Parcial 1 inteligencia artificial tema 7 clustering difuso

Juan Jose Galindo

*{est.juan.galindo6 }@unimilitar.edu.co*Profesor: Robinson Jimenez

***Resumen*—En el presente informe se observará el desarrollo de un programa realizado en Python el cual tiene el propósito de identificar las partes de una imagen por medio del método de clustering difuso**

***Palabras clave*—Python, clustering, difuso, imagen.**

# Marco Teórico

**Clustering Difuso:** Herramienta matemática para intentar descubrir estructuras, grupos o clases dentro de un conjunto de patrones (datos), de tal forma que los patrones en cada cluster muestran cierto grado de similaridad. Esta técnica de clasificación realiza comparaciones cuantitativas de diferentes características a los individuos de un conjunto para determinar los grupos dentro de la población. [1]

AGRUPAMIENTO FUERTE: Asignar cada patrón a uno y solo uno de los clusters, con un grado de membresía igual a 1, y con fronteras bien definidas “nítidas” entre los clusters. AGRUPAMIENTO DIFUSO: Permite a cada patrón pertenecer a más de un cluster con diferentes grados de membresía (entre 0 y 1) y fronteras “difusas” entre los clusters.

Para la detección de colores se utiliza un agrupamiento difuso, que toma en cuenta cada color como un cluster. Dándonos una imagen monocromática

# Metodología

Para el desarrollo de este trabajo se utilizó Python como lenguaje de programación con el fin de desarrollar el método de clustering difuso donde se creó una interfaz la cual nos permite cargar una imagen seleccionada por el usuario donde también se debe seleccionar la cantidad de objetos diferentes que se identificaran por medio de la cantidad de clusters que se utilizaran, para llevar este proceso a cabo primero se importaron las bibliotecas que se necesitaron en el código.

import numpy as np

import matplotlib.pyplot as plt

from skimage import io, color

import skfuzzy as fuzz

import tkinter as tk

from tkinter import filedialog

import colorsys

**Figura 1 Librerías importadas en Python.**

Se importaron librerías como lo son: skfuzzy la cual sirve para realizar el clustering difuso, skimage se utilizó para operaciones relacionadas con imágenes y numpy la cual se usa para manipular matrices numéricas, entre otras, siendo estas 3 las de mayor relevancia en el código.

A continuación se realizó la manipulación de las imágenes en el programa.

def fuzzy\_clustering(image\_path, num\_clusters):

image = io.imread(image\_path)

if image.shape[2] == 4:

image = image[:, :, :3]

image\_lab = color.rgb2lab(image)

data = image\_lab.reshape(-1, 3)

cntr, u, \_, \_, \_, \_, \_ = fuzz.cluster.cmeans(

data.T, num\_clusters, 2, error=0.005, maxiter=1000

)

cluster\_membership = np.argmax(u, axis=0)

clustered\_data = cntr[cluster\_membership]

clustered\_image\_lab = clustered\_data.reshape(image\_lab.shape)

clustered\_image\_rgb = color.lab2rgb(clustered\_image\_lab)

return clustered\_image\_rgb

**Figura 2 Manipulación de imágenes en Python.**

Inicialmente se realiza la segmentación difusa de una imagen en colores donde se toma la ruta de la imagen y el número deseado de clusters como entrada, para esto primero se lee la imagen usando io.imread de la biblioteca “skimage”, donde se verifica si la imagen tiene un canal alfa (4 canales), se quita este canal adicional para quedarse únicamente con los canales “RGB” utilizando la función “slicing image[:, :, :3]”, seguido a esto se convierte la imagen “RGB” a espacio de color “LAB” utilizando la función “color.rgb2lab” para poder manipular de mejor manera la imagen y reorganizar los datos en una matriz bidimensional, donde cada fila representa un píxel y sus valores de “ L”, “A” y “B”.

Una vez teniendo la imagen cargada y pudiendo manipular sus píxeles se utiliza la función “fuzz.cluster.cmeans” de la librería “skfuzzy” para así poder realizar el clustering difuso

en los datos “LAB” arrojados por la imagen, todo esto teniendo en cuenta que “cntr” contiene los centroides de los clusters y “u” contiene las pertenencias difusas a los clusters,

con esta función lo que se hace es calcular la membresía más alta para cada píxel y así recuperar los datos de centroide correspondientes para formar la imagen agrupada en “LAB” y finalmente ya habiendo realizado la manipulación respectiva a la imagen se debe convertir a RGB utilizando la función “color.lab 2 rgb” para así poder visualizar de nuevo la imagen con las partes de la foto definidas.

Para poder tener un correcto funcionamiento del programa debemos cargar la imagen que elija el usuario para esto realizamos el siguiente proceso.

def open\_and\_segment():

image\_path = filedialog.askopenfilename()

if image\_path:

num\_clusters = int(cluster\_entry.get())

segmented\_image = fuzzy\_clustering(image\_path, num\_clusters)

plt.imshow(segmented\_image)

plt.axis('off')

plt.show()

unique\_colors = np.unique(segmented\_image.reshape(-1, 3), axis=0)

color\_window = tk.Toplevel(root)

color\_window.title("Colores Presentes")

for idx, color\_rgb in enumerate(unique\_colors):

# ... (crear y mostrar etiquetas de color en la GUI)

**Figura 3 Apertura y segmentación de la imagen en Python.**

Para poder realizar todo el proceso de manipulación de la imagen antes debemos seleccionar la imagen para lo cual utilizamos la función “open\_and\_segment” esta se ejecuta cuando se presiona el botón "Abrir y Segmentar" en la “GUI” donde lo que realiza es abrir un cuadro de diálogo para seleccionar una imagen, luego se debe seleccionar una imagen, para así obtener el número de clusters determinados por el usuario en la entrada de texto “cluster\_entry”, por último se utiliza a la función “fuzzy\_clustering” para obtener la imagen segmentada y la así mostrarla utilizando la función“Matplotlib”, por ultimo para dejar la imagen lista para poder modificarla correctamente se deben calcular los colores únicos presentes en la imagen segmentada y se crea una ventana emergente con la función “color\_window” utilizando “Tkinter” en donde se deben mostrar los colores en forma de etiquetas y bloques de color.

Finalmente se debe crear la interfaz en donde van a estar el botón para cargar la imagen y el cuadro de texto para ingresar el número de clusters.

root = tk.Tk()

root.title("Fuzzy Clustering de Colores")

title\_label = tk.Label(root, text="Segmentación de Colores con Clustering Difuso")

title\_label.pack()

cluster\_label = tk.Label(root, text="Número de Clusters:")

cluster\_label.pack()

cluster\_entry = tk.Entry(root)

cluster\_entry.pack()

open\_button = tk.Button(root, text="Abrir y Segmentar", command=open\_and\_segment)

open\_button.pack()

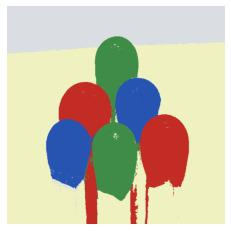
root.mainloop()

**Figura 4 Creación de la “GUI” y los botones en Python.**

Para realizar el proceso anteriormente descrito utilizamos la función “Tkinter” para crear la GUI principal seguido a esto se establece el título de la ventana y se agregan las etiquetas de cada función, también una entrada de texto para el número de clusters y el botón que llama a la función “open\_and\_segment” cuando se hace clic en él, también se debe tener la ventana abierta y en funcionamiento hasta que el usuario la cierre para esto se utilizó la función “root.mainloop()”.

# Análisis de Resultados

El programa al realizar el proceso de la manipulación de imágenes tenemos que la imagen dependiendo del número de clusters que se coloquen así mismo será el número de partes reconocibles de la imagen como se muestra en la figura.



**Figura 5 Resultado del clustering difuso Python.**

En la figura 5 podemos observar como la imagen al colocar 5 clusters vemos como se definen las cerillas de cada fósforo pudiendo distinguir las cerillas de palo y del fondo sin embargo también se puede observar como los palos de las cerillas se funden con los colores del fondo incluso con algunas partes de la cerilla, siendo la imagen de referencia la presentada en la figura 6.

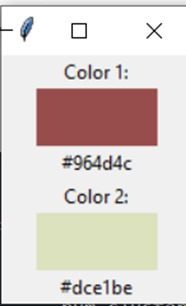


**Figura 6 Imagen cargada al programa de Python.**

Al realizar el proceso con solo dos cluster obtenemos como resultado la figura 7 teniendo en cuenta los colores dados por la figura 8

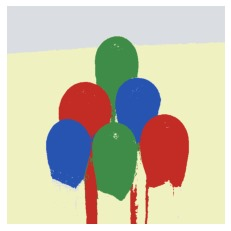


**Figura 7 Resultado del clustering difuso Python.**

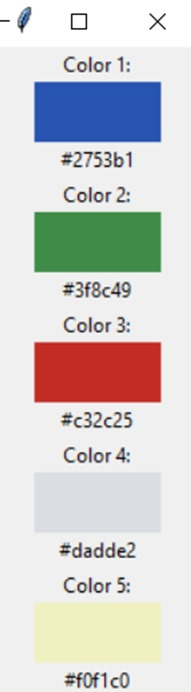
****

**Figura 8 Resultado del clustering difuso Python.**

Al realizar el proceso con cinco cluster obtenemos como resultado la figura 9 teniendo en cuenta los colores dados por la figura 10



**Figura 9 Resultado del clustering difuso Python.**

****

**Figura 10 Resultado del clustering difuso Python.**

Como se mencionó anteriormente los palos de las cerillas se confunden con el fondo, cuando en la imagen original esto se puede apreciar a simple vista sin embargo esto puede estar dado por la semejanza entre los colores, una forma de solucionar esto es posible gracias a tener un mayor número de clusters como veremos a continuación.



**Figura 11 Imagen cargada #2 al programa de Python.**

En esta nueva imagen tenemos la imagen original en la figura 11.



**Figura 12 Resultado de la imagen #2 del clustering difuso Python.**

Como podemos observar en la imagen de la figura128 se distingue por colores lo que es el gato y su fondo sin más componentes como lo son los ojos, esto es porque en este proceso solo se le pusieron 2 clusters, o dicho de otra forma solo se permitió que se detectasen dos colores, para este caso el café y el azul..



**Figura 13 Resultado de la imagen #2 del clustering difuso #2 Python.**

En la figura 13 ya podemos observar cómo al tener más clusters obtenemos más detalles seccionados de la imagen como lo puede ser las distintas sombras tanto del gato como de su fondo, y también apareciendo ya sus ojos.

Como conclusión final tenemos que al procesar la imagen y tener bajo número de clusters como 1 o 2 tenemos una disminución de los detalles distinguiendo las diferentes partes de la imagen de manera muy general, sin embargo

Al aumentar el número de clusters tenemos muchos más detalles en la imagen acercándose así a la imagen original.

Adicionalmente podemos ver una relación directa entre el número de clusters y el número de colores de la imagen , siendo el iguales, permitiendonos tener una imagen más detallada al usar mas clusters, debido a que se reconocerá una mayor cantidad de colores.

Referencias

1. https://virtual2.umng.edu.co/pluginfile.php/1447236/mod\_resource/content/0/CLASE%203%20FL.pdf