When the best windows are identified, another matlab program derived the cluster of the best of them and designs a 10x10 cross highlighting thus the best of the best and proving that altough the net has misclassified left eyes for right eyes and visa versa, it can definitely identify the clustered features with a 95 -99 % success percentage (see figures in Appendix A).

Note that the entire verification procedure was done on faces that the net had not encountered before.

A way to have a more general idea about how the net behaved throughout the whole procedure was to project all the probabilities on the actual image (see figures in Appendix A).

In the regions where the net had identified as nothing the probability distribution was quite smooth. Very small probabilities and zeroes were assigned to those windows that had been identified as nothing, while the windows situated around and on the feature had been assigned quite big probabilites.

In the case of the eigenvectors, the net seemed to be absolutely certain of its decisions since the majority of the assigned probilities was 0.9 and above.

In the case of the phase angle data, although the net had a smaller percentage of erroneous classification, it seemed to be 'less sure'. This means that the probabilites it assigned were not as absolute as in the eigenvectors' case. While in the previously mentioned case the majority of the probabilites was over 0.9, here it was ranging from 0.7 to 0.8.

In the last case, that of the ft reduced frequencies, the net still had successes, but the erroneous classification rate was the biggest. The probabilites in this case were ranging from 0.6 to 1.000. The following graph 5–2 represents schematically the postprocessing methodology.

When the best windows for each picture in each case were identified, another matlab program selected the best of these clusters. When they were identified, a 10x10 cross was plotted at the centre of the elite window these best clusters predefined. In this way, it is proven that although the net has misclassified left for right eyes and visa versa, it is nevertheless, capable of correctly identifying the majority of the preprocessed input patterns. Figure 5–3 shows schematically how the cross was plotted.

More specifically the matlab program that was plotting the cross was doing the following :

- Read in the normalised image
- Load the file containing the probabilities and the top left corner coordinates of the best windows
- Find the minimum and the maximum value coordinate for x and y
- For each facial feature [1] find the biggest probabilities and locate the best cluster
- Locate the centre of this cluster and plot a 10x10 cross centered there
- Show the face with the plotted crosses

[1]The features were coded from 1 to 5, 1 for left eye, 2 for right eye, 3 for the nose, 4 for the mouth and 5 for nothing, a fact that made this operation feasible since the program knew which coordinates correspond to each feature

5.2 Comparison

The results of the three nets are illustrated in the Appendix A that follows at the end of the thesis.The first three sets of four images each were generated by the eigenvector alpha values technique, the next three by the fourier transform and phase radians and rest by the fourier transform reduced frequencies. For each case there are four image sets. The first set shows the best of the best probabilities with the plotted cross upon the normalised face, the second set shows the normalised face with all the crosses plotted combined, the second the thresholded probabilities [2], the third the raw probabilities unthresholded, and finally the forth shows the actual normalised face. This is shown for all three cases, so as to allow for a constructive comparison of the three net results.

From these results, it is quite clear that the nets had a very big success percentage, and that the applied preprocessing techniques were well selected.

[2]The image intensity is modified so as to underline the difference between the normalised face and the best probabilities, it has actually been multiplied with 1.5

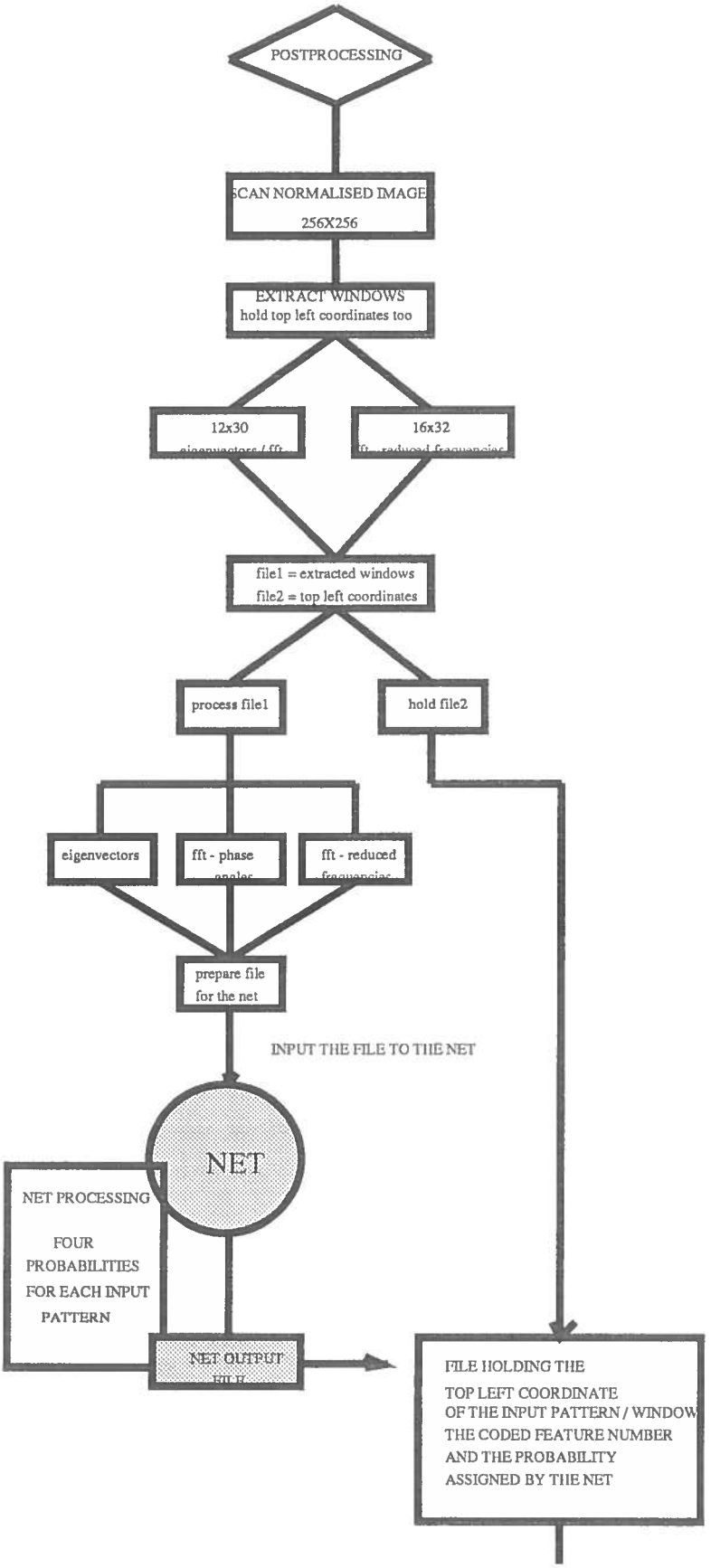

Figure 5–1: Graphical Postprocessing Overview Part-I

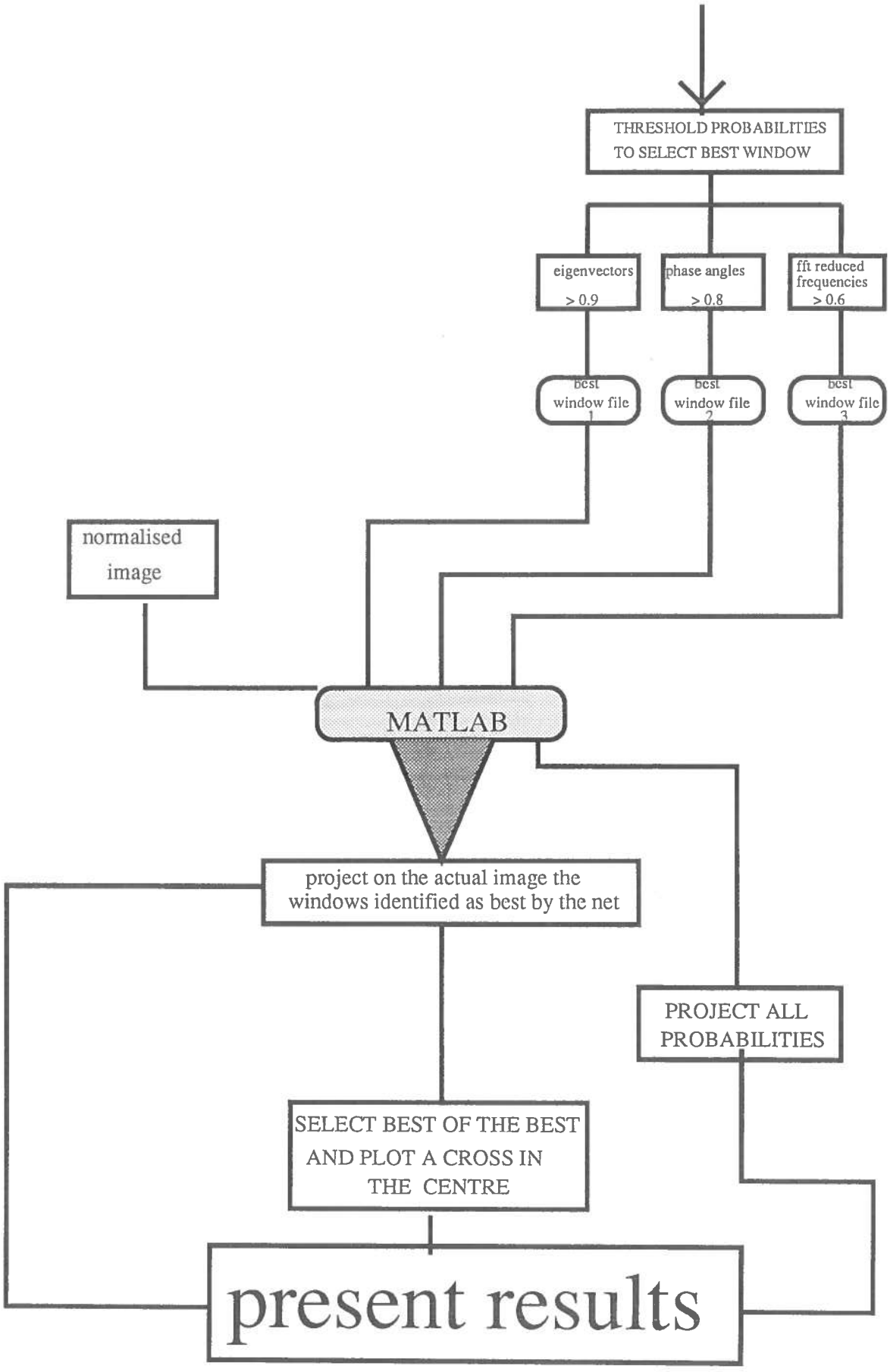

Figure 5–2: Graphical Postprocessing Overview Part-II

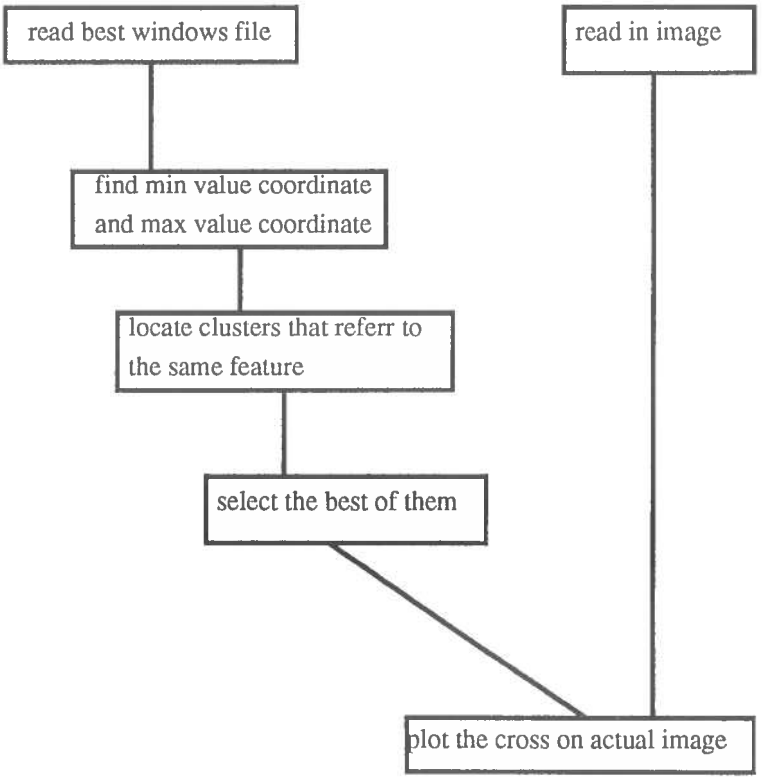

Figure 5–3: How the Cross is plotted

Appendix A

Net Results

In this Appendix the project results are included explicitely for each net configuration.

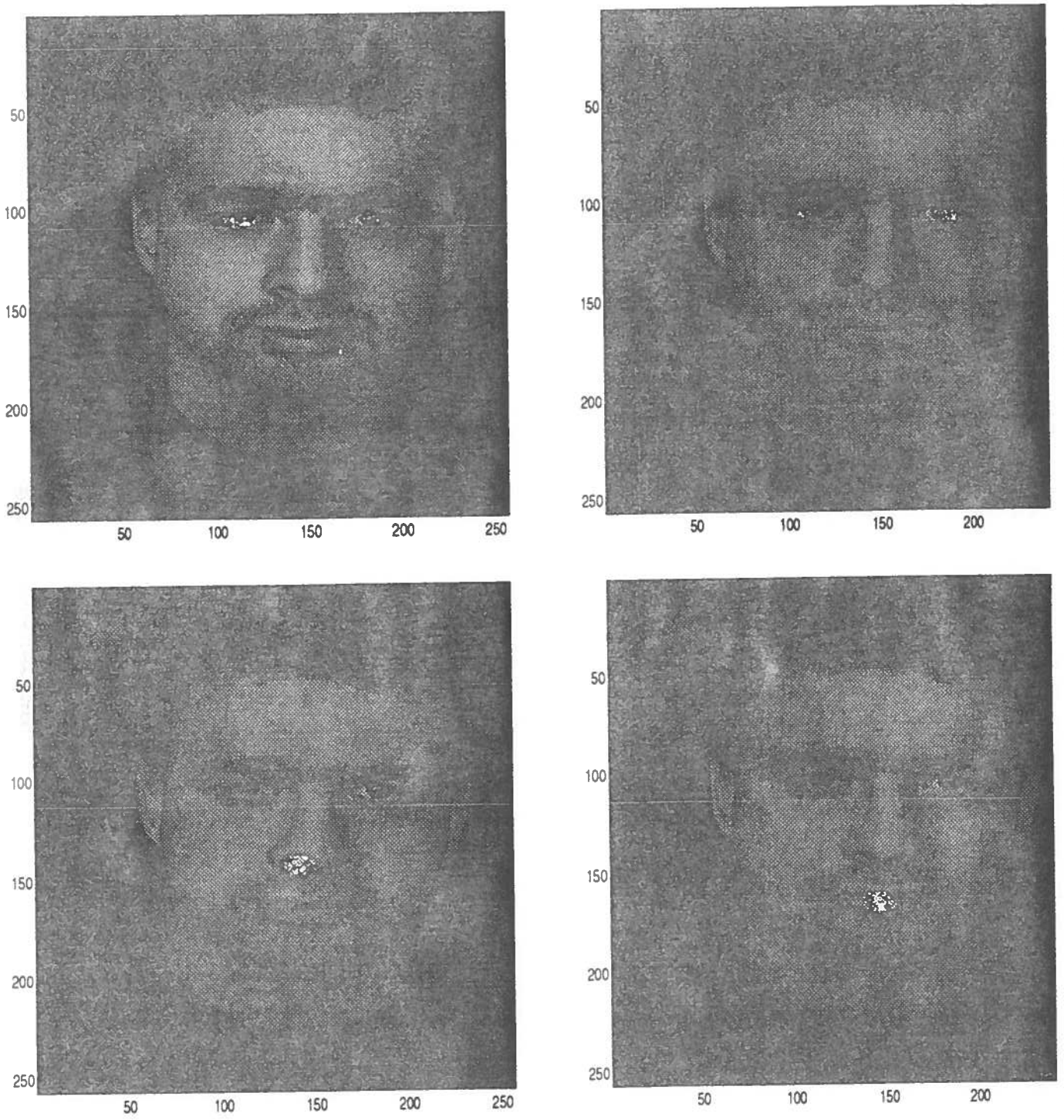

Figure A–1: Best Probabilities by the alpha net

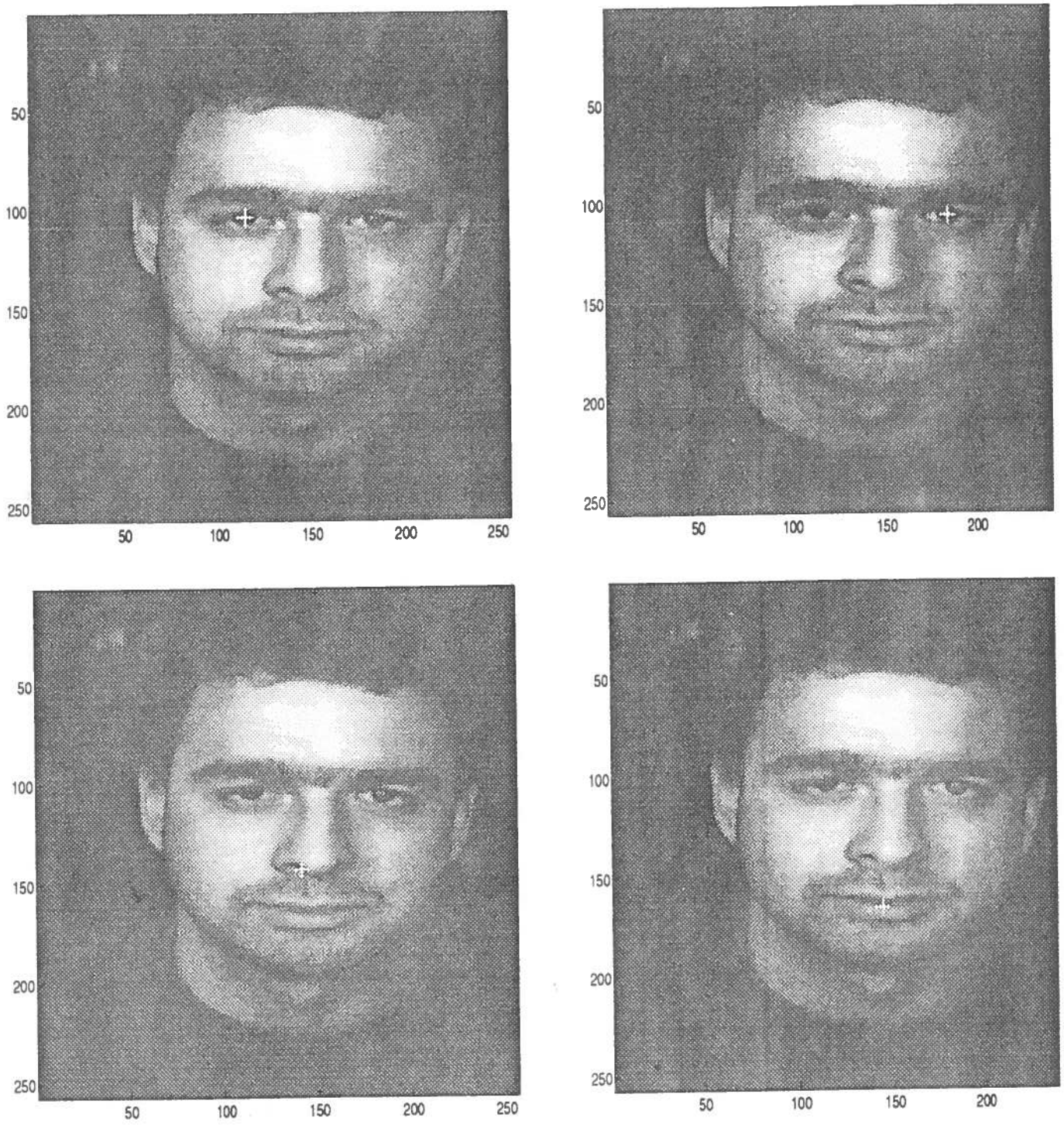

Figure A–2: Best of the Best by the alpha net

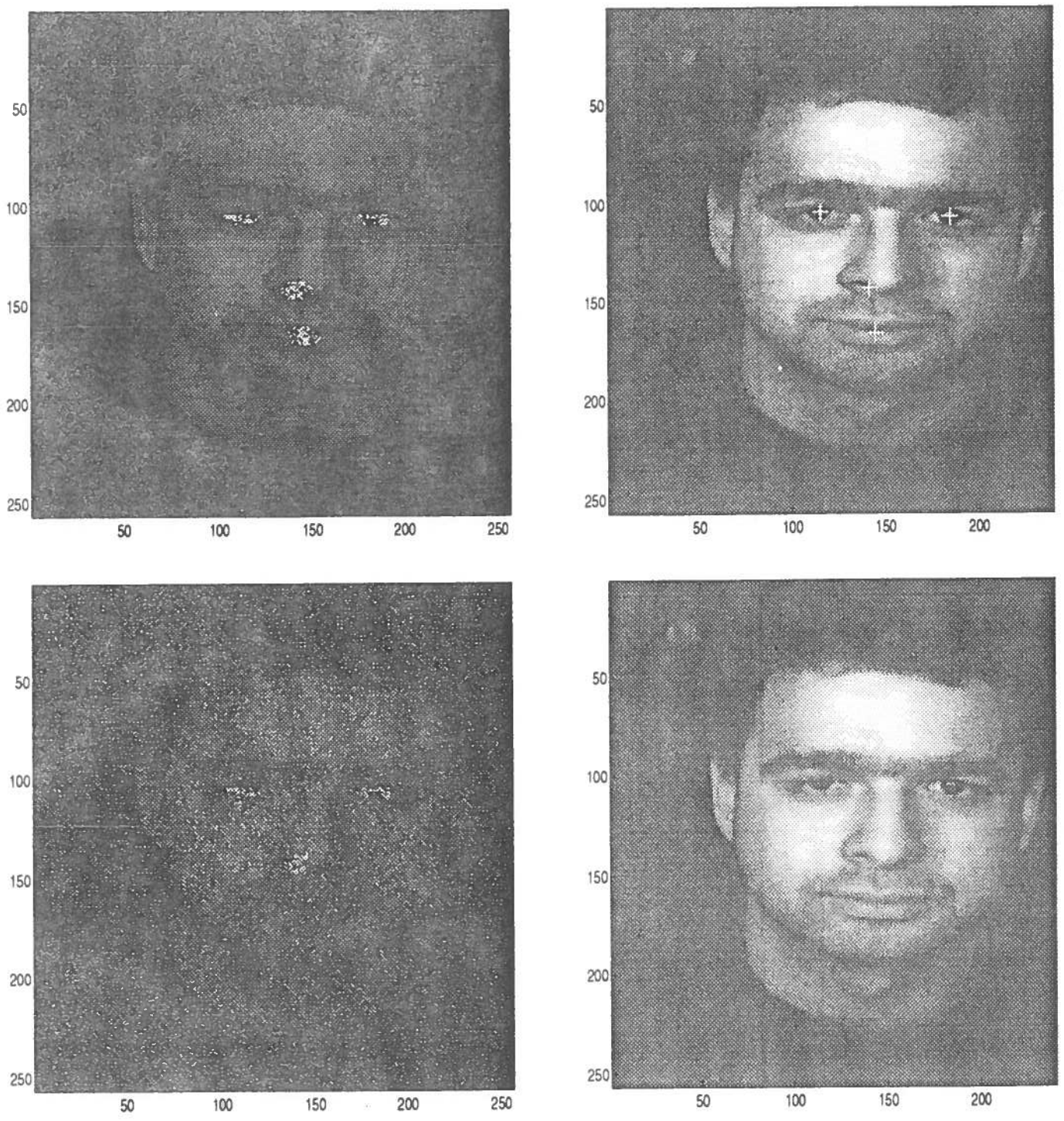

Figure A–3: Best Probabilities, Best of the Best, Raw Probabilities, Normalised Image by the alpha net

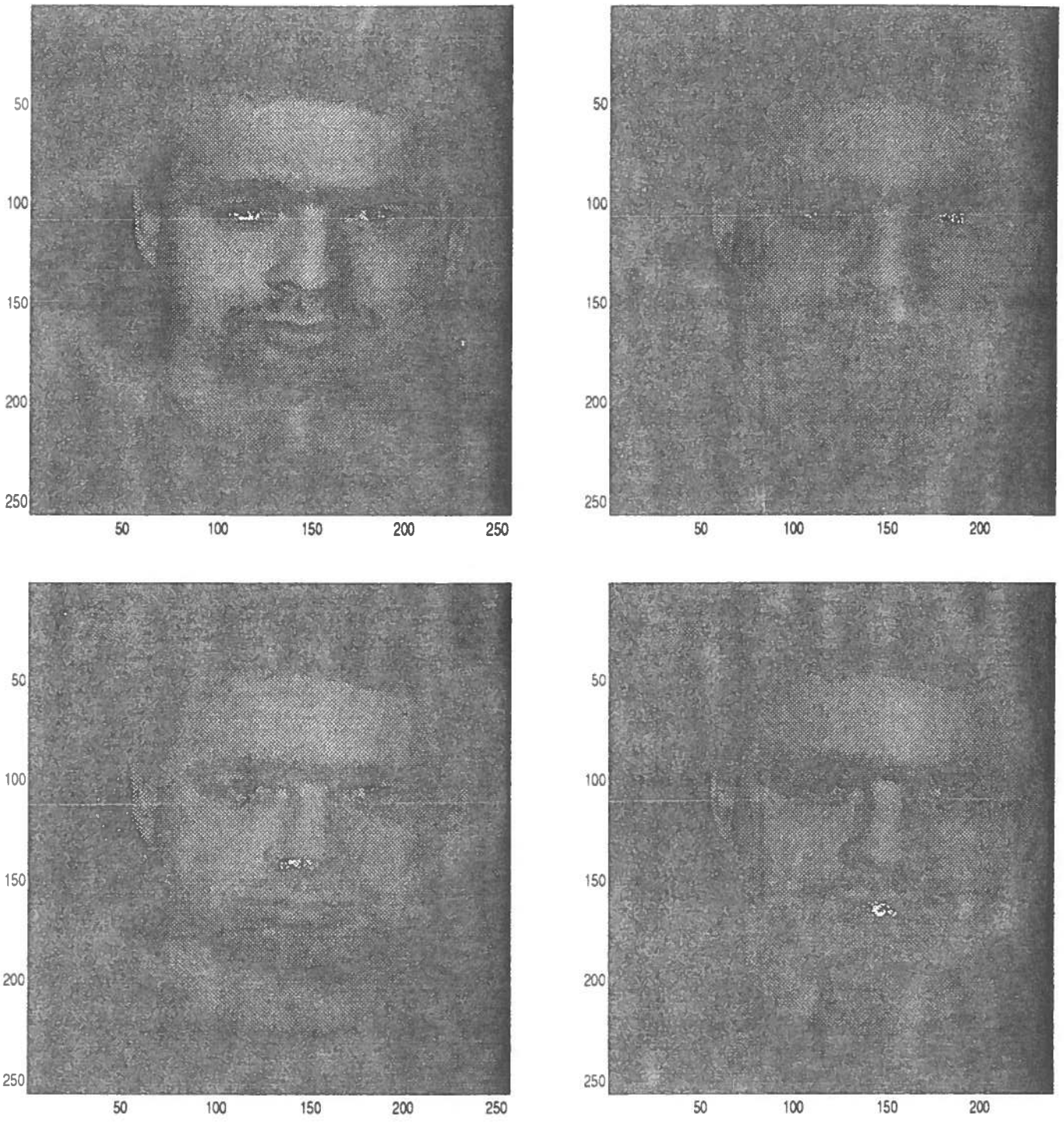

Figure A–4: Best Probabilities by the ft phase angles net

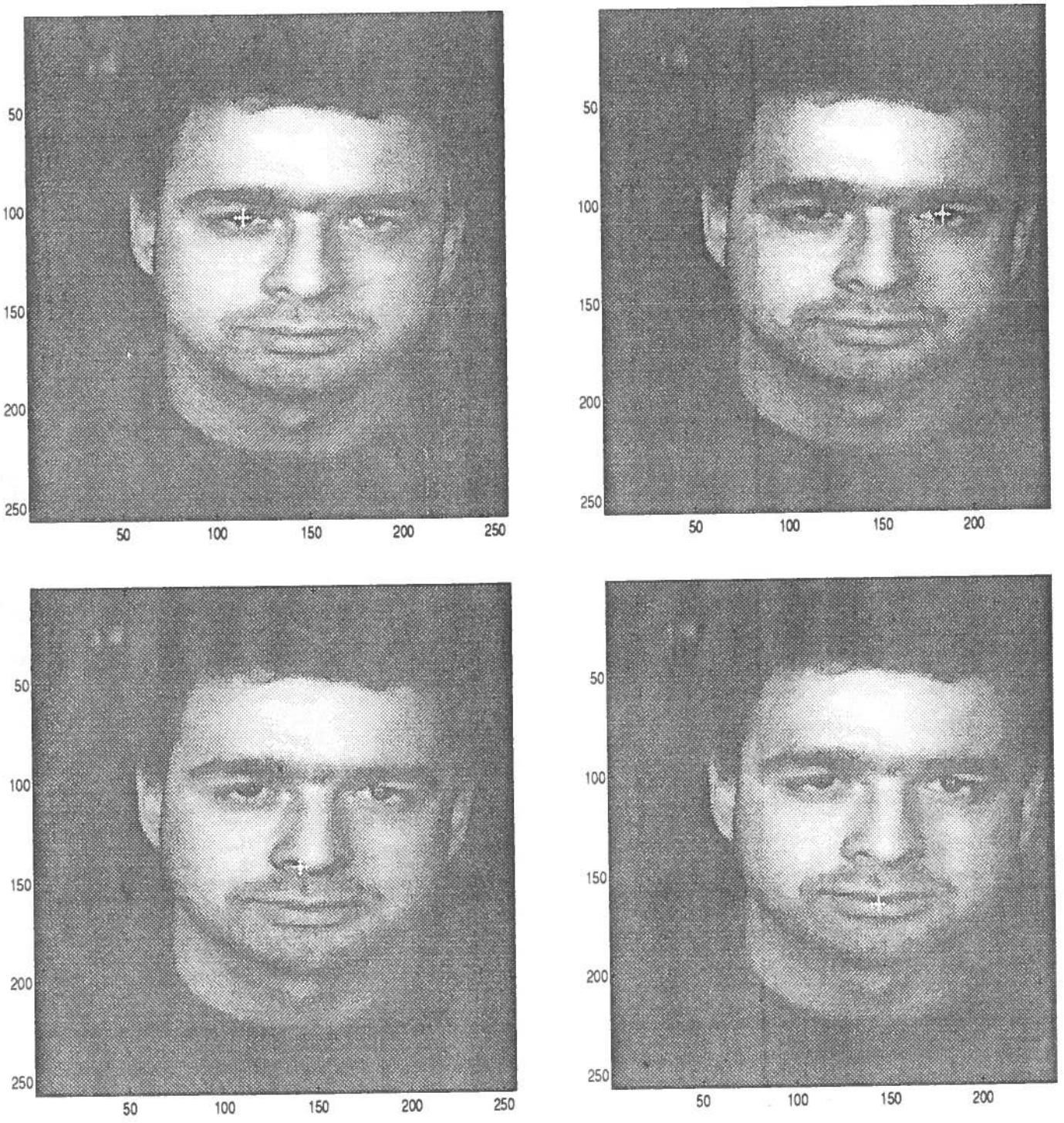

Figure A–5: Best of the Best by the ft phase angles net

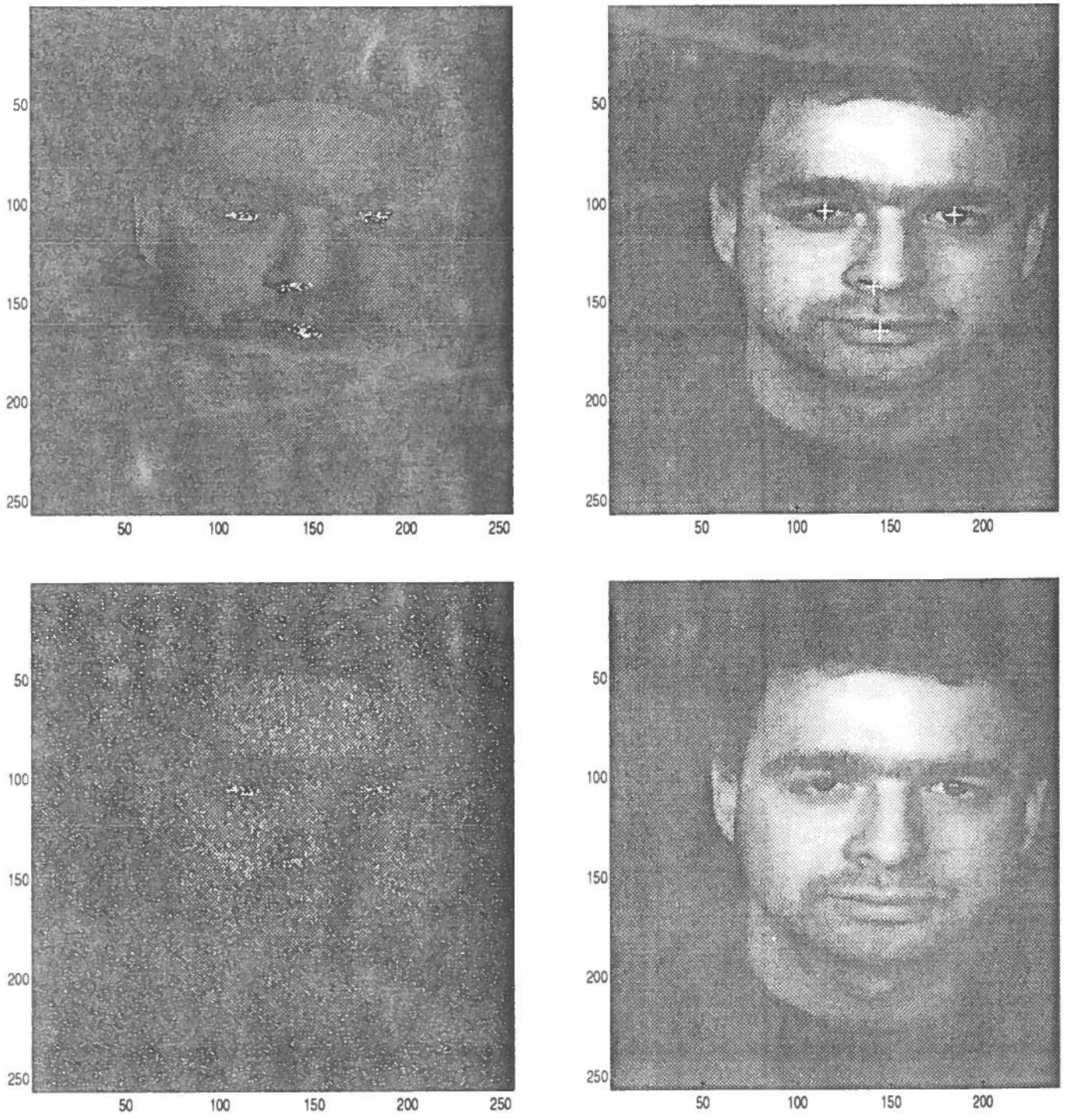

Figure A–6: Best Probabilities, Best of the Best, Raw Probabilities, Normalised Image by the ft phase angles method

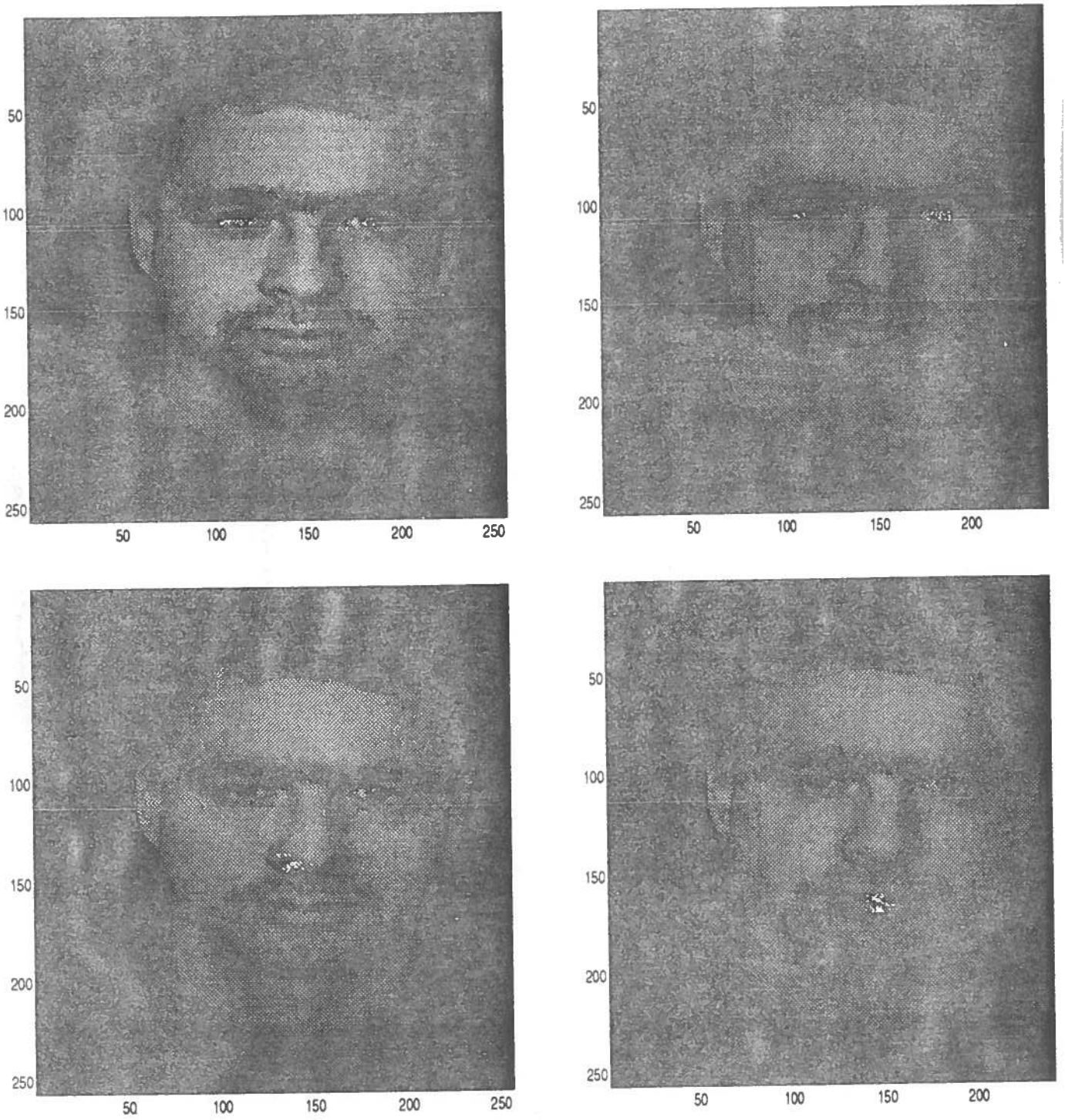

Figure A–7: Best probabilities by the ft reduced frequencies net

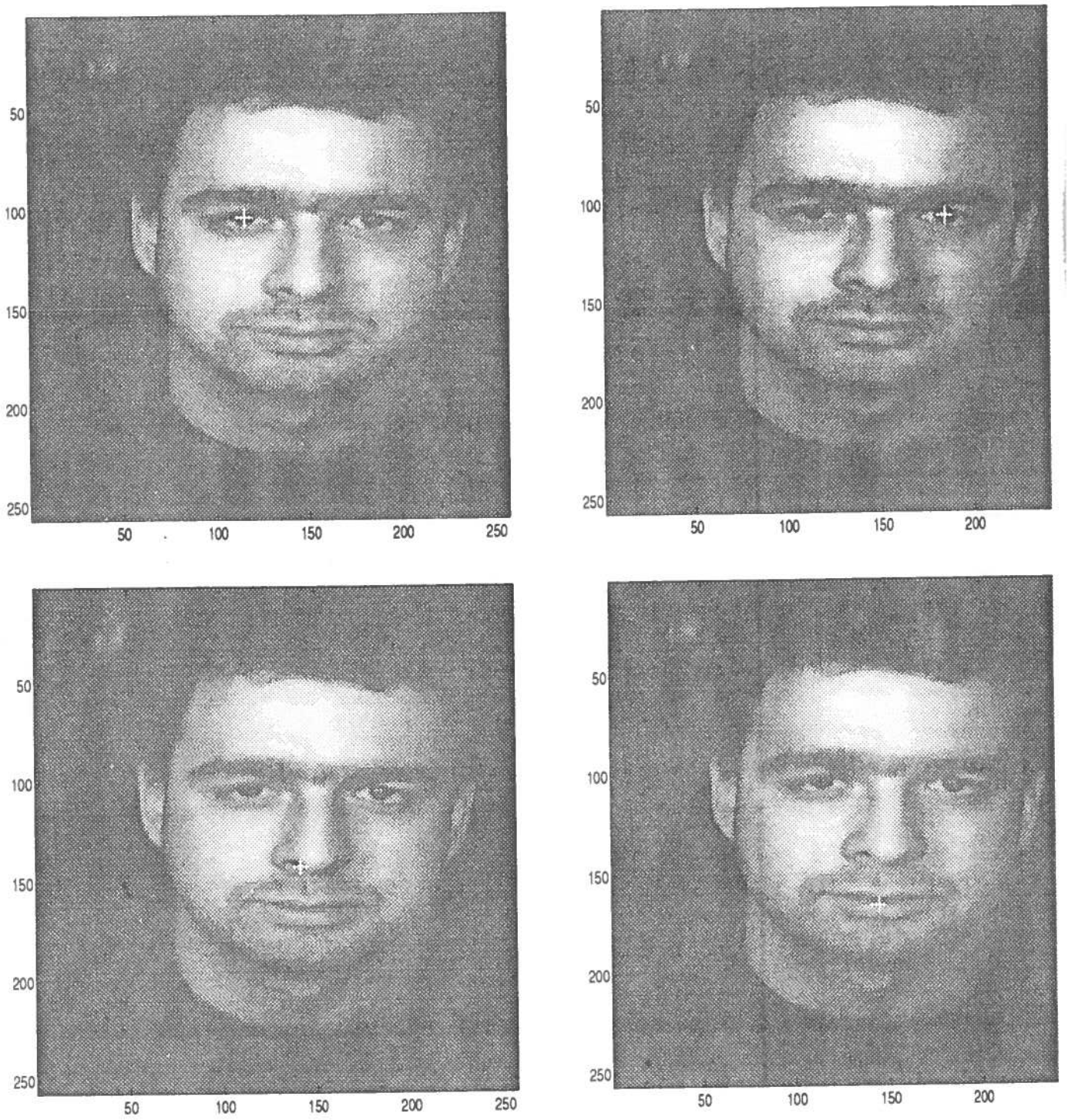

Figure A–8: Best of the Best by the ft reduced frequencies net

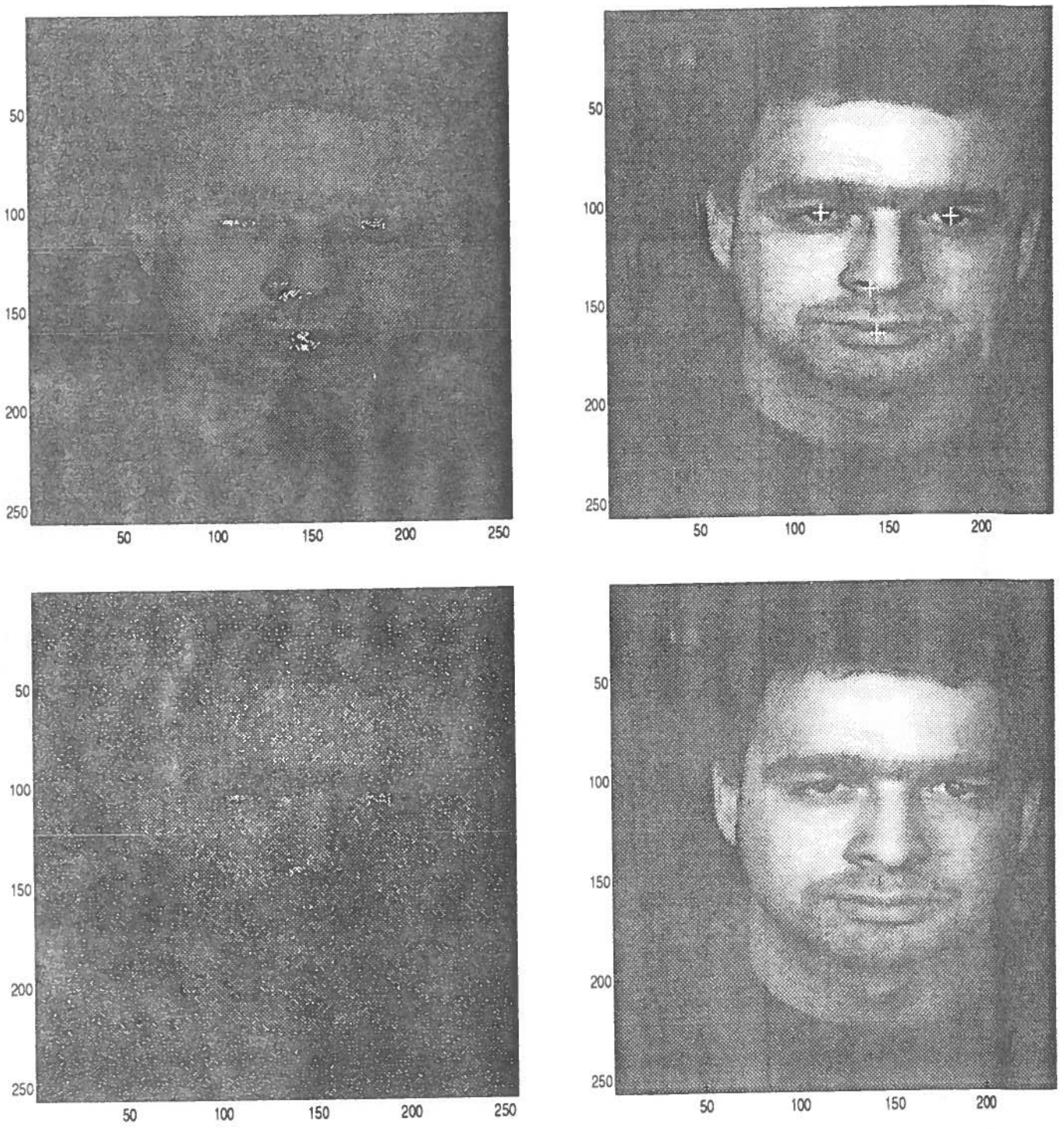

Figure A–9: Best Probabilities, Best of the Best, Raw Probabilities, Normalised Image by the ft reduced frequencies net

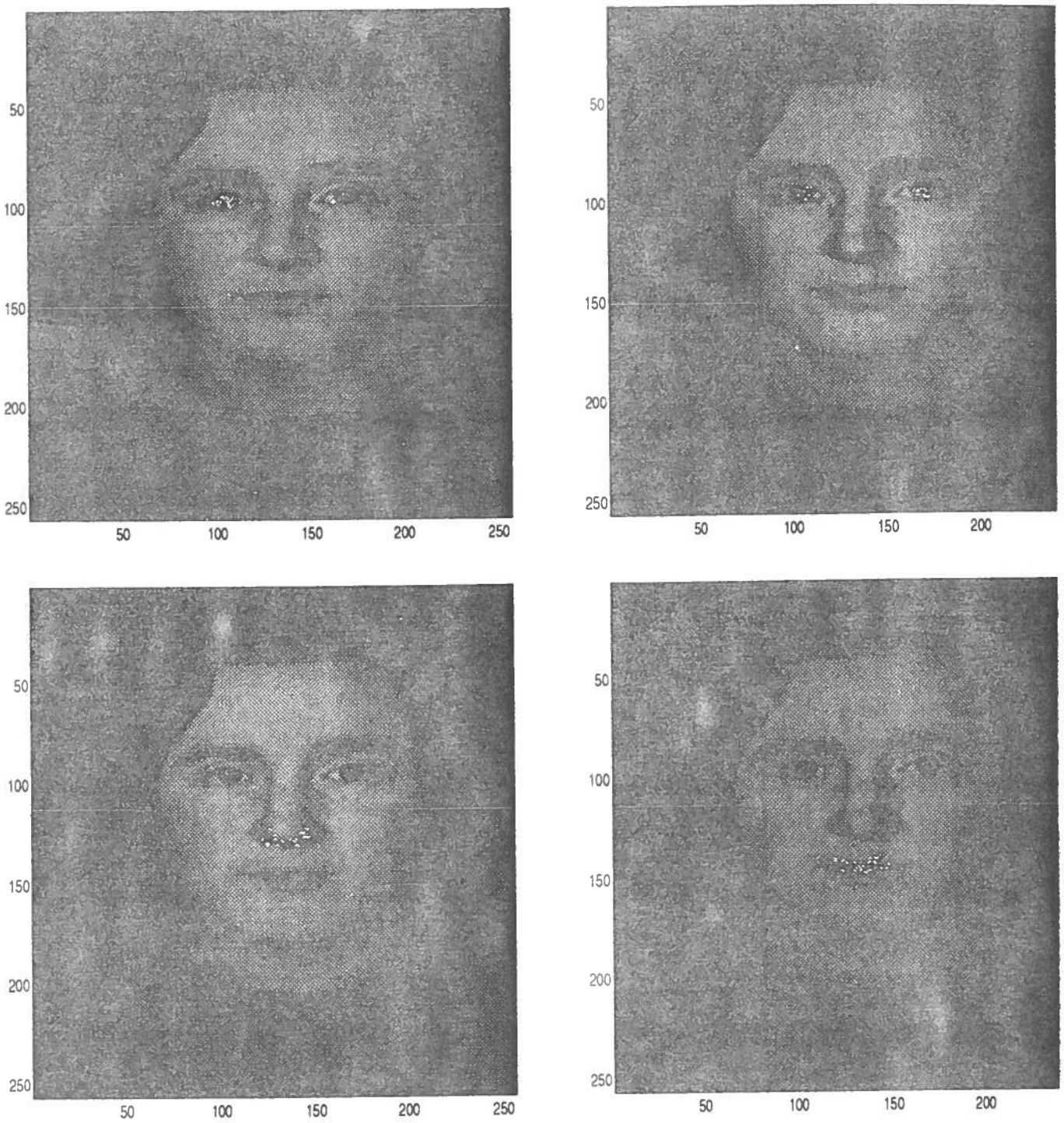

Figure A–10: Best Probabilities by the alpha net

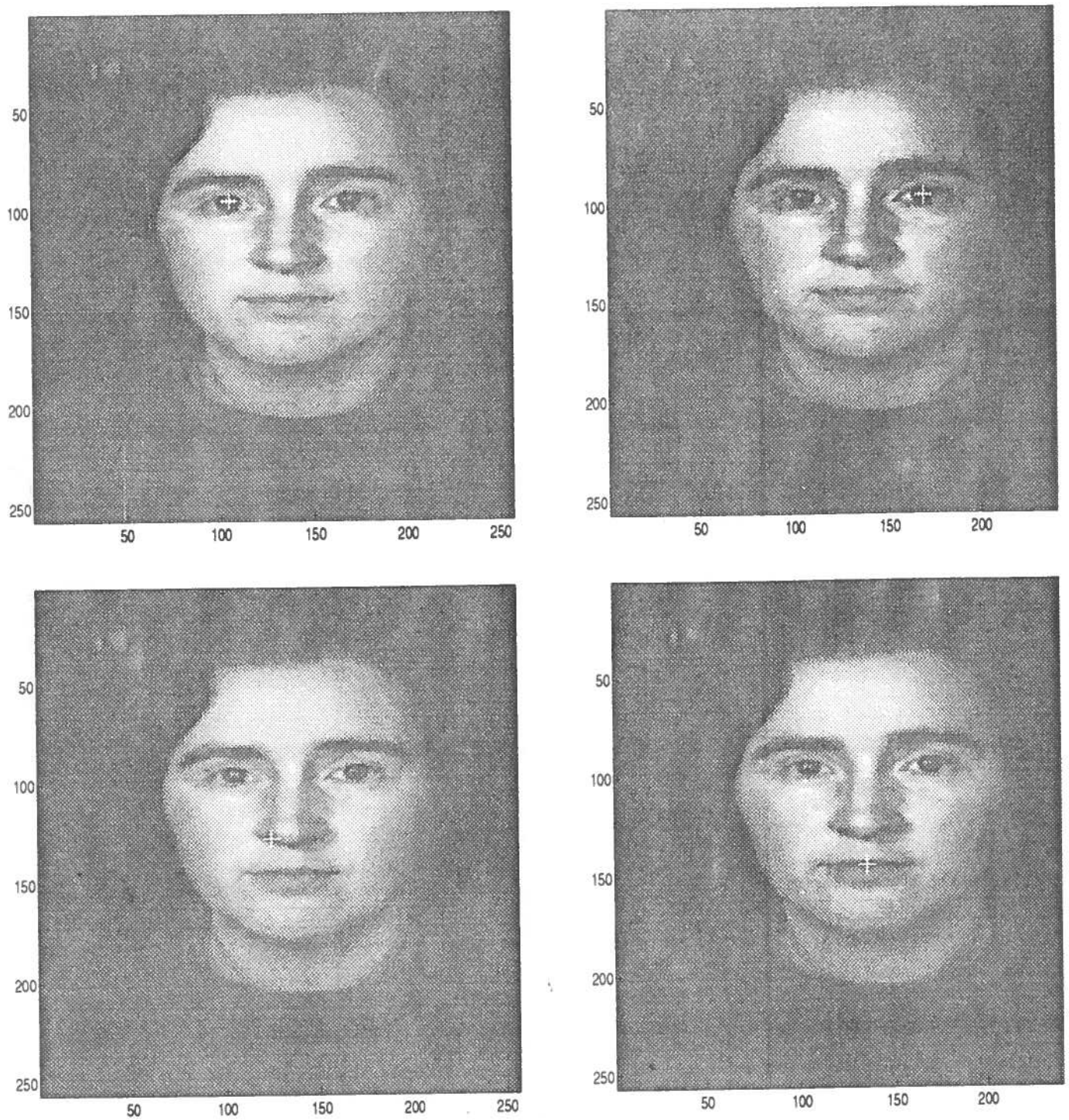

Figure A–11: Best of the Best by the alpha net

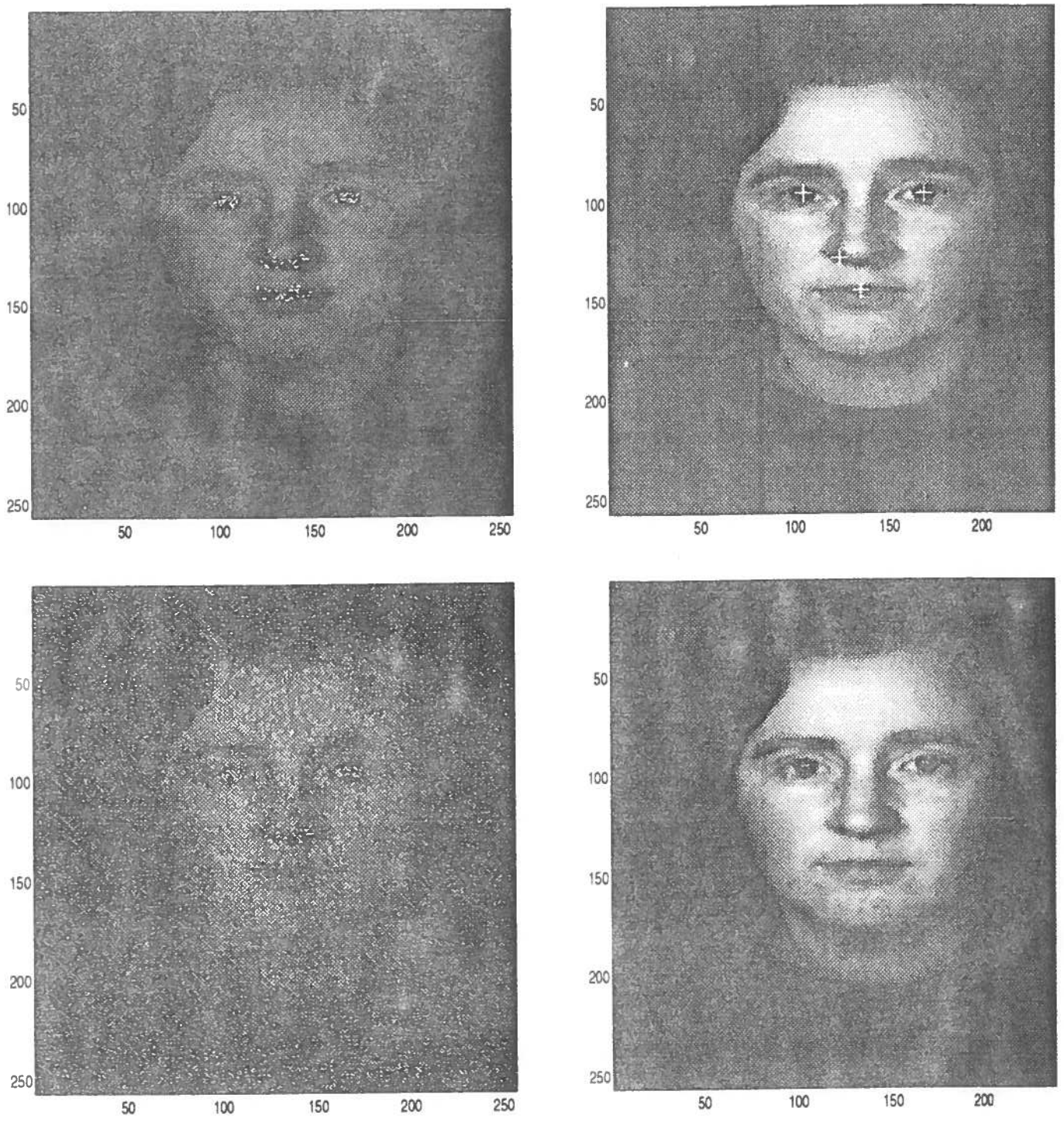

Figure A–12: Best Probabilities, Best of the Best, Raw Probabilities, Normalised Image by the alpha net

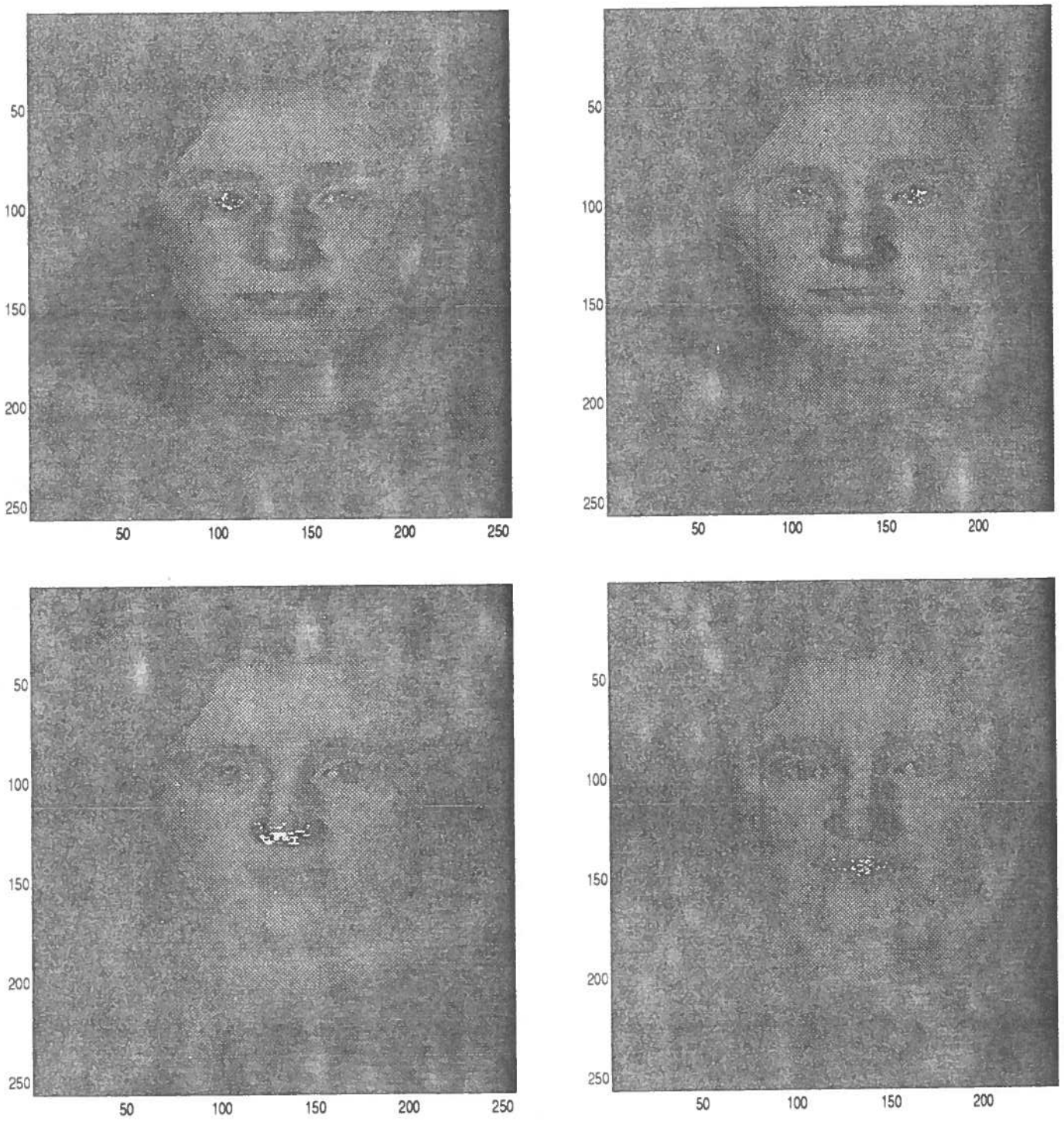

Figure A–13: Best Probabilities by the ft phase angles net

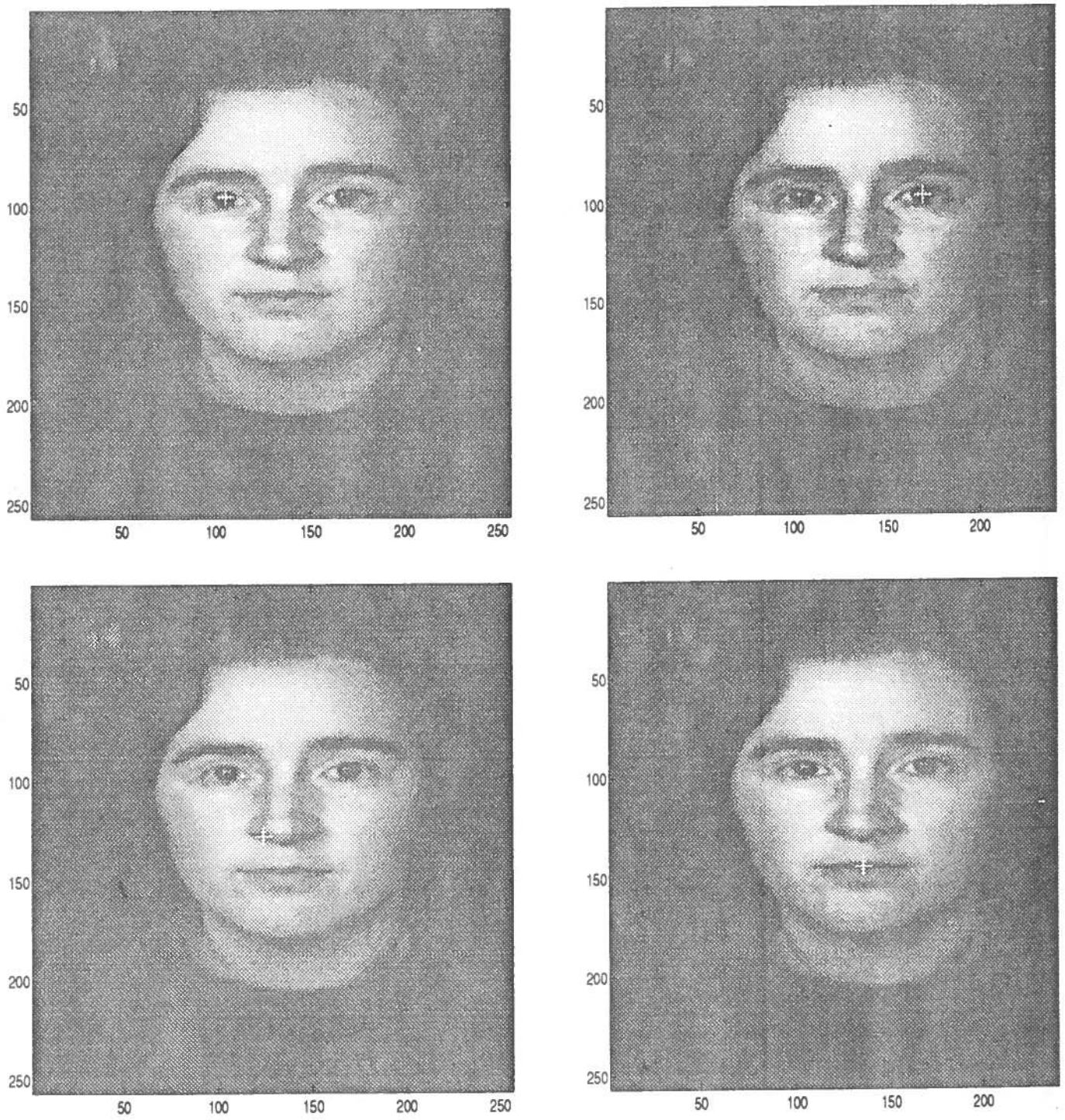

Figure A–14: Best of the Best by the ft phase angles net

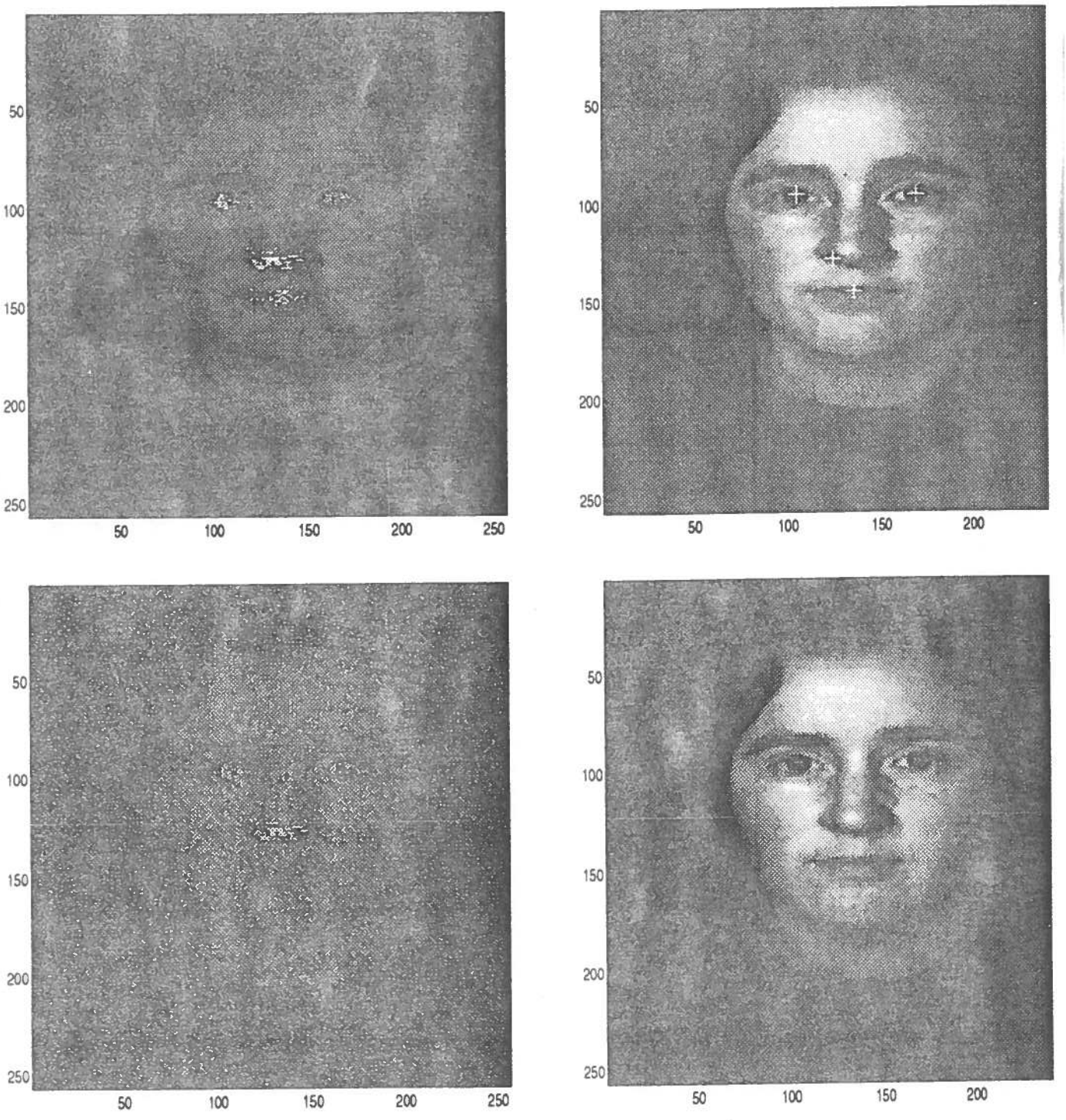

Figure A–15: Best Probabilities, Best of the Best, Raw Probabilities, Normalised Image by the ft phase angles method

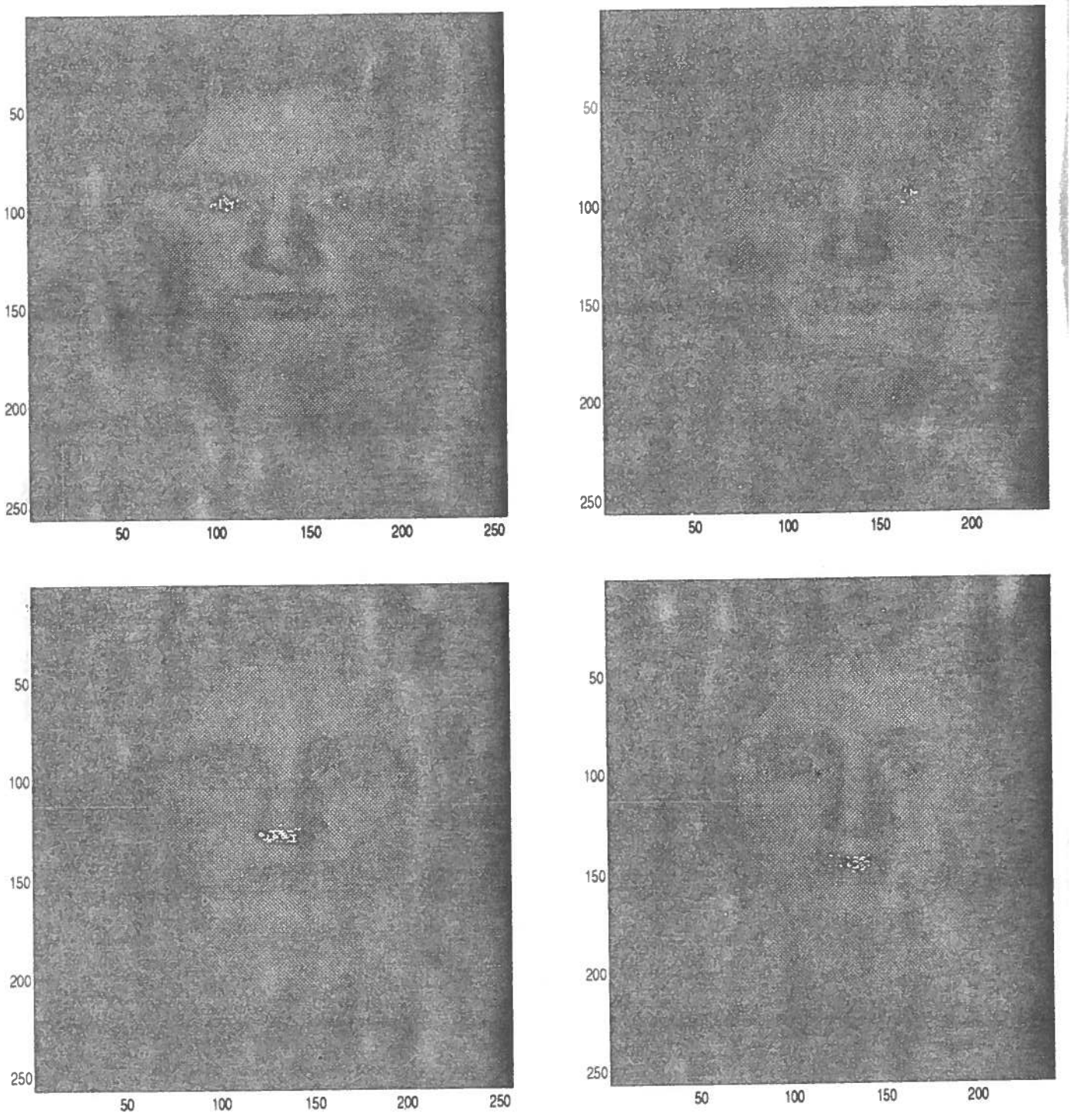

Figure A–16: Best probabilities by the ft reduced frequencies net

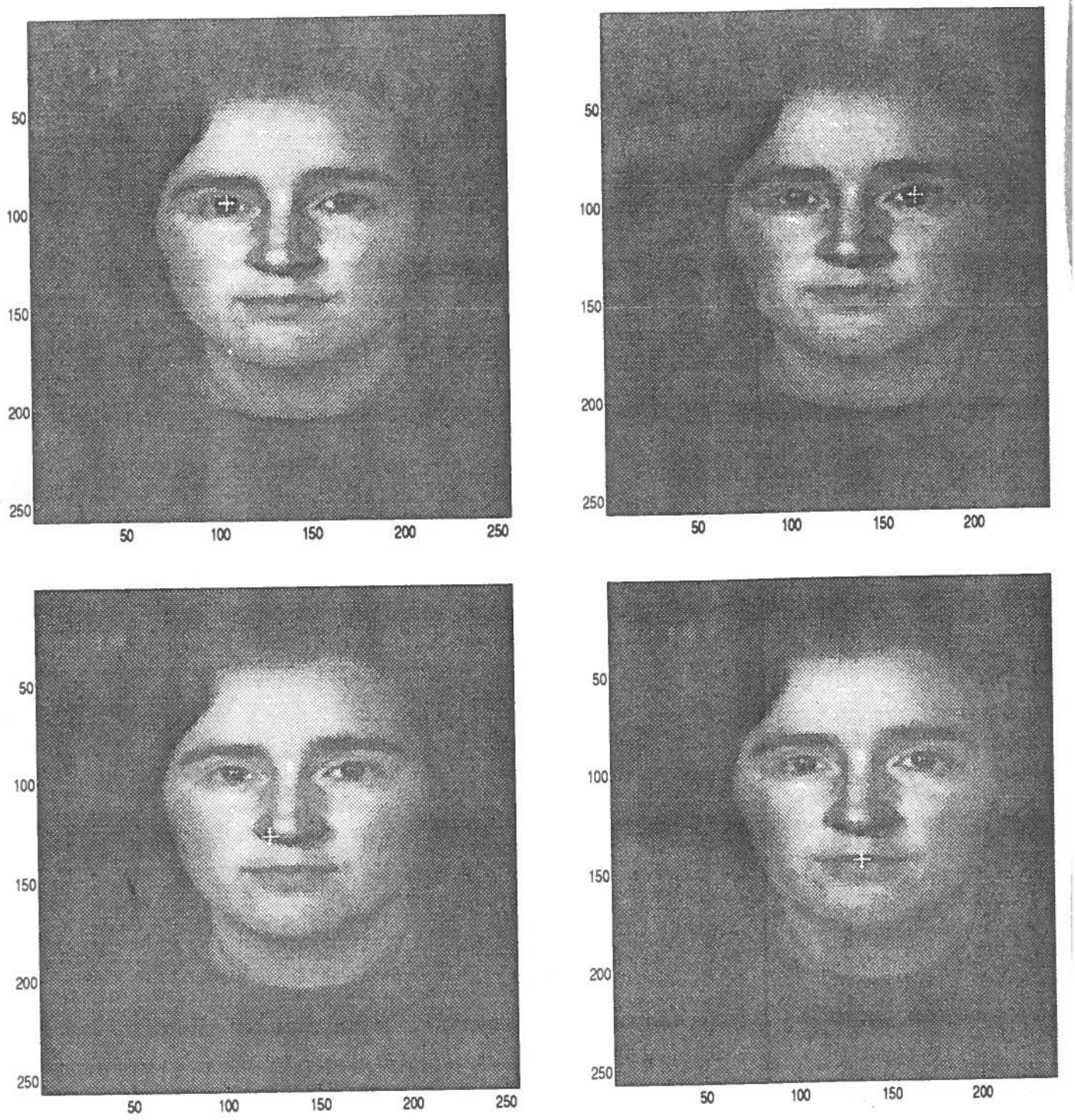

Figure A–17: Best of the Best by the ft reduced frequencies net

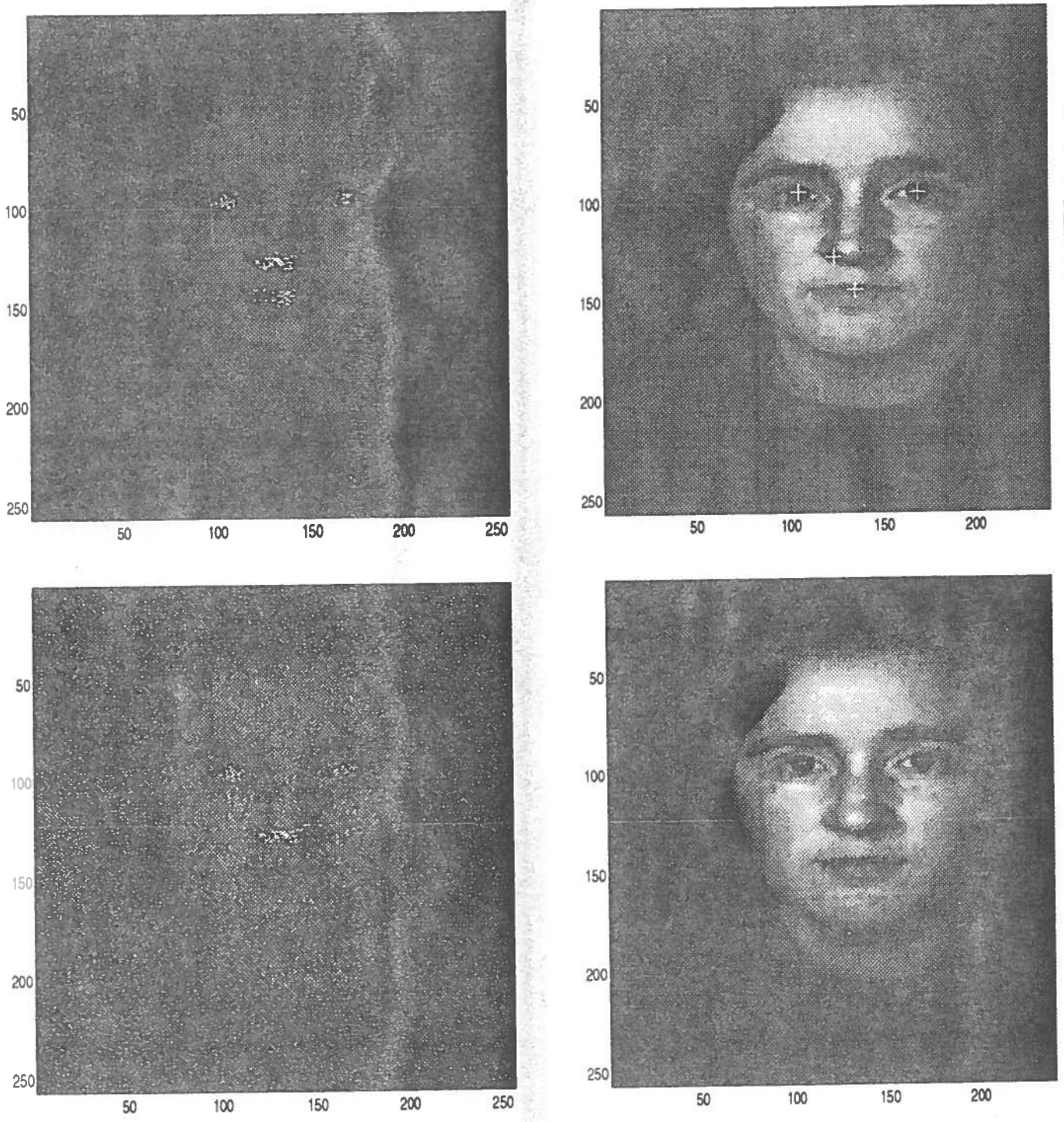

Figure A–18: Best Probabilities, Best of the Best, Raw Probabilities, Normalised Image by the ft reduced frequencies net

Appendix B

Matlab Programs

In this Appendix all the Matlab programs used for the preprocessing, the post-processing and the verification are included.

This program is doing the eigenvectors alpha values analysis and is generating the data file for the net training.

```
load -ascii faces;
m=223; % 223 samples
A=faces';
S=A/1000;
B=sum(S')/m;              % calc. average feature
NUMEIG=60;

C=zeros(360,m);
for i=1:1:m,
C(:,i)=S(:,i)-B';         % subtract average from features
end;

L=C'*C;
[V,D]=eig(L);             % eigen vectors

M=C*V;
for i=1:1:m T(:,i)=norm(M(:,i),2); end;
```

```
for i=1:1:m XR(:,i)=M(:,i)./T(:,i); end;

for i=1:1:NUMEIG,
I(:,i)=XR(:,224-i); %eigenvector images
end;
colormap(gray(255))
imagesc(reshape(A(:,1),12,30))
figure(2)
colormap(gray(255))
imagesc(reshape(I(:,1),12,30))

count=0;

for n=1:1:NUMEIG
for i=1:1:m
product=sum(I(:,n).*C(:,i));
count=count+1;

K(n,i)=product;
end;
end;

% reconstruct the images

for j=1:1:m
suma=B';
for n=1:1:NUMEIG
suma=suma+K(n,j).*I(:,n);
end;
R(:,j)=suma;
end;
figure(3)
colormap(gray(255))
imagesc(reshape(R(:,1),12,30))
```

```
for opa=1:1:m
RE(:,opa)=abs(imagesc(reshape(A(:,opa),12,30)-reshape(R(:,opa),12,30)
end; % find the error for  all the projections


colormap(gray)
figure(5)
plot(sort(abs(RE(:))))


metritis=0;

fid = fopen('alphaface.txt','w');
fprintf(fid,'* faces\n');

for i=1:1:m
for n=1:1:NUMEIG
fprintf(fid,'%8.4f',K(n,i));
metritis=metritis+1;
if metritis==20
fprintf(fid,'\n')
metritis=0
end;
while n==NUMEIG
if i <= 48
fprintf(fid,' 1 0 0 0\n')
break
elseif i <= 90
fprintf(fid,' 0 1 0 0\n')
break
elseif i <= 136
fprintf(fid,' 0 0 1 0\n')
```

```
break
elseif i <= 176
fprintf(fid,' 0 0 0 1\n')
break
elseif i <= 223
fprintf(fid,' 0 0 0 0\n')
break
end;
end;
end;
end;
end;
end;
end;
fclose(fid);
```

This program is doing the fast fourrier transform phase angles preprocessing and generates the data file for the net training.

```
load -ascii pros % 360 net inputs 12x30
X=pros';
[a,m]=size(pros)

for lo=1:1:m
K(:,lo)=fft2(X(:,lo));
end;
for oo=1:1:m
RR(:,oo)=angle(K(:,oo)); % prepare the phase angle matrix
end;


metritis=0;
```

```
fid = fopen('prosfft','w');

for k=1:1:m
for n=1:1:360
if metritis==20
fprintf(fid,'\n');
metritis=0;
end;
fprintf(fid,'%8.4f ',RR(n,k));
metritis=metritis+1;
if metritis==20
fprintf(fid,'\n');
metritis=0;
end;

while n==360
if i <= 48
fprintf(fid,' 1 0 0 0\n')
break
elseif i <= 90
fprintf(fid,' 0 1 0 0\n')
break
elseif i <= 136
fprintf(fid,' 0 0 1 0\n')
break
elseif i <= 176
fprintf(fid,' 0 0 0 1\n')
break
elseif i <= 223
fprintf(fid,' 0 0 0 0\n')
break
end;
end;
```

```
end;
end;
end;
end;
fclose(fid);
```

This program is doing the fast fourrier transform reduced frequencies analysis and generates the data file for the fft-II reduced frequencies net.

```
load -ascii train
X=train';   % images are 16*32
[a,m]=size(train)

for lo=1:1:m
K(:,lo)=fft2(X(:,lo));
end;        % fft analysis of the data


colormap(gray(255))
imagesc(reshape(K(:,1),16,32))
title('fft of a left eye')
print -dps lefft

figure(2)
colormap(gray(255))
imagesc(reshape(K(:,1),16,32))
title('Inverse fft of a left eye')

figure(3)
colormap(gray(255))
plot(reshape(K(:,1),16,32))
title('fft plot of a left eye')
print -dps fftleplot
```

```
colormap(gray(255))
im=reshape(X(:,1),16,32);
figure(5);imagesc(im);
FF=fft2(im);
colormap([ .2 .5 0 ; gray(254) ])
figure(4);

am=(abs((FF))>500).*FF; % threshold the frequencies that are > 500
imagesc(abs(log(am+1)))
figure(6);imagesc(ifft2((am)))


figure(7);
hist(abs(FF(:)))
figure(8);plot(sort(abs(FF(:))))
figure(9);plot(sort(log(1+abs(FF(:))))) % indicate why selecting 500

as=abs((K(:,1))) > 500; % select a mask

% save the mask
fid=fopen('sotiras','w');
for ii=1:1:512
fprintf(fid,'%d ',as(ii,1));
end;
fclose(fid);

for rr=1:1:m
EE(:,rr)=as.*K(:,rr);
end;              % put in a matrix all the selected frequencies

figure(10)
title('Inverse left eye fft with selected frequencies')
```

```
imagesc(reshape(ifft2(EE(:,1)),16,32))


WW=real(EE)/10000;
JJ=imag(EE)/10000;   % so as to have proper floating point values

fid=fopen('datakia','w');
for dd=1:1:m
fprintf(fid,'%8.4f ',WW(:,dd));
fprintf(fid,'\n');
fprintf(fid,'%8.4f ',JJ(:,dd));
fprintf(fid,'\n');
end;
fclose(fid);

load -ascii datakia;

for dd=1:1:512
if nnz(datakia(:,dd))==0
datakia(:,[dd])=[];
end;
end;

filo=datakia';
kolon=430;
metritis=0;count=0;
fid=fopen('data','w');

for dd=1:1:kolon
for nn=1:1:57
fprintf(fid,'%8.4f ',filo(nn,dd));
metritis=metritis+1;
count=count+1;
if metritis==19
```

```
fprintf(fid,'\n');
metritis=0;
end;
while count==114
if dd <= 96
fprintf(fid,' 1 0 0 0\n');
count=0;
break
elseif dd <=146
fprintf(fid,' 0 1 0 0\n');
count=0;
break
elseif dd <= 238
fprintf(fid,' 0 0 1 0\n');
count=0;
break
elseif dd <= 330
fprintf(fid,' 0 0 0 1\n');
count=0;
break
elseif dd <= 430
fprintf(fid,' 0 0 0 0\n');
count=0;
break
end;
end;
end;
end;
end;
end;
end;
fclose(fid);
```

This program is doing the postprocessing for the alpha net

```
function facet(SW)
[u,m]=size(SW);
A=SW;
S=A/1000;
B=sum(S')/m;            % calc. average feature
NUMEIG=60;

C=zeros(360,m);
for i=1:1:m,
C(:,i)=S(:,i)-B';       % subtract average from features
end;

L=C'*C;
[V,D]=eig(L);           % eigen vectors

M=C*V;
for i=1:1:m T(:,i)=norm(M(:,i),2); end;
for i=1:1:m XR(:,i)=M(:,i)./T(:,i); end;
for i=1:1:NUMEIG,
I(:,i)=XR(:,227-i); %eigenvector images
end;

count=0;

for n=1:1:NUMEIG
for i=1:1:m
product=sum(I(:,n).*C(:,i));
count=count+1;
K(n,i)=product;
end;
end;

% produce the pattern file for the net
metr=0;
```

```
count=0;
fid=fopen('datap1','w');

fprintf(fid,' p\n');

for n=1:1:m
for k=1:1:NUMEIG
if count==20
fprintf(fid,'\n');
count=0;
end;
if metr==60
fprintf(fid,' p\n');
metr=0;
end;
fprintf(fid,'%8.4f ',K(k,n));
count=count+1;
metr=metr+1;

end;
end;
fprintf(fid,' p\n');
metr=0;
count=0;
fclose(fid);

!rbp epal1 >> file1

clear A;
clear B;
clear S;
clear L;
clear C;
clear V;
```

```
clear M;
clear I;
clear XR:
clear K;
clear D;
```

This program is doing the post processing for the fft-I method

```
function fftfft(SW)
G=SW;
QQ=zeros(360,226);
RR=zeros(360,226);
for lo=1:1:226
QQ(:,lo)=fft2(G(:,lo));
end;

for oo=1:1:226
RR(:,oo)=angle(QQ(:,oo));
end;

metr=0;
count=0;

fid=fopen('datap','w');

fprintf(fid,' p\n');

for n=1:1:226
for k=1:1:360
if count==20
fprintf(fid,'\n');
count=0;
end;
if metr==360
fprintf(fid,' p\n');
```

```
metr=0;
end;
fprintf(fid,'%8.4f ',RR(k,n));
count=count+1;
metr=metr+1;
end;
end;

fprintf(fid,' p\n');

metr=0;
count=0;
fclose(fid);

!rbp epal >> file

clear SW;
clear QQ;
clear RR
clear G;
```

This program is doing the post processing for the fft-II method

```
function(G)
QQ=zeros(512,226);
RR=zeros(512,226);
for lo=1:1:226
QQ(:,lo)=fft2(G(:,lo));
end;

load -ascii sotiras % load the stored mask

as=sotiras;

for rr=1:1:m
```

```
EE(:,rr)=as.*QQ(:,rr);
end;              % put in a matrix all the selected frequencies

WW=real(EE)/10000;
JJ=imag(EE)/10000;   % so as to have proper floating point values

fid=fopen('data','w')
for dd=1:1:m
fprintf(fid,'%8.4f ',WW(:,dd));
fprintf(fid,'\n');
fprintf(fid,'%8.4f ',JJ(:,dd));
fprintf(fid,'\n');
end;
fclose(fid);

load -ascii data;

for dd=1:1:512
if nnz(data(:,dd))==0
data(:,[dd])=[];
end;
end; % create the masked data

filo=data';

kolon=430;
metritis=0;count=0;
fid=fopen('datapi','w');
fprintf(fid,'p\n');
for dd=1:1:kolon
for nn=1:1:57
if count=114
fprintf(fid,'p\n');
count=0;
```

```
end;
fprintf(fid,'%8.4f ',filo(nn,dd));
metritis=metritis+1;
count=count+1;

if metritis==19
fprintf(fid,'\n');
metritis=0;
end;

end;
end;

fclose(fid);
clear QQ;
clear RR;
clear G;
clear as;
clear am;
!rbp epali >> filei
!rm datapi
!rm data
```

This program is doing the verification through matlab for the first two nets.

```
load -ascii 'pic15'; % load the actual image
DD=pic15;
[r,w]=size(DD);

SW=zeros(360,226);

fd=fopen('synt','w'); % open a file that holds the coordinates as
                      % well

col=1;
```

```
metrao=0;
for p=1:1:(r-30)
for s=1:1:(w-12)
pp=p+29;
ss=s+11;

if metrao==226
metrao=0;

facet(SW) ;

fftfft(SW)  % cp the fft angle data file
        % its output will be again datap
SW=zeros(360,226);
end;



SW(:,col)=reshape(DD(p:pp,s:ss),360,1); % prints the windows
col=col+1;
if col==226
col=1;
end;
fprintf(fd,'%d %d\n',p,s);     % holds the coords of the top left cor
metrao =metrao+1;
end;                           % of the extracted window
end;

!compress file1
!compress file
!compress synt

fclose(fd);
!rm datap
```

```
!rm datap1
!rm dataf1
quit
```

This program is doing the verification for the fft-II net

```
load -ascii 'pic15'; % load the actual image
DD=pic15;
[r,w]=size(DD);

G=zeros(512,226);

fd=fopen('synt','w'); % open a file that holds the coordinates as
                      % well

col=1;
metrao=0;
for p=1:1:(r-32)
for s=1:1:(w-16)
pp=p+31;
ss=s+15;

if metrao==226
metrao=0;
ffttwo(G)
G=zeros(512,226);
end;

G(:,col)=reshape(DD(p:pp,s:ss),360,1); % prints the windows
col=col+1;
if col==226
col=1;
end;
fprintf(fd,'%d %d\n',p,s);     % holds the coords of the top left cor
metrao =metrao+1;
```

```
end;                                    % of the extracted window
end;

!compress filei
!compress synt

fclose(fd);

quit
```

This program is thresholding the best probabilities

```
s=fopen('net_outcome#','r');
A=fscanf(s,'%f %f %f %f',[4,inf]);
status=fclose(s);

[a,b]=size(A);

s=fopen('coords','r');
X=fscanf(s,'%d %d ',[2,inf]);
status=fclose(s);
[c,d]=size(X);

fid=fopen('best_wind#','w');

for i=1:1:b

best=max(A(:,i)); % find the biggest probability of the 4
if best > 0.9 % it becomes 0.8 in the fft reduced frequencies case
fprintf(fid,'%d %d %d %8.4f\n',X(:,i),best);
end;
end;
end;
fclose(fid);
```

This program projects the selected probabilities on the normalised image

```
s=fopen('pic19','r');
Z=fscanf(s,'%d ',[256,256]);
status=fclose(s);

[a,b]=size(Z);

B=zeros(256,256);
C=zeros(256,256);
D=zeros(256,256);
E=zeros(256,256);

s=fopen('best_wind','r');
A=fscanf(s,'%d %d %d %f',[4,inf]);
status=fclose(s);

[c,d]=size(A);
for i=1:1:d
   if A(3,i)==1     % if it is left eye
      C(A(1,i)+6,A(2,i)+15)=A(4,i);
elseif A(3,i)==2    % if it is right eye
      B(A(1,i)+6,A(2,i)+15)=A(4,i);
  elseif A(3,i)==3  % if it is a nose
      D(A(1,i)+6,A(2,i)+15)=A(4,i);
elseif A(3,i)==4    % if it is a mouth
      E(A(1,i)+6,A(2,i)+15)=A(4,i);
  end;
end;



B=B*255;
C=C*255;
```

```
D=D*255;
E=E*255;


colormap(gray(256));
title('Centres of windows identified by the net as facial features')
imagesc(1.5*Z+B+C+D+E);
```

This program plots the cross from the best probabilities

```
s=fopen('pic19','r');
Z=fscanf(s,'%d ',[256,256]);
status=fclose(s);


load -ascii best_wind;
X=best_wind;

[a,b]=size(X);

r=10; s=10;

for i=1:1:b

maxev=-1;
minx = min(min(X(:,1)));
maxx = max(max(X(:,1)));
miny = min(min(X(:,2)));
maxy = max(max(X(:,2)));
for x=minx:1:maxx

for y=miny:1:maxy

sum=0;
for j=1:1:a
```

```
if X(j,3)==i
        if x <= X(j,1)
            if X(j,1) < x+r
                if y<= X(j,2)
                    if X(j,2) < y+s
                        sum=sum + X(j,4);
                    end;
                end;
            end;
        end;
end;
end;

if sum > maxev

maxev=sum;
savex=x;
savey=y;
end;

end;
end;

for x=1:1:r-1
Z((savex+x+6),savey+(s/2)+15)=255;
end;

for y=1:1:s-1
Z(savex+(r/2)+6,(savey+y)+15)=255;
end;

end;
end;
```

```
end;

figure(3);
colormap(gray(256));
image(Z)
```

Chapter 6

Conclusions

All three net configurations were successful in recognising the features and identifying their relevant positions on the facial image. The erroneous classifications occured only in the case of the left and the right eyes; the net was getting slightly confused as to which eye was which, a fact that was expected, since the eyes are quite similar. In the nose and the mouth cases the net did not misclassify anything at all. This is quite logical since these two features are very different from each other and the centre of the extracted windows was also important in stressing out this difference.

The preprocessing and postprocessing techniques were essential to this success sinc e, there was an effort to maintain all the meaningful information to the net, while keeping the number of the input units as small as possible. This was attempted because a relatively simple net configuration can be handled better than a complex one. The results are more straightforward, the net configuration can be easily modified without disturbing the entire recognition mechanism, and the net training although still time consuming has a better success percentage. An important thing was that the preprocessing eliminated irrelevancies - like intensity gradients and shadows.

The three preprocessing techniques had something in common ; they all tried to maintain identical important information for each feature, i.e. the shape, the grayscale, the size. These important components were not isolated, because a detailed representation has usually more chances to succeed in the long run.

This was proven without any doubt in this project, since the nets recognised and correctly identified the position of each feature. Special consideration was placed on the configuration itself; it is well known that the right one can have impressive results. This is mainly the reason, so many configurations were tested repeatedly, with different combinations of seed, sharpenss, number of units in the second level, fully or partly connected, along with different learning rules (quick back propagation and the delta-bar-delta).

6.1 Further Work

Given more time, further testing should be done in order to obtain more information about the net behaviour. Unfortunately, due to the fact that there were three nets that had to be tested, it was impossible to do more experiments. Another important factor, was that matlab was rapidly consuming big amounts of swap space, thus causing a lot of difficulties for other users. It would also be interesting to find out more efficient search strategies that could reduce the postprocessing time. The classification could speed up by parallelism, concentrating on likely positions. Although the postprocessing was so slow the learning speed was not a problem. Ideally this project could continue by recognising individual faces, not as features, but classifying the scanned face as 'known' i.e. as a person with a name the net can distinguish among other faces. According to the author's opinion, more emphasis should be placed in the future in the experimentation and testing of various net categories as well as different kinds of learning ie supervised or unsupervised. Artificial Neural Nets have been proven quite successful where the traditional vision techniques were definitely unsuccessful. In the cases they were successful the entire recognition procedure was extremely difficult to automate or if they were eventually automated they were eventually automated they were memory and cpu time consuming.

In an era where speed and accuracy are essential, old recognition techniques will eventually become obsolete. Neural Nets, have too many advantages to be ignored even by those still dedicated to the old ways. As a concept, they are ideal, what better than to imitate a tested way to recognise and identify objects; that of the human being. Our memory capacity seems unlimited, we are able to

accomplish things under the most difficult conditions and that is all due to the human nervous system. If a more accurate representation is achieved, neural nets will be an immense success. They represent an intelligent way to discover things as oppossed to other methods.

In order to come to that point eventually, a close examination of the mammalian nervous system has to be attempted. If scientists know something more than its basic structure, they wil be able to come up with a more accurate model.

This will give the neural net scientists the opportunity to create an even better model, or even to improve the existing ones.

Bibliography

[1] I. Craw and P. Cameron, *Face Recognition by Computer.* Proc. British Machine Vision Conference, 1992.

[2] C. Ramsay, K. Sutherland, D. Renshaw and P. Denyer, *A Comparison of Vector Quantization Codebook Generation Algorithms Applied to Automatic Face Recognition.* Proc. British Machine Vision Conference, 1992.

[3] A. Bennett and I. Craw, *Finding Image Features Using Deformable Templates and Detailed Prior Statistical Knowledge.* Proc. British Machine Vision Conference, Glasgow, 1991.

[4] V. Govindaraju, S. Srihari and D. Sher, *A Computational Model For Face Location.* Proc. 3rd International Conference on Computer Vision, Japan, 1990.

[5] E. Rolls, *The Processing of Face Information in the Primate Temporal Lobe*, in V. Bruce and M. Burton (eds), Processing Images of Faces, Ablex Publishing, 1992, pp41-68.

[6] F. Fallside and L.-W. Chan, *Connectionist Models and Geometric Reasoning*, in Woodwark (ed), Geometric Reasoning, Clarendon Press, Oxford, 1989, 65-79.

[7] K. Fukushima, *Cognitron: A Self-Organizing Multi-layered Neural Network*, Biological Cybernetics 20, 121-136, 1975.

[8] K. Fukushima, *NeoCognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by a Shift in Position*, Biological Cybernetics 36, 193-202, 1980.

[9] J. Stonham, *Practical face recognition and verification in WISARD*, in Ellis, Jeeves, Sumby and Young (Eds), Aspects of face processing, Dordrect: Martinus Nijhoff, 1986.

[25] MSc thesis, Inger Solheim, 1991, *Backpropagation Neural Nets for Facial Recognition.*

[26] Gonzalez and Wintz, eds Addison Wesley Publishing Co., Advanced Book Program, World Science Division Reading, Massachusetts 1983 *Digital Image Processing*

[27] New York Chichester Wiley, 1973, Duda and Hart,*Pattern Classification and Scene Analysis*

[28] J.A. Scott Kelso, eds Haskins Laboratories and the University of Connecticat, 1987 *Concepts and Issues in Human Motor Behaviour : Coming to Grips with the Jargon*