PEC1: Prueba de evaluación continua 1

1) Eliminación de ruído en imágenes escaneadas.

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: deberá analizar las imágenes para identificar los distintos tipos de ruido existentes y buscar soluciones independientes para cada tipo de ruido encontrado (no existe una solución única que cubra todos los posibles tipos de ruido). Las soluciones podrán ser globales a toda la imagen o locales a una región de interés (ROI). Se aconseja construir un interfaz sencilla que permita cargar una imagen y aplicarle, de manera interactiva y secuencial, los operadores de eliminación de ruido necesarios en cada caso. Tenga 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), lo que implicará una secuencia de operaciones más o menos compleja (esqueletos, dilataciones, erosiones, ...).

En la memoria, utilice las imágenes N_328_THS_TOTAL-ev1-h.png y N_307_GLS_TOTAL-ev5-h.png para mostrar la secuencia de operadores aplicados y los resultados parciales obtenidos hasta llegar al resultado final.

Referencias:

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

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

2) Transformaciones geométricas.

Dentro del conjunto de transformaciones geométricas (toda aplicación biyectiva f del plano (o 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 a otro cuadrilátero de manera que la distancia entre los puntos puede variar pero se mantiene la estructura interna de la malla (ver Fig. 1). Obsérvese que estas transformaciones no modifican el valor de

intensidad, solo su posición* (* 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 y la del vecino más cercano).

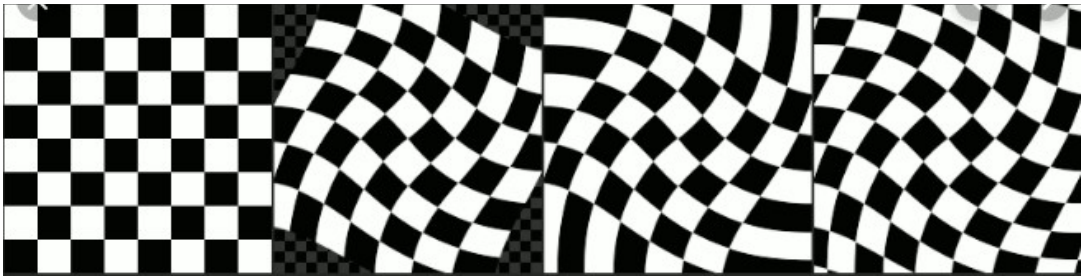


Figura 1: Transformaciones 2D que mantienen la topología

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 las transformaciones afines, que engloban las transformaciones de traslación, rotación -la reflexión se considera un tipo de rotación-, inclinación y cambio de escala, y 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 transformaciones perspectiva para trasladar un plano del mundo 3D al plano de la imagen.

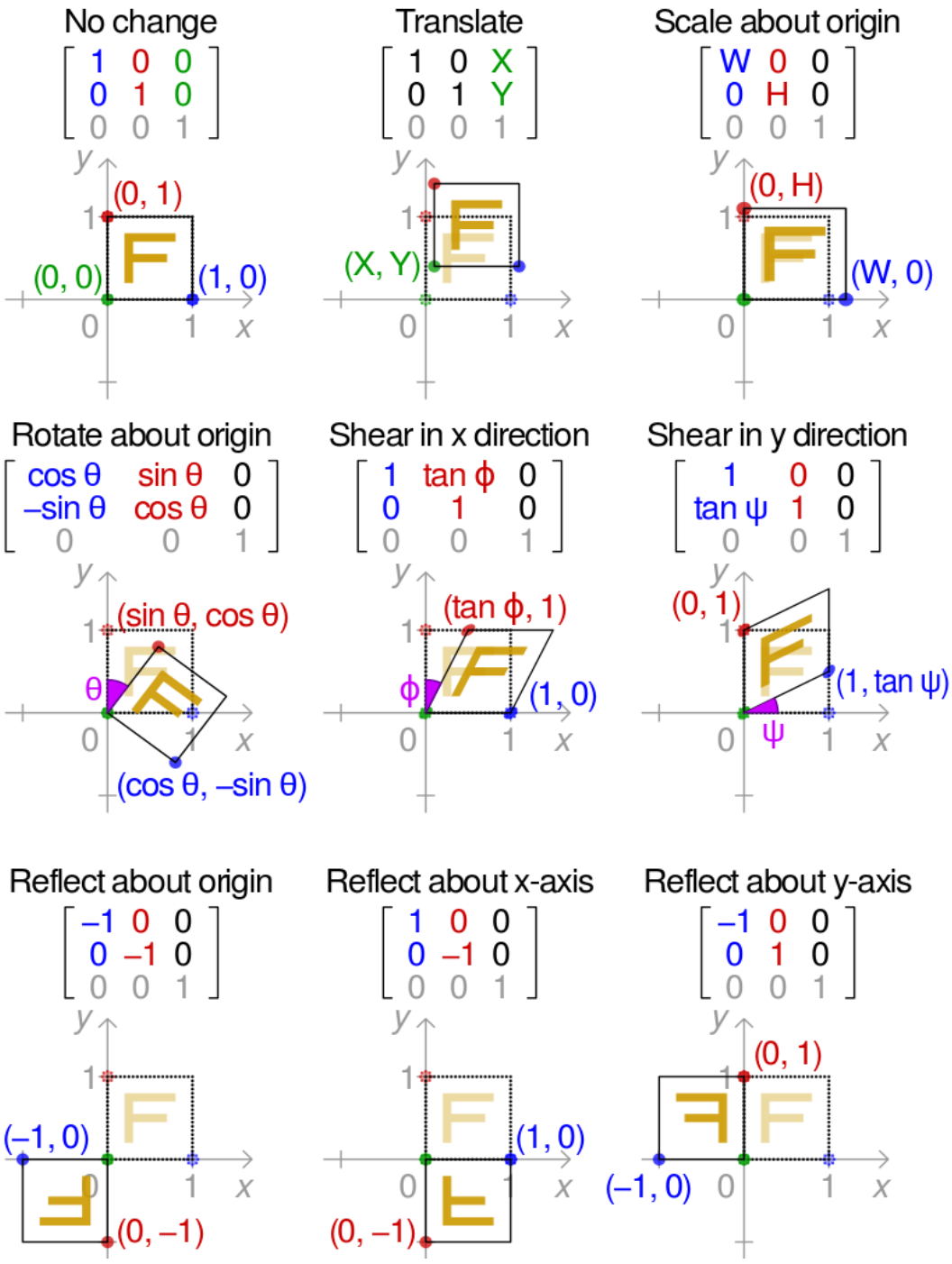


Figura 2: Transformaciones afines [wiki-transf]

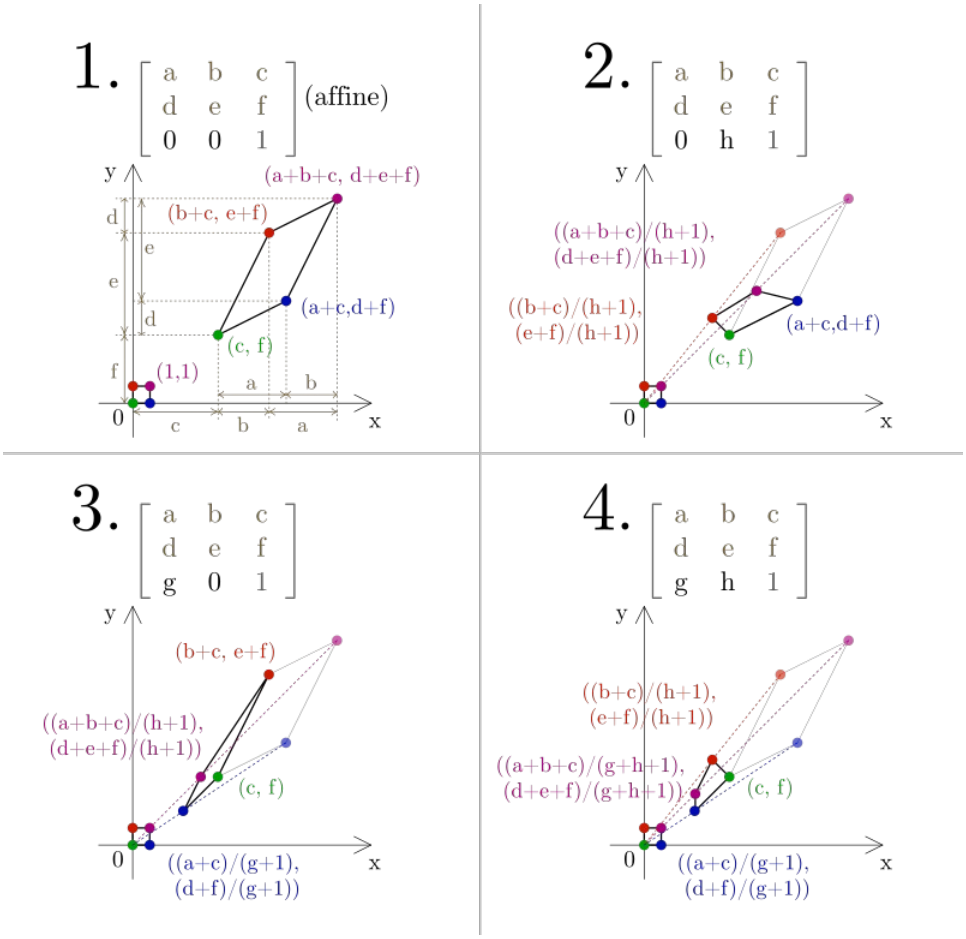


Figura 3: Transformación afin (1) y varias transformaciones perspectiva (2,3,4) [wiki-transf]

En cuanto a las transformaciones no lineales (aquellas en las que las líneas rectas no se mantienen rectas), 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 (fig. 4).

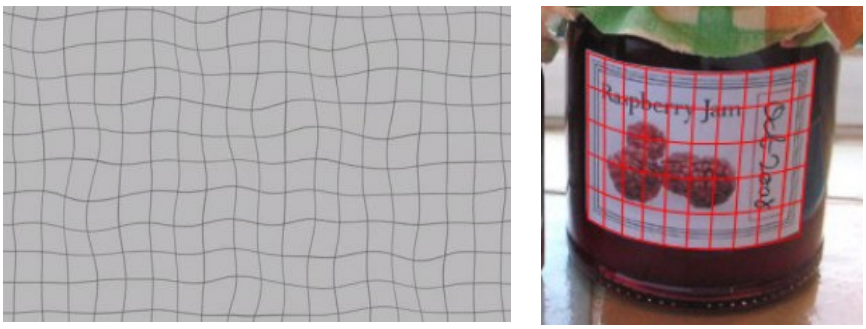


Figura 4: Ejemplos de malla de deformaciónno lineales

Para practicar todos estos tipos de transformaciones, implemente en python un programa que realice las siguientes operaciones sobre la imagen de la figura 5 (todas las operaciones de los distintos apartados se realizarán sobre la imagen original).

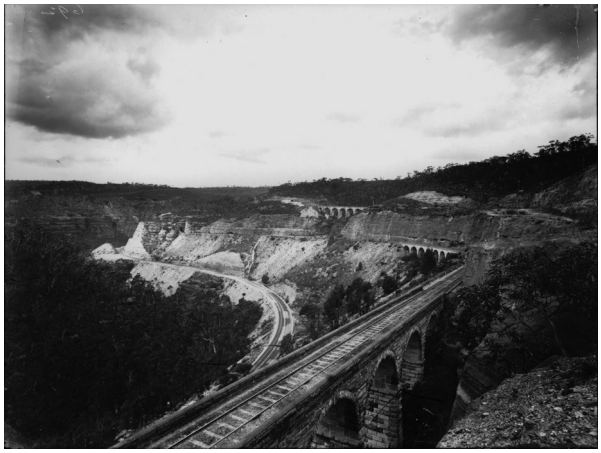


Figure 5: zigzag.jpg

a) Reescalar la imagen de la figura 5 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 la contiene es mayor, configure el tamaño de la imagen de salida para que las esquinas de la imagen original toquen los bordes de la imagen de salida.

b) Transformación afin compuesta:

b.1) Dada la imagen de la Figura 5 , se desea realizar la transformación compuesta, Tc, siguiente:

1.- T1: Inclinar 30 grados a la derecha la imagen de entrada (transformación “shear”, se mantienen las coordenadas “y”),

2.- T2: girar 90 grados a la izquierda (con centro de giro en el centro de la imagen) la imagen obtenida de T1

3.- Reescalar a la mitad en ambos ejes la imagen obtenida de T2.

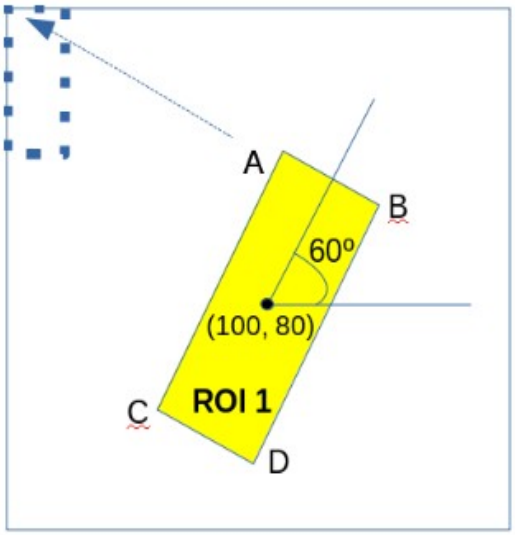
Teniendo en cuenta que a la salida no se debe perder información de la imagen (se deben mantener todos los píxeles de la figura 5), implementar el código para realizar Tc de dos maneras distintas:

b.1.1) Implementando cada transformación individual (T1,T2,T3) por separado (con matrices de transformación distintas).

b.1.1) Implementando la transformación Tc en un solo paso (con una única matriz de transformación obtenida por composición de matrices).

b.2) ¿Cree que el orden de las transformaciones es importante? Justifíquelo formalmente.

b.3) Dada la transformación indicada en la Figura 6. 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 6: Transformación para extracción de una región de interés de una imagen.

c) Transformación polar:

c.1) utilizar la función cv2.warpPolar para obtener una imagen logPolar cuadrada, centrada en el centro de la imagen original con una fila por grado y ajustada para ver toda la imagen original. Ver https://docs.opencv.org/4.5.3/da/d54/group__imgproc__transform.html#ga49481ab24fdaa0ffa4d3e63d14c0d5e4 . Comentar el resultado obtenido.

c.2) recuperar la imagen original a partir de la obtenida en el aparado c.1). Comentar el resultado obtenido.

e) Transformación no lineal:

- deformar la imagen original 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.

- ¿Qué operador considera que se debe utilizar para realizar esta operación warpAffine o warpPerspective? Justifique la respuesta.

- Justifique por qué es una transformación no lineal.

Referencias:

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

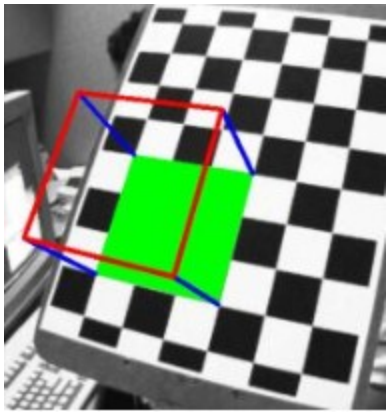
https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html

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

3) Calibración de la cámara:

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

Deberá entregar el código con los resultados de la calidad de la calibración y con la implementación que dibuja un cubo indicando la orientación del tablero cambiando el color de la base de verde a rojo y las aristas superiores de rojo a amarillo.



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, sencillo y en español)

3.2.- Suponga que puede localizar puntos significativos de la cabeza de una persona (esto se hará más adelante en el curso). ¿Se le ocurre alguna manera de estimar hacia adónde apunta la cabeza (hacia adónde mira)? Describa cómo lo haría.

3.3.- Haga la calibración de su propia cámara (tome fotos con su cámara web o con su teléfono y haye sus parámetros intrínsecos). Describa cómo lo realizó y presente los resultados obtenidos.

3.4.- Si tiene una cámara a la que se le pueden acoplar dos objetivos, uno de 25mm de distancia focal y otro de 50mm, ¿cual eligiría para obtener imágenes con mayor ángulo de visión? Justifique la respuesta.

4.- Flujo óptico.

Lea el tutorial [1] y analice el código que hay en la carpeta "data" [2].

4.1.- Comente la diferencia entre los enfoques utilizados por los dos programas:
lukas_kanade_track.py y optical_flow1.py

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

4.3.- ¿Qué hace la función draw_hsv() en optical_flow1.py?

4.4.- ¿Qué información contienen las variables "flow" y "res" en la función warp_flow() de optical_flow1.py?

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

5) Segmentación sin conocimiento del dominio

5.1 Explique la diferencia entre umbralización global (global thresholding) y local (adaptive thresholding). Indique las ventajas e inconvenientes de cada una de ellas.

Utilice openCV para ejemplificarlo sobre las imágenes del directorio PEC1/data/threshold/

5.2) Clustering k-means. Describa brevemente, con sus palabras, la idea detrás de este algoritmo de segmentación y dibuje un diagrama de flujo de su funcionamiento. Aplíquelo a la segmentación de la figura 7, de manera que se distinga la sustancia blanca (región más clara del cerebro) y la sustancia gris (región nenos 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 cárneo pero no se aprecian en esta modalidad de imagen de resonancia magnética, por lo que no se distinguen del fondo). No espere una segmentación perfecta, pero sí aproximada. Pruebe con distinto número de clases y justifique la configuración que obtiene visualmente el mejor resultado.

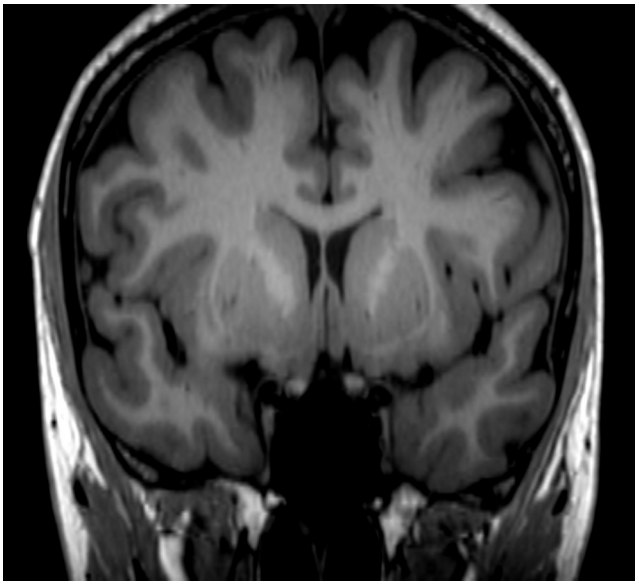


Figure 7: Imagen de resonancia magnética del cerebro modalidad T1 (brain1.png).

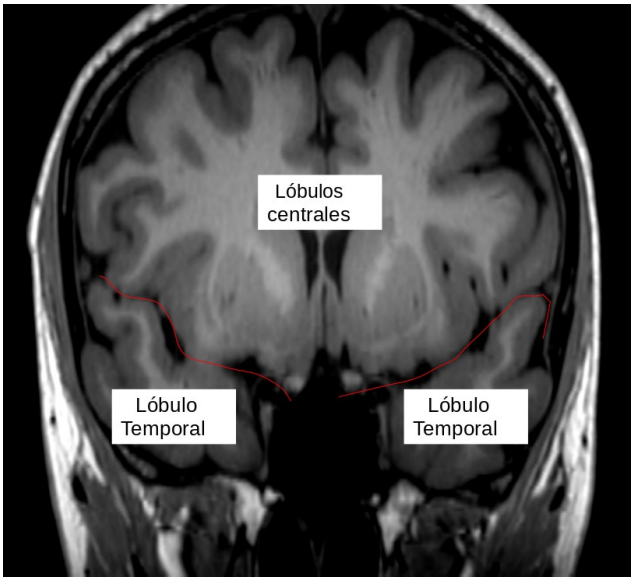


Figure 8: Zona de separación entre los lóbulos temporales y centrales.

5.3) Algoritmos de watershed y meanshift.

https://docs.opencv.org/master/d3/db4/tutorial_py_watershed.html

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

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

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

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

a) Describa brevemente, con sus palabras, la idea detrás de estos dos algoritmos de segmentación y dibuje un diagrama de flujo de su funcionamiento.

b) Indique ventajas e inconvenientes de cada uno y ponga un ejemplo de uso de cada método.

c) Comente si realmente se puede considerar que estos algoritmos son algoritmos de segmentación sin conocimiento del dominio.

5.4 Dada la figura 7, utilice el algoritmo más conveniente para segmentar las regiones correspondientes a los lóbulos temporales del cerebro, lo cual implica separarlas de la región central del cerebro (tal como aparece en la figura 8).

* Observación: Podemos suponer que conocemos la región de la sustancia blanca correspondiente a los lóbulos temporales y al lóbulo central. El objetivo es poder separar la sustancia gris perteneciente a los distintos lóbulos y que, cuando se unen los lóbulos temporales a la parte central, está unida (como se aprecia en la figura 7).