Pràctica 3
Reconeixement d'espècies animals

Jordi Nieto Maldonado
Jaime Arroyo Morales
12/06/2018
Visió per Computador - FIB

- Introducció

En aquesta pràctica de Visió per Computador, buscarem un mètode de classificació de 12 espècies d'animals diferents a partir d'imatges de mostra de cada una d'elles.

Per tal d'aconseguir aquest objectiu, haurem d'extreure característiques de cada una de les espècies i seleccionar quines són les més diferenciadores i que millor ens puguin funcionar. Un cop fet això, haurem d'escollir un classificador per tal d'entrenar-lo amb les característiques recollides i que sigui capaç de classificar les imatges que li passem com una de les 12 espècies disponibles.

- Descripció de les característiques utilitzades

Donat que tenim disponibles les coordenades aproximades del contorn dels animals i la seva *bounding box* amb cada imatge, hem aprofitat aquesta segmentació per tal de poder extreure el contorn dels animals utilitzant la transformada de *Fourier*. Aquesta transformació matemàtica ens permet representar qualsevol funció cíclica com una seqüència de sinus.

Gràcies a transformar el contorn mitjançant *Fourier*, podem utilitzar només una part de la transformada per definir la forma del animal (d'aquesta manera eliminem els detalls i ens fixem més en la forma general del animal de la imatge).

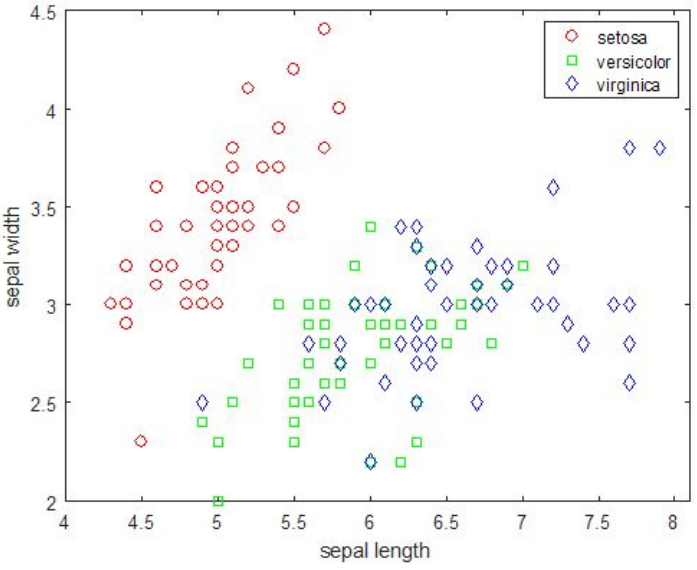
Un cop obtinguda la forma de l'animal amb la precissió desitjada mitjançant la transformada de *Fourier*, podem extreure les propietats de regió (*Region Props*) que desitgem mitjançant la funció de Matlab *regionprops*. Pel nostre programa hem escollit les següents propietats:

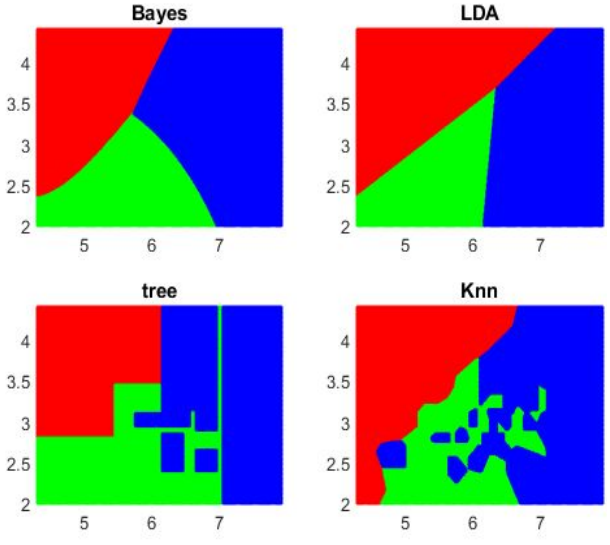
- **MajorAxisLength:** Ens diu la longitud màxima dels eixos d'una el·lipsis que es forma amb la mateixa variància que la regió.
- **MinorAxisLength:** Ens diu la longitud mínima dels eixos d'una el·lipsis que es forma amb la mateixa variància que la regió.
- **Eccentricity:** Ens diu la eccentricitat de la regió (és a dir, com de semblant és a una esfera)
- **Orientation:** Angle entre l'eix de les x i el major axis de l'el·lipsis.
- **ConvexArea:** Especifica la quantitat de píxels que hi ha en el mínim polígon convex que pot contenir la regió.
- **Extent:** Ratio total de pixels respecte al tamany de la bounding box.
- **Perimeter:** El perimetre de la figura és una region prop clau ja que animals amb moltes potes, per exemple, tindran un perimetre major.

A part de les *Region Props*, també hem tingut en compte per a la classificació de les imatges la mitja de les components R, G i B dels píxels continguts a la *Bounding Box* de cada animal. Per tal de poder diferenciar colors en el rang de grisos, no hem normalitzat la imatge (cosa que ens farà més dependents de la il·luminació), però tot i això aquesta propietat ens ajuda molt a diferenciar animals amb colors clarament diferenciats com per exemple el flamenc.

- Descripció dels classificadors utilitzats

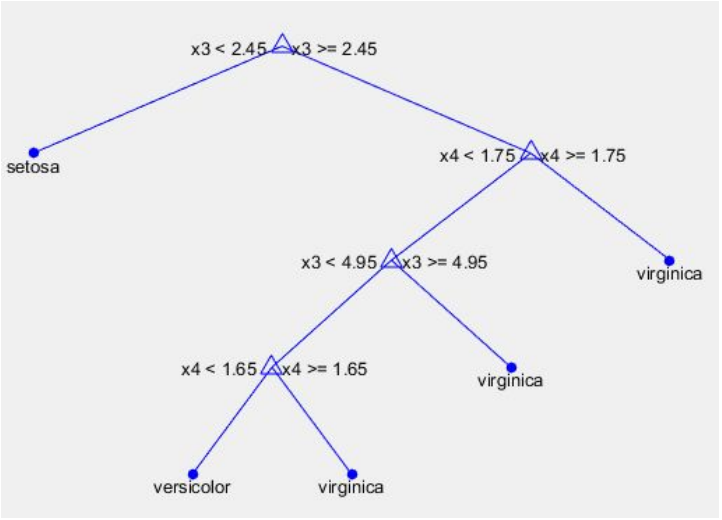
Un cop obtingudes i decidides les característiques a utilitzar, havíem de decidir el classificador que voldríem utilitzar. Donats els diferents mètodes de classificació que ens ofereix Matlab, vam provar amb varis dels classificadors que tenim disponibles i vam comprovar que n'hi havia dos que s'adaptaven bastant bé al nostre problema, que eren *K-nearest neighbor* i *Classification Tree*.





Al analitzar el comportament dels dos classificadors, vam veure que *K-nearest neighbor* utilitza un sistema de classificació mitjançant el qual escull el valor de grup que més a prop té, de manera que amb les dades de mostra ens donarà molt bon resultat però segurament no serà així amb les de test, ja que grups poc diferenciats segurament es trobin barrejats sense molt sentit (és massa sensible a mostres puntuals que es surtin dels valors).

D'altra banda, vam poder comprovar que el *Classification Tree* escolleix valors que diferenciïn un grup dels altres i va generant de manera automàtica un arbre binari de decisió, de manera que estem obtenint un mètode de classificació a partir de la mostra que és molt més general que l'anterior i que funciona millor amb les imatges de test.

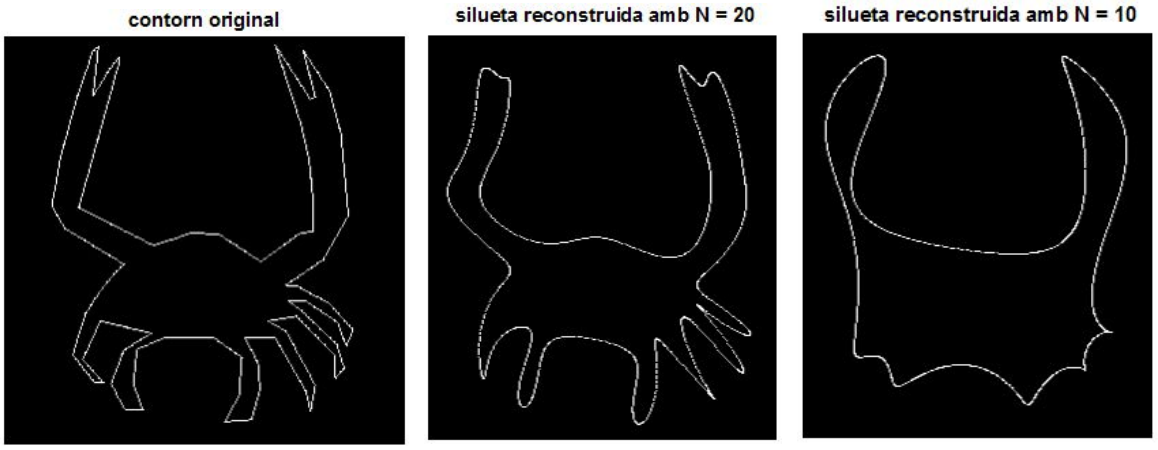


Per tant, hem escollit utilitzar l'arbre binari de classificació (*Classification Tree*) com a classificador del nostre programa donat que ha mostrat una millor obtenció de les característiques generals de les nostres classes i, per tant, ha sabut classificar de manera més exacta imatges de test que no havia vist mai.

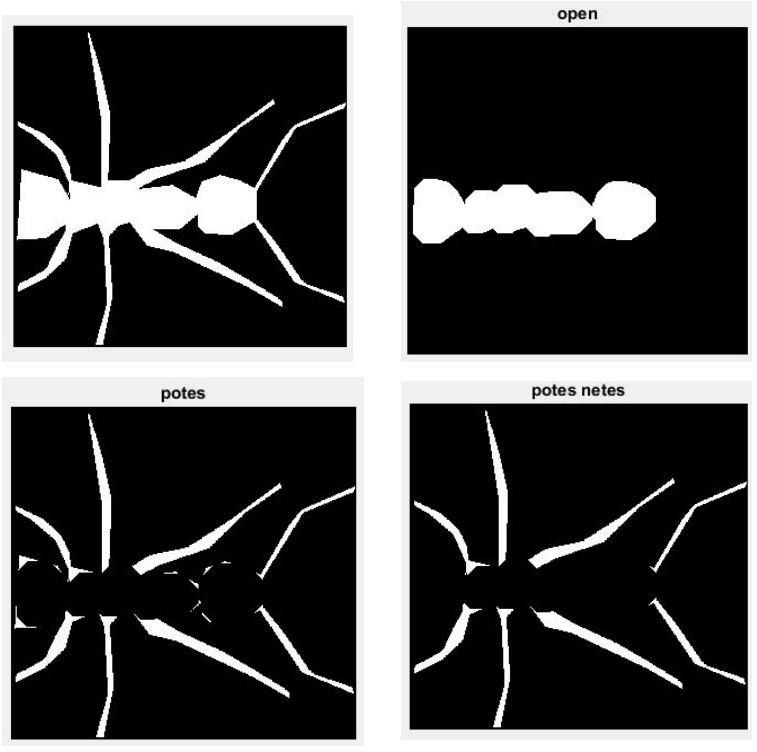
- Descripció dels experiments realitzats

Juntament amb les imatges dels animals, disposem d'un fitxer *annotation* d'on podem obtenir les dades necessaries per generar el contorn del animal i la seva *Bounding Box*. Aquest fitxer, però, només ens dóna les coordenades d'imatge dels vèrtex que formen el contorn, així que per obtenir la traça del contorn que desitgem hem hagut de generar una funció que calculi la seqüència de píxels corresponents a la aresta del contorn i els pinti en una matriu que representa una imatge binària. Gràcies a aquesta funció, hem pogut obtenir la seqüència de vèrtexs del contorn que necessitem per calcular la transformada de *Fourier*.

Un cop hem pogut obtenir la transformada de *Fourier*, sabíem que havíem de treballar amb contorns més genèrics, que tinguessin menys detall; i per aquesta raó vam estar provant agafar diferents quantitats de valors de la transformada de *Fourier* (com agafem més valors més precisió). Fent proves amb varis valors vam comprovar que, donat que volem que la forma quedi ben definida però sense gaires detalls, agafar N=20 valors de *Fourier* era suficient per obtenir una aproximació de la forma.



Per tal d'obtenir més dades rellevants, vam intentar segmentar del contorn altres propietats com pot ser la detecció de potes i extremitats dels animals. Per fer-ho vam realitzar un *Open* a la imatge amb un element estructurant circular de mida 15 i vam restar-li el resultat a la imatge original. Després vam eliminar petites zones no significatives per acabar obtenint la imatge amb les extremitats de l'animal.



Els resultats semblen convincents però ens vam adonar que aquesta metodologia és molt depenent de la perspectiva del animal, així que vam decidir no afegir-la a la implementació final perquè amb moltes imatges ens dóna resultats molt inestables.

Finalment, com hem explicat en el punt anterior, hem provat el comportament de diversos classificadors, quedant-nos amb el que ens ha proporcionat uns millors resultats, que ha estat l'arbre binari de classificació (*Classification Tree*).

- Resultats obtinguts

Després de realitzar diverses proves amb diferents classificadors i propietats, hem entrenat el nostre *Classification Tree* utilitzant el 80% de les imatges disponibles de cada animal com a imatges de mostra i el 20% restant con a imatges de test. Els resultats els hem representat en una matriu de confusió, on els valors que es troben a la diagonal són els resoltats correctes i els que no són resultats erronis:

```
CM =

     1     0     0     4     1     0     0     2     0     0     0     0
     0     4     0     0     2     2     0     0     0     0     0     0
     3     3     1     3     1     0     0     3     0     0     0     0
     4     0     2     4     0     0     0     1     1     1     1     0
     0     0     3     1     3     3     0     0     0     0     0     0
     0     0     1     0     0    10     0     0     0     0     1     0
     1     0     0     0     0     0     6     0     0     2     4     0
     0     1     0     2     1     1     0     6     1     0     0     0
     0     1     0     1     0     0     0     1     5     2     0     0
     0     2     0     1     0     0     1     1     5     3     0     0
     1     0     0     0     0     0     1     1     2     0    10     1
     0     3     0     0     0     0     0     0     1     1     1     1
```

Els animals de la matriu de confusió anterior segueixen el següent ordre alfabètic (tant per files com per columnes):

ant, beaver, crab, crayfish, crocodile, dolphin, dragonfly,
elephant, emu, flamingo, kangaroo, panda

- **Funcions utilitzades**

adivina(filename, annotationname): Donada una imatge i la seva annotation file, calcula de quin animal creu que es tracta la imatge. Primer obté les seves caracteristiques, després carrega el classificador i finalment fa la crida a la funció matlab "predict".

bordeIMG(imgfile,annotation_file): Donada una imatge i la seva annotationfile, crea una imatge formada per zeros i sobre ella va unint els punts indicats per l'annotation file per tal de crear el cortorn de la imatge.

countLegs(img_file, annotation_file): Donada una imatge i la seva annotation file, treu les parts principals de la imatge per quedar-se amb les potes i després les compta (no utilitzada).

createClassificator: Carrega totes les imatges que s'utilitzen per a l'entrenament del classificador, calcula les seves propietats i crea i entrena un classificador de tipus arbre.

expand(contorn, x, y): Funció recursiva que, donat un contorn, pinta els píxels que es troben en el seu interior (no utilitzada).

getProps(filename, annotationname): Funció que donada una imatge i el seu annotation file, realitza de manera automàtica la recolecció de les característiques que utilitzem.

histo(filename, annotation_file): Calcula el valor mitja de la component R, la component G i la B, per tal de després afegir-ho al vector de propietats de cada imatge.

loadTest: Carrega les imatges utilitzades per a testejar el classificador i calcula les seves propietats.

showNFourier(imgfile, annotation_file, N): Funció principal per calcular les propietats de les imatges. Fa la crida a la funció bordeIMG i al resultat li aplica la transformada de Fourier amb n=N. Finalment fa la crida a la funció regionprops per obtenir les propietats comentades anteriorment.

unionPuntos(img, X, Y): Donada una imatge buida i els dos vèrtexs d'una aresta, dibuixa un vèrtex als píxels que passen entre X i Y en línia recta.

- **Codi:**

adivina.m

```
function animal = adivina(filename, annotationname)
   stats = getProps(filename, annotationname);
   stats = stats';
   load clasificador.mat;
   animal = predict(cl, stats);
end
```

bordeIMG.m

```
function contorn = bordeIMG(imgfile, annotation_file)
%% imgfile: string
%% annotation_file: string

IMTYPE = 'jpg';
GUIDELINE_MODE = 1;
%% Parameters
%label_abbrev = {'LE', 'RE', 'LN', 'NB', 'RN', 'LM', 'RM'};
LARGEFONT = 28;
MEDFONT = 18;
BIG_WINDOW = get(0,'ScreenSize');
SMALL_WINDOW = [100 100 512 480];

%% load the annotated data
load(annotation_file, 'box_coord', 'obj_contour');

%% Read image
ima = imread(imgfile);

[filas, cols, ~] = size(ima);
contorn = zeros(filas, cols);
%% show contour
for cc = 1:size(obj_contour,2)
  if cc < size(obj_contour,2)
    contorn = unionPuntos(contorn, [obj_contour(1,cc), obj_contour(1,cc+1)]+box_coord(3),
[obj_contour(2,cc), obj_contour(2,cc+1)]+box_coord(1));
    %plot([obj_contour(1,cc), obj_contour(1,cc+1)]+box_coord(3), [obj_contour(2,cc),
obj_contour(2,cc+1)]+box_coord(1), 'r','linewidth',4);
  else
     contorn = unionPuntos(contorn, [obj_contour(1,cc), obj_contour(1,1)]+box_coord(3),
[obj_contour(2,cc), obj_contour(2,1)]+box_coord(1));
  end
end
```

countLegs.m

```
function councountLegs( img_file,annotation_file )
   % Función que cuenta la cantidad de patas de un animal

   contorn = bordeIMG(img_file,annotation_file);

   % Obtenemos la forma del animal (blanco forma, negro fondo)
   BW = expand(contorn,1,1);
   BW = imcomplement(BW);
   figure(1), imshow(BW)

   % Separamos las patas del cuerpo
   ee = strel('disk',15);
   op = imopen(BW,ee);
   figure(2), imshow(op), title('open')
   res = BW-op;
   figure(3), imshow(res), title('potes')

   % Traiem zones petites que siguin soroll
   res = bwareaopen(res,100);
   figure(4), imshow(res), title('potes netes')

   % Contamos los componentes conexos y guardamos la cantidad
   CC = bwconncomp(res);
   num_potes = CC.NumObjects
end
```

createClassificator.m

```
function clasificador = createClassificator()

   %% Script per executar els fitxers tots seguits
   % Añadimos el directorio de imagenes al path
   folder = fileparts(which(mfilename));
   addpath(genpath(folder));
   % Leemos ficheros .jpg del directorio actual y sus subdirectorios

   dirAnterior = ' ';

   %animals = ["ant", "beaver", "crab", "crayfish", "crocodile", "dolphin", "dragonfly", "elephant", "emu",
"flamingo", "kangaroo", "panda"];
   animals = [string('ant'), string('beaver'), string('crab'), string('crayfish'), string('crocodile'), string('dolphin'),
string('dragonfly'), string('elephant'), string('emu'), string('flamingo'), string('kangaroo'), string('panda')];

   data = [];
   species = [];
   for i = 1 : length(animals)
      path = strcat('imatges_animals_80_20\entreno\', animals(i),'\*.jpg');
      dinfo = dir(char(path));
      for j = 1 : length(dinfo)
         thisfolder = dinfo(j).folder;    % carpeta del fichero
```

```
        thisfilename = dinfo(j).name;              % nombre del fichero
        filename = strcat(thisfolder,'\',thisfilename);
        species = [species,animals(i)];
        %filename
        annotationname = replace(replace(filename,'.jpg','.mat') ,'image','annotation');
        stats = getProps(filename, annotationname);
        data = [data,stats];
      end
   end
   species = categorical(species');
   labels = categories(species);
   data = data';
   clasificador = fitctree(data, species);
end
```

expand.m

```
function contorn = expand(contorn, x, y)
   [filas, cols] = size(contorn);
   contorn(x,y) = 1;
   if x > 1
      if contorn(x-1,y) == 0
         contorn = expand(contorn, x-1, y);
      end
   end
   if x < filas
      if contorn(x+1,y) == 0
         contorn = expand(contorn, x+1, y);
      end
   end
   if y > 1
      if contorn(x,y-1) == 0
         contorn = expand(contorn, x, y-1);
      end
   end
   if y < cols
      if contorn(x,y+1) == 0
         contorn = expand(contorn, x, y+1);
      end
   end
end
```

getProps.m

```
function stats = getProps(filename, annotationname)
   stats = showNFourier(filename,annotationname,20);
   rgb = histo(filename, annotationname);
   stats = cat(1, stats, rgb');
end
```

histo.m

```
function rgb = histo(filename, annotation_file)
   im = imread(filename);
   [a,b, dim] = size(im);
   load(annotation_file, 'box_coord', 'obj_contour');
   %box_coord
   im = im(box_coord(1):box_coord(2), box_coord(3):box_coord(4), :);
   num = (box_coord(2)-box_coord(1)) .* (box_coord(4)-box_coord(3));

   if dim == 1
      rgb(1) = sum(sum(im(:,:,1)))./num;
      rgb(2) = rgb(1);
      rgb(3) = rgb(1);
   else
      rgb(1) = sum(sum(im(:,:,1)))./num;
      rgb(2) = sum(sum(im(:,:,2)))./num;
      rgb(3) = sum(sum(im(:,:,3)))./num;
   end
end
```

loadTest.m

```
function [data, species] = loadTest()
   % Añadimos el directorio de imagenes al path
   folder = fileparts(which(mfilename));
   addpath(genpath(folder));
   % Leemos ficheros .jpg del directorio actual y sus subdirectorios

   dirAnterior = ' ';

   animals = ["ant", "beaver", "crab", "crayfish", "crocodile", "dolphin", "dragonfly", "elephant", "emu",
"flamingo", "kangaroo", "panda"];

   data = [];
   species = [];
   for i = 1 : length(animals)
      path = strcat('imatges_animals_80_20\test\', animals(i),'\*.jpg');
      dinfo = dir(path);
      for j = 1 : length(dinfo)
         thisfolder = dinfo(j).folder;    % carpeta del fichero
         thisfilename = dinfo(j).name;              % nombre del fichero
         filename = strcat(thisfolder,'\',thisfilename);
         species = [species,animals(i)];
         %filename
         annotationname = replace(replace(filename,'.jpg','.mat') ,'image','annotation');
         stats = getProps(filename, annotationname);
         data = [data,stats];
      end
   end

   species = categorical(species');
   data = data';
end
```

showNFourier.m

```
%% Test Fourier

function stats = showNFourier(imgfile, annotation_file, N)
   % showNFourier('crab/image_0001.jpg','crab/annotation_0001.mat',20)
   contorn = bordeIMG(imgfile,annotation_file);
   %figure(1), imshow(contorn), title('contorn original')

   [row, col] = find(contorn,1);
   B = bwtraceboundary(contorn,[row col],'N');
   cdm = mean(B);
   Bc = B-cdm;
   s = Bc(:,1)+i*Bc(:,2);
   %z = fft(s);
   tmp = fft(s);

   %tmp = z;
   tmp(N+1:end-N) = 0;

   ss2 = ifft(tmp);
   files = round(real(ss2)+cdm(1).*2);
   columnes = round(imag(ss2)+cdm(2).*2);
   aux = zeros(size(contorn).*2);
   aux(sub2ind(size(aux),files,columnes)) = 1;
   %figure(2), imshow(aux), title(sprintf('silueta reconstruida amb N = %d',N));
   stats = regionprops(aux, 'MajorAxisLength','MinorAxisLength', 'Eccentricity', 'Orientation', 'ConvexArea',
'EquivDiameter', 'Extent', 'Perimeter');
   stats = cell2mat(struct2cell(stats));
end
```

unionPuntos.m

```
function img = unionPuntos(img, X, Y)
   x0 = X(1);
   x1 = X(2);
   y0 = Y(1);
   y1 = Y(2);

   if x0 < 1
      x0 = 1;
   end
   if x1 < 1
    x1 = 1;
   end
   if y0 < 1
      y0 = 1;
   end
   if y1 < 1
      y1 = 1;
   end

   img(uint32(y0), uint32(x0)) = 1;
```

```
   img(uint32(y1), uint32(x1)) = 1;

   degree = atan2(double(y1 - y0), double(x1 - x0));
   D  = sqrt(((x1-x0).^ 2)+((y1-y0).^ 2));
   for i = 1:D
      pointx = uint32(i*cos(degree) + x0);
      pointy = uint32(i*sin(degree) + y0);
      img(pointy, pointx) = 1;
   end
   %bala->setPosition(glm::vec2(3 * cos(baladegree), 3 * sin(baladegree)) + bala->position());

end
```