Webcam-based User Interface

Submitted To

Dr. Bovik

Prepared By

Nathan Shepard

EE371R Image and Video Processing
Electrical and Computer Engineering Department
University of Texas at Austin

Spring 2006

CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION

As the personal computing industry pursues more user-friendly, inexpensive user interfaces, the concept of a touchless interface is worthy of inspection. This report documents the implementation of a webcam-based user interface using an average personal computer and an inexpensive, low-quality web camera. Using the video feed from the webcam, this product would track the user's eyes and find their focus on the screen, controlling a cursor. This product is useful for sterile environments such as hospitals as well as public terminals where sanitation is important. The disabled can make use of such a device, as well as anyone who would like to operate their television from across the room without having to find the remote.

2.0 PROJECT GOALS

Although this project pursues the vision of an entire software system of image processing, cursor control, graphical user interface (GUI), and a calibration routine, this is an unreasonable expectation during one semester of part-time work. Therefore, it will be the purpose of this project to demonstrate a proof-of-concept, documenting the core image processing and neural network methods that will interpret useful user input through an inexpensive web-camera in real time.

3.0 REJECTED METHODS

Before the final image processing technique was implemented, several other algorithms were devised and discarded for reasons of quality and efficiency.

3.1 PATTERN-MATCHING ALGORITHM

The first attempt at interpreting the focus of the user's eye involved a simple pattern-matching algorithm that searched the image for round, dark spots. Although the algorithm could be trained to avoid non-symmetric spots and other confusing non-eye patterns, it performed so very slowly that no reasonable attempt could be made to implement this algorithm at 15 frames per second

(FPS) on an average personal computer (PC). From this attempt, I learned that the first stage of processing must be very fast and it must crop the image down to a more reasonable size.

3.2 FACE- DETECTION ALGORITHM

The second implementation of the webcam-based cursor attempted to quickly distinguish the face from the background of the image using edge detection and flesh tones as guides. The length and complexity of this algorithm grew much more quickly than its success rate (at best 70%), though it was able to quickly reduce the size of the image that needed processing, rejecting meaningless background data. Still, the first stage needed to be faster and more accurate.

4.0 SUCCESSFUL METHOD

Learning both from failed attempts and class lectures, an algorithm capable of quickly discerning the location of the user's eyes from a low-resolution image was implemented.

4.1 SYMMETRY METHOD FOR IMAGE REDUCTION

During the training of the error-rejection in the pattern-matching algorithm described in section 3.1, a symmetry algorithm was implemented and found to be surprisingly accurate. By calculating the symmetry across the vertical axis for each column, the column through the center of the face could be found with accuracy over 95% (See figure 1). Although symmetric background items occasionally confused the algorithm, the size of the face relative to any other symmetric object almost always took precedence. When a similar algorithm was implemented to find the row-center of the face in the image, overwhelmingly the least symmetric point indicated the horizontal center of the face (See figure 1).

Once the error for the entire algorithm was below 5%, this process was made faster by skipping 3 out of every 4 rows and columns, so that it found the center to within 4 pixels at a rate of 30 FPS on a 900MHz PC. Once this center is found, the eyes are assumed to be within an appropriate distance, and the image is cropped to approximately 16% of the original size around that center (See figure 2).

4.2 FREQUENCY ANALYSIS

In order to quickly find the location of the user's eyes, analysis of the Fourier transform of the reduced image is the most efficient. After preliminary thresholding and morphological filtering of the reduced image (See figure 3), the frequency-transforms of 50 sample images were inspected for common indicators. Afterwards, a suitable filter was designed to uniquely highlight the location of the eyes (See figure 4). This process takes longer than the first stage; however, they run in series at a rate of 18 FPS.

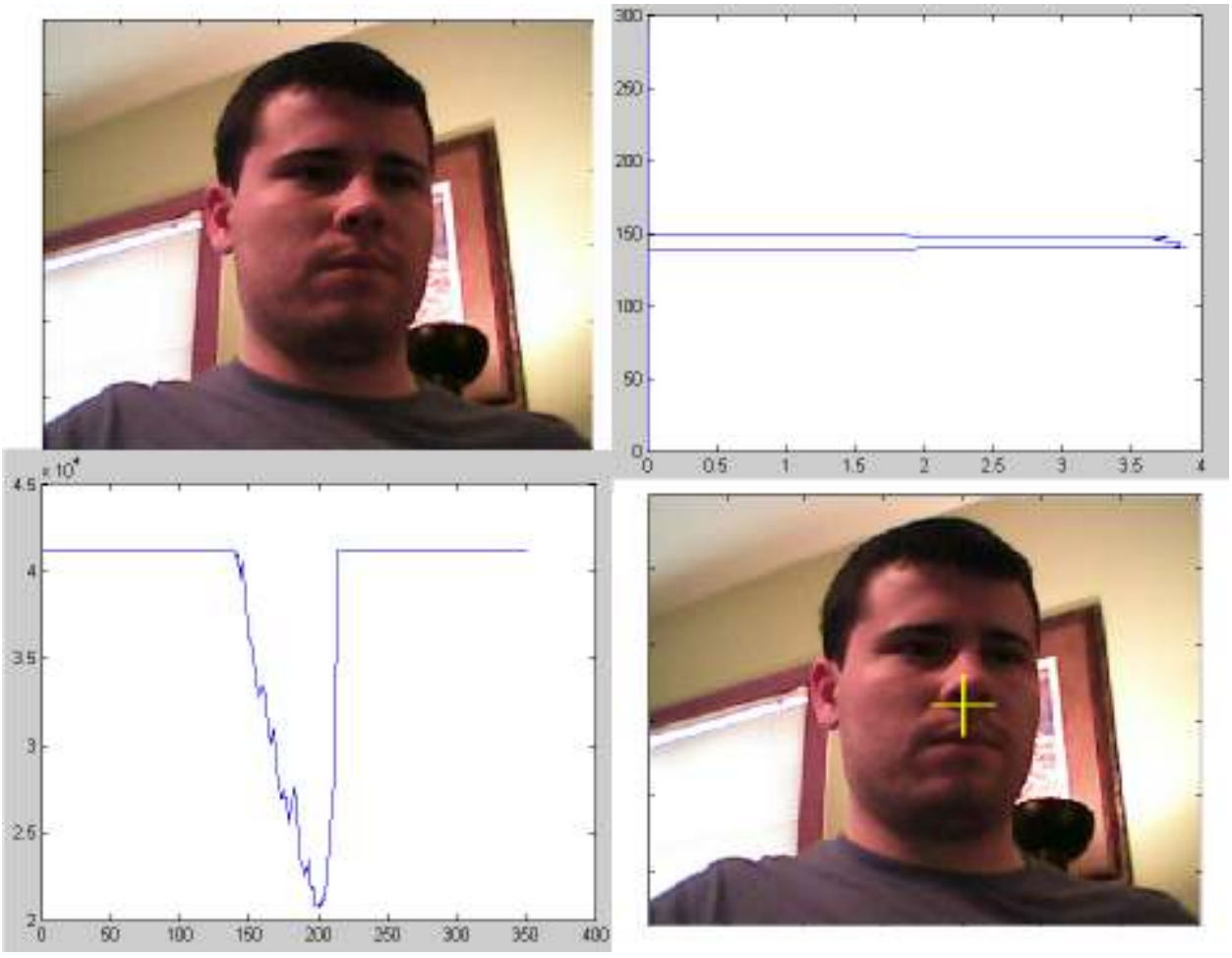

Figure 1: Symmetry of Face. Original Image (Top-left) Horizontal (Top-right) Vertical (Bottom-left) d) Center of Face (Bottom-right)

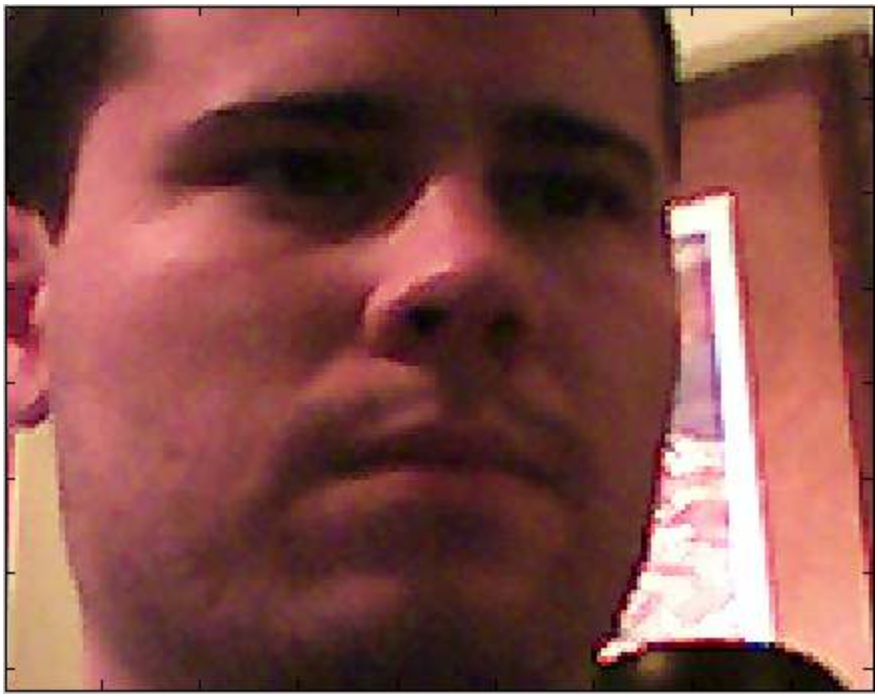

Figure 2: Symmetry-Reduced Image

Figure 3: Morphologically-Filtered Reduced Image

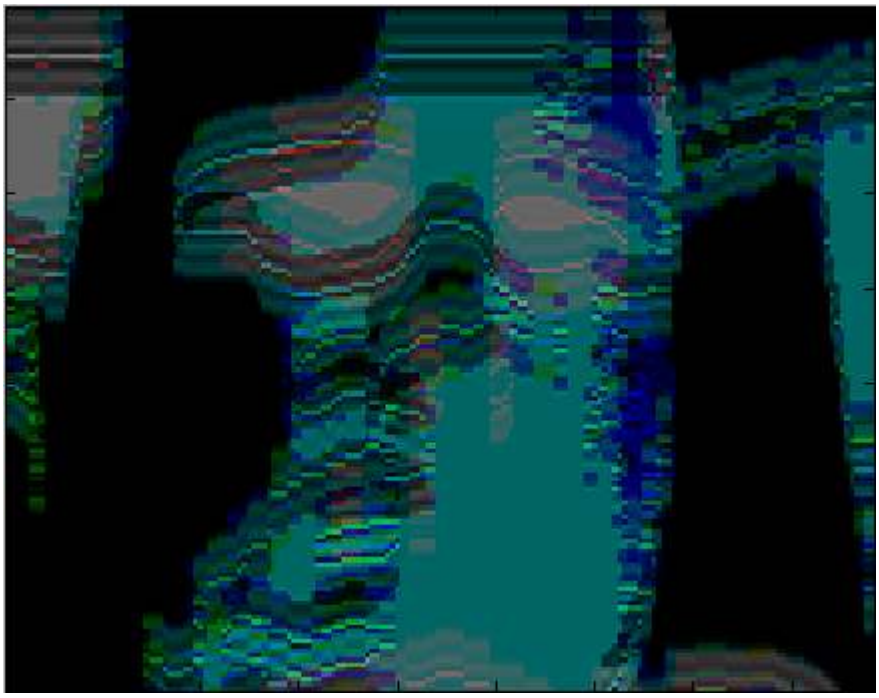

Figure 4: Band-pass Filtered Reduced Image

4.3 THRESHOLDING AND REDUCING

Once this filtering is complete, the result is thresholded and the binary image is ‘ blob colored.’ Each ‘blob’ area is given a number, and the area of each is computed (See figure 5). Those with areas outside parameters are discarded, as well as those which do not have a closely matching twin (See figure 6). This way, the ‘blobs’ representing the eyes are isolated from the noise and can be easily picked out with an algorithm that finds the centers of the two most likely ‘blobs.’ Once these centers are found, the algorithm further reduces the image into 2 eye images and then performs pattern analysis to center the reduced images on the pupils (See figure 7). Unfortunately, the filtering and additional processing necessary to ensure an acceptable accuracy (about 90%) makes the algorithm more cumbersome in terms of efficiency. The entire process up to this point can process webcam video at 3 FPS.

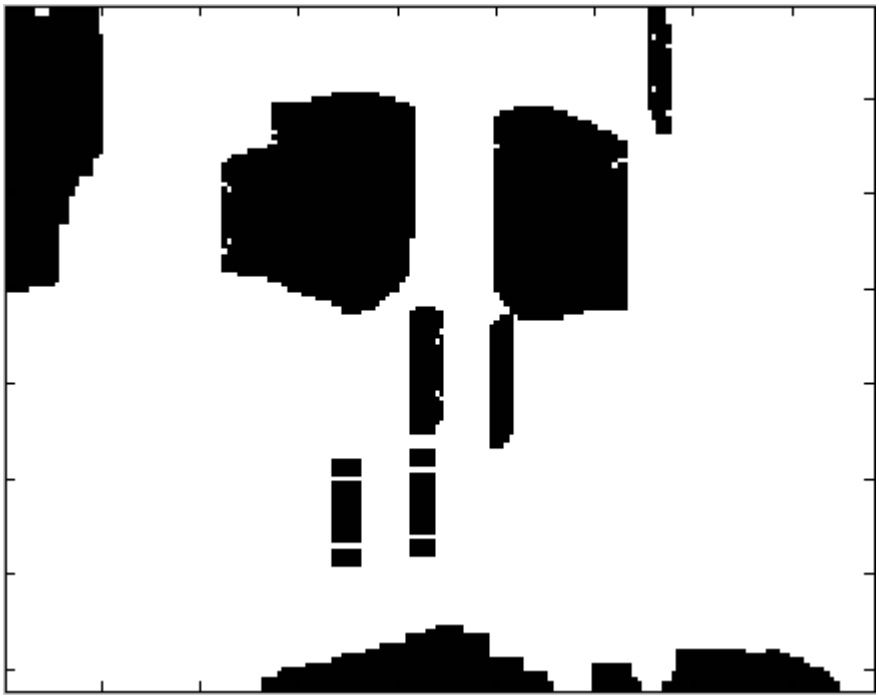

Figure 5: Thresholded, Filtered, Reduced Image

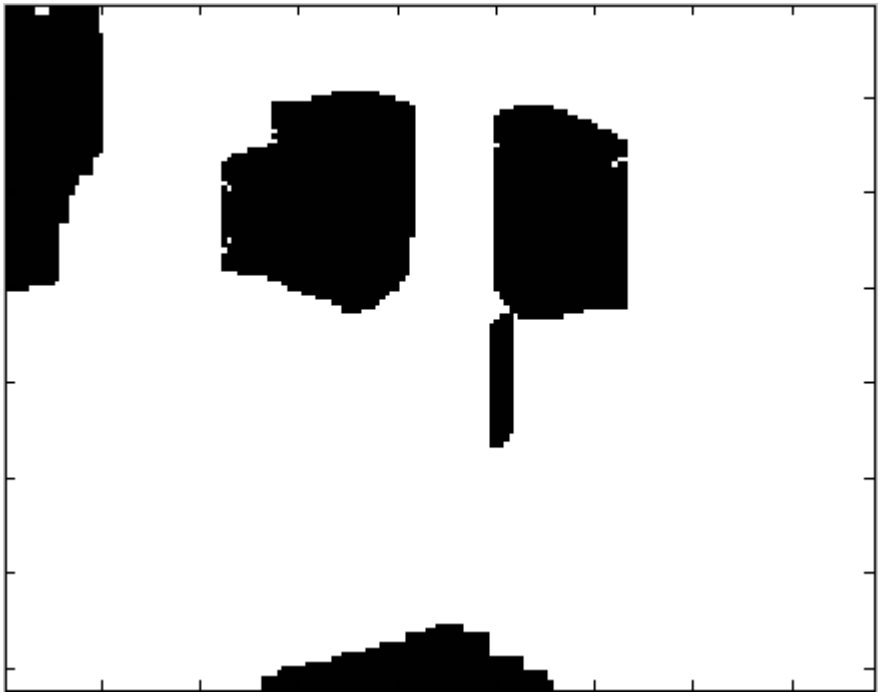

Figure 6: Figure 5 After Coloring and 'Blob' Rejection

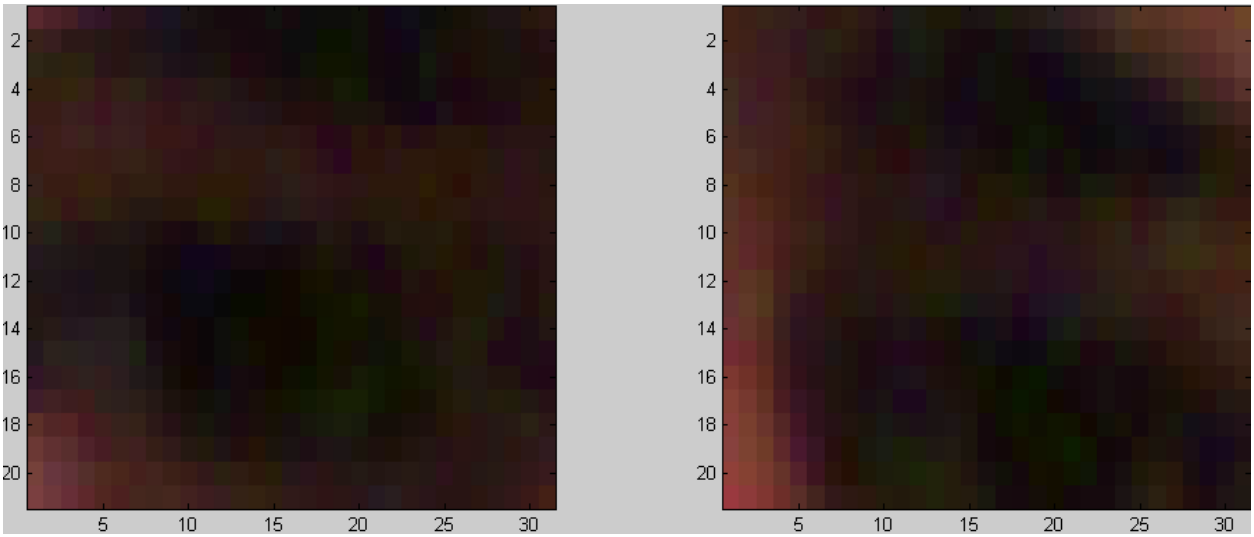

Figure 7: Reduced Eye Images

4.4 VECTORIZING AND NEURAL NETWORK IMPLEMENTATION

Once the two eye images are recorded, the image data is vectorized and combined with the coordinates of the center of the face and each eye (See figure 8). This vector is passed to a trained neural network, which returns an on-screen coordinate in less than 350 add/multiply cycles (See figure 9). The entire process from image acquisition to on-screen coordinate calculation can be implemented at 2 FPS on a 900MHz PC.

5.0 RESULTS

Unfortunately, time and resources did not allow the implementation of the calibration function needed to properly train the neural network. With a small set of training data, however, this algorithm performed fairly well. The accuracy of the on-screen coordinates is not high, and the variance between frames is much too high; however, with low-pass filtering of the coordinates with respect to time, a delayed but acceptable result has been observed. One notable problem occurred during user quick motion or lack of focus in the webcam. Image blur caused the algorithm to fail drastically, but time limited my ability to pursue a solution. However, in over 90% of the 50 test cases, the algorithm returned the proper eye images, coordinates, and input

vectors to the neural network, and this is the proof-of-concept necessary to make this experiment a success.

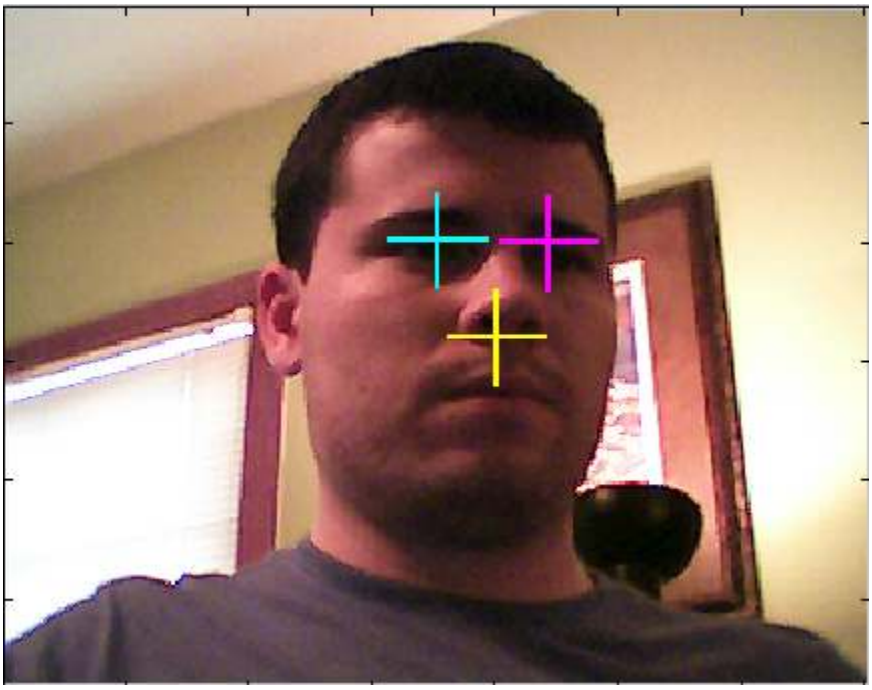

Figure 8: Original Image Marked with Center of Face (Yellow), Left Pupil (Magenta), and Right Pupil (Cyan)

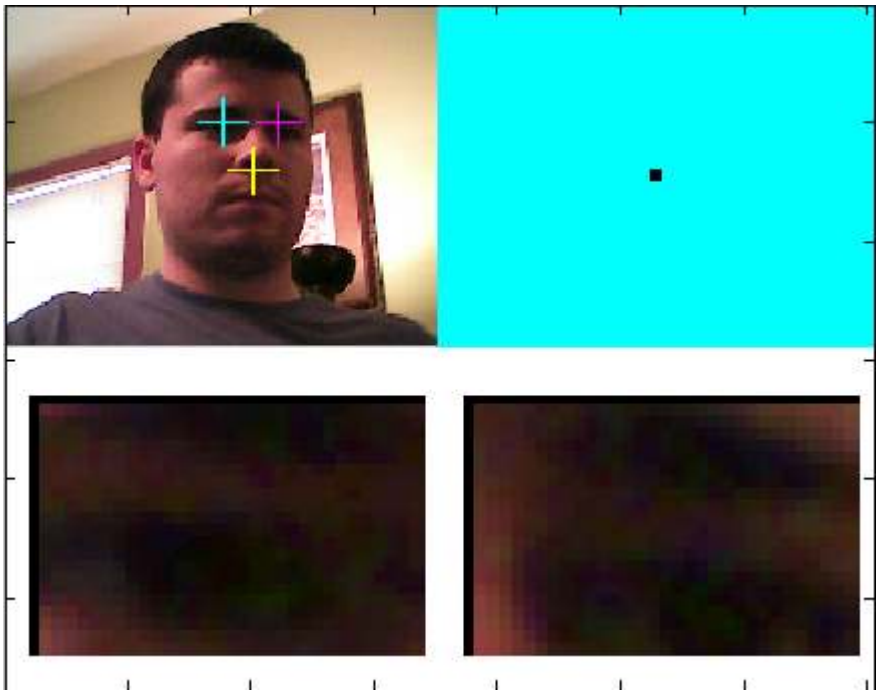

Figure 9: Original Marked Image (top left), On-screen Coordinate (top right)

6.0 FUTURE IMPROVEMENTS

Because the results of this project are so encouraging, the implementation of a calibration function would be a relatively simple improvement that would make the result much more impressive. Basically, this function would train the neural network by having the user stare at a series of points on the screen for a period of about thirty seconds each time they logged in. It would then train the neural network to associate the resulting vectors with the known on-screen coordinates and adjust for ambient light conditions and differences in users' visual appearance. This requires both the capture stream and the processing to be handled on the same terminal, and as I did not have MATLAB installed on the computer that was capturing the video stream, this was not possible in this project.

Once this calibration function is implemented, a base neural network should be trained using a variety of users and lighting conditions. Changes in these variables proved to be the most difficult to adjust for, and a base network will train to a specific person much more quickly than an unbiased network will.

In order to increase the processing speed, it may be beneficial to run the eye location function every nth frame instead of every frame due to the fact that they eyes do not move very quickly. Experimentation is necessary to justify this hypothesis.

7.0 CONCLUSIONS

Although many implementation issues remain, most of the problems lie in the neural network and hardware interfacing. As a software solution, this image processing algorithm demonstrates the ability to quickly determine the on-screen focus of a user's eyes by means of a low-cost (<$30) web camera. Although the processing speed is well below the camera standard of 15 FPS, I was able to attain 2 FPS, which is an acceptable result and certainly demonstrates the ability of the average PC to handle this task.

APPENDIX A –MATLAB CODE

APPENDIX A – MATLAB CODE

```
function [net error its X T] = dir_train(nhidden,tlimit,X,T);

%nhidden = 500;
nout = 2;

filenames = ['2ft - color              '; '2ft - color bottom left '; '2ft - color bottom right'; '2ft - color top left    '; '2ft - color
top right   '];
dirs = [0.5 0.5; 1 0; 1 1; 0 0; 0 1];

if X == 1
   [X,T] = trainingdata(filenames,dirs);
end

x = vect(filenames(1,:),1);
nin = length(x);

its = 0;
while (nin+nout)*nhidden*(its+1) < tlimit*400000
   its = its + 1;
end

net = mlp(nin,nhidden,nout,'linear');
[net error] = mlptrain(net,X,T,its);

end
```

```
function [X T] = trainingdata(filenames,dirs);

[num_movs num_chars] = size(filenames);
index = 0;

for i = 1:1:num_movs
   mov = aviread(filenames(i,:));
   num_frames = length(mov);
   for j = 1:1:num_frames
      movie_frame = [i j]
      index = index + 1;
      x = vect(filenames(i,:),j);
      X(index,:) = x;
      T(index,:) = dirs(i,:);
   end
end
end

function x = vect(filename,frame);

sample = 4;
[A lI rI cvlo chlo cvro chro] = mousecam(filename,frame);
[m,n,l] = size(lI);
i = 1:sample:m;
j = 1:sample:n;
k = 1:1:l;
s = 1:1:length(i);
t = 1:1:length(j);

lA(s,t,k) = lI(i,j,k);
rA(s,t,k) = rI(i,j,k);

[m,n,l] = size(lA);
lB = int16(reshape(lA,1,m*n*l,1));
rB = int16(reshape(rA,1,m*n*l,1));
x = double([lB rB cvlo chlo cvro chro]);
end
```

```
function [B lI rI cvlo chlo cvro chro] = mousecam(filename,frame);

s = 140;
sample = 16;
scale = 0.5;
hwindow = 15;
vwindow = 10;

mov = aviread(filename);
A = mov(1,frame).cdata;

[ch cv diffh diffv] = center(A,s,sample);

[m,n,l] = size(A);
B = A;
B(max(cv-20,1):1:min(cv+20,m),ch-1:1:ch,3) = 0;
B(cv-1:1:cv,max(ch-20,1):1:min(ch+20,n),3) = 0;
B(cv-1:1:cv,max(ch-20,1):1:min(ch+20,n),[1 2]) = 256;
B(max(cv-20,1):1:min(cv+20,m),ch-1:1:ch,[1 2]) = 256;

C = A;
%C(:,ch,:) = 1;
%C(cv,:,:) = 1;

[m,n,l] = size(A);
newm = int16(m*scale);
newn = int16(n*scale);
i = 1:1:newm;
j = 1:1:newn;
k = 1:1:l;

p = (-newm/2):1:(newm/2)-1;
q = (-newn/2):1:(newn/2)-1;

D(i,j,k) = C((cv+p),(ch+q),k);

[E h] = thresh(D,50);
```

```
F = dilate(E);
G = erode(F);
H = erode(G);
P = dilate(H);

Q = xor(P,ones(size(P)));
Q = double(Q);

filt = firpm(20,[0 0.4 0.41 1],[0 0 1 1]);
U = abs(filter(filt,1,Q));

U = 256*U;

threshold1 = 1;
V(i,j) = (U(i,j,1) > threshold1) & (U(i,j,2) > threshold1) & (U(i,j,3) > threshold1);

percent = 0.6;
while sum(sum(V)') > percent*newn*newm
   V = dilate(V);
end

W = color(V);
V = xor(V,ones(size(V)));

[I hW] =thresh(W,1);
[m,n] = size(V);
for a = 1:1:length(hW)
   if ((hW(a) > 3000) | (hW(a) < 400)) & (hW(a) > 0)
      match = (W == a*ones(size(W)));
      V = V + match;
   end
end
i = 1:1:m;
j = 1:1:n;
V(i,j) = min(V(i,j),1);
```

```
V = double(V);
[chl cvl chr cvr] = findeyes(V,20,1);
[V h] = thresh(V*256,1);

B = A;

if chl ~= -1
    V(max(cvl-20,1):1:min(cvl+20,newm),chl) = 0;
    V(cvl,max(chl-20,1):1:min(chl+20,newn)) = 0;
    V(max(cvr-20,1):1:min(cvr+20,newm),chr) = 0;
    V(cvr,max(chr-20,1):1:min(chr+20,newn)) = 0;

    V = double(V);
    [m,n] = size(V);
    i = 1:1:m;
    j = 1:1:n;
    temp = V(i,j,1);
    V(i,j,2) = temp;
    V(i,j,3) = temp;

    [m,n,l] = size(V);
    newm = int16(m*0.3);
    newn = int16(n*0.3);

    f =1.75;
    chlo = ch + chl - f*newn;
    cvlo = cv + cvl - f*newm;
    chro = ch + chr - f*newn;
    cvro = cv + cvr - f*newm;

    search = 10;

    for i = cvro-search:1:cvro+search;
        for j = chro-search:1:chro+search;
            if darkness(A,i,j) < darkness(A,cvro,chro)
                cvro = i;
                chro = j;
```

```
        end
    end
end

for i = cvlo-search:1:cvlo+search;
    for j = chlo-search:1:chlo+search;
        if darkness(A,i,j) < darkness(A,cvlo,chlo)
            cvlo = i;
            chlo = j;
        end
    end
end

[m,n,l] = size(A);
chlo = max(hwindow+1,chlo);
chro = max(hwindow+1,chro);
chlo = min(m-hwindow,chlo);
chro = min(m-hwindow,chro);
cvlo = max(vwindow+1,cvlo);
cvro = max(vwindow+1,cvro);
cvlo = min(n-vwindow,cvlo);
cvro = min(n-vwindow,cvro);

i = cvlo-vwindow:1:cvlo+vwindow;
j = chlo-hwindow:1:chlo+hwindow;
k = 1:1:l;
s = 1:1:length(i);
t = 1:1:length(j);

lI(s,t,k) = A(i,j,k);

i = cvro-vwindow:1:cvro+vwindow;
j = chro-hwindow:1:chro+hwindow;
s = 1:1:length(i);
t = 1:1:length(j);

rI(s,t,k) = A(i,j,k);
```

```
    B(max(cv-20,1):1:min(cv+20,m),ch-1:1:ch,3) = 0;
    B(cv-1:1:cv,max(ch-20,1):1:min(ch+20,n),3) = 0;
    B(cv-1:1:cv,max(ch-20,1):1:min(ch+20,n),[1 2]) = 255;
    B(max(cv-20,1):1:min(cv+20,m),ch-1:1:ch,[1 2]) = 255;

    B(max(cvlo-20,1):1:min(cvlo+20,m),chlo-1:1:chlo,1) = 0;
    B(cvlo-1:1:cvlo,max(chlo-20,1):1:min(chlo+20,n),1) = 0;
    B(cvlo-1:1:cvlo,max(chlo-20,1):1:min(chlo+20,n),[2 3]) = 255;
    B(max(cvlo-20,1):1:min(cvlo+20,m),chlo-1:1:chlo,[2 3]) = 255;

    B(max(cvro-20,1):1:min(cvro+20,m),chro-1:1:chro,2) = 0;
    B(cvro-1:1:cvro,max(chro-20,1):1:min(chro+20,n),2) = 0;
    B(cvro-1:1:cvro,max(chro-20,1):1:min(chro+20,n),[1 3]) = 255;
    B(max(cvro-20,1):1:min(cvro+20,m),chro-1:1:chro,[1 3]) = 255;
else
    i = 1-vwindow:1:1+vwindow;
    j = 1-hwindow:1:1+hwindow;
    k = 1:1:l;
    s = 1:1:length(i);
    t = 1:1:length(j);

    lI(s,t,k) = 0;
    rI = lI;

    cvlo = -1;
    chlo = -1;
    cvro = -1;
    chro = -1;
end
end

%figure(1);
%subplot(1,2,1);
%plot(1:1:length(diffh'), diffh);
%subplot(1,2,2);
%plot(diffv,1:1:length(diffv'));
```

```
function [ch cv diffh diffv] = center(A,s,sample);

[m,n,l] = size(A);
A = double(A);

j = s:1:n-s;
diffh(j) = 0;
for i = 1:sample:m
   for k = 1:1:l
      for t = 1:sample:s-1
         diffh(j) = diffh(j) + abs(A(i,j+t,k) - A(i,j-t,k));
      end
   end
end

top = max(diffh');

j = 1:1:s-1;
diffh(j) = top;
j = n-s+1:1:n;
diffh(j) = top;

[sm ch] = min(diffh);



for k = 1:1:l
   a = A(:,:,k);
   B(:,:,k) = a';
end

[m,n,l] = size(B);
B = double(B);

j = s:1:n-s;
diffv(j) = 0;
for i = 1:sample:m
```

```
    for k = 1:1:l
        for t = 1:sample:s-1
            diffv(j) = diffv(j) + abs(B(i,j+t,k) - B(i,j-t,k));
        end
    end
end

top = max(diffv');

j = 1:1:s-1;
diffv(j) = 0;
j = n-s+1:1:n;
diffv(j) = 0;

[sm cv] = max(diffv);

end



function [E h] = thresh(A, threshold)

A = double(A);
[m,n,l] = size(A);

B = reshape(A, (m*n*l), 1);
h = hist(B,1:1:256);

i = 1:1:(m*n*l);
C(i) = (A(i) >= threshold);
D = reshape(C,m,n,l);
E = double(D);

end
```

```
function B = dilate(A);

[m,n,l] = size(A);
k = 1:1:l;
i = 1;
j = 2:1:n-1;
B(i,j,k) = A(i,j-1,k) & A(i+1,j-1,k) & A(i,j,k) & A(i+1,j,k) & A(i,j+1,k) & A(i+1,j+1,k);

i = m;
j = 2:1:n-1;
B(i,j,k) = A(i-1,j-1,k) & A(i,j-1,k) & A(i-1,j,k) & A(i,j,k) & A(i-1,j+1,k) & A(i,j+1,k);

i = 2:1:m-1;
j = 1;
B(i,j,k) = A(i-1,j,k) & A(i,j,k) & A(i+1,j,k) & A(i-1,j+1,k) & A(i,j+1,k) & A(i+1,j+1,k);

i = 2:1:m-1;
j = n;
B(i,j,k) = A(i-1,j-1,k) & A(i,j-1,k) & A(i+1,j-1,k) & A(i-1,j,k) & A(i,j,k) & A(i+1,j,k);

i = 2:1:m-1;
j = 2:1:n-1;
B(i,j,k) = A(i-1,j-1,k) & A(i,j-1,k) & A(i+1,j-1,k) & A(i-1,j,k) & A(i,j,k) & A(i+1,j,k) & A(i-1,j+1,k) & A(i,j+1,k) &
A(i+1,j+1,k);

B(1,1,k) = A(1,1,k) & A(1,2,k) & A(2,1,k) & A(2,2,k);
B(m,1,k) = A(m,1,k) & A(m,2,k) & A(m-1,1,k) & A(m-1,2,k);
B(1,n,k) = A(1,n,k) & A(1,n-1,k) & A(2,n,k) & A(2,n-1,k);
B(m,n,k) = A(m,n,k) & A(m,n-1,k) & A(m-1,n,k) & A(m-1,n-1,k);

end
```

```
function B = erode(A);

[m,n,l] = size(A);
k = 1:1:l;
i = 1;
j = 2:1:n-1;
B(i,j,k) = A(i,j-1,k) | A(i+1,j-1,k) | A(i,j,k) | A(i+1,j,k) | A(i,j+1,k) | A(i+1,j+1,k);

i = m;
j = 2:1:n-1;
B(i,j,k) = A(i-1,j-1,k) | A(i,j-1,k) | A(i-1,j,k) | A(i,j,k) | A(i-1,j+1,k) | A(i,j+1,k);

i = 2:1:m-1;
j = 1;
B(i,j,k) = A(i-1,j,k) | A(i,j,k) | A(i+1,j,k) | A(i-1,j+1,k) | A(i,j+1,k) | A(i+1,j+1,k);

i = 2:1:m-1;
j = n;
B(i,j,k) = A(i-1,j-1,k) | A(i,j-1,k) | A(i+1,j-1,k) | A(i-1,j,k) | A(i,j,k) | A(i+1,j,k);

i = 2:1:m-1;
j = 2:1:n-1;
B(i,j,k) = A(i-1,j-1,k) | A(i,j-1,k) | A(i+1,j-1,k) | A(i-1,j,k) | A(i,j,k) | A(i+1,j,k) | A(i-1,j+1,k) | A(i,j+1,k) |
A(i+1,j+1,k);

B(1,1,k) = A(1,1,k) | A(1,2,k) | A(2,1,k) | A(2,2,k);
B(m,1,k) = A(m,1,k) | A(m,2,k) | A(m-1,1,k) | A(m-1,2,k);
B(1,n,k) = A(1,n,k) | A(1,n-1,k) | A(2,n,k) | A(2,n-1,k);
B(m,n,k) = A(m,n,k) | A(m,n-1,k) | A(m-1,n,k) | A(m-1,n-1,k);

end
```

```
function [chl cvl chr cvr] = findeyes(A,s,sample);

[m,n,l] = size(A);
A = double(A);
i = 1:1:m;
j = 1:1:n;
k = 1:1:l;
diff(i,j) = 0;

for j = s:sample:n-s;
   for i = s:sample:m-s
      for k = 1:1:l
         if A(i,j,k) == 0
            for t = 1:sample:s-1
               diff(i,j) = diff(i,j) + abs(A(i+t,j-t,k) - A(i+t,j+t,k)) + abs(A(i-t,j-t,k) - A(i-t,j+t,k)) + abs(A(i+t,j+t,k) -
A(i-t,j+t,k)) + abs(A(i+t,j-t,k) - A(i-t,j-t,k));
            end
         end
      end
   end
end

top = max(max(diff)');

adjust = top*(diff == zeros(m,n));
diff = diff + adjust;

[minv indv] = min(diff);
[minh chl] = min(minv');
cvl = indv(chl);

i = 1:1:m;
j = 1:1:n;
diff2(i,j) = diff(i,j);

i = [1:1:max(cvl-15,1) min(cvl+15,m):1:m];
diff2(i,:) = top;
```

```
j = [max(chl-30,1):1:min(chl+30,n) 1:1:max(1,chl-100) min(n,chl+100):1:n];
diff2(:,j) = top;

if min(min(diff2)') == top
   i = 1:1:m;
   j = 1:1:n;
   diff2(i,j) = diff(i,j);

   i = max(cvl-20,1):1:min(cvl+20,m);
   j = max(chl-20,1):1:min(chl+20,n);
   diff2(i,j) = top;

   [minv indv] = min(diff2);
   [minh chl] = min(minv');
   cvl = indv(chl);

   i = [1:1:max(cvl-20,1) min(cvl+20,m):1:m];
   diff2(i,:) = top;

   j = [max(chl-20,1):1:min(chl+20,n) 1:1:max(1,chl-80) min(n,chl+80):1:n];
   diff2(:,j) = top;
end

[minv indv] = min(diff2);
[minh chr] = min(minv');
cvr = indv(chr);

if chr < chl
   temph = chr;
   tempv = cvr;
   chr = chl;
   cvr = cvl;
   chl = temph;
   cvl = tempv;
end
```

```
if (chl == 1) && (cvl == 1)
    chl = -1;
    cvl = -1;
end
```

```
function [chl cvl chr cvr] = findeyes(A,s,sample);

[m,n,l] = size(A);
A = double(A);
i = 1:1:m;
j = 1:1:n;
k = 1:1:l;
diff(i,j) = 0;

for j = s:sample:n-s;
   for i = s:sample:m-s
      for k = 1:1:l
         if A(i,j,k) == 0
            for t = 1:sample:s-1
               diff(i,j) = diff(i,j) + abs(A(i+t,j-t,k) - A(i+t,j+t,k)) + abs(A(i-t,j-t,k) - A(i-t,j+t,k)) + abs(A(i+t,j+t,k) -
A(i-t,j+t,k)) + abs(A(i+t,j-t,k) - A(i-t,j-t,k));
            end
         end
      end
   end
end

top = max(max(diff)');

adjust = top*(diff == zeros(m,n));
diff = diff + adjust;

[minv indv] = min(diff);
[minh chl] = min(minv');
cvl = indv(chl);

i = 1:1:m;
j = 1:1:n;
diff2(i,j) = diff(i,j);

i = [1:1:max(cvl-15,1) min(cvl+15,m):1:m];
diff2(i,:) = top;
```

```
j = [max(chl-30,1):1:min(chl+30,n) 1:1:max(1,chl-100) min(n,chl+100):1:n];
diff2(:,j) = top;

if min(min(diff2)') == top
   i = 1:1:m;
   j = 1:1:n;
   diff2(i,j) = diff(i,j);

   i = max(cvl-20,1):1:min(cvl+20,m);
   j = max(chl-20,1):1:min(chl+20,n);
   diff2(i,j) = top;

   [minv indv] = min(diff2);
   [minh chl] = min(minv');
   cvl = indv(chl);

   i = [1:1:max(cvl-20,1) min(cvl+20,m):1:m];
   diff2(i,:) = top;

   j = [max(chl-20,1):1:min(chl+20,n) 1:1:max(1,chl-80) min(n,chl+80):1:n];
   diff2(:,j) = top;
end

[minv indv] = min(diff2);
[minh chr] = min(minv');
cvr = indv(chr);

if chr < chl
   temph = chr;
   tempv = cvr;
   chr = chl;
   cvr = cvl;
   chl = temph;
   cvl = tempv;
end
```

```
if (chl == 1) && (cvl == 1)
    chl = -1;
    cvl = -1;
end
```