UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DEGRAU

Grau en Enginyeria Electrónica i Automatització Industrial

SISTEMA ROBÒTIC DE CLASSIFICACIÓ D'OBJECTES EN PERSPECTIVA AMB VISIÓ PER COMPUTADOR

Annexos

Autor: Eudald Ballescà Casas
Director: Antoni Grau Saldes

Convocatòria: Maig 2018

Índex

ANNEX A CODI ____ **3**

1.1. Main ... 3
1.2. TrobarTagColor ... 8
1.3. AlgEtiquetatge ... 9
1.4. AlgTortPapert ... 10
1.5. AlgEtiquetatge2 ... 11
1.6. Mostrar_error ... 12
1.7. AlgCercles ... 13
1.8. CalculCentre ... 14
1.9. Eix ... 15
1.10. AlgCantonades ... 16
1.11. AbatirPlano ... 18
1.12. AlgBase ... 19
1.13. CalculCentre ... 20
1.14. AlgRelacioPixelRealitat ... 21
1.15. AlgOrientacioA ... 22
1.16. FuncioAltura ... 23
1.17. AlgLimitZonaTreball ... 24
1.18. AlgTransformacio ... 25
1.19. TrobarPeces ... 27
1.20. CalculArea ... 28
1.21. FuncioMarges ... 29
1.22. AlgVertex ... 30
1.23. AlgOrientacioB ... 32
1.24. RotacioPosicions ... 34
1.25. AlgIdentificacio ... 36
1.26. FuncioIdentificacioImatge ... 38

ANEX B DADES ____ **39**

Graella 1 Àrees obtingudes(mm^2) _____ 40

Graella 1Mitjana (mm^2) àrees obtingudes Graella 1 _____ 40

Graella 1Error absolut respecte Graella 2 amb les dades de Graella 1(mm^2) _____ 41

Graella 1Error relatiu (%) respecte Graella 2 amb les dades de Graella 1 _____ 41

Graella 1Valors en les primeres 23 iteracions_____ 42

Graella 1Valors en les ultimes 17 iteracions _____ 43

Annex A Codi

1.1. Main

```
close all
clear all


% Parametres
%-------------------------------------------------------
IMmin=100;                    %   Valors a descartar que es consideraran
IMmax=1000;                  % entorn
Altura_peces= 6.4/2;         %   Relació entre altura de les peces i la
                             % amplada dels tags
Profunditat_zona=21.7/2;     %   Relació entre la separació dels tags i la
                             % amplada dels tags
Amplada_zona=36/2;           %   Relació entre la separació dels tags i la
                             % amplada dels tags

error = 0;   % error general; mirar llista d'errors
% % ------------------------------------------------
% Base de dades de peces

Base_Dades_Peces{1,1}= [1 1100 1400];
Base_Dades_Peces{2,1}= [2 750 900];
Base_Dades_Peces{3,1}= [3 600 800];
Base_Dades_Peces{4,1}= [4 1100 1500];

Base_Dades_Peces{6,1}= [6 800 900];
Base_Dades_Peces{7,1}= [7 1050 1300];

Base_Dades_Peces{9,1}= [9 700 820];

Base_Dades_Peces{11,1}=[11 350 500];
Base_Dades_Peces{12,1}=[12 350 500];
Base_Dades_Peces{13,1}=[13 500 800];
Base_Dades_Peces{14,1}=[14 350 600];

Base_Dades_Peces{1,2}= 'Hexagon';
Base_Dades_Peces{2,2}= 'Pentagon';
Base_Dades_Peces{3,2}= 'Quadrat petit';
Base_Dades_Peces{4,2}= 'Quadrat mitja';
Base_Dades_Peces{5,2}= 'Quadrat gran';
Base_Dades_Peces{6,2}= 'Rectangle petit';
Base_Dades_Peces{7,2}= 'Rectangle mitja';
Base_Dades_Peces{8,2}= 'Rectangle gran';
Base_Dades_Peces{9,2}= 'Rombe petit';
Base_Dades_Peces{10,2}='Rombe gran';
Base_Dades_Peces{11,2}='Triangle equilater';
Base_Dades_Peces{12,2}='Triangle rectangle petit isosceles';
Base_Dades_Peces{13,2}='Triangle rectangle gran isosceles';
Base_Dades_Peces{14,2}='Triangle rectangle escalar';
Base_Dades_Peces{15,2}='No identificada';

%Base_Dades_Precanvi=cat(1, Base_Dades_Peces{:,1});
```

```
%-------------------------------------------------------------
%                              Main
%-------------------------------------------------------------

% % Per  treballar en imatges en temps real
% ---------------------------------------------
cameraParams = importdata('cameraParams.mat');
vidobj = videoinput('winvideo',1,'RGB24_640x480');
preview(vidobj);
input('\n Si la camera esta ben centrada pica ENTER per acabar ');
closepreview(vidobj);
t = cputime;
im = getsnapshot(vidobj);
imshow(im)

im = undistortImage(im, cameraParams);

inclinacio=0;
for i=1:2

im1 = imrotate(im,inclinacio,'bilinear','loose');
DimIM = size(im1);
[IM10]= TrobarTagColor(im1,DimIM);
se = strel('disk', 2);
IM10 = imopen(IM10,se);
[IM11,k]= AlgEtiquetatge(IM10,DimIM);
[IM11,Qua,error]= AlgEtiquetatge2(IM11,DimIM,k);
if error ~=0
    mostra_error(error)
    return
end

%   Un cop es te la imatge separada per conjunts es triarà quins d'aquest
% son útils per a la orientació de la imatge, traient tant petites taques
% com les grans taques que es  formen als afores de la plataforma.
%   Amb kTag s'acabarà obtenint els valors assignats als ARTag que
% s'assignaran a Elements_IM12 per tal de agilitzar futures operacions.

karea=zeros(1,k+1);
for i=2:DimIM(1)+1
    for l=2:DimIM(2)+1
        karea(1,IM11(i,l))=karea(1,IM11(i,l))+1;
    end
end

IM12=IM11;
for i=2:DimIM(1)+1
    for l=2:DimIM(2)+1
        if karea(1,IM11(i,l)) < IMmin ||karea(1,IM11(i,l)) > IMmax
            IM12(i,l)=1;
        end
    end
end


k=1;
Area_IM12=zeros(1,k);
Elements_IM12=Area_IM12;
Qua12=Area_IM12;
for i=1:length(karea)
    if karea(i)~=0
        Elements_IM12(k)=i;
```

```
        Area_IM12(k)=karea(i);
        Qua12(k)=Qua(i);
        k=k+1;
    end
end

[Elements_TAG,Cercle,Area_Cercle,Quadrat,error]...
    = AlgCercles(IM12,Elements_IM12,Area_IM12,DimIM,Qua12);

if error ~=0
    mostra_error(error)
    return
end
[PosY, PosX]= CalculCentre (IM12, Elements_TAG, DimIM);

[Angles,DisMax] = Eix (IM12, Elements_TAG, DimIM, PosY, PosX);

%   Un cop l'algoritme te dades suficients per saber on estan els ARTag
% seleccionara quin es mes proxim a cada cantonada

[Punter_Proxim, Id_Proxim]=AlgCantonades(Elements_TAG,PosX,PosY,DimIM);
[X_fuga, Y_fuga]= AlgFuga(PosX, PosY,Id_Proxim, Elements_TAG,DimIM);

for i=1:4
    if DisMax(i,4)>=DisMax(:,4)
        Pendent_Pla=(sind(Angles(i))*DisMax(i,4)*Altura_peces)/ ...
            (PosY(i)-Y_fuga);

    end
end

[IM50]= AbatirPlano(IM11, Elements_TAG, DimIM, Pendent_Pla, Y_fuga);

[Y_base]= AlgBase (IM50, Elements_TAG, DimIM);

[PosY_Elevat]= CalculCentre (IM50, Elements_TAG, DimIM);
%   Com que nomes s'han desplaçat cap una direcció els punts segueixen
% mantenint l'ordre trobat en la imatge plana.

[RelacioPixelRealitat]=AlgRelacioPixelRealitat(IM50,DimIM,...
    Elements_IM12,Elements_TAG, Cercle, Quadrat, X_fuga, Y_fuga,...
    Y_base,Id_Proxim,Punter_Proxim,PosY_Elevat);

desbX = (PosX(Id_Proxim(1)==Elements_TAG)+...
    PosX(Id_Proxim(2)==Elements_TAG))/2-...
    (PosX(Id_Proxim(3)==Elements_TAG)+...
    PosX(Id_Proxim(4)==Elements_TAG))/2;
desbY =(PosY_Elevat(Id_Proxim(1)==Elements_TAG)+...
    PosY_Elevat(Id_Proxim(2)==Elements_TAG))/2-...
    (PosY_Elevat(Id_Proxim(3)==Elements_TAG)+...
    PosY_Elevat(Id_Proxim(4)==Elements_TAG))/2;
teta= atand(-desbX/desbY);

[D,direccio]= AlgOrientacioA(Id_Proxim,Elements_IM12, Quadrat,...
    Amplada_zona,Profunditat_zona,X_fuga,Y_fuga,Y_base,PosX,...
    PosY_Elevat,Punter_Proxim);

[X_altura, X_alturae,X_altura2, X_alturae2]= FuncioAltura (PosX,...
    PosY_Elevat,Id_Proxim, Elements_TAG, X_fuga, Y_fuga, Y_base, D);
```

```
if X_altura ~=0 && X_altura2 ~=0
    inclinacio=(90-acosd(abs(X_altura2-X_altura)/(X_altura2-X_fuga)))...
        *(abs(X_altura2-X_altura)/(X_altura2-X_altura));
else
    if X_altura ==0
        X_altura=X_altura2;
        X_alturae=X_alturae2;
    else
        X_altura2=X_altura;
        X_alturae2=X_alturae;
    end
    fprintf("3 o més tag han quedat situats al mateix canto de la columna")
    fprintf(" X_fuga, aixo fa que la correcció de l'inclinació respecte ")
    fprintf("l'horitzo no sigui possible i per tant incrementa l'error en")
    fprintf("les coordenades de les figures ")
    break

end
end

[esquerra,dreta,superior]= AlgLimitZonaTreball (IM50,Elements_TAG,...
    DimIM,X_fuga,Y_fuga,Y_base,X_altura2,X_alturae2);

%   Un cop tenim els punts necessaris per coneixe en quina posicio esta la
% camera podem passar a transformar la imatge (de forma inversa per evitar
% els pixels buits).
[IM51]= AlgTransformacio(X_altura2,X_alturae2,...
    X_fuga,Y_fuga,Y_base,IM50,superior, esquerra, dreta);

im2 = imrotate(im,inclinacio,'bilinear','crop');
[IM140]= AlgTransformacio(X_altura2,X_alturae2,...
    X_fuga,Y_fuga,Y_base,im2,superior, esquerra, dreta);
figure, imshow(IM140) % En Color
title('color');
DimIM140=size(IM140);

[IM141]= TrobarPeces(IM140,DimIM140);
se = strel('disk', 2);
IM142 = imclose(IM141,se);

[IM150,k]= AlgEtiquetatge(IM142,DimIM140);

[Area_Figures, Id_Figures, IM150]= CalculArea(IM150,DimIM140,k, ...
    Area_Cercle);

[PosY_F, PosX_F]= CalculCentre (IM150, Id_Figures, DimIM140);



[IM157,Perimetre_F]=FuncioMarges(IM150,DimIM140,Id_Figures);

[Info_Figures]= AlgVertex (IM157, Id_Figures, PosX_F, PosY_F,...
    Perimetre_F, DimIM140);



X_fuga140=DimIM140(2)*0.500001;

[teta20,Id_rotacio,PosX2,PosY2]=AlgOrientacioB(X_fuga140, Y_fuga,...
```

```
    Y_base, X_altura2, X_alturae2, PosX, PosY_Elevat, Elements_IM12,...
    Id_Proxim, Quadrat,Elements_TAG,esquerra,superior);

[PosX_Final,PosY_Final,RelacioPixelRealitat]=RotacioPosicions (PosX2,...
    PosY2, PosX_F,PosY_F,teta20,direccio,Id_rotacio,Elements_TAG, ...
    Profunditat_zona,Amplada_zona,RelacioPixelRealitat,Punter_Proxim);

[nomfigura]= AlgIdentificacio(Info_Figures,Area_Figures,cat(1,...
    Base_Dades_Peces{:,1}), RelacioPixelRealitat);

t1=cputime-t;

[IM200]=FuncioIdentificacioImatge(IM140, PosX_F, PosY_F, ...
    nomfigura,PosX_Final,PosY_Final,Base_Dades_Peces);
figure,imshow(IM200)
```

1.2. TrobarTagColor

```
function [IM_Out]= TrobarTagColor(IM_In,DimIM)
IM_Out=zeros(DimIM(1),DimIM(2));
for c=1:DimIM(2)
    for f=1:DimIM(1)
        if (IM_In(f,c,2)>(IM_In(f,c,1)+5)&& IM_In(f,c,2)>60 &&...
                IM_In(f,c,2)>(IM_In(f,c,3)+20) && IM_In(f,c,2)<175 )
            IM_Out(f,c)=0;
        else
            IM_Out(f,c)=1;
        end
    end

end
end
```

1.3. AlgEtiquetatge

```
function [IM_Out,k]= AlgEtiquetatge(IM_In,DimIM)
k=2;
%   Es crea un  marc al voltant de la imatge per tal d'evitar errors al
% consultar els veïns en els extrems i per a que l'algoritme de la
tortuga
% de Papert no tingui problemes.
IM_Out=ones(DimIM(1)+2,DimIM(2)+2);
for f=2:DimIM(1)+1
    for c=2:DimIM(2)+1
        IM_Out(f,c)=IM_In(f-1,c-1);
    end
end
%   El següent for inicia el procés d'etiquetatge de la imatge. El procés
% comença consultant si en la posició que esta te una peça sense
% identificar(anomenarem peça tant els objectes com a les imatges
impreses)
% o no, en el cas de tenir-la es consultarà si (de ser possible) en les
% posicions anteriors (en les duges direccions) hi ha una peça
etiquetada,
% ja que compartirien el mateix identificador, en el cas de no tenir cap
% píxel veí etiquetat voldrà dir que s'ha trobat amb una nova peça i per
% tant s'iniciarà l'algoritme tortuga de Papert modificat
for f=1:DimIM(1)+2
    for c=1:DimIM(2)+2
        if IM_Out(f,c)==0
            if f~=1
                if IM_Out(f-1,c)~=1
                    IM_Out(f,c)=IM_Out(f-1,c);
                end
            end
            if (c~=1) && (IM_Out(f,c)==0)
                if (IM_Out(f,c-1)~=1)
                    IM_Out(f,c)=IM_Out(f,c-1);
                end
            end
            if IM_Out(f,c)==0
                [k,IM_Out] = AlgTortPapert(f,c,k,IM_Out);
            end %fi comprovacio veins
        end
    end
end


end
```

1.4. AlgTortPapert

```
function [k,IM10] = AlgTortPapert(f,c,k,IM10)
%   Algoritme per a resseguir i marcar el contorn d'un objecte.
%   La funció serà cridada quan durant la execució ens trobem amb un
% objecte encara no diferenciat (marcat el bit com a 1) i es modificarà
tot
% el seu contorn de tal manera que en les següents ocasions que ens el
% trobem sapiguem de quin objecte es tracta evitant així tenir que fer
% varies passades de la funció de etiquetar, ja que amb una de sola
% quedaran tots els components que formaran la imatge amb un sol valor
% (cada un el seu).
%   L'algoritme basa els seus moviments en la primera fila de la matriu
% MATtor que indica quin es el següent moviment de l'algoritme de Papert
% interpretat en primera columna moviment de l'eix Y segona columna
% moviment de l'eix X. El moviment del següent pas es modificarà rotant
la
% matriu en comptes de modificar una variable enforma de punter.
%   L'algoritme es repetirà fins que no arribem a la posició inicial
% indicada per fi i ci.
MATtor=[0 1; 1 0; 0 -1; -1 0];  %constants i variables tortuga
fi= f; ci= c-1; fa=f; ca=c; % <---- es pot obviar
while(fi~=fa||ci~=ca)% inici algoritme tortuga
   if IM10(fa,ca)~=1% es contorn
       MATtor= circshift(MATtor,1);
       IM10(fa,ca)=k;
   else
       MATtor= circshift(MATtor,-1);
   end
   fa=fa+MATtor(1,1);
   ca=ca+MATtor(1,2);

end
k=k+1;
end
```

1.5. AlgEtiquetatge2

```
function [IM_Out,aux,error]= AlgEtiquetatge2(IM_In,DimIM,k)
aux=zeros(1,k+1);
IM_Out=IM_In;
IM_Out2=0;

IM_Out2=IM_Out;
for f=DimIM(1)+1:-1:1
    for c=DimIM(2)+1:-1:1
        if  IM_Out(f,c)~=1
            if IM_Out(f+1,c)~=1 && IM_Out(f+1,c)~=IM_Out(f,c)
                aux(1,IM_Out(f+1,c))=1;
                aux2=IM_Out(f+1,c);
                aux3=IM_Out(f,c);
                for f2=DimIM(1)+1:-1:1
                    for c2=DimIM(2)+1:-1:1
                        if IM_Out(f2,c2)==aux3
                            IM_Out(f2,c2)=aux2;
                       end
                    end
                end
            elseif IM_Out(f,c+1)~=1 && IM_Out(f,c+1)~=IM_Out(f,c)
                aux(1,IM_Out(f,c+1))=1;
                aux2=IM_Out(f,c+1);
                aux3=IM_Out(f,c);
                for f2=DimIM(1)+1:-1:1
                    for c2=DimIM(2)+1:-1:1
                        if IM_Out(f2,c2)==aux3
                            IM_Out(f2,c2)=aux2;
                       end
                    end
                end
            end
        end
    end
end
if aux==0
    error=2;
else
    error=0;
end

end
```

1.6. Mostrar_error

```
function mostra_error(error)

switch(error)
    case 1
        fprintf('Error en AlgCercles, no s ha trobat els tags\n\n');
    case 2
        fprintf('Error en la detecció del quadrat')


end
end
```

1.7. AlgCercles

```
function [Elements_Out, Cercle,Area_Cercle,Qua_ref,error]...
    = AlgCercles(IM_In, Elements_In, Area, DimIM,Qua)
%   Per tal de trovar el cercles que identifiquen els ARTag i la seva
% orientació utilitzarem la relació entre perimetre i area, aillant de
les
% dugues equacións el radi i comparan entre aquest. En el cas de un
cercle
% perfecte el valor resultant tindria que ser 1 pero no en la resta de
% casos, i com més s'allunyi d'aquest valor menys possiblitats te de ser
un
% cercle
error=0;
Qua_ref=0;
Area_Cercle=0;
Perimetre=zeros(1,length(Elements_In));
Elements_Out=Perimetre;
Cercle=Elements_Out;
Cerclecitat=Cercle;
for i=2: DimIM(1)+1 %   Calculem els perimetres
    for j=2: DimIM(2)+1
        if IM_In(i,j)~=1
            if IM_In(i-1,j)==1 || IM_In(i+1,j)==1 || IM_In(i,j-1)==1 ...
                    || IM_In(i,j+1)==1
               punter = find(Elements_In==IM_In(i,j));
               Perimetre(punter)=Perimetre(punter) + 1;
            end
        end
    end
end
for i=1:length(Elements_In) %   Fem la relació perimetre area
    Cerclecitat(i)=sqrt(Area(i)/pi)/(Perimetre(i)/(2*pi));
end

for i=1:length(Elements_In)
    if (Cerclecitat(i)<1.2 && Cerclecitat(i)>0.8) || Qua(i)==1
        Cercle(i)=1;
    else
        Cercle(i)=0;
    end
end
CerclecitatAux=Cerclecitat(Cercle==1);

if length(CerclecitatAux)~=4
    error=1;
    return
end

Qua_ref=Qua;
 Elements_Out=Elements_In(Cercle==1);
 Area_Cercle=Area(Cercle==1);
 end
```

1.8. CalculCentre

```
function [PosY, PosX]= CalculCentre (IM_In, ID_Tag, DimIM)
Area=zeros(1,length(ID_Tag));
PosX=Area;
PosY=PosX;
for i=2:DimIM(1)+1
    for u=2:DimIM(2)+1
        Area(ID_Tag==IM_In(i,u))=Area(ID_Tag==IM_In(i,u))+1;
        PosX(ID_Tag==IM_In(i,u))=PosX(ID_Tag==IM_In(i,u))+u;
        PosY(ID_Tag==IM_In(i,u))=PosY(ID_Tag==IM_In(i,u))+i;
    end
end
PosX=PosX./Area;
PosY=PosY./Area;

end
```

1.9. Eix

```
function [Angles,DisMax] = Eix(IM_In, Elements_In, DimIM, PosY, PosX)
DisMax=[Elements_In' zeros(4,3)];
DisMin=[Elements_In' ones(4,3)*DimIM(2)];

for i=2: DimIM(1)+1 %   busquem els marges
    for j=2: DimIM(2)+1
        for k=1:length(Elements_In)
            if IM_In(i,j)==Elements_In(k)
                if IM_In(i-1,j)==1 || IM_In(i,j-1)==1 ||IM_In(i+1,j)==1
...
                        ||IM_In(i,j+1)==1
                    D= sqrt((PosY(k)-i)^2+(PosX(k)-j)^2);
                    if D>DisMax(k,4)
                        DisMax(k,2:4)= [i j D];
                    end
                    if D<DisMin(k,4)
                        DisMin(k,2:4)= [i j D];
                    end
                end
            end
        end
    end
end

DisMax(:,4)=DisMax(:,4)*2; %treballarem en diametres en comptes de amb
el  radi
DisMin(:,4)=DisMin(:,4)*2;
Angles= (DisMin(:,4)./DisMax(:,4));
Angles=acosd(Angles);


end
```

1.10. **AlgCantonades**

```
function [Punter_Proxim, Id_Proxim]=AlgCantonades(Id,PosX,PosY,DimIM)
%       Cantonades              Cantonades: matriu que representa la
%      1           2            posicio de les cantonades.
%                               Id_Proxim: vector que recull en l'ordre de
%                               les cantonades quin area es la mes propera.
%                               Punter_Proxim: vector que relaciona la
%      3           4            cantonada proxima amb les dades
%                               anteriorment recullides.
Cantonades = [0 DimIM(2)+2 0 DimIM(2)+2; 0 0 DimIM(1)+2 DimIM(1)+2];
Id_Proxim=zeros(1,4);
Punter_Proxim=ones(1,4);
t_in=cputime;


for i=1:4
   Id_Proxim(i)=Id(1);
   Dist_min=sqrt((Cantonades(1,i)-PosX(1))^2+(Cantonades(2,i)-
PosY(1))^2);
   for t=1:length(Id)
       if Dist_min>sqrt((Cantonades(1,i)-PosX(t))^2+(Cantonades(2,i)...
               -PosY(t))^2)
           Dist_min=sqrt((Cantonades(1,i)-PosX(t))^2+(Cantonades(2,i)...
               -PosY(t))^2);
           Id_Proxim(i)=Id(t);
           Punter_Proxim(i)=t;
       end
   end
end
Auxiliar=[0 0 0 0];
for i=1:4
    Auxiliar(Punter_Proxim(i))=1 + Auxiliar(Punter_Proxim(i)) ;
end
for q=1:4
    if Auxiliar(q)>1
        Dis_Cant=zeros(4,4);
        for i=1:4
           for t=1:4
               Dis_Cant(i,t)=sqrt((Cantonades(1,t)-PosX(i))^2+ ...
                   (Cantonades(2,t)-PosY(i))^2);
           end
        end
        Dis_Min=1000*DimIM(2);

        for i=1:4
            for t=1:4
                if i~=t
                    for u=1:4
                        if ((i~=u) && (t~=u))
                            for y=1:4
                                if ((i~=y) && (t~=y) && (u~=y))
                                    Aux3=Dis_Cant(i,1)+Dis_Cant(t,2)+ ...
                                        Dis_Cant(u,3)+Dis_Cant(y,4);
                                    if Dis_Min>Aux3
                                        Dis_Min=Aux3;
                                        Id_Proxim=Id([i t u y]);
```

```
                                        Punter_Proxim=[i t u y];
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end


end
```

1.11. **AbatirPlano**

```
function [IM_Out]= AbatirPlano(IM_In, ID_Tag, DimIM, Pendent, Y_fuga)
IM_Out=IM_In;
%   Primer netegem la imatge eliminant el que considerem els Tag a nivell
% de base

for  i=1:DimIM(1)
    for u=1:DimIM(2)
        for t=1:length(ID_Tag)
            if IM_In(i,u)==ID_Tag(t)
                IM_Out(i,u)=1;
                break
            else
                IM_Out(i,u)=IM_In(i,u);
            end
        end
    end
end

for i=1:DimIM(1)
    k=floor((i-Y_fuga*Pendent)/(1-Pendent));
    if k>0   &&  k<DimIM(1)
        for u=1:DimIM(2)
            for t=1:length(ID_Tag)
                if IM_In(k,u)==ID_Tag(t)
                    IM_Out(i,u)=ID_Tag(t);
%                    IM_Out(i,u)=0;
                    break
                end
            end
        end
    end
end


end
```

1.12. **AlgBase**

```
function [Y_base]= AlgBase (IM_In, ID_Tag, DimIM)
Y_base=0;
for c=1:DimIM(2)
    for f=1:DimIM(1)
        for i=1:length(ID_Tag)
            if IM_In(f,c)==ID_Tag(i)
                if f>Y_base
                    Y_base=f;
                end
            end
        end
    end

end
if Y_base < DimIM(1)-5
    Y_base=Y_base+5;
else
    Y_base=DimIM(1);
end

end
```

1.13. **CalculCentre**

```
function [PosY, PosX]= CalculCentre (IM_In, ID_Tag, DimIM)
Area=zeros(1,length(ID_Tag));
PosX=Area;
PosY=PosX;
for i=2:DimIM(1)+1
    for u=2:DimIM(2)+1
        Area(ID_Tag==IM_In(i,u))=Area(ID_Tag==IM_In(i,u))+1;
        PosX(ID_Tag==IM_In(i,u))=PosX(ID_Tag==IM_In(i,u))+u;
        PosY(ID_Tag==IM_In(i,u))=PosY(ID_Tag==IM_In(i,u))+i;
    end
end
PosX=PosX./Area;
PosY=PosY./Area;

end
```

1.14. AlgRelacioPixelRealitat

```
function [RelacioPixelRealitat]=AlgRelacioPixelRealitat(IM_In,DimIM, ...
    Elements_IM12,Elements_Tag, Cercle, Quadrat, X_fuga,
Y_fuga,Y_base,...
    Id_Proxim,Punter_Proxim,PosY)
%   Inicialitzem les variables, per els valors que busquem de esquerra i
% dreta establim el punt mes crític possible (cada una en el punt oposat
% mes lluny oposat possible.

dreta= (Y_base-1)*(X_fuga-1)/(Y_fuga-1)+1;
esquerra=(Y_base-1)*(X_fuga-DimIM(2))/(Y_fuga-1)+DimIM(2);
i=1;
%   Primer es busca un element que sigui un Tag de tipus cercle
while Cercle(i)~=1 || Quadrat(i)==1
    i=i+1;
    if i>length(Cercle)
    msgbox('No es pot diferenciar els cercles del cuadrat',...
        'AlgRelacioPixelRealitat','error');
    break
    end
end
minX=0;
for i=1:4
    if Id_Proxim(i)~=Elements_IM12(Quadrat==1)
        if minX<PosY(Elements_IM12(Cercle==1)==Elements_Tag ...
                (Punter_Proxim(i)))
        minX=PosY(Elements_IM12(Cercle==1)==Elements_Tag ...
            (Punter_Proxim(i)));
        punter=i;
        end
    end
end
%   Ara es repassa la imatge buscant els pixels on hi hagi el mateix
% valor del tag trobat i es fa la projecció sobre la base per establir
% la distancia que considerarem 2 cm
for c=1:DimIM(2)
    for f=1:DimIM(1)
        if IM_In(f,c)==Elements_Tag(punter)
            proj_X= (Y_base-f)*(X_fuga-c)/(Y_fuga-f)+c;
            if proj_X < esquerra
                esquerra= proj_X;
            end
            if proj_X > dreta
                dreta= proj_X;
            end
        end
    end
end
RelacioPixelRealitat=dreta-esquerra;
end
```

1.15. **AlgOrientacioA**

```
function [D,direccio]= AlgOrientacioA (Id_Proxim,Elements, Quadrat,...
    Amplada_zona,Profunditat_zona,X_fuga,Y_fuga,Y_base,PosX,PosY,...
    Punter_Proxim)
for i=1:length(Id_Proxim)
    if Id_Proxim(i)==Elements(Quadrat==1)
        if i==1
            Xa=(Y_base-Y_fuga)*(PosX(Punter_Proxim(3))-X_fuga)/...
                (PosY(Punter_Proxim(3))-Y_fuga)+X_fuga;
            Xb=(Y_base-Y_fuga)*(PosX(Punter_Proxim(4))-X_fuga)/...
                (PosY(Punter_Proxim(4))-Y_fuga)+X_fuga;
           % D=(Xb-Xa)/(Profunditat_zona/Amplada_zona);
            D=(Xb-Xa)*(Amplada_zona/Profunditat_zona);
            direccio='esque';
        elseif i==4
            Xa=(Y_base-Y_fuga)*(PosX(Punter_Proxim(3))-X_fuga)/...
                (PosY(Punter_Proxim(3))-Y_fuga)+X_fuga;
            Xb=(Y_base-Y_fuga)*(PosX(Punter_Proxim(4))-X_fuga)/...
                (PosY(Punter_Proxim(4))-Y_fuga)+X_fuga;
            % D=(Xb-Xa)/(Profunditat_zona/Amplada_zona);
            D=(Xb-Xa)*(Amplada_zona/Profunditat_zona);
            direccio='dreta';
        elseif i==2
            Xa=(Y_base-Y_fuga)*(PosX(Punter_Proxim(3))-X_fuga)/...
                (PosY(Punter_Proxim(3))-Y_fuga)+X_fuga;
            Xb=(Y_base-Y_fuga)*(PosX(Punter_Proxim(4))-X_fuga)/...
                (PosY(Punter_Proxim(4))-Y_fuga)+X_fuga;
            D=(Xb-Xa)*(Profunditat_zona/Amplada_zona);
            direccio='devan';
        elseif i==3
            Xa=(Y_base-Y_fuga)*(PosX(Punter_Proxim(3))-X_fuga)/...
                (PosY(Punter_Proxim(3))-Y_fuga)+X_fuga;
            Xb=(Y_base-Y_fuga)*(PosX(Punter_Proxim(4))-X_fuga)/...
                (PosY(Punter_Proxim(4))-Y_fuga)+X_fuga;
            D=(Xb-Xa)*(Profunditat_zona/Amplada_zona);
            direccio='darre';
        end
    end
end

end
```

1.16. **FuncioAltura**

```
function [X_altura, X_alturae,X_altura2, X_alturae2]= FuncioAltura (PosX,
PosY,Punter_Proxim,...
    ID_Tag, X_fuga, Y_fuga, Y_base,D)
%   Ens assegurem que els dos punts seleccionats estiguin en el mateix
% canto de l'eix que representa el punt de fuga, en cas de que no es
% cumpleixi comprovarem si amb l'altre parella de punts podem fer els
% calculs de forma correcte
X_altura=0; X_alturae=0; X_altura2=0; X_alturae2=0;

if PosX(Punter_Proxim(1)==ID_Tag)<X_fuga && ...
        PosX(Punter_Proxim(3)==ID_Tag)<X_fuga

    Ya=PosY(Punter_Proxim(1)==ID_Tag);
    Xa=PosX(Punter_Proxim(1)==ID_Tag);
    Yb=PosY(Punter_Proxim(3)==ID_Tag);
    Xb=PosX(Punter_Proxim(3)==ID_Tag);

    a=((Y_base-Y_fuga)*(Xa-X_fuga))/(Ya-Y_fuga)+X_fuga;
    b=((Y_base-Y_fuga)*(Xb-X_fuga))/(Yb-Y_fuga)+X_fuga;

    f1=(D+Xa-Xb-a+b)*(Y_fuga-Ya)*(Y_fuga-Yb);
    f2=Xb*(Y_base-Yb)*(Y_fuga-Ya);
    f3=Xa*(Y_base-Ya)*(Y_fuga-Yb);
    f4=(Y_base-Yb)*(Y_fuga-Ya)-(Y_base-Ya)*(Y_fuga-Yb);
    X_altura=(f1+f2-f3)/f4;

    X_alturae=2*X_fuga-X_altura;
end
if PosX(Punter_Proxim(2)==ID_Tag)> X_fuga && ...
        PosX(Punter_Proxim(4)==ID_Tag)> X_fuga

    Yb=PosY(Punter_Proxim(2)==ID_Tag);
    Xb=PosX(Punter_Proxim(2)==ID_Tag);
    Ya=PosY(Punter_Proxim(4)==ID_Tag);
    Xa=PosX(Punter_Proxim(4)==ID_Tag);

    a=((Y_base-Y_fuga)*(Xa-X_fuga))/(Ya-Y_fuga)+X_fuga;
    b=((Y_base-Y_fuga)*(Xb-X_fuga))/(Yb-Y_fuga)+X_fuga;

    f1=(D+Xa-Xb-a+b)*(Y_fuga-Ya)*(Y_fuga-Yb);
    f2=Xb*(Y_base-Yb)*(Y_fuga-Ya);
    f3=Xa*(Y_base-Ya)*(Y_fuga-Yb);
    f4=(Y_base-Yb)*(Y_fuga-Ya)-(Y_base-Ya)*(Y_fuga-Yb);
    X_alturae2=(f1+f2-f3)/f4;

    X_altura2=2*X_fuga-X_alturae2;

end % final del if
end
```

1.17. **AlgLimitZonaTreball**

```
function [esquerra,dreta,superior]= AlgLimitZonaTreball (IM_In,ID_Tag,...
    DimIM,X_fuga,Y_fuga,Y_base,X_altura,X_alturae)
esquerra=DimIM(2)/2;
dreta=DimIM(2)/2;
superior=0;
% L'analizis el relitzarem en dos parts per evitar fer varios if cada cop
% que es canvia de columna.
for c=1:floor(X_fuga)
    for f=1:Y_base
        for u=1:length(ID_Tag)
            if ID_Tag(u)==IM_In(f,c)
                proj_X= (Y_base-f)*(X_fuga-c)/(Y_fuga-f)+c;
                if proj_X < esquerra
                    esquerra = proj_X;
                end
                proj_Y= (Y_base-f)*(X_altura-c)/(Y_fuga-f)+c;
                if superior < (proj_X - proj_Y)
                   superior = proj_X - proj_Y;
                end
            end
        end
    end
end
for c=(floor(X_fuga)+1):DimIM(2)% repetim per l'altre canto
    for f=1:Y_base
        for u=1:length(ID_Tag)
            if ID_Tag(u)==IM_In(f,c)
                proj_X= (Y_base-f)*(X_fuga-c)/(Y_fuga-f)+c;
                if proj_X > dreta
                    dreta = proj_X;
                end
                proj_Y= (Y_base-f)*(X_alturae-c)/(Y_fuga-f)+c;
                if superior < (proj_Y - proj_X)
                   superior = proj_Y - proj_X;
                end

            end
        end
    end
end

end
```

1.18. **AlgTransformacio**

```
function [IM_Out]= AlgTransformacio(X_altura,X_alturae,...
    X_fuga,Y_fuga,Y_base, IM_In,superior, inici, final)
superior=floor(superior);
inici=floor(inici);
final=floor(final);
ample=final-(inici);
X_fuga=X_fuga-inici;
X_altura=X_altura-inici;
X_alturae=X_alturae-inici;
DimIM=size(IM_In);
if length(DimIM)==3
    IM_Out=zeros(superior,ample,3,'uint8');
else
    IM_Out=zeros(superior,ample,'uint8');
end
for c=1:ample
    if c < X_fuga
        X_altura2 = X_altura;
    else
        X_altura2 = X_alturae;
    end
     for f=1:superior
        if c < X_fuga
            X_delta= c - (superior-f);
        else
            X_delta= c + (superior-f);
        end
        f1=-c*X_altura2+X_delta*X_fuga;
        Ox= f1/(-X_altura2+X_delta+X_fuga-c);
        f2=(Ox-c)*(Y_fuga-Y_base);
        if f2==0
            Oy=0;
        else
            Oy= f2/(X_fuga-c)+Y_base;
        end
        Ox=Ox+inici;
        if Ox>1 && Ox<DimIM(2)&& Oy>1 && Oy<superior+Y_base
            inOx = floor(Ox);
            inOy = floor(Oy);
            DOx = Ox-inOx;
            DOy = Oy-inOy;
            if length(DimIM)==3
                for p=1:3
                IM_Out(f,c,p)=((1-DOx)*(1-DOy)*IM_In(inOy,inOx,p)+...
                    DOy*(1-DOx)*IM_In(inOy+1,inOx,p)+...
                    DOx*(1-DOy)*IM_In(inOy,inOx+1,p)+...
                    DOx*DOy*IM_In(inOy+1,inOx+1,p))/...
                    ((1-DOy)*(1-DOx)+DOy*(1-DOx)+DOx*...
                    (1-DOy)+DOx*DOy);
                end
            else
                IM_Out(f,c)=IM_In(floor(Oy),floor(Ox));
            end
        end
    end
```

```
end

end
```

1.19. **TrobarPeces**

```
function [IM_Out]= TrobarPeces(IM_In,DimIM)
IM_Out=zeros(DimIM(1),DimIM(2));
for c=1:DimIM(2)
    for f=1:DimIM(1)
        if (IM_In(f,c,1)+30)<IM_In(f,c,3) && IM_In(f,c,3)<200 && ...
                IM_In(f,c,3)>100   && IM_In(f,c,1)<120
            IM_Out(f,c)=0;
        else
            IM_Out(f,c)=1;
        end
    end

end
end
```

1.20. **CalculArea**

```
function [area,Id, IM_In]= CalculArea(IM_In,DimIM,k,area_Tag)
%   En aquesta funció es realitza el primer càlcul (Area) i es modifica
la
% imatge per descartar tot allò que es consideren errors.

% Inicialització de variables
area=zeros(1,k+1);
arearef=area_Tag(1);

% Càlcul d'àrees
for f=2:DimIM(1)+1
    for c=2:DimIM(2)+1
        area(1,IM_In(f,c))=area(1,IM_In(f,c))+1;
    end
end

%   Es selecciona l'àrea més gran ja  que s'utilitzen les àrees dels Tag
en
% un moment previ a la transformació de la imatge però aquest valor es
% suficientment aproximat per ser útil.

for i=2:length(area_Tag)
    if arearef<area_Tag(i)
        arearef=area_Tag(i);
    end
end

%   Un cop es te l'area de referencia adecuada es realitza el filtrat on
es
% descarten com a areas significatives valors superiors a 20 vegades
l'area
% de referencia (eliminant aixi l'area del fons) i valors més petits que
% l'area de referencia

for f=2:DimIM(1)+1
    for c=2:DimIM(2)+1
        if area(1,IM_In(f,c)) < arearef ||area(1,IM_In(f,c)) > arearef*20
            IM_In(f,c)=1;
        end
    end
end

%   Finalment filtrem la informació que extraem de la imatge per nomes
% enviar dades utils i no vectors plens de zeros.
Id=1:1:k;
Id=Id((area > area_Tag(4))&(area<20*area_Tag(4))==1);
area=area((area > area_Tag(4))&(area<20*area_Tag(4))==1);
end
```

1.21. **FuncioMarges**

```
function [IM_Out3,Perimetre]=FuncioMarges(IM_In,DimIM, Id)
IM_Out3=IM_In;
Perimetre=zeros(1,length(Id));

for c=2:DimIM(2)+1
    for f=2:DimIM(1)+1
        if IM_In(f,c)~=1
            if IM_In(f,c-1)~=1 && IM_In(f-1,c)~=1 && IM_In(f,c+1)~=1 ...
                    && IM_In(f+1,c)~=1
                IM_Out3(f,c)=0;

            else
                Perimetre=(IM_In(f,c)==Id)+Perimetre;
            end

        end
    end
end
end
```

1.22. **AlgVertex**

```
function [Info_Figures]= AlgVertex (IM_In, Id, PosX,
PosY,Perimetre,DimIM)
Info_Figures=zeros(4,6,length(Id));
i=1;
for f=1:DimIM(1)
    for c=1:DimIM(2)
        if length(Id)>=i
        if IM_In(f,c)==Id(i)
            Maxims=zeros(3,Perimetre(i),2);
            fa=f;
            ca=c;
            MATtor=[-1 0; 0 1; 1 0; 0 -1];  %constants i variables
tortuga
            p=1;
            while((p+3)~=Perimetre(i)) %Recegueix el perfil i calcula
distancia
                if IM_In(fa,ca)~=1      %al centre de masses
                    MATtor= circshift(MATtor,1);
                    if IM_In(fa,ca)==Id(i)% es contorn
                        p=p+1;
                        Maxims(:,p,1)=[ca,fa,sqrt((fa-PosY(i))^2+ ...
                            (ca-PosX(i))^2)];
                        IM_In(fa,ca)=0;
                    end
                else
                    MATtor= circshift(MATtor,-1);
                end
                fa=fa+MATtor(1,1);
                ca=ca+MATtor(1,2);
            end
            Maxims=[Maxims(:,p-4:p,:) Maxims Maxims(:,1:5,:)];
            for k=1:p
                if Maxims(3,k+5,1) > Maxims(3,k:k+4,1)
                    if Maxims(3,k+5,1) > Maxims(3,k+6:k+10,1)
                        Maxims(3,k+5,2)=1;
                    end
                end
            end
            Maxims=Maxims(:,(Maxims(3,:,2)==1),:);
            t=size(Maxims);
            Maxims=[Maxims(:,t(2),:) Maxims Maxims(:,1,:)];
            t=size(Maxims);
            for u=1:t(2)-1
                ax=Maxims(1,u,1);ay=Maxims(2,u,1);
                bx=Maxims(1,u+1,1);by=Maxims(2,u+1,1);
                if sqrt((ax-bx)^2+(ay-by)^2)< Perimetre(i)/9
                    if Maxims(3,u,1)<Maxims(3,u+1,1)
                        Maxims(3,u,2)=0;
                    else
                        Maxims(3,u+1,2)=0;
                    end
                end
            end
            Maxims=Maxims(:,2:t(2)-1,:);
            Maxims=(Maxims(:,(Maxims(3,:,2)==1),:));
```

```
            t=size(Maxims);
            Maxims=[Maxims(:,t(2),:) Maxims Maxims(:,1,:)];
            t=size(Maxims);
            for u=2:t(2)-1
                A=[Maxims(1,u,1)-Maxims(1,u-1,1) (Maxims(2,u,1)...
                    -Maxims(2,u-1,1))];
                B=[Maxims(1,u,1)-Maxims(1,u+1,1) (Maxims(2,u,1)...
                    -Maxims(2,u+1,1))];
                AB=A(1)*B(1)+A(2)*B(2);
                IAI=sqrt(A(1)^2+A(2)^2);
                IBI=sqrt(B(1)^2+B(2)^2);
                teta=acosd(AB/(IAI*IBI));
                if teta>130
                    if Maxims(3,u,1)<Maxims(3,u-1,1)&& ...
                            Maxims(3,u,1)<Maxims(3,u+1,1)
                       Maxims(3,u,2)=0;
                    elseif Maxims(3,u+1,1)<Maxims(3,u,1)&& ...
                            Maxims(3,u+1,1)<Maxims(3,u-1,1)
                        Maxims(3,u+1,2)=0;
                    else
                        Maxims(3,u-1,2)=0;
                    end
                end
            end
            Maxims=Maxims(:,2:t(2)-1,:);
            t=size(Maxims(:,(Maxims(3,:,2)==1),1));
            Info_Figures(1:3,:,i)=[Maxims(:,(Maxims(3,:,2)==1),1) ...
                zeros(3,6-t(2))];
            i=i+1;

        end
        end

    end
end

end
```

1.23. **AlgOrientacioB**

```
function [teta,Id_rotacio,PosX2,PosY2]=AlgOrientacioB(X_fuga, ...
    Y_fuga, Y_base, X_altura, X_alturae, PosX, PosY_Elevat,...
    Elements_IM12, Proxim, Cuadrats, Id_Tag,esquerra,superior)

PosX2=PosX;
PosY2=PosY_Elevat;
X_fuga140=X_fuga-esquerra;
X_altura=X_altura-esquerra;
X_alturae=X_alturae-esquerra;

for i=1:4
    if PosX(i)<X_fuga
       PosX2(i)= (Y_base-PosY_Elevat(i))*(X_fuga-PosX(i))/...
           (Y_fuga-PosY_Elevat(i))+PosX(i);
       PosY2(i)= -((Y_base-PosY_Elevat(i))*(X_altura-PosX(i))/...
           (Y_fuga-PosY_Elevat(i))+PosX(i))+PosX2(i);
       PosX2(i)= PosX2(i)-esquerra;


    else
       PosX2(i)= (Y_base-PosY_Elevat(i))*(X_fuga-PosX(i))/...
           (Y_fuga-PosY_Elevat(i))+PosX(i);
       PosY2(i)= ((Y_base-PosY_Elevat(i))*(X_alturae-PosX(i))/...
           (Y_fuga-PosY_Elevat(i))+PosX(i))-PosX2(i);
       PosX2(i)= PosX2(i)-esquerra;

    end

end

PosY2=superior-PosY2;
if Proxim([1 3])~=Elements_IM12(Cuadrats==1)
%     if PosX2(Proxim(1)==Id_Tag)<X_fuga140 && ...
%         PosX2(Proxim(3)==Id_Tag)<X_fuga140

        d=PosY2(Proxim(1)==Id_Tag);
        b=PosX2(Proxim(1)==Id_Tag);
        c=PosY2(Proxim(3)==Id_Tag);
        a=PosX2(Proxim(3)==Id_Tag);

        Id_rotacio=Proxim(3);

%     end
end
if Proxim([2 4])~=Elements_IM12(Cuadrats==1)
%     if PosX2(Proxim(2)==Id_Tag)> X_fuga140 && ...
%         PosX2(Proxim(4)==Id_Tag)> X_fuga140

        d=PosY2(Proxim(2)==Id_Tag);
        b=PosX2(Proxim(2)==Id_Tag);
        c=PosY2(Proxim(4)==Id_Tag);
        a=PosX2(Proxim(4)==Id_Tag);
```

```
        Id_rotacio=Proxim(4);

%     end % final del if
end
AuxX=b-a;
AuxY=d-c;
teta=atand((AuxX)/(AuxY));
if and( AuxX<0 , AuxY >0)
    teta= teta+360;
elseif and( AuxX>0 , AuxY <0)
    teta=teta+180;
elseif and( AuxX<0 , AuxY <0)
    teta=teta+180;
    % la 4a opcio es que estigues en el primer quadrant i per tant no es
    % necessari modificar el seu valor

end
end
```

1.24. **RotacioPosicions**

```
function [PosX_Final,PosY_Final,RelacioPixelRealitat]=RotacioPosicions
(PosX,PosY,PosX_F,...
    PosY_F,teta,direccio,Id_rotacio,Id_Tag,Profunditat_zona,...
    Amplada_zona,RelacioPixelRealitat,Punter_Proxim)
PosX_Final=PosX_F;
PosY_Final=PosY_F;
Ox=PosX(Id_rotacio==Id_Tag);
Oy=PosY(Id_rotacio==Id_Tag);
Profunditat_zona=Profunditat_zona*RelacioPixelRealitat;
Amplada_zona=Amplada_zona*RelacioPixelRealitat;
for i=1:length(PosX_F)
    AuxX=PosX_F(i)-Ox;
    AuxY=PosY_F(i)-Oy;
    Vector=sqrt(AuxX^2+AuxY^2);
    teta2=atand(AuxX/AuxY);
    if and( AuxX<0 , AuxY >0)
        teta2= teta2+360;
    elseif and( AuxX>0 , AuxY <0)
        teta2=teta2+180;
    elseif and( AuxX<0 , AuxY <0)
        teta2=teta2+180;
    end

    PosX_Final(i)=Vector*sind(teta2-teta);
    PosY_Final(i)=Vector*cosd(teta2-teta);

end
for i=1:length(PosX)
    if i~=find(Id_Tag==Id_rotacio)
        AuxX=PosX(i)-Ox;
        AuxY=PosY(i)-Oy;
        Vector=sqrt(AuxX^2+AuxY^2);
        teta2=atand(AuxX/AuxY);
        if and( AuxX<0 , AuxY >0)
            teta2= teta2+360;
        elseif and( AuxX>0 , AuxY <0)
            teta2=teta2+180;
        elseif and( AuxX<0 , AuxY <0)
            teta2=teta2+180;
        end
        PosX(i)=Vector*sind(teta2-teta);
        PosY(i)=Vector*cosd(teta2-teta);
    else
        PosX(i)=0;
        PosY(i)=0;

    end


end
%Pos_quadrat_rotat-Pos_rotacio-Pos_F_rotada*2/RelacioPixelRealitat
%Amplada_zona=-Amplada_zona;
if direccio=='darre'
    Amplada_zona=abs(PosX(Punter_Proxim(3))-PosX(Punter_Proxim(4)));
```

```
    Profunditat_zona=abs(PosY(Punter_Proxim(1))-PosY(Punter_Proxim(3)));
    RelacioPixelRealitat=Amplada_zona/36;
    PosX_Final=Amplada_zona-PosX_Final;
    %PosY_Final=Profunditat_zona-PosY_Final;
elseif direccio == 'dreta'
    Amplada_zona=abs(PosY(Punter_Proxim(1))-PosY(Punter_Proxim(3)));
    Profunditat_zona=abs(PosX(Punter_Proxim(3))-PosX(Punter_Proxim(4)));
    RelacioPixelRealitat=Amplada_zona/36;
    aux=PosY_Final;
    PosY_Final=Profunditat_zona-abs(PosX_Final);
    PosX_Final=aux;
elseif direccio == 'devan'
    Amplada_zona=abs(PosX(Punter_Proxim(3))-PosX(Punter_Proxim(4)));
    Profunditat_zona=abs(PosY(Punter_Proxim(1))-PosY(Punter_Proxim(3)));
    RelacioPixelRealitat=Amplada_zona/36;
    PosX_Final=Amplada_zona-abs(PosX_Final);
    PosY_Final=Profunditat_zona-abs(PosY_Final);
else
    Amplada_zona=abs(PosY(Punter_Proxim(1))-PosY(Punter_Proxim(3)));
    Profunditat_zona=abs(PosX(Punter_Proxim(3))-PosX(Punter_Proxim(4)));
    RelacioPixelRealitat=Amplada_zona/36;
    aux=PosY_Final;
    PosY_Final=Profunditat_zona-PosX_Final;
    PosX_Final=Amplada_zona-aux;

end

PosY_Final=PosY_Final/RelacioPixelRealitat;
PosX_Final=PosX_Final/RelacioPixelRealitat;
RelacioPixelRealitat=RelacioPixelRealitat*2;
end
```

1.25. **AlgIdentificacio**

```
function [nomfigura]= AlgIdentificacio(Info_Figures,Area_F,Base_Dades,...
    RelacioPixelRealitat)
DimInfo_Figures=size(Info_Figures);
if length(DimInfo_Figures)<3
    DimInfo_Figures=[DimInfo_Figures 1];
end
Matinfo=zeros(7,6,DimInfo_Figures(3));
Matinfo(1:DimInfo_Figures(1),:,:)=Info_Figures;
nomfigura=zeros(1,DimInfo_Figures(3));
Area_F=Area_F/(RelacioPixelRealitat/10);

for i=1:DimInfo_Figures(3)
    Aux=Matinfo(:,(Matinfo(1,:,i)~=0),i);
    DimAux=size(Aux);
    for p=1:DimAux(2)
        if p~=DimAux(2)
            Aux(4,p)=sqrt((Aux(1,p)-Aux(1,p+1))^2+(Aux(2,p)-
Aux(2,p+1))^2);
        else
            Aux(4,p)=sqrt((Aux(1,p)-Aux(1,1))^2+(Aux(2,p)-Aux(2,1))^2);
        end

    end
    for p=1:DimAux(2)
        if Aux(3,p)>=Aux(3,:)
            EixLlarg=Aux(3,p);
        end
        if Aux(3,p)<=Aux(3,:)
            EixCurt=Aux(3,p);
        end
        if Aux(4,p)>=Aux(4,:)
            ArestaMax=Aux(4,p);
        end
        if Aux(4,p)<=Aux(4,:)
            ArestaMin=Aux(4,p);
        end
    end

    if DimAux(2)==6
        if (Base_Dades((Base_Dades(:,1)==1),2)<Area_F(i) && ...
        Base_Dades((Base_Dades(:,1)==1),3)>Area_F(i))
            nomfigura(i)=1;
        end
    elseif DimAux(2)==5
        if (Base_Dades((Base_Dades(:,1)==2),2)<Area_F(i) && ...
        Base_Dades((Base_Dades(:,1)==2),3)>Area_F(i))
            nomfigura(i)=2;
        end
    elseif DimAux(2)==4
        if ((EixLlarg-EixCurt)/EixLlarg)>0.2
            %rombe
            if (Base_Dades((Base_Dades(:,1)==9),2)<Area_F(i) && ...
                    Base_Dades((Base_Dades(:,1)==9),3)>Area_F(i))
                nomfigura(i)=9;
```

```
            else
                nomfigura(i)=10;
            end
        else
            if ((ArestaMax-ArestaMin)/ArestaMax)>0.30
            %rectangles
                if (Base_Dades((Base_Dades(:,1)==6),2)<Area_F(i) && ...
                        Base_Dades((Base_Dades(:,1)==6),3)>Area_F(i))
                    nomfigura(i)=6;
                elseif (Base_Dades((Base_Dades(:,1)==7),2)<Area_F(i) &&
...
                        Base_Dades((Base_Dades(:,1)==7),3)>Area_F(i))
                    nomfigura(i)=7;
                else
                    nomfigura(i)=8;
                end
            else
                %quadrat
                if (Base_Dades((Base_Dades(:,1)==3),2)<Area_F(i) && ...
                            Base_Dades((Base_Dades(:,1)==3),3)>Area_F(i))
                    nomfigura(i)=3;
                elseif (Base_Dades((Base_Dades(:,1)==4),2)<Area_F(i) &&
...
                            Base_Dades((Base_Dades(:,1)==4),3)>Area_F(i))
                    nomfigura(i)=4;
                else
                    nomfigura(i)=5;
                end
            end
        end
    elseif DimAux(2)==3
        if ((ArestaMax-ArestaMin)/ArestaMax)<0.25
            nomfigura(i)=11;
        elseif ((ArestaMax-ArestaMin)/ArestaMax)>0.5
            nomfigura(i)=14;
        else
            if (Base_Dades((Base_Dades(:,1)==12),2)<Area_F(i) && ...
                        Base_Dades((Base_Dades(:,1)==12),3)>Area_F(i))
                nomfigura(i)=12;
            else
                nomfigura(i)=13;
            end

        end

    end
    if nomfigura(i)==0
        nomfigura(i)=15;
    end

end


end
```

1.26. **FuncioIdentificacioImatge**

```
function [IM200]=FuncioIdentificacioImatge(IM140, PosX_F, PosY_F, ...
    nomfigura,PosX_Final,PosY_Final,Base_Dades_Peces)


IM200=IM140;
PosF=[PosX_F' PosY_F'];
%label=repmat(' ',[1 length(nomfigura)]);
for i=1:length(nomfigura)
label=strcat(num2str(PosX_Final(i)),' X \n ',num2str(PosY_Final(i)),...
    ' Y \n',Base_Dades_Peces{nomfigura(i),2});
label=sprintf(label);
IM200=insertText(IM200,PosF(i,:),label,'AnchorPoint','LeftBottom');

end
end
```

Anex B Dades

En aquest apartat s'introduirant les dades obtingudes i calculades en la obtenció de resultats .

Àreas obtingudes

Hexagon	Pentagon	Qua. Petit	Qua. Mitja	Qua. Gran	Rec. Petit	Rec. Mitja	Rec. Gran	Romb. Petit	Romb. Gran	Tr. Equilater	Tr. Iso. Pe	Tr. Iso. Gra	Tr. Escal
1257,8	863,1	721,9	1429,2	1726,5	851,1	1163,3	1917,1	760,7	980,9	412,4	407,8	696,2	499,3
1173,5	808,8	672,7	1342,1	1617,1	854	1158,3	1905,5	773,7	977,5	433,5	433,5	744,8	525,6
1264,8	863,3	727,7	1442,2	1735,1	849,5	1149,3	1882	759,9	966,7	424	418,6	701,7	498,9
1233,6	846,2	702,1	1405,3	1696,8	844,8	1145,4	1893,9	769,4	972,5	425,2	420,3	706,6	502,8
1250,9	854,5	720,3	1422,8	1725,4	851,1	1147,7	1874,9	754	950,1	419,3	412,4	695,3	490,35
1174,9	804,8	673,2	1342,9	1620,9	845,6	1137,9	1877,3	752,9	964,3	421,2	417,1	705,2	497

Graella 1 Àrees obtingudes(mm^2)

Valor Mig

1242,25	850,35	711,2	1414,05	1711,1	850,3	1148,5	1887,95	760,3	969,6	422,6	417,85	703,45	499,1

Graella 2Mitjana (mm^2) àrees obtingudes Graella 1

Errors absoluts respecte mitjana medida

15,55	12,75	10,7	15,15	15,4	0,8	14,8	29,15	0,4	11,3	-10,2	-10,05	-7,25	0,2
-68,75	-41,55	-38,5	-71,95	-94	3,7	9,8	17,55	13,4	7,9	10,9	15,65	41,35	26,5
22,55	12,95	16,5	28,15	24	-0,8	0,8	-5,95	-0,4	-2,9	1,4	0,75	-1,75	-0,2
-8,65	-4,15	-9,1	-8,75	-14,3	-5,5	-3,1	5,95	9,1	2,9	2,6	2,45	3,15	3,7
8,65	4,15	9,1	8,75	14,3	0,8	-0,8	-13,05	-6,3	-19,5	-3,3	-5,45	-8,15	-8,75

-67,35	-45,55	-38	-71,15	-90,2	-4,7	-10,6	-10,65	-7,4	-5,3	-1,4	-0,75	1,75	-2,1

Graella 3Error absolut respecte Graella 2 amb les dades de Graella 1(mm^2)

Error relatiu respecte mitjana

1,2363	1,4810	1,5587	1,0658	0,8900	0,0897	1,2844	1,4022	0,0526	1,0970	-2,6747	-2,6009	-1,0231	0,0352
-5,4659	-4,8265	-5,6083	-5,0619	-5,4325	0,4149	0,8505	0,8442	1,7613	0,7669	2,8583	4,0502	5,8354	4,6598
1,7928	1,5043	2,4036	1,9804	1,3870	-0,0897	0,0694	-0,2862	-0,0526	-0,2815	0,3671	0,1941	-0,2470	-0,0352
-0,6877	-0,4821	-1,3256	-0,6156	-0,8264	-0,6168	-0,2690	0,2862	1,1961	0,2815	0,6818	0,6341	0,4445	0,6506
0,6877	0,4821	1,3256	0,6156	0,8264	0,0897	-0,0694	-0,6278	-0,8281	-1,8930	-0,8653	-1,4105	-1,1501	-1,5386
-5,3546	-5,2911	-5,5355	-5,0056	-5,2129	-0,5271	-0,9199	-0,5123	-0,9727	-0,5145	-0,3671	-0,1941	0,2470	-0,3693

Graella 4Error relatiu (%) respecte Graella 2 amb les dades de Graella 1

Taula amb les iteracions per obtenir les figures de posicionament Ground truth

Iteració	Actual mm^2	Desviacio mm^2	X real (cm)I	Y real(cm)I	X (cm)I	Y (cm)I	Error (cm)I	error X (cm)I	error Y(cm)I
1	702,58	-16,1	12	7	11,5979	7,3909	0,5607916	-0,4021	0,3909
2	709,06	-22,58	8	7	7,2942	7,5173	0,8750731	-0,7058	0,5173
3	786	-99,52	5	7	3,2731	8,6165	2,36542932	-1,7269	1,6165
4	792	-105,52	2	7	0,38779	8,1597	1,98598217	-1,61221	1,1597
5	673,5	12,98	4	3	3,83	0,59637	2,40963424	-0,17	-2,40363
6	737,17	-50,69	6	2	4,7741	1,6504	1,27477487	-1,2259	-0,3496
7	706,04	-19,56	10	2	9,07	2,35	0,99368003	-0,93	0,35
8	698,34	-11,86	15	2	14,6733	2,159	0,36333716	-0,3267	0,159
9	678,99	7,49	20	2	19,916	1,8231	0,19583056	-0,084	-0,1769
10	677,67	8,81	25	2	25,2477	1,9059	0,26497189	0,2477	-0,0941
11	680	6,48	30	2	30,4303	1,7468	0,49926779	0,4303	-0,2532
12	686,95	-0,47	34	4	34,712	3,7622	0,7506616	0,712	-0,2378
13	684,5	1,98	27	4	27,3471	3,8967	0,36214541	0,3471	-0,1033
14	687,75	-1,27	22	4	21,9352	4,1838	0,19488838	-0,0648	0,1838
15	693,743	-7,263	17	4	16,6099	4,4467	0,59305893	-0,3901	0,4467
16	712,1	-25,62	12	4	11,281	4,51	0,88151064	-0,719	0,51
17	727,64	-41,16	4	7	2,7481	7,927	1,55774921	-1,2519	0,927
18	719,38	-32,9	9	7	7,8161	8,056	1,58642844	-1,1839	1,056
19	690,88	-4,4	14	7	13,4276	7,48	0,74702193	-0,5724	0,48
20	678,8	7,68	19	7	18,853	7,083	0,16881351	-0,147	0,083
21	662,27	24,21	24	7	24,413	6,548	0,61226873	0,413	-0,452
22	683,9	2,58	29	7	29,3676	6,935	0,3733025	0,3676	-0,065
23	723,7	-37,22	8	12	7,424	12,8163	0,9990604	-0,576	0,8163

Graella 5Valors en les primeres 23 iteracions

Iteració	Actual mm^2	Desviacio mm^2	X real (cm)I	Y real (cm)I	X (cm)I	Y (cm)I	error(cm)I	error X(cm)I	error Y(cm)I
24	689,6	-3,12	14	12	13,942	11,9988	0,05801241	-0,058	-0,0012
25	708,7	-22,22	20	12	19,9149	12,7943	0,79884573	-0,0851	0,7943
26	683,77	2,71	25	12	25,5711	11,6135	0,68959224	0,5711	-0,3865
27	705,59	-19,11	30	12	30,4064	12,4049	0,57367671	0,4064	0,4049
28	685,1197	1,3603	30	16	30,6742	16,3954	0,78159248	0,6742	0,3954
29	675,8	10,68	25	16	25,6139	16,8858	1,07773598	0,6139	0,8858
30	678,5	7,98	20	16	20,2597	16,9182	0,95421975	0,2597	0,9182
31	671,18	15,3	15	16	14,9573	17,0103	1,01120195	-0,0427	1,0103
32	680,8067	5,6733	10	16	9,5624	17,3981	1,46498374	-0,4376	1,3981
33	716,65	-30,17	6	15	5,2443	16,93	2,0726752	-0,7557	1,93
34	723,28	-36,8	6	19	5,3373	20,9247	2,03559362	-0,6627	1,9247
35	707,87	-21,39	11	19	10,7479	20,8301	1,84738204	-0,2521	1,8301
36	709,31	-22,83	16	19	15,7658	20,9267	1,94088189	-0,2342	1,9267
37	681,54	4,94	21	19	21,3923	19,7217	0,82143179	0,3923	0,7217
38	684,14	2,34	26	19	26,4088	19,894	0,98303278	0,4088	0,894
39	684,74	1,74	30	19	30,7249	19,6838	0,99652519	0,7249	0,6838
40	673,5	12,98	2	2	1,50394	1,48	0,71866232	-0,49606	-0,52

Graella 6Valors en les ultimes 17 iteracions