APÉNDICE B

ARCHIVOS .m DE MATLAB

En este Apéndice se presentan los códigos tal cual se usaron en los programas de este trabajo.

LearnImage.m

```
%Trains and creates a neural network to learn the
%distance transform of the image file given.
%"l" will be the name of the neural network to recall.
%training algorithm used: Levenberg-Marquardt
function l=LearnImage
image=input('Enter name of the image file to be learned: ','s');
global dt
tic;
image=imread(['C:\Path\' image '.jpg']);
DistanceTransform(image);
disp('Training...')
l=NNDT(dt);
toc;
```

DistanceTransform.m

```
%Computes the distance transform of "image" in a matrix of 64X64
%Distance metric: Euclidean
function DistanceTransform(image)
global dt
dt=edge(image, 'canny'); %edge in binary
dt=~dt; %function NOT so that pixels within the edge worth 0
```

```
dt=255*dt; %pixels out of the edge worth 255
%Distance transform
for i=2: 63,
   for j=2: 63,
      neighbors=[dt(i-1,j-1)+1.4142,dt(i-1,j)+1,dt(i-1,j+1)+1.4142,dt(i,j-
1)+1,dt(i,j),dt(i+1,j-1)+1.4142];
      dt(i,j)=min(neighbors);
   end
end
for i=63:-1:2,
   for j=63:-1:2,
      neighbors=[dt(i-1,j+1)+1.4142,dt(i,j),dt(i,j+1)+1,dt(i+1,j-
1)+1.4142,dt(i+1,j)+1,dt(i+1,j+1)+1.4142];
      dt(i,j)=min(neighbors);
   end
end
```

NNDT.m

```
%Neural Network-Distance Transform
%Creates a feedforward neural network of 2X9X25X1 that computes the
%distance transform "dt"
function net=NNDT(dt)
%selecting patterns
global dt

inputs=zeros(2,1922);
outputs=zeros(1,1922);
for i=2: 63,
   c=mod(i,2);
   for j=1: 31,
      inputs(1,j+(i-2)*31)=i;
```

```
        inputs(2,j+(i-2)*31)=(2+(j-1)*2+c);
        outputs(j+(i-2)*31)=dt(i,2+(j-1)*2+c);
    end
end

%defining and training neural network
net=newff(minmax(inputs),[9,25,1],{'tansig','tansig','purelin'},'trainlm');
net.trainParam.show = 10;
net.trainParam.epochs = 2000;
net.trainParam.goal = 5e-2;
net.trainParam.mu_max = 1000000000000;
net = init(net); %weights and thresholds are initialized
net=train(net,inputs,outputs);
```

Recall.m

```
%Recalls object from imagev.jpg in folder "images"
%imagev.jpg MUST be grayscale
v=1;
while v>0
    v=input('Enter number of image: ');
    if v<=0
        break
    end
    disp(' ');
    tic;
    image=imread(['C:\Path\' int2str(v) '.jpg']);
    image=edge(image, 'canny'); %edge in binary
    image=~image; %function NOT so that pixels within edge worth 0
    x=0;
    inputs=zeros(2,1000); %guessing that less than 1000 pixels could define the edge
    for i=1: 64,
```

```
    for j=1: 64,
        if(image(i,j)==0)
            x=x+1;
            inputs(1,x)=i;
            inputs(2,x)=j;
        end
    end
end
d1=sim(Object1,inputs([1:2:x*2;2:2:x*2]));
d2=sim(Object2,inputs([1:2:x*2;2:2:x*2]));
d3=sim(Object3,inputs([1:2:x*2;2:2:x*2]));
d4=sim(Object4,inputs([1:2:x*2;2:2:x*2]));
d5=sim(Object5,inputs([1:2:x*2;2:2:x*2]));
%...
%object will be given only by the minimum average of distances da,de...
%since is the same that the absolute value of the difference among
%an output vector and zero. Zero is the value of those pixels within
%the edge of the image
d1=mean(d1);d2=mean(d2);d3=mean(d3);d4=mean(d4);d5=mean(d5); %...
l=min([d1,d2,d3,d4,d5]);
if (l==d1)
    disp(['image' int2str(v) '.jpg is an Object1'])
elseif (l==d2)
    disp(['image' int2str(v) '.jpg is an Object2'])
elseif (l==d3)
    disp(['image' int2str(v) '.jpg is an Object3'])
elseif (l==d4)
    disp(['image' int2str(v) '.jpg is an Object4'])
else disp(['image' int2str(v) '.jpg is an Object5'])
%...
end
```

```
   toc;
end
disp('Recalling terminated')
```

Read.m

```
%Recalls object from imagev.jpg in folder "images"
tic;
for n=1:70,
   image=imread(['C:\Path\images\image' int2str(n) '.jpg']);
   image=edge(image, 'canny'); %edge in binary
   image=~image; %function NOT so that pixels within edge worth 0
   x=0;
   inputs=zeros(2,1000); %guessing that less than 1000 pixels could define the edge
   for i=1: 64,
      for j=1: 64,
         if(image(i,j)==0)
            x=x+1;
            inputs(1,x)=i;
            inputs(2,x)=j;
         end
      end
   end
   d1=sim(Object1,inputs([1:2:x*2;2:2:x*2]));
   d2=sim(Object2,inputs([1:2:x*2;2:2:x*2]));
   d3=sim(Object3,inputs([1:2:x*2;2:2:x*2]));
   d4=sim(Object4,inputs([1:2:x*2;2:2:x*2]));
   d5=sim(Object5,inputs([1:2:x*2;2:2:x*2]));
   %...
   %object will be given only by the minimum average of distances da,de...
   %since is the same that the absolute value of the difference among
   %an output vector and zero. Zero is the value of those pixels within
```

```
    %the edge of the image
    d1=mean(d1);d2=mean(d2);d3=mean(d3);d4=mean(d4);d5=mean(d5); %...
    l=min([d1,d2,d3,d4,d5]);
    if (l==d1)
       disp(['image' int2str(n) '.jpg is an Object1'])
    elseif (l==d2)
       disp(['image' int2str(n) '.jpg is an Object2'])
    elseif (l==d3)
       disp(['image' int2str(n) '.jpg is an Object3'])
    elseif (l==d4)
       disp(['image' int2str(n) '.jpg is an Object4'])
    else disp(['image' int2str(n) '.jpg is an Object5'])
    %...
    end
end
toc;
disp('Recalling terminated')
```