# Trabajo Final Inteligencia Artificial I – año 2019

# Sistema de Clasificación de Piezas Metálicas por Visión Artificial

Marcelo E Mellimaci

Facultad de Ingeniería, Universidad Nacional de Cuyo

Inteligencia Artificial 1

Dra. Ing. Selva S. Rivera

Mayo de 2021

## Contenido

## Resumen

Este proyecto sobre visión artificial fue realizado como trabajo final de la cátedra Inteligencia Artificial 1 de la Facultad de Ingeniería de la Universidad Nacional de Cuyo

El objetivo del presente trabajo consiste en obtener un agente que bajo un entorno con condiciones controladas sea capaz de clasificar piezas de ferretería (arandelas, clavos, tornillos y tuercas) haciendo uso de imágenes

Se hizo un pre procesamiento de las fotografías con la finalidad de eliminar ruido y acentuar características para la extracción de las mismas. Dichas características fueron usadas para el reconocimiento de las diferentes piezas metálicas a través de los algoritmos K-means y K-nn

Para la implementación de los algoritmos se empleó MATLAB Online, cuya versión de prueba (shareware) puede ser usada de forma gratuita a través de Internet

Por último, se analizaron los resultados y se evaluó el rendimiento de ambos algoritmos

## Introducción

### Tipo de Agente

- Es un agente *Racional*:

En cada posible secuencia de percepciones, un agente racional deberá emprender aquella acción que supuestamente maximice su medida de rendimiento, basándose en las evidencias aportadas por la secuencia de percepciones y en el conocimiento que el agente mantiene almacenado

Un agente racional actúa buscando el mejor resultado o, cuando hay incertidumbre, el mejor resultado esperado

- El agente es *No Omnisciente*:

No posee total información de todo su entorno, no conoce el resultado se sus acciones y no actúa de acuerdo a él

- Es un *Agente Que Aprende*:

"Se dice que un agente aprende cuando su desempeño mejora con la experiencia; es decir, cuando la habilidad no estaba presente en su genotipo o rasgos de nacimiento"

El agente aprende por la propia naturaleza de los algoritmos K-means y K-nn que utiliza. El dataset le provee un modelo de su entorno

- Es *Autónomo*:

Es capaz de aprender a actuar basándose en su experiencia gracias a sus propias percepciones

**Tabla REAS: Rendimiento, Entorno, Actuadores y Sensores**

- *Rendimiento:*

Se considera la confiabilidad de la categorización de la fotografía con margen de error pequeño

La cantidad de predicciones correctas de todos los elementos que se prueben permitirá dar una medida de rendimiento

- *Entorno:*

Es el área donde está ubicada la pieza a ser reconocida, y la misma pieza de ferretería en sí. Al entorno también lo afectan la luminosidad, la calidad de la imagen y el fondo. Por simplicidad el entorno está restringido a un fondo verde con iluminación fija y una sola pieza por fotografía. En cada fotografía debe estar un objeto completo, y no sólo parte de él

- *Actuadores:*

Es la forma que tiene el agente de comunicarse y notificar el resultado de la clasificación. En este caso la clasificación se comunicará de dos maneras: mediante un mensaje a través de la pantalla y de forma auditiva con un mensaje de voz

- *Sensores:*

Es la cámara fotográfica con que se toman las fotos de las piezas. Sin embargo, la fotografía también puede provenir de otras fuentes

**Propiedades del Entorno de Trabajo**

- *Parcialmente Observable:* La fotografía podría no llegar a dar una visión completa de la pieza a analizar. Podría suceder que la pieza no esté ubicada de forma adecuada (por ejemplo, si se colocara un tornillo de forma vertical apoyado sobre su base o la tuerca verticalmente apoyada sobre uno de los lados de su perímetro)
- *Episódico:* La experiencia del agente se dividirá en episodios atómicos. El siguiente estado del entorno no depende de las acciones previas. La cámara puede tomar las fotos de las arandelas, clavos, tornillos y tuercas en cualquier orden y el rendimiento no se verá afectado

- *Determinista:* El estado siguiente se obtiene a partir del actual y de las acciones del agente. La fotografía, una vez tomada, no puede ser alterada por variaciones aleatorias
- *Estático:* El entorno no puede cambiar mientras el agente está deliberando. La imagen de entrada una vez tomada es inalterable en principio
- *Discreto:* existe un número concreto de percepciones y acciones claramente definidos. Las imágenes captadas por cámaras digitales son discretas, en sentido estricto, pero se tratan típicamente como representaciones continuas de localizaciones e intensidades variables. El agente debe clasificar la foto entre una cantidad finita de clases
- *Individual:* Está claramente en un entorno de agente individual, puesto que en la interacción agente-entorno el único racional es el agente en cuestión

## Diseño del Sistema

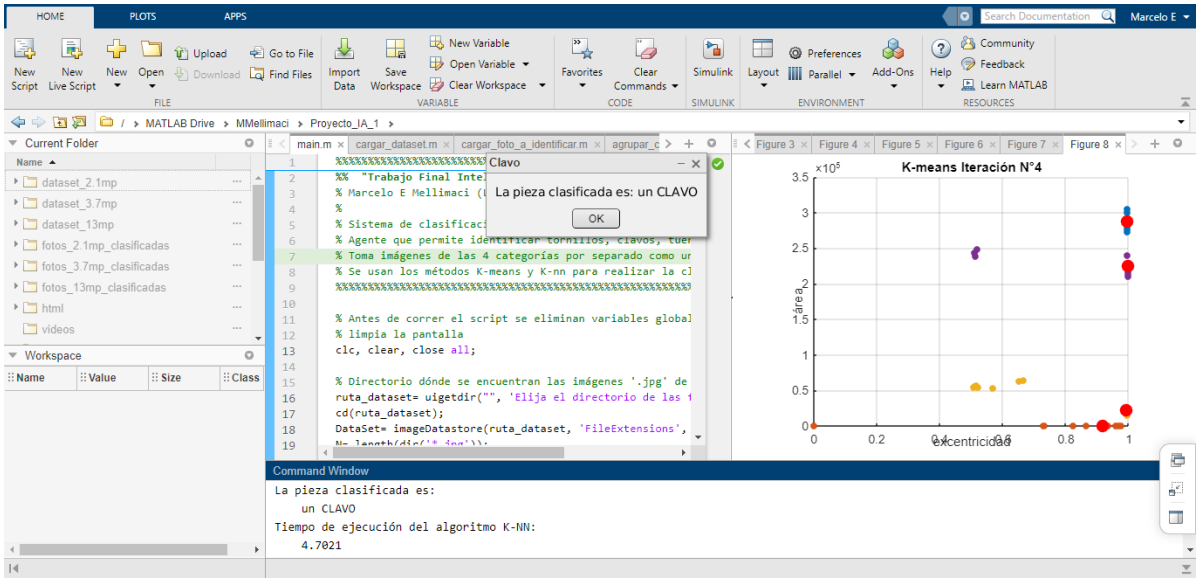
Entre las opciones disponibles, se consideró utilizar Python, MATLAB y Octave

Se decide finalmente hacer uso de MATLAB por disponer ya incorporados filtros, funciones que permiten trabajar fácilmente con fotografías y tener facilidad para extraer características de las imágenes

Se eligió usar MATLAB Online ya que desde prácticamente cualquier navegador web permite usar completamente de todas las funciones de la edición más actualizada de MATLAB en su versión de prueba gratuita (shareware)

**Figura 1**

*MATLAB Online Corriendo este Trabajo Final*




Se diseña como soporte una base de madera para, de esa manera, lograr tomar las fotografías con ángulo y altura constante. Se usó de fondo para todas las fotografías una cartulina verde con la finalidad de colocar cada pieza en un área homogénea de color uniforme

**Figura 2**

*Soporte de Altura Constante para Fotografiar las Piezas de Ferretería Propuestas*

Se usó la cámara fotográfica trasera de un celular LG K10, usando un zoom de 1.8x constante para todas las instantáneas

**Figura 3**

*LG K10 con la Base Confeccionada Fotografiando Tornillo sobre el Fondo de Cartulina Verde*

Se crearon 3 datasets:

- Primer base de datos: 48 fotos de 2.1 megapíxeles
- Segunda base de datos: 60 fotos de 3.7 megapíxeles
- Tercer base de datos: 40 fotos de 13 megapíxeles

Se compararon los resultados, tanto en precisión de la clasificación como en tiempo de ejecución para los 3 datasets

**Aclaración sobre las Piezas Clasificadas**

***Arandela:***

Una arandela es una placa delgada (generalmente en forma circular, aunque a veces cuadrada) con un orificio (usualmente en el medio) que generalmente se usa para impedir el rozamiento de las piezas entre las que se coloca y asegura su inmovilidad. Las arandelas también son importantes para prevenir la corrosión galvánica, aislando tornillos de metal de superficies de aluminio

Existen distintos tipos para diferentes aplicaciones; y se pueden agrupar en tres tipos fundamentales:

- distribución de carga: corresponde a las arandelas más sencillas, que reparten en una superficie más amplia la presión que reciben,
- seguridad por apriete elástico: representada por las arandelas Grower, especie de aros partidos a modo de espiras de muelle que impiden que se aflojen las tuercas por las vibraciones, y
- seguridad por rozamiento, que se consigue con las arandelas dentadas o en estrella, cuyos dientes se agarran a la superficie de contacto para impedir el giro

Para este proyecto se usó una arandela común con un diámetro interno de 25 mm y un diámetro exterior de 44,6 mm

**Figura 4**

*Fotografía de la Arandela Analizada*

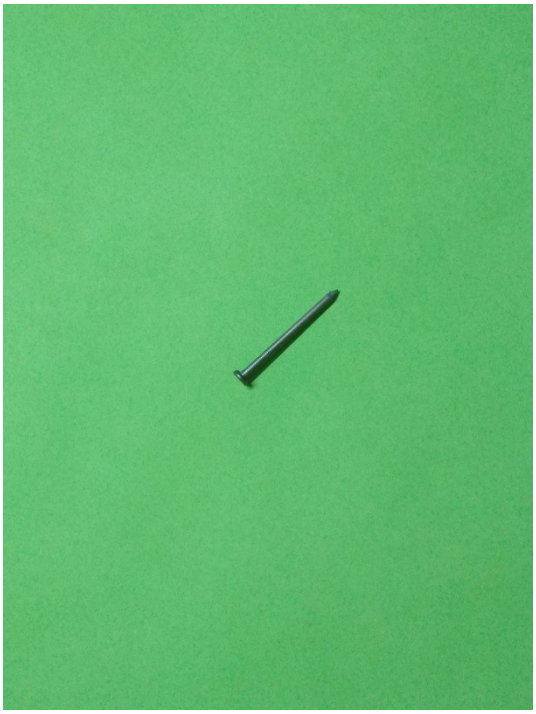
***Clavo:***

Se denomina clavo a un objeto delgado y alargado que dispone de punta y de cabeza y se utiliza para fijar algún elemento a otra cosa distinta. Los clavos se introducen en la pieza trabajada con un martillo o una pistola de clavos neumática. Un clavo mantiene los materiales juntos por fricción en la dirección axial, y por resistencia al corte lateralmente. La punta del clavo a veces también se dobla o se asegura luego de introducirlo para evitar que se salga. Existen clavos de diferentes tamaños. Mientras más grueso y extenso resulte, podrá soportar más peso

En este trabajo final se eligió utilizar un clavo común de acero que tiene un largo de 42,2 mm con un diámetro de 2,5 mm, y un diámetro de la cabeza de 5,6 mm

**Figura 5**

*Fotografía del Clavo Analizado*



***Tornillo:***

Un tornillo es un elemento que se utiliza para la sujeción de un objeto. Los tornillos son sujetadores típicamente hechos de metal y caracterizados por una cresta helicoidal, conocida como rosca macho (rosca externa). Los tornillos y pernos se utilizan para sujetar materiales mediante el acoplamiento de la rosca del tornillo con una rosca hembra similar (rosca interna) complementaria. Un tornillo cuenta con un cuerpo (caña) alargado y enroscado que se introduce en el lugar correspondiente y con una cabeza que dispone de ranuras para que pueda emplearse una herramienta y así realizar la fuerza correspondiente para su fijación

Se optó por emplear un tornillo de tapa de cilindro con una longitud de 136 mm y un diámetro calibrado de 9,1 mm, y la cabeza del tornillo tiene un diámetro de 21,9 mm

**Figura 6**

*Fotografía del Tornillo Analizado*

***Tuerca:***

Una tuerca es un tipo de pasador con un orificio roscado. Es una pieza con un hueco labrado en espiral que ajusta exactamente en el filete de un tornillo
Las tuercas casi siempre se usan junto con un tornillo para sujetar múltiples partes juntas. Los dos partes se mantienen unidos mediante una combinación de la fricción de sus hilos (con una ligera deformación elástica), un ligero estiramiento del tornillo y la compresión de las partes que se van a unir

En el presente proyecto final se hizo uso de una tuerca hexagonal con un diámetro interior de aproximadamente 10,05 mm y un diámetro externo de 18,9 mm, con paso milimétrico

**Figura 7**

*Fotografía de la Tuerca Analizada*

**Filtrado y Adecuación de las Imágenes**

El área dónde se colocó la pieza a ser reconocida es una cartulina verde, para intentar lograr un efecto similar al que se logra con el método Chroma Key en televisión. El fondo es lo suficientemente opaco como para evitar que se produzcan reflejos

Se realizó un soporte que mantiene la cámara fotográfica a una distancia constante de 292 mm y se usó un zoom de 1.8x para todas las tomas que se llevaron a cabo. Además, se trató de mantener el área bien iluminada al fotografiar cada uno de los objetos

**Figura 7**

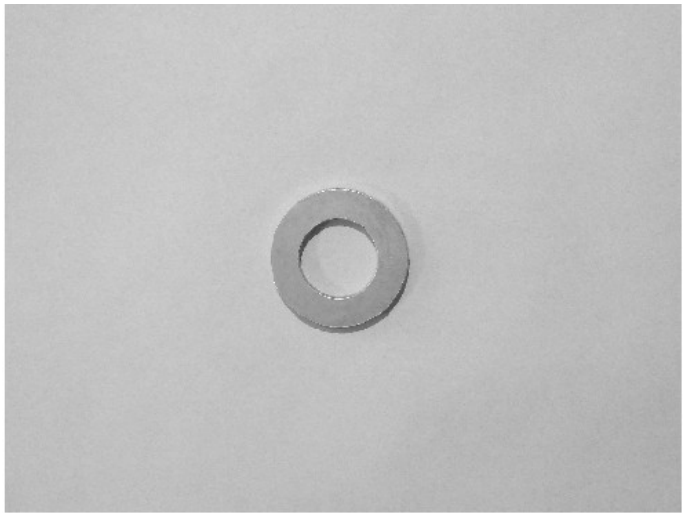
*Soporte con Altura Constante de 292 mm para Fotografiar las Piezas Examinadas*

En cada toma se fotografía sólo un objeto completo, y no sólo una parte de él en la captura. Tampoco se usan fotos con más de una pieza por imagen

Para poder acondicionar las imágenes obtenidas, primeramente, se extrae el canal verde de cada una de las fotos para poder eliminar el fondo verdoso que se ha usado como superficie homogénea deliberadamente

**Figura 8**

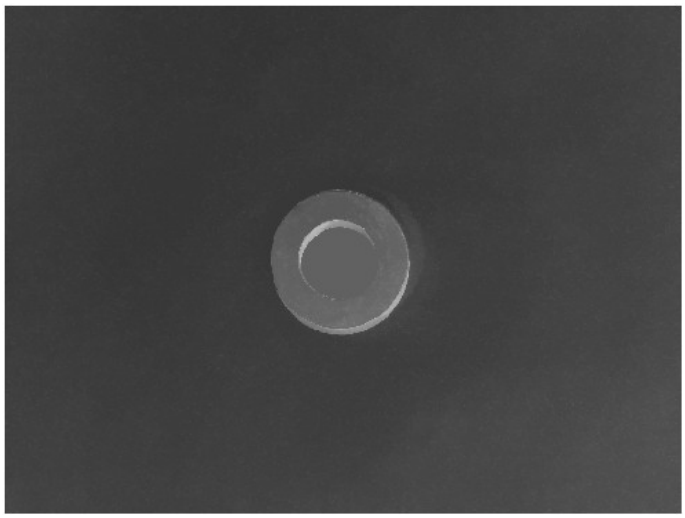
*Canal Verde*



Después de ello, se invierten los colores de dichas imágenes, obteniéndose "el negativo" con los colores opuestos para cada imagen. Luego se llenan los huecos vacíos cada figura, para conseguir así eliminar discontinuidades

**Figura 9**

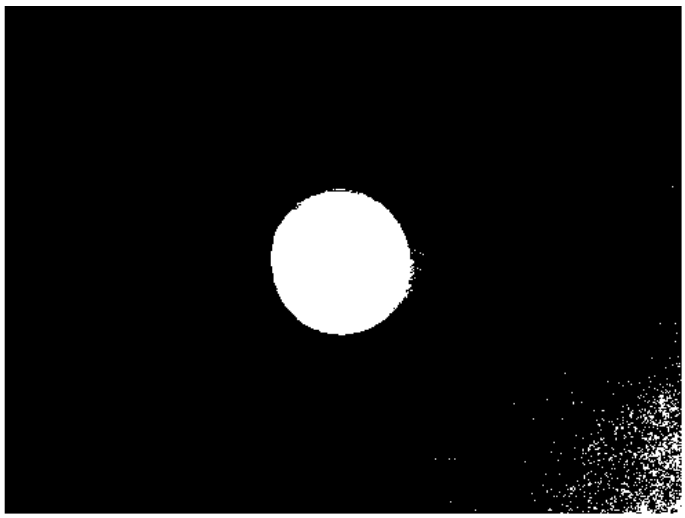
*Negativo de la Imagen sin Huecos*



Posteriormente, las imágenes así obtenidas se convierten a blanco y negro, consiguiéndose de esta manera imágenes binarias monocromáticas, adecuadas para extraer las características geométricas deseadas

**Figura 10**

*Imagen Binaria Monocromática*



**Extracción de Características de la Imagen**

Se pretende hallar un conjunto de rasgos capaces de describir cuantitativamente los objetos fotografiados, con el objetivo de así utilizarlos como parámetros de entrada para los métodos K-means y K-nn

Existen diversas técnicas de extracción de características en imágenes. Desde un punto de vista geométrico y matemático, es posible medir características tales como el área, el perímetro, el diámetro del círculo y la excentricidad. Desde el punto de vista de las características de la imagen, es posible usar: las relaciones de Inercia, el Histograma de gradientes orientados (HOG)

+ SVM, el análisis textural de Haralick, la estimación de los autovalores, el histograma de color, y los momentos de Hu, entre otros

Se optó por aplicar un análisis de las características geométricas, enfocado en la forma y las dimensiones de los elementos aislados en las imágenes

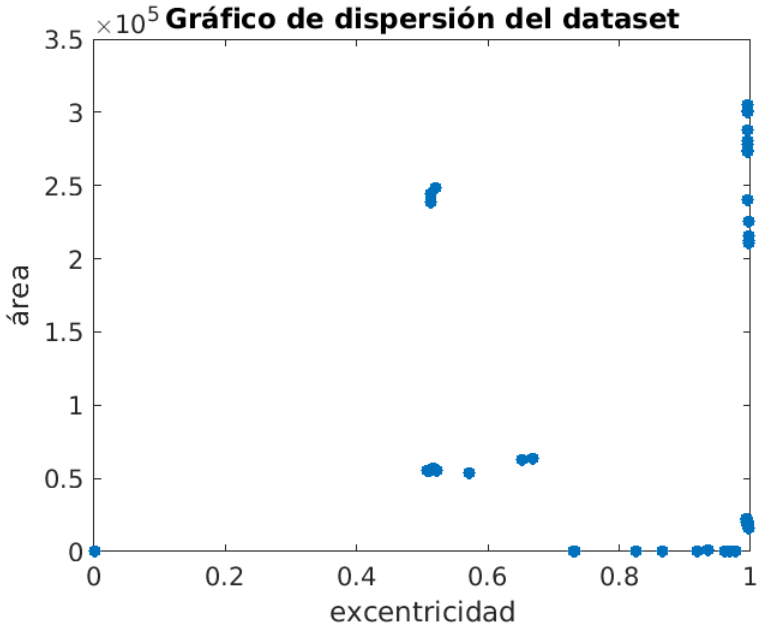
Para cuantificar las propiedades de las regiones de cada figura se hizo uso de la función regionprops() de MATLAB, que puede calcular múltiples atributos de las regiones en una imagen binaria

Se logró comprobar que se consiguen resultados buenos y representativos extrayendo las siguientes propiedades geométricas de las imágenes:

- *Área*: es el número de pixeles existentes dentro de la región. Es la superficie acotada, que se distingue de lo que la rodea
- *Excentricidad*: es un parámetro que determina el grado de desviación de una sección cónica con respecto a una circunferencia. Es la proporción entre la distancia entre los focos de la elipse y la longitud de su eje mayor. El valor varía entre cero y uno (Una elipse cuya excentricidad es cero es en realidad una circunferencia, mientras que una elipse cuya excentricidad es uno sería un segmento de línea recta)

**Figura 11**

*Diagrama de Dispersión para las Características de la Segunda Base de Datos*




**Método de Agrupamiento K-Medias**

K-medias (o K-means) es un método de agrupamiento, que tiene como objetivo la clasificación de un conjunto de N datos en K grupos en el que cada dato pertenece al grupo cuyo valor medio es más cercano. El algoritmo de clustering K-means es uno de los más usados para encontrar grupos sobre un conjunto de datos

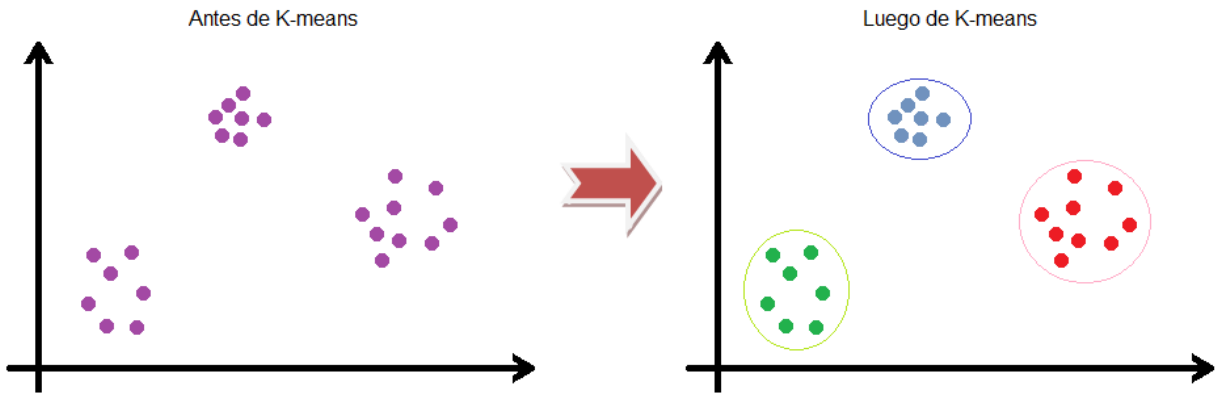
El método K-Means es utilizado en minería de datos. K-means considera K centroides que usa para definir grupos o clústeres. Se considera que un punto está en un grupo particular si está más cerca del centroide de ese grupo que cualquier otro centroide

El algoritmo trabaja iterativamente para asignar a cada “punto” (en este caso, cada foto del dataset forma una coordenada) uno de los K grupos basado en sus propiedades. Son agrupados

en base a la similitud de sus características (que vienen siendo las características extraídas de excentricidad y área)

**Figura 12**

*Algoritmo de Agrupamiento K-means (o K-medias)*



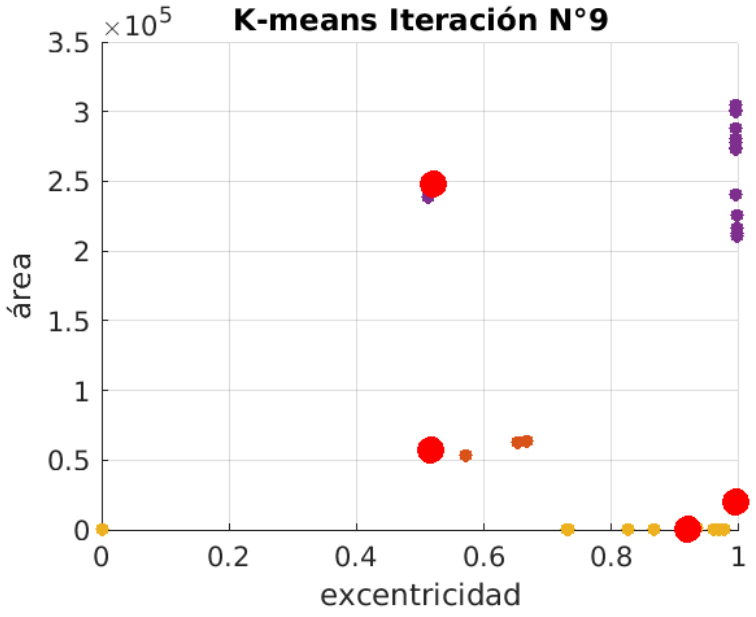

En nuestro caso, se conoce de antemano el valor de K, ya que trabajamos con 4 categorías o clases de piezas en el dataset, Como resultado de ejecutar el algoritmo tendremos:

- Centroides de cada grupo que serán las "coordenadas" de cada uno de los K conjuntos que se utilizarán para poder etiquetar las muestras de acuerdo a su categoría
- Categorías para los elementos del dataset. Cada una de esta etiqueta perteneciente a una de las K clases o grupos (arandela, clavo, tornillo o tuerca)

Los grupos van ajustando su posición en cada iteración del proceso, hasta que converge el algoritmo. Estos grupos son la categorización que genera el algoritmo

**Figura 12**

*Grupos y Centroides Hallados para el Segundo Dataset*



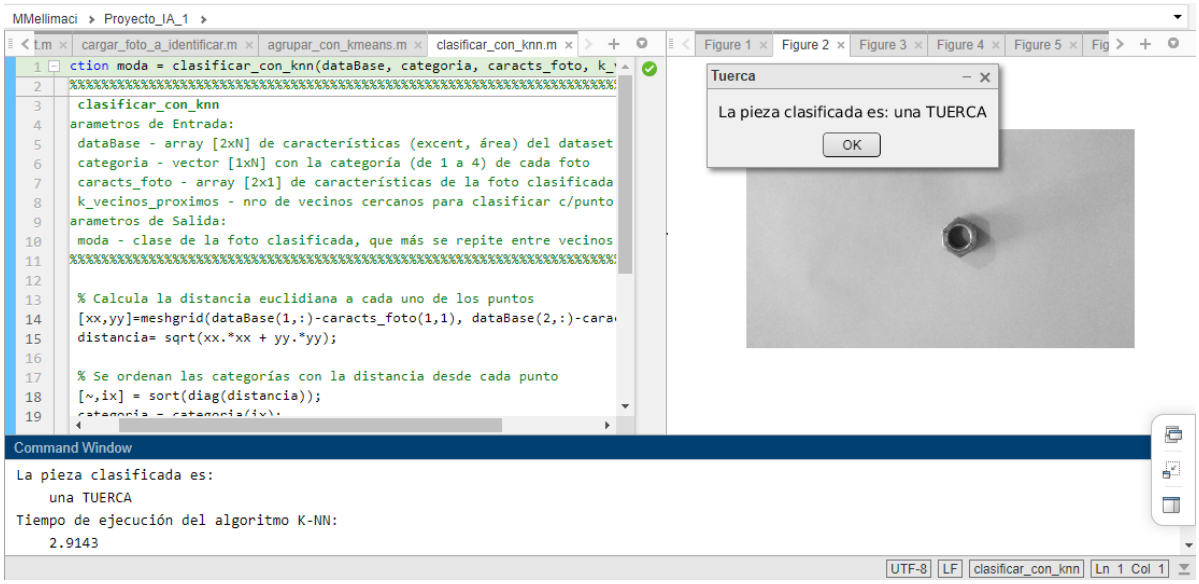

Si bien MATLAB ya tiene incorporada la función kmeans(), se decidió crear una de elaboración propia por motivos didácticos

**Método K-NN de Clasificación Supervisada**

"Dime con quién andas, y te diré quién eres" es un refrán que dice que se puede conocer a una persona a través de sus amigos cercanos y de sus compañías. Dicha idea es usada en el algoritmo de clasificación K-nn (K-Nearest Neighbor), que busca en las observaciones más cercanas a la que se está tratando de predecir (que aquí será la imagen de la pieza clasificada) y categorizar el punto de interés basado en la mayoría de datos que le rodean

**Figura 13**

*Tuerca Clasificada con el Algoritmo K-NN Implementado*




Es decir, calcula la distancia (para este trabajo final se usó la distancia euclidiana) del dato nuevo a cada uno de los existentes (usando como base de datos las características extraídas de las fotos del dataset), y ordena dichas distancias de menor a mayor para poder así seleccionar el grupo al que pertenece. Este grupo será, entonces, el de mayor frecuencia (moda) entre las menores distancias

A diferencia de K-means, que es un algoritmo "no supervisado" donde la “K” significa la cantidad de “grupos” (clústeres) en lo que se desea clasificar, en K-nn la “K” significa la cantidad de “puntos vecinos” que se tienen en cuenta en las cercanías del nuevo dato para clasificar los “N” grupos (que ya se conocen de antemano, ya que es un algoritmo Supervisado)

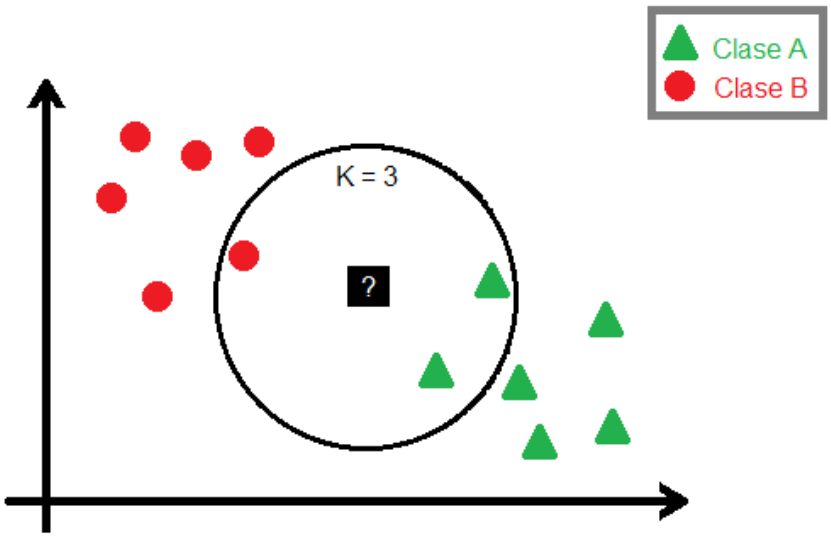
En este algoritmo el número de vecinos "K" NO debe ser un múltiplo de número de categorías (o clases), ya que no es recomendable porque podría llegar a dar lugar a que se produzca un empate entre dos o más clases durante la clasificación

Se fija un valor para K, preferentemente pequeño, y se hace que el algoritmo calcule una instancia con los datos asignados. La mejor elección de K depende mucho de los datos; generalmente valores grandes de K reducen el efecto de ruido en la clasificación, pero tardará más en procesarse el algoritmo y darnos una respuesta. Un buen K puede ser determinado a prueba y error mediante una optimización de uso. El caso especial en que la clase es predicha mediante la clase más cercana al nuevo dato de entrenamiento (cuando K = 1) es llamado Algoritmo del vecino más cercano (Nearest Neighbor Algorithm)

También se podría haber utilizado el método K-NN por medio del método fitcknn() que es una función interna de MATLAB

**Figura 14**

*Método de Clasificación K-Nearest Neighbors (o K Vecinos más Cercanos)*



## Código

El código fuente que se usó en este proyecto final está disponible un repositorio en: https://github.com/M-marce-E/t-final-ia1-2019

Se modularizó el código por medio de los subsiguientes archivos de código fuente:

**main.m**

El método main() es el punto de entrada donde comienza la ejecución del programa, que para este caso no es una función propiamente dicha. Esta parte proporciona el mecanismo para controlar la aplicación, y es el cuerpo principal del programa. En él se manejan los parámetros de entrada (en nuestro caso, serían por ejemplo el dataset, la foto clasificada, el número de iteraciones para el algoritmo K-means y el número de vecinos próximos K para el método K-nn) y se adecúan parámetros de salida (tales como el resultado de la categorización, el diagrama de dispersión de las características de la base de datos y los tiempos de ejecución, para este caso)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  "Trabajo Final Inteligencia Artificial I - año 2019"                          %
% Marcelo E Mellimaci (Leg 10764)                                                 %
%                                                                                 %
% Sistema de clasificación de piezas metálicas por visión artificial              %
% Agente que permite identificar tornillos, clavos, tuercas y arandelas           %
% Toma imágenes de las 4 categorías por separado como una base de datos           %
% Se usan los métodos K-means y K-nn para realizar la clasificación               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Antes de correr el script se eliminan variables globales y se
% limpia la pantalla
clc, clear, close all;

% Directorio dónde se encuentran las imágenes '.jpg' de base de datos
ruta_dataset= uigetdir("", 'Elija el directorio de las fotos del dataset:');
cd(ruta_dataset);
DataSet= imageDatastore(ruta_dataset, 'FileExtensions', {'.jpg'});
N= length(dir('*.jpg'));

% Se va al directorio del proyecto para leer los fuentes de las funciones
cd('/MATLAB Drive/MMellimaci/Proyecto_IA_1/');
% Se extraen características del dataset y se almacenan en dos arrays
[dataBase, categoria]= cargar_dataset(DataSet, N);
% Se muestra el diagrama de las características de las fotos del dataset
figure;
plot(dataBase(1,:), dataBase(2,:), '.', 'MarkerSize', 17);
xlabel('excentricidad');
ylabel('área');
title("Gráfico de dispersión del dataset");
```

```
% Carga la fotografía de la pieza a clasificar
[FileName,PathName] = uigetfile('*.jpg','Elija la foto de la pieza a clasificar:');
imagen= imread(strcat(PathName,FileName));
caracts_foto= cargar_foto_a_identificar(imagen);
% Muestra las características extratídas de la imagen a reconocer:
% el área y la excentricidad de la forma redondeada
disp('Excentricidad y Área de la imagen clasificada:');
disp(caracts_foto');

% Se agrupa en 4 grupos el dataBase con las características de las fotos
tic;
% answer= inputdlg({'Ingrese el nro de iteraciones del método K-means:'}, ...
%     'Número de iteracs para K-means:', [1 40], {'7'});
% iteraciones = str2double(answer{1});
iteraciones= input('Ingrese el Nro de Iteraciones para el método K-means:');
agrupar_con_kmeans(dataBase, N, 4, iteraciones);
disp("Tiempo de ejecución del algoritmo K-means:");
disp(toc);

% Se identifica la categoría de la foto a clasificar a través de KNN
tic;
% En KNN el nro de vecinos 'k' NO debe ser un múltiplo de nro de categorías
% answer= inputdlg({'Ingrese el valor K de vecinos próximos para K-nn:'}, ...
%     'K vecinos para el método K-nn:', [1 40], {'13'});
% k_vecinos_proximos = str2double(answer{1});
k_vecinos_proximos= input('Ingrese el valor K de vecinos próximos para K-nn:');
moda= clasificar_con_knn(dataBase, categoria, caracts_foto, k_vecinos_proximos);
% Se muestra el resultado de la categorización, además de un mensaje de voz
disp('La pieza clasificada es:');
if (moda == 1)
    disp('    una ARANDELA');
    disp("Tiempo de ejecución del algoritmo K-NN:");
    disp(toc);
    f = msgbox('La pieza clasificada es: una ARANDELA','Arandela');
    % Se escucha la frase "La pieza clasificada es una arandela"
    [y, Fs] = audioread('/MATLAB Drive/MMellimaci/Proyecto_IA_1/voces/arandela.wav');
    player=audioplayer(y,Fs);
    play(player);
elseif(moda == 2)
    disp('    un CLAVO');
    disp("Tiempo de ejecución del algoritmo K-NN:");
    disp(toc);
    f = msgbox('La pieza clasificada es: un CLAVO','Clavo');
    % Se escucha la frase "La pieza clasificada es un clavo"
    [y, Fs] = audioread('/MATLAB Drive/MMellimaci/Proyecto_IA_1/voces/clavo.wav');
    player=audioplayer(y,Fs);
    play(player);
elseif(moda == 3)
    disp('    un TORNILLO');
    disp("Tiempo de ejecución del algoritmo K-NN:");
    disp(toc);
    f = msgbox('La pieza clasificada es: un TORNILLO','Tornillo');
    % Se escucha la frase "La pieza clasificada es un tornillo"
    [y, Fs] = audioread('/MATLAB Drive/MMellimaci/Proyecto_IA_1/voces/tornillo.wav');
    player=audioplayer(y,Fs);
    play(player);
elseif(moda == 4)
    disp('    una TUERCA');
    disp("Tiempo de ejecución del algoritmo K-NN:");
    disp(toc);
    f = msgbox('La pieza clasificada es: una TUERCA','Tuerca');
    % Se escucha la frase "La pieza clasificada es una tuerca"
    [y, Fs] = audioread('/MATLAB Drive/MMellimaci/Proyecto_IA_1/voces/tuerca.wav');
    player=audioplayer(y,Fs);
    play(player);
end
```

**cargar_dataset.m**

Es una función que divide el dataset en 4 partes de igual cantidad de foto cada una según el orden alfabético de los nombres de los archivos .jpg de las imágenes en la base de datos, de esta manera se logra categorizar las fotos de arandelas, clavos, tornillos y tuercas. También se extrae la excentricidad y el área de cada una de las imágenes procesadas y se registra dichos valores en un array

```
function [dataBase, categoria] = cargar_dataset(DataSet, N)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  cargar_dataset                                                               %
% Parametros de Entrada:                                                        %
%   DataSet - conjunto de imágenes (imageDatastore) del dataset                 %
%   N - número total de fotos en la base de datos                               %
% Parametros de Salida:                                                         %
%   dataBase - array [2xN] con características (excen, área) del dataset        %
%   categoria - vector [1xN] con la categoría (de 1 a 4) para cada foto         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    dataBase= zeros(2, N);
    categoria= zeros(1, N);
    i= 1;
    %Los índices en Matlab/Octave empiezan en 1
    while (hasdata(DataSet))
        % Carga las imágenes
        imagen= read(DataSet);

        % Separa el canal verde de cada imagen para eliminar el fondo
        greenChannel= imagen(:,:,2);
        % Se rellenan las discontinuidades del negativo de las imágenes
        lleno= imfill(imcomplement(greenChannel), 'holes');
        % La imagen inversa se vuelve imagen binaria (monocromática)
        BW= imbinarize(lleno);

        % Se extraen características de la imagen
        cc= regionprops(BW, 'Eccentricity', 'Area');
        % Mide la excentricidad de la forma redondeada y
        % se guarda en la matriz con características
        dataBase(1,i)= cc.Eccentricity;
        % Se calcula el área de la región y se registra dicha característica
        dataBase(2,i)= cc.Area;

        % Se clasifican las N fotos en 4 categorías diferentes
        %quedando N/4 fotografías en cada categoría
        if (i <= N/4)
            % Arandelas
            categoria(1,i)=1;
        elseif(i>N/4 && i<=2*N/4)
            % Clavos
            categoria(1,i)=2;
        elseif(i>2*N/4 && i<=3*N/4)
            % Tornillos
            categoria(1,i)=3;
        else
            % Tuercas
            categoria(1,i)=4;
        end
        % Suma las posiciones para cada sección del arreglo
        i= i + 1;
    end
end
```

**cargar_foto_a_identificar.m**

Aquí se procesa la fotografía para clasificar, de la misma manera que se hizo con las fotografías del dataset, y de la misma forma se extrae de dicha imagen las características de excentricidad y área para poder luego categorizar la figura analizada

```
function caracts_foto = cargar_foto_a_identificar(imagen)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  cargar_foto_a_identificar                                                    %
% Parametros de Entrada:                                                        %
%   imagen - foto de la pieza que se va a clasificar                            %
% Parametros de Salida:                                                         %
%   caracts_foto - array [2x1] con la excentricidad y el área de la fig  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Se extrae el canal verde de la foto para eliminar el fondo verdoso
    greenChannel= imagen(:,:,2);
    % Se muestra en la pantalla el canal verde de la imagen
    figure;
    imshow(greenChannel);
    title('Canal Verde');

    % Se rellenan las discontinuidades del negativo de la imagen
    lleno= imfill(imcomplement(greenChannel), 'holes');
    % Se muestra la figura del negativo de la imagen
    figure;
    imshow(lleno);
    title('Negativo sin huecos');

    % La imagen inversa se vuelve imagen binaria (monocromática)
    BW= imbinarize(lleno);
    % Se muestra en la pantalla la imagen binaria
    figure;
    imshow(BW);
    title ('Imagen Monocromática');

    % Se extraen características de la foto
    cc= regionprops(BW, 'Eccentricity', 'Area');
    % Mide la excentricidad de la forma circular y se
    % guarda en el array con las características extraídas
    caracts_foto(1,1)= cc.Eccentricity;
    % Se calcula el área de la región y se registra dicha característica
    caracts_foto(2,1)= cc.Area;

end
```

**agrupar_con_kmeans.m**

Aplicando el algoritmo iterativo K-means se agrupan los datos del dataset en K grupos (para este caso K = 4, al tener 4 categorías) de acuerdo a sus características comunes (en este caso de acuerdo a la excentricidad y el área de la figura en cada imagen). Al terminar, se muestran las gráficas de dispersión de características de la base de datos con los centroides calculados para cada iteración y el agrupamiento obtenido

```
function agrupar_con_kmeans(dataBase, N, K, iteraciones)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  agrupar_con_kmeans                                                          %
% Parametros de Entrada:                                                       %
%   dataBase - array [2xN] con características del dataset                     %
%   N - número total de fotos en la base de datos                              %
%   K - número de categorías ó clasificaciones                                 %
%   iteraciones - número de iteraciones que realizará el método K-means        %
% Parametros de Salida:                                                        %
%   gráficas de dispersión del dataset con los centroides obtenidos            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    X= dataBase';
    % Toma K centroides aleatorios
    indC= randperm(N, K);
    for i = 1 : iteraciones
        ind= knnsearch(X(indC,:), X);
        % El centroide más cercano determina la categoría de cada punto
        for k = 1 : K
            indk= ind == k;
            meank= mean(X(indk,:));
            indC(k)= knnsearch(X, meank);
        end
        figure;
        hold all;
        % Se recalculan los centroides
        for k = 1 : K
            indk= find(ind == k);
            % Muestra el diagrama de dispersión de las características para
            % cada iteración
            plot(X(indk,1), X(indk,2), '.', 'MarkerSize', 17);
        end
        plot(X(indC,1), X(indC,2), 'or', 'MarkerSize',10, 'MarkerFaceColor','r');
        xlabel('excentricidad');
        ylabel('área');
        title(['K-means Iteración N°' num2str(i)]);
        grid on;
    end

end
```

**clasificar_con_knn.m**

Usando el método de clasificación K-nn, se determina el tipo (arandela, clavo, tornillo o tuerca) de la fotografía de la pieza a clasificar. Esta categorización será dada por la clase de objeto que más se repite entre los K vecinos más cercanos al punto a clasificar, teniendo en cuenta las características extraídas de excentricidad y área para cada imagen

```
function moda = clasificar_con_knn(dataBase, categoria, caracts_foto, k_vecinos_proximos)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  clasificar_con_knn                                                          %
% Parametros de Entrada:                                                       %
%   dataBase - array [2xN] de características (excent, área) del dataset       %
%   categoria - vector [1xN] con la categoría (de 1 a 4) de cada foto          %
%   caracts_foto - array [2x1] de características de la foto clasificada       %
%   k_vecinos_proximos - nro de vecinos cercanos para clasificar c/punto       %
% Parametros de Salida:                                                        %
%   moda - clase de la foto clasificada, que más se repite entre vecinos       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    % Calcula la distancia euclidiana a cada uno de los puntos
    [xx,yy]=meshgrid(dataBase(1,:)-caracts_foto(1,1), dataBase(2,:)-caracts_foto(2,1));
    distancia= sqrt(xx.*xx + yy.*yy);

    % Se ordenan las categorías con la distancia desde cada punto
    [~,ix] = sort(diag(distancia));
    categoria = categoria(ix);

    % El valor k NO debe ser un múltiplo de número de categorías
    % (k): Cantidad de puntos vecinos cercanos
    for i = 1 : k_vecinos_proximos
        for j = 1 : i
            etiqueta= categoria(1:i);
            moda= mode(etiqueta');
        end
    end

end
```

## Ejemplos de Aplicación

### Aplicaciones del Reconocimiento de Objetos por Visión Artificial

Las aplicaciones van desde tareas tales como sistemas industriales de visión artificial que, por ejemplo, inspeccionan botellas que pasan a gran velocidad en una línea de producción, hasta computadoras y robots que pueden por inteligencia artificial comprender el mundo que las rodea. En la mayoría de las aplicaciones de Machine Vision (MV), las computadoras están pre programadas para resolver una determinada tarea en particular, sin embargo, los agentes que aprenden se están volviendo cada vez más comunes.

Algunos ejemplos son:

- inspección automática en fabricación y manufactura
- ayudar en tareas de identificación de especímenes
- controlar un robot industrial
- para vigilancia o recuento de personas, por ejemplo, en restaurantes
- como entrada en un dispositivo para interacción computadora-humano
- análisis de imágenes médicas o modelado topográfico
- exploración mediante un vehículo autónomo o un robot móvil

- organizar bases de datos de imágenes o secuencias de imágenes
- procesamiento de imágenes médicas para diagnosticar a un paciente
- control de calidad donde los productos se inspeccionan automáticamente para encontrar defectos
- la detección de soldados o vehículos militares enemigos y la guía de misiles
- en vehículos autónomos capaces de percibir lo que los rodea y actuar en consecuencia
- detección de micro ondulaciones y calibración de manos robóticas
- ayuda para crear de efectos visuales en el cine, por ejemplo: seguimiento de cámara (matchmoving)
- vigilancia
- detección de somnolencia del conductor
- seguimiento y recuento de organismos en las ciencias biológicas

**Aplicaciones del Algoritmo K-Means**

El algoritmo K-means es muy popular y se utiliza en gran variedad de aplicaciones, tales como: agrupamiento de documentos, segmentación de mercado, segmentación de imágenes, compresión de imágenes, etc. Algunos ejemplos donde se aplica el clustering son:

- se utiliza en investigación de mercado, el reconocimiento de patrones, el análisis de datos y el procesamiento de imágenes.
- puede ayudar a los especialistas en marketing a descubrir grupos distintos en su base de clientes. También pueden categorizar a los clientes en función de los patrones de compra
- en biología, se puede utilizar para obtener taxonomías de plantas y animales, categorizar genes con funcionalidades similares y obtener información sobre poblaciones

- ayuda a identificar terrenos de uso similar en una base de datos para de observación de la tierra. También ayuda a distinguir grupos de casas en una ciudad según el tipo de casa, el valor y la ubicación geográfica
- ayuda a clasificar documentos en la web para hallar información
- se usa en detección de valores atípicos, como la detección de fraudes con tarjetas de crédito
- sirve como una herramienta para observar las características de cada cluster

**Aplicaciones del Algoritmo K-NN**

Podemos llegar a aplicar el algoritmo de K-nn en situaciones tales como: reconocimiento facial, minería de textos, finanzas, medicina, agricultura, y sistemas de recomendación (como en Netflix, Amazon, YouTube, etc.).

Se pueden citar como ejemplos:

- puede ser usado para sistemas de recomendación. Muchas empresas hacen una recomendación personalizada para sus consumidores, como Netflix, Hulu, Amazon, YouTube y muchas más. Sin embargo, en el mundo real se utilizan algoritmos más sofisticados para sistemas de recomendación
- se puede utilizar eficazmente para detectar valores atípicos o anómalos. Un ejemplo sería la detección de fraudes con tarjetas de crédito
- puede buscar documentos significativamente similares. Cada documento se considera un vector. Si los documentos están cerca unos de otros, eso significa que los documentos contienen temas idénticos

## Resultados

Se crearon 2 documentos para mostrar los resultados de la implementación realizada

- documento para la base de datos de 48 fotos de 2.1 megapíxeles disponible en: https://drive.google.com/file/d/1A0UsRyQ4n-ck-KRItSBp4b1xiEX4BrrA/view
- documento para la base de datos de 60 fotos de 3.7 megapíxeles disponible en: https://github.com/M-marce-E/t-final-ia1-2019/blob/main/html/proyect_ia1_mmellimaci.pdf

También se grabaron 3 videos ejemplificativos donde se ilustran los resultados obtenidos para cada uno de las 3 bases de datos:

- video para el dataset de 48 fotos de 2.1 Mpx en: https://youtu.be/k7-MO7hv0Uw
- video para el dataset de 60 fotos de 3.7 Mpx en: https://youtu.be/j6K_7znbn4E
- video para el dataset de 40 fotos de 13 Mpx en: https://youtu.be/fCaSXD2cCrk

Fueron analizados los tiempos de ejecución promedio del método K-means para los 3 datasets propuestos según sea el número de iteraciones realizadas:

- para el dataset de 48 fotos de 2.1 megapíxeles:

Tabla 1

*Resultados del Algoritmo K-Means para la Primer Base de Datos*

| | 3 iteraciones | 5 iteraciones | 8 iteraciones |
|---|---|---|---|
| Tiempo de ejecución medio | 0,3297 | 0,5082 | 0,8855 |

**Nota.** Planilla de Cálculo con los Datos en: https://drive.google.com/file/d/1YnLXsCy4NgMGkLkWnv5_UwsRyOi2kiO6/view

- para el dataset de 60 fotos de 3.7 megapíxeles:

Tabla 2

*Resultados del Algoritmo K-Means para la Segunda Base de Datos*

| | 3 iteraciones | 5 iteraciones | 8 iteraciones |
|---|---|---|---|
| Tiempo de ejecución medio | 0,3065 | 0,5283 | 0,919 |

**Nota.** Planilla de Cálculo con los Datos en:
https://drive.google.com/file/d/1YnLXsCy4NgMGkLkWnv5_UwsRyOi2kiO6/view

- para el dataset de 40 fotos de 13 megapíxeles:

Tabla 3

*Resultados del Algoritmo K-Means para la Tercer Base de Datos*

| | 3 iteraciones | 5 iteraciones | 8 iteraciones |
|---|---|---|---|
| Tiempo de ejecución medio | 0,335 | 0,549 | 0,9273 |

**Nota.** Planilla de Cálculo con los Datos en:
https://drive.google.com/file/d/1YnLXsCy4NgMGkLkWnv5_UwsRyOi2kiO6/view

Se puede observar que no se aprecian diferencias significativas en el tiempo de ejecución para el agrupamiento K-means con los 3 datasets, esto posiblemente sea debido a que el algoritmo K-means trabaja con las características extraídas de las imágenes en forma de valores numéricos, y no con las fotografías en sí

Además, se analizó la precisión alcanzada en los resultados y el tiempo de ejecución promedio del algoritmo K-nn para cada uno de los 3 datasets:

- para el dataset de 48 fotos de 2.1 Mpx:

Tabla 4

*Resultados del Método K-NN para el Primer Dataset*

| | K = 3 | K = 5 | K = 7 |
|---|---|---|---|
| Tiempo de ejecución medio | 0,0029 | 0,0021 | 0,0024 |
| Precisión de las predicciones | 75,00 % | 75,00 % | 75,00 % |

**Nota.** Planilla de Cálculo con los Datos en: https://drive.google.com/file/d/1YnLXsCy4NgMGkLkWnv5_UwsRyOi2kiO6/view

- para el dataset de 60 fotos de 3.7 Mpx:

Tabla 5

*Resultados del Método K-NN para el Segundo Dataset*

| | K = 3 | K = 5 | K = 7 |
|---|---|---|---|
| Tiempo de ejecución medio | 0,0021 | 0,0024 | 0,0024 |
| Precisión de las predicciones | 83,33 % | 83,33 % | 83,33 % |

**Nota.** Planilla de Cálculo con los Datos en: https://drive.google.com/file/d/1YnLXsCy4NgMGkLkWnv5_UwsRyOi2kiO6/view

- para el dataset de 40 fotos de 13 Mpx:

Tabla 6

*Resultados del Método K-NN para el Tercer Dataset*

| | K = 3 | K = 5 | K = 7 |
|---|---|---|---|
| Tiempo de ejecución medio | 0,0023 | 0,0028 | 0,0053 |
| Precisión de las predicciones | 83,33 % | 83,33 % | 83,33 % |

**Nota.** Planilla de Cálculo con los Datos en: https://drive.google.com/file/d/1YnLXsCy4NgMGkLkWnv5_UwsRyOi2kiO6/view

Es posible concluir que el tiempo de ejecución de esta implementación del algoritmo K-nn no se ve alterado por el valor cantidad K de vecinos analizado, tal vez debido a que sin importar el

valor adoptado para K de todas formas siempre se ordenan todas las piezas del dataset según su distancia euclidiana calculada desde el punto de la imagen que se desea clasificar. Se observa que se aumentó la precisión de clasificación aumentando la resolución de las fotos y la cantidad de fotografías del dataset

## Conclusiones

Para llevar a cabo el presente proyecto se realizaron 3 datasets: uno de 48 fotografías de 2.1 megapíxeles, otro con 60 fotografías de 3.7 megapíxeles, y el último de 40 fotografías de 13 megapíxeles

El algoritmo de agrupamiento K-means podría resultar de mayor utilidad si no se conociera de antemano a cuál de cada una de las cuatro categorías de piezas pertenece cada una de las fotografías de la base de datos, lo que no sucede en este caso debido a que las fotos del dataset ya se encuentran en cierto modo “etiquetadas”

Una limitación que encontró, debido al modo en que se cargan las imágenes que se toman de la base de datos, es que cualquier dataset que se desee utilizar debe tener el mismo número de fotografías de cada clase (por ejemplo, una base de datos con 40 fotos deberá tener: 10 fotos de arandelas, 10 fotos de clavos, 10 fotos de tornillos y 10 fotos de tuercas). Si se necesitara emplear un dataset que contenga distinto número de fotografías para cada categoría, se debería modificar el código fuente del fichero “cargar_dataset.m”

Por motivos didácticos se muestra en pantalla para cada iteración que se realiza la gráfica de dispersión con el agrupamiento conseguido y el centroide calculado de cada uno de los cuatro grupos. Esto hace que la implementación del método K-medias resulte ser lenta que si solamente

se mostrara al final el diagrama con los centroides para los cuatro grupos correspondiente a la última iteración realizada

Fue posible notar que la categorización resultante del uso del método K-means no fue demasiado precisa, ya que los grupos formados de esta manera no coinciden exactamente con las cuatro categorías de elementos que se tiene de antemano en la base de datos. Esto tal vez sea una consecuencia de la aleatoriedad al elegir los centroides iniciales

Asimismo, el método K-Means podría ser también usado para clasificar un elemento dentro de K grupos ya definidos si se calcula la distancia a los diferentes means

En análisis realizado a lo largo de todo este trabajo final, se puede llegar a concluir que el método K-nn que se implementado logra clasificar datos de forma sumamente fidedigna y tiene un tiempo de ejecución muy adecuado y eficaz. Dicho algoritmo logró alcanzó un rendimiento de hasta un 83,33 % para los datasets empleados

Con lo estudiado, se corroboró que el algoritmo K-Nearest Neighbors logra tener un tiempo de ejecución eficiente y veloz (demora menos de 0,0053 segundos), lo que lo hace bastante adecuado para un ámbito de aplicación donde se requiera un runtime rápido, como sería el caso de una cinta transportadora industrial, la caja de un mercado, o un lector de huella dactilar

Una opción a analizar para perfeccionar esta implementación de K-nn sería ponderar la contribución de cada vecino dentro del dataset según la distancia entre él y la fotografía a ser clasificada, dando mayor importancia a los vecinos más próximos (por ejemplo, usando el cuadrado inverso de sus distancias)

Para los dos algoritmos implementados se encontró que, aumentando la cantidad de fotos en la base de datos o la resolución en megapíxeles de las mismas se produce una mejora la exactitud

de la categorización realizada, pero esto trae como inconveniente que lentifica el tiempo de ejecución de todo el código, particularmente ralentiza el procesamiento de las fotos

Una alternativa que se podría llegar plantear para aligerar los tiempos de ejecución consiste en llevar toda implementación a un lenguaje de mayor desempeño, como es el caso de C, C++ o Julia, pero con la desventaja de que dichos lenguajes de programación no disponen de muchas funcionalidades para trabajar con imágenes

Podría mejorarse la precisión de la clasificación obtenida optimizando los procedimientos que se utilizan para procesar y acondicionar las fotografías, y los métodos para extracción de características de las mismas. Otra alternativa que se podría llegar a plantear es analizar los momentos invariantes de Hu de las imágenes, lo cual podría llegar a mejorar la performance de la clasificación realizada. Además, se podría tratar trabajando con más de dos características (en este trabajo final se usaron sólo dos: la excentricidad y el área). También se podría llegar a considerar analizar el número de agujeros que tiene la pieza fotografiada

Se logró mejorar considerablemente la precisión en los resultados usando del soporte con altura constante y un fondo verde homogéneo para fotografiar cada una de las piezas de ferretería analizadas. Un factor que se podría mejorar es el modo en que se realiza la iluminación, por ejemplo, adicionando una lámpara al soporte de madera utilizado para tomar las fotos

## Referencias Bibliográficas

Stuart Russell, Peter Norvig (2010). Artificial Intelligence: A Modern Approach, Third Edition. Pearson Education Limited

Jim Sizemore, John Paul Mueller (2014). MATLAB For Dummies. John Wiley & Sons Inc

Richard Szeliski (2010). Computer Vision: Algorithms and Applications. Springer London Ltd

Dra. Ing. Selva S. Rivera (2020). Aprendizaje [Apuntes de la cátedra de Inteligencia Artificial I]. Facultad de Ingeniería, Universidad Nacional de Cuyo

Saketh Saxena (2017). Analysis of 3-fold Cross Validation for KNN, K-means and Hierarchical Clustering implementations. https://github.com/sakethsaxena/Image-Classification-and-clustering

MathWorks, Oman Wisni (2018). how to mask the image and keep only one color "green channel"? [MATLAB Answers - MATLAB Central]. https://www.mathworks.com/matlabcentral/answers/420754-how-to-mask-the-image-and-keep-only-one-color-green-channel

MathWorks (2021). Fill image regions and holes - MATLAB imfill. https://www.mathworks.com/help/images/ref/imfill.html

MathWorks (2021). Measure properties of image regions - MATLAB regionprops. https://www.mathworks.com/help/images/ref/regionprops.html

Venkatesh Umamaheswaran, Becoming Human: Artificial Intelligence Magazine (2018). Comprehending K-means and KNN Algorithms. https://becominghuman.ai/comprehending-k-means-and-knn-algorithms-c791be90883d

INCAE Business School, dataminingincae (2013). Data Mining: K-Vecinos Mas Cercanos (K-Nearest Neighbors, KNN). https://youtu.be/V_D3N2UuDCU

Elias David Niño Ruiz (2020). Minería de Datos - Explicación Simple de K-Means - Implementación en MATLAB. https://youtu.be/ggJW4Hh9PiA