PRÁCTICA 2:

CORREGISTRO DE IMÁGENES

Fco. Javier L.Vivié Salas

Máster universitario en sistemas inteligentes (MUSI)

Processament d'Imatges Mèdiques (UIB 2021-2022)

Índice

Introducción.

A lo largo de la presente memoria se pretende mostrar el procedimiento seguido para la realización de la práctica.

En ella se propone realizar el corregistro entre una imagen anonimizada y un atlas anatómico estándar, además de realizar la validación cualitativa del corregistro. El hecho de corregistrar dos imágenes nos permite ver las diferencias entre un antes y un después de un suceso cualquiera como, por ejemplo, una operación. En nuestro caso, se utilizarán imágenes del cerebro humano para llevar a cabo la práctica, en ella se utilizarán:

- Imagen de un paciente anonimizado.
- Imagen de un paciente estándar o "*phantom*".
- Atlas anatómico estándar.

El objetivo final es poder visualizar la región del hipotálamo sobre la imagen de nuestro paciente.

Análisis preliminar de las imágenes.

Como hemos comentado durante la presentación de la práctica, se utilizarán tres tipos de imágenes:

- Imagen de un paciente anonimizado (a partir de ahora lo llamaremos TENSOR_BRAIN o TB). Nos muestra el resultado de realizar una resonancia magnética, o RM, de la cabeza de nuestro paciente.
- Imagen de un paciente estándar o "*phantom*". Se denomina de esta manera debido a que no es una imagen tomada directamente de un paciente como la imagen TENSOR_BRAIN, sino que es el resultado medio de multitud de RM de la cabeza de distintos pacientes, obteniendo así, unas proporciones "estándar" que cumplen la mayor parte de seres humanos. La conoceremos como PHANTOM_TENSOR.

  Hay que mencionar que en este caso disponemos de dos phantoms diferentes, ambos siendo el paciente medio pero con la diferencia en el número de vóxeles que presentan cada uno de los tensores. También es importante remarcar, que el tamaño de los vóxeles de los phantom no coincide con el tamaño de los vóxeles del TENSOR_BRAIN, por lo que debemos resolver este inconveniente, antes de nada. Por último, al representar este tensor, podemos comprobar que no se representa la cabeza en su totalidad, sino que falta la parte inferior de ella (en concreto toda la zona del maxilar inferior).

- Atlas anatómico estándar. Ésta es una imagen diferente a las otras dos, pues no es una imagen propiamente dicha, sino que es la segmentación por zonas de, en este caso, el cerebro, donde cada una de las zonas está representada con un único valor de píxel. En el caso de la práctica, la zona a visualizar (el hipotálamo), tiene un margen de valores que comprende desde el 121 hasta el 150, ambos incluidos.

Visualizador síncrono (de imagen doble).

Este apartado corresponde al primer paso a realizar para el desarrollo de la práctica. En él, se mostrará la implementación para el visualizador síncrono que se utilizará a lo largo de la práctica. En primer lugar, debemos saber que un visualizador de tres cortes nos permite ver dos imágenes simultáneamente, que pueden corresponder a los momentos previos y posteriores de una operación, como se ha comentado durante la introducción. En nuestro caso no solo permitirá verlas simultáneamente, sino que además podremos seleccionar el valor de un píxel y se nos mostrará en ambas imágenes dicho píxel. De esta manera, podremos ver si realmente el píxel seleccionado en la zona de la primera imagen corresponde a la misma zona en la segunda. A continuación, se puede ver un ejemplo de ello.

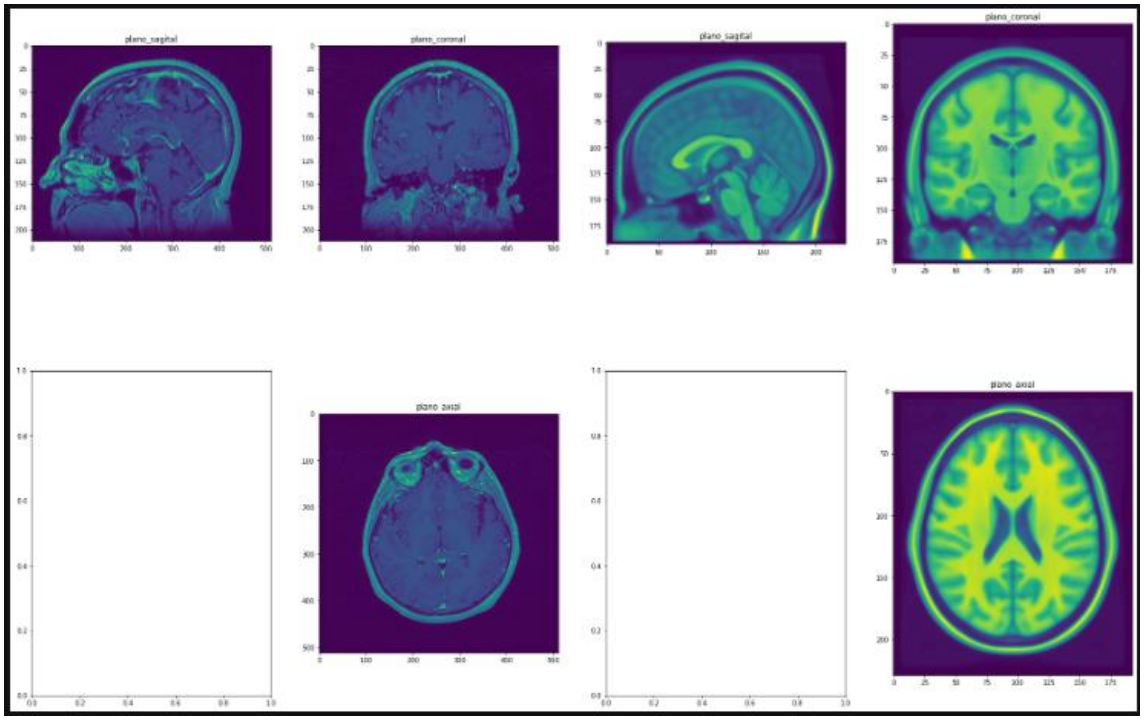


Ilustración 1: Visualizador síncrono.

Para obtener este resultado, se ha implementado el código que aparece en el anexo I. En él, podemos ver como se trata de un subprograma al que le podemos pasar cuatro parámetros, los dos tensores a visualizar (de los que nos mostrará las vistas sagital, coronal y axial) y en el caso

de que se quiera, visualizar el cursor, para ello deberemos declarar y pasarle el punto que queramos visualizar.

Podemos destacar que, aunque se haya definido un subprograma, en ciertas ocasiones no se podrá utilizar, puesto que se requiera de aplicar el algoritmo del pintor, algoritmo para el cuál no ha sido pensado el subprograma.

Corregistro de las imágenes.

En este apartado, el segundo de la práctica, veremos la mayor parte del trabajo a realizar durante la misma.

Introducción al corregistro.

En primer lugar, debemos hablar de qué es el corregistro; dadas dos imágenes, una imagen de entrada A y una imagen de referencia B, el objetivo de corregistrar dos imágenes es, a grosso modo, modificar la imagen A mediante traslaciones y rotaciones, de forma que al superponer A con B, éstas casen de la mejor manera posible. Esta herramienta permitiría ver claramente las diferencias entre las imágenes A y B si se visualizaran superpuestas. En nuestro caso, lo que se buscará obtener, serán las transformaciones necesarias (traslaciones y rotaciones) para transformar la imagen A en la imagen B. Sin embargo, la complicación de este punto es que no sólo se deberá hacer en la imagen 2D, sino que se tendrá que modificar todo el tensor que comprende la imagen del paciente anonimizado (TENSOR_BRAIN).

Tratamiento preliminar.

Como se ha comentado durante el análisis de las imágenes, el tamaño de los vóxeles entre ellas es diferente, por lo que, en primer lugar, debemos modificar su tamaño. Este paso es importante puesto que, de lo contrario, no se podrá aplicar el corregistro de una manera efectiva. Para conseguirlo, se ha utilizado la siguiente línea de código:

BRAIN_RESHAPE = zoom(TENSOR_BRAIN, pixel_len_mm)

Para ello vemos que utilizamos la instrucción “zoom“ de la librería scipy, esta instrucción nos permitirá reescalar el tamaño de los vóxeles de la imagen de nuestro paciente, para así poder aplicar el corregistro. El factor de reescalado utilizado, pixel_len_mm, corresponde a [1, 0.5078, 0.5078], parámetros obtenidos de las cabeceras de la imagen dicom del paciente. El resultado de reescalar los vóxeles se puede ver a continuación:

Procedimiento práctico.

Para llevar a cabo el apartado, es necesario tener claros los pasos que se deberán seguir.

Como se ha comentado en el apartado anterior, un punto importante a la hora de hacer el corregistro es que las dos imágenes tengan el mismo tamaño de vóxeles. Por otro lado, no solo basa con reescalar los vóxeles, sino que también es necesario que las imágenes tengan las mismas dimensiones:

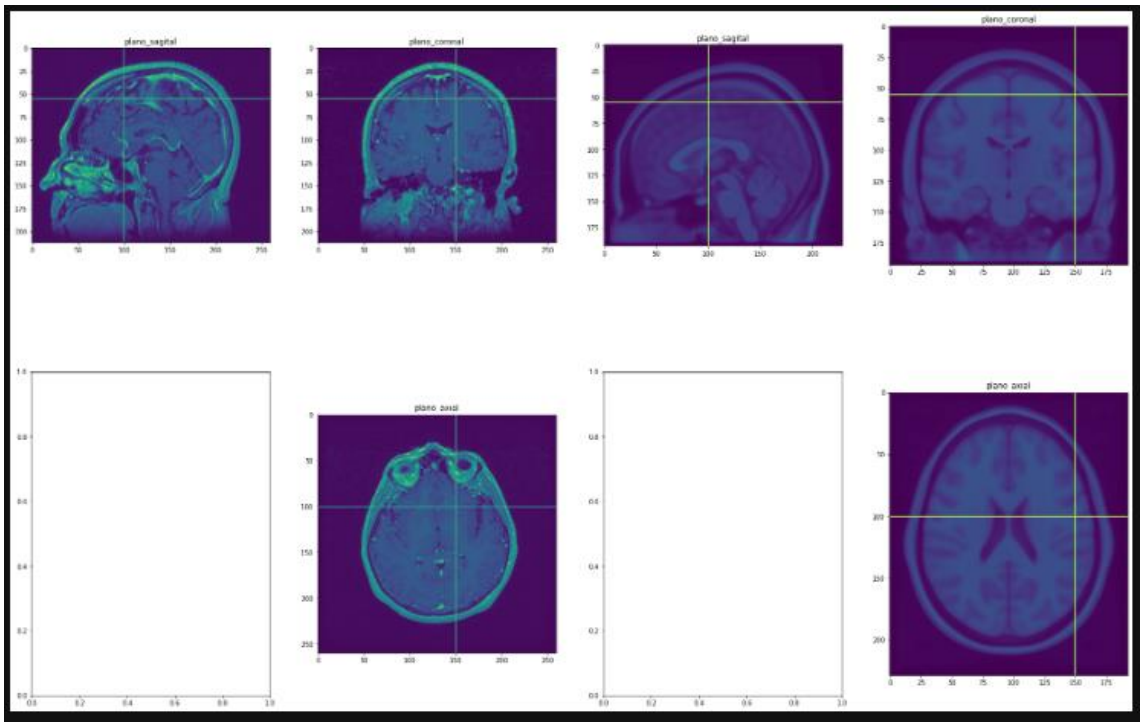


Ilustración 2: Los tensores tienen tamaños diferentes.

Podemos ver como las imágenes no tienen las mismas dimensiones. El siguiente paso es, por tanto, solventar este problema. Para ello se procede a recortar los bordes del tensor, de manera que las dimensiones cuadren entre el tensor del paciente y el tensor “phantom”.

Aplicación del corregistro.

En este momento ya nos es posible empezar a aplicar el corregistro, para ello se volverá a utilizar la librería "*scipy*", en concreto, las instrucciones: *ndimage.shift* y *ndimage.interpolation.rotate*. En nuestro caso, se ha aplicado un vector de traslación de (17, -1, 0) y un vector de rotación de (8, 0 ,0). El código utilizado es el correspondiente al anexo II, y a continuación, se puede observar el resultado de haber aplicado el corregistro:

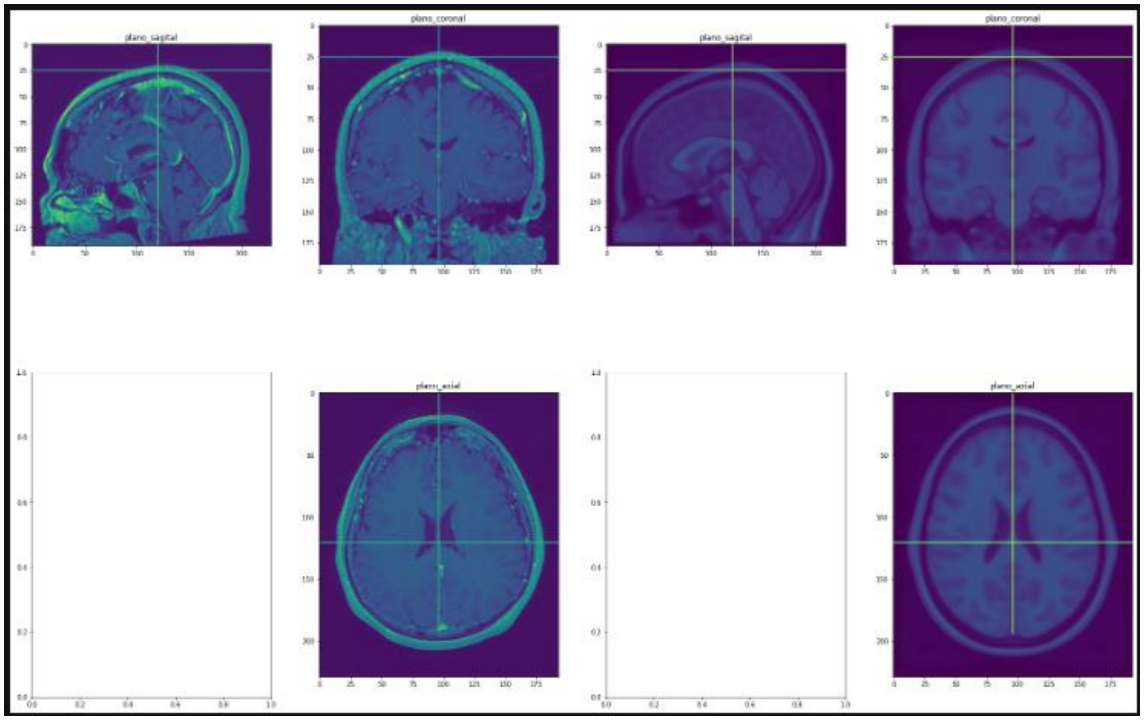


Ilustración 3:Resultado del corregistro.

En este caso la función de pérdida diseñada consiste en la diferencia entre los tensores, mediante la instrucción absdiff, por lo que el resultado final nos indicará, mediante un entero, la diferencia que existe entre los tensores. Esta diferencia idealmente tiene un valor de 0, por lo que se buscará minimizar este valor lo máximo posible.

Si aplicamos el algoritmo del pintor para representar la superposición de los tensores, obtenemos el siguiente resultado:

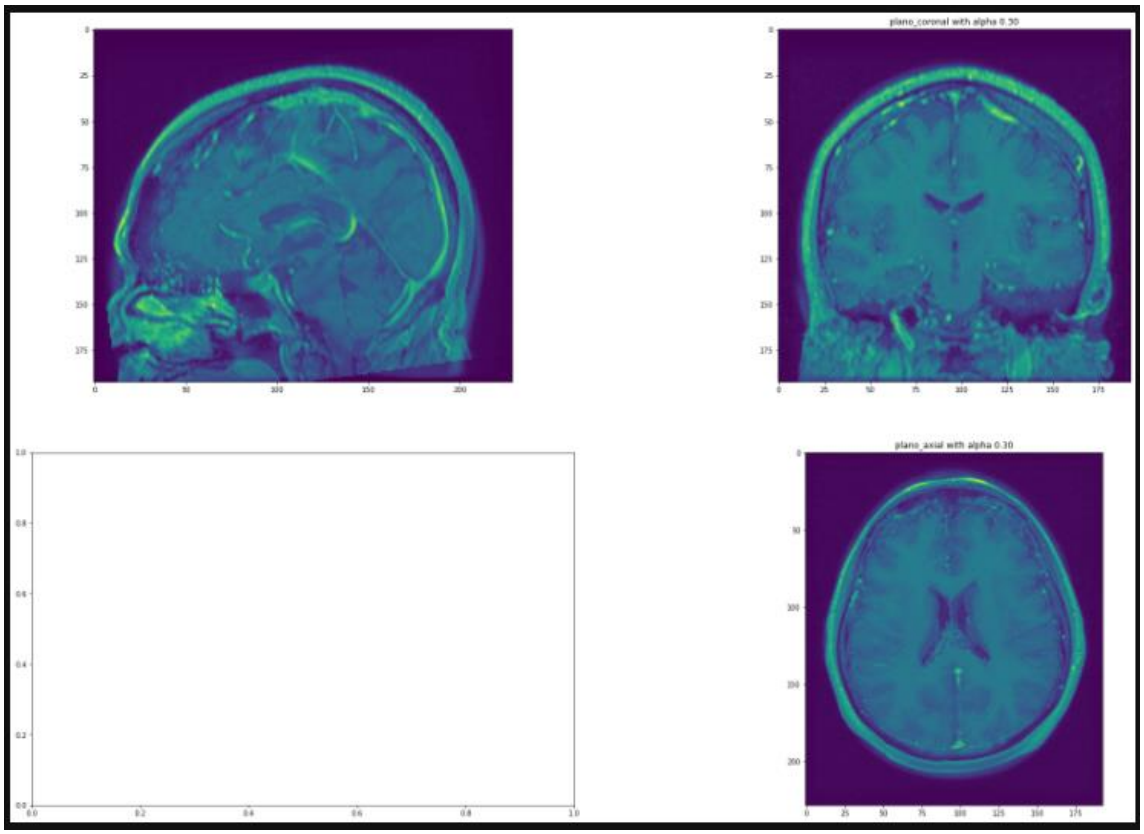


Ilustración 4:Superposición de los tensores del corregistro

Se puede observar que el resultado tras haber aplicado el corregistro es bastante satisfactorio, a excepción de la vista sagital, que como se puede ver, es donde hay pequeñas diferencias entre las imágenes. Supuestamente debido a que las posiciones espaciales de los tensores no son la misma y, sin embargo, no se ha podido corregir con el corregistro, aún con ese pequeño detalle, podemos afirmar que la coincidencia entre ellas es bastante alta.

Visualización del hipotálamo sobre la imagen.

Para finalizar, en esta última parte de la práctica se busca representar la zona del hipotálamo dentro del tensor modificado. Esta será una manera de comprobar que las transformaciones hechas en el paso anterior sean las correctas, puesto que, de lo contrario, no podremos visualizarlo.

Para este último apartado, volveremos a utilizar el visualizador síncrono desarrollado en la primera parte juntamente con el algoritmo del pintor utilizado durante la primera práctica.

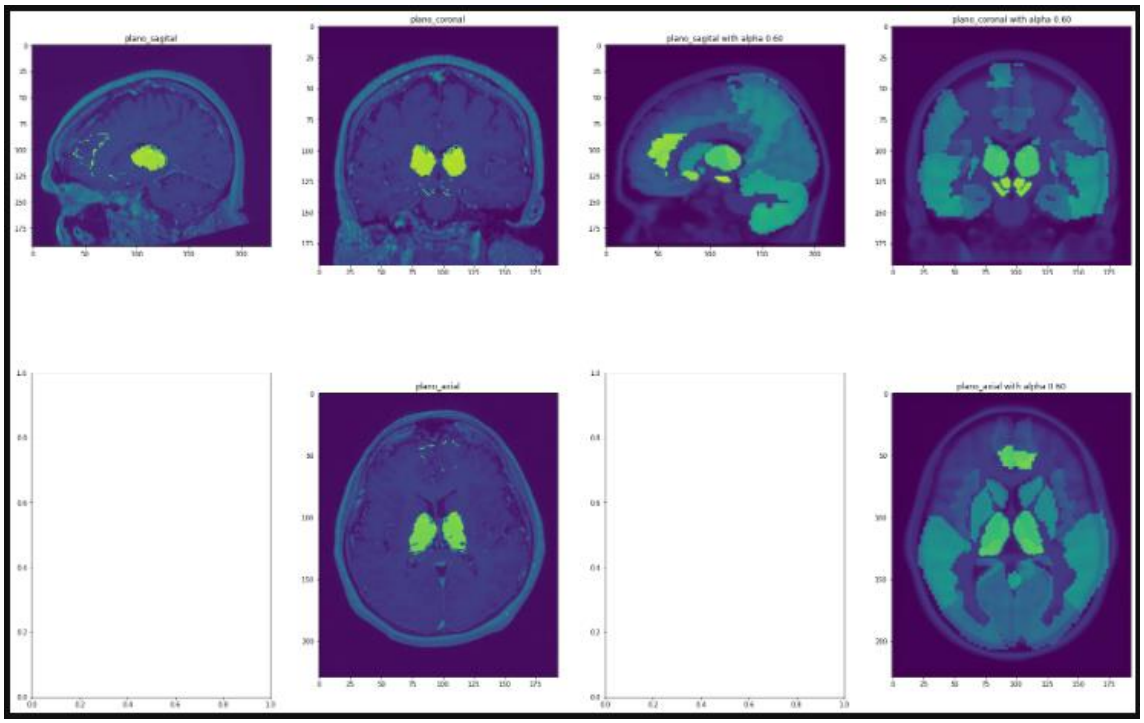
Hay que recordar que, el algoritmo del pintor juega con la transparencia de las imágenes, de forma que se puedan superponer una encima de la otra y ver el resultado de la combinación. De hecho, el algoritmo del pintor es uno de los métodos propuestos para poder visualizar la eficacia

del corregistro durante el primer apartado, nosotros, por el contrario, hemos decidido optar por implementar el visualizador síncrono.

Por tanto, dado que tenemos TENSOR_BRAIN transformado de tal forma que está corregistrado con el PHANTOM_TENSOR, únicamente queda “pintar” encima del primero el atlas comentado al inicio de la práctica, haciendo servir el método del pintor explicado anteriormente.

El código implementado para realizar este paso corresponde a los códigos correspondientes a los anexos III y IV.

El resultado de utilizar el código es el que se puede ver a continuación:



Gracias a este último paso, podemos ver que el corregistro ha salido como se esperaba. En el lado derecho podemos observar el resultado de superponer el Atlas sobre el tensor del “Phantom” para tener una referencia, mientras que, a la derecha, podemos ver la imagen de nuestro paciente con el hipotálamo representado.

Conclusiones.

A lo largo de la práctica se ha podido ver cómo se han afrontado diversos problemas, como el de la diferencia en el tamaño de los vóxeles o la diferencia de tamaño en los propios tensores. Finalmente, hemos podido aplicar una serie de transformaciones a la imagen para completar el corregistro y además, se ha mostrado la localización del hipotálamo sobre la imagen del paciente a partir del Atlas.

Directorio github:

Para acceder al código se ha subido a un repositorio Github, al cual se puede acceder desde el siguiente enlace:

Anexos.

Anexo I: Código visualizador síncrono.

```
def visualiz_sincrono(TENSOR_A, TENSOR_B, POINT = (0, 0, 0), PAINTED = False):

    sagital_A = TENSOR_A[:, :, TENSOR_A.shape[2]//2]
    coronal_A = TENSOR_A[:, TENSOR_A.shape[1]//2, :]
    axial_A = TENSOR_A[TENSOR_A.shape[0]//2, :, :]

    sagital_B = TENSOR_B[:, :, TENSOR_B.shape[2]//2]
    coronal_B = TENSOR_B[:, TENSOR_B.shape[1]//2, :]
    axial_B = TENSOR_B[TENSOR_B.shape[0]//2, :, :]

    #-----------------------------------------------------------------------------------------------------------------------

    if (PAINTED == True):

        sagital_paint_A = sagital_A.copy()
        coronal_paint_A = coronal_A.copy()
        axial_paint_A = axial_A.copy()

        sagital_paint_B = sagital_B.copy()
        coronal_paint_B = coronal_B.copy()
        axial_paint_B = axial_B.copy()

        sagital_paint_A = cv2.line(sagital_paint_A, (point[0], 0), (point[0], sagital_paint_A.shape[1]), (255,0,0), 1)
        sagital_paint_A = cv2.line(sagital_paint_A, (0, point[2]), (sagital_paint_A.shape[1], point[2]), (255,0,0), 1)

        coronal_paint_A = cv2.line(coronal_paint_A, (point[1], 0), (point[1], coronal_paint_A.shape[1]), (255,0,0), 1)
        coronal_paint_A = cv2.line(coronal_paint_A, (0, point[2]), (coronal_paint_A.shape[1], point[2]), (255,0,0), 1)

        axial_paint_A = cv2.line(axial_paint_A, (point[1], 0), (point[1], axial_paint.shape[1]), (255,0,0), 1)
        axial_paint_A = cv2.line(axial_paint_A, (0, point[0]), (axial_paint.shape[1], point[0]), (255,0,0), 1)

        sagital_paint_B = cv2.line(sagital_paint_B, (point[0], 0), (point[0], sagital_paint_B.shape[1]), (255,0,0), 1)
        sagital_paint_B = cv2.line(sagital_paint_B, (0, point[2]), (sagital_paint_B.shape[1], point[2]), (255,0,0), 1)

        coronal_paint_B = cv2.line(coronal_paint_B, (point[1], 0), (point[1], coronal_paint_B.shape[1]), (255,0,0), 1)
```

```
    coronal_paint_B = cv2.line(coronal_paint_B, (0, point[2]), (coronal_paint_B.shape[1], point[2]), (255,0,0), 1)

    axial_paint_B = cv2.line(axial_paint_B, (point[1], 0), (point[1], axial_paint.shape[1]), (255,0,0), 1)
    axial_paint_B = cv2.line(axial_paint_B, (0, point[0]), (axial_paint.shape[1], point[0]), (255,0,0), 1)

  #-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------

  if (PAINTED == False):
    fig, axs = plt.subplots(2,4, figsize=(30,20))
    axs[0,0].imshow(sagital_A)
    axs[0,1].imshow(coronal_A)
    axs[1,1].imshow(axial_A)

    #cambiar les seguents imatges per les corresponents de l'atlas
    axs[0,2].imshow(sagital_B)
    axs[0,3].imshow(coronal_B)
    axs[1,3].imshow(axial_B)

    axs[0,0].set_title("plano_sagital")
    axs[0,1].set_title("plano_coronal")
    axs[1,1].set_title("plano_axial");

    axs[0,2].set_title("plano_sagital")
    axs[0,3].set_title("plano_coronal")
    axs[1,3].set_title("plano_axial");

  if (PAINTED == True):
    fig, axs = plt.subplots(2,4, figsize=(30,20))
    axs[0,0].imshow(sagital_paint_A)
    axs[0,1].imshow(coronal_paint_A)
    axs[1,1].imshow(axial_paint_A)

    #cambiar les seguents imatges per les corresponents de l'atlas
    axs[0,2].imshow(sagital_paint_B)
    axs[0,3].imshow(coronal_paint_B)
    axs[1,3].imshow(axial_paint_B)

    axs[0,0].set_title("plano_sagital")
    axs[0,1].set_title("plano_coronal")
    axs[1,1].set_title("plano_axial");

    axs[0,2].set_title("plano_sagital")
    axs[0,3].set_title("plano_coronal")
    axs[1,3].set_title("plano_axial");
```

Anexo II: Algoritmo de transformación del tensor.

```
def TRASLATE_TENSOR(TENSOR, vector):
    shifted = ndimage.shift(TENSOR, (vector[0], vector[1], vector[2]))
    return shifted
def ROTATE_TENSOR(TENSOR, ROT_VECTOR):
    TENSOR = ndimage.interpolation.rotate(TENSOR, ROT_VECTOR[0], (0,1),reshape=False)
    TENSOR = ndimage.interpolation.rotate(TENSOR, ROT_VECTOR[1], (0,2),reshape=False)
    TENSOR = ndimage.interpolation.rotate(TENSOR, ROT_VECTOR[2], (1,2),reshape=False)
    return TENSOR

t_vector = (20, -7, 0) #t_vector = (20, -7, 0)
r_vector = (15, 0 ,0) #r_vector = (15, 0 ,0)
TENSOR_TL = TRASLATE_TENSOR(TENSOR_TEST, t_vector)
TENSOR_FINAL = ROTATE_TENSOR(TENSOR_TL, r_vector)
```

Anexo III: Algoritmo de creación de máscaras.

```
def get_hipothalamus_mask(img_atlas, inf, sup):
    #log.info('Task 1.1: Find binary mask of amygdala')
    mask = np.zeros(img_atlas.shape)
    TENSOR_MASK = ((inf < img_atlas[:,:,:]) & (img_atlas[:,:,:] < sup))

    for z in range(img_atlas.shape[2]):
        for y in range(img_atlas.shape[1]):
            for x in range(img_atlas.shape[0]):
                if (TENSOR_MASK[x,y,z] == True):
                    mask[x,y,z] = 255

    return mask
```

Anexo IV: Visualización superpuesta del hipotálamo.

```
hipothalamus_mask = get_hipothalamus_mask(TENSOR_ATLAS_RESHAPE, inf = 121, sup = 150)

plano_medio_sagital_mask = hipothalamus_mask[:, :, 110]
plano_medio_coronal_mask = hipothalamus_mask[:, hipothalamus_mask.shape[1]//2, :]
plano_medio_axial_mask = hipothalamus_mask[105, :, :]

plano_medio_sagital_FIN = TENSOR_FINAL[:, :, 110]
plano_medio_coronal_FIN = TENSOR_FINAL[:, TENSOR_FINAL.shape[1]//2, :]
plano_medio_axial_FIN = TENSOR_FINAL[110, :, :]

plano_medio_sagital_phantom = TENSOR_PHANTOM[:, :, 110]
plano_medio_coronal_phantom = TENSOR_PHANTOM[:, TENSOR_PHANTOM.shape[1]//2, :]
plano_medio_axial_phantom = TENSOR_PHANTOM[110, :, :]

plano_medio_sagital_atlas_2 = TENSOR_ATLAS_RESHAPE[:, :, 110]
plano_medio_coronal_atlas_2 = TENSOR_ATLAS_RESHAPE[:,
TENSOR_ATLAS_RESHAPE.shape[1]//2, :]
plano_medio_axial_atlas_2 = TENSOR_ATLAS_RESHAPE[110, :, :]

fig, axs = plt.subplots(2,4, figsize=(30,20))
axs[0,0].imshow(plano_medio_sagital_FIN * 0.40 + plano_medio_sagital_mask * 0.60)
axs[0,1].imshow(plano_medio_coronal_FIN * 0.40 + plano_medio_coronal_mask * 0.60)
axs[1,1].imshow(plano_medio_axial_FIN * 0.40 + plano_medio_axial_mask * 0.60)

axs[0,2].imshow(plano_medio_sagital_phantom * 0.40 + plano_medio_sagital_atlas_2 * 0.60)
axs[0,3].imshow(plano_medio_coronal_phantom * 0.40 + plano_medio_coronal_atlas_2 *
0.60)
axs[1,3].imshow(plano_medio_axial_phantom * 0.40 + plano_medio_axial_atlas_2 * 0.60)

axs[0,0].set_title("plano_sagital")
axs[0,1].set_title("plano_coronal")
axs[1,1].set_title("plano_axial");

axs[0,2].set_title("plano_sagital with alpha 0.60")
axs[0,3].set_title("plano_coronal with alpha 0.60")
axs[1,3].set_title("plano_axial with alpha 0.60");
```