of the false detection rates, especially in scenes containing lots of textures or in images with an important high frequency spectrum. The following images demonstrate the latter behaviour. The image on the right hand side was blurred using a 3 x 3 gaussian mask:



Without blurring With blurring

FIGURE 21. Comparisons with / without blurring

The gaussian filtering was integrated into the face detector and can be switched on and off. In the case of the multi-scale face detector, this also reduces disturbances caused by anti-aliasing when doing sub-sampling of the input image for the multi-scale face detection purpose as described in the next section.

6.1.2 Multi-Scale Search

Scanning an image at several resolutions can be done by sub-sampling the whole test image at several scales before feeding the various sub-images obtained later on to the neural network.
During the computation of the cross-correlation we work in the frequency spectrum. the sub-sampling can be entirely performed there using the following property of the Fourier transform:

$$f(ax, by) \Leftrightarrow \frac{1}{|ab|} F\left(\frac{u}{a}, \frac{v}{b}\right) \tag{EQ 39}$$

One could chose for example $a = b = 2$ in order to get an image reduced to half of the original size. The main cross-correlation operation in Equation 13 can therefore be modified to accommodate multiple scales (resolutions) in the face detector:

$$\Im\big|_{a,b} \otimes \Phi_i = F^{-1}\left(\frac{1}{|ab|}F\left(\Im\left(\frac{u}{a}, \frac{v}{b}\right)\right) \bullet F^*(\Phi_i)\right) \qquad \text{(EQ 40)}$$

Note that the size of the weight matrices (right hand side components in above equation) must be equal to the size of the sub-sampled input images in the frequency domain (left hand side components).

6.1.3 Method Combination

We can combine now all described methods into one, featuring input centring, input blurring and multi-scale search in a single expression. Applying equations (13) and (30) we get:

$$C_i^1\Big|_{a,b} = g\left(\frac{\Im_{Blurred}\big|_{a,b} \otimes (\Phi_i - \sigma_i \mathbf{1}_{mn})}{\Lambda} + B^i{}_O\right) \qquad \text{(EQ 41)}$$

Or alternatively doing input normalization and using equations (22), (28) and (29) with the standard deviation method:

$$C^2{}_i\Big|_{a,b} = g\left(\frac{\Im_{Blurred}\big|_{a,b} \otimes (\Phi_i - \sigma_i \mathbf{1}_{mn})}{\sqrt{\frac{1}{mn-1}\left(\Im_{Blurred}\big|_{a,b}{}^2 \otimes \mathbf{1}_{mn} - \frac{\left(\Im_{Blurred}\big|_{a,b} \otimes \mathbf{1}_{mn}\right)^2}{mn}\right) + \alpha}} + B^i{}_O\right) \qquad \text{(EQ 42)}$$

In both cases, the nominator can be computed as follows:

$$\Im_{Blurred}\big|_{a,b} \otimes (\Phi_i - \sigma_i \mathbf{1}_{mn}) = F^{-1}\left(\frac{1}{|ab|}F\left(\Im\left(\frac{u}{a}, \frac{v}{b}\right)\right) \bullet F^*(P(G, \Phi_i, \sigma_i))\right) \qquad \text{(EQ 43)}$$

$C_i^1\big|_{a,b}$ and $C^2{}_i\big|_{a,b}$ stand for the activity of the hidden unit i in combination with the above mentioned methods.

Note: One can get the two convolutions in the denominator of Equation 42 in a similar way as done in Equation 43.

The sub-sampling section forms an integral part of the multi-scale face detector. Each input image is being processed in neural networks at various resolutions. The output is then fed into an arbitrator structure which decides at which resolution a face has been detected.

Figure 22 shows a graphical representation of the fast multi-scale face detector. The pictures surrounded by a black frame represent the sub-sampled images in the frequency domain.

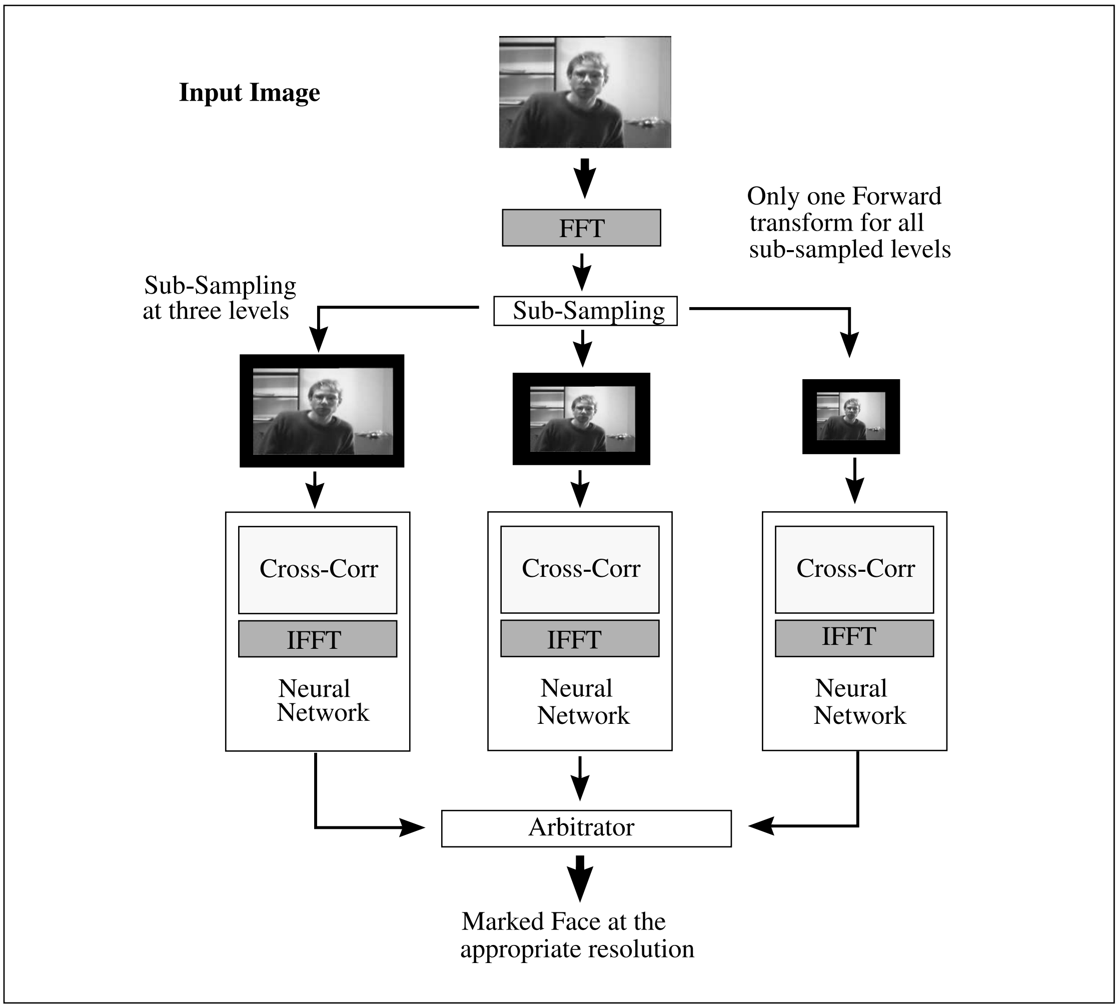


FIGURE 22. Fast Multi-Scale Face Detector

On the left hand side of Figure 23 is shown the unprocessed output of the multi-scale-face detector whereas on the right hand side the output was postprocessed using methods described in the next section.

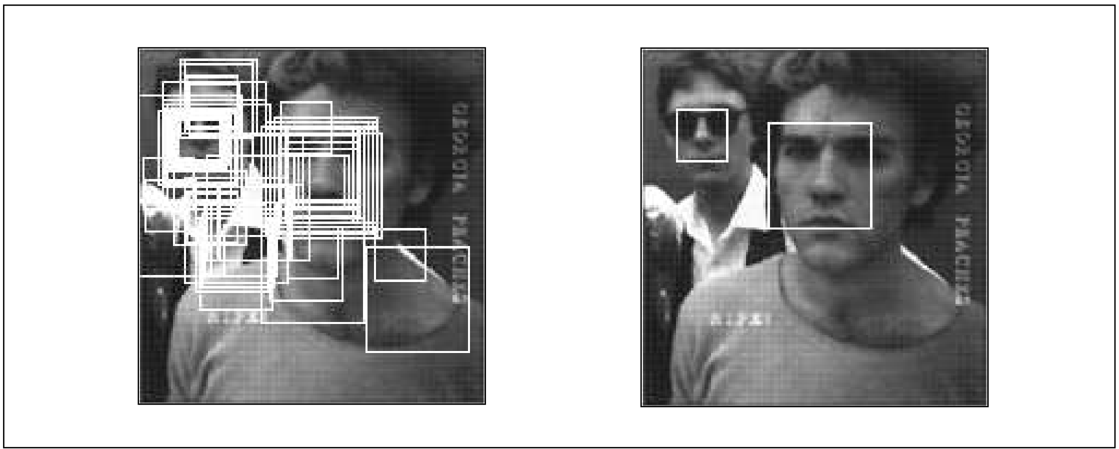


FIGURE 23. Multi Resolution Example

Detection was done at three resolutions.

6.2 Postprocessing - Single Network Arbitration

In contrast to the preprocessing stages that are closely interweaved with the neural network as described above, the postprocessing takes place in a separate module, using the outputs delivered by the neural network(s). Postprocessing is an important step in order to reduce false detections, while confirming correct detections.

6.2.1 Face Properties

Faces demonstrate an interesting property: a face can easily be recognized as such when being viewed through the sliding window not only in the centre position but also at positions located near to it. Further on, the face images used to train the neural networks were taken at different scales of resolution, which leads to detections of a given face at multiple resolutions. The face detector generates in this way clusters of 2D and 3D multiple detections when it comes across a face.
Using this property we have in addition to the detection rate information that says something about the probability of the existence of a face located at the current sliding window position. Depending on whether we work at a single or with a multitude of resolutions, several additional decision strategies may be applied.

6.2.2 2D Multiple Detections

If we work at one resolution, we may use the number of 2D multiple detections as an indication for faceness in addition to the detection rate given at a certain position in the input image. On the upper left hand side in Figure 24 we see a typical output of a neural network. The face has been detected several times at close distance. In a next step, the square with the highest detection value is being chosen and its centre location taken (see image on the upper right hand side) to define the centre of the *multiple window* (hatched). All further detections which have their centre in multiple window are then removed and one retains only the detection with the highest probability value found (see image on the lower right hand side).

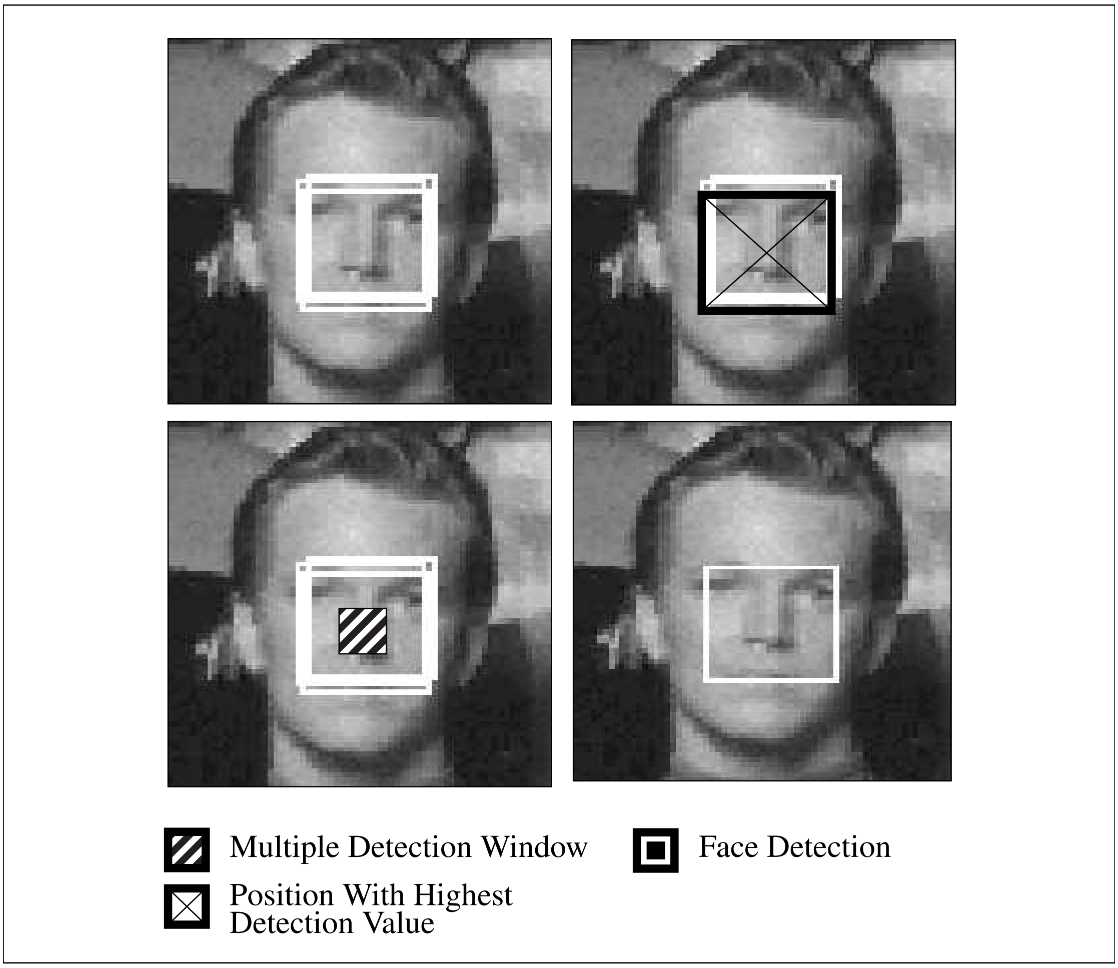


FIGURE 24. 2 D Multiple Detection Examples

The 2D-Multiple-Detection-Marking procedure works as follows:

1. All outputs of the neural network that are equal or larger than a given threshold are stored in a list, together with the position in the input image, as well as the associated detection value.
2. The obtained list is being sorted in descending order of the obtained values
3. Next we parse the list. The candidate located at the first position is the one with the highest detection value. Then, marking as multiple all double detections further down the list (those having a centre in the multiple detection window) prevents from starting the same procedure for these later on. Every time we encounter such a multiple, we increment the multiple counter of the first candidate by one.
4. The next non-multiple marked candidate in the list, starts again the multiple marking process described in point three. This goes on till there are no more candidates left in the list.

This marking procedure is graphically represented in Figure 25.

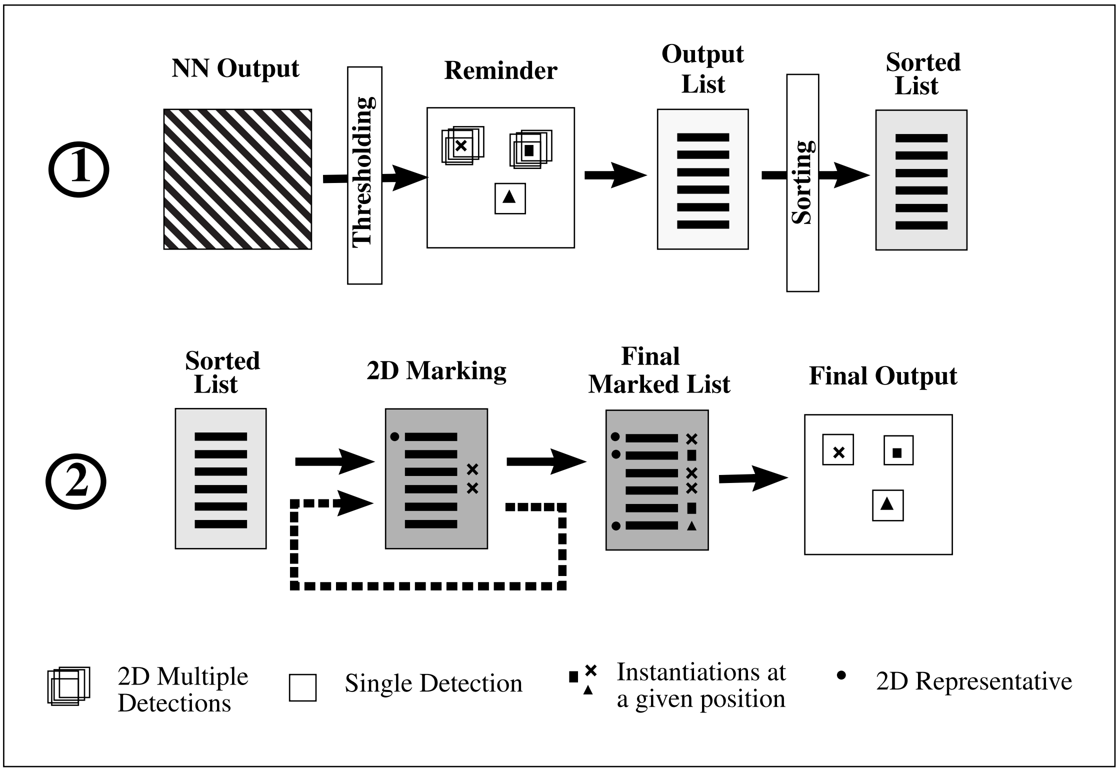


FIGURE 25. 2D Multiple Marking Procedure

6.2.3 3D Multiple Detections

3D Multiple detections are clusters of 2D detections at the same location at different resolutions in the test image. The occurrence of such a cluster indicates a high probability of a face at the given location.

Another value that is a measure of confidence of a certain detection is the *2D-Aggregate value*. That is the number of 2D-Multiple detections that occurred in a given *3D-Cluster* at all resolutions involved.

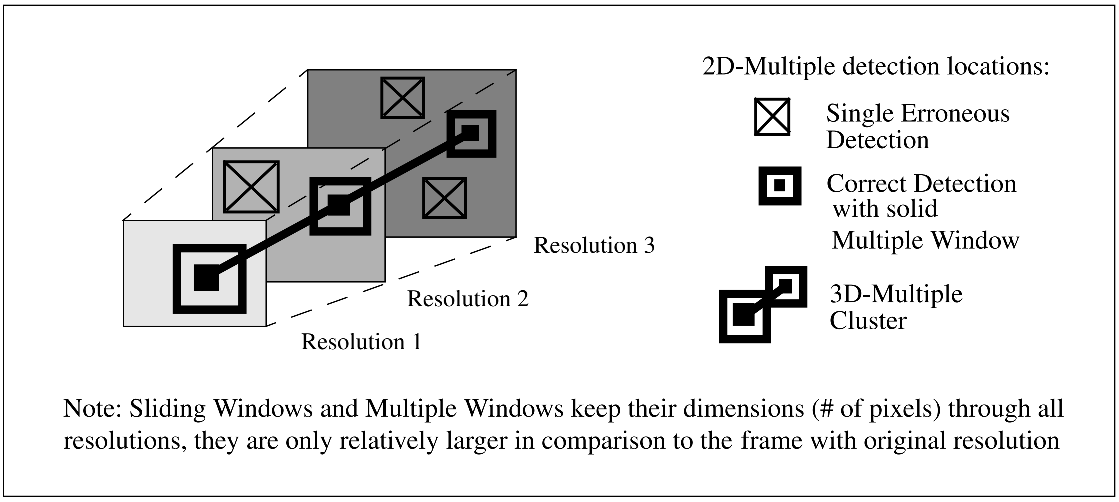


FIGURE 26. 3D-Multiple-Detection Set-Up

Figure 26 shows the principle of the 3D-Clusters. All single detections have been removed beforehand and there remain only 2D-Multiple detection for further evaluation. All positions containing no 3D-Clusters may eventually be discarded by the system.

The 3D-Multiple detection marking and 2D-Aggregate summations take place after the 2D marking process described above. Merging detections at all scales into one list allows for a simpler handling of the 3D marking:

After 2D marking, parsing of the output list starts again from the beginning. This time, all non-multiple 2D detections found in different resolution of the input image but located at the same position (with centre located in the multiple window) are attributed to the resolution, where the highest detection value has been registered. In addition every 2D-Multiple encountered increments the 2D-Aggregate counter of the main representative of the 3D-cluster. Repeating these steps through the whole list finally yields all 3D-Multiple detections featuring just one 3D detection per 3D cluster.

6.2.4 Decision Criteria

In addition to the detection rate we have now the above described three additional indicators at disposition: 2D-Multiples, 3D-Multiples and 2D-Aggregate values. The question is now how to combine all those values in order to get robust results. That is a high face detection rate and a low false detection rate.

Following strategies may be applied:

1. Strategy one (single resolution)

- Reject all detection values below a certain detection threshold

2. Strategy two (single resolution)

- Reject all detection values below a certain detection threshold
- Reject all 2D-Multiple detections below a certain 2D threshold

3. Strategy three (single resolution)

- Reject all detection values that fall below a certain detection threshold
- Reject all 2D-Multiple detections that fall below a certain 2D threshold
- Take only a given percentage of all detections and reject those with the lowest decision values

4. Strategy four (multi resolution)

- Reject all positions with detection values below a certain detection threshold
- Reject all positions with 2D-Multiple detections that fall below a certain 2D threshold
- Reject all positions with 3D-Multiple detections that fall below a certain 3D threshold

5. Strategy five (multi resolution)

- Reject all positions with detection values below a certain detection threshold
- Reject all positions with 2D-Multiple detections that fall below a certain 2D threshold
- Reject all positions with 3D-Multiple detections that fall below a certain 3D threshold
- Reject all positions with 3D-Clusters containing less 2D-Aggregate values than demanded by the 2D-Aggregate threshold

The arbitrator settings given in Table 2 showed to deliver acceptable detection results on various test images. The first one is of interest when doing scans at one resolution, strategy 5 is the most restrictive one and of interest when doing multi-scale face detection.

Features	Strategy 1	Strategy 5
2D-Multiple Threshold	2	2
3D-Multiple Threshold	-	2
2D-Aggregate Threshold	-	8 or 4 * # resolutions
Detection Threshold	0.85	0.85
Scale Ratio	1.2	1.2

TABLE 2. Strategy Threshold Settings

Figure 27 shows an actual example of how the postprocessing strategy five works. An image has been sampled at three resolutions. The number and kind of detections are given in the lower part of the figure.

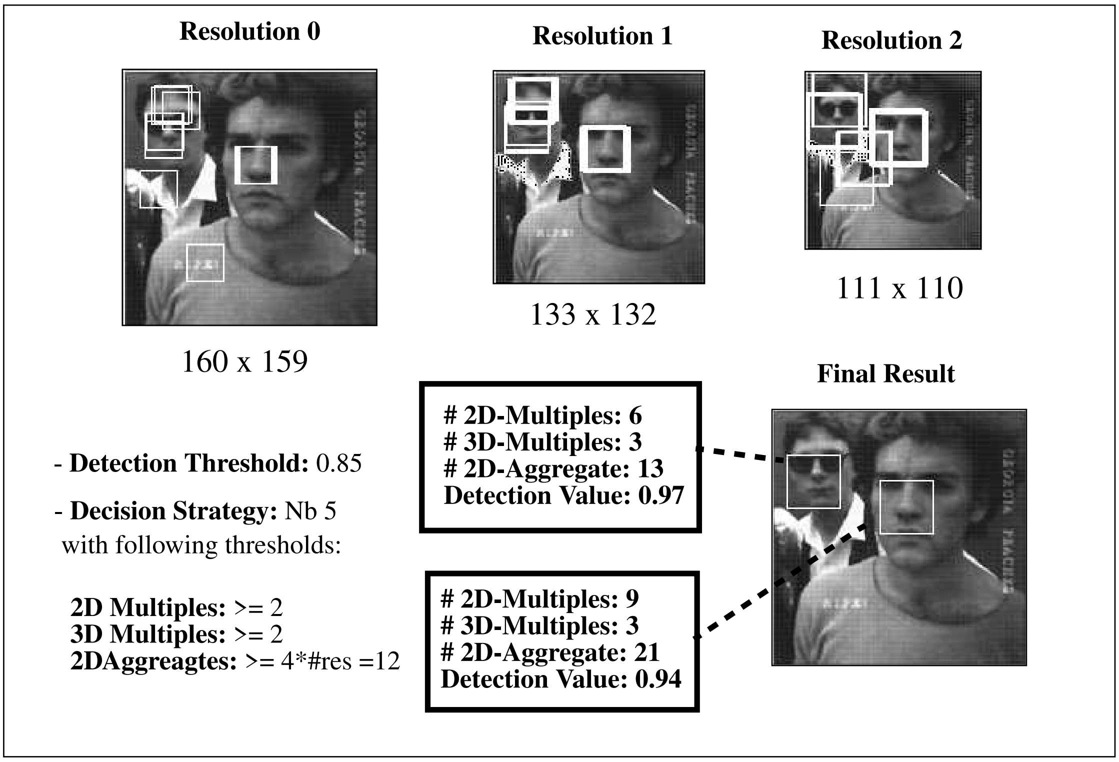


FIGURE 27. 3D Multiple Detection Examples

The accumulated 2D detections values (2D-Aggregates) are important indicators: In Figure 27 we can see an erroneous detection in the upper left hand corner that occurs in all three resolutions. This would have produced a false detection without the 2D-Aggregate value threshold.

6.3 Postprocessing - Multiple Network Arbitration

Superior detection results can be obtained by combining several neural networks that have been trained differently (see Section 3 on page 7). This results in lower false detection rates without an important reduction of correct face detections.

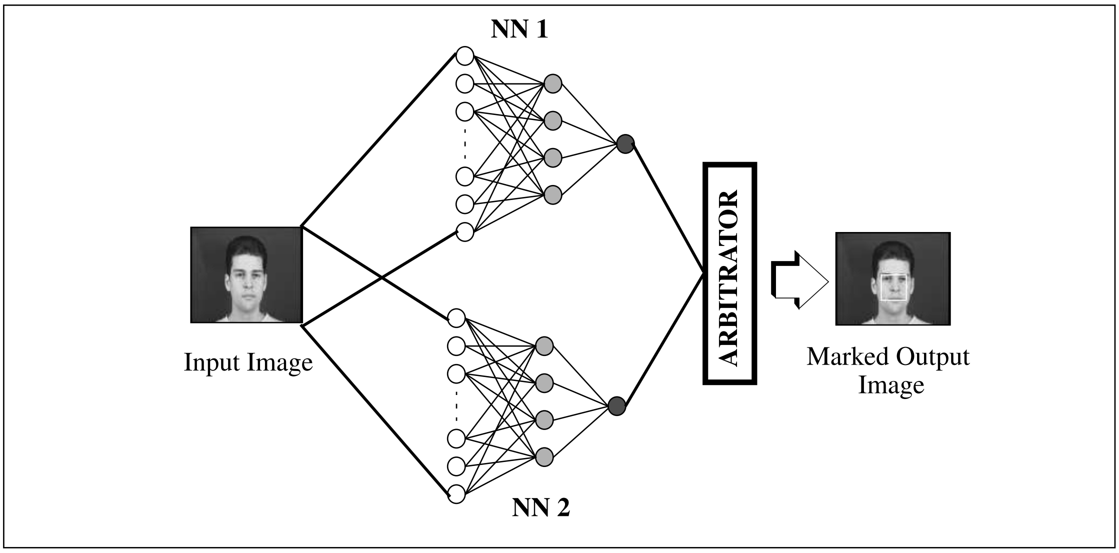


FIGURE 28. Multiple Networks in Parallel applied to the same Input Image

Figure 28 is shows the set-up of a multiple neural network face detector. On the contrary to what we did in the previous chapter, the input image is here treated at the same resolution. There are various possibilities of how to arbitrate between multiple neural networks (AND, OR, Arbitrator network, etc.). We use here adding over a multiple window, that means, all detection that have their centre in the same region (multiple window) are considered to belong to the same detection and therefore will lead to a positive detection at the output of the network arbitrator.

6.4 Implementation

The Fast Face Detector (FFD) consists of three basic modules, the Neural Network Trainer (NN-TRAIN), the off-line Precalculator (FFD-PREC) and the on-line Face Detector (FFD-ON). First a neural network is trained on a given database. All weights of this network are then written into a *Network file*. Reading this file, the on-line process can later on instantiate the same neural network, ready for face detection.

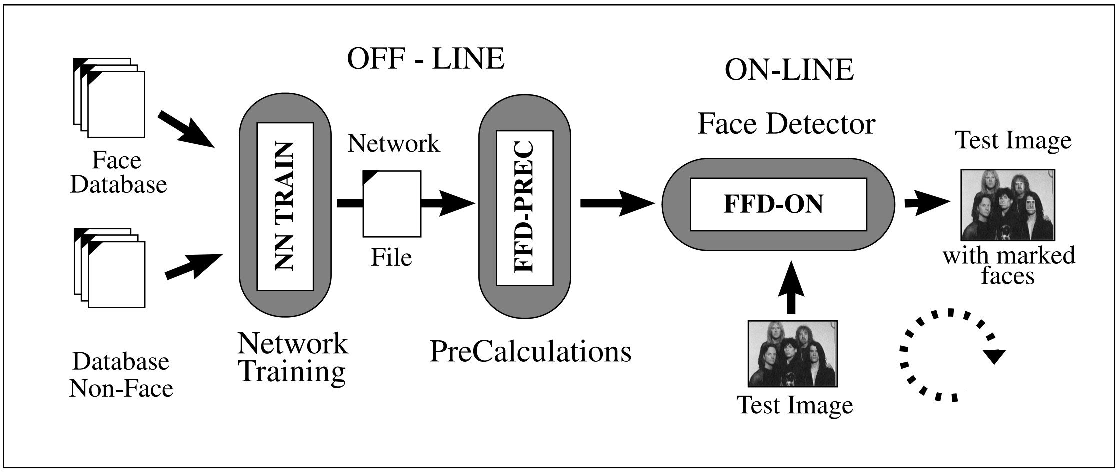


FIGURE 29. The Still Image Face Detector

The Precalculator in Figure 29 determines the hatched parts of the cross-correlations displayed in Figure 14. The actual Face Detector is contained in the on-line module. It computes the cross-correlations using Fast Fourier Transforms.

6.4.1 Off-line / Total Execution time

The time spent in the off-line section evolves in an exponential way. This goes also for the total time, which gives an indication of much time is needed to scan an image of a given size, see Figure 30.

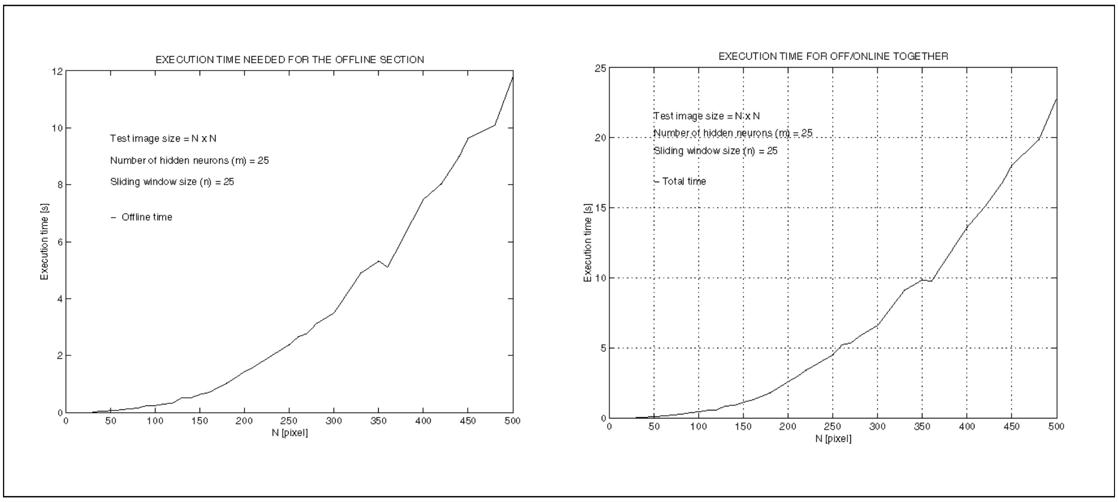


FIGURE 30. Off-line and Total execution time, Centered version

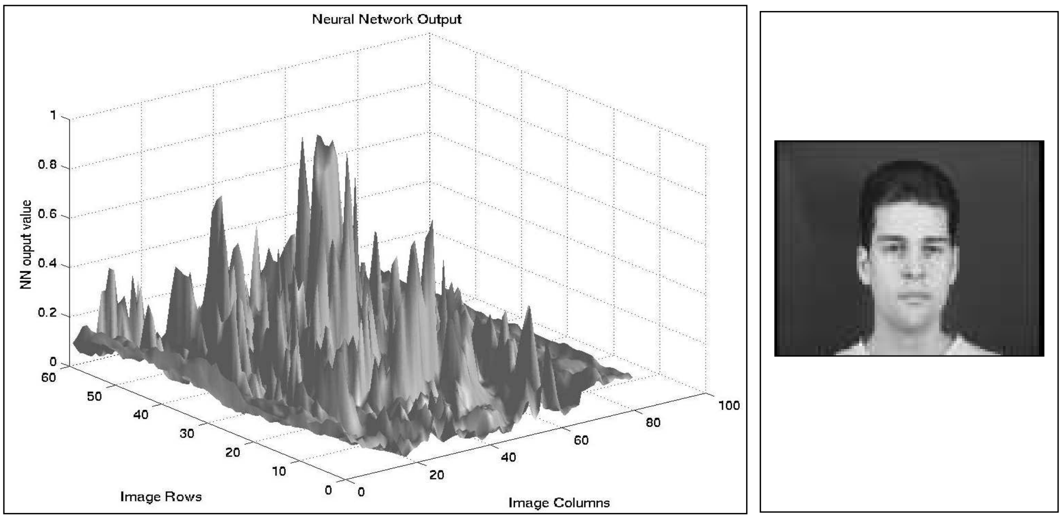
A speed-up of about three can be obtained in comparison to the classic neural network (see Figure 16), provided that we work with image sizes that allow us to exploit the minimum region of the FFT, see Figure 44.

6.5 NN Output Discussion

In the following sections we have a look at outputs produced by neural networks and discuss postprocessing arbitration and decision level settings.

6.5.1 NN Output Visualization

On the left hand side of Figure 31 is shown a 3D plot of the raw output of a neural network obtained with the image on the right hand side as input (without the decision threshold being applied). The surface is rather jagged with lots of needle-like peaks. The highest of these represent face location.



NN Output for "Face", 3D view **Image "Face"**

FIGURE 31. Neural Network Output Visualization

Between the peaks are wide open spaces featuring very low values, see Figure 32. This results in rather low average values as will be discussed in the next section. Clearly can be seen the white spot in the middle of the figure. It corresponds to the highest peak in Figure 31 and represents the face location.

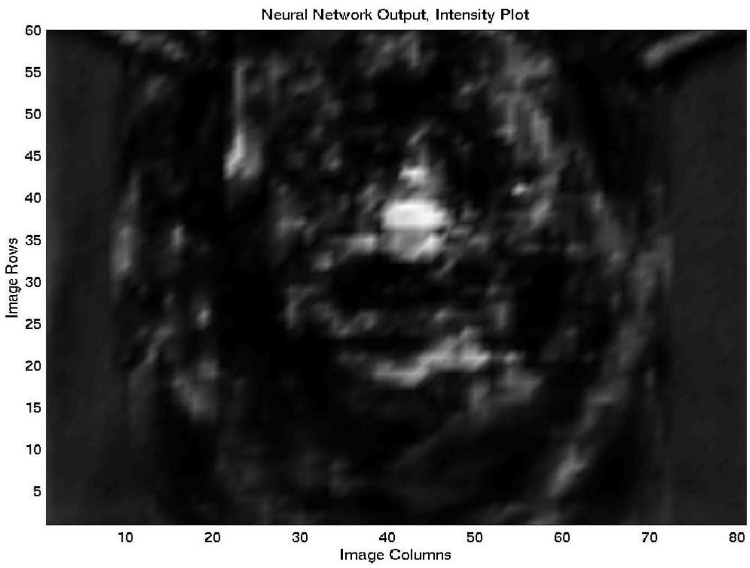


FIGURE 32. Probability Map for Image “Face”

Setting correctly the output threshold is of great importance as can bee seen in Figure 33. If you set it to high, you may loose detections and if it is set to low, secondary peaks appear and introduce false detections. So we should set it in between in order that we have enough 2D multiple detections allowing us to applicate the multiple arbitrator correctly.

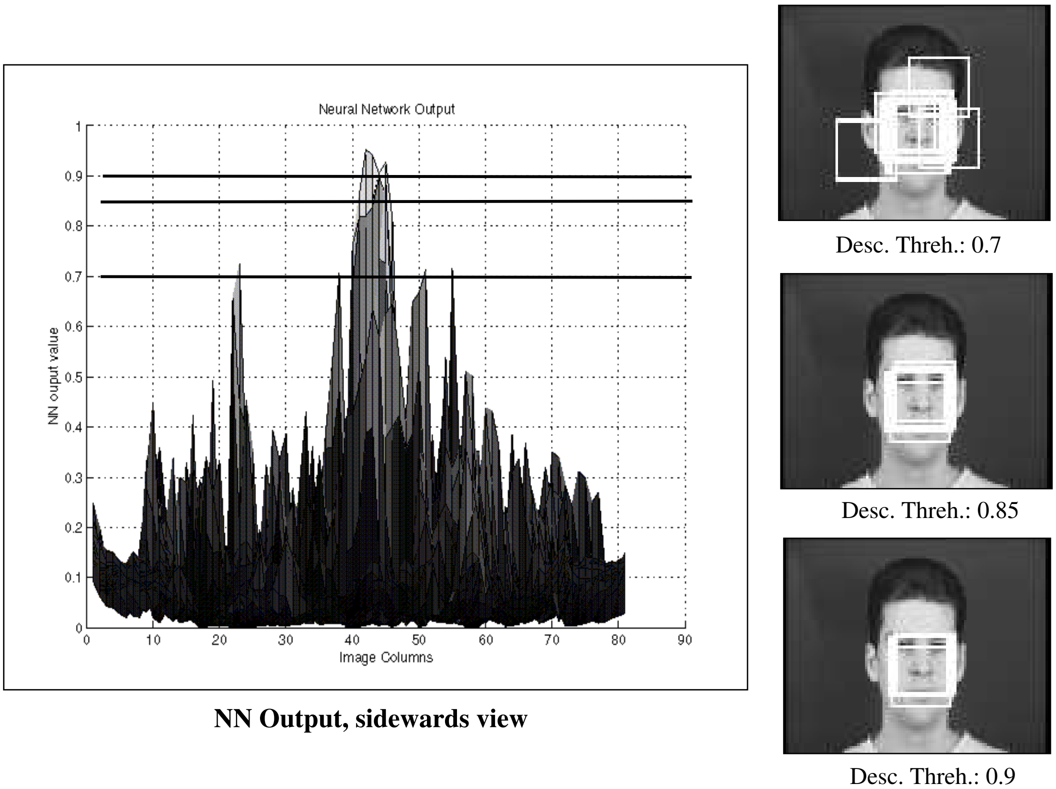


FIGURE 33. Decision Thresholds

In Figure 34 we see a typical result of the sill image face detector, doing a close scale search on two levels. Face number on was detected with highest probability on level zero (original input image size), the others in the sub-sampled image.

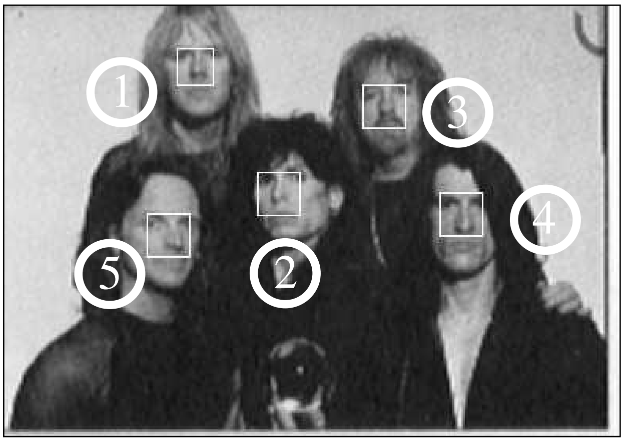


FIGURE 34. Face detection

Table 3 represents all values associated with the faces detected in Figure 34. There is a correlation between the detection value at the output of the neural network and the number of 2D multiple detections, respectively 2D aggregate values. This shows that its is important to set correctly the decision threshold, because strong multiple detections can occur also at non-face positions - but with lower detections probabilities.

Features	Face 1	Face 2	Face 3	Face 4	Face 5
# 3D-Detections	2	2	2	2	2
# 2D-Detections	16	12	9	11	5
# 2D-Aggregates	37	17	14	14	8
Detection Value	0.96	0.94	0.94	0.93	0.92
Dominant Scale	0	1	1	1	1

TABLE 3. Multiple detection results

Face number five got detected with the lowest detection values of all faces. This due to its slightly sidewards position.

2D-Aggregate values are important to indicate, whether at a given location a face has been detected or not. Figure 35 compares the number of summed 2D-Aggregate values that occurred on the faces locations in Figure 34.

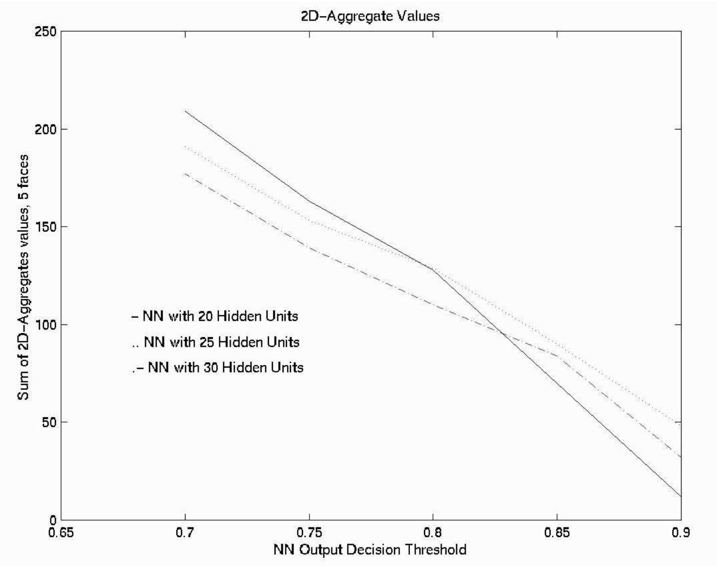


FIGURE 35. 2D-Aggregate Values, Network Comparison

Networks with fewer hidden units produce in general more double detections, because these can probably less specialize to face features than networks with higher numbers of hidden units.

6.5.2 Decision Threshold Settings

One must set the decision threshold for the output of the neural network according to the network currently applied. It depends on how the network was trained (number of epochs, training base, initial weight settings). One can only estimate the appropriate threshold by doing some test runs using a variety of thresholds. A good measure is to maximize the number of correct detections divided by the number of missed detections plus the number of false detections, see Figure 36.

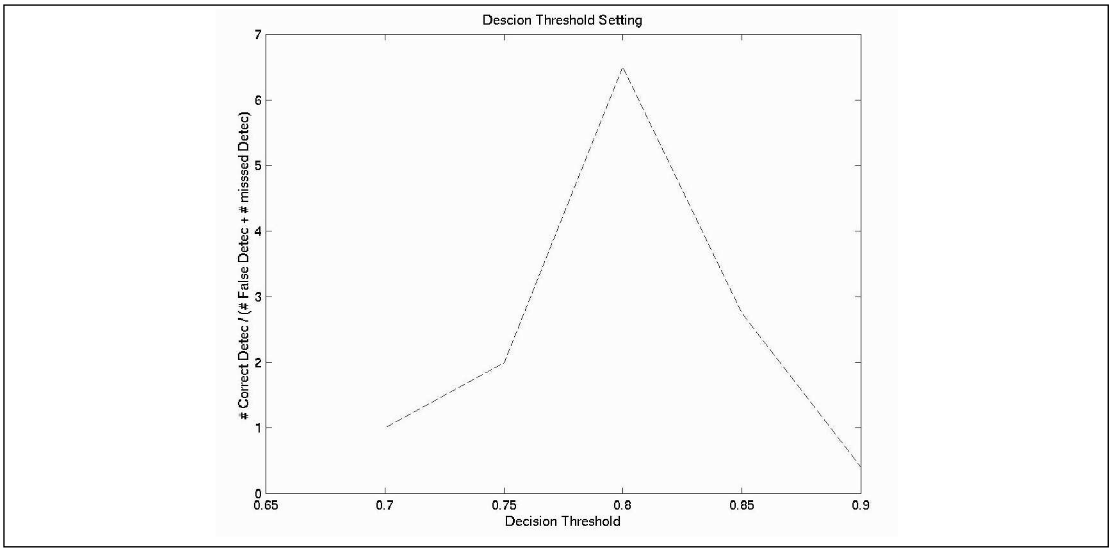


FIGURE 36. Decision Threshold Comparison

The maximum of the curve is at 0.8. That's the value we should adopt as decision threshold for the given network in order to have a maximum of faces detected with a minimum of false detections. Decision thresholds vary heavily with the neural network applied. We found optimum settings that were as low as 0.3 or as high as 0.9.

A decision threshold must also be set according to the arbitrating strategy applied. With a strategy that is very selective, this threshold must be set to a quite low value in order to get enough multiple detections / aggregate values to allow a face being detected. On the other hand when the only postprocessing measure applied is the decision threshold (strategy one), we must set it to a high value in order to reduce the number of false detections. See also Section 6.5.1 on page 42.

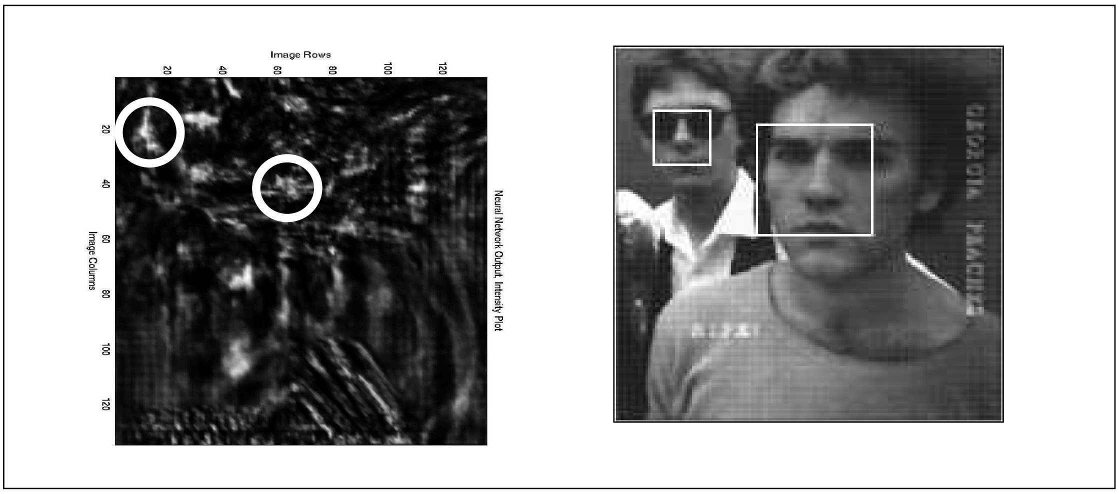


FIGURE 37. Sample Image and NN Output

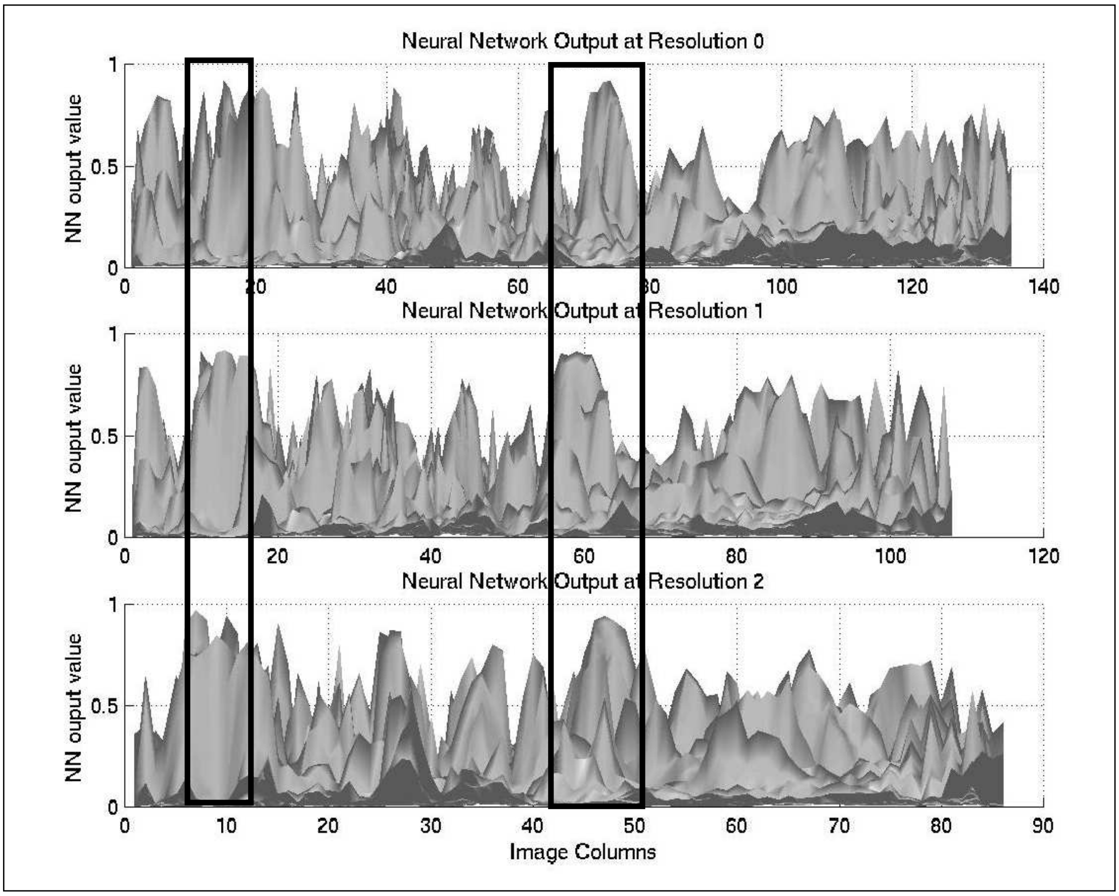


FIGURE 38. NN Output, 3 Resolutions

Figure 38 shows neural network outputs obtained at three different resolution using as the input the image depicted in Figure 37, viewed from the left side to the right. The marked zones correspond to the face locations and can be distinguished by high detection values as well as resemblances in shape. Clearly, resolution two is dominating, the arbitrator will finally draw detection squares conforming to this dimensions.

Another point is to determine a decision threshold without using an priori fixed decision threshold, so the detection process could adapt itself to various conditions. The mean detection values of the sample are 0.1131 for resolution zero and 0.1187 respectively 0.1273 for resolutions one and two, which are quite low in comparison to the maxima that are close to 1. Clearly, the average can't be taken directly as a measure to estimate a decision threshold. A better approach consists of setting the decision threshold to about 90% of the highest peak found in the neural networks output list. This would allow to take into account most 2D multiple detections situated on the upper parts of the peaks found in Figure 38.

6.6 Test Runs on CMU Data-Base

In this section we present tests that were performed on an image database (test set A) provided by CMU[1] and allows to compare detection results directly to those obtained

1. http://www.cs.cmu.edu/~har/faces.html

by other research groups.
Test set A contains forty images with different backgrounds, illumination and faces in all scales.

6.6.1 Trained Neural Networks

Table 4 and 5 show the neural networks that have been trained for our face detectors. The main differences consist in the trained image size, preprocessing type and different initial seeds (resulting in different initialization of the neural networks weights).

Net Features	GreenNet	Red1Net	Red2Net
# Hidden Units	25	30	30
Trained Image Size	25 x 25	25 x 25	25 x 25
Preprocessing Type	Centered	Centered	Centered
# Training Epochs	400	400	108
# Bootstrap Iterations	2	2	2
Final Errors (Sum / Out)	< 25	< 25	24.8
Initial seed	Type 1	Type 1	Type 2
Data Base + # Clips	Standard + 60	Standard + 60	Standard + 150

TABLE 4. Trained Neural Networks 1

Net Features	BlueNet	YellowNet	OrangeNet
# Hidden Units	20	15	30
Trained Image Size	25 x 25	25 x 25	25 x 25
Preprocessing Type	Centered	Centered	Stand. Dev.
# Training Epochs	132	200	100
# Bootstrap Iterations	2	2	2
Final Errors (Sum / Out)	25.4	26.3	15.38
Initial seed	Type 1	Type 1	Type 1
Data Base + # Clips	Standard +	Standard +	Standard +

TABLE 5. Trained Neural Networks 2

Both the average neural networks output error (over all trained images) as well as the sum of the hidden and output error during one epoch were measured to indicate the level of adaptation of the trained neural networks.

The training data base (Standard, without bootstrap clippings) consisted of about 1500 face images (700 faces and mirrored versions) and about 3500 non-faces.

The network with a sliding window size of 12 x 12 was used only in combination with the camera (Moving Picture Face Detector), as we can then do face detection on smaller input frames which means faster execution.

6.6.2 Post/Preprocessing results

To evaluate the effectiveness of the multiple detection arbitrator, comparisons were made between the raw output obtained by the neural networks and those being post-processed by the arbitrator applying strategy 5. We obtained following results with the test set A:

Features - GreenNet	Without Postprocessing	With Postprocessing	With Postprocessing + Blurring
# false detections	1316	466	278
# correct detections - total nb	110 - 169	95 - 169	68 - 169
# missed detections	59	64	101

TABLE 6. Results With / Without Arbitration (2D,3D-Multiples)

Note that overlapping false detections were counted as one false detection. The number of faces detected is far from the total number of faces contained in the images. This is due to the fact that we did just rescale the original images and processed these at two different resolutions in order to produce the 3D-Multiples needed by the postprocessing strategy.The relative changes in Table 6 are clearly visible. The postprocessing strategy applied showed to be quite effective in order to reduce false detections without changing much on the correct detection rate. Blurring reduces false detections further on but has a quite negative effect on the number of faces being detected. This method should only be used with a single neural network and when doing either scans at only one resolution or at different resolutions that are not close to each other, so we couldn't apply the 3D-Multiples postprocessing strategy.

6.6.3 Multiple Network Arbitration

In order to reduce false detections, two differently trained neural networks were combined, using a network arbitrator to decide on possible face locations. The arbitration distance (maximum distance at which to detection are combined) was set to 5 pixels. The result is compared to those obtained using a single network:

Features	Single Network Red1Net	Two Networks Red1Net+Red2Net
# false detections	957	355
# correct detections	132 - 169	124 - 169
# missed detections	37	45

TABLE 7. Comparison Single / Multiple Neural Networks

When doing multiple network arbitration, the input shouldn't be blurred, in order to get a maximum of positive face detections. The decision thresholds should also be set lower than usual, resulting in more faces being detect. An increasing number of false detection can be bypassed, due to the fact that these occur normally in different locations.

6.7 Vector Normalized / Standard Deviation Preprocessing

For comparison purpose, we present here some results obtained with the normalized (vector length normalized) version of the face detector. We included also a version that doesn't employ preprocessing at all. All neural networks were trained on relatively small databases not applying the bootstrap algorithm.

Note the high Epoch rates, necessary due to non-diving the input image by the maximum of the luminance, therefore having high values (0 .. 255 instead of 0 .. 1) at the input of the neural networks which results in poor learning rates.

Four neural networks were trained differently:

Features	one.net	two.net	three.net	four.net
# Faces	500	500	500	1528
# Non-Faces	500	500	1600	1528
# Hidden Units	10	25	25	25
Interval Size	1	1	1	1
Epoch Size	300	300	1000	1000

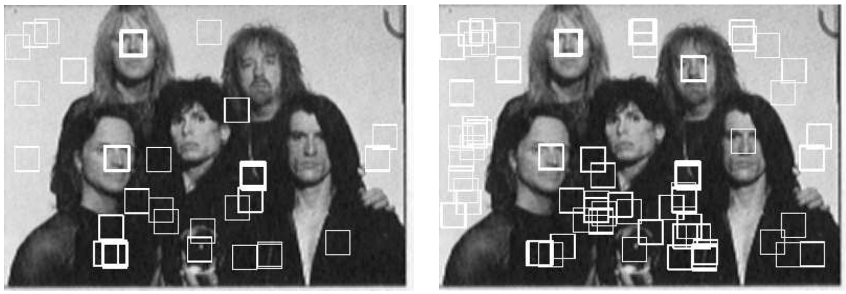
TABLE 8. Four different neural networks

Remarks concerning the chosen parameters:

- The first two networks don't use all the images available in the databases, to be able to compare the neural networks behaviour with the next two networks
- *Network one* has only 10 hidden neurons which makes it faster during the detection process (more than doubling the speed) but reduces also its capability to store facial features
- *Network three* was trained using all of the non-faces but just 500 faces. This with the idea in mind that some hundreds of faces should be enough to give the network an impression, of what a face looks like
- *Network four* was trained with all face images available and almost all non-face images, an equal number was chosen to have a regular face - non-face succession during the training phase. The image interval (series of the same type of images (either faces or non-faces) that follow each other during the training phase was set to 2, a reasonably low value which results in a large learning rate
- 25 hidden units showed to delivers acceptable results. These may not deliver optimal results, but not too great a number of hidden units should be chosen in order not to increase too much the execution time of the face detector

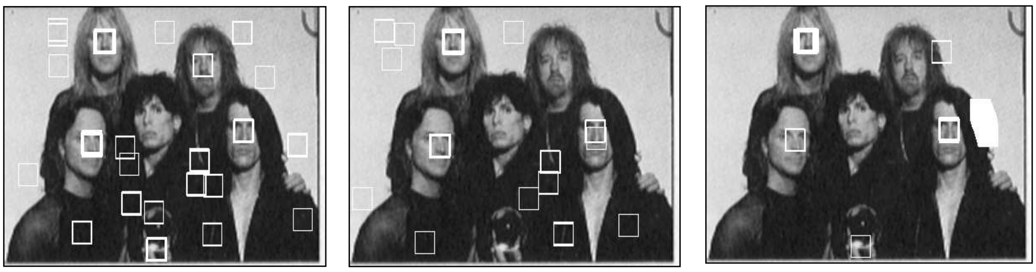
Note: Between network three and four we have a difference not just in the size of the database used but also the number of epochs the training has been gone through, as well as a different interval size. Obviously, in order to be able to compare the results properly, just one parameter should change at a time. The reason why we still included network four is for giving an overview.

Note: the sliding window seems to be rather small in comparison to the size of the of the faces found in the test images below, i.e. faces captured with the sliding window below contain sometimes no mouth and just parts of the eyes. A successful detection is still possible because the learning process was carried out using various face positions.



one.net two.net

FIGURE 39. Face detection results (one.net + two.net)



three.net four.net two.net

Vector Normalization **No Preprocessing**

FIGURE 40. Face detection results (three.net + four.net)

Features	one.net	two.net	three.net	four.net
# correct detections (multiples)	18	28 // 34	38	24
# false detections	60	134 // 51	51	17
Total nb of faces detected	78	162 // 85	89	41
# correct / # false	0.3	0.21 // 0.67	0.75	1.41
Normalization	ON	ON // OFF	ON	ON
Test image	Aero 272x392	Aero 272x392	Aero 272x392	Aero 272x392

TABLE 9. Number of faces detected

Of interest in Table 9 is the quotient of the number of correctly detected faces (multiple detections shifted by a few bits are allowed) and false detections. This value gives us an indication to which extent one can trust the results delivered by the face detector. Network four tops on all the other networks. But one should also consider that network two in its non-normalized version did false detections virtually just in one place of the test image (and multiple times for the same motif). If one would include the falsely detected element into the non-face learning set, false detections would drop down to almost zero for this image.

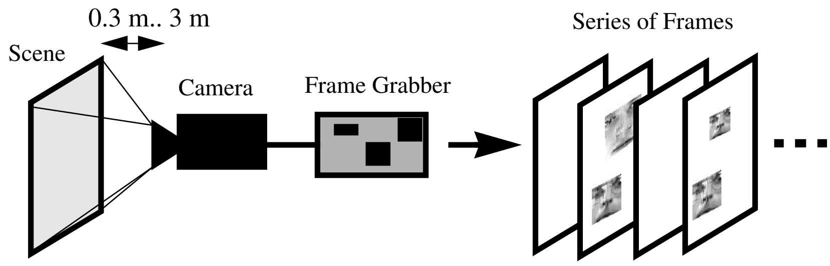
One can see immediately, that the longer the neural networks have been trained (more epochs accomplished), the lower is the false detection. On the contrary, correct face detections may not profit too much from higher epoch rates as suggests Figure 40.

Surprisingly well behaved the *non-preprocessed* network with 3 faces detected and false detections virtually just in one place. A reason for this probably, that with normalization activated, the network is able to detect faces for example in a dark background that look just faintly like faces.

7 Moving Picture Face Detector

The Moving Picture Face Detector consists of a camera, a frame grabber and a program processing the frames delivered by the latter one and doing face detection on the input frames in real-time. The resulting frames marked with a square on locations a face has been found are then displayed on a Monitor, see “The Face Detection System” on page 2.

Figure 41 shows the setup of the Moving Picture Face Detector. A cluttered scene is being captured by a camera connected to a frame grabber (Osprey-1500, see Section B-1 on page 78), transforming the incoming video stream into a series of frames (30 frames per second).



The Sliding Window is always of constant size - *25 x 25 pixels.* Larger faces are detected at a different scale obtained by sub-sampling the input frames

FIGURE 41. System for the detecting faces in a live video stream

Frames delivered by the camera are of size 288 x 384, 144 x 192, 72 x 96. For speed reasons only the latter two sizes are of interest. With a square sliding widow of size 25 x 25, an input image of size 144 x 192 is an appropriate when doing face detection at an intermediate distance from the camera (30 cm to about 3 m). When using a square sliding window of size 12 x 12, a frame size of 72 x 96 is a good choice.

7.1 Theory

Two methods lead to an improved speed-up: Active zone search restricts the search on those areas containing movements. They are smaller than the original input image which means less area must be scanned for faces. The computation of the squash function can be done by employing an adequate approximation as will be shown after the next sub-section.

7.1.1 Active Zone Search

The idea here is to define active search areas based on the differences found in subsequent pictures in a given stream. Using such a sub-zone clipping, an important reduction in execution time should be obtained. Especially interesting is the fact that smaller images are handled in a non-linear faster way by the cross-correlation operation used throughout this project, see also speed-up curve in Figure 18.

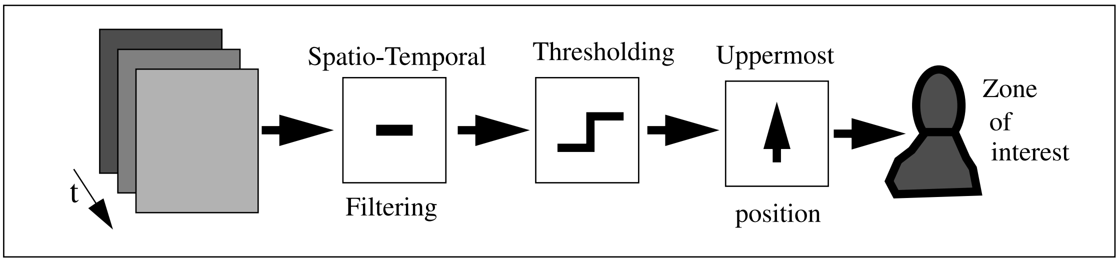


FIGURE 42. Simple Head Tracking system

Initially, there is a scan for faces through the whole first frame, in next frames one would only concentrate on zones of interest (active zones).

Figure 43 shows how the precomputed clipping form (it can be prepared in the off-line section of the detector) that gets applied to the upper part of the moving area where the face may be found with a high chance. If no new detection has been obtained, the previous one stays valid.

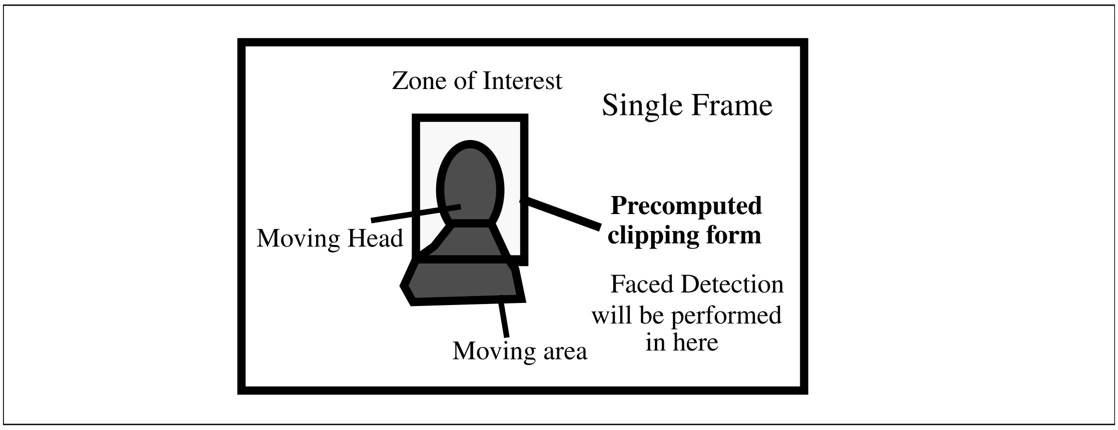


FIGURE 43. Single Frame containing a zone of interest

7.1.2 Execution Time Reduction

In order to reduce execution time to maximum, several measures have been taken:

- IFFT Measurements: The FFT subroutine starts during the initialization process a short series of measurements in order to adapt some internal parameters optimally to the machine the FFT transform is later used on.
- Compiler options have been activated in order to receive faster code. See Section B-2, “Software,” on page 78.
- The code was written as lean as possible, using examples found in [16].
- The sorting algorithm applied in the output list is of QuickSort type

Neural networks simulations often spend a large proportion of their time computing exponential functions. Since the exponentiation routines of typical math libraries are rather slow, their replacement with a fast approximation can greatly reduce the overall computation time. The exponentiation is being approximated by manipulating the components of a standard floating-point representation, hence modelling the exponential function as well as a lookup table with linear interpolation, but is significantly faster and more compact [10].

The squash function $g(x) = 1/(1+e^{-x})$ at the output of each neuron can be computed more efficiently than presently done using the standard C math library.

In [10] a speed-up factor of about 10 had been obtained on UltraSparcs running Solaris. Using this method leads to a reduction of the execution time in the on-line process, as the squash function is called ω times by the FNN during a test image scan:

$$\omega = (q+1)(S-m+1)(T-n+1) \qquad \text{(EQ 44)}$$

where q is equal the number hidden units, n the sliding window size and S, T the number of rows respectively columns of the input image. See also Figure 17.

7.1.3 Measured Execution times

In Figure 44 we see the execution time needed by a detector based on the fast neural network using FFTs. There are strong oscillations, but both the maximum and the minimum values follow exponential curves.

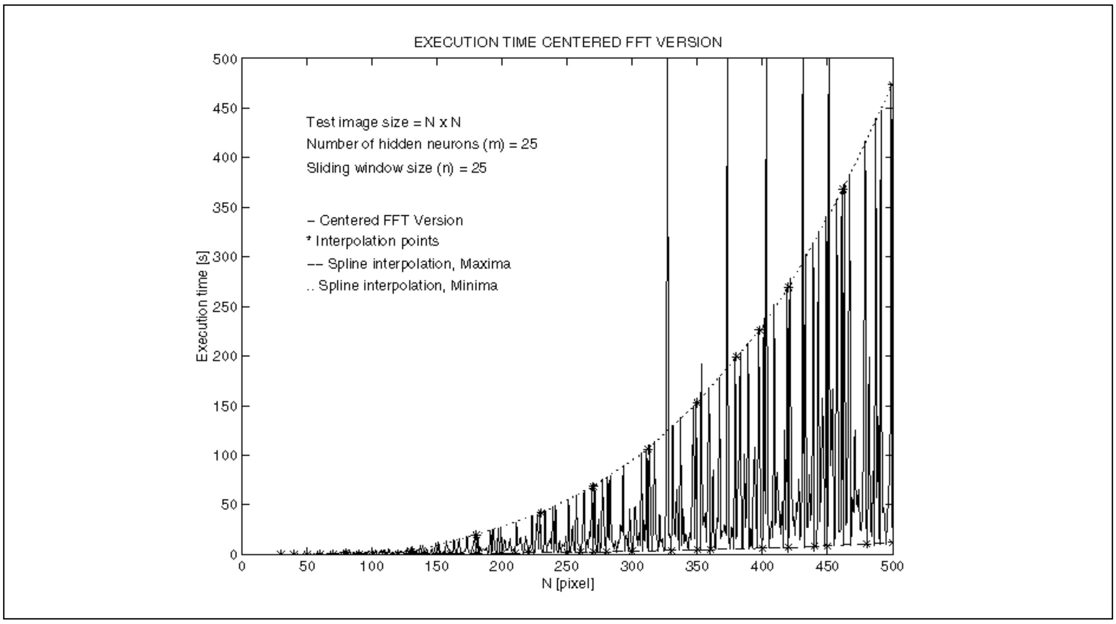


FIGURE 44. Execution times comparison FNN raw and interpolated

Having a closer look reveals even values in between that demonstrate the same exponential growth. The maximum values represent those cases where general purpose algorithms must be employed to compute the FFT. On the other hand, the minimums reveal the real power of the Fast Fourier Transform. That's were we use the fast fourier neural network. Note the five spikes that don't follow the general behaviour, they were caused by erroneous measurements in our program and can be neglected.

The dimensions of the input images must be chosen carefully [18], in order not to fall into a region where the FFT library must apply a general algorithm. Especially in multi-scale mode, one can easily obtain dimensions producing slow computations. Special search methods have been implemented that find, on the base of the above given formula, valid dimensions.

7.1.4 Reduced Sliding Window Size

Gaussian Blurring of the input image as described in Section 6.1.1 results in a reduction of the information delivered to the neural network. Nevertheless, it performs well and produces less false detections. So there must be a redundancy of information carried in the input image, namely in the high frequencies. Cutting off high frequencies parts and padding the reminder with zeros is the same as taking a smaller input image and enlarging it by adding zeroes in the frequency domain. With a smaller input image we must also reduce the sliding window size in order to still able to detect faces at the same scale. But of interest is the smaller input image, as it gets treated as a whole in the cross-correlation operations performed in the hidden units of the fast neural network described in chapter 5.

With an input image size 72 x 96 (one of the resolutions sustained by the frame grabber), the sliding window size should be set to 12 x 12 in order to get a similar operation range (distance from the camera, where face detection is possible).

7.1.5 Multi Resolution

With the Moving Picture Face Detectors, we cannot scan at a multitude of resolutions like in the case of the still image face detector. Since we have time restrictions, scans at two different resolutions must be sufficient to cover a large field. Therefore, post-processing arbitration can't be based on 3D-Multiples, this leaves us strategies 1-3 for arbitration, see Section 6.2.4

On the other hand if detections must take place at more or less the same distance from the camera, two close scans are would be possible and we could then also use strategies 3-5.

7.2 Implementation

Figure 45 shows the data flow in the Moving Picture Face Detector. After a general initialization, the detection engine is getting started, searching the frames delivered by the camera for faces on different scales (resolutions). Then the detection results are being arranged in a list for the multiple marking purpose. Finally a multiple Detection Arbitrator decides where a face are located in the input frame. Marked frames are displayed in real-time on a monitor.

Note that the Moving Picture Face Detector employs only one single neural network, sequentially applied on different scales. We cannot use multiple neural networks with network arbitration as described in Section 6.3, as we do have time constraints. Unless we could run several networks parallel on different machines, see also Section 10.2.

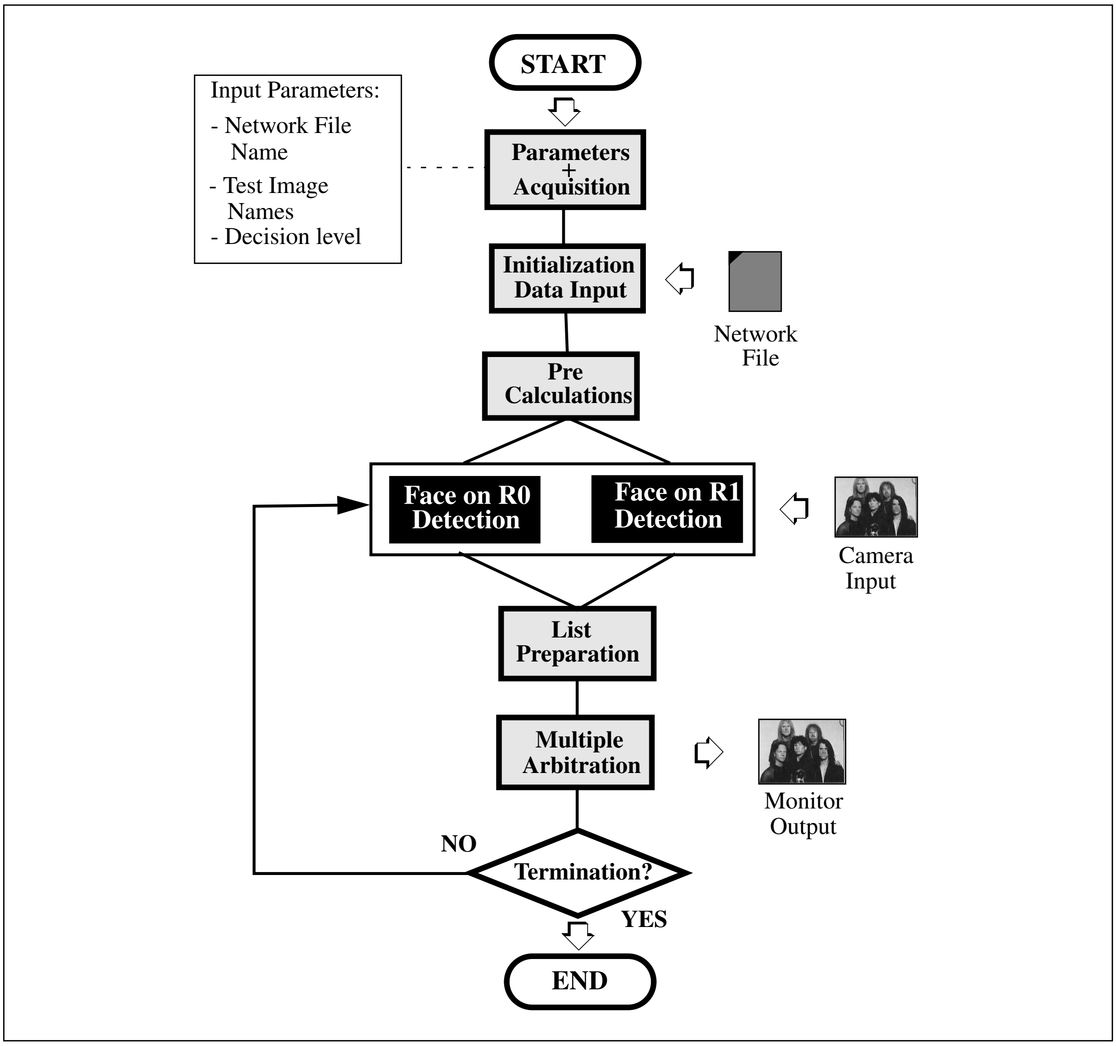


FIGURE 45. Data Flow in the Moving Picture Face Detector

7.3 Results

As the detection engine used with the Moving Picture Face Detector is the same as the one described in the Still Image Face Detector section, we present here primarily results that concern the execution time of the Moving Picture Face Detector.

7.3.1 Camera Series

Given here are two examples obtained from real-time face detection using a camera. The face detector delivers pictures at a rate of about 1.5 frames/s. See also Figure 51 - Figure 55.

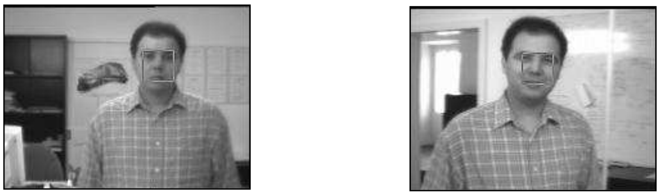


FIGURE 46. Two pictures from a live series taken by our camera (0.7 s interval)

We experienced relatively few false detection (in an office environment) after the introduction of an appropriate post-processing strategy 3. In Figure 51 for example we have zero false detections and just one missing positive detection in 40 frames.

7.3.2 Execution Time Speed-Ups

The following time values have been obtained running the programs on an Ultra-Sparc 30 under Solaris with an input image size of 144x 192:

- Time needed for a single FFT transformation: 0.01 s
- Time needed to calculate 25 hidden units: ~ 0.6s
- Total time for the on-line process (with normalization): ~ 0.7s
- Total time needed by the legacy network: ~ 11s

Here, the speed-up factor for the FNN network obtained was: $11/0.7 = 15.7$. This is somewhat faster than one would expect, see also Figure 18 for comparison. This may be due to the fact that the code used for the legacy network hasn't been optimized for speed.

7.3.3 Required Resources

The fast face detector in its serial implementation must run on a fast machine in order to obtained acceptable refresh rates of the detected scene. On an Ultra Sparc 30 the processing time needed per frame is 0.6s for an image of size 144x192.
Another point should not be neglected either: the amount of memory needed by the program. Running detections on images of size 144 x 192, the still image face detector needs about 33 Mb of RAM, with the Moving Picture Face Detector about 150 MB of RAM is mandatory, depending on the kind of preprocessing chosen. A vector normalized or standard deviation normalized version may demand more. When using the detector on larger images (for example when scanning still images), about 427 MB or more of RAM is needed when the images size is in the order of 500 x 500.

8 Discussion

One has to be careful when comparing results obtained by different face detection systems. There are various parameters to be taken into consideration. Test sets differ in difficulty. The one we used (Test Set A) contains a variety of images (40 in total), featuring real-world scenes.

Here follows a comparison of our still image face detector with another neural networks approach (Rowley et al. [2]), that is close to ours in the sense that it uses also a feed-forward neural network and a comparable training image size (20 x 20 pixels):

Type	Rowley's System	Our System	Rowley's Results	Our Results
Single Network, using Heuristics	**Net1** 3D-Multiples Overlap elim.	**GreenNet** 2D-Multiples 3D-Multiples	**85.8%** corr. 222 false	**71.1%** corr. 496 false
Single Network, using Heuristics	**Net2** 3D-Multiples Overlap elim.	**RedNet** 2D-Multiples 3D-Multiples	**84%** corr. 179 false	**73.3%** corr. 957 false
Arbitration Among two Networks	**Net1 + Net2** AND	**GreenNet + RedNet**	**80.5%** corr. 67 false	**69.5%** corr. 355 false
Fast Version	Candidate Verification Method	-	**65.7%** corr. 3 false	-

TABLE 10. System Comparisons

This test was performed on the Test Set A, as mentioned above. Rowley's networks are of retinally connected type. Net1 features 2905, while Net2 has 4357 connections. The first one uses 52 hidden neurons, the latter one 78. The preprocessing strategies applied are advanced, employing mask, linear fitting functions and histogram equalization. Postprocessing is done by 3D-Multiple detection and elimination of overlapping detections.

Table 10 shows that our system has lower positive detection rates and produces more false detections than their system. This is due to simpler network training on our side (we used a training base of about 1500 face images and 4000 non-face examples, while they trained with 16000 face images and 9000 non-face images, where 8000 were obtained by applying the bootstrap algorithm several times. We on the other hand collected only 300 bootstrap clippings during two bootstrap operations. Another point is the extensive preprocessing applied by the other group. Histogram equalization for example is a non-linear operation and can't be integrated into the Fourier framework of our system.

Note that the rather high number of false detections on our side was caused by only three images of the test set, that contained printed letters (parts of a book). Unfortunately the lines where separated with about the same distance found between the eye line and the mouth in a sliding window. This lead the system do reproduce always the same errors. Just a few non-face examples of this type in the training set would resolve this problem though.

The fast face detector of the other group produced a rather low detection rate, because they skipped the sliding window over the test image as described in Section 4.2. This caused also lower false detections, because less positions were parsed and also due to their candidate verification method applied (basically consisting of a combined coarse and fine scan of the test image).

Both our vector normalized preprocessing method, as well as the standard deviation method showed to perform poorly, yielding many false detections. The difference between these, is the fast adapting to the training set of the latter one during network training. The vector normalized method had to be trained with high epoch rates, as the error of the hidden units decreased only slowly. This is probably due to the fact that homogenous zones in the input image caused a high normalizing denominator, which in turns lead to a high activation in the neural network.

The YellowNet (15 hidden units) and the BlueNet (20 hidden units) performed quite well, but they cant compete with networks containing 25 hidden units or more, as they tend to produce high false detection rates. But even though, they can be applied in the Moving Picture Face Detector with acceptable face detection rates and due to the decreased number of hidden units, they execute also faster than their counterparts with more hidden units.

9 Conclusions

The Still Image Face Detector delivered acceptable results on a variety of images as has been shown in the result section. Currently, there are still too many false detections, but this may be overcome by advanced network training. Especially the novel 2D-Multiple marking based postprocessing showed to be a powerful method in reducing false detections. It can be applied both with single and multiple neural networks. In contrast to other systems, our face detector perform a full search of the input frames which means an improved face detection capability.

The fast fourier based neural network described in this report consists of a 100% software approach and allowed the Moving Picture Face Detector to do real-time face tracking, running on a single standard workstation with following characteristics:

- 0.6 s / frame, 144x192 frame size, NN with 25 hidden units
- 0.23 s / frame, 72x96 frame size, NN with 25 hidden units
- 0.17s / frame 72 x 96 frame size, NN with 20 hidden units

Although progress has been made, the Moving Picture Face detector still lacks some robustness:

- No direct sun light exposure of a scene where faces must be detected, because this results in partly shaded faces that cannot be detected due to missing normalization of the input (centered only version).
- Glasses with dark frames or glasses mirroring light lead to non-detection of faces present in a given scene
- Rotated faces, lateral view, partly obscured faces

The Moving Picture Face Detector demonstrated its capability during a demonstration, where people presented themselves before a camera and were detected successfully by the system.

Martigny, June 30 1998

Beat Fasel

10 Possible Extensions

Following suggestions could be of interest for future extensions of this project:

10.1 All Frequency approach

For each hidden unit in the Fast Fourier based neural network, we must apply a back-transformation, which means a considerable time spent doing Fourier transforms. We came up with the idea of reformulating the squash function in the frequency domain. This would allow to combine the outputs of the hidden units directly in the frequency domain and doing so, only one back-transformation would be necessary for all hidden units. The problem though is how to formulate the squash function in the frequency domain. An alternative consist of introducing a modulation $xf(x)$, replacing the current squash function (here being tanh), where

$$f(x) = \frac{\tanh(x)}{x} \quad \text{(EQ 45)}$$

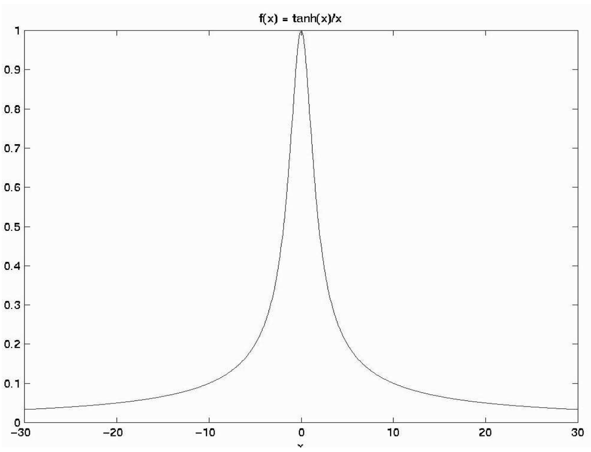


FIGURE 47. Modulation function

The following property of the Fourier Transform holds:

$$x(t)\cos(2\pi f_0 t) \Leftrightarrow Y(f) = \frac{1}{2}[X(f + f_0) + X(f - f_0)] \quad \text{(EQ 46)}$$

A similar property governs also a modulation by a sine. The idea here is to approximate $f(x)$ by a *Fourier Series*. A few terms may be sufficient to model the function. Note that functions described by fourier series are periodic. This may not be a problem as we would use $f(x)$ in the first interval - output values produced by the neurons seem to be limited (see also Figure 19).

10.2 Parallel Multiple Networks

Using a reduced image size, faster program code, and a fast squash function [10], a pure sequential implementation yielded acceptable results and a frame rate that still allowed a dynamic detection. But if high frame refresh rates are mandatory or the deployment of several different networks in combination would be an issue, there would be too much computations to be handled by a single computer, so one would eventually be forced to use a cluster of computers. The MPI (Message Passing Interface) showed to be an interesting approach for implementing programs in parallel.

Parallelization of the Face detector can be done either

- by using a machine with several processors
- or distributing the task over a cluster of workstations

The question that arises is how to parallelize the program in order not to loose too much time with communication between the sub-processes (granularity). This means that one should probably try an implementing on either the neuron os neural network level, instead of doing FFTs in parallel.

10.3 Lighting Independence

One of the main weakness of the present face detector is its dependence on good lighting conditions. The light source should be in front of face to be detected. Partly shaded faces cause enormous problems. Nonlinear methods (such as histogram equalization) don't work because they must be integrated into the FFT framework which is not feasible. Linear fit functions such as applied in [2] may improve the situation.

10.4 Network Combinations

In addition to the presently trained networks, some others may interesting to consider, possibly in combination with the first:

- Neural networks using different preprocessing strategies
- Flipping neural networks - same networks, but with a flipped (mirrored) input - faces are symmetric!

10.5 Advanced Neural Network Training

The training process can be speeded-up, thus gaining also in quality when using a method called boosting technique [11], so far especially employed in speech recognition research due to the considerable amount of computation that is required to train networks for large vocabulary speech recognition. The basic idea is a procedure that make selective use of training data to increase performance. Boosting is based on an ensemble of networks that are trained sequentially on data that has been filtered by the previously trained networks in the ensemble. This ensures only data that is likely to result in improved generalisation performance is used for training.
The whole training process described in section 3.1, should be automated. This way, the network trainer could be able to teach dynamically the neural network under training the best images at a given moment - automatic supervised learning. Images produc-

ing constant bad result could be clipped automatically out of the current database, re-introduced later on, or eliminated completely after user supervision. Training sets delivering good results may be momentarily taken out of the database in order to allow faster training on images less well handled by the network, this way one could combat forgetting of previously learned material, which manifests itself with the use of large databases and therefore fewer repeated epochs.

10.6 Rotation Faces Scan

The aim here is to bring faces in the input image into an up-right position for better detection. One could also rotate the weights of the hidden units that operate at the momentary position of the sliding window to obtain the same result. Direct treatment in the frequency domain is possible and rotation of the weights can be done in the off-line section which means no further overhead:

$$f(r, \theta + \theta_0) \Leftrightarrow F(w, \phi + \theta_0) \tag{EQ 47}$$

We used the polar coordinates, r is the range, θ the angle. The weights of the neural network may be turned in the frequency range which results also in a rotation in the time domain. This way, turned faces may be detected and this in a straight forward manner without the need of sophisticated set-ups such as for example a preprocessing rotator neural network as described in [3].

11 Table of Symbols and Abbreviations

Note: Matrices are held in capital letters.

q	Number of hidden units (neurons)
n	Sliding Window columns (if square also rows)
m	Sliding Window rows
h_i	Activity of a particular neuron *i* in the hidden unit H
H_i	Activity of a particular neuron *i* extended to the global image $\Im$
o	Output activity
$\Im$	Global input image
S	Row of the global input image $\Im$
T	Column of the global input image $\Im$
w	Neural Network weights
$[X]_{rc}$	Sub-image located at position *(r,c)*
$g(x)$	Activation function or squashing function
G	Gaussian mask
N	Normalized activation
C	Combined activation
Φ	Filter of size *mn*

NN	Neural Network
CNN	Classical Neural Network
FNN	Fast Neural Network (Based on FFT)
MLP	Multi Layer Perceptron, Neural Network

12 References

12.1 Papers

[1] Souheil Ben-Yacoub, “Fast Object Detection using MLP and FFT“, IDIAP-RR 9711, 1997

[2] H.A. Rowley, S. Baluja and T. Kanade, “Human Face Detection in Visual Scenes“, CMU CS Technical Report, CMU-CS-95-158R, Carnegie Mellon University, 1995

[3] H. A. Rowley, S. Baluja, T. Kanade, “Rotation Invariant Neural Network-Based Face Detection“, CMU-CS-97-201, Carnegie Mellon University, 1997

[4] R. Chellappa, S. Sirohey, C.L. Wilson, C.S. Barnes, “Human and Machine Recognition of Faces: A Survey”, Technical Report, CAR-TR-731, University of Maryland, 1994

[5] K. K. Sung and T. Poggio, “Example-based Learning for View-based Human Face Detection”, C.B.C.L Paper No. 112 MIT, 1994

[6] D. Valentin, H. Abdi, A. J. O‘Toole, G. W. Cottrell, “Connectionist Models of Processing: A Survey“, Pattern Recognition. Vol. 27, pp. 1209-1230, 1994

[7] M. Turk and A. Pentland, “Eigenfaces for Recognition“, MIT, 1991

[8] A. J. Colmenarez and T. S. Huang, “Face Detection with Information-Based Maximum Discrimination“, University of Illinois, 1997

[9] S. Oka, M. Kitabata, Y. Ajioka and Y. Takefuji, “Grouping Complex Face Parts by Nonlinear Oscillations”, Keio University, 1998

[10] N.N. Schraudolph, “A Fast, Compact Approximation of the Exponential Function“, IDSIA-TR 9807, 1998

[11] G.Cook, T Robinson, “Boosting the performance of connectionist large vocabulary speech recognition“, Cambridge university, 1997

12.2 Books / Booklets

[12] B. Moghaddam and A. Pentland, “Probabilistic Visual Learning for Object Representation”, MIT, 1995

[13] R. C. Gonzales and R.E. Woods, “Digital Image Processing”, Addison Wesley, 1993

[14] Message Passing Interface Forum, “MPI: A Message-Passsing Interface Standard”, 1994

[15] N. MacDonald et al., “Writing Message Passing Parallel Programs with MPI”, Edinburgh Parallel Computing Centre

[16] Kenneth A. Reek, “Pointers on C”, Addison Wesley, 1998

[17] Sun, “Sun Video Plus for PCI, User’s Guide”, 1997

[18] M. Frigo, S.G. Johnson, “FFTW 1.3 Users’s Manual”, 1998

[19] C. M. Bishop, “Neural Networks for Pattern Recognition”, Oxford, 1996

[20] S. Haykin, “Neural Networks”, Macmillan, 1994

12.3 Internet Resources

- Homepage of the fast face detector:
  http://www.idiap.ch/vision/faceresults.html
- The face recognition home page:
  http://www.cs.rug.nl/~peterkr/FACE/face.html
- The computer vision home page:
  http://www.cs.cmu.edu/~cil/vision.html
- Facial analysis:
  http://mambo.ucsc.edu/psl/fanl.html
- Face detection:
  http://www.cs.cmu.edu/~har/faces.html
- FFTW Homepage
  http://theory.lcs.mit.edu/~fftw/
- Source code of MLP
  http://www.cs.cmu.edu/~tom/faces.html

Appendices

A Face Detection Test Runs

Both still images and moving pictures were evaluated by either the Still Image or the Moving Picture Face Detector. The most interesting samples can be found further below.

A-1 Still Images

The following images were scanned with the Still Image Detector. As we don't have time restrictions, some more sophisticated methods (3D-Multiple Detections, 2D-Aggregate detections and several neural networks in parallel) can be applied than in the case of the Moving Picture Face Detector.

A-1-1 CMU Data Base

The following results were obtained using a single neural network with 30 hidden units and a fixed decision threshold of 0.85. Detection was performed at two resolutions close by (scale factor: 1.2).

There's a label in the upper left hand corner of the images, indicating how its was prep-ocessed, respectively the results obtained.

Here is an example: **[N,B, 3-5, 2]** means Normalized, Blurred, 3 correct detections out of 5 possible and 2 false detections. *C* stands for centered and *W* for without blurring.

Even though the face detector had been trained on natural faces, it can also detect hand-drawn ones, as can be seen in Figure 48.

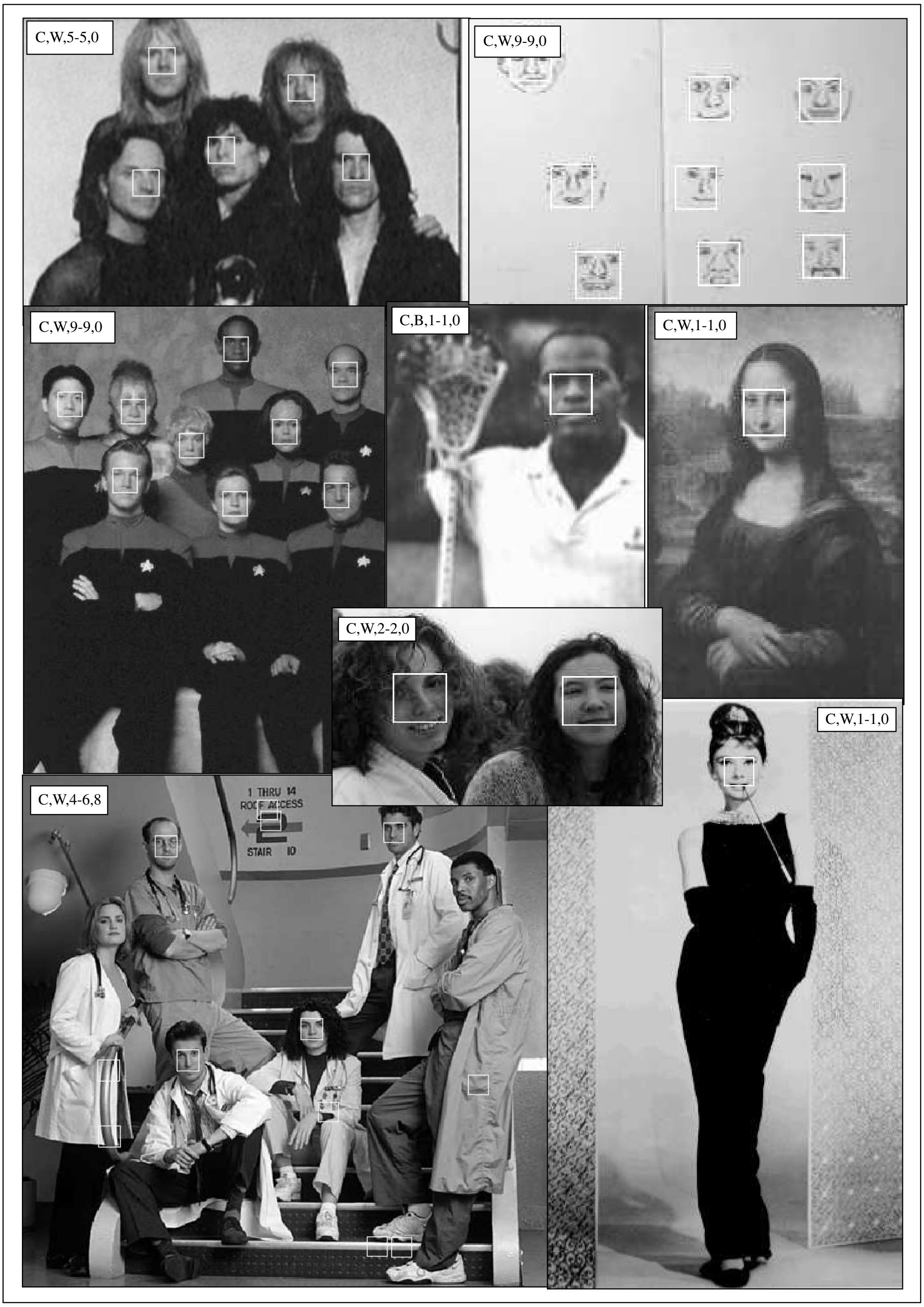


FIGURE 48. CMU Data Base Test Examples (Single Network)

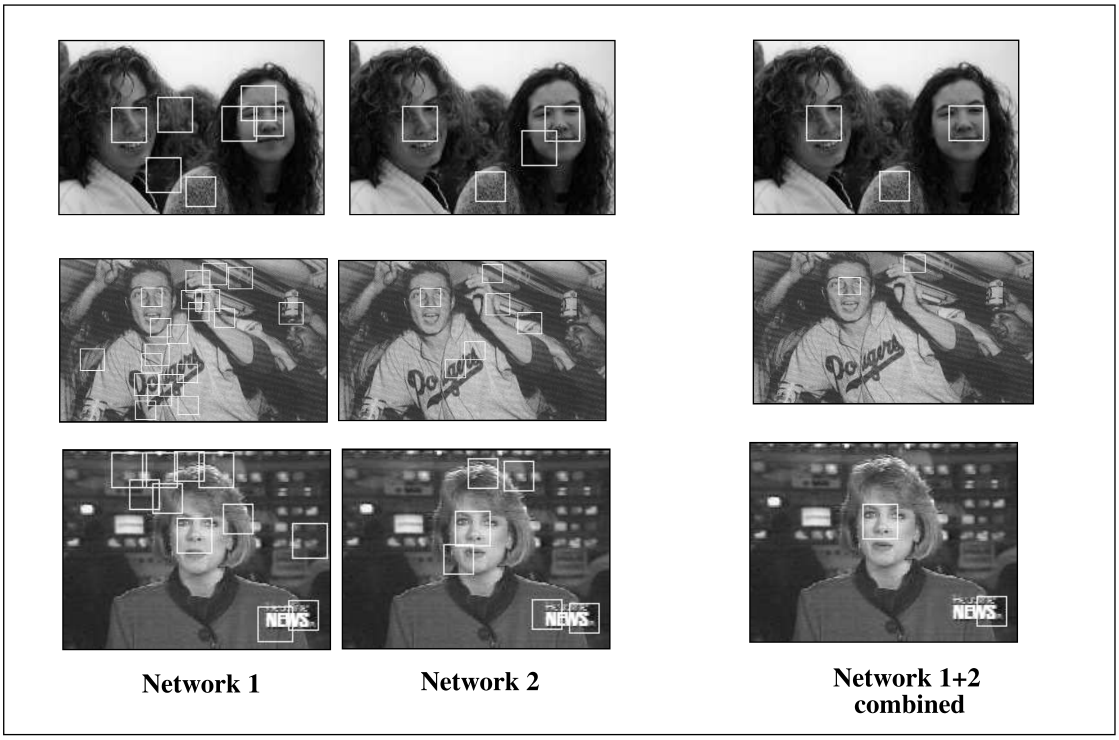


FIGURE 49. CMU Data Base Test Examples (Multiple Network)

Figure 49 shows how false detections can be reduced when combining the output of two neural networks.

A-1-2 Other Data Base

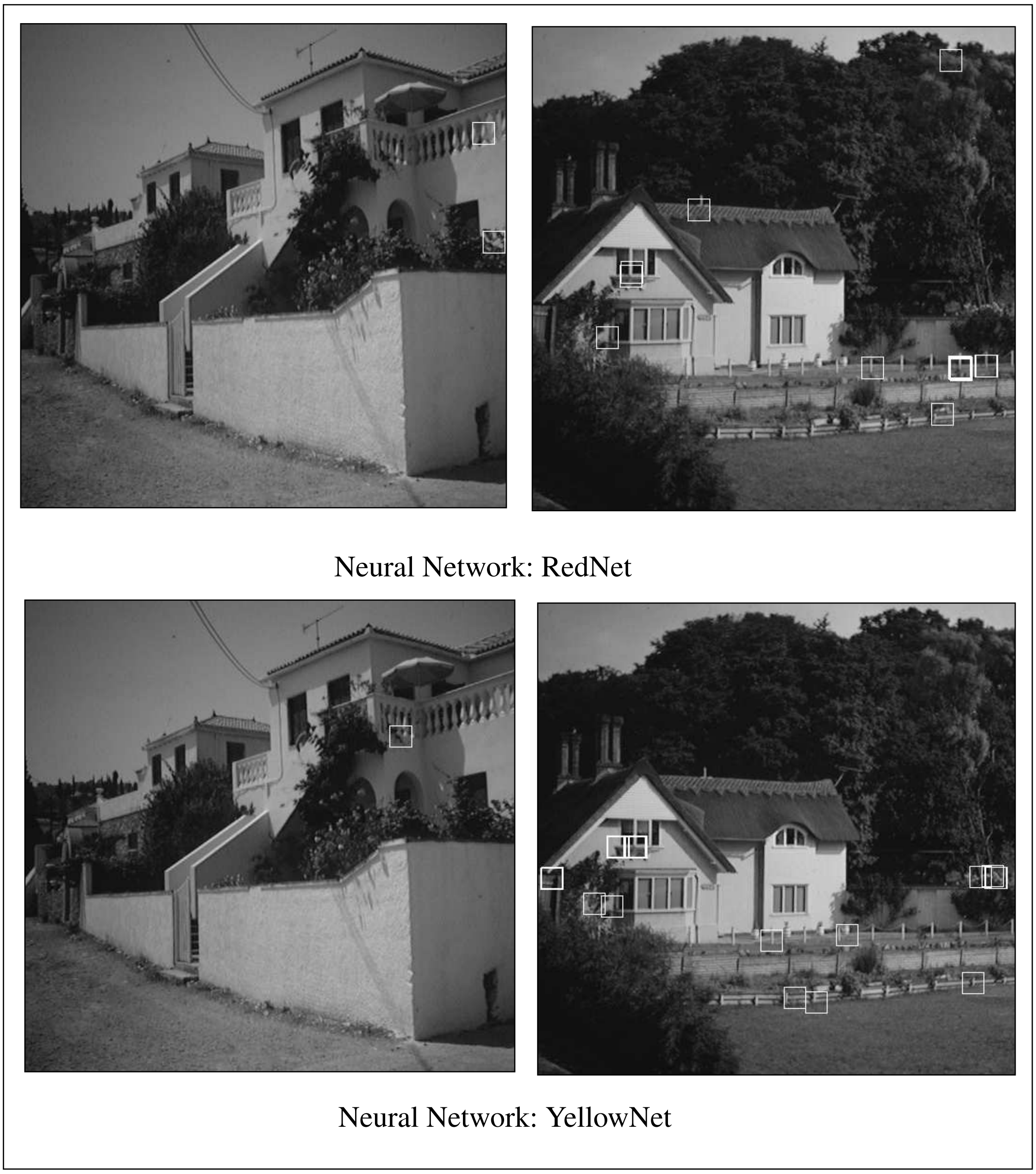


FIGURE 50. Other Data Base Test Examples (Single Network)

Note that false detections occur in general not in the same location when using different networks. This property is used when combining several networks with output arbitration.

A-2 Moving Pictures Series

In the following sub-sections, we present some samples of real-time face detection in an office environment. The frame rate obtained was about 1.5 frames per second. Sequences shown here were recorded in the MPEG format. Only one centered neural network was deployed, the post-processing strategy was of type 3, see Section 6.2.4 on page 38. Further, in Section A-2-4 we performed a multiple resolution scan. Otherwise, just single resolution scans were sufficient to track the faces reasonably.

A-2-1 Different Illuminations and Background

Figure 51 demonstrates the robustness of our Moving Picture Detector in various lighting conditions. A person is walking through an office, followed by a camera. The background changes considerably, but there are virtually no false detections.

As you may notice, the lighting conditions change from dark to light. Face detection is still assured, due to zero-mean centring of the input frames in the preprocessing section of the face detector.

FIGURE 51. **With Different Illuminations and Background (one scale)**

A-2-2 Multiple Faces

Here we have a sample demonstration the face detectors capability of multiple face detection. Since we applied arbitrating strategy three where only highly probable faces get accepted, some may get lost in the detection process. To combat this behaviour, the detection level was set slightly lower to allow more faces being detected. As a drawback we did also encounter more false detections. In addition, all persons are not at the same distance from the camera, this influences also detections values in an unpredictable way, depending on what examples the neural network was trained on and how it adapted to these.

FIGURE 52. Multiple Face Detection (one scale)

A-2-3 Rotation, Scale-Changes (one resolution)

Even though we applied just a one resolution scan of the input frames, the object can vary the distance from the camera and still get detected. That's because the neural network was trained on face image taken at slightly different distances as well, see "Training Database" on page 10. Sideward tilt of the head is possible, but both eyes must be still visible, otherwise the face detector looses the face as can be seen in Figure 53.

FIGURE 53. Sidewards Tilt and Scale Changes (one scale)

Face detection of rotated faces is quite limited, see Figure 54. This was expected, because there are only a few slightly rotated faces in the training database. This could possibly be addressed by the method suggested in Section 10.6 on page 64.

Downwards tilted faces are easier detected than upwards tilted ones, this may be due to the fact that more faces of the former type are in the training base. Further on, un upwards tilted face seems to be smaller, than an upright one, leading to a non-detection with a given sliding-window size having a superior size. Another reason may be that eyes are less shaded when the face looks up, so they are also less dominant. The upper part of the face region is more important than the lower part as has been recalled in [4].

FIGURE 54. Rotation and Upwards/Downwards Tilt (one scale)

A-2-4 Scale Changes (two resolutions)

Figure 55 and 56 show frame scans performed at two resolutions: a face at close distance from the camera gets - upon successful detection - marked with a larger square than when being further away. Note that the face detector works fine also with persons wearing glasses (if the glasses are not too dominating).

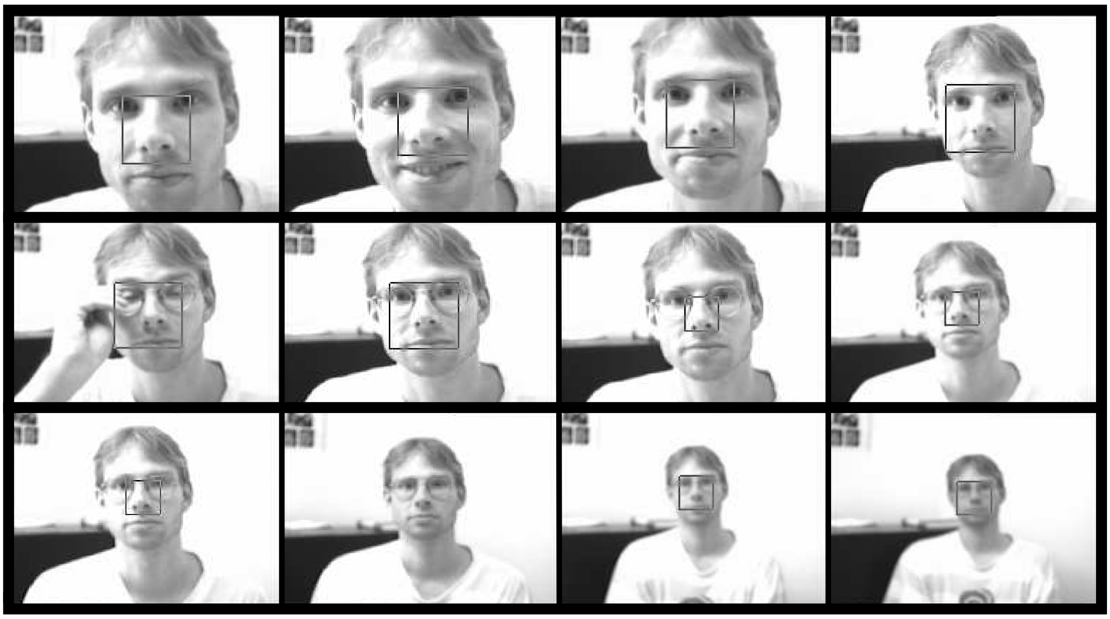


FIGURE 55. Scale Changes (two scales) 1

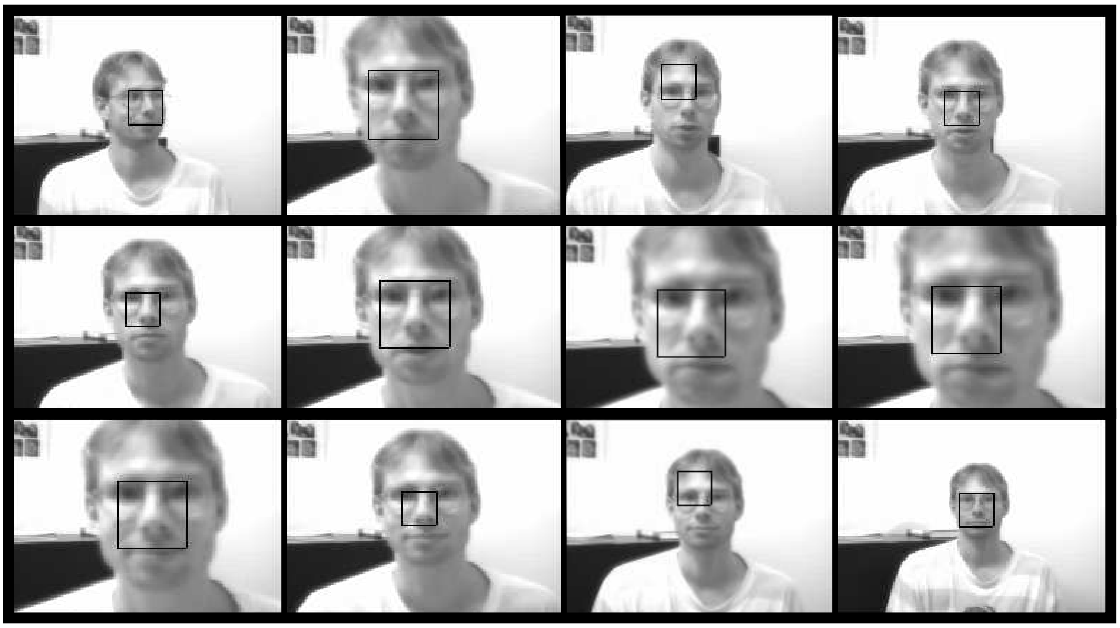


FIGURE 56. Scale Changes (two scales) 2

B Sources

B-1 Hardware

- **Frame Grabber**
  The Sun Video Plus (Osprey-1500) card controls the video camera and provides audio/video processing engines for a broad range of multimedia applications on Sun PCI-based workstations. The Osprey-1500 is a video/audio processor designed as a cornerstone for video communication systems.Through different microcode modules, video standards such as H.261, H.263, CellB, JPEG, MPEG-1 and uncompressed formats can be supported with no change in the hardware.

- **Video Camera**
  Just one relative low cost video camera will be used (no stereo-view of objects).

- **Technical choices**
  - Operating system for the actual setup: Solaris 2.5.6 running on an Ultra-Sparc 30.
  - Programming language: C

B-2 Software

- **FFT Library**
  For this project FFTW (developed at the MIT by Matteo Frigo and Steven G. Johnson.) will be used, which is a C subroutine library for performing the Discrete Fourier Transform (DFT) in one or more dimensions. The benchmarks performed on a variety of platforms, show that FFTW's performance (time needed by the FFT to compute N x N images) is typically superior to that of other publicly available FFT software.

- **MLP**
  Source code of a simple Multi-Layer Perceptron from the Carnegie Mellon University was available, see 12.3. This helped greatly speeding-up the implementation of a running fast neural network that initially performed face recognition and got later converted into one doing face detection.

- **Compiler**
  The **cc** C-compiler delivered the best results on the platform we used. Several optimization flags were activated in order to minimize execution time of the face detectors:
  -DSOLARIS**:** Indicates the operating system we are using, works like the #define preprocessing directive.
  -fast**:** Selects the optimum combination of compilation options for speed, depending on the platform in use. This provides close to the maximum performance for most realistic applications. With the sparc platform, methods such as *native* (generate code targeted for a certain processor and its instruction set) and *dalign* (allows the compiler to generate double-word load/store instructions wherever profitable for improved performance) are being activated automatically.
  -xO5**:** Object code optimization. This flag generates the highest level of optimization. It uses algorithms that take more compilation time or that don not have as high a certainty of improving execution time.

Acknowledgements

The author likes to thank S. Ben-Yacoub and J. Luettin for their support, guidance and introduction to the fascinating worlds of object detection and neural networks. Thanks go also to J.M. Andersen for the useful discussions on how to train neural networks and S. Marchand-Maillet for his advices and corrections of his reports.

IDIAP

Martigny - Valais - Suisse

Institut Dalle Molle d'Intelligence Artificielle Perceptive

IDIAP is a non-profit research institute which was founded in 1991 on the occasion of the 20th anniversary of the Dalle Molle Foundation. The foundation was originally focusing on studies in algorithmic linguistics and became, in 1985, the Dalle Molle Foundation for the Quality of Life. IDIAP is one of the three semi-public research centers founded by the Dalle Molle Foundation, after ISSCO in Geneva and IDSIA in Lugano, primarily concentrating on automatic translation and fundamental artificial intelligence.

IDIAP is primarily funded by long-term support from the City of Martigny, the Canton of Valais, the Swiss Confederation and Swisscom among others. In addition, IDIAP receives substantial research grants from the Swiss National Science Foundation (SNSF) for national projects and the Swiss Office for Science and Education (SOSE) for European projects. Today there is and average of about 25 scientists in residence at IDIAP including permanent staff, postdoctoral fellows, PhD students and industrial visitors. IDIAP's research activities are of both theoretical and applied nature. Focusing on a few, well defined research axes, IDIAP carries out fundamental research and develops prototype systems to validate its models against the reality of applications. The main research and development activities of IDIAP are centered on the general issues of perception, cognition and pattern analysis (mainly to respond to the present and future needs related to user-machine interaction). In 1996, the activities were split up among three research groups:

- Machine learning, including artificial neural networks, data analysis and data knowledge extraction
- Speech processing, including automatic speech recognition and speaker verification
- Vision, including visual speech/speaker recognition and handwriting recognition

EURECOM

Institute Eurécom was created, in 1992, jointly by Ecole Nationale Supérieure des Télécommunications de Paris (ENST) and the Federal Polytechnic Institute of Lausanne (EPFL) to offer a curriculum in communications system engineering. After an undergraduate education in telecommunications, students from either school and from associated establishments (like Politecnico di Torino) join Eurécom for a three-semester graduate curriculum constituting the final phase of their engineering studies.

This approach is designed to satisfy growing demand from corporate users, manufacturers and operators in the field of telecommunications, for high level generalist engineers able to become the architects of future communications systems, i.e. able not only to master the technologies of these networks, but also to match their services to end-users' needs, taking related socioeconomic effects into account.

Students major in the following fields (Current research activities are indicated likewise):

- **Corporate Communications**
  - High speed ATM networks
  - Network security
  - Simulation and network management
  - Network infrastructure for multimedia applications
- **Multimedia Communications**
  - Multimedia document indexing
  - Image and video for multimedia applications
  - Multimedia collaborative applications
- **Mobile Communications**
  - Digital signal processing
  - Radio system engineering
  - Mobile internetworking

Each option is organized by a Research Unit. The relations and cooperation of the Units with industrial partners insure the high level and practical applicability of the courses.

EPFL
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Swiss Federal Institute of Technology - Lausanne

- Founded in 1853 (Ecole Speciale de Lausanne).
- Became E.P.F.L in 1969 (Swiss Federal Institute of Technology, Lausanne).
- 704 degrees awarded in 1992: 420 Engineers and Architects, 122 PhD, 162 Post-graduate certificates.
- 1000 research and teaching assistants.
- Departments:
  Architecture, chemistry, electrical engineering, civil engineering, rural engineering, computer science, material science, mathematics, mechanical engineering, micro-technics, physics.