Inteligencia Artificial I

Proyecto de Trabajo Final:

Clasificador de frutas basado en algoritmos K-means y K-nn

Alumno: Olguin Nahuel

Legajo: 12297

Contenido

1. Resumen 3
2. Introducción 3
3. Especificación del agente 3
4. Diseño del agente 4
5. Planificación de las tareas 9
6. Ejemplo de aplicación 10
7. Resultados 11
8. Conclusiones 12
9. Bibliografía y/o referencias 13
10. Anexo: código 14

1. Resumen

En este trabajo correspondiente a la catedra “Inteligencia Artificial I” se implementaran algoritmos que permitan usar visión artificial para clasificar objetos físicos a partir de imágenes fotográficas

Para este caso en particular, se desarrolla un sistema de reconocimiento de frutas por visión artificial con la intención de agilizar el proceso de cobro en las cajas de un supermercado. Se tomaran imágenes de bananas, naranjas, limones y tomates para formar una base de datos.

Se hará uso de los algoritmos K-means y K-nn para realizar la clasificación y a partir de los resultados se deberá sugerir el más óptimo

2. Introducción

La visión artificial, también conocida como visión por computadora (del inglés computer vision) o visión técnica, es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por un ordenador.

Tal y como los humanos usamos nuestros ojos y cerebros para comprender el mundo que nos rodea, la visión artificial trata de producir el mismo efecto para que los ordenadores puedan percibir y comprender una imagen o secuencia de imágenes y actuar según convenga en una determinada situación. Esta comprensión se consigue gracias a distintos campos como la geometría, la estadística, la física y otras disciplinas. La adquisición de los datos se consigue por varios medios como secuencias de imágenes, vistas desde varias cámaras de video o datos multidimensionales desde un escáner médico.

En este proyecto se hará uso de la visión artificial para desarrollar un sistema de reconocimiento de frutas. La comprensión de las imágenes por parte de la computadora se debe al análisis de las propiedades geométricas y en base a estas determinara de qué fruta se trata

3. Especificación del agente

Tipos de agentes:

- K-means: es un algoritmo de aprendizaje no supervisado, ya que no presenta ejemplos con etiquetas conocidas y solo relaciona los ejemplos de acuerdo a que tan similares son sus propiedades.
- K-nn: es un algoritmo de aprendizaje supervisado, ya que recibe como entradas una colección de ejemplos con etiquetas conocidas cuyas propiedades son utilizadas para compararlas con los ejemplos cuyas etiquetas son desconocidas

Tabla "REAS":

Tipo de agente	Medidas de rendimiento	Entorno	Actuadores	Sensores
Clasificador de frutas	Numero de frutas clasificadas correcta e incorrectamente	Cinta de la caja registradora, operario, cliente, cámara	Cinta transportadora	Cámara

Propiedades del entorno de trabajo:

Entorno de trabajo	Totalmente vs Parcialmente observable	Determinista vs Estocástico	Episódico vs Secuencial	Estático vs Dinámico	Discreto Vs Continuo	Agente
Cinta de la caja registradora	Totalmente observable	Determinista	Episódico	Estático	Discreto	Agente individual

4. Diseño del agente

Etapas del agente usando el algoritmo Knn

1. La primera etapa consiste en leer las imágenes de entrenamiento las cuales están separadas de acuerdo a sus respectivas etiquetas, las cuales en nuestro caso son limón, naranja, tomate y banana (Figura1)

Ilustración 1: Imágenes de entrenamiento

2. Se aplican filtros y funciones de tratamientos de imágenes a cada fotografía con el objetivos de obtener una imagen que represente solo la forma del objeto a analizar (Figura 2)

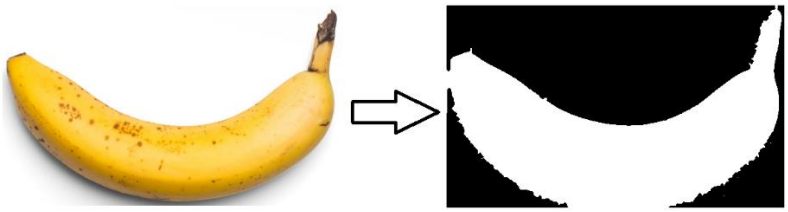


Ilustración 2 Tratamiento de imágenes

3. De esta imagen se extraen dos propiedades: la excentricidad "exc", y la relación entren el eje menor y mayor de la figura "b/a". La relación "b/a" se acercara al valor de uno cuando la imagen es completamente circular y a cero cuando se aleja de este valor, lo mismo sucede con la excentricidad pero de forma inversa (Figura 3).

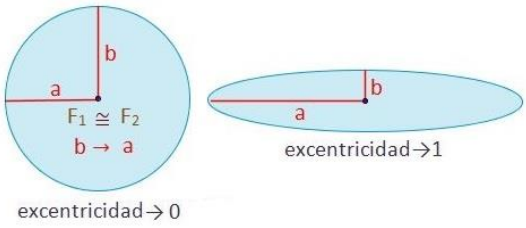


Ilustración 3 Valores de excentricidad

Estas propiedades geométricas solo dependen de la forma de la figura y no de sus dimensiones, como por ejemplo lo serían el area o el perímetro. Evitar análisis que estén afectados por la dimensión de los elementos es importante, ya que podemos clasificar independientemente de la variación de tamaño de frutas del mismo tipo

De esta forma obtenemos nuestra base de datos mostrada en la figura 4

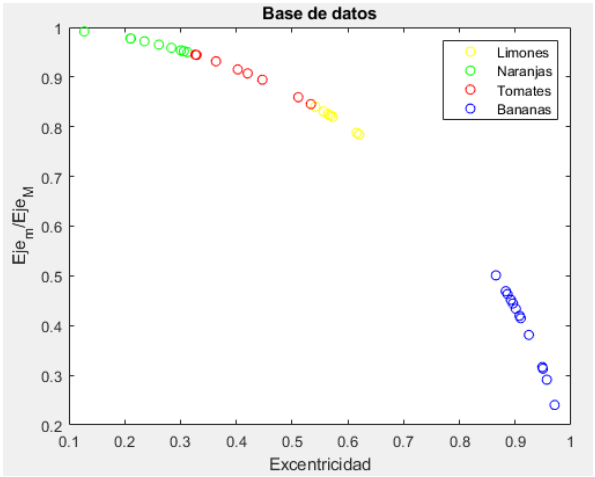


Ilustración 4 Base de datos Knn

4. Se procede a leer las imágenes de testeo que serán analizadas y evaluadas observadas en la figura 5

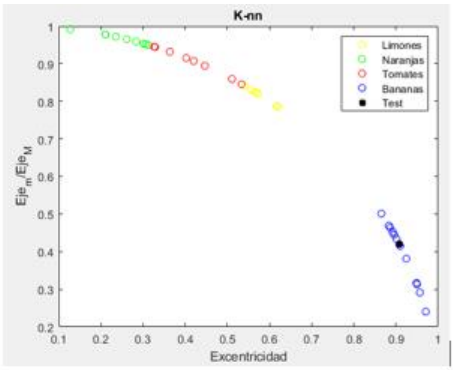
Ilustración 5 Imágenes de testeo

5. Para mejorar el rendimiento, antes de aplicar el algoritmo K-nn se hace un análisis de color en la imagen. Se analiza la imagen pixel a pixel para verificar que color predomina de acuerdo a los valores del amarillo, naranja y rojo en la escala del modelo RGB

Ilustración 6 Ejemplos de colores RGB

Si predominan de forma clara el color naranja o rojo, se le asigna al objeto la etiqueta de naranja o tomate de forma directa. Por otro lado, si predomina el color amarillo se procede a realizar el análisis geométrico con el algoritmo K-nn para identificar si se trata de limones o bananas. Finalmente, si ningún color predomina con mucha seguridad se procede a realizar el análisis completamente basado en la forma geométrica con K-nn y seleccionar entre las 4 etiquetas posibles

6. Se aplica el algoritmo K-nn en donde se obtienen las propiedades de la forma geométrica como vimos en el punto 3 y se compara con los ejemplos de entrenamiento, asignándole al objeto la etiqueta de los tres vecinos más cercanos como se observa en la figura 7 (el número de vecinos a considerar se determina a través de ensayo y error, pero siempre debe ser impar)




```
La imagen analizada es:
banana_26.jpg
Clasificacion de color fallida. Se procede a clasificar segun la forma
El objeto es una banana
```

Ilustración 7 Clasificación basada en algoritmo Knn

Etapas del agente usando el algoritmo K-means

1. En este caso se aplica un algoritmo K-means basado en prototipos, en donde los centroides iniciales de cada clase estará dada por un elemento de cada etiqueta tomado como referencia.

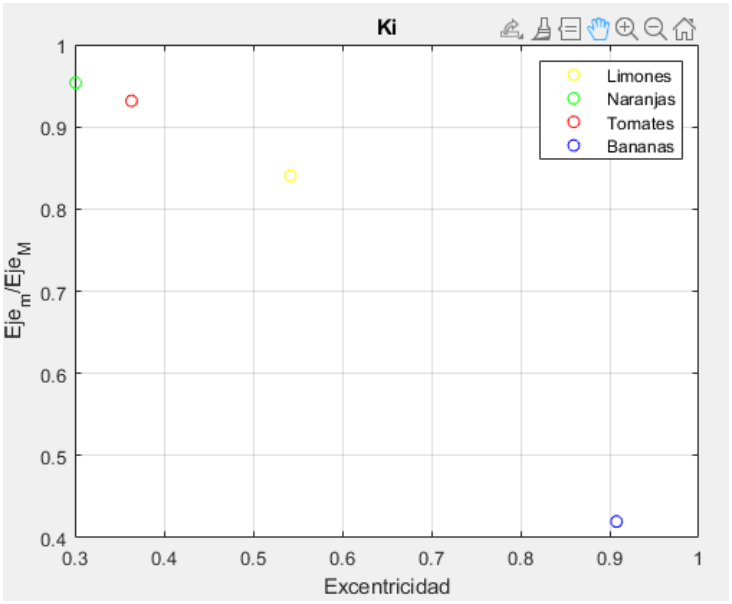


Ilustración 8 Centroides iniciales

2. Como se trata de aprendizaje no supervisado, se utilizan las imágenes de testeo para ajustar los centroides: se asigna a cada elemento la clase del nodo más cercano, se calcula la media de las propiedades de cada clase, se ajustan las posiciones de los nodos hacia las medias de cada clase y se repite el proceso una cantidad de iteraciones definida; como se observa en la figura 9

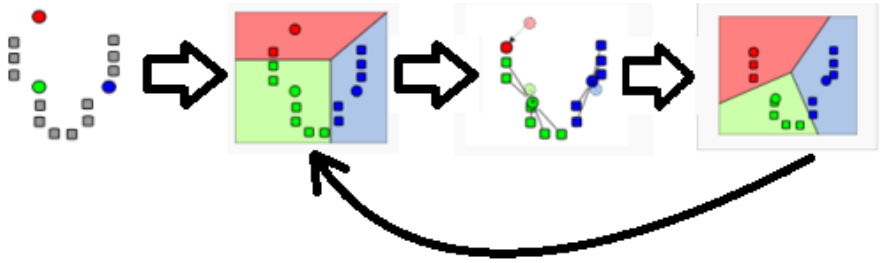


Ilustración 9 Proceso de ajuste de centroides

En nuestro caso realizamos cinco iteraciones, ya que eran suficientes para obtener un posicionamiento decente de los centroides como se observa en la figura 10

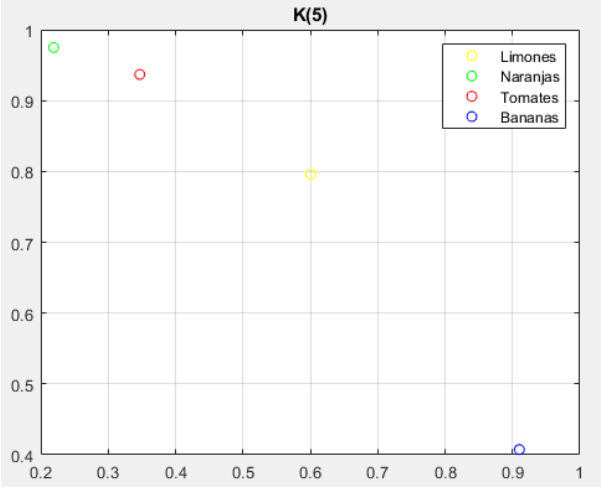
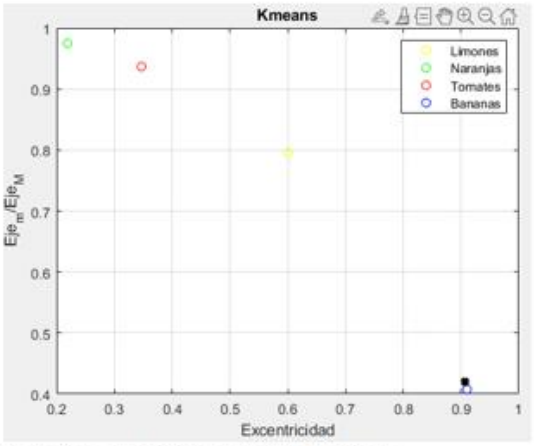


Ilustración 10 Centroides finales

3. Finalmente, se realiza una clasificación por color al igual que hicimos en el método Knn. Si este control de color no alcanza los requisitos de precisión, se procede a realizar una clasificación de acuerdo a las propiedades de la forma geométrica (las propiedades son obtenidas de la misma forma que en el método knn), en donde la clase es asignada de acuerdo al centroide más cercano al elemento analizado




```
Clasificacion de color fallida. Se procede a clasificar segun la forma
La imagen analizada es:
banana_26.jpg
El objeto es una banana
```

Ilustración 11 Clasificación basada en algoritmo K-means

5. Planificación de las tareas

Numero de etapa	Descripción	Tiempo dedicado(días)
1	Búsqueda y edición de imágenes para entrenar y testear	1
2	Aprender a implementar filtros y herramientas para obtener la forma geométrica de la imagen	2
3	Selección y obtención de características más optimas: Color, Excentricidad y “Eje_menos/Eje_mayor”	1
4	Investigación sobre los algoritmos K-nn y K-means	1
5	Diseño y programación de algoritmos	4
6	Elaboración de informe	2

De acuerdo a la planificación vemos que el tiempo total para realizar el proyecto son 11 días. Sin embargo, si el trabajo es realizado por varias personas muchas de las tareas pueden realizarse simultáneamente acortando el trabajo a, aproximadamente, 6 días

6. Ejemplo de aplicación

- **Algoritmo K-nn**

Ilustración 12 Aplicación Knn

- **Algoritmo K-means**

Ilustración 13 Aplicación K-means

7. Resultados

- **Algoritmo Knn**

  El algoritmo presenta solo error en dos de 50 imágenes por lo que presenta un rendimiento de 96%.

  Los dos errores se encuentran en la figura de naranjas:

  1. El primer error se debe a que la naranja de la figura 14, que tiene una forma que se asemeja por muy poco a los tomates y la clasifica como tal

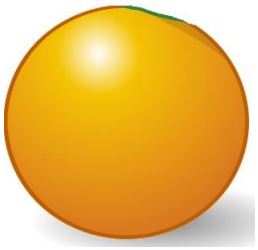


Ilustración 14 Naranja clasificada como tomate debido a su forma

  2. En el segundo error se observa que la naranja de la figura 15 presenta un color amarillento debido al reflejo de la luz, por lo que es clasificada como un limón

Ilustración 15 Naranja clasificada como limón debido al color

- **Algoritmo K-means**

  El algoritmo presento los mismos errores y en los mismos dos casos del anterior por lo que su rendimiento también es del 96%.

  Sin embargo, cabe resaltar que este método se apoya muchos más en la clasificación por colores, ya que el método K-means que se basa en la forma geométrica presenta menor precisión que el K-nn anterior. La imprecisión es más observable en los clusters cuyos centroide son más próximos, en nuestro caso serían las naranjas y los tomates

8. Conclusiones

Se puede observar que ambos algoritmos funcionan con bastante precisión y mucho más si los reforzamos con el clasificador basado en colores

Consideraciones generales:

- A la hora de fotografiar imágenes se debe tener precaución con la iluminación, de forma tal que no refleje mucho sobre la fruta y que esté ubicada de tal forma que no se observe una sombra considerable, ya que esto introduce un ruido considerable sobre los filtros y herramientas para tratar las imágenes.
- La superficie de la cinta debe ser lo más uniforme posible, sin presencia de objetos o suciedad que introduzcan ruido
- Se debe seleccionar con mucha rigurosidad las propiedades que se tendrán en cuenta a la hora de obtener las distancias euclidianas, ya que es muy probable considerar algunas que son representativas de un elemento pero no de una clase (por ej: todas las relacionadas con dimensiones, tales como el area o perímetro)

Consideraciones Knn:

- Se debe tener especial cuidado en las imágenes utilizadas para entrenar el algoritmo, se deben elegir las más representativas de cada clase
- Mientras mayor sea la cantidad de imágenes, podemos considerar mayor cantidad de vecinos a la hora de realizar la clasificación
- Recordar que la cantidad de vecinos considerados debe ser siempre impar para evitar posibles conflictos de igualdad

Consideraciones K-means:

- Se debe tener especial cuidado en los prototipos utilizados para establecer los centroides iniciales ya que estos serán determinantes en el rendimiento final del algoritmo
- El ruido introducido en este método afecta en mayor medida el rendimiento con respecto al anterior, ya que este ruido puede desplazar el centroide en gran medida sacándolo de su posición optima
- La cantidad de iteraciones para posicionar los centroides deben realizarse hasta que los mismos presenten desplazamientos despreciables

9. Bibliografía y/o referencias

- Apuntes de la cátedra Inteligencia Artificial 1
- Documentación de “Image Processing Toolbox MATLAB”
- Visión Artificial: https://es.wikipedia.org/wiki/Visi%C3%B3n_artificial
- Teoría de K-means: https://www.youtube.com/watch?v=YcJF94-ht74
- Teoría de Knn: https://www.youtube.com/watch?v=eOhTuHAU5Uk

10. Anexo: código

```
clc;
clear all;
close all;

%Leemos directorios de las imagenes de entrenamiento
RUTA = pwd;    %pwd devuelve la ruta a la carpeta actual.
%dir enumera los archivos y carpetas de la carpeta actual.
%strcat: concatena la RUTA con lo que sigue, para completar la direccion
DirLi = dir(strcat(RUTA,'\Datasets\train_zip\train\Limon\*.jpg'));
DirNa = dir(strcat(RUTA,'\Datasets\train_zip\train\Naranja\*.jpg'));
DirTo = dir(strcat(RUTA,'\Datasets\train_zip\train\Tomate\*.jpg'));
DirBa = dir(strcat(RUTA,'\Datasets\train_zip\train\Banana\*.jpg'));
%Leemos directorios test
Dirtest = dir(strcat(RUTA,'\Datasets\test_zip\test\*.jpg'));
%Sacamos las propiedades de los elementos del directorio de entrenamiento
%Los numeros 1,2,3 y 4 son las etiquetas correspondientes a cada clase
PropBa = PropClase(DirBa,4);
PropLi = PropClase(DirLi,1);
PropNa = PropClase(DirNa,2);
PropTo = PropClase(DirTo,3);

Formas_M = [PropLi PropNa PropTo PropBa];

%  Metodo a utilizar: 1 Knn y 2 Kmeans
metodo = 2;
```

```
%  Knn (metodo = 1)
if metodo == 1
    %  Obtenemos el archivo de prueba
    for test = 1:length(Dirtest)
        nombre = Dirtest(test).name;
        ruta = (strcat(RUTA,'\Datasets\test_zip\test\'));
        Imgtest = imread(strcat(ruta,nombre));
        % Obtenemos las propiedades de forma y color de la imagen
        f=figure; imshow(Imgtest);set(gcf,'Position',[0 300 500 300]); hold on;
        Forma = PropForma(Imgtest);
        Color = PropColor(Imgtest);
        % Aplicamos el metodo Knn
        knn_test = KNN(Formas_M,Forma,Color,Dirtest(test));
        %Graficamos
            figure;
            plot(PropLi(1,:),PropLi(2,:),'yo');hold on;
            plot(PropNa(1,:),PropNa(2,:),'go');hold on;
            plot(PropTo(1,:),PropTo(2,:),'ro');hold on;
            plot(PropBa(1,:),PropBa(2,:),'bo');hold on;
            plot(Forma(1,:),Forma(2,:),'kx','linewidth',3);hold on;
            legend('Limones','Naranjas','Tomates','Bananas','Test');
            xlabel('Excentricidad');ylabel('Eje_m/Eje_M');
            title('K-nn');
        pause; clc; close all;
    end
end
%  Kmeans (metodo = 2)
if metodo == 2
%Sacamos las propiedades de los elementos de test, colocamos etiqueta "0"
%a todos indicando como tipo indefinido
PropTest=PropClase(Dirtest,0);
% Referencias de clusters. usando como prototipos iniciales las primeras
% sacamos las proiedades a las imagenes de prueba
PropTest=PropTest(1:2,:);
[M,N] = size(PropTest);
%R:referencias a cada clase
 R=zeros(M,4);
 R(:,1)=PropLi(1:2,1);
 R(:,2)=PropNa(1:M,1);
 R(:,3)=PropTo(1:M,1);
 R(:,4)=PropBa(1:M,1);
 %Graficamos los Ki
    figure;
    plot(R(1,1),R(2,1),'yo');hold on;
    plot(R(1,2),R(2,2),'go');hold on;
    plot(R(1,3),R(2,3),'ro');hold on;
    plot(R(1,4),R(2,4),'bo');hold on;
    grid on
    legend('Limones','Naranjas','Tomates','Bananas');
    xlabel('Excentricidad');ylabel('Eje_m/Eje_M');
    title('Ki');
```

```
R=Kmeans_nodos(R,PropTest,M,N);

for test = 1:length(Dirtest)
        nombre = Dirtest(test).name;
        ruta = (strcat(RUTA,'\Datasets\test_zip\test\'));
        Imgtest = imread(strcat(ruta,nombre));
    % Imagen a color
        f=figure; imshow(Imgtest);set(gcf,'Position',[0 300 500 300]); hold on;
        Forma = PropForma(Imgtest);
        Color = PropColor(Imgtest);
        %Aplicamos el algoritmo
        Kmeans_test = Kmeans(Forma, Color, R, M,Dirtest(test));
        pause; clc; close all;
    end
end
function Prop = PropClase(dir,tipo)
    %Recorremos uno a uno los archivos dentro de la carpeta
    for i = 1:length(dir)
        %almacenamos el nombre del archivo imagen
        Nombre = dir(i).name;
        %almacenamos la ruta de la carpeta que almacena el archivo imagen
        Ruta = dir(i).folder;
        %Leemos la imagen
        Img_Entrenamiento = imread(strcat(Ruta,'\',Nombre));
        %figure; imshow(strcat(Ruta,'\',Nombre)); hold on; pause(1);
        disp('Entrenando al algoritmo con imagen')
        disp(Nombre)

        %Obtenemos las propiedades de las imagenes de entrenamiento
        Prop(:,i) = [PropForma(Img_Entrenamiento);tipo];

        %Limpiamos la pantalla
        close all; clc;
    end
end
```

```
function Prop = PropForma(Img_color)
%Imagen en escala de grises
    Img_gris = rgb2gray(Img_color);
    %figure; imshow(Img_g); hold on; pause(3);

%Filtro ft
    % fspecial crea un filtro bidimensional
    %unsharp: filtro mejora de contraste (bordes y los detalles finos en la imagen sean más nítidos)
    ft = fspecial('unsharp');
    %aplicamos el filtro a la imagen
    I = imfilter(Img_gris,ft);
    %figure; imshow(I); hold on; pause(1);
```

```
%Deteccion de forma
    %Detección de bordes.
    %Detecta bordes en los puntos en los que el gradiente I es máximo
    [~,threshold] = edge(I,'prewitt');
    %threshold: Umbral de sensibilidad,ignora todos los bordes cuya intensidad no es mayor que threshold
    fudgeFactor = 0.3;
    BWs = edge(I,'sobel',threshold * fudgeFactor);
    se90 = strel('line',4,90); se0 = strel('line',4,0);
    BWsdil = imdilate(BWs,[se90 se0]);
    %figure; imshow(BWsdil) ;title('Dilated Gradient Mask'); pause(3);
    BWdfill = imfill(BWsdil,'holes');
    %figure; imshow(BWdfill) ;title('Binary Image with Filled Holes'); pause(1);
    BWnobord = imclearborder(BWdfill,1);
    %figure; imshow(BWnobord) ;title('Cleared Border Image')
    seD = strel('disk',1); BWfinal = imerode(BWnobord,seD); BWfinal = imerode(BWfinal,seD);
    %figure; imshow(BWfinal) ;title('Segmented Image'); pause(5)

%Propiedades de la figura extraida
    stats = regionprops(BWfinal,'Eccentricity','Area','MajorAxisLength','MinorAxisLength','Perimeter');
    [~,Ind] = max([stats.Area]);
    exce = [stats.Eccentricity];
    exce = exce(Ind);
    eje_M = [stats.MajorAxisLength];
    eje_M = eje_M(Ind);
    eje_m = [stats.MinorAxisLength];
    eje_m = eje_m(Ind);
    Prop = [exce;eje_m/eje_M];
end
```

```
function C = PropColor(Img_color)
    r = Img_color(:,:,1);
    g = Img_color(:,:,2);
    b = Img_color(:,:,3);
    [M,N] = size(r);
    Na = [255, 128, 0];
    Li = [255, 225, 53];
    To = [161, 35, 18];
    CNa = 0;
    CLi = 0;
    CTo = 0;
```

```
% Tolerancia de color
    tol = 10;
    for j=1:N
        for i=1:M
            if r(i,j)<Na(1)+tol && r(i,j)>Na(1)-tol
                if g(i,j)<Na(2)+tol && g(i,j)>Na(2)-tol
                    if b(i,j)<Na(3)+tol && b(i,j)>Na(3)-tol
                        CNa = CNa + 1;
                    end
                end
            end
            if r(i,j)<Li(1)+tol && r(i,j)>Li(1)-tol
                if g(i,j)<Li(2)+tol && g(i,j)>Li(2)-tol
                    if b(i,j)<Li(3)+tol && b(i,j)>Li(3)-tol
                        CLi = CLi + 1;
                    end
                end
            end
            if r(i,j)<To(1)+tol && r(i,j)>To(1)-tol
                if g(i,j)<To(2)+tol && g(i,j)>To(2)-tol
                    if b(i,j)<To(3)+tol && b(i,j)>To(3)-tol
                        CTo = CTo + 1;
                    end
                end
            end
        end
    end
    if CNa > CLi && CNa > CTo
        C = 2;
    elseif CLi > CNa && CLi > CTo
        C = 1;
    elseif CTo > CLi && CTo > CNa
        C = 3;
    else
        C=0;
    end
end
```

```
function out = KNN(Formas_M,Forma,Color,Dir)
% Comparacion de propiedades de forma
    out = 0;
    [M,N] = size(Formas_M);
    %Armamos un vector dist que almacena las distancias a cada elemento de
    %entrenamiento (primera fila) e indicando la etiqueta del mismo (segunda fila)
    dist = zeros(2,N);

    for j = 1:N
        sum = 0;
        for i = 1:M-1
            sum = sum + (Formas_M(i,j)-Forma(i,1))^2;
        end
        dist(1,j)=sqrt(sum);
        dist(2,j)= Formas_M(M,j);
    end
    [~,s] = sort(dist(1,:));
```

```
%cantidad de vecinos a considerar 3
K=3;
limon = 0;
naranja = 0;
tomate = 0;
banana = 0;
    for i=1:K
        j = s(i);
        if dist(2,j)==1
            limon = limon+1;
        end
        if dist(2,j)==4
            banana = banana+1;
        end
        if dist(2,j)==3
            tomate = tomate+1;
        end
        if dist(2,j)==2
            naranja = naranja+1;
        end
    end
    %conclusion de la clasificacion
    disp('La imagen analizada es:'); disp(Dir.name);
    if Color == 1
        if banana>limon && banana>tomate && banana>naranja
            disp('El objeto es una banana');
        else
            disp('El objeto es un limon');
        end
    end
    if Color == 2
        disp('El objeto es una naranja');
    end
    if Color == 3
        disp('El objeto es un tomate');
    end
    if Color == 0
        disp('Clasificacion de color fallida. Se procede a clasificar segun la forma');
        if banana>limon && banana>tomate && banana>naranja
            disp('El objeto es una banana');
        elseif limon>banana && limon>tomate && limon>naranja
            disp('El objeto es un limon');
        elseif tomate>banana && tomate>limon && tomate>naranja
            disp('El objeto es un tomate');
        elseif naranja>banana && naranja>limon && naranja>tomate
            disp('El objeto es un naranja');
        end
    end
end
```

```
function out = Kmeans(Forma, Color, R, M, Dir)
out=0;
    % Comparamos con la imagen de testeo
    suml = 0;
    sum2 = 0;
    sum3 = 0;
    sum4 = 0;
    for i = 1:M
        suml = suml + (Forma(i)-R(i,1))^2;
        sum2 = sum2 + (Forma(i)-R(i,2))^2;
        sum3 = sum3 + (Forma(i)-R(i,3))^2;
        sum4 = sum4 + (Forma(i)-R(i,4))^2;
    end
    d(1)=sqrt(suml);
    d(2)=sqrt(sum2);
    d(3)=sqrt(sum3);
    d(4)=sqrt(sum4);
%Graficamos
figure;
plot(R(1,1),R(2,1),'yo');hold on;
plot(R(1,2),R(2,2),'go');hold on;
plot(R(1,3),R(2,3),'ro');hold on;
plot(R(1,4),R(2,4),'bo');hold on;
plot(Forma(1),Forma(2),'kx','linewidth',3);hold on;
grid on
legend('Limones','Naranjas','Tomates','Bananas');
xlabel('Excentricidad');ylabel('Eje_m/Eje_M');
title('Kmeans');

if (d(1)<d(4) || d(1)==0) && Color == 1 %Limon
    disp('La imagen analizada es:'); disp(Dir.name);
    disp('El objeto es un limon');
end
if (d(4)<d(1) || d(4)==0) && Color == 1 %Banana
        disp('La imagen analizada es:'); disp(Dir.name);
        disp('El objeto es una banana');
end
if Color == 2 %Naranja
    disp('La imagen analizada es:'); disp(Dir.name);
    disp('El objeto es una naranja');
end
```

```
    if Color == 3 %Tomate
        disp('La imagen analizada es:'); disp(Dir.name);
        disp('El objeto es un tomate');
    end
    if Color == 0 %Color no definido
        disp('Clasificacion de color fallida. Se procede a clasificar segun la forma');
        if d(1)<d(2) && d(1)<d(3) && d(1)<d(4) %se le asigna limon
            disp('La imagen analizada es:'); disp(Dir.name);
            disp('El objeto es un limon');
        end
        if d(2)<d(1) && d(2)<d(3) && d(2)<d(4) %Naranja
            disp('La imagen analizada es:'); disp(Dir.name);
            disp('El objeto es una naranja');
        end
        if d(3)<d(2) && d(3)<d(1) && d(3)<d(4) %Tomate
             disp('La imagen analizada es:'); disp(Dir.name);
             disp('El objeto es un tomate');
        end
        if d(4)<d(2) && d(4)<d(3) && d(4)<d(1) %Banana
             disp('La imagen analizada es:'); disp(Dir.name);
             disp('El objeto es una banana');
        end
    end
end
```

```
function R = Kmeans_nodos(R,PropTest, M, N)
%Iteraciones para aproximar los centroides
    iteraciones = 10;
%d:distancia de un elemento a cada R de cada clase
    d=zeros(4);
    %dd: asignacion de etiqueta de acuerdo a la distancia al R
    dd1=zeros(M,N);
    dd2=zeros(M,N);
    dd3=zeros(M,N);
    dd4=zeros(M,N);
for n=1:iteraciones
        NumLi=0;
        NumNa=0;
        NumTo=0;
        NumBa=0;
        for j = 1:N
            sum1 = 0;
            sum2 = 0;
            sum3 = 0;
            sum4 = 0;
            for i = 1:M
                sum1 = sum1 + (PropTest(i,j)-R(i,1))^2;
                sum2 = sum2 + (PropTest(i,j)-R(i,2))^2;
                sum3 = sum3 + (PropTest(i,j)-R(i,3))^2;
                sum4 = sum4 + (PropTest(i,j)-R(i,4))^2;
            end
```

```
            d(1)=sqrt(sum1);
            d(2)=sqrt(sum2);
            d(3)=sqrt(sum3);
            d(4)=sqrt(sum4);
            if (d(1)<d(2) && d(1)<d(3) && d(1)<d(4))|| d(1)==0 %se le asigna limon
                NumLi=NumLi+1;
                dd1(:,NumLi) = PropTest(1:M,j); % guardamos las propiedades de limon
            end
            if (d(2)<d(1) && d(2)<d(3) && d(2)<d(4)) || d(2)==0 %Naranja
                NumNa=NumNa+1;
                dd2(:,NumNa) = PropTest(1:M,j);
            end
            if (d(3)<d(2) && d(3)<d(1) && d(3)<d(4)) || d(3)==0 %Tomate
                NumTo=NumTo+1;
                dd3(:,NumTo) = PropTest(1:M,j);
            end
            if (d(4)<d(2) && d(4)<d(3) && d(4)<d(1)) || d(4)==0 %Banana
                NumBa=NumBa+1;
                dd4(:,NumBa) = PropTest(1:M,j);
            end
        end
        sum = zeros(M,4);
        for t=1:NumLi
            sum(:,1) = sum(:,1)+dd1(:,t);
        end
        for t=1:NumNa
            sum(:,2) = sum(:,2)+dd2(:,t);
        end
        for t=1:NumTo
            sum(:,3) = sum(:,3)+dd3(:,t);
        end
        for t=1:NumBa
            sum(:,4) = sum(:,4)+dd4(:,t);
        end
% Centroides obtenidos para cada cluster
        R(:,1)=sum(:,1)/NumLi;
        R(:,2)=sum(:,2)/NumNa;
        R(:,3)=sum(:,3)/NumTo;
        R(:,4)=sum(:,4)/NumBa;
end
end
```