Datos Personales:

Integrantes:

- Trejo Miranda Miguel Ariel
- Rodríguez López Óscar Gabriel
- Páez Ortega Oswaldo Emmanuel

Grupo: 2TM11

Número de departamental: Segundo departamental.

Marco teórico

K-means:

K-means es un algoritmo de clasificación no supervisada (*clusterización*) que agrupa objetos en *k* grupos basándose en sus características. El agrupamiento se realiza minimizando la suma de distancias entre cada objeto y el centroide de su grupo o *cluster*. Se suele usar la distancia cuadrática.

El algoritmo consta de tres pasos:

1. **Inicialización:** una vez escogido el número de grupos, *k*, se establecen *k* centroides en el espacio de los datos, por ejemplo, escogiéndolos aleatoriamente.
2. **Asignación objetos a los centroides:** cada objeto de los datos es asignado a su centroide más cercano.
3. **Actualización centroides:** se actualiza la posición del centroide de cada grupo tomando como nuevo centroide la posición del promedio de los objetos pertenecientes a dicho grupo.

Se repiten los pasos 2 y 3 hasta que los centroides no se mueven, o se mueven por debajo de una distancia umbral en cada paso. El algoritmo *k-means* resuelve un problema de optimización, siendo la función para optimizar (minimizar) la suma de las distancias cuadráticas de cada objeto al centroide de su cluster.

Los objetos se representan con vectores reales de *d* dimensiones ($x_1, x_2, \ldots, x_n$) y el algoritmo *k-means* construye *k* grupos donde se minimiza la suma de distancias de los objetos, dentro de cada grupo $S=\{S_1, S_2, \ldots, S_k\}$, a su centroide. El problema se puede formular de la siguiente forma:

$$\min_{\mathbf{S}} E(\boldsymbol{\mu_i}) = \min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x_j} \in S_i} \|\mathbf{x_j} - \boldsymbol{\mu_i}\|^2 \quad (1)$$

donde S es el conjunto de datos cuyos elementos son los objetos x_j representados por vectores, donde cada uno de sus elementos representa una característica o atributo. Tendremos *k* grupos o *clusters* con su correspondiente centroide μ_i

En cada actualización de los centroides, desde el punto de vista matemático, imponemos la condición necesaria de extremo a la función *E*(***μi***) que, para la función cuadrática (1) es

$$\frac{\partial E}{\partial \boldsymbol{\mu}_i} = 0 \implies \boldsymbol{\mu}_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

y se toma el promedio de los elementos de cada grupo como nuevo centroide.

Las principales ventajas del método *k-means* son que es un método sencillo y rápido. Pero es necesario decidir el valor de *k* y el resultado final depende de la inicialización de los centroides. En principio no converge al mínimo global sino a un mínimo local.

Método de la montaña:

El método de la Montaña está basado en crear una rejilla del espacio dimensional sobre el cual se encuentran los datos , las intersecciones de las líneas de la rejilla representan nodos candidatos a centros de cúmulo, cabe resaltar que al contar con una rejilla con una distancia entre intersecciones más pequeña, la aproximación tiene una mayor exactitud; sin embargo se requiere de un mayor tiempo de procesamiento, por lo cual no es recomendable manejar distancias muy pequeñas.

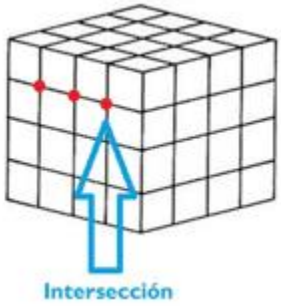

Figura 1. Ejemplificación de la rejilla de un espacio tridimensional.

El siguiente paso consiste en construir una función montaña basada en la distribución de los datos cuyo valor máximo representa el primer centro del cúmulo donde se concentra la mayor cantidad de datos. El conjunto *n* de datos a analizar se representan por:

$$X = \{x_1, x_2, \dots, x_n\} \dots (1)$$

La siguiente ecuación crea la montaña:

$$M(N_i) = \sum_{K=1}^{n} e^{-\propto d(x_k, N_i)}, \dots (2)$$

donde α es una constante positiva y *d* es la distancia euclidiana entre cada dato y cada intersección de la rejilla. El primer centro de cúmulo calculado está definido por la ecuación:

$$M_1^* = Max_i[M(N_i)]. \ldots (3)$$

Después de esto, deben obtenerse los demás centros de cúmulo, por lo cual se deben de eliminar los efectos del centro de cúmulo que fue encontrado previamente. Para esto, se resta a la primera función de montaña calculada la matriz de las distancias del punto con el valor más alto con respecto a cada uno de los cruces de la rejilla, ésta última multiplicada por el valor máximo de la función de la montaña anterior.

Lo anterior está dado por la siguiente ecuación:

$$\widehat{M}^k(N_i) = max\left[\widehat{M}^{k-1}(N_i) - M_{k-1}^* \sum_{k=1}^{n} e^{-\beta d\left(N_{k-1}^*, N_i\right)}, 0\right]. \ldots (4)$$

El paso anterior se repite hasta encontrar el número de centros de cúmulo deseados o bien hasta que una condición de paro sea cumplida, dicha condición está dada por el cociente que resulta de dividir el primer valor máximo encontrado entre el penúltimo valor máximo encontrado, es decir, el valor del primer centro de cúmulo entre el penúltimo obtenido, este cociente debe ser menor a una cierta constante positiva δ la Ecuación 5 muestra la condición de paro.

$$\frac{M_1^*}{M_{k-1}^*} < \delta \qquad \ldots (5)$$

En el ejemplo mostrado más adelante los parámetros utilizados fueron: $\delta = 1,\ \alpha = 7, \beta = 20$ y la resolución de la rejilla es de 83=512 puntos.

Problemática:

A partir de una imagen con al menos 3 objetos contrastantes con el fondo, binarizarla y aplicar los métodos:

-Montaña o substractivo para inicializar número de clústeres y posición inicial de centros.

-K-means o C-menas para optimizar la posición de los centros.

El sistema deberá asignar la pertenencia de cada pixel de los objetos a determinado cúmulo, de tal forma que, si se pide presentar en pantalla únicamente a cierto clúster, se aprecia alguno de los objetos.

Desarrollo:

Para el desarrollo del examen, se utilizó la imagen, de 240x 320 pixeles, mostrada en la figura 2, en la cual se pueden observar tres objetos que contrastas con el fondo, lo cual es requerido por las instrucciones de este examen.

Figura 2. Imagen analizada para la realización del examen.

Cabe mencionar que los parámetros que se utilizaron para el algoritmo de montaña fueron:

- α=1
- β=0.1
- $\delta=0$
- Separación entre los puntos de la regilla:

Y para K-means:

- Número de clusters:3
- Valores iniciales de los clústeres: los de salida de la función de montaña.

A continuación, se puede observar el código utilizado para la realización de dicho examen departamental:

```
%Leer la imagen
color = imread('imagenE3.jpg');
%para ver la imagen
figure
imshow(color);
%esacala de grises
gris = rgb2gray(color);
figure
% imshow(gris);
% %Reducir el tamaño
 imagen = imresize(gris,[240,320]);
 imshow(imagen)
%Desprescieando el fondo blanco
[x y] = find(imagen<180);
x = x';
y = y';
c = 4;
ejeX = 0:3:240;
ejeY = 0:3:320;
alpha = 01;
beta = 0.1;
delta = 0;
k = 1;
```

```
% %
for i = 1:length(ejeX)
    for j = 1:length(ejeY)
        for h = 1:length(x)
            D = sqrt( (ejeX(i)-x(h))^2 + (ejeY(j)-y(h))^2);
            aux(h) = exp(-alpha*D);
        end
        M(j,i) = sum(aux);
    end
end
M;
disp('Mensaje 1')
% %
M1(k) = max(max(M));
while (delta < 1) || (k < 4)
    for i = 1:length(ejeX)
        for j = 1:length(ejeY)
            if M(j,i) == M1(k)
                X1(k) = i;
                Y1(k) = j;
            end
        end
    end
    k
    M(Y1(k),X1(k))
    EX=ejeX(X1);
    EY=ejeY(Y1);
    for i = 1:length(ejeX)
        for j = 1:length(ejeY)
            for n = 1:k
                D = sqrt( ( ejeX(i)-ejeX(X1(n)) )^2 + ( ejeY(j)-
ejeY(Y1(n)) )^2 );
                aux(n) = exp(-beta*D);
            end
            aux2 = M(j,i) - M1(k)*sum(aux);
            M(j,i) = aux2;
            M(j,i) = max(M(j,i),0);
        end
    end
    k = k + 1;
    M1(k) = max(max(M(:,:)));
    delta = M1(1)/M1(k);
end
hold on
plot(EY,EX,'r*','linewidth',2)
title('Algoritmo de montaña')
xlabel(' X')
ylabel(' Y')
figure
[X,Y] = meshgrid(ejeX,ejeY);
% mat_scala = zeros(size(M));
[tam_x tam_y] = size(M);
% mat_scala = [repmat(1,26,tam_y); repmat(0.8,26,tam_y);
repmat(0.6,26,tam_y); repmat(0.4,29,tam_y)];
% M = M .* mat_scala;
mesh(X,Y,M)
```

```
%Empieza kmeans
% Determinación de número de clusters
[y x] = find(imagen<180);
P = [x y];
datos = length(x);
no_clust = 4;
A = 4;
B = 2;
C = [EX; EY]'
U = inicial(no_clust,datos);
int = 0;

fig = [];
nf = 1;
li = 1;
for i=2:length(x)
    dif = x(i)-x(i-1);
    if dif > 3
        fig(nf,:) = [li,i-1];
        li = i;
        nf = nf + 1;
    end
end
fig(nf,:)= [li,length(x)];
% Se itera con base a un parámetro epsilon
epocas = 20;
for ciclo=1:3
    P = [x(fig(ciclo,1):fig(ciclo,2)) y(fig(ciclo,1):fig(ciclo,2))];
    datos = length(P);
    no_clust = 1;
    A = 1;
    B = 2;
    C = [EX(ciclo); EY(ciclo)]'
    U = inicial(no_clust,datos);
    int = 0;
while 1
    ============= Calculo de Distancias ==============
    int = int+1;
    D = [];
    Ji = 0;
    for i=1:no_clust
        for j=1:datos
            d = 0;
            for k=1:B
                d = d+(P(j,k)-C(i,k))^2;
            end
            D(i,j) = sqrt(d);
        end
        Ji = [Ji sum(D(i,:))]; %Suma las distancias de los datos a
un clouster
    end
```

```
============= Calculo de la U ==============
    for i=1:A
        k = strfind(D(:,i)',min(D(:,i)));
        U(:,i) = 0;
        U(k,i) = 1;
    end
    ================= Calculo de J ==========
    J = sum(Ji); %suma todas las distancias hacia todos los clouster
    ============== Actualizacion de Clouster ====================
    for i=1:no_clust
        for j=1:B
            C(i,j) = sum(U(i,:).*P(:,j)')/sum(U(i,:));
        end
    end
    C;
    %pause();

    if int==epocas
        break;
    end


end
    CF(ciclo,:) = C;
end

C=CF;

figure()
imshow(imagen);
hold on
plot(C(:,1), C(:,2), '*r', 'LineWidth', 2);
title('Algoritmo de montaña con Kmeans.')
xlabel(' X')
ylabel(' Y')

%Finaliza K-means

% %
%Seleccionar un cluster

promt = 'Seleccione el cluster que quiere mostrar: ';
clus =  input(promt);

while 1
    promt = 'Seleccione el cluster que quiere mostrar: ';
    clus =  input(promt);
    figure()
    imshow(imagen);
    hold on
    plot(x(fig(clus,1):fig(clus,2)), y(fig(clus,1):fig(clus,2)),'*')
end
```

También se implementó la siguiente función, llamada “inicial”, para iniciar el parámetro U de manera aleatoria:

```
function U = inicial(Ncloster,Ndatos)

h=rand(Ncloster,Ndatos);

for i=1:Ndatos
    %k=max(h(:,i))
    k=strfind(h(:,i)',max(h(:,i)));
    h(:,i)=0;
    h(k,i)=1;
end
U = h;
end
```

Resultados:

Después de procesar la imagen, teniéndola en blanco y negro, se observa en la siguiente figura, dicha imagen, pero con los centros de los clústeres, después de haber utilizado el algoritmo de montaña.

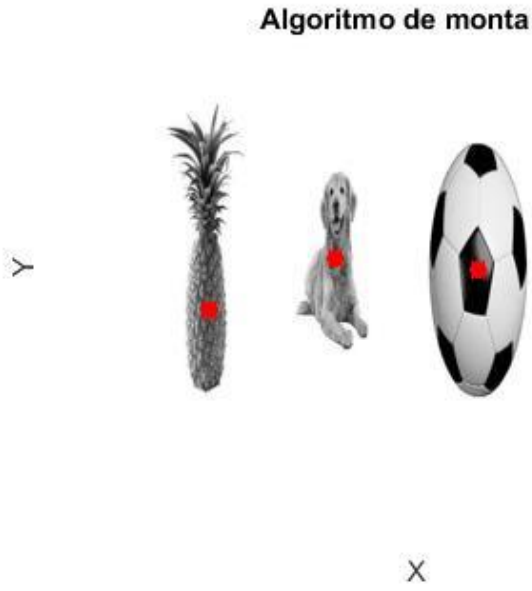

Figura 3. Centros de clúster obtenidos con el algoritmo de montaña sobra la imagen procesada.

Después de haber obtenido los clústeres con el algoritmo de montaña, se toman como base para utilizar el algoritmo de K-means, sobre la imagen procesada, como se observa a continuación:

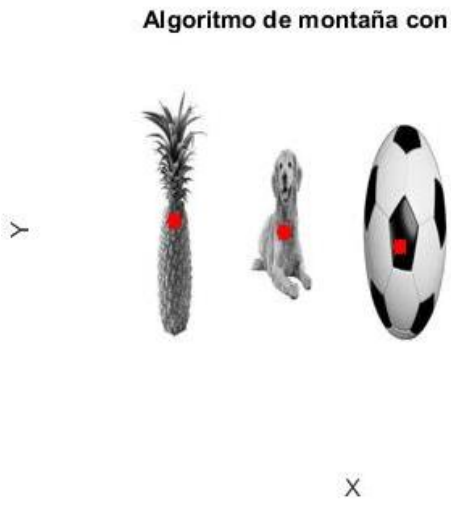

Figura 4. Centros de clústeres obtenido con K-means después del algoritmo de montaña.

La superficie de salida que se obtiene se puede observar a continuación en las siguientes dos figuras:

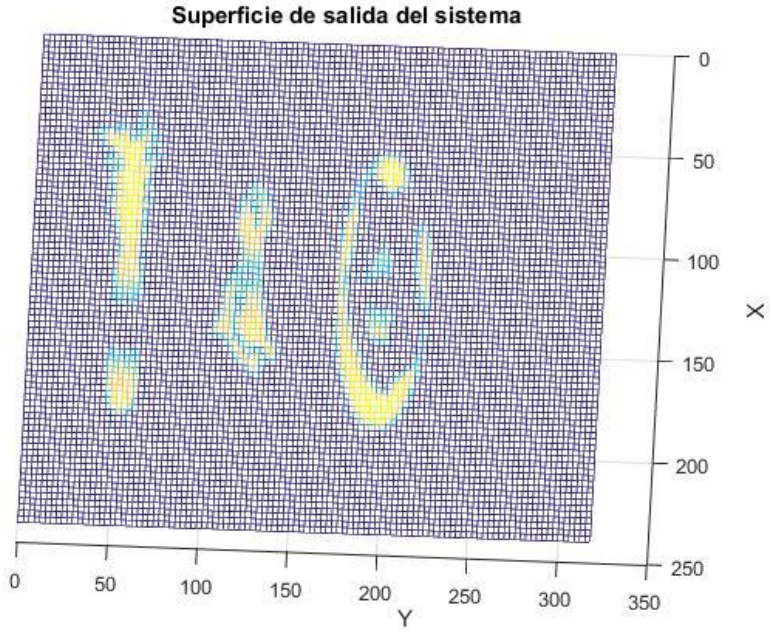

Figura 5. Superficie de salida del sistema en 2D.

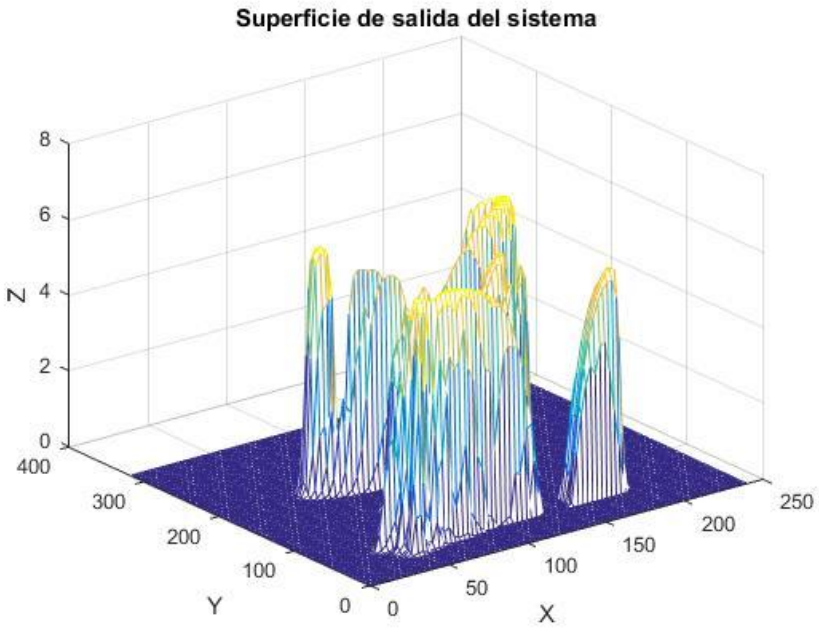

Figura 6. Superficie de salida del sistema en 3D.

Después, para fines del examen, se asigno la pertenencia de los pixeles, para que el usuario pudiera realizar la selección de un clúster en particular y pueda observar todos los pixeles asociados a dicho clúster.

Si se selecciona el clúster número uno, se obtiene:

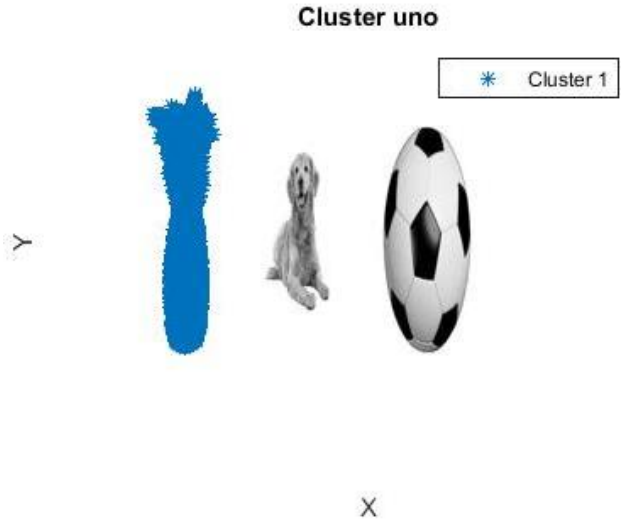

Figura 7. Selección del clúster uno.

Mientras que si se selecciona el clúster 2:

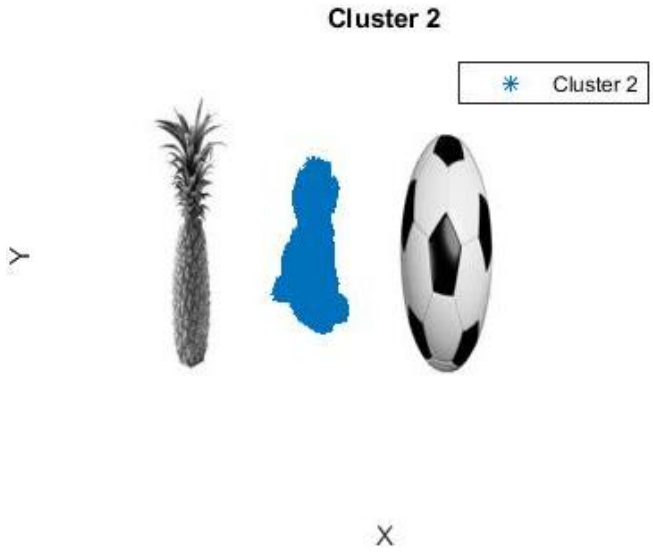

Figura 8. Selección del clúster dos.

Y por último para el clúster 3:

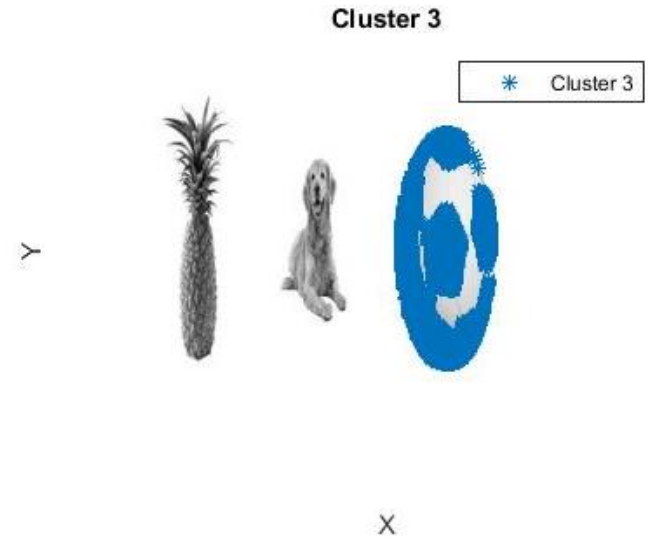

Figura 9. Selección del clúster tres.

Conclusiones:

El objetivo de este examen era identificar objetos en una imagen a partir de algoritmos de lógica difusa. El objetivo se logró mediante 2 algoritmos:

- Método de montaña.
- *K-means*

En ambos casos se tuvieron que estar variando parámetros para lograr el resultado deseado. Para el método de la montaña se estuvieron variando los parámetros de la distribución. Una conclusión a la que se llegó es la siguiente

$$0 < \alpha \leq 1.2 \quad \ldots \quad (6)$$

$$0 < \beta < \frac{\alpha}{10} \quad \ldots (7)$$

Cuando estas restricciones no se cumplían, el algoritmo de la montaña no funcionaba ya que los puntos ya sea que los localizaba fuera de toda figura o todos en una figura. Sin embargo, se obtuvo esta conclusión y se obtuvieron dichas restricciones de manera experimental.

Por otro lado, para el algoritmo de *K-means* no hubo gran complicación ya que los parámetros fueron bien definidos, tal como se vio en clase, por lo que la implementación de dicho algoritmo no fue problema.

Para finalizar, escoger la figura según el usuario, solo bastó con hacer uso de índices de la matriz de la imagen para seleccionar el objeto tal y como se muestran en las figuras 7, 8 y 9.