Appendice

Di seguito riportiamo i listati delle procedure sviluppate in Matlab. Si tratta dei file:

- “TrovareaMT.m”
- “Trecontorni.m”
- “Confronto.m”
- “Leggidat.m”

E delle funzioni:

- “Polydistf.m”
- “Dx.m” e “Dy.m”
- “G.m”

TrovareaMT.m

```
%file trovareamt.m
%Crea una immagine binaria a partire dall'immagine RGB del manual tracing
%in ingresso usando come criterio discriminante dei particolari range nei
%numeri che indicano i colori di base (R,G,B) poi calcola l'area, espressa in
%numero di pixel, della superficie delimitata dal contorno del tracciamento
%manuale

nomeimmagine=input('Inserire il nome dell''immagine in formato jpg del
manual tracing: ','s');
rgb=imread(nomeimmagine,'jpg');
disp('Posizionarsi con il cursore sull''immagine e ritagliare l''area da
analizzare')


rgb2=imcrop(rgb);
rgbtest=rgb2;
figure, imshow (rgbtest)
dim=size(rgbtest);
y=dim(1,1);  %m righe
x=dim(1,2);  %n colonne
area=x*y;
disp('Dimensioni Area risultante dopo l''esecuzione del comando crop')

fprintf('numero righe dell''immagine (pixel): %4.1f\n',y')
fprintf('numero colonne dell''immagine (pixel): %4.1f\n',x')

fprintf('Area totale dell''immagine ritagliata(pixel): %4.1f\n',area')



livinfR=22;
livsupR=139;

livinfG=147;
livsupG=213;

livinfB=21;
livsupB=146;
```

```
contan=0;
contap=0;
for j = 1:1:y
   for i = 1:1:x

      if rgbtest(j,i,1)<=livsupR & rgbtest(j,i,1)>=livinfR &...
            rgbtest(j,i,2)<=livsupG & rgbtest(j,i,2)>=livinfG &...
            rgbtest(j,i,3)<=livsupB & rgbtest(j,i,3)>=livinfB

         contap = contap + 1;

      else
         contan = contan + 1;

         rgbtest(j,i,1)=0;
         rgbtest(j,i,2)=0;
         rgbtest(j,i,3)=0;


      end

   end
end
fprintf('totale pixel verdi perimetrali: %4.1f\n',contap)
fprintf('totale pixel oscurati: %4.1f\n',contan)

%mostra figura ripulita
figure, imshow (rgbtest)

%mostra figura binaria
bin=im2bw(rgbtest, 0.4);
figure, imshow (bin)

%mostra figura crop e figura ripulita affiancate
figure
subplot(1,2,1), subimage(rgb2)
subplot(1,2,2), subimage(rgbtest)

%mostra figura,crop, figura ripulita e figura binaria affiancate
figure
subplot(1,3,1), subimage(rgb2)
subplot(1,3,2), subimage(rgbtest)
subplot(1,3,3), subimage(bin)
```

```
%Chiudo il contorno con un segmento congiungente gli ultimi due punti

%Trovo i due ultimi punti del perimetro

disp('Selezionare sull"immagine binaria i due punti per i quali deve passare il
segmento e poi premere invio')
figure, imshow (bin)
[xcord,ycord]=ginput
xcord = round(xcord);
ycord = round(ycord);
disp(xcord)
disp(ycord)

pausa
=input('********************pausa*****pausa**********************
*****','s');


x1=xcord(1,1);
x2=xcord(2,1);
y1=ycord(1,1);
y2=ycord(2,1);

%Calcolo il coefficiente angolare m della retta passante per i due punti
%ho i punti A(x1,y1) e B(x2,y2)


m =(y2-y1)/(x2-x1);
q =(-m*x1)+y1;

%traccio il segmento che gli unisce sulla figura binaria

for i=x1:1:x2
   j=(m*i)+q;
   bin(j,i)=1;
end

%Mostro la figura con il contorno chiuso

figure, imshow (bin)


%Calcolo l'area racchiusa dal contorno scandendo i pixel dell'immagine
binaria ottenuta
```

```
ContaPixRiga=0;
ContaPix=0;
fine=0;
trovato=0;


%Trovo il primo pixel bianco dell'intera figura
disp('Sto trovando il primo pixel bianco dell''intera figura')
findprimo=0;
i=1;
j=1;
disp('Mostro i,j,x,y,findprimo')
disp(i)
disp(j)
disp(x)
disp(y)
disp(findprimo)
xprimo=0;
yprimo=0;

while (findprimo ~= 1)
   i=1;
   while (findprimo ~= 1) & (i <= x)

      if bin(j,i)==1
         findprimo=1;
         xprimo=i;
         yprimo=j;
      end

      i=i+1;
   end
   j=j+1;
end

disp('Coordinate primo pixel bianco dell''intera figura')
disp ('Coordinata X')
disp(xprimo)
disp ('Coordinata Y')
disp(yprimo)
```

```
pausa
=input('********************pausa*****pausa**********************
*****','s');


ContaPixRiga=0;
ContaPix=0;
fine=0;

j=yprimo+1;
disp('prima riga dalla quale inizio a contare i pixel dell''area')
disp(j)

%Ciclo per il calcolo dell'area all'interno della figura

while (j<y) & (fine ~= 1)

   i=1;bordo1=0;
   %trovo il primo pixel bianco del contorno all'interno della j-esima riga

   while (bin(j,i) ~= 1) & (i<x)
      i=i+1;
   end

   %bin(j,i) ora è il primo pixel bianco della riga j


   %controllo se ho una riga tutta nera, in questo caso interrompo il conteggio
   if i==x
      fine=1;
   end

   disp('Coordinata x,y del primo bianco della riga j')
   disp('riga')
   disp(j)

   disp('coordinata x ')
   disp(i)

   %pausa
=input('********************pausa*****pausa**********************
*****','s');

   % Caso semplice contorni spessi 1 solo pixel,il secondo pixel dopo quello
```

```
   % bianco è nero.
   if fine ~=1

      i2=i+1;

      while (bin(j,i2) ~= 1) & ( i2< x)

         ContaPixRiga=ContaPixRiga+1;bordo1=1;
         i2=i2+1;

      end

   end

   disp (i2)
   %pausa
=input('********************pausa*****pausa**********************
*****','s');

   % Altri possibili casi

   % Caso di bordo spesso 2 o più pixel bianchi, si fa solo se bordo1 è zero
   if (fine ~=1) & (bordo1 == 0)
      i3=i+1;

      while (bin(j,i3) == 1) & (i3 < x)

         i3=i3+1;

      end

      % fine del bordo spesso due o più pixel

      while (bin(j,i3) ~= 1) & (i3 < x)
         ContaPixRiga=ContaPixRiga+1;
         i3=i3+1;
      end

   end %chiudo l'if del caso di bordo spesso 2 o più pixel

   j=j+1;


   disp('area parziale della riga numero pixel neri all''interno del contorno')
```

```
   disp(ContaPixRiga)

   ContaPix=ContaPix+ContaPixRiga;
   ContaPixRiga=0;
   disp('Area totale calcolata fino a questa riga')
   disp(ContaPix)
   %pausa
=input('********************pausa*****pausa**********************
*****','s');

end
disp('Immagine analizzata: ')
disp(nomeimmagine)
fprintf('numero pixel all''interno dell''immagine (Area ventricolo):
%4.1f\n',ContaPix')
```

Trecontorni.m

```
%file trecontorni.m
%Crea e salva una immagine rgb a partire dall'immagine grayscale con
%sovrapposti tre contorni Rosso LevelSet, Verde Manual Tracing Oliver, Blu
%Manual Tracing Renè.
%Inoltre salva in due file .dat le funzioni distanza del manual tracing di Oliver
%e di Renaudt

clear all;
nomeimmagine=input('Inserire il nome delle immagini in formato jpg del
manual tracing (di Oliver e di Renaudt) che vuoi ridimensionare es.
103_14_1_58: ','s');
frame=input('Inserire il numero del frame ultimo numero del nome
(soggetto_parabola_blocco_frame): ','s');
nomeimmagine2=nomeimmagine;
nomeimmagineO=[nomeimmagine2,'_O'];
nomeimmagineR=[nomeimmagine2,'_R'];
disp(nomeimmagineO)
disp(nomeimmagineR)


%%%%%%%%%%%%%%%%%%%%%%%%
%   Procedura per il Manual tracing di Oliver    %
%%%%%%%%%%%%%%%%%%%%%%%%


disp('Posizionarsi con il cursore sull''immagine del Manual Tracing di Oliver e
ritagliare l''area dell''immagine eco da analizzare')
immaginemtoliver=imread(nomeimmagineO,'jpg');

%conversione da RGB a grayscale
immaginemtolivergray=rgb2gray(immaginemtoliver);

figure(1)
gray2=imcrop(immaginemtolivergray);

%figure, imshow (gray2)
dim=size(gray2);
y=dim(1,1);  %m righe
x=dim(1,2);  %n colonne
```

```
fprintf('numero righe dell''immagine del Manual Tracing di Oliver (pixel):
%4.1f\n',y')
fprintf('numero colonne dell''immagine del Manual Tracing di Oliver (pixel):
%4.1f\n',x')

%nrighe=input('Inserire il numero delle righe del nuovo formato: ');
%ncolonne=input('Inserire il numero delle colonne del nuovo formato: ');

%imridimensionataoliver=imresize(gray2,[nrighe ncolonne]);
imridimensionataoliver=imresize(gray2,[267 384]);

immagineco=imread(nomeimmagineO,'bmp');



%Calcolo la funzione distanza con segno per l'immagine del manual tracing
ritagliata

figure(2),imshow (immaginemtoliver)
figure(3), imshow (imridimensionataoliver)
A=polydistf(imridimensionataoliver);
disp(A)
pausa
=input('********************pausa*****pausa**********************
*****','s')

%Salvataggio del file dat contenente la funzione distanza del manual tracing
di Oliver

patholiver='C:\SamueleMatlab\FileDatOliver\'
nomefileO=[nomeimmagineO,'.dat']
percorsoliver=[patholiver,nomefileO];
id=fopen(percorsoliver,'w');
disp (id)
elementiscritti=fwrite(id,A,'float32');
disp('elementi scritti')
disp(elementiscritti)
fclose (id);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Procedura per il Manual tracing di Renaudt      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp('Posizionarsi con il cursore sull''immagine del Manual Tracing di Renè e
ritagliare l''area dell''immagine eco da analizzare')
figure(4),immaginemtrene=imread(nomeimmagineR,'jpg');

%conversione da RGB a grayscale
immaginemtrenegray=rgb2gray(immaginemtrene);

gray3=imcrop(immaginemtrenegray);

dim=size(gray3);
y=dim(1,1);  %m righe
x=dim(1,2);  %n colonne

fprintf('numero righe dell''immagine del Manual Tracing di Renè (pixel):
%4.1f\n',y')
fprintf('numero colonne dell''immagine del Manual Tracing di Renè(pixel):
%4.1f\n',x')

%nrighe=input('Inserire il numero delle righe del nuovo formato: ');
%ncolonne=input('Inserire il numero delle colonne del nuovo formato: ');

imridimensionatarene=imresize(gray3,[267 384]);


%Calcolo la funzione distanza con segno per l'immagine del manual tracing
ritagliata


figure(5),imshow (immaginemtrene)
figure(6), imshow (imridimensionatarene)
C=polydistf(imridimensionatarene);
disp(C)
pausa
=input('******************pausa*****pausa********************
*****','s')

%Salvataggio del file dat contenente la funzione distanza del manual tracing
di Renè

pathrene='C:\SamueleMatlab\FileDatRene\'
```

```
nomefileR=[nomeimmagineR,'.dat']
percorsorene=[pathrene,nomefileR];

id2=fopen(percorsorene,'w');
disp (id2)
elementiscritti2=fwrite(id2,C,'float32');
disp('elementi scritti')
disp(elementiscritti2)
fclose (id2);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Sezione apertura file .dat e tracciamento dei contorni sull'immagine eco %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

id=fopen(percorsoliver,'r');
[B,elementiletti]=fread(id,[267,384],'float32');

fclose(id);
%disp(B)
disp('elementi letti')
disp(elementiletti)

figure(7)

subplot(1,1,1), subimage(immagineco)
hold on
contour(B,[0,0],'g')

id2=fopen(percorsorene,'r');
[C,elementiletti2]=fread(id2,[267,384],'float32');
fclose(id2);
disp('elementi letti')
disp(elementiletti2)
hold on
contour(C,[0,0],'b')

pathlevelset='C:\samueleMatlab\DatLevelSet\'
nomefilelevelset1=['distvol',frame]
nomefilelevelset=[nomefilelevelset1,'.dat']
percorsolevelset=[pathlevelset,nomefilelevelset];

id3=fopen(percorsolevelset,'r');
[D,elementiletti3]=fread(id3,[267,384],'float32');
```

```
fclose(id3);
hold on
contour(D,[0,0],'r')
hold off

catturata=getframe;
img=frame2im(catturata);

%colormap(gray);

figure(8),imshow(img)



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Sezione salvataggio file contenente i tre contorni sovrapp. all'immagine eco
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

percorso ='C:\samueleMatlab\Im3Contorni\';
nome1=[nomeimmagine,'TreC']
nome=[percorso,nome1];
disp (nome)

imwrite(img,nome,'bmp');


immaginesalvata=imread(nome,'bmp');
figure(9),imshow(immaginesalvata)
```

Confronto.m

```
%confronto.m
%questa procedura calcola la funzione distanza con segno
%tra i contorni endocardici stimati dai manual tracing
%di Oliver e Renaudt e quelli ottenuti dalla procedura
%di segmentazione, e fornisce in uscita l'errore massimo e
%l'errore medio

clear all;

soggetto=input('introdurre il numero del soggetto: ');
parabola=input('introdurre il numero della immagine: ');
blocco=input('introdurre il numero del frame: ');
frame=input('introdurre il n. della sezione: ');

i=int2str(soggetto);
p=int2str(parabola);
b=int2str(blocco);
f=int2str(frame);

%apertuta e display dato originale
fil=strcat(i,'_',p,'_',b,'_',f,'.bmp');
S1=imread(fil,'bmp');
S1=double(S1)./255;
figure(1);
colormap(gray);
image(S1.*70,'EraseMode','None'); axis image;
figure(2);
colormap(gray);
image(S1.*70,'EraseMode','None'); axis image;
figure(3);
colormap(gray);
image(S1.*70,'EraseMode','None'); axis image;


%sovrapposizione contorno LS in giallo
figure(1); hold on;
filedist1=strcat('distvol_',f,'.dat');
fdist1=fopen(filedist1,'r');
D1=fread(fdist1,[267,384],'float');
```

```
contour(D1,[0,0],'r');
drawnow;
%salvataggio di una bmp con dato originale e contorno LS sovrapposto
imcont1=strcat('sogg_',i,'_',f,'.bmp');
g=getframe;
[img,gray]=frame2im(g);
image(img);
imwrite(img,imcont1,'bmp');

hold off;
%apertura manual tracing Oliver e sovrapposizione contorno in verde
figure(2); hold on;
filedist2=strcat('distvol_',f,'_O.dat');
fdist2=fopen(filedist2,'r');
D2=fread(fdist2,[384,267],'float');
contour(D2,[0,0],'g');
drawnow;
%salvataggio di una bmp con dato originale e contorno Oliver sovrapposto
imcont2=strcat('sogg_',i,'_',f,'_0.bmp');
g2=getframe;
[img2,gray]=frame2im(g2);
image(img2);
imwrite(img2,imcont2,'bmp');

hold off;
%apertura manual tracing Renaudt e sovrapposizione contorno in blu
figure(3); hold on;
filedist3=strcat('distvol_',f,'_R.dat');
fdist3=fopen(filedist3,'r');
D3=fread(fdist3,[384,267],'float');
contour(D3,[0,0],'b');
drawnow;
%salvataggio di una bmp con dato originale e contorno Renaudt sovrapposto
imcont3=strcat('sogg_',i,'_',f,'_R.bmp');
g3=getframe;
[img3,gray]=frame2im(g3);
image(img3);
imwrite(img3,imcont3,'bmp');

fclose(fdist1);
fclose(fdist2);
fclose(fdist3);

pause;
```

```
clear all;
% Creazione della MAPPA DELLE DISTANZE
im=imread('sogg_102_151.bmp');
imr=im(:,:,1); % matrice contenente l'intensità del colore rosso
img=im(:,:,2); % matrice contenente l'intensità del colore verde
imb=im(:,:,3); % matrice contenente l'intensità del colore blu

imr=double(imr)./255; % trasformazione in double e normalizzazione
img=double(img)./255; %        "          "              "
imb=double(imb)./255; %        "          "          "

figure(4); colormap(gray);
image(imr.*70,'EraseMode','None'); axis image; drawnow; hold on;

mt=(imr==1).*(img==0).*(imb==0); % mt è una matrice con degli 1 in
corrispondenza  del
                        % manual tracing (il manual tracing deve essere rosso)
[nr,nc]=size(mt);

for i=1:nr                  % Creazione di una matrice mt2 in cui ci sono degli
zeri in
  for j=1:nc                % corrispondenza del manual tracing, degli 1 all'
interno e
    if(mt(i,j)==1)           % dei -1 all' esterno
      mt2(i,j)=0;
    else
      if(sum(mt(i,[1:(j-1)]))~=0
        if(sum(mt(i,[(j+1):nc]))~=0
          if(sum(mt([1:(i-1)],j))~=0
            if(sum(mt([(i+1):nr],j)))~=0
              mt2(i,j)=1;
            else
              mt2(i,j)=-1;
            end
          else
            mt2(i,j)=-1;
          end
        else
          mt2(i,j)=-1;
        end
      else
        mt2(i,j)=-1;
      end
```

```
    end
  end
end
mt3=mt2; % Salvataggio della matrice mt2

mappa_distanze=999.*ones(nr,nc);
mappa_distanze=mappa_distanze.*(mt==0);

for k=1:30

  appoggio=zeros(nr,nc);
  mt2 = mt2 + G(mt2);
  for j=1:nc
    for i=1:(nr-1)
      if(mt2(i,j)*mt2(i+1,j))<0
        if(abs(mt2(i,j))<(abs(mt2(i+1,j))))
          appoggio(i,j)=k;
        else
          appoggio(i+1,j)=k;
        end
      end
    end
  end

  for i=1:nr
    for j=1:(nc-1)
      if(mt2(i,j)*mt2(i,j+1))<0
        if(abs(mt2(i,j))<abs(mt2(i,j+1)))
          appoggio(i,j)=k;
        else
          appoggio(i,j+1)=k;
        end
      end
    end
  end
  mappa_distanze=mappa_distanze.*(appoggio==0) + appoggio;
end

mt2=mt3;

for k=1:30

  appoggio=zeros(nr,nc);
  mt2 = mt2 - G(mt2);
```

```
  for j=1:nc
    for i=1:(nr-1)
      if(mt2(i,j)*mt2(i+1,j))<0
        if(abs(mt2(i,j))<(abs(mt2(i+1,j))))
          appoggio(i,j)=k;
        else
          appoggio(i+1,j)=k;
        end
      end
    end
  end

  for i=1:nr
    for j=1:(nc-1)
      if(mt2(i,j)*mt2(i,j+1))<0
        if(abs(mt2(i,j))<(abs(mt2(i,j+1))))
          appoggio(i,j)=k;
        else
          appoggio(i,j+1)=k;
        end
      end
    end
  end
  mappa_distanze=mappa_distanze.*(appoggio==0) + appoggio;

end


% Acquisizione del contorno stimato
clear im
im=imread('sogg_102_151_O.bmp');
imr=im(:,:,1); % matrice contenente l'intensità del colore rosso
img=im(:,:,2); % matrice contenente l'intensità del colore verde
imb=im(:,:,3); % matrice contenente l'intensità del colore blu

imr=double(imr)./255;
img=double(img)./255;
imb=double(imb)./255;

stim=(imr==0).*(img==1).*(imb==0); % mt è una matrice con degli 1 in
corrispondenza
                          % del manual tracing

  npixel_contorno1=sum(sum(stim));
```

```
  confr1=mappa_distanze.*stim;
  max_errore_O=max(max(confr1));
  errore_medio_O=sum(sum(confr1))/npixel_contorno1;

clear im
im=imread('sogg_102_151_R.bmp');
imr=im(:,:,1); % matrice contenente l'intensità del colore rosso
img=im(:,:,2); % matrice contenente l'intensità del colore verde
imb=im(:,:,3); % matrice contenente l'intensità del colore blu

imr=double(imr)./255;
img=double(img)./255;
imb=double(imb)./255;

stim=(imr==0).*(img==0).*(imb==1); % mt è una matrice con degli 1 in
corrispondenza
                        % del manual tracing


  npixel_contorno2=sum(sum(stim));
  confr2=mappa_distanze.*stim;
  max_errore_R=max(max(confr2));
  errore_medio_R=sum(sum(confr2))/npixel_contorno2;

% Visualizzazione dei risultati

disp(['Oliver:       errore massimo = ' num2str(max_errore_O)]);
disp(['              errore medio   = ' num2str(errore_medio_O)]);
disp(['Renaudt:  errore massimo = ' num2str(max_errore_R)]);
disp(['          errore medio   = ' num2str(errore_medio_R)]);
```

Leggidat.m

```
%file Leggidat2.m
%Questo script legge e visualizza (una riga alla volta) i file dat di Oliver,
Renaudt, e del level set
%

clear;
nomefile=input('Inserire il nome del file dat da leggere: ','s');
frame=input('Inserire il numero del frame ultimo numero del nome
soggetto_parabola_blocco_frame: ','s');

nomefile2=nomefile;
nomeO=[nomefile2,'_O'];
nomeR=[nomefile2,'_R'];
disp(nomeO)
disp(nomeR)


%%%%%%%%%%%%%
%       Percorsi MT          %
%%%%%%%%%%%%%

patholiver='C:\SamueleMatlab\FileDatOliver\'
nomefileO=[nomeO,'.dat']
percorsoliver=[patholiver,nomefileO];

pathrene='C:\SamueleMatlab\FileDatRene\'
nomefileR=[nomeR,'.dat']
percorsorene=[pathrene,nomefileR];

pathlevelset='C:\samueleMatlab\DatLevelSet\'
nomefilelevelset1=['distvol',frame]
nomefilelevelset=[nomefilelevelset1,'.dat']
percorsolevelset=[pathlevelset,nomefilelevelset];


nometipo=input('Inserire il nome del tipo di file dat da leggere (O=Oliver,
R=Renè, L=Level Set) : ','s');

switch  nometipo
   case 'O'
```

```
      percorso=percorsoliver;
   case 'R'
      percorso=percorsorene;
   case 'L'
      percorso=percorsolevelset;
   otherwise
      disp('Inserimento non valido')
 end


%%%%%%%%%%%%%%%%%%%%
%  Sezione apertura file .dat scelto    %
%%%%%%%%%%%%%%%%%%%%



id2=fopen(percorso,'r');
[C,elementiletti2]=fread(id2,[267,384],'float32');
%pausa
=input('********************pausa*****pausa**********************
*****','s')
fclose(id2);
%disp(C)
for j=1:267
   for i=1:384
      fprintf('riga n. %4.2f ',j)
      fprintf(' colonna n. %4.2f\n ',i)

      disp (C(j,i))

   end
   pausa
=input('********************pausa*****pausa**********************
*****','s')

end
disp('elementi letti')
disp(elementiletti2)


id3=fopen(percorsolevelset,'r');
[D,elementiletti3]=fread(id3,[267,384],'float32');
fclose(id3);
```

Polydistf.m

```
function D=polydistf(I);
[ny nx]=size(I);


% input the polyline for the manual segmentation
[x,y]=ginput;
np=size(x);
xn=[x ; x(1)];
yn=[y ; y(1)];
%plot(xn,yn);
%drawnow;

% compute the distance function from the polyline
x=[x; x(1); x(2)];
y=[y; y(1); y(2)];
[X,Y]=meshgrid(1:nx,1:ny);
D= X.*0+2000;
colormap(gray);

for i=2:size(x)-1
        % segment
        x0=x(i); x1=x(i+1); xm1=x(i-1); y0=y(i); y1=y(i+1); ym1=y(i-1); %
previous and next segment
        cteta=(x1-x0)/sqrt( (x1-x0).^2+(y1-y0).^2); % director cosine
        steta=-(y1-y0)/sqrt( (x1-x0).^2+(y1-y0).^2); % director sine
        A=(X-x0).*steta+(Y-y0).*cteta; % analitical distance from the line
        B=( (X-x0).*cteta-(Y-y0).*steta >0); % upper constraint for the
segment
        C=( (X-x1).*cteta-(Y-y1).*steta <0); % lower contraint for the segment
        I=A.*(B.*C)+(1-B.*C).*2000; % distance from the segment
        D=D.*(abs(I)>abs(D))+I.*(abs(I)<=abs(D)); % min distance
        %corner
   s=(-1)*((x0-xm1)*(y1-y0)-(x1-x0)*(y0-ym1)); % cross product
        I=sqrt((X-x0).^2+(Y-y0).^2).*sign(s) ; % positive or negative cones
        D=D.*(abs(I)>abs(D))+I.*(abs(I)<=abs(D)); % min distance

end
```

Dx.m

```
%%%%%%%%%%%%%
%  Use:  f = Dx(Matrix)    %
%%%%%%%%%%%%%

function f = Dx(Mat)
[m n] = size(Mat);
f = (Mat([2:m m],1:n) - Mat([1 1:m-1],1:n))./2;
```

Dy.m

```
%%%%%%%%%%%%%
%  Use:  f = Dy(Matrix)  %
%%%%%%%%%%%%%
function f = Dy(Mat)
[m n] = size(Mat);
f = (Mat(1:m,[2:n n]) - Mat(1:m,[1 1:n-1]))./2;
```

G.m

```
%%%%%%%%%%%%%
%  Use:  f = G(Matrix)      %
%%%%%%%%%%%%%
function f = G(Mat)
f = sqrt (Dx(Mat).^2 + Dy(Mat).^2);
```