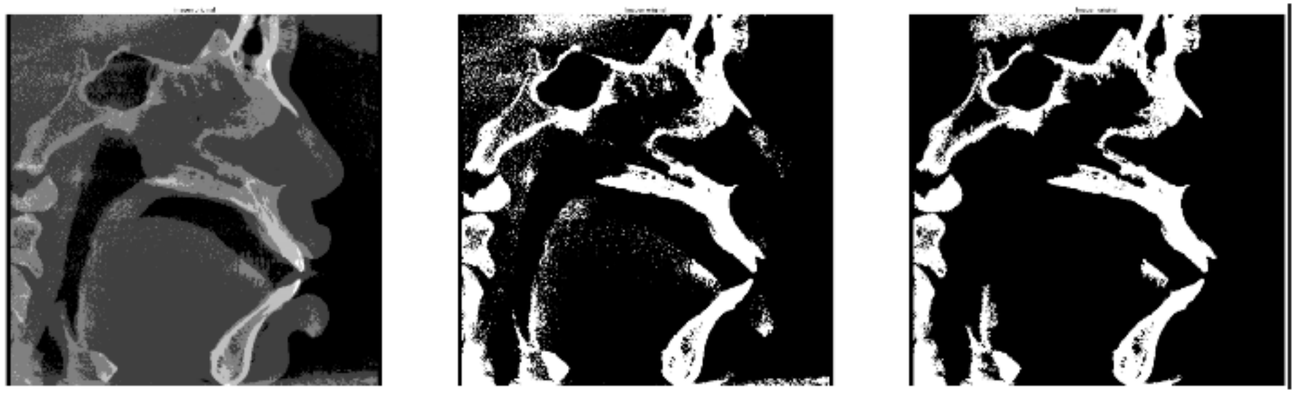
0.47

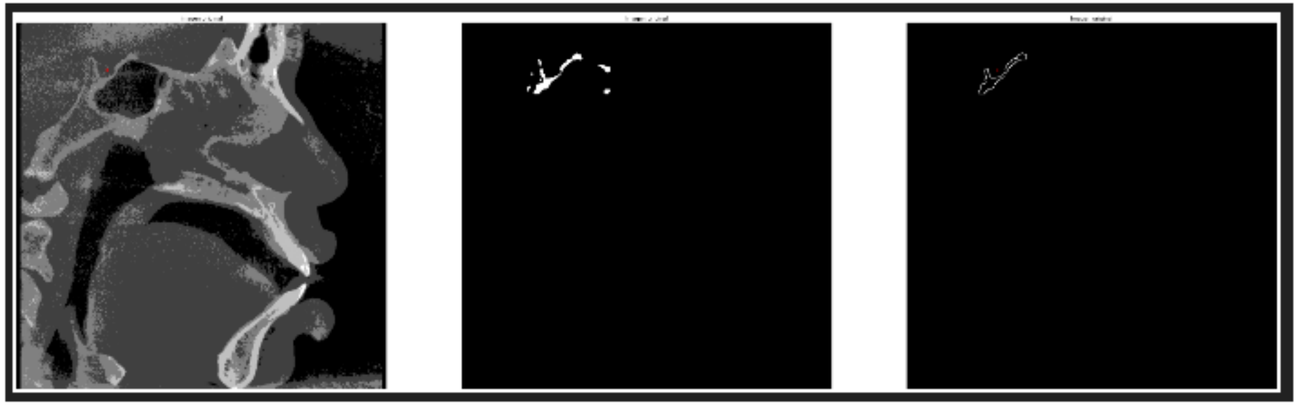
0.4



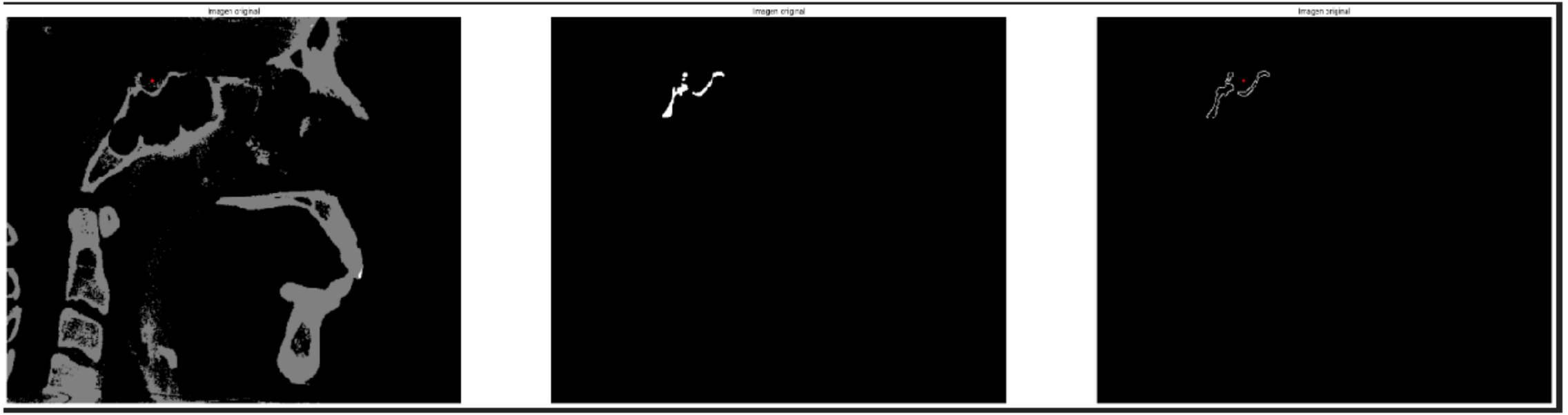
0.26

```
cleaned_img[c_ancho-20:, :] = 0
cleaned_img[:,c_largo-50:]=0
#bordes_S =  canny(vol_S)
cleaned_img[:40,:]=0
```

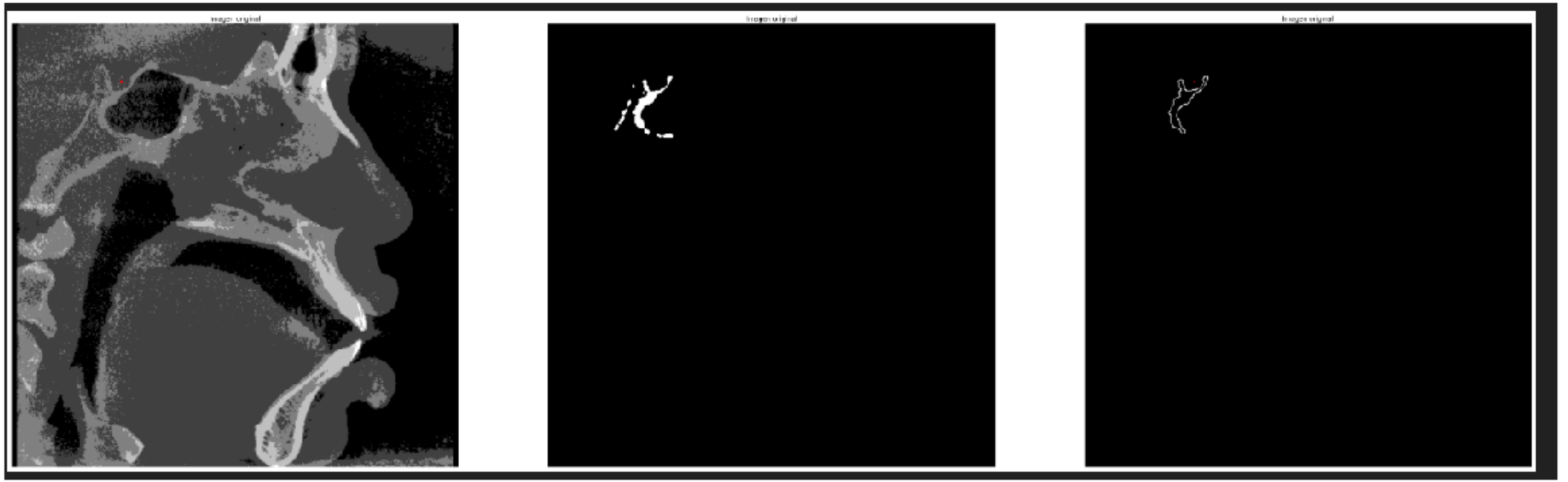
```
 cleaned_img[c_ancho-50:, :] = 0
 cleaned_img[:,c_largo-50:]=0
 #bordes_S =  canny(vol_S)
 cleaned_img[:40,:]=0
```



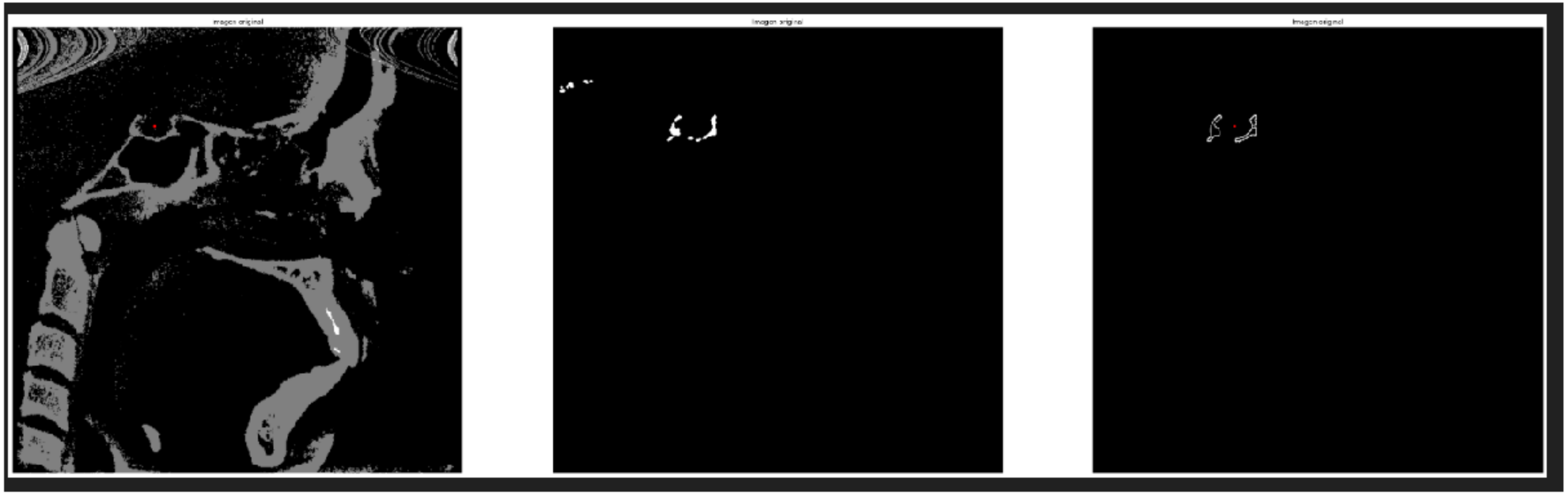
```
kernel_e =
cv.getStructuringElement(cv.MORPH
_RECT, (3, 3))
kernel_d =
cv.getStructuringElement(cv.MORPH
_ELLIPSE, (3, 3))
```




```
cleaned_img[c_ancho-39:, :] = 0
cleaned_img[:,c_largo-77:]=0
#bordes_S =  canny(vol_S)
cleaned_img[:60,:]=0
```



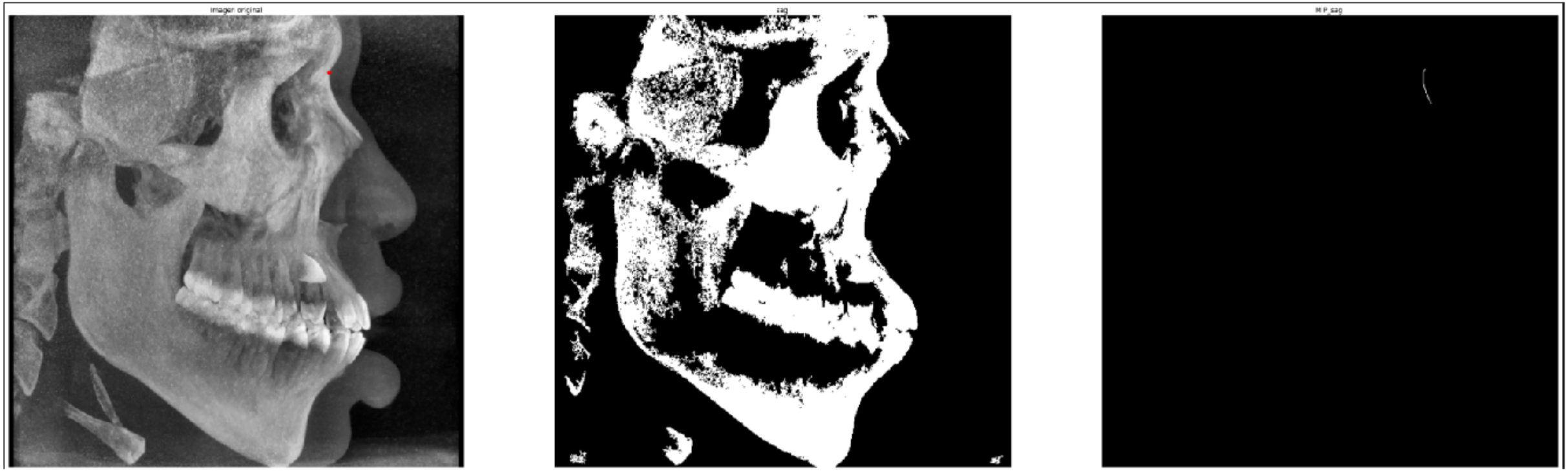
```
cleaned_img[c_ancho-39:, :] = 0
cleaned_img[:,c_largo-77:]=0
#bordes_S =  canny(vol_S)
cleaned_img[:60,:]=0
```




```
cleaned_img[c_ancho-39:, :] = 0
cleaned_img[:,c_largo-70:]=0
#bordes_S =  canny(vol_S)
cleaned_img[:60,:]=0
```

```
label_img_s = measure.label(bordes_S)
cleaned_img_s = morphology.remove_small_objects(label_img_s, min_size=100)
```

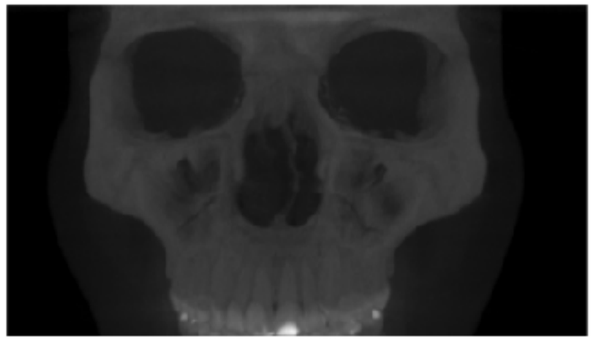

```
vol_Nasion[c_ancho-155:, :] = 0
vol_Nasion[:, :c_largo*2+10] = 0
vol_Nasion[:50, :] = 0
```




```
vol_Nasion[c_ancho-155:, :] = 0
vol_Nasion[:, :c_largo*2+10] = 0
vol_Nasion[:60, :] = 0
```


```
vol_Nasion[c_ancho-147:, :] = 0
vol_Nasion[:, :c_largo*2+10] = 0
vol_Nasion[:50, :] = 0
```

Imagen original



sag

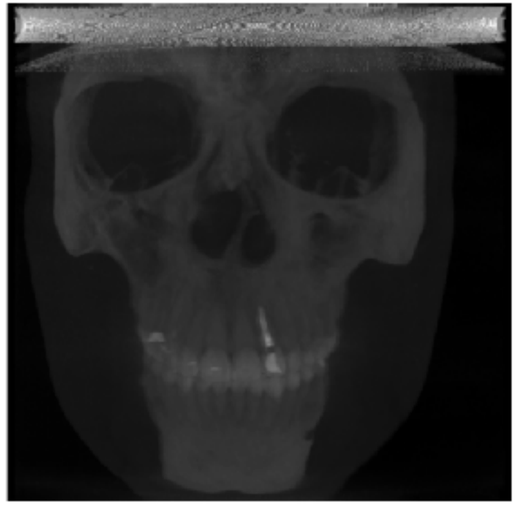
MIP_sag

Imagen original

sag

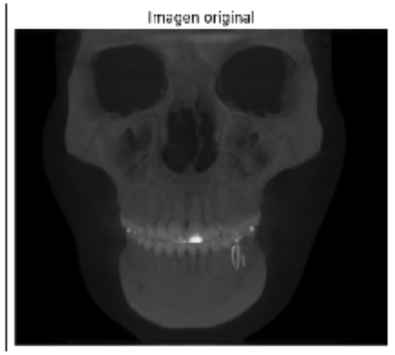
MIP_sag

Imagen original



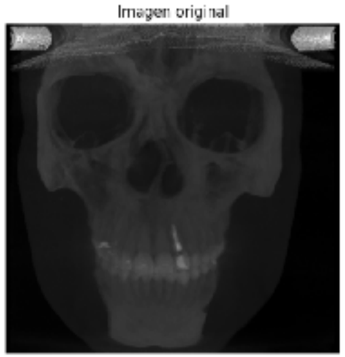
sag

MIP_sag

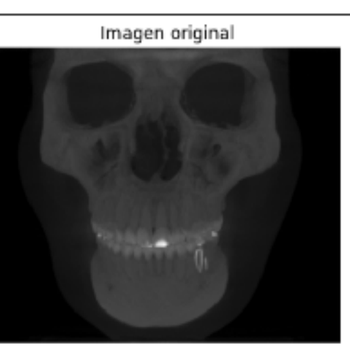


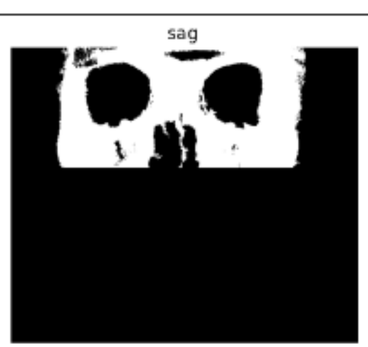

```
cuencas =slices_coronales[:,:, 100: cx//2-30]
mip_coronal_roll = np.rot90(np.max(cuencas,axis=2))
```

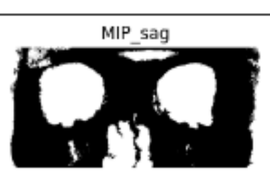

```
cuencas =slices_coronales[:,:, 100: cx//2-22]
mip_coronal_roll = np.rot90(np.max(cuencas,axis=2))
```

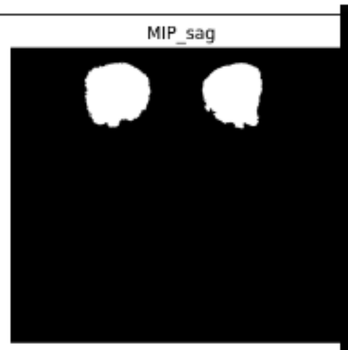


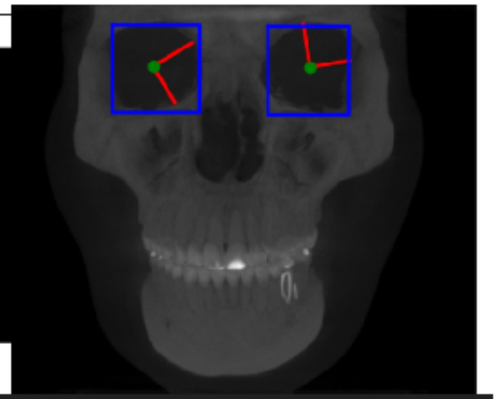

```
cuencas =slices_coronales[:,:, 100: cx//2-30]
mip_coronal_roll = np.rot90(np.max(cuencas,axis=2))
```

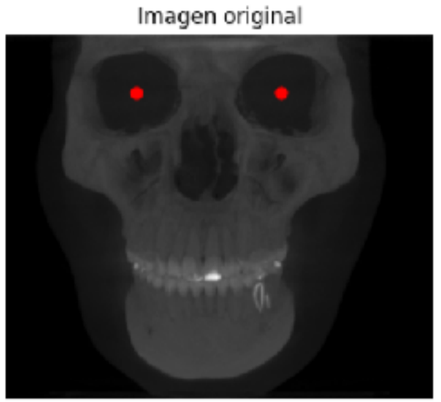




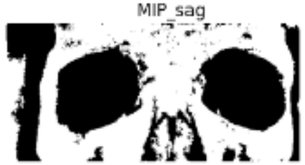


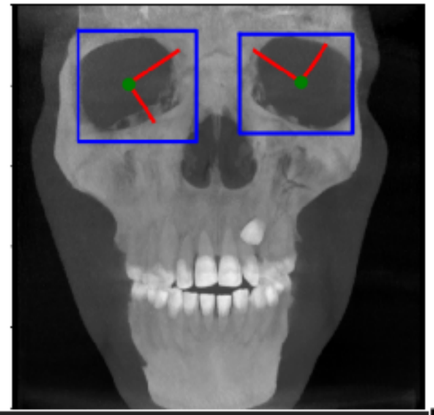


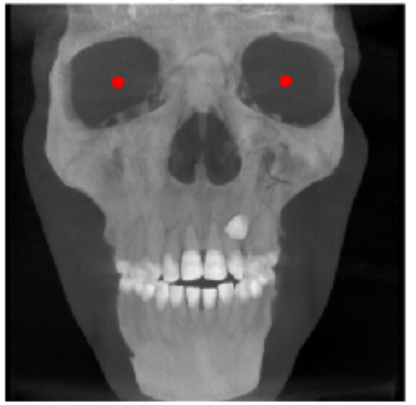




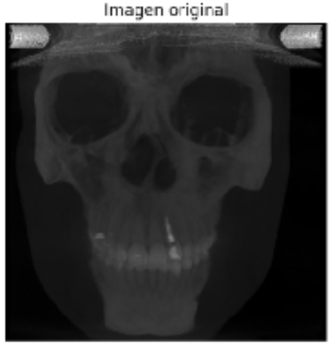
angulo de rotacion Roll: -0.25303081635718144



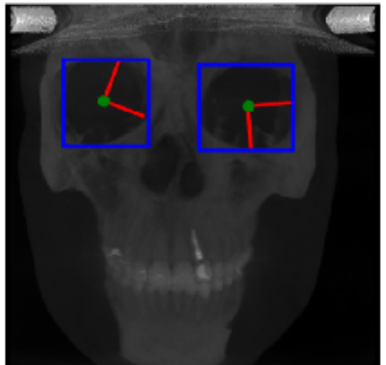


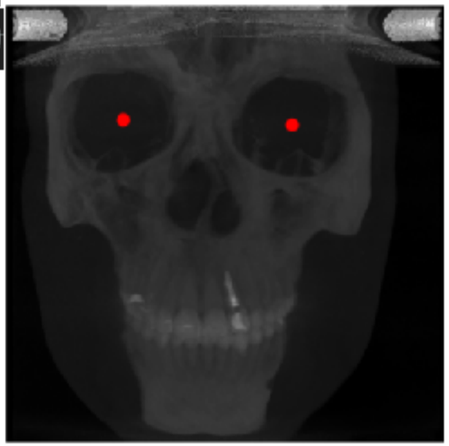


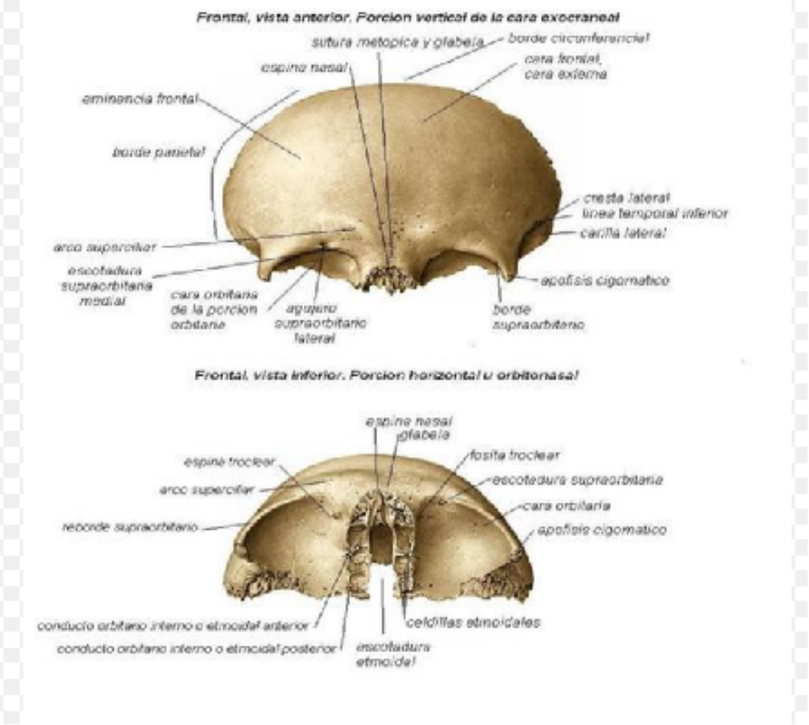
angulo de rotacion Roll: 0.4999

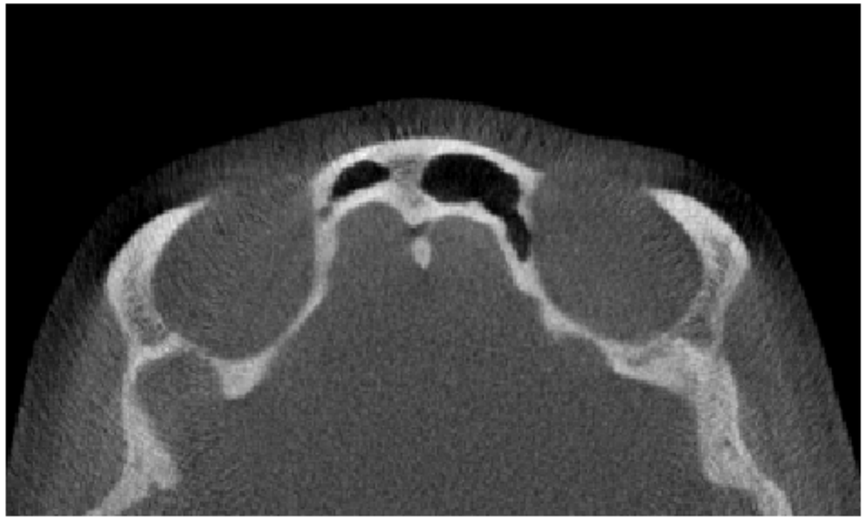


angulo de rotacion Roll: -1.90016157756505 69






Frontal, vista anterior. Porcion vertical de la cara exocraneal
sutura metopica y glabela
borde circunferencial
espina nasal
cara frontal, cara externa
eminencia frontal
borde parietal
cresta lateral
linea temporal inferior
cavilla lateral
arco superciliar
escotadura supraorbitaria medial
cara orbitaria de la porcion orbitaria
agujero supraorbitario lateral
apofisis cigomatico
borde supraorbitario
Frontal, vista inferior. Porcion horizontal u orbitonasal
espina nasal
glabela
espina troclear
fosita troclear
arco superciliar
escotadura supraorbitaria
cara orbitaria
reborde supraorbitario
apofisis cigomatico
conducto orbitario interno o etmoidal anterior
conducto orbitario interno o etmoidal posterior
celdillas etmoidales
escotadura etmoidal




1: Unnamed Series 1

Imagen original afz

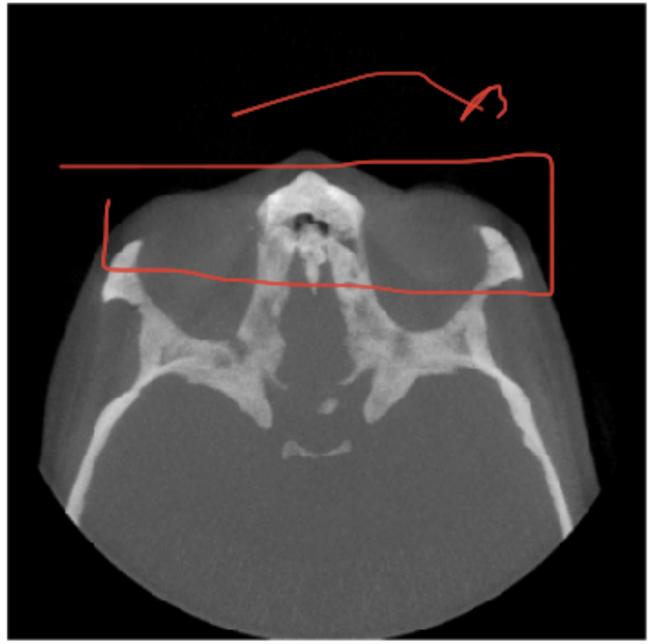
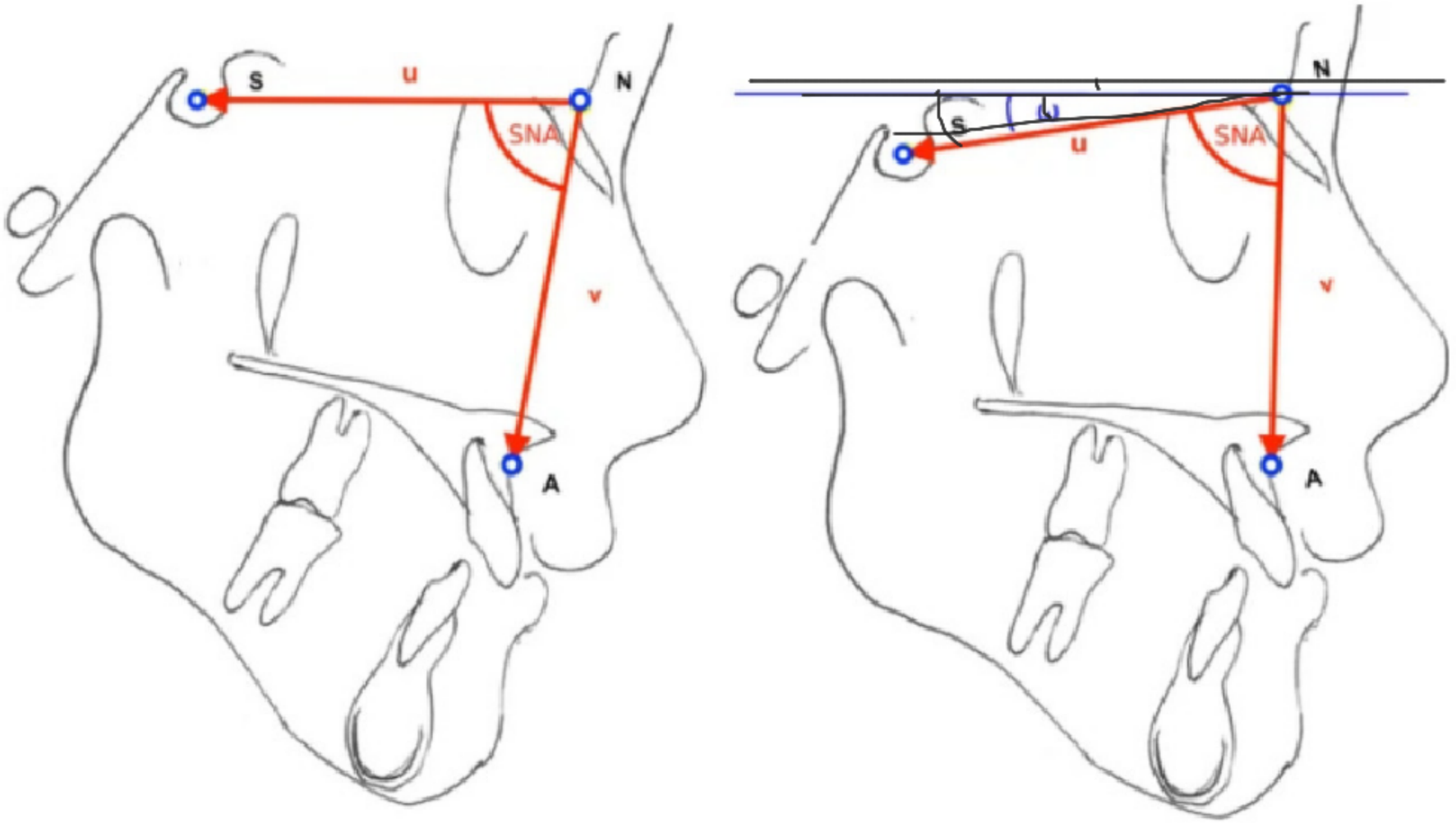


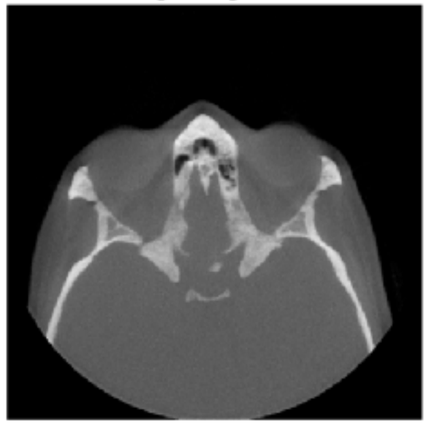
Imagen original afz

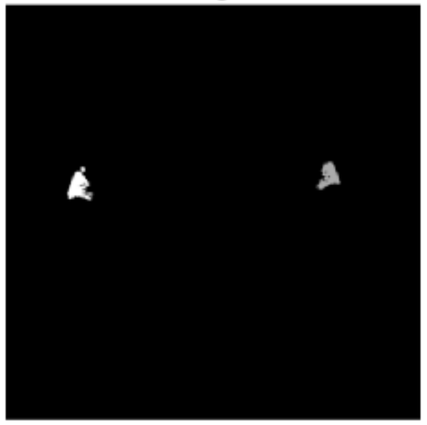
```
= volumen_org[:,:, cy-80:cy-48]
-np.fliplr(np.rot90(np.rot90(np.rot90(np.max(MIP_axial,axis=2)))))
```

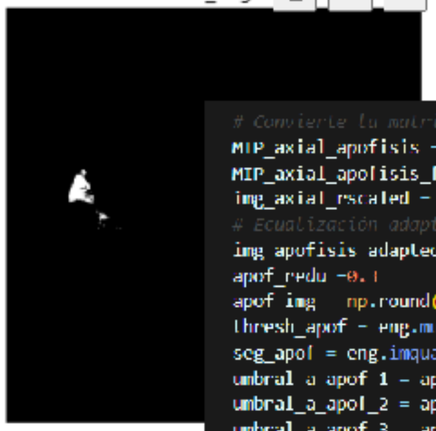
Fig. 1



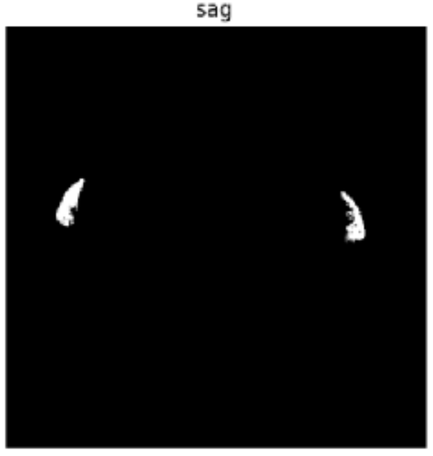
Sketch of the reference points S, N and A, the SNA-angle and the horizontal angle ω. For illustration purposes, here SN in the left image is ideally orientated parallel to the horizontal plane, hence the ω = 0° in this case

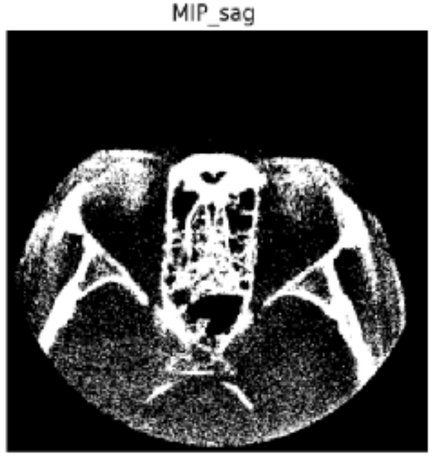


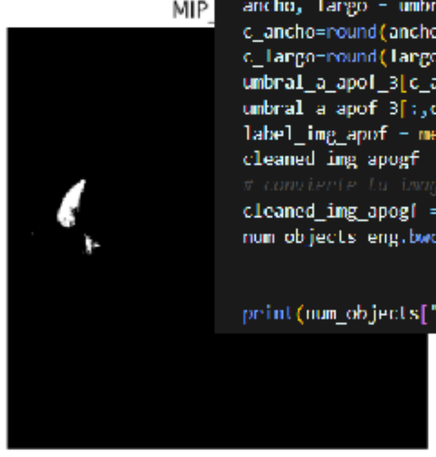




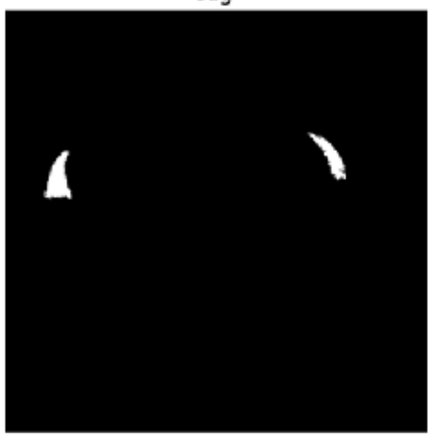

```
# Convierte tu matriz en C-contiguous
MIP_axial_apofisis = np.ascontiguousarray(MIP_axial).astype('f
MIP_axial_apofisis_l = denoise_bilateral(MIP_axial_apofisis, s
img_axial_rscaled = exposure.rescale_intensity(MIP_axial_apofi
# Ecualización adaptativa
img apofisis adapteq   exposure.equalize_adapthist(img_axial_r
apof_redu =0.1
apof_img   np.round(img_apofisis_adapteq/apof_redu)*apof_redu
thresh_apof = mg.multithresh(apof_img,3)
seg_apof = cng.imquantize(apof_img,thresh_apof)
umbral_a_apof_1 = apof_img > list(thresh_apof)[0][0]
umbral_a_apof_2 = apof_img > list(thresh_apof)[0][1]
umbral_a_apof_3   apof_img > list(thresh_apof)[0][2]
ancho, largo = umbral_a_apof_1.shape
c_ancho=round(ancho/2)
c_largo=round(largo/3)
umbral_a_apof_3[c_ancho - 20:, :] = 0
umbral_a_apof_3[:,c_largo:c_largo*2]   0
label_img_apof = measure.label(umbral_a_apof_3)
cleaned_img_apogf   morphology.remove_small_objects(label_img_a
# convierte tu imagen etiquetada en una imagen binaria nuevamen
cleaned_img_apogf = cleaned_img_apogf > 0
num_objects= cng.bwconncomp(cleaned_img_apogf)


print(num_objects["NumObjects"])
```







0.3



0.26

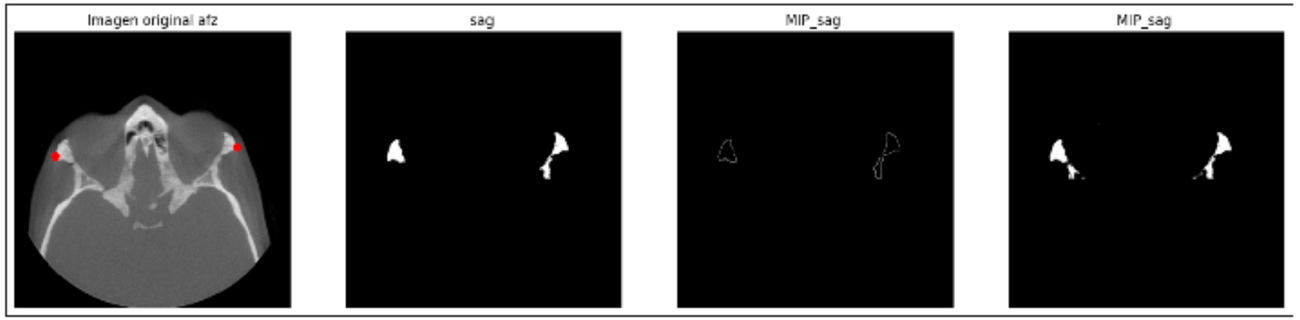


Imagen original afz
sag
MIP_sag
MIP_sag

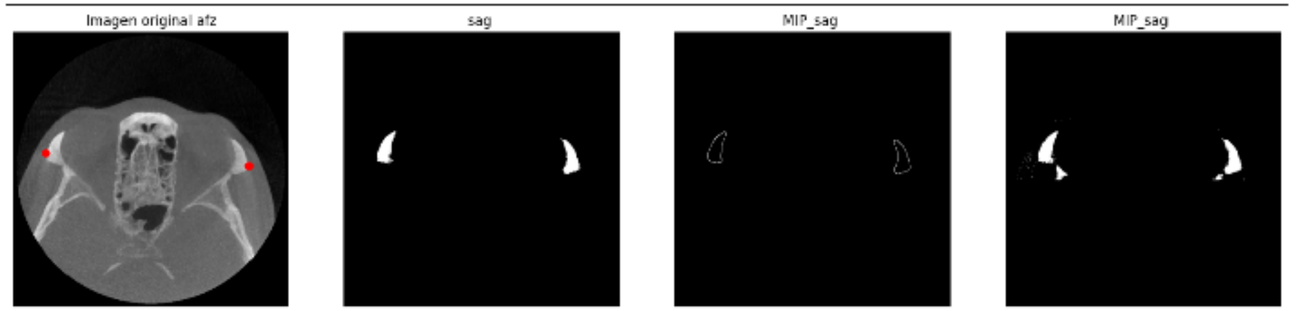


Imagen original afz
sag
MIP_sag
MIP_sag

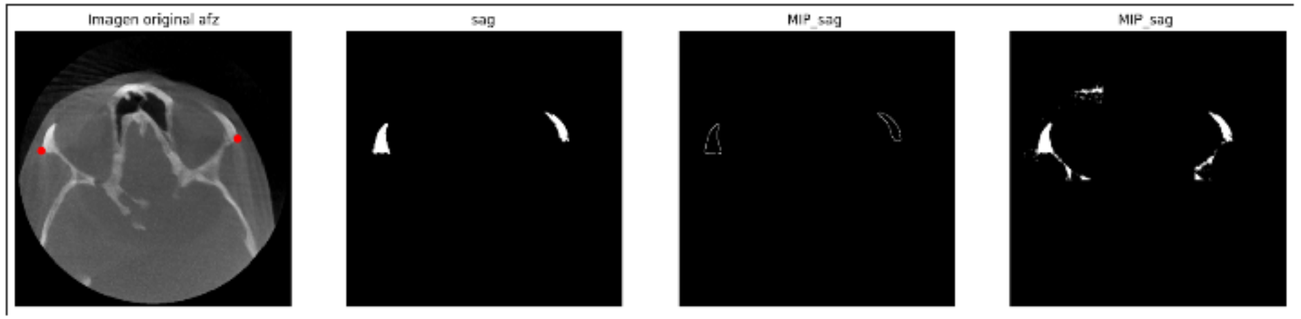


Imagen original afz
sag
MIP_sag
MIP_sag