# PEC1: Prueba de evaluación continua 1

## Normas para la entrega:

La entrega consistirá en un fichero comprimido que debe contener:

- La **memoria** en formato pdf
- El **código implementado** en formato .py o .pynb y organizado por directorios. El código debe ser fácilmente ejecutable. Para ello, los paths a las imágenes utilizadas deben ser relativos a un directorio DATA dentro de la estructura de directorios entregada, el cual contendrá todas las imágenes utilizadas.

En la evaluación de esta PEC se tendrá en cuenta tanto la resolución de los problemas planteados (el código implementado y los resultados obtenidos) como la calidad de la memoria (redacción que justifica y describe la solución realizada). La memoria no debe detallar nuevamente el código que se ha implementado sino justificar por qué se han seleccionado determinadas operaciones (en función del contexto, los datos de partida, ...) y evaluar la solución obtenida (rendimiento del sistema, por qué se producen algunos fallos, cómo se podrían solucionar, ...). Por ello, y para fomentar la capacidad de síntesis, **la longitud máxima de la memoria será de 12 páginas**. Se valorará que la información presentada esté debidamente estructurada y que la redacción cumpla con los criterios de corrección gramatical y ortográfica.

## Enunciado de los problemas

### 1) Eliminación de ruido.

Una manera sencilla de evaluar el deterioro cognitivo es mediante los test neuropsicológicos gráficos, en los cuales se pide al sujeto realizar una copia a mano alzada de un dibujo geométrico dado y después evaluar la calidad de éste. Distintos tipos de fallos corresponden a deficiencias en distintas funciones cognitivas y distintas áreas cerebrales afectadas.

En el directorio "`data/DibujosNPT`" tenemos una serie de imágenes de estos dibujos escaneados, las cuales presentan distintos tipos de ruido. Realizar una aplicación interactiva que permita eliminar el ruido manteniendo el trazado aproximado de las líneas.

Observaciones: En procesamiento de imágenes, ruido se entiende como cualquier señal o alteración no deseada que se introduce en la imagen y que dificulta la percepción o el análisis de la información contenida en la imagen original. Para resolver el problema se deberán analizar las imágenes para identificar los distintos tipos de ruido existentes y buscar soluciones independientes para cada tipo de ruido encontrado (hablamos de ruido en sentido amplio, no solo de ruido a distintas frecuencias como pueden ser el ruido gaussiano, el ruido de sal y pimienta o el ruido de alta frecuencia). Muy posiblemente, por tanto, no existirá una única transformación que, por sí sola, permita solucionar el problema. En su lugar, será necesario aplicar una serie de operaciones que podrán ser aplicadas globalmente a toda la imagen o solo de manera local a una región de interés (ROI).

Se aconseja construir una interfaz sencilla que permita cargar una imagen y aplicarle, de manera interactiva y secuencial, tanto los operadores para el análisis de la imagen como los de eliminación de ruido necesarios en cada caso. Una opción sencilla y de código abierto es streamlit https://streamlit.io/

Ejemplos de operaciones que deberá contener la interfaz son las siguientes: seleccionar ROI, calcular el histograma (de la imagen completa y de la ROI), aplicar operadores de transformación de la imagen como umbralización con threshold manual, filtros de ruido, operaciones morfológicas, etc. Tenga también en cuenta que la eliminación de algún tipo de ruido puede implicar la pérdida de las líneas trazadas, por lo que deberá realizar una restauración aproximada (reconstruir una línea del mismo grosor que el resto).

Utilice las imágenes N_328_THS_TOTAL-ev1-h.png y N_307_GLS_TOTAL-ev5-h.png para diseñar una secuencia de operadores para eliminar el ruido y mostrar los resultados obtenidos en la memoria.

**Referencias:**

https://docs.opencv.org/master/dc/d4d/tutorial_py_table_of_contents_gui.html

https://docs.opencv.org/master/db/deb/tutorial_display_image.html

https://docs.opencv.org/master/d2/d96/tutorial_py_table_of_contents_imgproc.html

https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

**2) Transformaciones geométricas.**

Dentro del conjunto de transformaciones geométricas (toda aplicación biyectiva f del plano o del espacio en sí mismo), nos centraremos en las transformaciones que mantienen la topología de los objetos (relación de vecindad entre puntos contiguos), esto es, que transforman un cuadrilátero en otro cuadrilátero de manera que la distancia entre los puntos puede variar pero se mantiene la estructura

interna de la malla (ver Figura 1). Obsérvese que estas transformaciones no modifican el valor de intensidad, solo su posición, aunque cuando no coinciden las coordenadas finales con puntos iniciales exactos será necesario realizar una aproximación (interpolación) en función de los vecinos más cercanos. Se pueden considerar distintos tipos de interpolaciones, siendo las más habituales la interpolación lineal, bilineal o bicúbica, y la interpolación por el vecino más cercano.

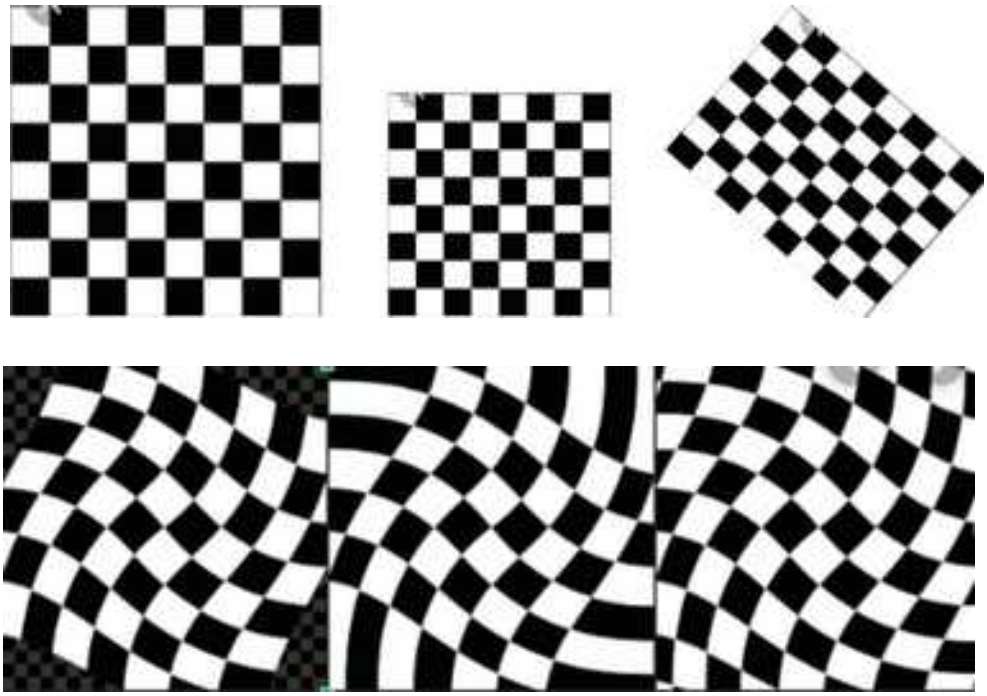


*Figura 1: Transformaciones 2D que mantienen la topología.*

**Transformaciones lineales y no lineales**.

Las transformaciones lineales en el plano 2D se pueden representar matricialmente como:

$$\begin{pmatrix} x_1' \\ y_1' \\ 1 \end{pmatrix} = \begin{pmatrix} A & B & C \\ D & E & F \\ G & H & I \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}$$

donde $(x_1,y_1)$ son las coordenadas iniciales de un punto, y $(x_1',y_1')$ son las coordenadas en el espacio transformado. La definición aquí mostrada trabaja con vectores columna en coordenadas homogéneas para representar los puntos y, por tanto, la multiplicación con la matriz de transformación se realiza por la derecha. En 3D la representación es similar, añadiendo una dimensión más.

Dentro del grupo de las transformaciones lineales, distinguimos 1) las transformaciones afines, que engloban las transformaciones de traslación, rotación -la reflexión se considera un tipo de rotación-, inclinación y cambio de escala, y 2) la transformación perspectiva. En las figuras 2 y 3 se muestran ejemplos de estas transformaciones y su formulación matricial. Se observa que las transformaciones afines mantienen el paralelismo de las líneas pero la transformación perspectiva no.

En visión artificial, utilizaremos las transformaciones afines para realizar cambios del sistema de coordenadas de referencia y la transformación perspectiva para proyectar puntos del espacio tridimensional (mundo 3D) al plano de la imagen. Las figuras 2 y 3 muestran las matrices de transformación asociadas a distintas transformaciones espaciales lineales, tanto afines como perspectiva.

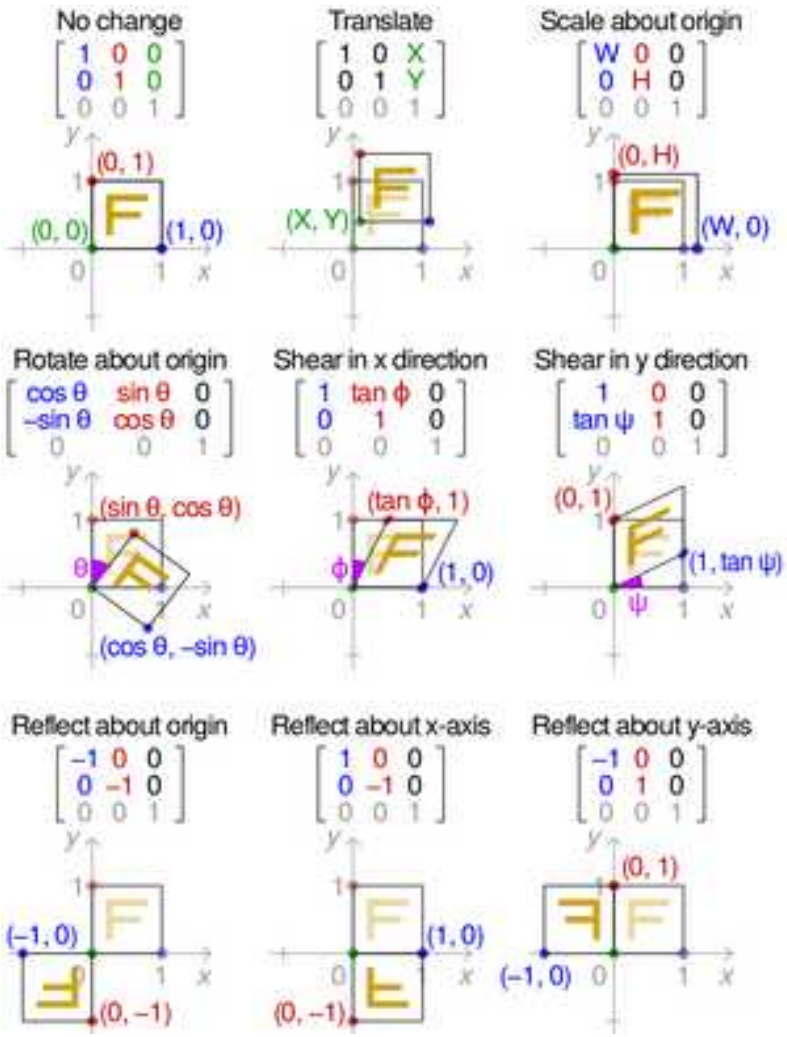



*Figura 2: Transformaciones afines [wiki-transf]*

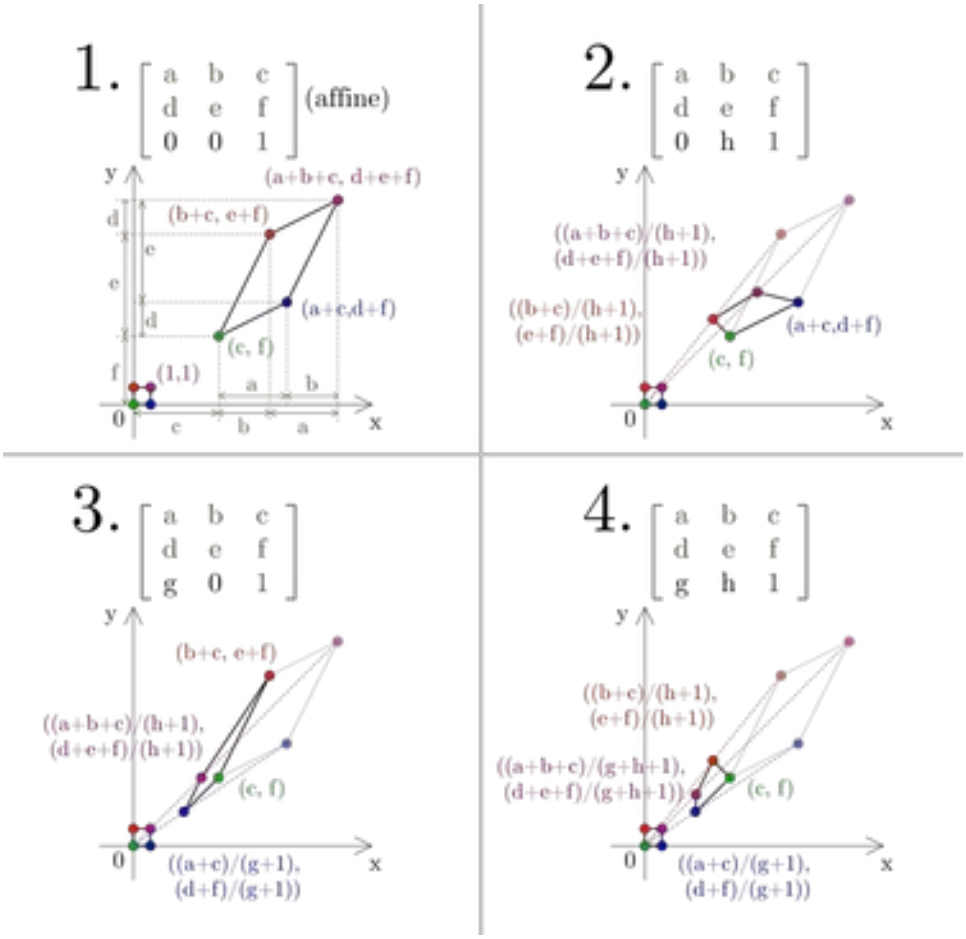



*Figura 3: Transformación afin (1) y varias transformaciones perspectiva (2,3,4) [wiki-transf]*

En cuanto a las transformaciones no lineales (aquellas en las que las líneas rectas no se mantienen rectas, como las mostradas en la segunda fila de la Figura 1), aunque algunas podríamos representarlas mediante matrices, la técnica más flexible y fácil de interpretar consiste en representar la malla con la localización de los nuevos puntos en la transformación (Figura 4).

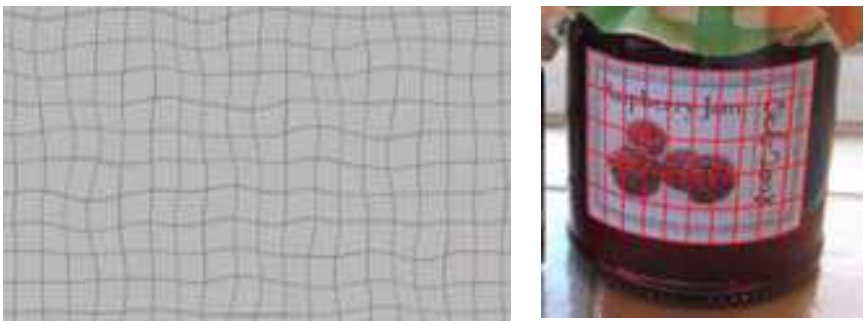


*Figura 4: Ejemplos de mallas de deformación no lineales*

Para practicar todos estos tipos de transformaciones, implemente en python un programa que realice las siguientes operaciones:

a) Reescalar las imágenes de las Figuras 5 y 6 para obtener una imagen de 200x200 pixels y rotarla 45 grados tomando como centro de rotación el centro de la imagen. Teniendo en cuenta que al rotar la imagen la “bounding box” paralela a los ejes que contiene la imagen es mayor que la propia imagen, configure el tamaño de la imagen de salida para que las esquinas de la imagen original toquen los bordes de la imagen de salida (para que no se pierda la información de una parte de la imagen).

*Figure 5: zigzag.jpg*

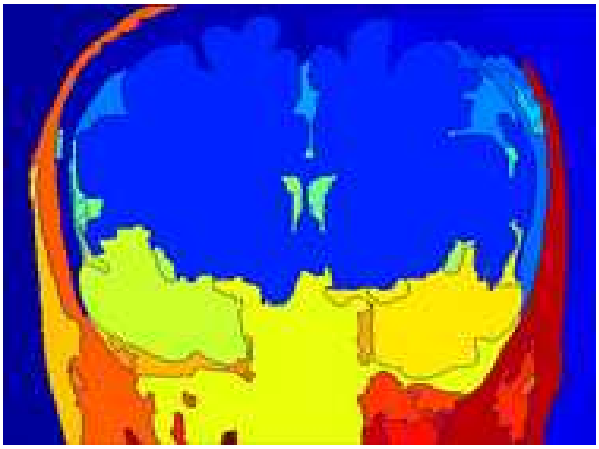


*Figure 6: brainLabels.png*

b) Transformación afin compuesta:

Dada la imagen de la Figura 5, se desea realizar la transformación compuesta, Tc, siguiente:

T1: Reescalar a un cuarto en ambos ejes la imagen.

T2: Inclinar (shear) -30 grados la imagen obtenida de T1 (transformación “shear”, se mantienen las coordenadas “y”; dado que la imagen se representan en coordendas ij (el origen (0,0) es la esquina superior izquierda de la imagen), la imagen se saldrá del primer cuadrante),

T3: girar 90 grados a la izquierda (con centro de giro en el centro de la imagen) la imagen obtenida de T2

Tenga en cuenta que a la salida no se debe perder información de la imagen (se deben mantener todos los píxeles de la Figura 5).

b.1) Implementar el código para realizar Tc de dos maneras distintas:

b.1.1) Implementando cada transformación individual (T1,T2,T3) por separado (con matrices de transformación distintas y mostrando los resultados parciales).

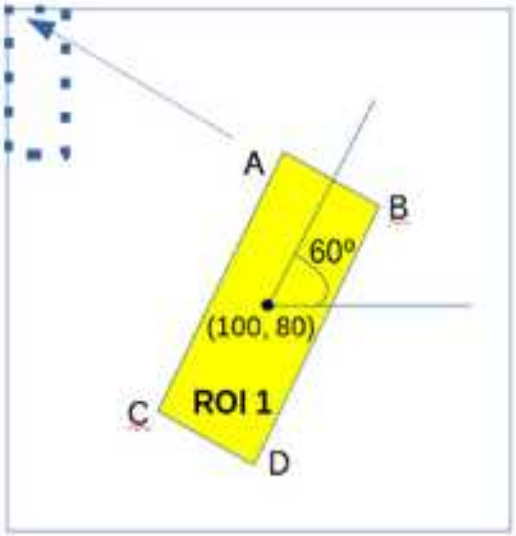
b.1.2) Implementando la transformación Tc en un solo paso (con una única matriz de transformación obtenida por composición de matrices).

Además, en la narración de la memoria, justifique los siguientes puntos:

- El orden de las transformaciones afecta al resultado de la transformación (a la posición final que ocupan los píxeles de la imagen).

- La interpolación por el vecino más cercano hay que usarla cuando ...

b.2) Dada la transformación indicada en la Figura 7. Proponga la secuencia de transformaciones simples (rotaciones, traslaciones, cambios de escala, ...) que permitan recolocar la región rectangular ROI 1 a la posición objetivo. Genere una imagen sintética con las especificaciones indicadas y programe la solución en python. Tenga en cuenta que no se puede perder el contenido de la subimagen ROI1 en ningún momento.




**Posición inicial ROI 1:**
centro de masas = (100,80)

d(A,C) = d(B,D) =2*d(A,B) =
2* d(C,D) = 40

donde d(X,Y) = distancia euclídea

sin(60)=sqrt(3)/2; cos(60) = 1/2

**Posición objetivo:**
A = (0,0)
B = (10,0)
C = (0,20)
D = (10,20)

*Figura 7: Transformación para extracción de una región de interés de una imagen.*

c) Transformación no lineal:

c.1) Deforme la imagen de la Figura 5 de manera que la mitad izquierda de la imagen quede comprimida en un tercio de la imagen final y la mitad derecha se expanda para ocupar los dos tercios restantes. Justifique qué operador considera que se debe utilizar para realizar esta operación warpAffine o warpPerspective. ¿Esta transformación es no lineal? Justifique por qué.

c.2) Interprete el resultado de aplicar la siguiente transformación a la imagen de la Figura 8:

$$y' = y + 0.4*(x-0.5)^2 - 0.1$$

$$x' = x$$

Considere las posiciones de los píxeles de la imagen normalizadas entre 0 y 1 en ambos ejes.

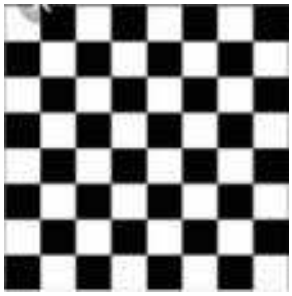


*Figura 8: Imagen de tablero de ajedrez.*

**Referencias:**

https://en.wikipedia.org/wiki/Transformation_matrix

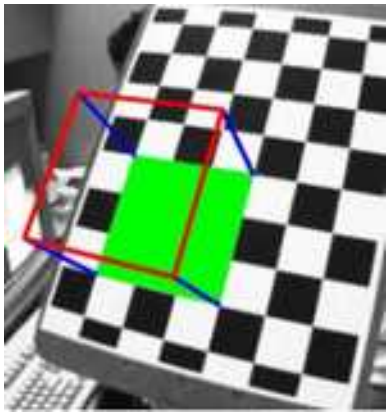
https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html

https://docs.opencv.org/4.5.3/d6/d00/tutorial_py_root.html

**3) Calibración de la cámara:**

3.1.- Realice los tutoriales de openCV dedicados a la calibración de la cámara y estimación de la pose. Las imágenes para la calibración se encuentran en el directorio “PEC1/data/calib”.

Deberá modificar el código para dibujar un cubo indicando la orientación del tablero como el mostrado en la imagen siguiente pero cambiando el color de la base de verde a amarillo y las aristas superiores de rojo a verde.



**Referencias a los tutoriales:**

https://docs.opencv.org/master/d9/db7/tutorial_py_table_of_contents_calib3d.html

https://www.youtube.com/watch?v=jfvh_F-jjzE (video con explicación sencilla y en español)

3.2.- Haga la calibración de su propia cámara: tome fotos con su cámara (la de su movil o la webcam) y haye sus parámetros intrínsecos. Describa el proceso seguido y presente los resultados obtenidos. ¿Como ha conseguido que el patrón ”tablero de ajedrez” esté situado rígidamente sobre un plano?

3.3.- Suponga se dispone de un sistema de visión artificial que puede reconocer y localizar en la imagen partes significativas de la cabeza de una persona (región de los ojos, de las orejas, de la nariz, de la barbilla, etc. Teniendo en cuenta el ejercicio 3.1, ¿se le ocurre alguna manera de estimar hacia adónde apunta la cabeza de la persona (hacia adónde mira)? Justifique cómo lo haría.

**4.- Flujo óptico y seguimiento (Tracking)**

4.1.- Realice el tutorial [1] y analice el código que hay en la carpeta “data” [2]. Comente la diferencia entre los enfoques utilizados por los dos programas: lukas_kanade_track.py y optical_flow1.py.

¿Por qué el primero no detecta el movimiento en las regiones suavizadas de la imagen?

Describa 3 maneras de visualizar el flujo óptico presente en una secuencia de imágenes.

4.2.- Utilice el algoritmo meanshift [3] para implementar una aplicación para seguir el movimiento de una mano en una webcam de escritorio. Detalle el protocolo y las consideraciones a tener en cuenta para utilizar correctamente la aplicación.

Referencias:

[1] https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html

[2] PEC1/data/opticalFlow/lukas_kanade_track.py

PEC1/data/opticalFlow/optical_flow1.py

[3] Algoritmo Meanshift. https://en.wikipedia.org/wiki/Mean_shift

**5) Segmentación sin conocimiento del dominio**

5.1.- Estudie el tutorial [1] y analice las diferencias entre umbralización global y local (global vs adaptive thresholding) en las imágenes TH_noisy2.png, TH_TE.gif, TH_scanTable.png del directorio en las imágenes del directorio PEC1/data/threshold/ con distintas configuraciones de parámetros.

5.2.- ¿Sería posible segmentar la región correspondiente al rectángulo blanco en la imagen TH_noisy2.png o la pieza en forma de T en TH_TE.gif utilizando umbralización local?

5.3.- Clustering k-means. Estudie el algoritmo de segmentación k-means y aplíquelo a la segmentación de la Figura 11, de manera que se distinga la sustancia blanca (región más clara del cerebro) y la sustancia gris (región menos clara, grisácea). Tenga en cuenta que en la cabeza existen otras estructuras, como son: el cráneo (hueso, hiperintenso) y el líquido cerebroespinal (son los píxeles negros dentro del cráneo que no se distinguen del fondo porque esta sustancia no se aprecia en esta modalidad de imagen de resonancia magnética). No espere una segmentación perfecta, pero sí aproximada. Pruebe con distinto número de clases y justifique la configuración que obtiene visualmente el mejor resultado.

5.3.- Segmentación con el algoritmo meanshift: ejemplifique el uso de meanshift para la segmentación de las imagenes de las figuras 9 y 10. Comente cómo influyen los parámetros de configuración del algoritmo. ¿Qué diferencias encuentra a la hora de aplicarlo respecto al uso para tracking que hizo en el apartado 4.2?

*Figure 9: brain1.png*

*Figure 10: avion1.png*

[1] https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html

[2] https://en.wikipedia.org/wiki/Mean_shift

[3] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUZEL1/MeanShift.pdf

[4] https://docs.opencv.org/3.0-beta/modules/imgproc/doc/filtering.html#pyrmeanshiftfiltering

[5] https://docs.opencv.org/3.4/d7/d00/tutorial_meanshift.html

5.4.- Algoritmo de watershed: Estudie el algoritmo y realice el siguiente tutorial:
https://docs.opencv.org/master/d3/db4/tutorial_py_watershed.html

Dada la imagen de la Figura 11, implemente un script que utilice el algoritmo watershed para segmentar las regiones correspondientes a los lóbulos temporales del cerebro, lo cual implica separar estas regiones de la región central del cerebro (tal como aparece marcado en la Figura 12 con las líneas rojas). Justifique las decisiones de diseño tomadas.

* Observaciones:

- El proceso completo requiere de varios pasos.

- Tenemos información del dominio: en el cerebro, la sustancia blanca está rodeada de sustancia gris.

- Localizar la sustancia blanca es relativamente fácil.

- El objetivo es separar en regiones diferentes la sustancia gris correspondiente a los lóbulos de la correspondiente a la parte central del cerebro (estos dos volúmenes cerebrales están en contacto y por eso su sustancia gris aparece pegada y sin cambio de color).

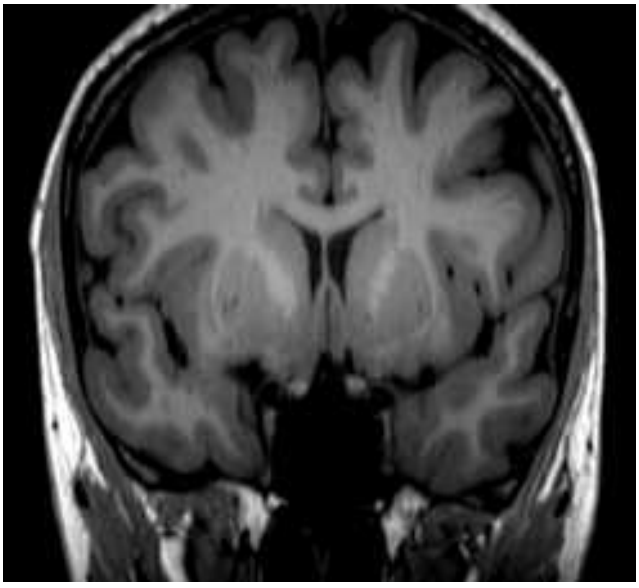


*Figure 11: Imagen de resonancia magnética del cerebro modalidad T1 (brain1.png).*

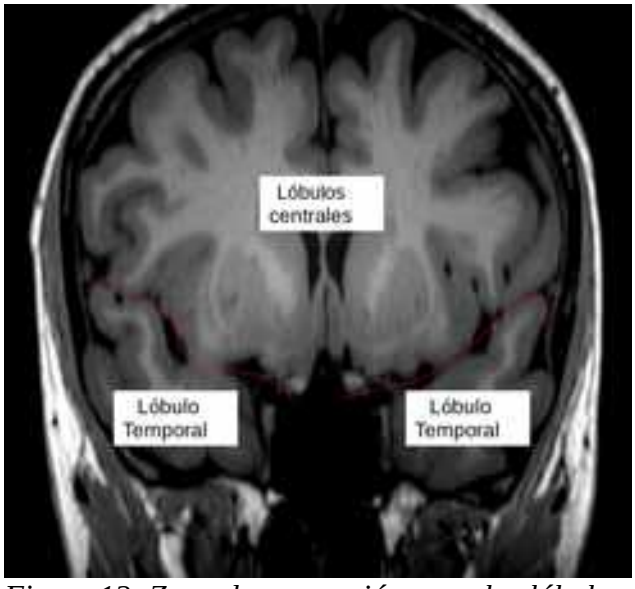



*Figure 12: Zona de separación entre los lóbulos temporales y centrales (líneas rojas).*