

004 - Proyecto de Grado- Calculo de Anormalidad o Normalidad

January 20, 2025

1 Proyecto de Grado Analisis de Estructura Creaneal Cavum Septum Peducillum

- Estudiante: Milton Fabian Cifuentes Ortega

2 4.1. Limpieza de Memoria y Carga de Librerías

Primero realizamos la limpieza de memorar y carga de librerías

```
[31]: #pip install opencv-python
      #pip install pandas
      #pip install numpy
      #pip install matplotlib
      #pip install openpyxl
      #pip install plotly
      #pip install nbformat --upgrade
```

```
[32]: import gc
      gc.collect()
```

[32]: 4532

3 4.2. Carga De Librerías

Procedemos a la carga de las librerías que se necesitan para el funcionamiento del proceso de cálculo del área de las máscaras predichas

```
[33]: import os
      import cv2
      import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      import plotly.express as px
```

3.1 4.4. Carga de Imágenes, Mascaras y Mascaras Predichas para Análisis

Como primera tarea realizaremos la carga de las imágenes, Mascaras y Mascaras predichas, esto nos permitirá realizar la manipulación y poder calcular el área de la estructura.

```

[34]: # Definir rutas principales
base_path = r'D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de_
↳Grado\003 - Trabajo de grado\Resultados Modelos Unet\ImagenesPredichas'
carpetas = ['Trans-Cerebellum', 'Trans-Thalamic', 'Trans-Ventricular']

# Función para cargar imágenes de una carpeta con un sufijo específico
def cargar_imagenes_con_sufijo(ruta, sufijo):
    imagenes = {}
    for nombre_archivo in os.listdir(ruta):
        if nombre_archivo.endswith('.png') and sufijo in nombre_archivo:
            img_path = os.path.join(ruta, nombre_archivo)
            imagen = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
            if imagen is not None:
                # Extraer la parte base del nombre del archivo antes del sufijo
                base_nombre = nombre_archivo.replace(sufijo + '.png', '')
                imagenes[base_nombre] = imagen # Cargar la imagen en lugar de_
↳solo la ruta
            else:
                print(f"Advertencia: no se pudo cargar la imagen {img_path}")
    return imagenes

# Listas para almacenar datos
datos = []

# Recorrer cada carpeta (Trans-Cerebellum, Trans-Thalamic, etc.)
for carpeta in carpetas:
    # Definir las rutas para imagenes, mascaras, mascaras_predichas
    ruta_imagenes = os.path.join(base_path, carpeta, 'imagenes')
    ruta_mascaras = os.path.join(base_path, carpeta, 'mascaras')
    ruta_mascaras_predichas = os.path.join(base_path, carpeta,
↳'mascaras_predichas')

    # Cargar las imágenes, máscaras y máscaras predichas utilizando la parte_
↳base del nombre
    imagenes = cargar_imagenes_con_sufijo(ruta_imagenes, '_imagen')
    mascarar = cargar_imagenes_con_sufijo(ruta_mascaras, '_mascara')
    mascarar_predichas = cargar_imagenes_con_sufijo(ruta_mascaras_predichas,
↳'_mascara_predicha')

    # Procesar cada imagen base
    for base_nombre in imagenes.keys():
        # Cargar imagen, máscara y máscara predicha
        imagen = imagenes.get(base_nombre)
        mascara = mascarar.get(base_nombre)
        mascara_predicha = mascarar_predichas.get(base_nombre)

        if imagen is None:

```

```

        print(f"Advertencia: imagen no encontrada para {base_nombre}")
    if mascara is None:
        print(f"Advertencia: máscara no encontrada para {base_nombre}")
    if mascara_predicha is None:
        print(f"Advertencia: máscara predicha no encontrada para_
↪{base_nombre}")

    # Almacenar datos en una lista
    datos.append({
        'Carpeta': carpeta,
        'Base_Nombre': base_nombre,
        'Imagen': imagen,
        'Mascara': mascara,
        'Mascara_Predicha': mascara_predicha
    })

# Crear un DataFrame con los resultados
df_resultados2 = pd.DataFrame(datos)

# Mostrar el DataFrame cargado
df_resultados2

```

```

[34]:
      Carpeta      Base_Nombre \
0  Trans-Cerebellum Patient00644_Plane3_2_of_3_5
1  Trans-Cerebellum Patient00662_Plane3_1_of_1_2
2  Trans-Cerebellum Patient00662_Plane3_1_of_1
3  Trans-Cerebellum Patient00675_Plane3_1_of_1_2
4  Trans-Cerebellum Patient00675_Plane3_1_of_1_3
5  Trans-Cerebellum Patient00687_Plane3_1_of_1_8
6  Trans-Cerebellum Patient00706_Plane3_5_of_5_2
7  Trans-Cerebellum Patient00706_Plane3_5_of_5_4
8  Trans-Cerebellum Patient00706_Plane3_5_of_5_6
9  Trans-Cerebellum Patient00706_Plane3_5_of_5_8
10 Trans-Thalamic Patient00168_Plane3_2_of_3
11 Trans-Thalamic Patient00216_Plane3_1_of_5_3
12 Trans-Thalamic Patient00216_Plane3_1_of_5_5
13 Trans-Thalamic Patient00216_Plane3_1_of_5_6
14 Trans-Thalamic Patient00305_Plane3_5_of_5_2
15 Trans-Thalamic Patient00305_Plane3_5_of_5_3
16 Trans-Thalamic Patient00305_Plane3_5_of_5
17 Trans-Thalamic Patient00640_Plane3_1_of_1
18 Trans-Thalamic Patient00647_Plane3_1_of_1
19 Trans-Thalamic Patient00658_Plane3_1_of_1
20 Trans-Ventricular Patient00188_Plane3_2_of_3_6
21 Trans-Ventricular Patient00216_Plane3_2_of_5_2
22 Trans-Ventricular Patient00216_Plane3_2_of_5
23 Trans-Ventricular Patient00305_Plane3_1_of_5_2

```

24 Trans-Ventricular Patient00305_Plane3_1_of_5_4
 25 Trans-Ventricular Patient00644_Plane3_1_of_3_2
 26 Trans-Ventricular Patient00644_Plane3_1_of_3_5
 27 Trans-Ventricular Patient00644_Plane3_1_of_3_8
 28 Trans-Ventricular Patient00688_Plane3_1_of_1_7
 29 Trans-Ventricular Patient00700_Plane3_2_of_2_3

Imagen \

0 [[0, 7, 111, 118, 101, 149, 107, 112, 52, 64, ...
 1 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 2 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 3 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 4 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 11, 0, 0...
 5 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 6 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 7 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 8 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 9 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 10 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 11 [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
 12 [[0, 0, 0, 0, 0, 0, 24, 18, 17, 14, 0, 0, 0, 0, 0, ...
 13 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15, 8...
 14 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 15 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 16 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 17 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 18 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 19 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 20 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 21 [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
 22 [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
 23 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 24 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 25 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 26 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12...
 27 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 28 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 29 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...

Mascara \

0 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 1 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 2 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 3 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 4 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 5 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
 6 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...


```

22 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
23 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
24 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
25 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
26 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
27 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
28 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
29 [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...

```

Como podemos ver realizamos la carga de las imágenes, máscaras y máscaras predichas al dataframe y mostramos las 30 imágenes creadas

```

[35]: # Itera sobre cada fila del DataFrame
for index, row in df_resultados2.iterrows():
    # Cargar las imágenes desde el DataFrame
    imagen = row['Imagen']
    mascara = row['Mascara']
    mascara_predicha = row['Mascara_Predicha']

    # ojo las imagenes deben ser del mismo tamaño
    if imagen.shape != mascara.shape or imagen.shape != mascara_predicha.shape:
        raise ValueError("Las imágenes, máscara y máscara predicha deben tener
        el mismo tamaño.")

    # Convertir las imágenes a formato RGB si están en escala de grises
    if len(imagen.shape) == 2: # Si es una imagen de un solo canal
        imagen = cv2.cvtColor(imagen, cv2.COLOR_GRAY2BGR)

    # Crear una imagen de color para las máscaras (rojo para la máscara
    original y azul para la máscara predicha)
    mascara_color = np.zeros_like(imagen)
    mascara_color[mascara == 255] = [0, 0, 255] # Rojo para la máscara original

    mascara_predicha_color = np.zeros_like(imagen)
    mascara_predicha_color[mascara_predicha == 255] = [255, 0, 0] # Azul para
    la máscara predicha

    # Combinar las imágenes originales con las máscaras utilizando una opacidad
    más baja
    resultado_mas_mascara = cv2.addWeighted(imagen, 0.8, mascara_color, 0.2, 0)
    # Máscara original en rojo
    resultado_mas_mascara_predicha = cv2.addWeighted(imagen, 0.8,
    mascara_predicha_color, 0.2, 0) # Máscara predicha en azul

    # Concatenar las imágenes superpuestas
    resultado_final = np.hstack((resultado_mas_mascara,
    resultado_mas_mascara_predicha))

```

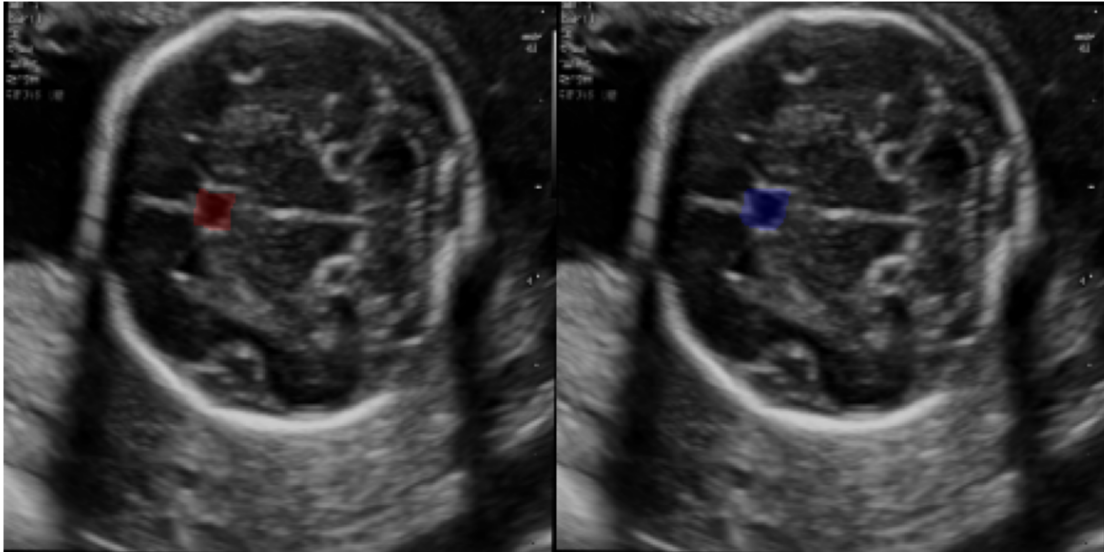
```

# Unir el título con Carpeta y Base_Nombre
carpeta = row['Carpeta']
base_nombre = row['Base_Nombre']
titulo = f'Carpeta: {carpeta}, Base Nombre: {base_nombre} - Index {index}'

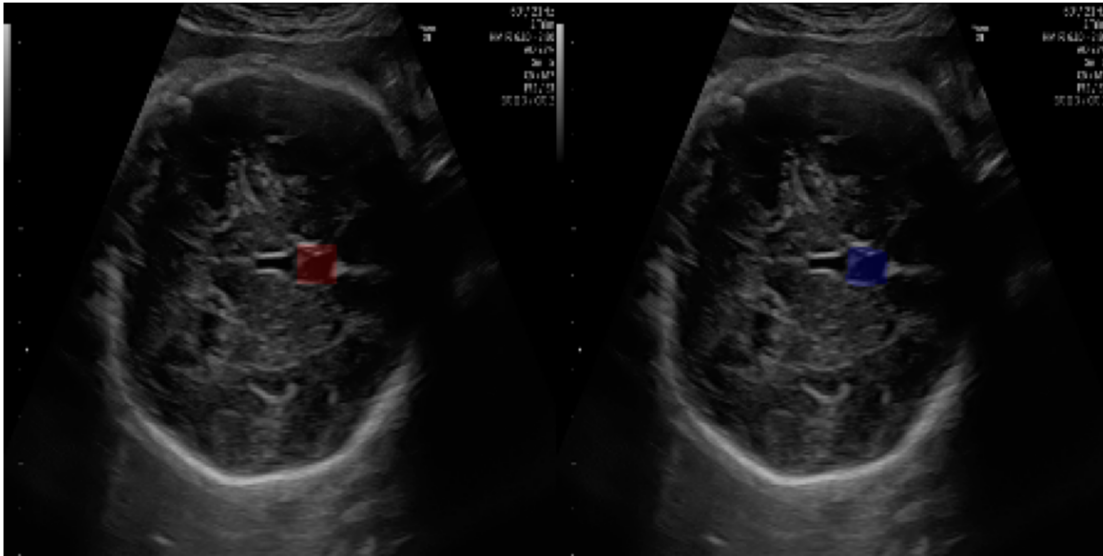
# Mostrar la imagen en Visual Studio Code
plt.figure(figsize=(15, 10))
plt.imshow(cv2.cvtColor(resultado_final, cv2.COLOR_BGR2RGB)) # Convertir a
→RGB para la visualización
plt.axis('off') # Oculta los ejes
plt.title(titulo)
plt.show()

```

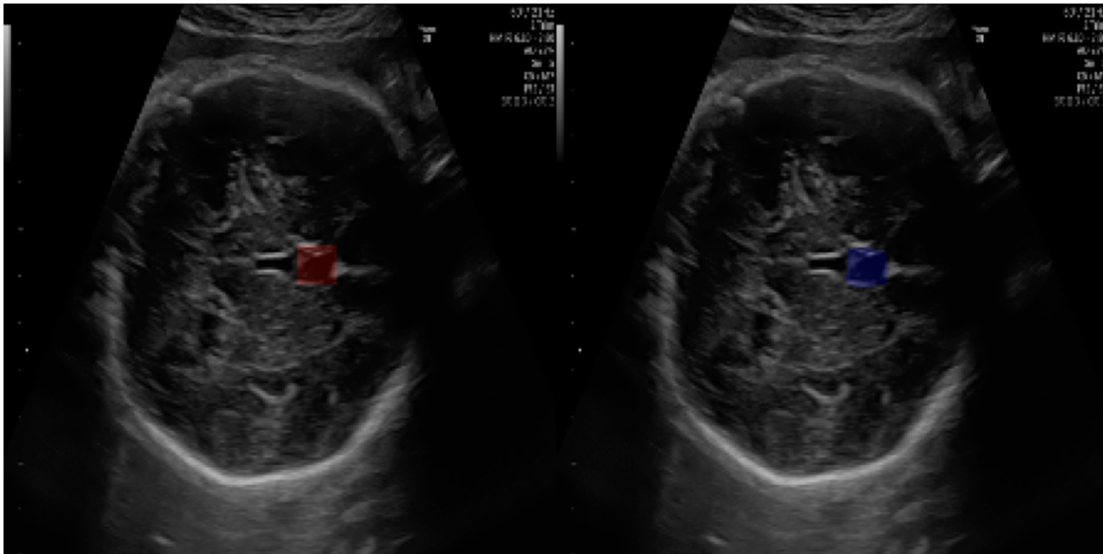
Carpeta: Trans-Cerebellum, Base Nombre: Patient00644_Plane3_2_of_3_5 - Index 0



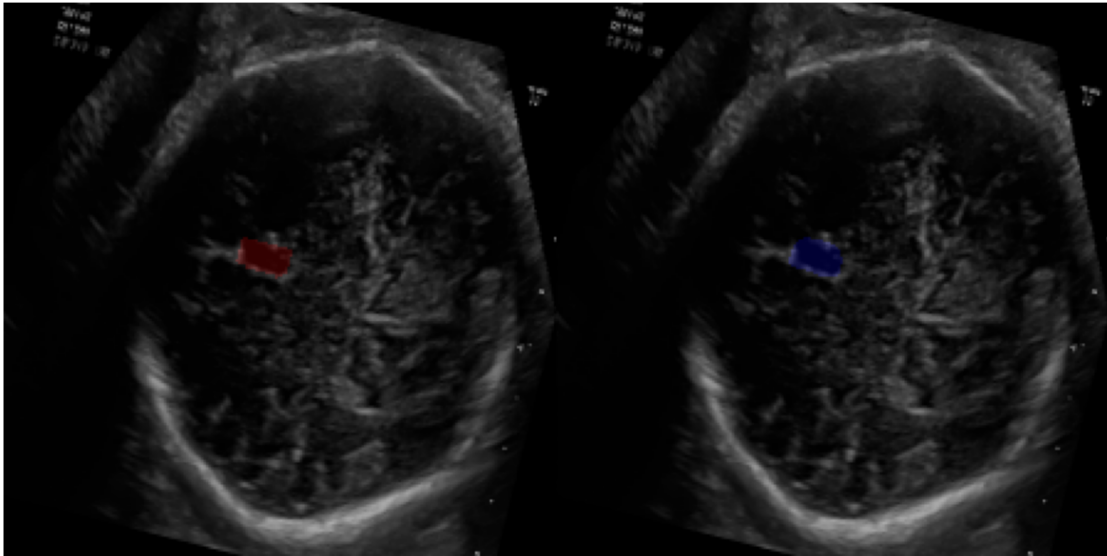
Carpeta: Trans-Cerebellum, Base Nombre: Patient00662_Plane3_1_of_1_2 - Index 1



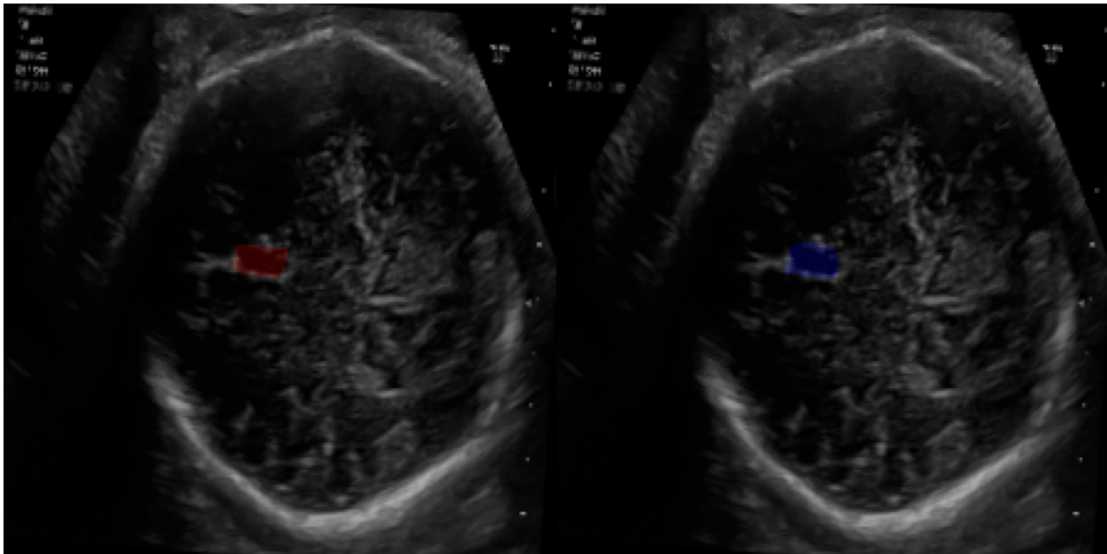
Carpeta: Trans-Cerebellum, Base Nombre: Patient00662_Plane3_1_of_1_1 - Index 2



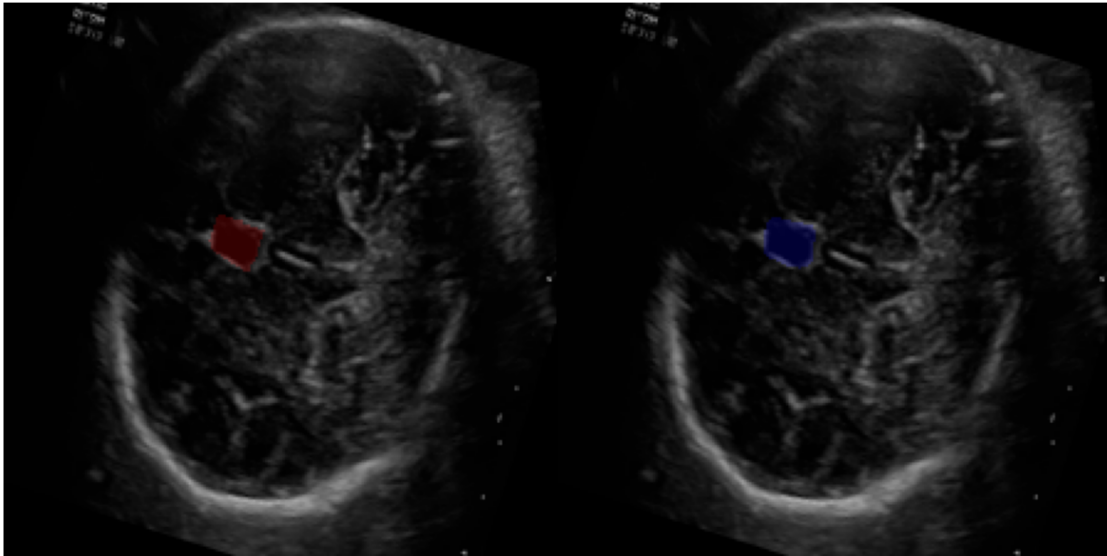
Carpeta: Trans-Cerebellum, Base Nombre: Patient00675_Plane3_1_of_1_2 - Index 3



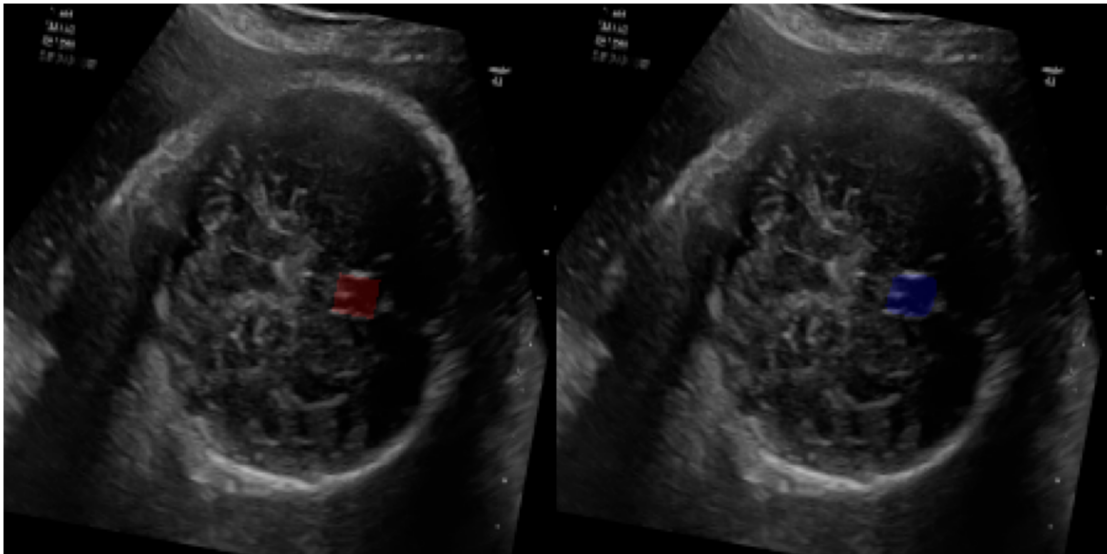
Carpeta: Trans-Cerebellum, Base Nombre: Patient00675_Plane3_1_of_1_3 - Index 4



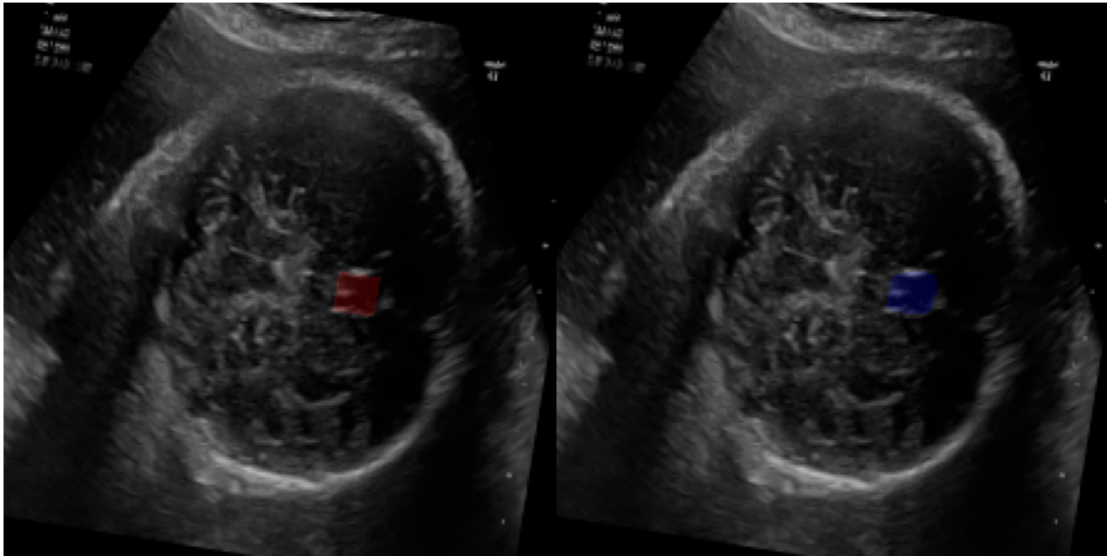
Carpeta: Trans-Cerebellum, Base Nombre: Patient00687_Plane3_1_of_1_8 - Index 5



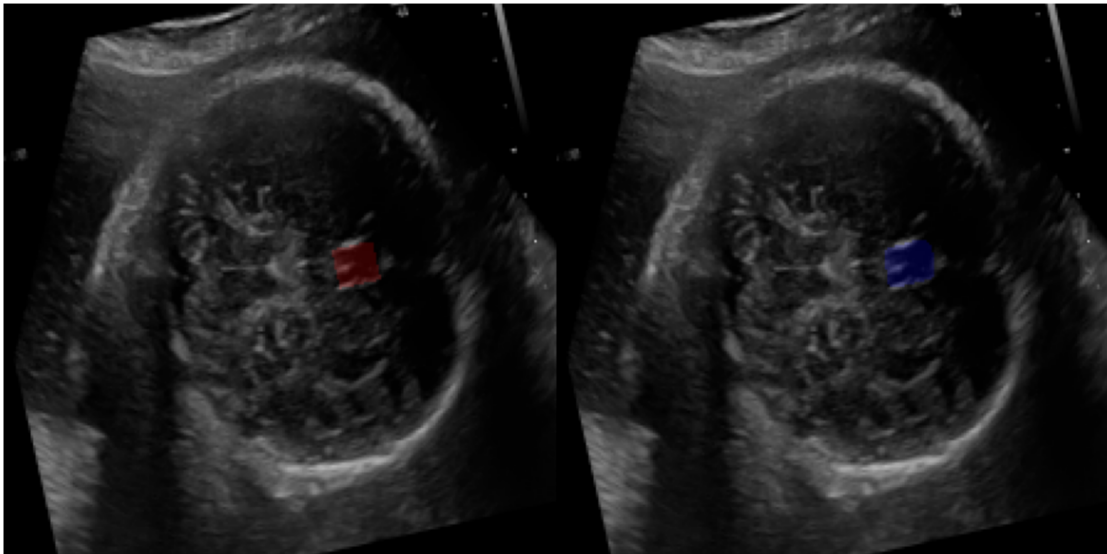
Carpeta: Trans-Cerebellum, Base Nombre: Patient00706_Plane3_5_of_5_2 - Index 6



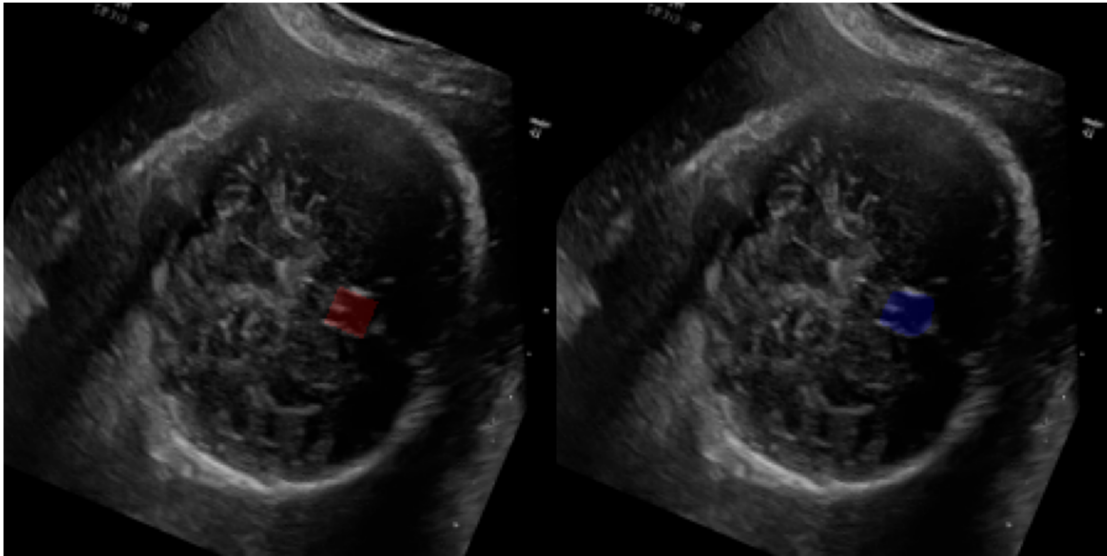
Carpeta: Trans-Cerebellum, Base Nombre: Patient00706_Plane3_5_of_5_4 - Index 7



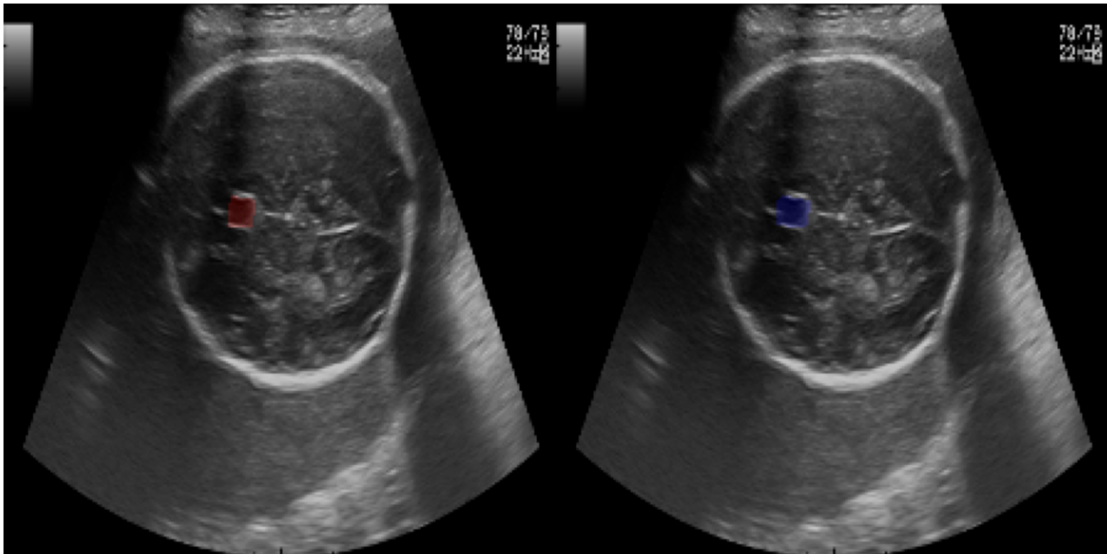
Carpeta: Trans-Cerebellum, Base Nombre: Patient00706_Plane3_5_of_5_6 - Index 8



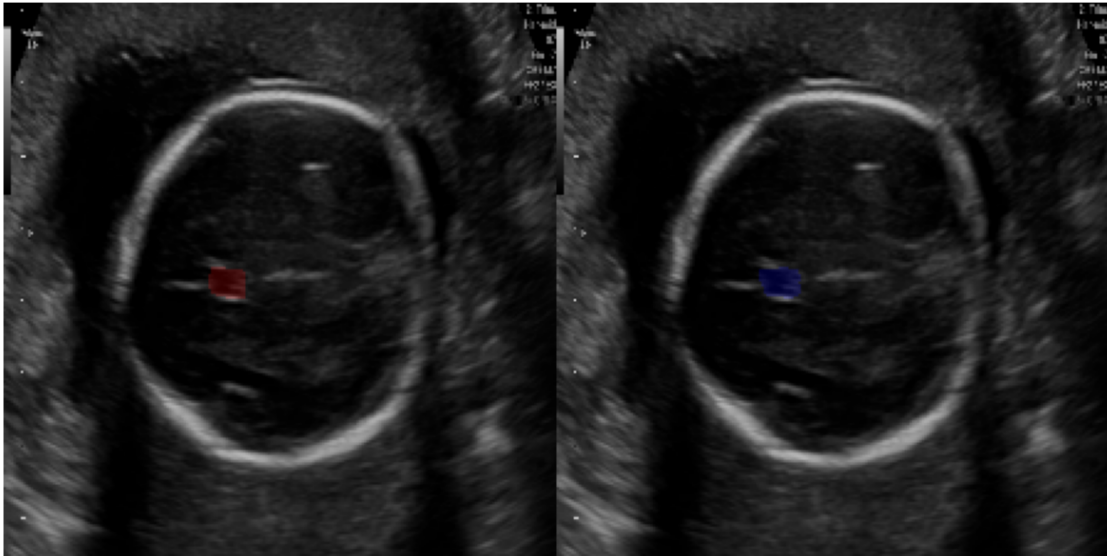
Carpeta: Trans-Cerebellum, Base Nombre: Patient00706_Plane3_5_of_5_8 - Index 9



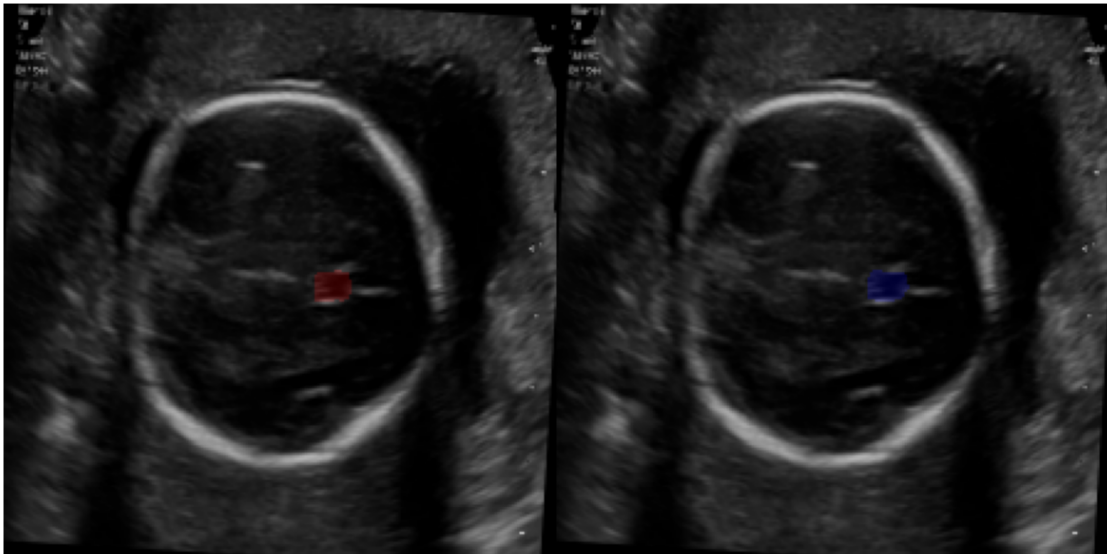
Carpeta: Trans-Thalamic, Base Nombre: Patient00168_Plane3_2_of_3 - Index 10



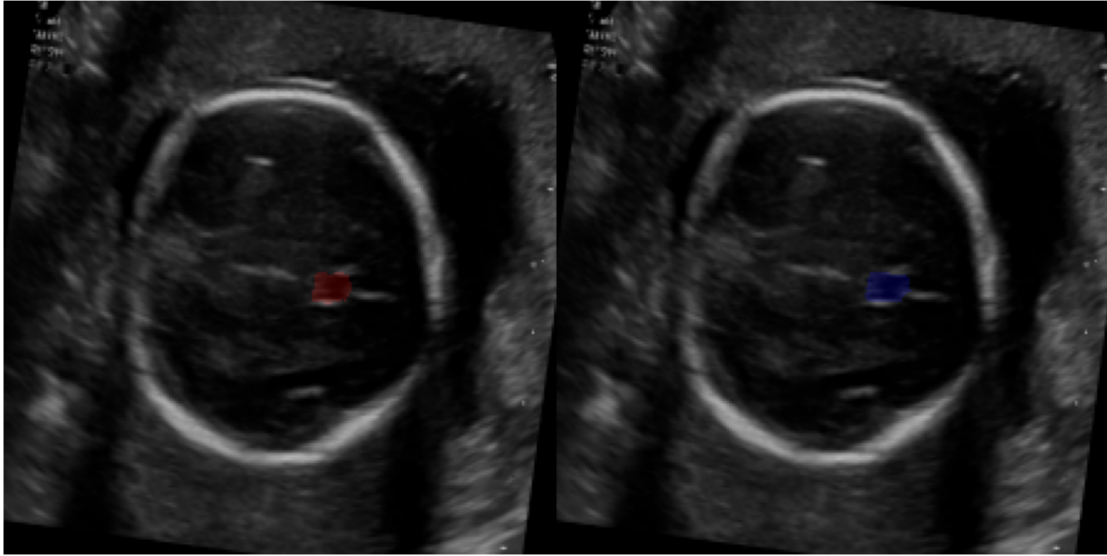
Carpeta: Trans-Thalamic, Base Nombre: Patient00216_Plane3_1_of_5_3 - Index 11



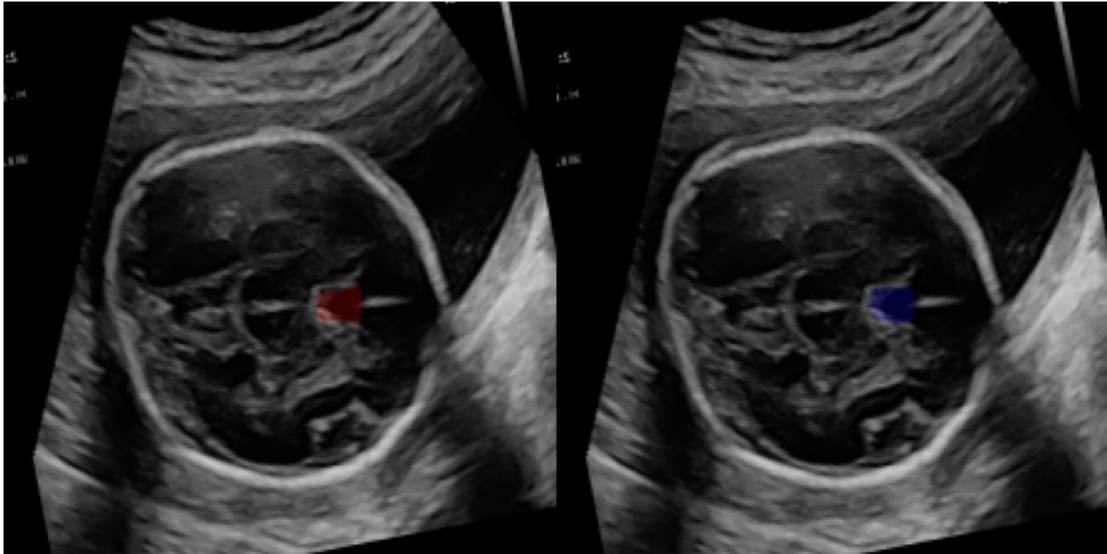
Carpeta: Trans-Thalamic, Base Nombre: Patient00216_Plane3_1_of_5_5 - Index 12

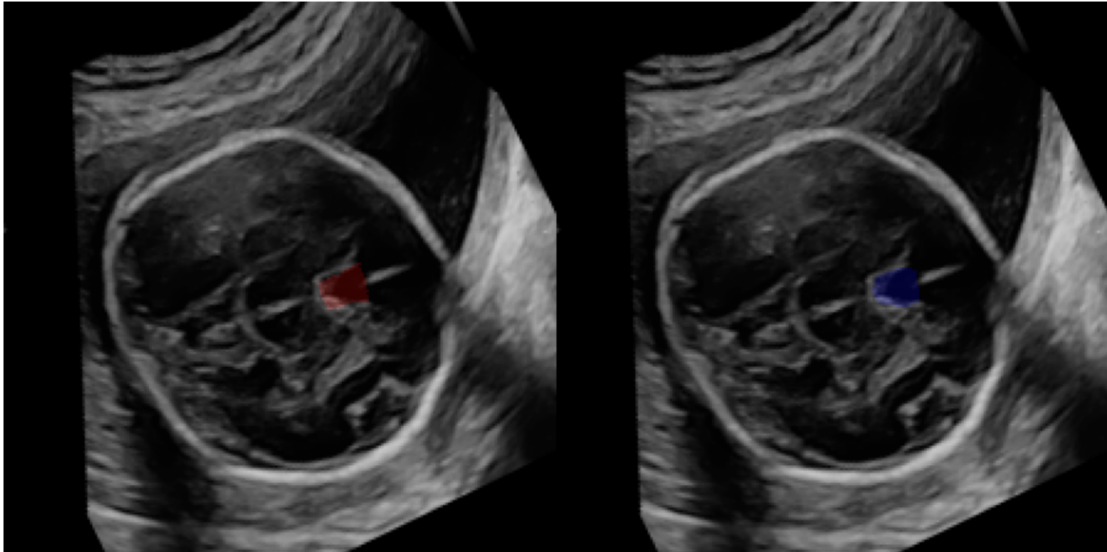
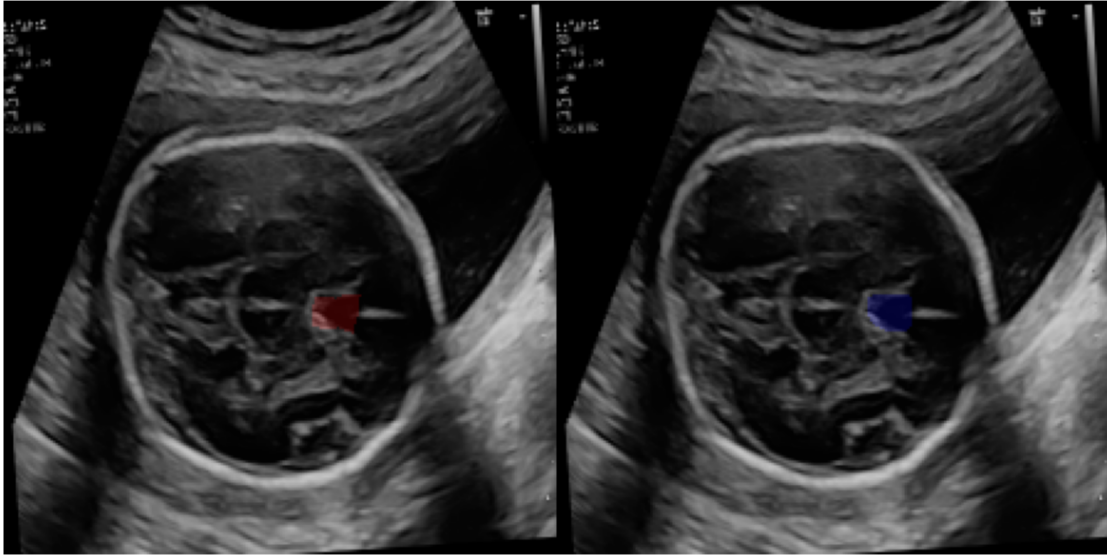


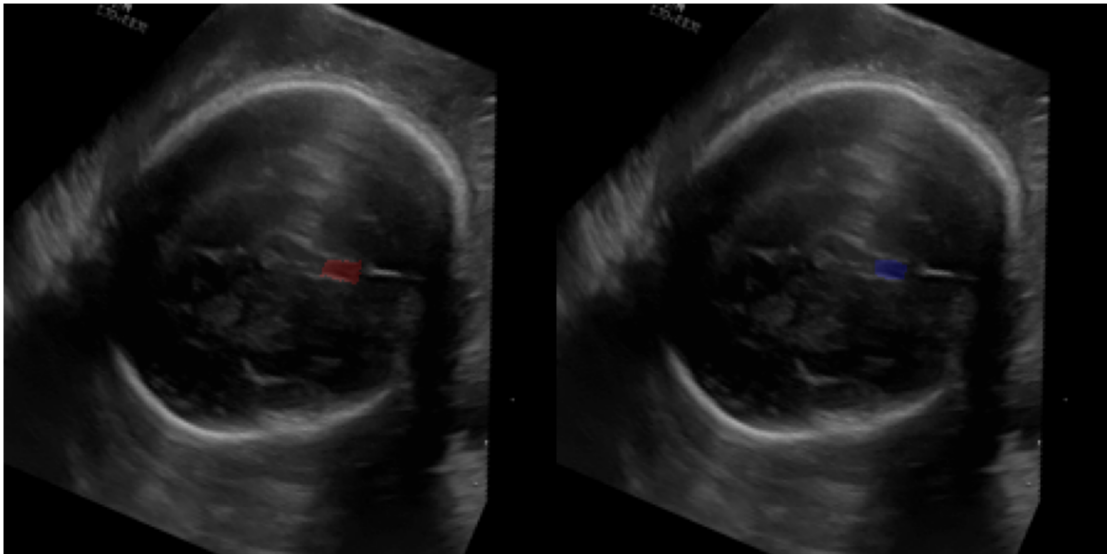
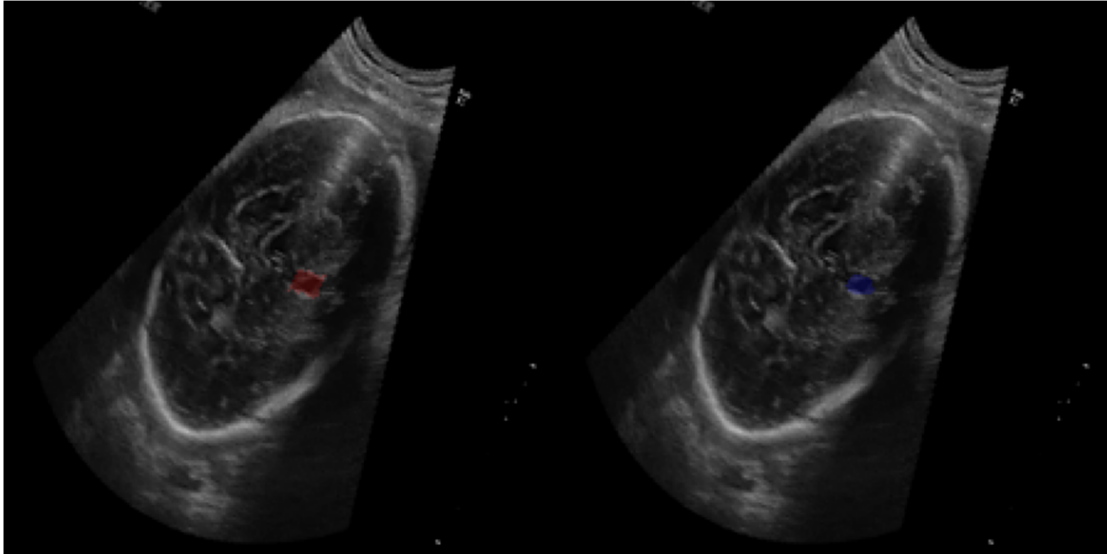
Carpeta: Trans-Thalamic, Base Nombre: Patient00216_Plane3_1_of_5_6 - Index 13



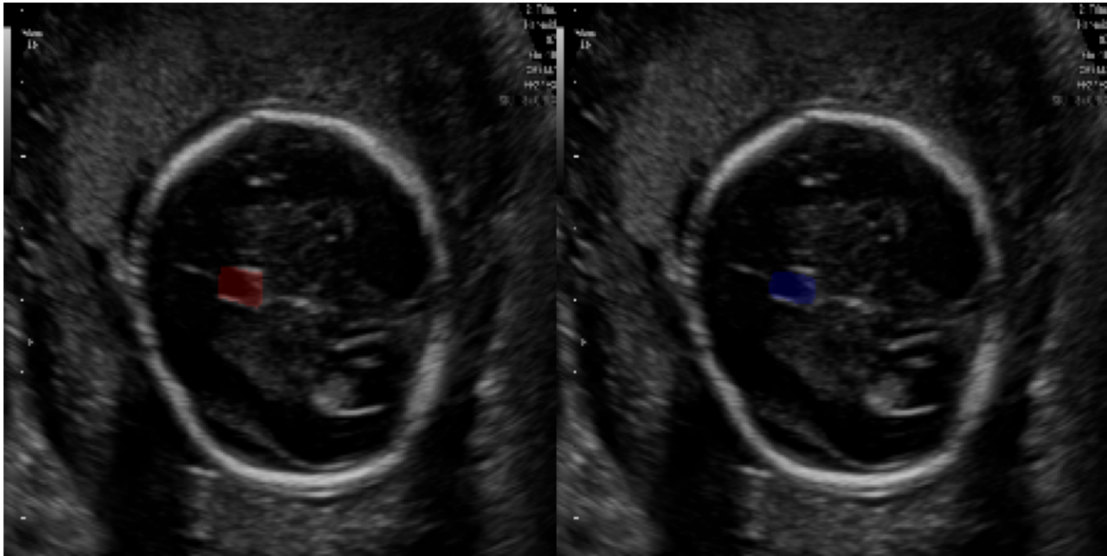
Carpeta: Trans-Thalamic, Base Nombre: Patient00305_Plane3_5_of_5_2 - Index 14



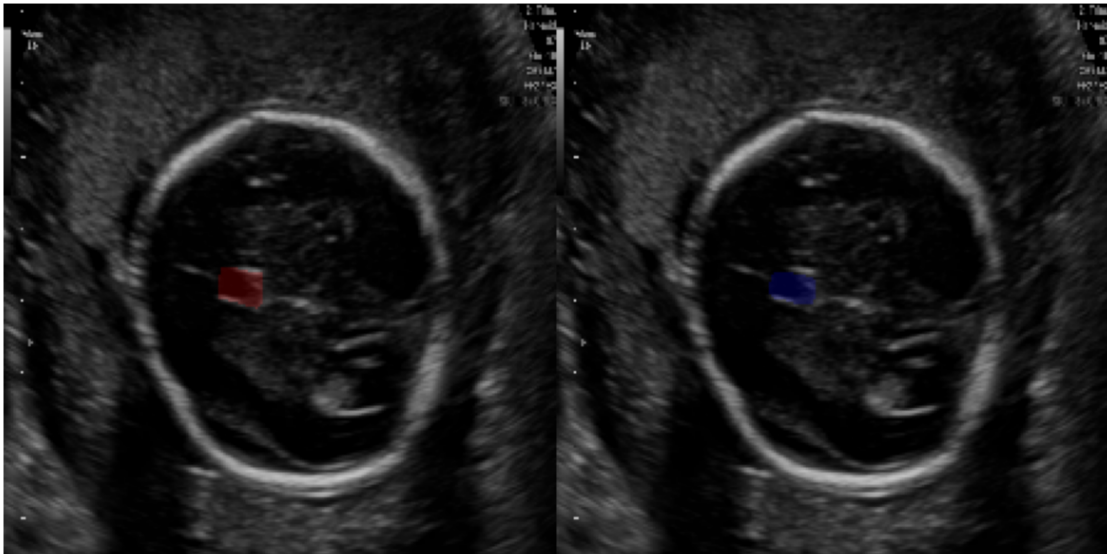


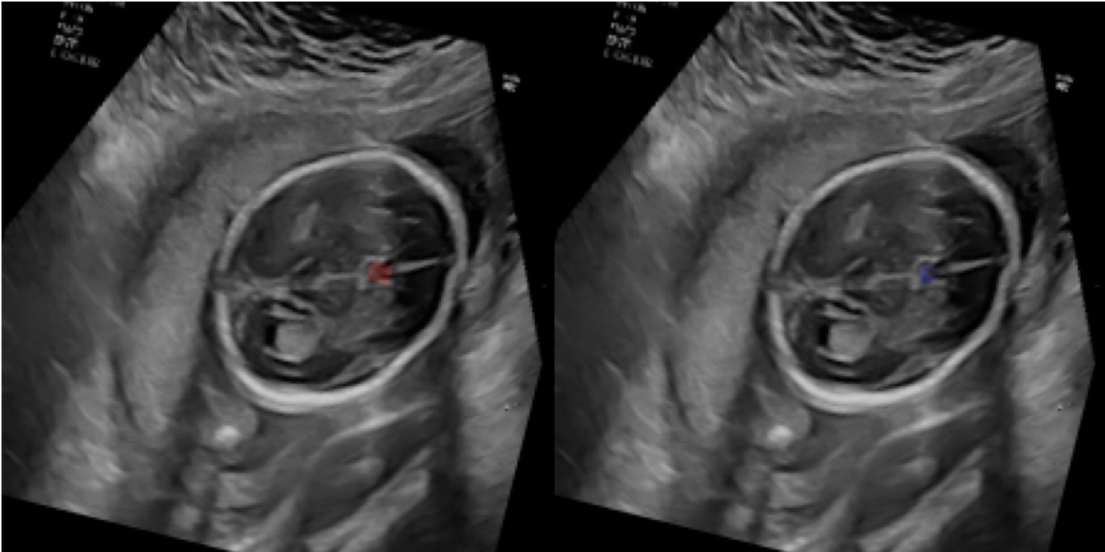
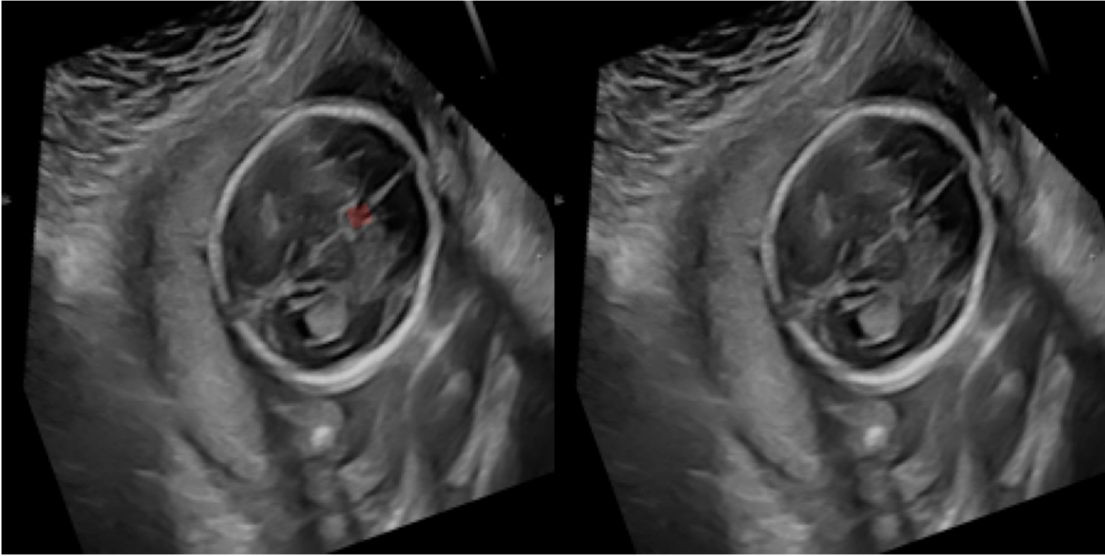


Carpeta: Trans-Ventricular, Base Nombre: Patient00216_Plane3_2_of_5_2 - Index 21

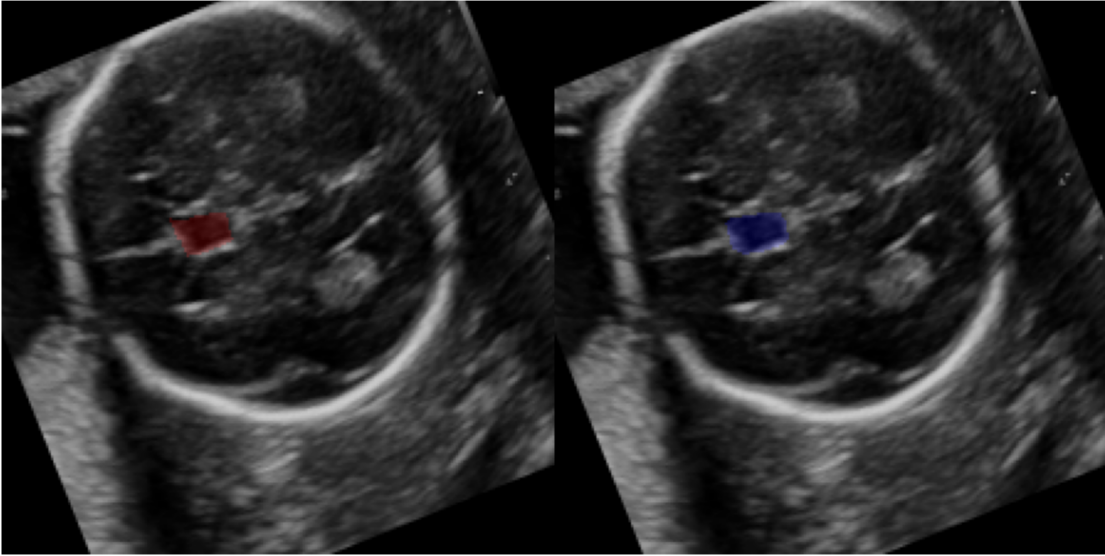


Carpeta: Trans-Ventricular, Base Nombre: Patient00216_Plane3_2_of_5 - Index 22

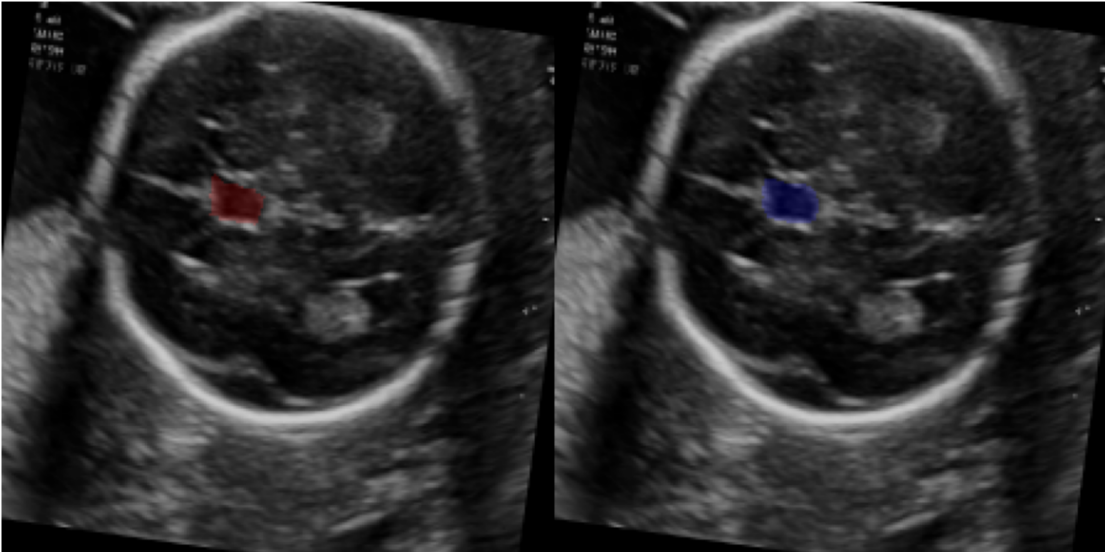




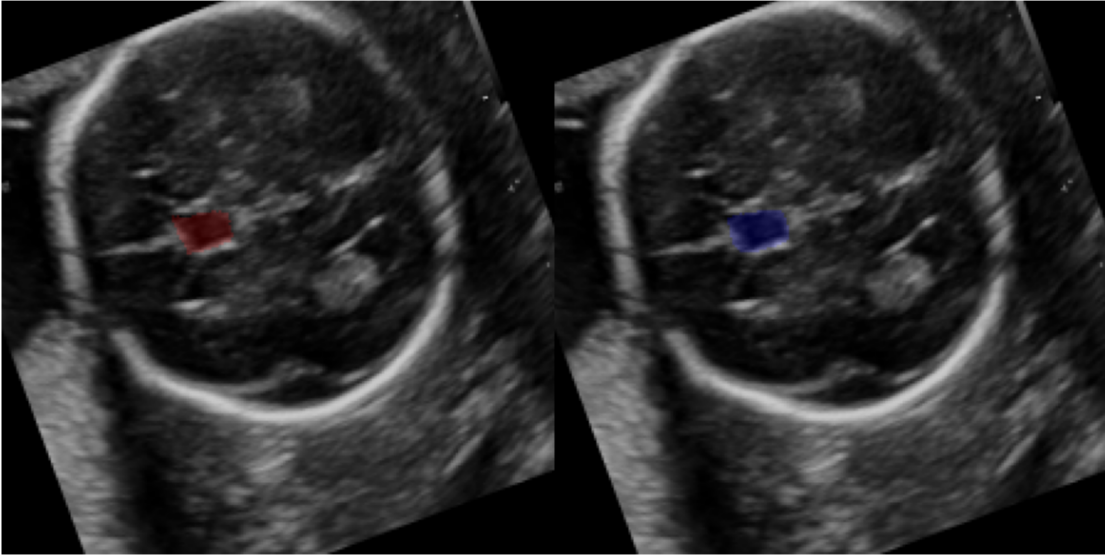
Carpeta: Trans-Ventricular, Base Nombre: Patient00644_Plane3_1_of_3_2 - Index 25



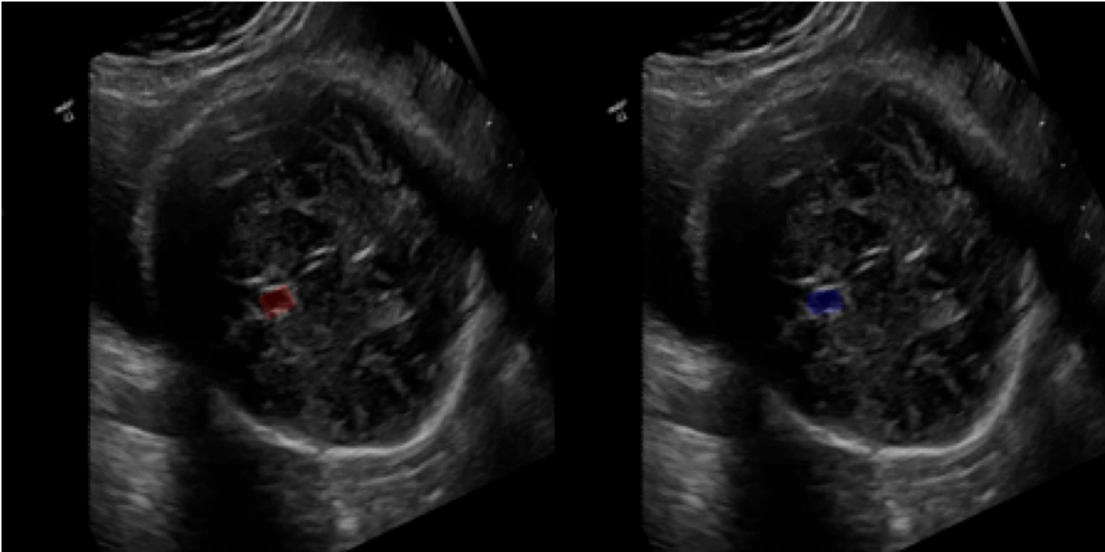
Carpeta: Trans-Ventricular, Base Nombre: Patient00644_Plane3_1_of_3_5 - Index 26

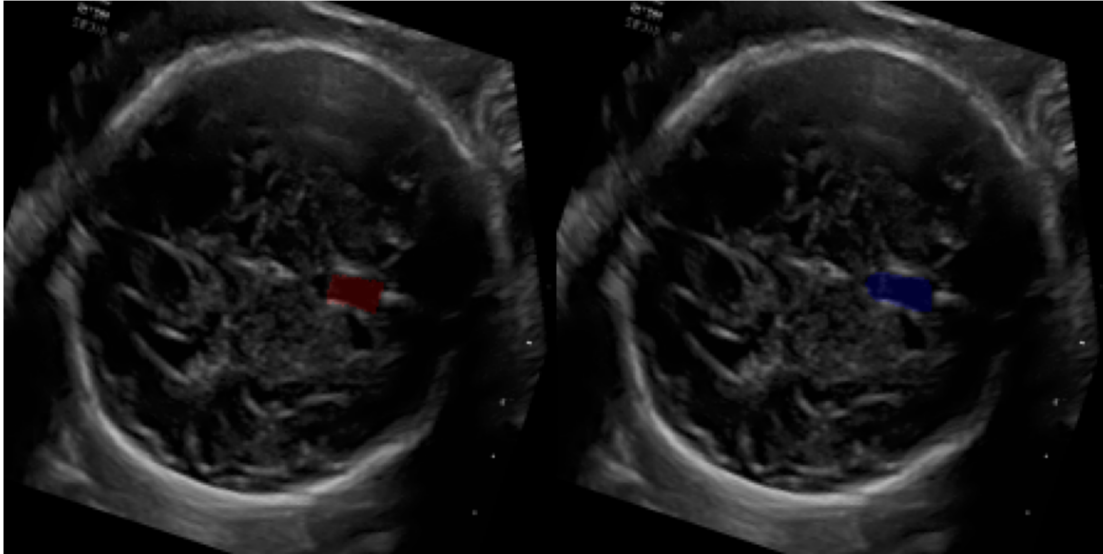


Carpeta: Trans-Ventricular, Base Nombre: Patient00644_Plane3_1_of_3_8 - Index 27



Carpeta: Trans-Ventricular, Base Nombre: Patient00688_Plane3_1_of_1_7 - Index 28





las anteriores imágenes nos muestran cómo quedan las imágenes con sus respectivas máscara original y máscara predicha, lastimosamente solo una imagen no genero máscara predicha, pero suponemos que se debe a que es muy pequeño el espacio de la estructura, se tendría que colocar las capas para obtener mejores detalles de las formas.

La máscara original se encuentra en color Rojo y la máscara predicha por el modelo está en color Azul

3.2 3.5. Calculo de área de la Estructura Cavum Septum Pellucidum dentro de la Máscara Predicha

Ahora procederemos a realizar el enmarcado del área del cavum septum pellucidum, para lo que usaremos `contourArea` de `opencv`, esto creara un recuadro alrededor de la estructura y calculara el ancho y el alto de la forma

```
[36]: # Seleccionar 2 ejemplos por cada valor único en la columna 'Carpeta'
df_muestras = df_resultados2.groupby('Carpeta').head(2)

# Recorrer cada fila del DataFrame con las muestras seleccionadas
for idx, row in df_muestras.iterrows():
    # Obtener la máscara y la máscara predicha del DataFrame
    mascara = row['Mascara']
    mascara_predicha = row['Mascara_Predicha']

    # Encontrar contornos en la máscara
    contornos, _ = cv2.findContours(mascara, cv2.RETR_EXTERNAL, cv2.
    CHAIN_APPROX_SIMPLE)
    if contornos:
        # Asumimos que solo hay un contorno relevante
```

```

    contorno = max(contornos, key=cv2.contourArea)
    x, y, ancho_mascara, alto_mascara = cv2.boundingRect(contorno)

    # Encontrar contornos en la máscara predicha
    contornos_predicha, _ = cv2.findContours(mascara_predicha, cv2.
    ↳RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if contornos_predicha:
        contorno_predicha = max(contornos_predicha, key=cv2.contourArea)
        x_predicha, y_predicha, ancho_mascara_predicha, alto_mascara_predicha =
    ↳cv2.boundingRect(contorno_predicha)

    # Graficar la máscara y la máscara predicha
    fig, axs = plt.subplots(1, 2, figsize=(12, 6))

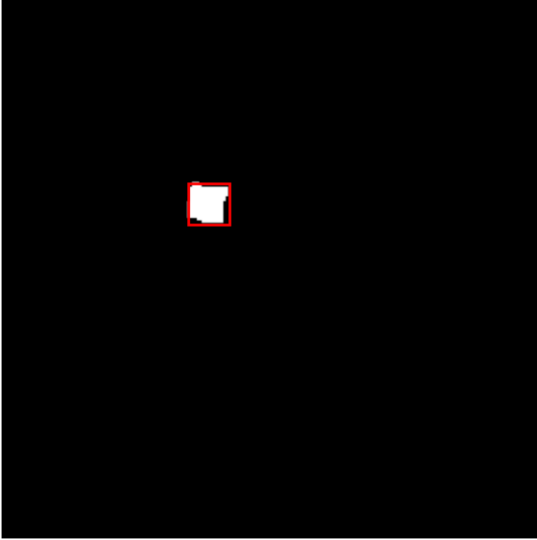
    # Graficar la máscara
    axs[0].imshow(mascara, cmap='gray')
    # Dibujar un rectángulo en la máscara
    rect_mascara = plt.Rectangle((x, y), ancho_mascara, alto_mascara,
    ↳linewidth=2, edgecolor='red', facecolor='none')
    axs[0].add_patch(rect_mascara)
    axs[0].set_title(f'Máscara (Carpeta: {row["Carpeta"]})')
    axs[0].axis('off')

    # Graficar la máscara predicha
    axs[1].imshow(mascara_predicha, cmap='gray')
    # Dibujar un rectángulo en la máscara predicha
    rect_mascara_predicha = plt.Rectangle((x_predicha, y_predicha),
    ↳ancho_mascara_predicha,
    alto_mascara_predicha, linewidth=2,
    ↳edgecolor='blue', facecolor='none')
    axs[1].add_patch(rect_mascara_predicha)
    axs[1].set_title(f'Máscara Predicha (Carpeta: {row["Carpeta"]})')
    axs[1].axis('off')

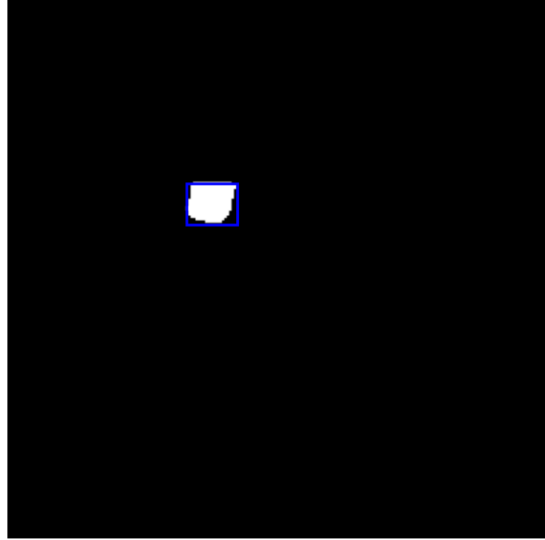
    plt.tight_layout()
    plt.show()

```

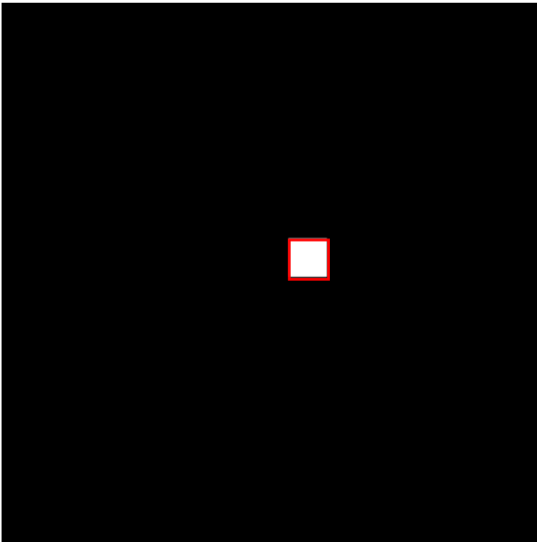
Máscara (Carpeta: Trans-Cerebellum)



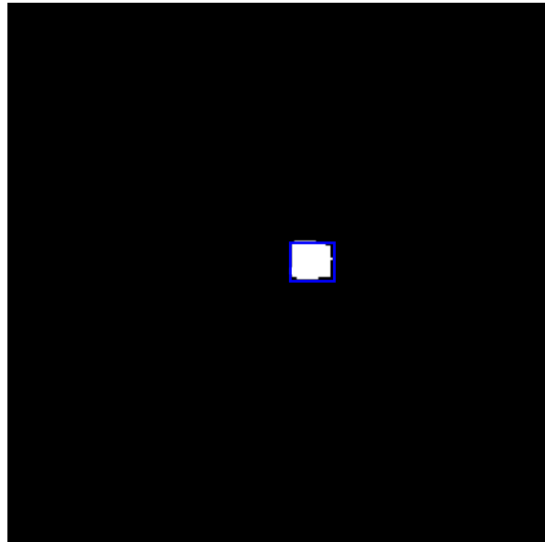
Máscara Predicha (Carpeta: Trans-Cerebellum)



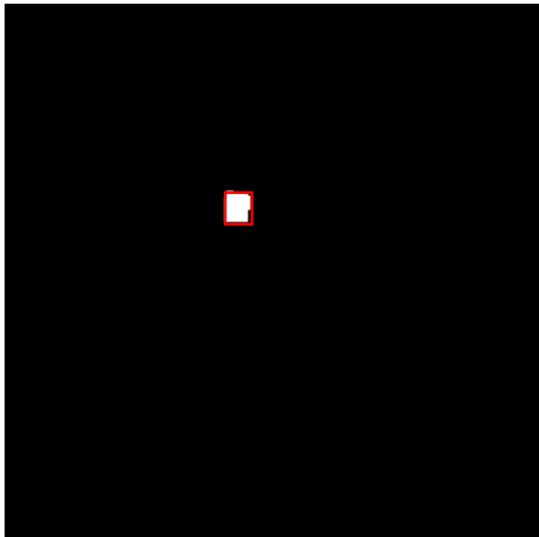
Máscara (Carpeta: Trans-Cerebellum)



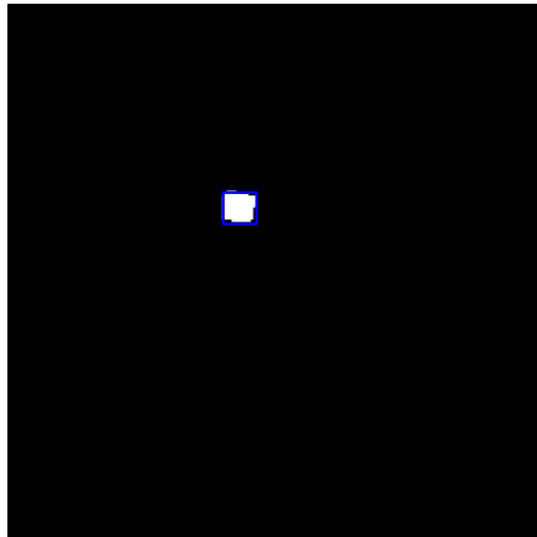
Máscara Predicha (Carpeta: Trans-Cerebellum)



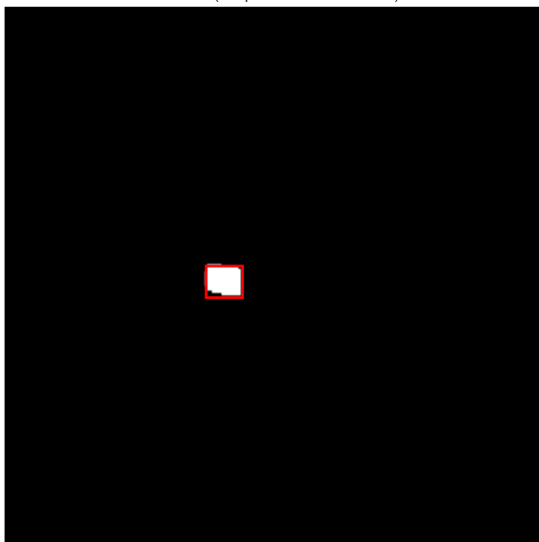
Máscara (Carpeta: Trans-Thalamic)



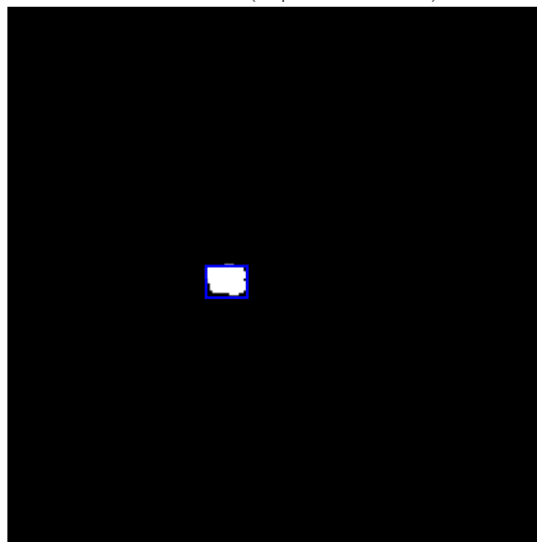
Máscara Predicha (Carpeta: Trans-Thalamic)

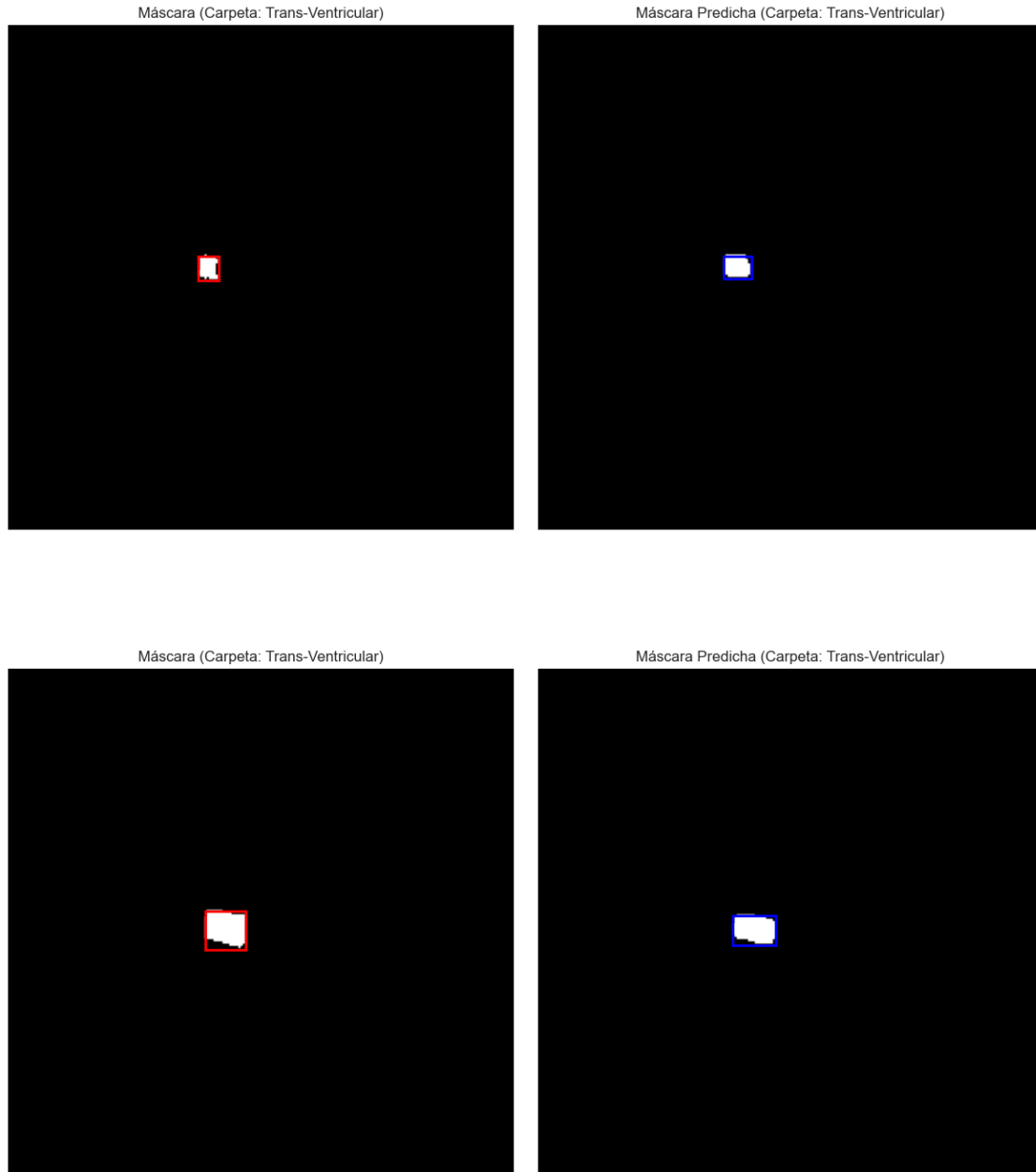


Máscara (Carpeta: Trans-Thalamic)



Máscara Predicha (Carpeta: Trans-Thalamic)





Como podemos ver estos son ejemplos de cada plano craneal

4 4.5. Calculo de área para determinar una posible anomalía por Tamaño

Primero realizaremos el cálculo de PPI (Píxeles por pulgada), teniendo en cuenta que la imagen tiene 18.97 mm de alto y 18.97 mm de Ancho, además de 224 píxeles por alto y 224 píxeles por ancho

```
[37]: # Datos proporcionados
ancho_pixeles = 224
alto_pixeles = 224
ancho_mm = 18.97 # tamaño en milímetros
alto_mm = 18.97 # tamaño en milímetros

# Convertir milímetros a pulgadas (1 pulgada = 25.4 mm)
ancho_pulgadas = ancho_mm / 25.4
alto_pulgadas = alto_mm / 25.4

# Calcular PPI
ppi_ancho = ancho_pixeles / ancho_pulgadas
ppi_alto = alto_pixeles / alto_pulgadas

# Usar cualquiera de los dos para el valor de PPI
ppi = ppi_ancho # o ppi_alto, ambos serán iguales
ppi
```

[37]: 299.9261992619926

el ppi tentativo es de 299.92 por lo que tomaremos el estándar de 300 PPI

Agregaremos al dataframe las columnas de alto, ancho y área, tanto de la máscara, como de la máscara predicha

```
[38]: # Creamos una copia
df_resultados = df_resultados2.copy()

# Agregar columnas al DataFrame para almacenar las métricas en milímetros
df_resultados['Alto_Mascara_Completa_mm'] = 0.0
df_resultados['Ancho_Mascara_Completa_mm'] = 0.0
df_resultados['Area_Mascara_Completa_mm2'] = 0.0
df_resultados['Alto_Estructura_Blanca_mm'] = 0.0
df_resultados['Ancho_Estructura_Blanca_mm'] = 0.0
df_resultados['Area_Estructura_Blanca_mm2'] = 0.0
df_resultados['Alto_Estructura_Blanca_Predicha_mm'] = 0.0
df_resultados['Ancho_Estructura_Blanca_Predicha_mm'] = 0.0
df_resultados['Area_Estructura_Blanca_Predicha_mm2'] = 0.0

# Definir PPI
ppi = 300
mm_per_inch = 25.4

# Calcular y almacenar métricas en milímetros
for idx, row in df_resultados.iterrows():
    # Obtener la máscara y la máscara predicha
    mascara = row['Mascara']
    mascara_predicha = row['Mascara_Predicha']
```

```

# Alto y ancho de la máscara completa (en milímetros)
alto_mascara_completa, ancho_mascara_completa = mascara.shape
df_resultados.at[idx, 'Alto_Mascara_Completa_mm'] =
↪float(alto_mascara_completa / ppi * mm_per_inch)
df_resultados.at[idx, 'Ancho_Mascara_Completa_mm'] =
↪float(ancho_mascara_completa / ppi * mm_per_inch)
df_resultados.at[idx, 'Area_Mascara_Completa_mm2'] =
↪float((alto_mascara_completa * ancho_mascara_completa) / (ppi**2) *
↪(mm_per_inch**2))

# Encontrar contornos en la máscara
contornos, _ = cv2.findContours(mascara, cv2.RETR_EXTERNAL, cv2.
↪CHAIN_APPROX_SIMPLE)
if contornos:
    contorno = max(contornos, key=cv2.contourArea)
    x, y, ancho_blanqueado, alto_blanqueado = cv2.boundingRect(contorno)
    area_blanqueada = cv2.contourArea(contorno)

# Almacenar valores de la estructura blanca (en milímetros)
df_resultados.at[idx, 'Alto_Estructura_Blanca_mm'] =
↪float(alto_blanqueado / ppi * mm_per_inch)
df_resultados.at[idx, 'Ancho_Estructura_Blanca_mm'] =
↪float(ancho_blanqueado / ppi * mm_per_inch)
df_resultados.at[idx, 'Area_Estructura_Blanca_mm2'] =
↪float(area_blanqueada / (ppi**2) * (mm_per_inch**2))

# Alto y ancho de la máscara predicha (en píxeles)
alto_mascara_predicha, ancho_mascara_predicha = mascara_predicha.shape
df_resultados.at[idx, 'Alto_Mascara_Completa_Predicha_mm'] =
↪float(alto_mascara_predicha / ppi * mm_per_inch)
df_resultados.at[idx, 'Ancho_Mascara_Completa_Predicha_mm'] =
↪float(ancho_mascara_predicha / ppi * mm_per_inch)
df_resultados.at[idx, 'Area_Mascara_Completa_Predicha_mm2'] =
↪float((alto_mascara_predicha * ancho_mascara_predicha) / (ppi**2) *
↪(mm_per_inch**2))

# Encontrar contornos en la máscara predicha
contornos_predicha, _ = cv2.findContours(mascara_predicha, cv2.
↪RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
if contornos_predicha:
    contorno_predicho = max(contornos_predicha, key=cv2.contourArea)
    x_predicho, y_predicho, ancho_blanqueado_predicho,
↪alto_blanqueado_predicho = cv2.boundingRect(contorno_predicho)
    area_blanqueada_predicha = cv2.contourArea(contorno_predicho)

```



```

        # Almacenar valores de la estructura blanca predicha (en milímetros)
        df_resultados.at[idx, 'Alto_Estructura_Blanca_Predicha_mm'] =
↪float(alto_blanqueado_predicho / ppi * mm_per_inch)
        df_resultados.at[idx, 'Ancho_Estructura_Blanca_Predicha_mm'] =
↪float(ancho_blanqueado_predicho / ppi * mm_per_inch)
        df_resultados.at[idx, 'Area_Estructura_Blanca_Predicha_mm2'] =
↪float(area_blanqueada_predicha / (ppi**2) * (mm_per_inch**2))

# Agregar columnas para el largo y ancho geométrico
df_resultados['Largo_Geometrico_mm'] = 0.0
df_resultados['Ancho_Geometrico_mm'] = 0.0

# Calcular métricas adicionales
for idx, row in df_resultados.iterrows():
    # Obtener la máscara
    mascara = row['Mascara_Predicha']

    # Encontrar contornos en la máscara
    contornos, _ = cv2.findContours(mascara, cv2.RETR_EXTERNAL, cv2.
↪CHAIN_APPROX_SIMPLE)
    if contornos:
        # Seleccionar el contorno más grande
        contorno = max(contornos, key=cv2.contourArea)

        # Calcular extremos geométricos
        x_min = min(point[0][0] for point in contorno)
        x_max = max(point[0][0] for point in contorno)
        y_min = min(point[0][1] for point in contorno)
        y_max = max(point[0][1] for point in contorno)

        # Calcular largo y ancho geométrico
        largo_geom_px = x_max - x_min # Largo en píxeles (distancia entre
↪extremos en X)
        ancho_geom_px = y_max - y_min # Ancho en píxeles (distancia entre
↪extremos en Y)

        # Convertir a milímetros
        largo_geom_mm = largo_geom_px / ppi * mm_per_inch
        ancho_geom_mm = ancho_geom_px / ppi * mm_per_inch

        # Almacenar en el DataFrame
        df_resultados.at[idx, 'Largo_Geometrico_mm'] = largo_geom_mm
        df_resultados.at[idx, 'Ancho_Geometrico_mm'] = ancho_geom_mm

# Crear la columna diagnostico_tamaño con la condición especificada

```

```

df_resultados['diagnostico_tamaño'] = df_resultados.apply(
    lambda row: 'Anormal' if row['Largo_Geometrico_mm'] == 0 or
    row['Ancho_Geometrico_mm'] == 0 or row['Largo_Geometrico_mm'] > 1.3 *
    row['Ancho_Geometrico_mm'] else 'Normal',
    axis=1
)

# Crear la columna 'Radio' calculando la relación entre 'Largo_Geometrico_mm' y
# 'Ancho_Geometrico_mm'
df_resultados['Radio'] = df_resultados.apply(
    lambda row: row['Largo_Geometrico_mm'] / row['Ancho_Geometrico_mm'] if
    row['Ancho_Geometrico_mm'] != 0 else 0,
    axis=1
)

# Definir la ruta donde guardar el archivo Excel
ruta = r'D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de Grado\003 -
Trabajo de grado\Data\Mascaras e Imagenes\Vazlidacion\df_resultados.xlsx'

# Verificar si el archivo ya existe y eliminarlo si es así
if os.path.exists(ruta):
    os.remove(ruta)
    print(f"El archivo {ruta} ya existía y ha sido eliminado.")

# Guardar el DataFrame en un archivo Excel
df_resultados[['Base_Nombre',
                'Alto_Mascara_Completa_mm',
                'Ancho_Mascara_Completa_mm',
                'Area_Mascara_Completa_mm2',
                'Largo_Geometrico_mm',
                'Ancho_Geometrico_mm',
                'Alto_Estructura_Blanca_mm',
                'Ancho_Estructura_Blanca_mm',
                'Area_Estructura_Blanca_mm2',
                'Alto_Estructura_Blanca_Predicha_mm',
                'Ancho_Estructura_Blanca_Predicha_mm',
                'Area_Estructura_Blanca_Predicha_mm2',
                'diagnostico_tamaño',
                'Radio']].to_excel(ruta, index=False)

print(f"El archivo Excel se ha guardado en: {ruta}")

# Mostrar el DataFrame con las nuevas columnas
df_resultados[['Base_Nombre',
                'Alto_Mascara_Completa_mm',
                'Ancho_Mascara_Completa_mm',

```

```

'Area_Mascara_Completa_mm2',
'Largo_Geometrico_mm',
'Ancho_Geometrico_mm',
#'Alto_Estructura_Blanca_mm',
#'Ancho_Estructura_Blanca_mm',
#'Area_Estructura_Blanca_mm2',
#'Alto_Estructura_Blanca_Predicha_mm',
#'Ancho_Estructura_Blanca_Predicha_mm',
#'Area_Estructura_Blanca_Predicha_mm2',
'diagnostico_tamaño',
'Radio',
'Carpeta',
#'Imagen',
#'Mascara',
#'Mascara_Predicha'
]] .head(30)

```

El archivo D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de Grado\003 - Trabajo de grado\Data\Mascaras e Imagenes\Vazlidacion\df_resultados.xlsx ya existía y ha sido eliminado.

El archivo Excel se ha guardado en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de Grado\003 - Trabajo de grado\Data\Mascaras e Imagenes\Vazlidacion\df_resultados.xlsx

```

[38]:
      Base_Nombre  Alto_Mascara_Completa_mm  \
0  Patient00644_Plane3_2_of_3_5           18.965333
1  Patient00662_Plane3_1_of_1_2           18.965333
2    Patient00662_Plane3_1_of_1           18.965333
3  Patient00675_Plane3_1_of_1_2           18.965333
4  Patient00675_Plane3_1_of_1_3           18.965333
5  Patient00687_Plane3_1_of_1_8           18.965333
6  Patient00706_Plane3_5_of_5_2           18.965333
7  Patient00706_Plane3_5_of_5_4           18.965333
8  Patient00706_Plane3_5_of_5_6           18.965333
9  Patient00706_Plane3_5_of_5_8           18.965333
10 Patient00168_Plane3_2_of_3             18.965333
11 Patient00216_Plane3_1_of_5_3           18.965333
12 Patient00216_Plane3_1_of_5_5           18.965333
13 Patient00216_Plane3_1_of_5_6           18.965333
14 Patient00305_Plane3_5_of_5_2           18.965333
15 Patient00305_Plane3_5_of_5_3           18.965333
16 Patient00305_Plane3_5_of_5             18.965333
17 Patient00640_Plane3_1_of_1             18.965333
18 Patient00647_Plane3_1_of_1             18.965333
19 Patient00658_Plane3_1_of_1             18.965333
20 Patient00188_Plane3_2_of_3_6           18.965333
21 Patient00216_Plane3_2_of_5_2           18.965333

```

22	Patient00216_Plane3_2_of_5	18.965333
23	Patient00305_Plane3_1_of_5_2	18.965333
24	Patient00305_Plane3_1_of_5_4	18.965333
25	Patient00644_Plane3_1_of_3_2	18.965333
26	Patient00644_Plane3_1_of_3_5	18.965333
27	Patient00644_Plane3_1_of_3_8	18.965333
28	Patient00688_Plane3_1_of_1_7	18.965333
29	Patient00700_Plane3_2_of_2_3	18.965333

	Ancho_Mascara_Completa_mm	Area_Mascara_Completa_mm2	Largo_Geometrico_mm	\
0	18.965333	359.683868	1.693333	
1	18.965333	359.683868	1.439333	
2	18.965333	359.683868	1.439333	
3	18.965333	359.683868	1.778000	
4	18.965333	359.683868	1.778000	
5	18.965333	359.683868	1.693333	
6	18.965333	359.683868	1.693333	
7	18.965333	359.683868	1.693333	
8	18.965333	359.683868	1.608667	
9	18.965333	359.683868	1.778000	
10	18.965333	359.683868	1.100667	
11	18.965333	359.683868	1.354667	
12	18.965333	359.683868	1.354667	
13	18.965333	359.683868	1.524000	
14	18.965333	359.683868	1.524000	
15	18.965333	359.683868	1.439333	
16	18.965333	359.683868	1.524000	
17	18.965333	359.683868	0.931333	
18	18.965333	359.683868	1.016000	
19	18.965333	359.683868	1.439333	
20	18.965333	359.683868	0.931333	
21	18.965333	359.683868	1.524000	
22	18.965333	359.683868	1.524000	
23	18.965333	359.683868	0.000000	
24	18.965333	359.683868	0.508000	
25	18.965333	359.683868	2.116667	
26	18.965333	359.683868	1.862667	
27	18.965333	359.683868	2.032000	
28	18.965333	359.683868	1.185333	
29	18.965333	359.683868	2.201333	

	Ancho_Geometrico_mm	diagnostico_tamaño	Radio	Carpeta
0	1.354667	Normal	1.250000	Trans-Cerebellum
1	1.270000	Normal	1.133333	Trans-Cerebellum
2	1.270000	Normal	1.133333	Trans-Cerebellum
3	1.270000	Anormal	1.400000	Trans-Cerebellum
4	1.185333	Anormal	1.500000	Trans-Cerebellum

5	1.693333	Normal	1.000000	Trans-Cerebellum
6	1.354667	Normal	1.250000	Trans-Cerebellum
7	1.354667	Normal	1.250000	Trans-Cerebellum
8	1.524000	Normal	1.055556	Trans-Cerebellum
9	1.439333	Normal	1.235294	Trans-Cerebellum
10	1.016000	Normal	1.083333	Trans-Thalamic
11	1.016000	Anormal	1.333333	Trans-Thalamic
12	1.016000	Anormal	1.333333	Trans-Thalamic
13	0.931333	Anormal	1.636364	Trans-Thalamic
14	1.185333	Normal	1.285714	Trans-Thalamic
15	1.185333	Normal	1.214286	Trans-Thalamic
16	1.270000	Normal	1.200000	Trans-Thalamic
17	0.592667	Anormal	1.571429	Trans-Thalamic
18	0.592667	Anormal	1.714286	Trans-Thalamic
19	1.185333	Normal	1.214286	Trans-Thalamic
20	0.762000	Normal	1.222222	Trans-Ventricular
21	1.016000	Anormal	1.500000	Trans-Ventricular
22	1.016000	Anormal	1.500000	Trans-Ventricular
23	0.000000	Anormal	0.000000	Trans-Ventricular
24	0.508000	Normal	1.000000	Trans-Ventricular
25	1.354667	Anormal	1.562500	Trans-Ventricular
26	1.439333	Normal	1.294118	Trans-Ventricular
27	1.354667	Anormal	1.500000	Trans-Ventricular
28	0.762000	Anormal	1.555556	Trans-Ventricular
29	1.270000	Anormal	1.733333	Trans-Ventricular

```
[39]: # Definir el PPI y los valores de conversión a milímetros
ppi = 300
mm_per_inch = 25.4

# Definir la ruta donde guardar las imágenes
ruta_guardar_imagenes = r'D:\Archivos Milton\Maestria en Ciencia de
↳ Datos\Trabajo de Grado\003 - Trabajo de grado\Data\Mascaras e
↳ Imagenes\Vazlidacion'

# Recorrer todas las filas del DataFrame
for idx, row in df_resultados.iterrows():
    imagen = row['Imagen']
    mascara_predicha = row['Mascara_Predicha']
    base_nombre = row['Base_Nombre']
    diagnostico_tamaño = row['diagnostico_tamaño']
    radio = row['Radio']

# Crear una figura para graficar
fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB))
```

```

# Verificar si existe una máscara predicha
if mascara_predicha is not None and mascara_predicha.any(): # Si la
↳máscara no está vacía
    # Encontrar contornos de la máscara predicha
    contornos, _ = cv2.findContours(mascara_predicha, cv2.RETR_EXTERNAL,
↳cv2.CHAIN_APPROX_SIMPLE)

    # Calcular extremos si hay contornos
    if contornos:
        # Seleccionar el contorno más grande
        contorno = max(contornos, key=cv2.contourArea)

        # Obtener extremos geométricos
        x_min = min(point[0][0] for point in contorno)
        x_max = max(point[0][0] for point in contorno)
        y_min = min(point[0][1] for point in contorno)
        y_max = max(point[0][1] for point in contorno)

        # Calcular largo y ancho en milímetros
        largo_mm = (x_max - x_min) / ppi * mm_per_inch
        ancho_mm = (y_max - y_min) / ppi * mm_per_inch

        # Dibujar la máscara predicha sobrepuesta
        ax.contour(mascara_predicha, colors='magenta', linewidths=1.5)

        # Dibujar el largo (línea horizontal entre extremos X)
        ax.plot([x_min, x_max], [y_min, y_min], 'r-', label=f'Largo
↳geométrico (X): {largo_mm:.2f} mm')
        ax.plot([x_min, x_max], [y_max, y_max], 'r-')

        # Dibujar el ancho (línea vertical entre extremos Y)
        ax.plot([x_min, x_min], [y_min, y_max], 'g-', label=f'Ancho
↳geométrico (Y): {ancho_mm:.2f} mm')
        ax.plot([x_max, x_max], [y_min, y_max], 'g-')

        # Colocar los valores de largo y ancho en milímetros sobre la imagen
        ax.text(x_min, y_min - 10, f'Largo: {largo_mm:.2f} mm',
↳color='red', fontsize=12)
        ax.text(x_min, y_max + 5, f'Ancho: {ancho_mm:.2f} mm',
↳color='green', fontsize=12)
    else:
        largo_mm = ancho_mm = 0

else:
    largo_mm = ancho_mm = 0
    ax.set_title('Imagen sin Máscara Predicha')

```

```

# Agregar diagnóstico y radio a la imagen
ax.text(10, 30, f'Diagnóstico: {diagnostico_tamaño}', color='blue',
↪fontsize=12)
ax.text(10, 60, f'Radio: {radio:.2f}', color='purple', fontsize=12)

# Mostrar la imagen antes de guardarla
ax.axis('off')
plt.show()

# Guardar la imagen con un nombre único basado en el Base_Nombre
nombre_imagen_guardar = f"{base_nombre}_con_mascara.png"
ruta_imagen = os.path.join(ruta_guardar_imagenes, nombre_imagen_guardar)

# Verificar si la imagen ya existe y eliminarla si es así
if os.path.exists(ruta_imagen):
    os.remove(ruta_imagen)
    print(f"Imagen {nombre_imagen_guardar} eliminada y recreada.")

# Guardar la imagen
fig.savefig(ruta_imagen, bbox_inches='tight', pad_inches=0.1)
plt.close(fig) # Cerrar la figura para liberar memoria

print(f"Imagen guardada en: {ruta_imagen}")

```

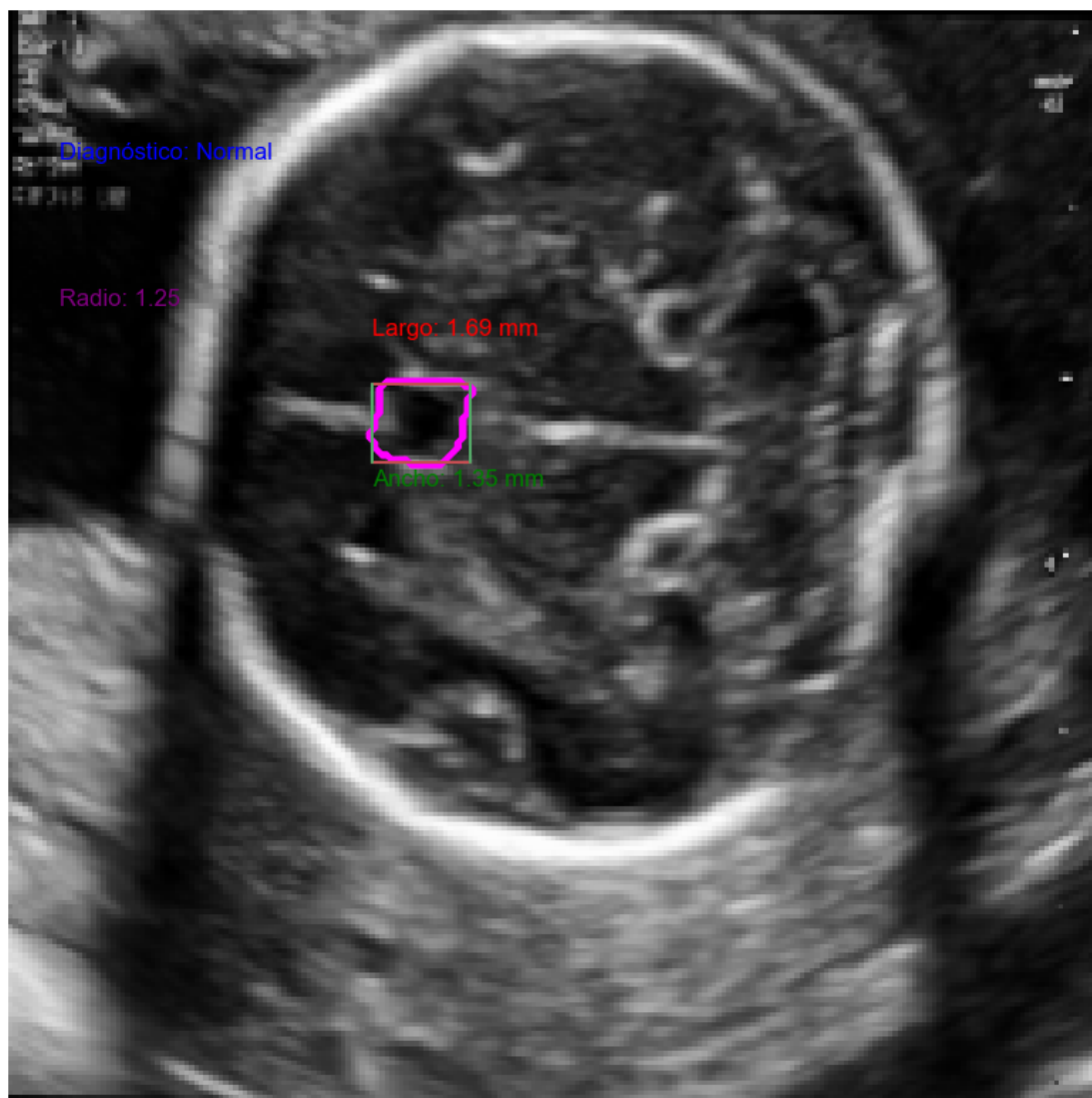


Imagen Patient00644_Plane3_2_of_3_5_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00644_Plane3_2_of_3_5_con_mascara.png

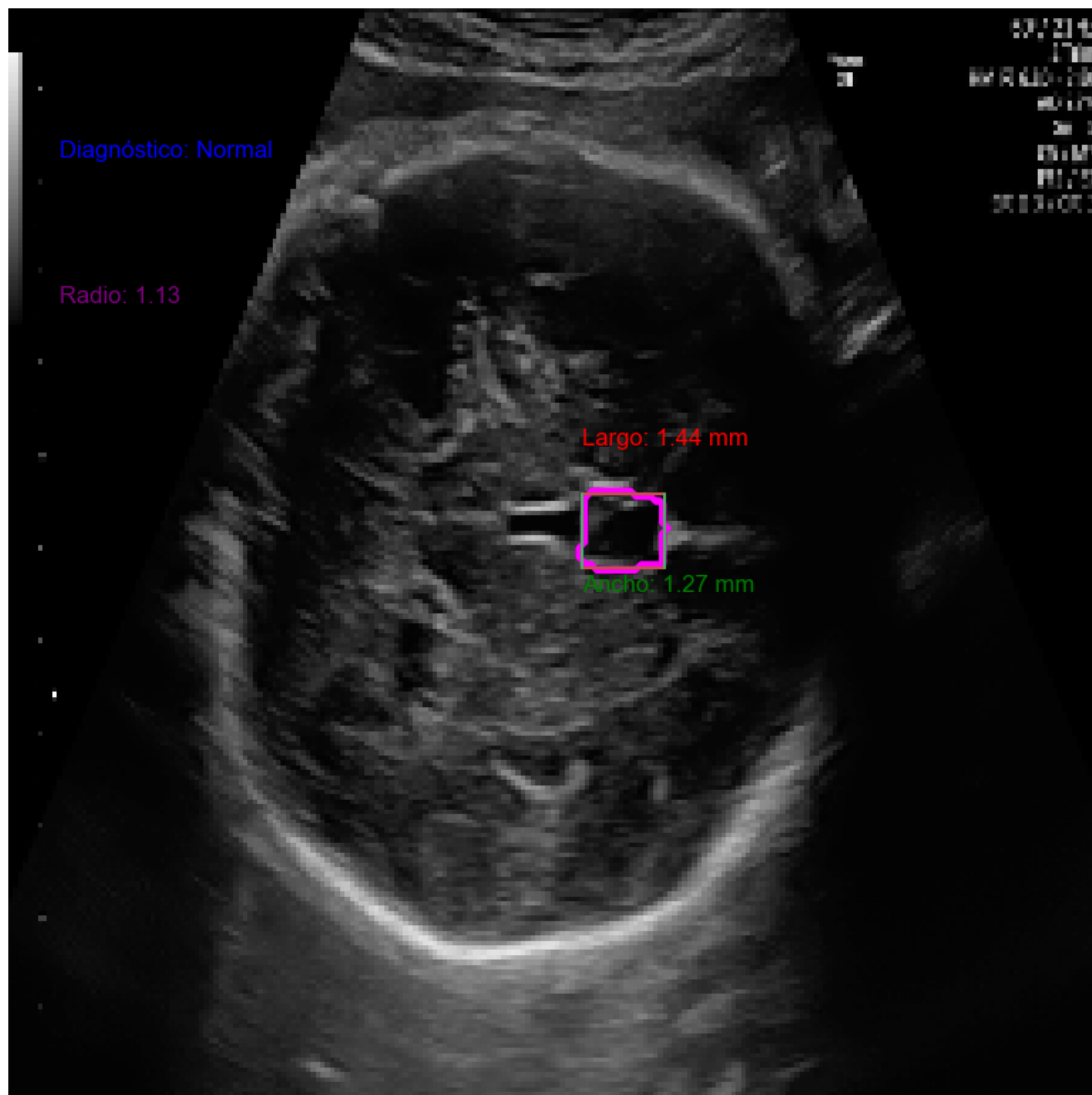


Imagen Patient00662_Plane3_1_of_1_2_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00662_Plane3_1_of_1_2_con_mascara.png

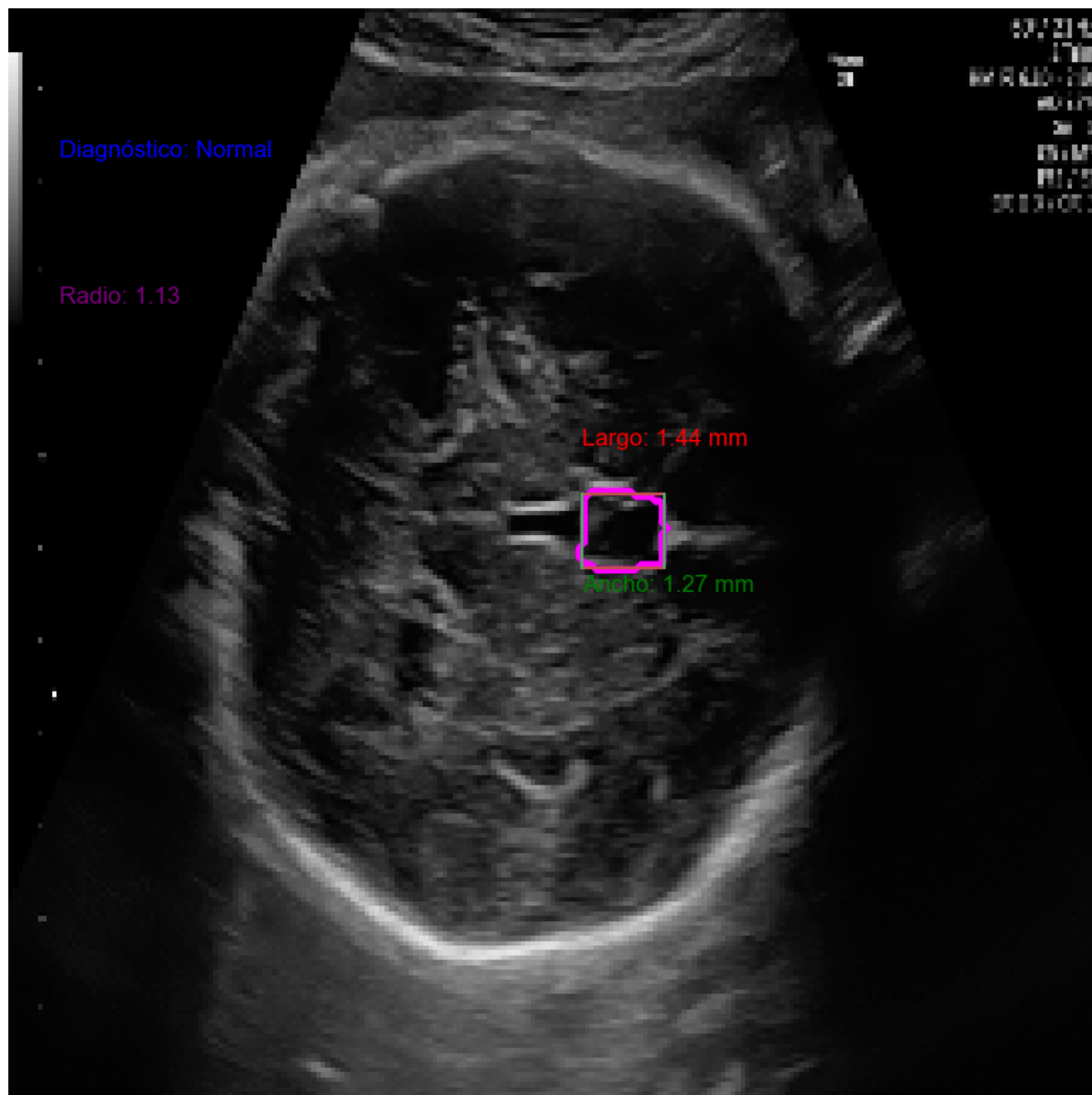


Imagen Patient00662_Plane3_1_of_1_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00662_Plane3_1_of_1_con_mascara.png

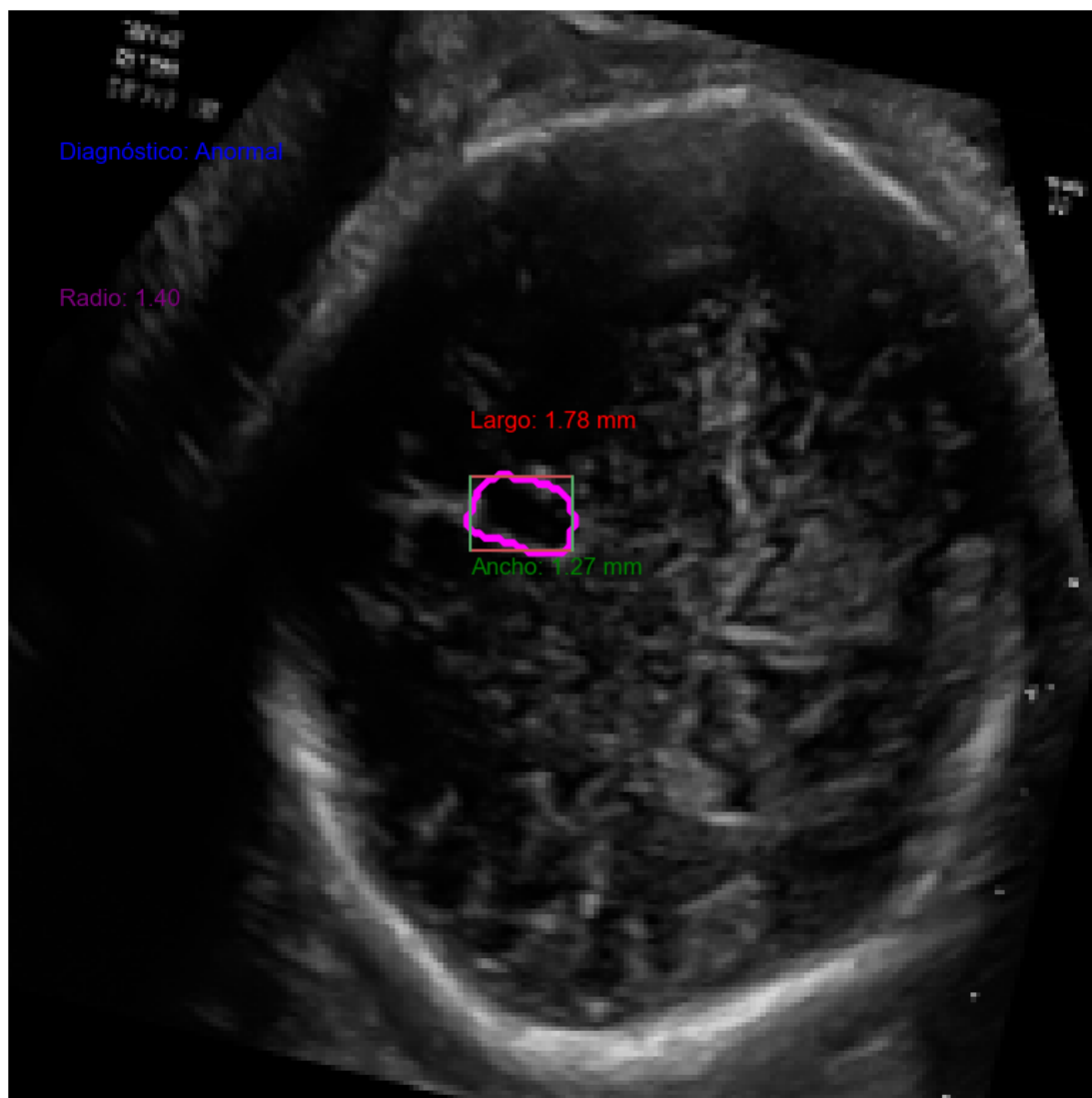


Imagen Patient00675_Plane3_1_of_1_2_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00675_Plane3_1_of_1_2_con_mascara.png

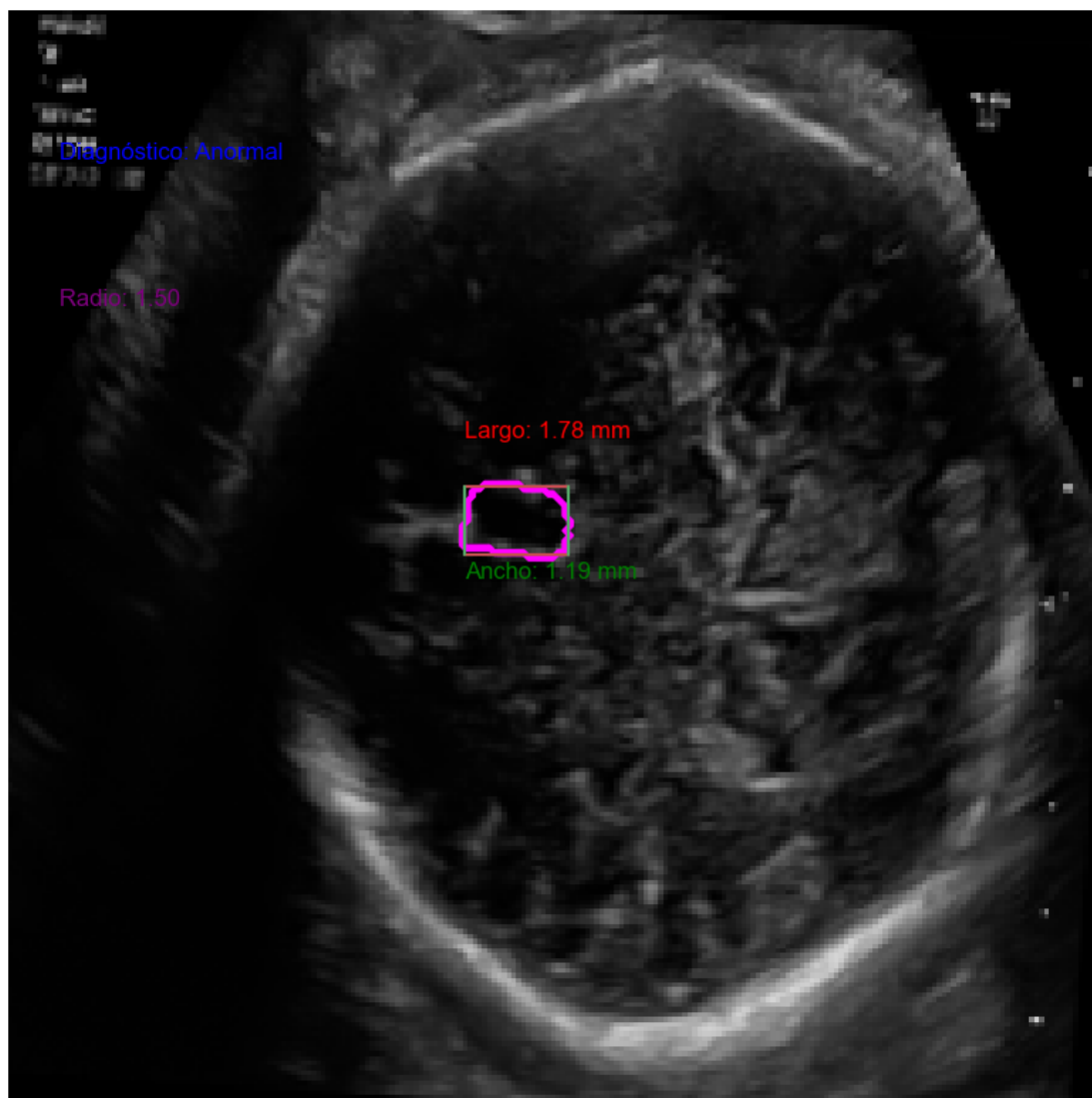


Imagen Patient00675_Plane3_1_of_1_3_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00675_Plane3_1_of_1_3_con_mascara.png

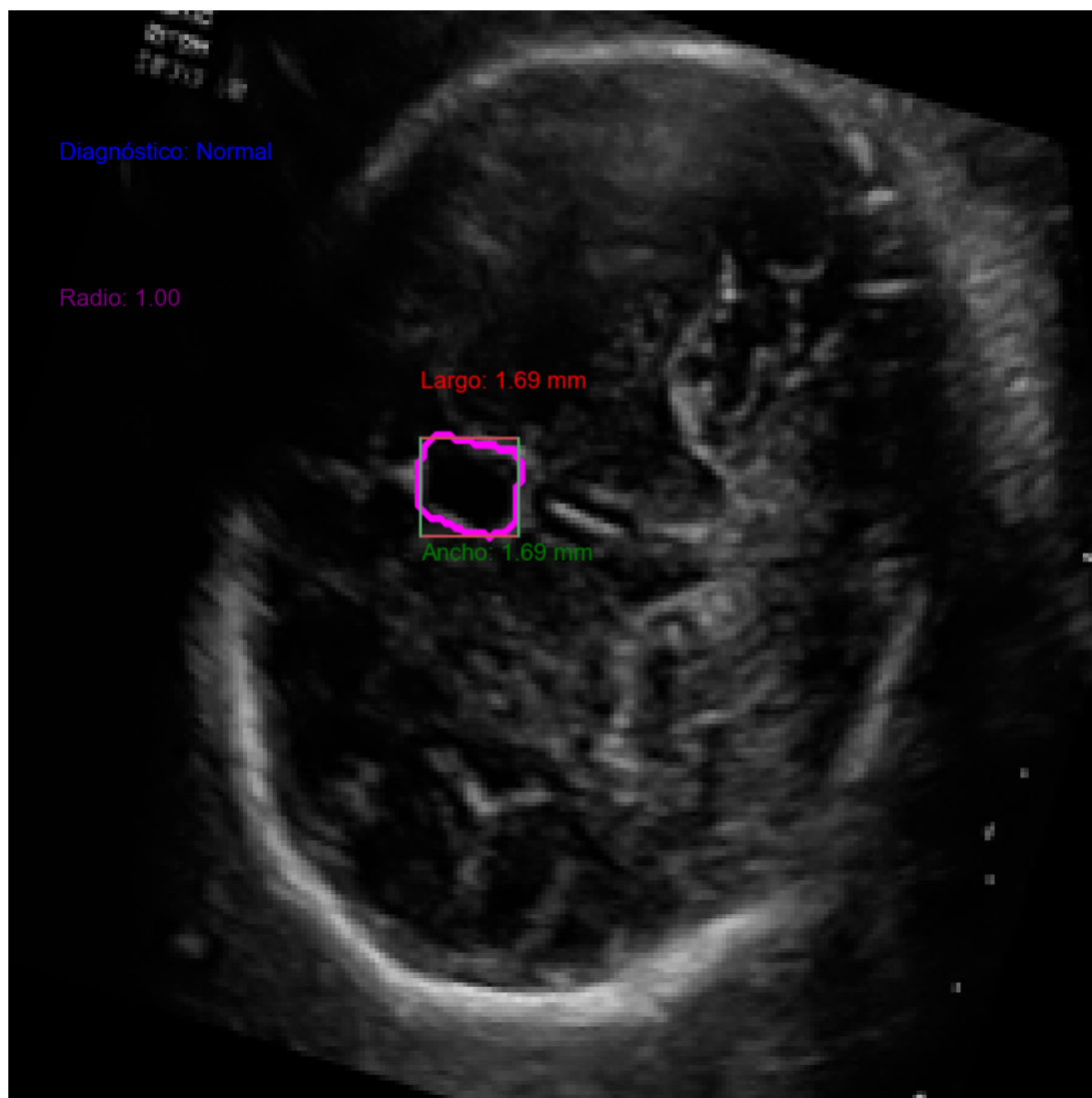


Imagen Patient00687_Plane3_1_of_1_8_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00687_Plane3_1_of_1_8_con_mascara.png

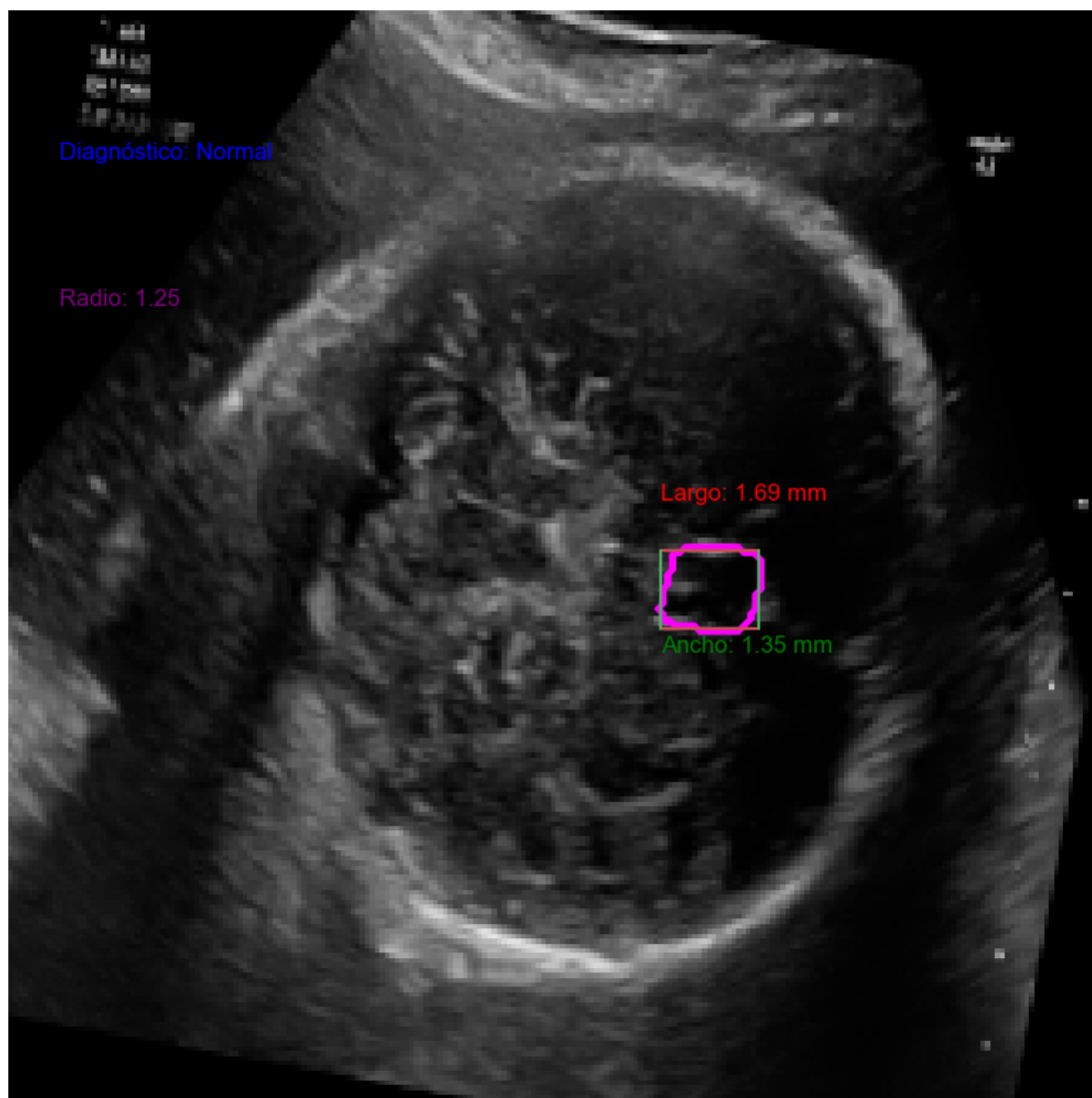


Imagen Patient00706_Plane3_5_of_5_2_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00706_Plane3_5_of_5_2_con_mascara.png

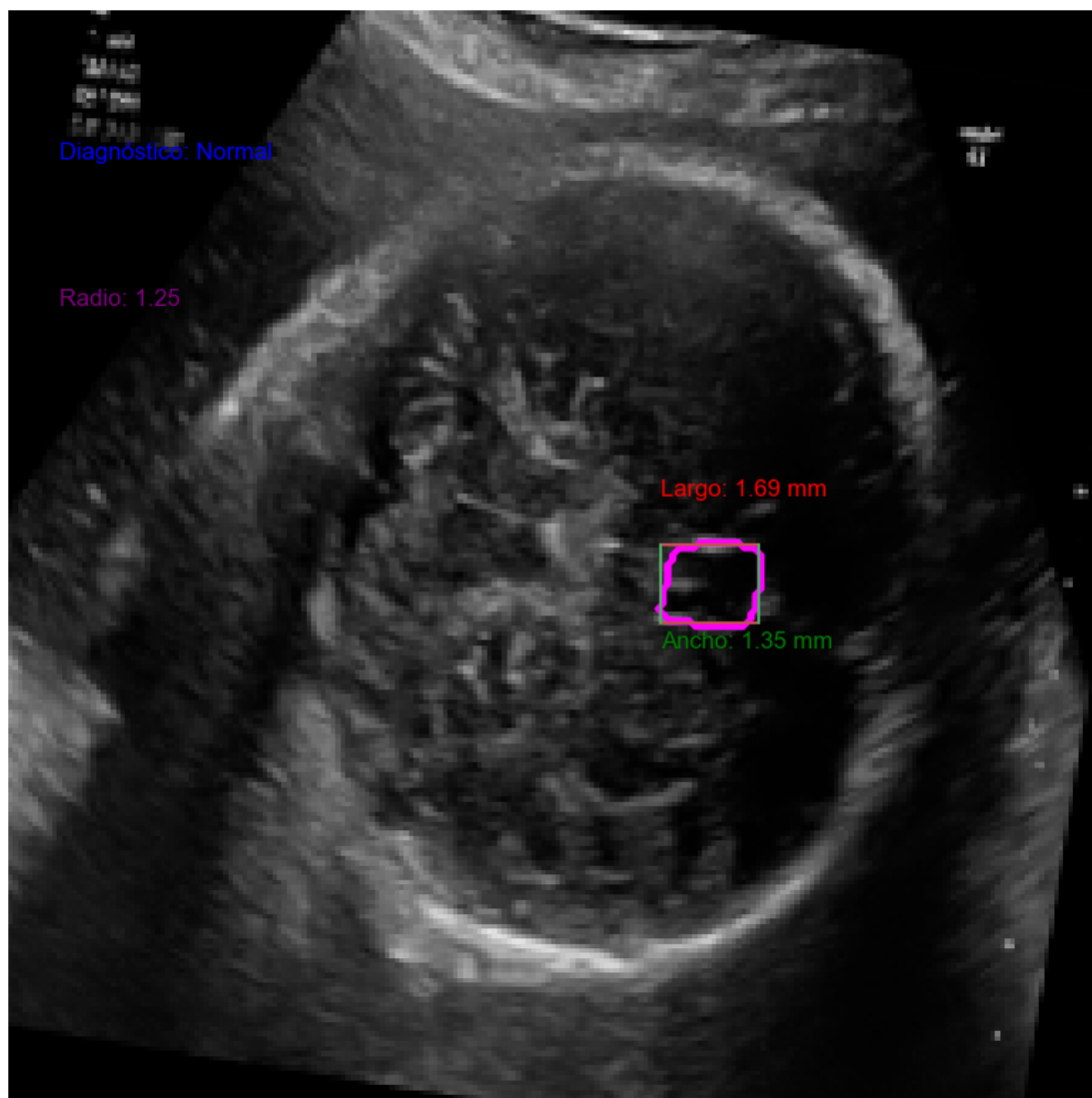


Imagen Patient00706_Plane3_5_of_5_4_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00706_Plane3_5_of_5_4_con_mascara.png

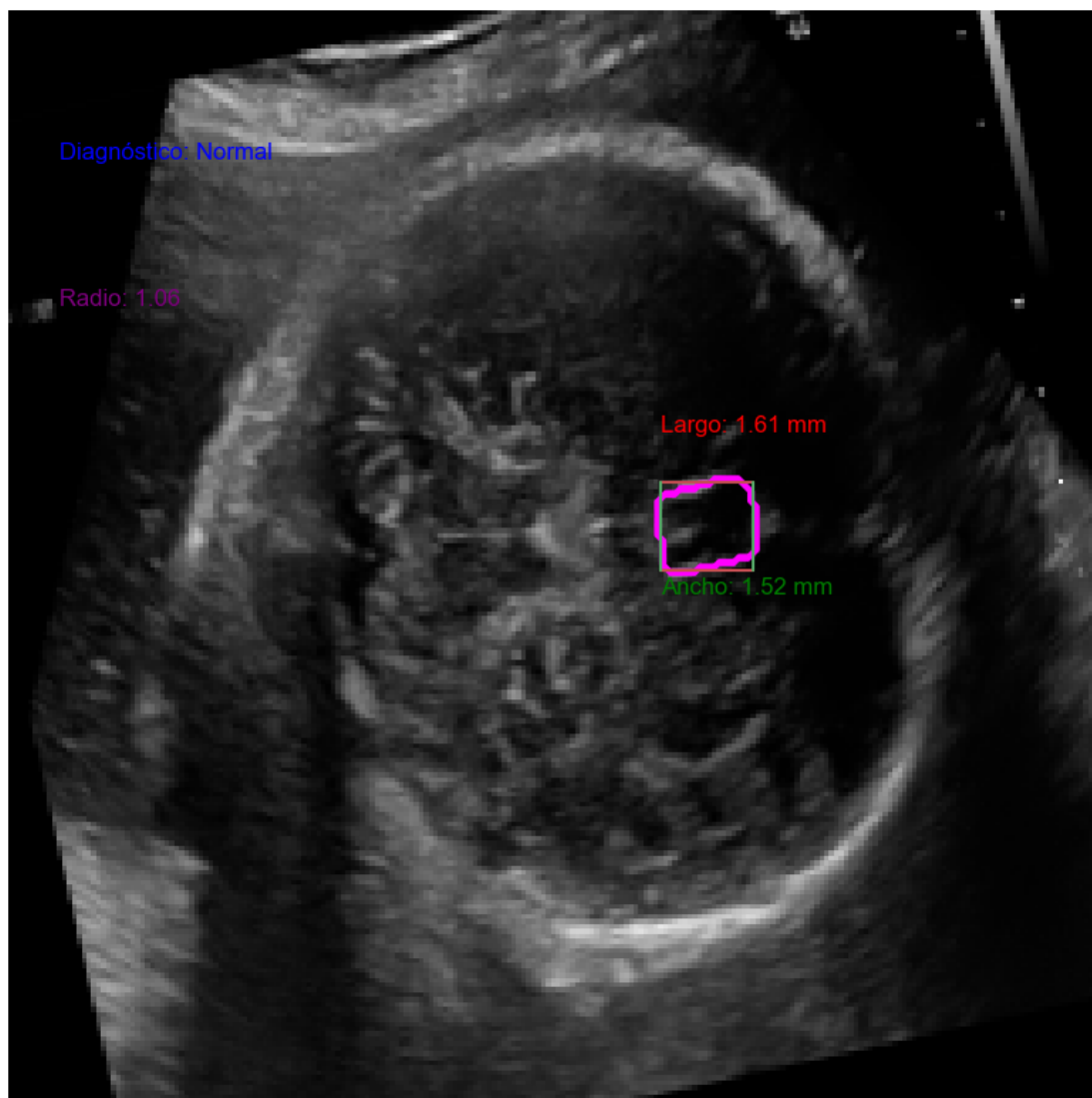


Imagen Patient00706_Plane3_5_of_5_6_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00706_Plane3_5_of_5_6_con_mascara.png

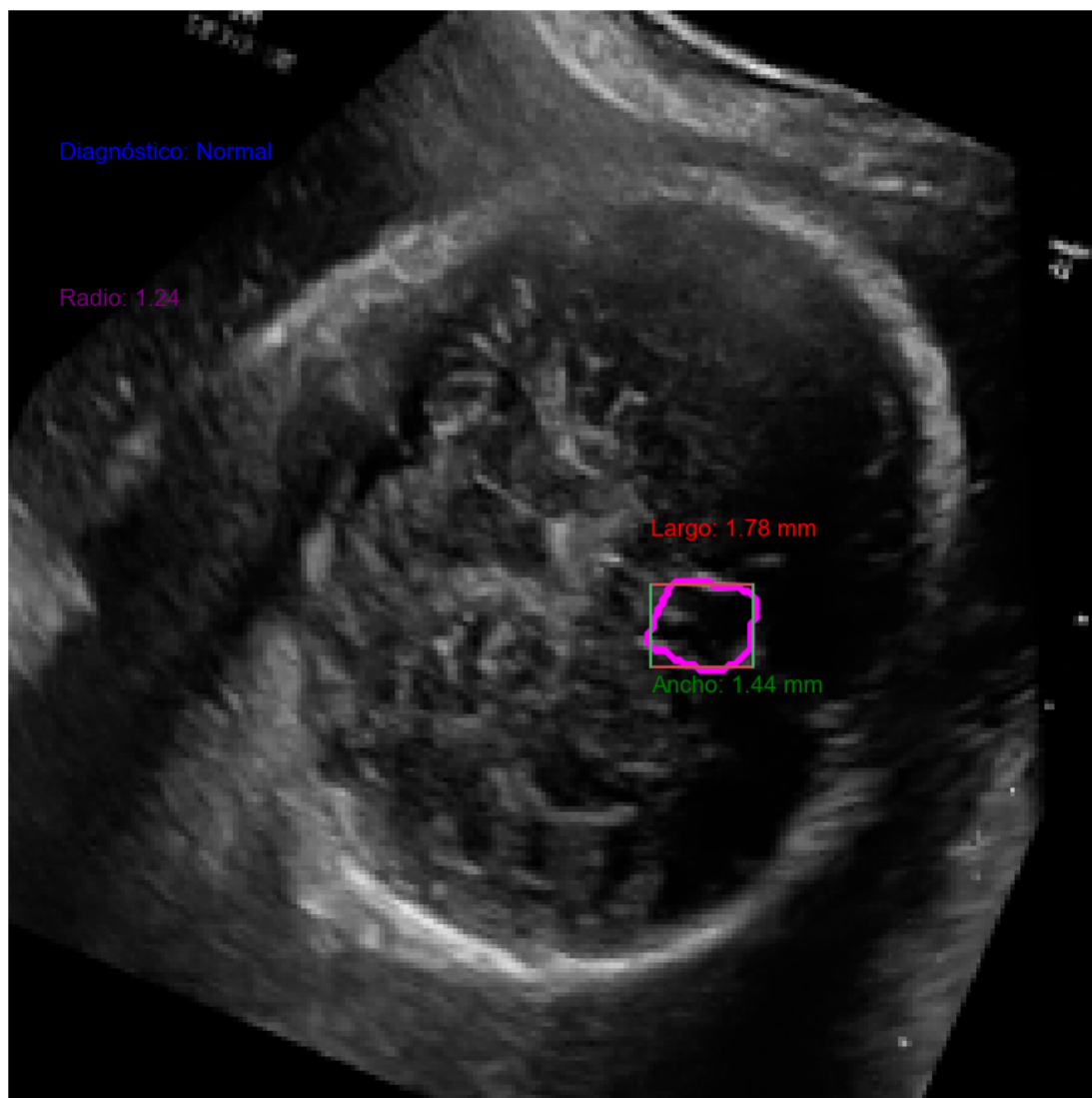


Imagen Patient00706_Plane3_5_of_5_8_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00706_Plane3_5_of_5_8_con_mascara.png

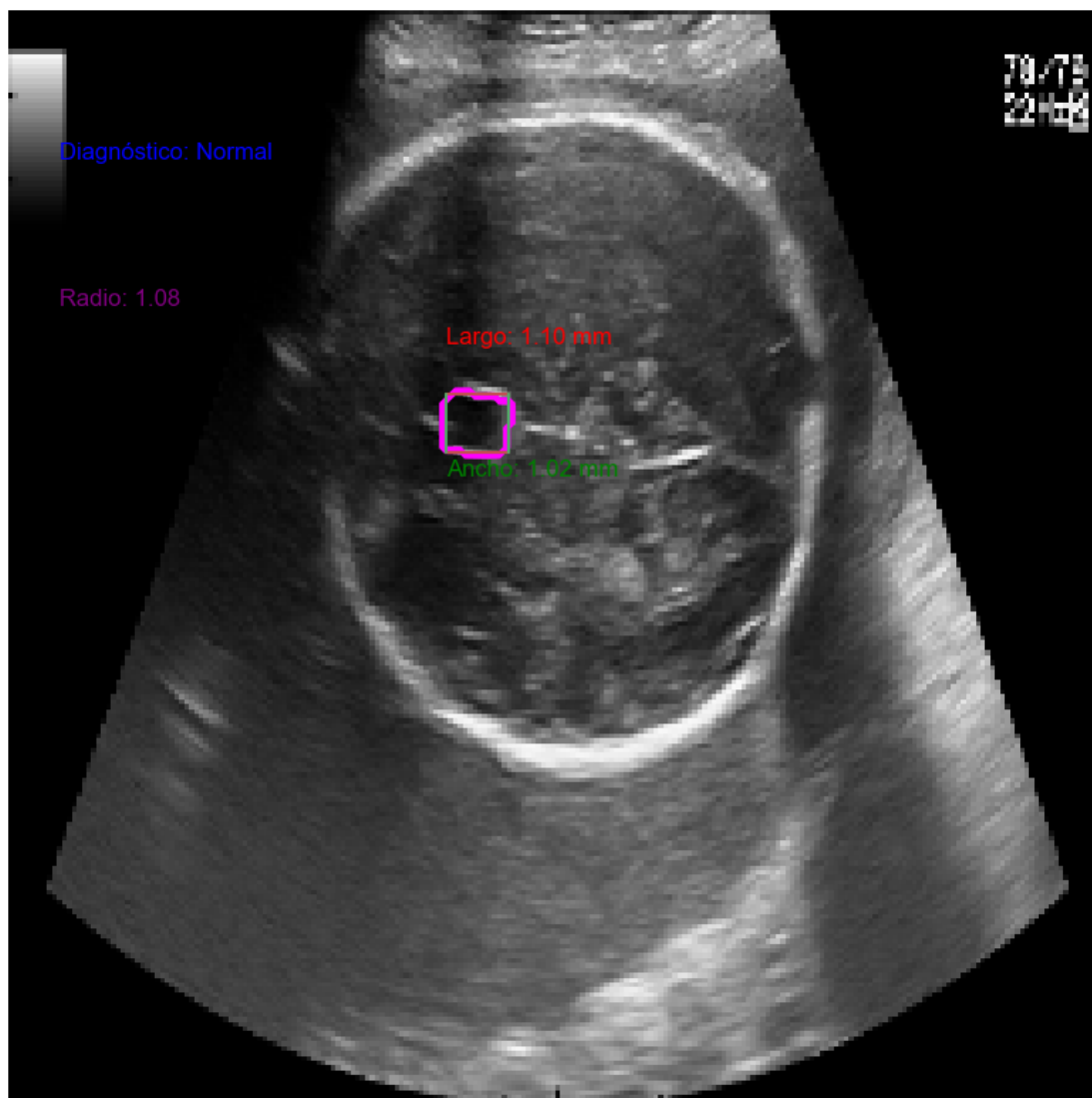


Imagen Patient00168_Plane3_2_of_3_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00168_Plane3_2_of_3_con_mascara.png

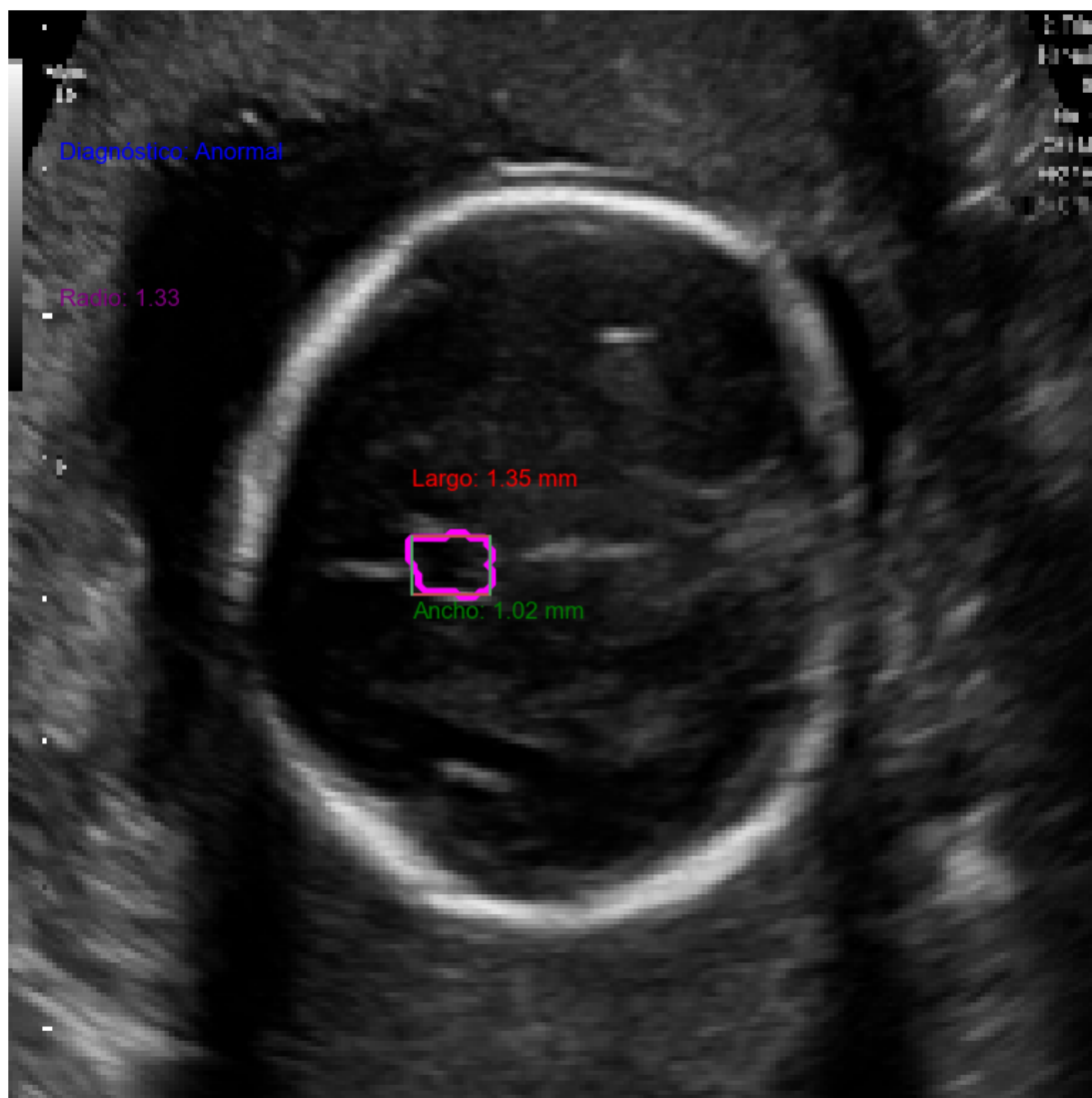


Imagen Patient00216_Plane3_1_of_5_3_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00216_Plane3_1_of_5_3_con_mascara.png

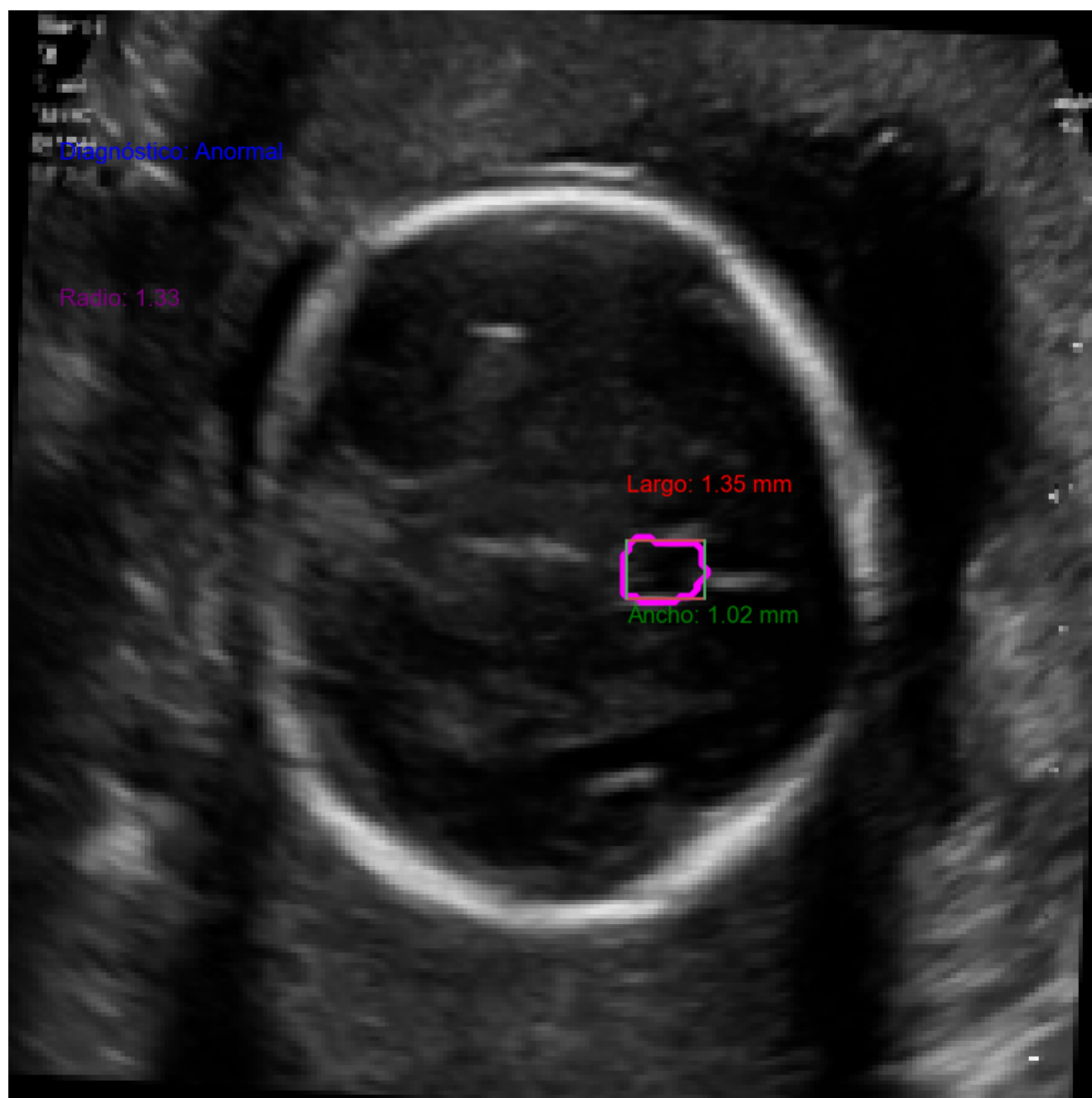


Imagen Patient00216_Plane3_1_of_5_5_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00216_Plane3_1_of_5_5_con_mascara.png

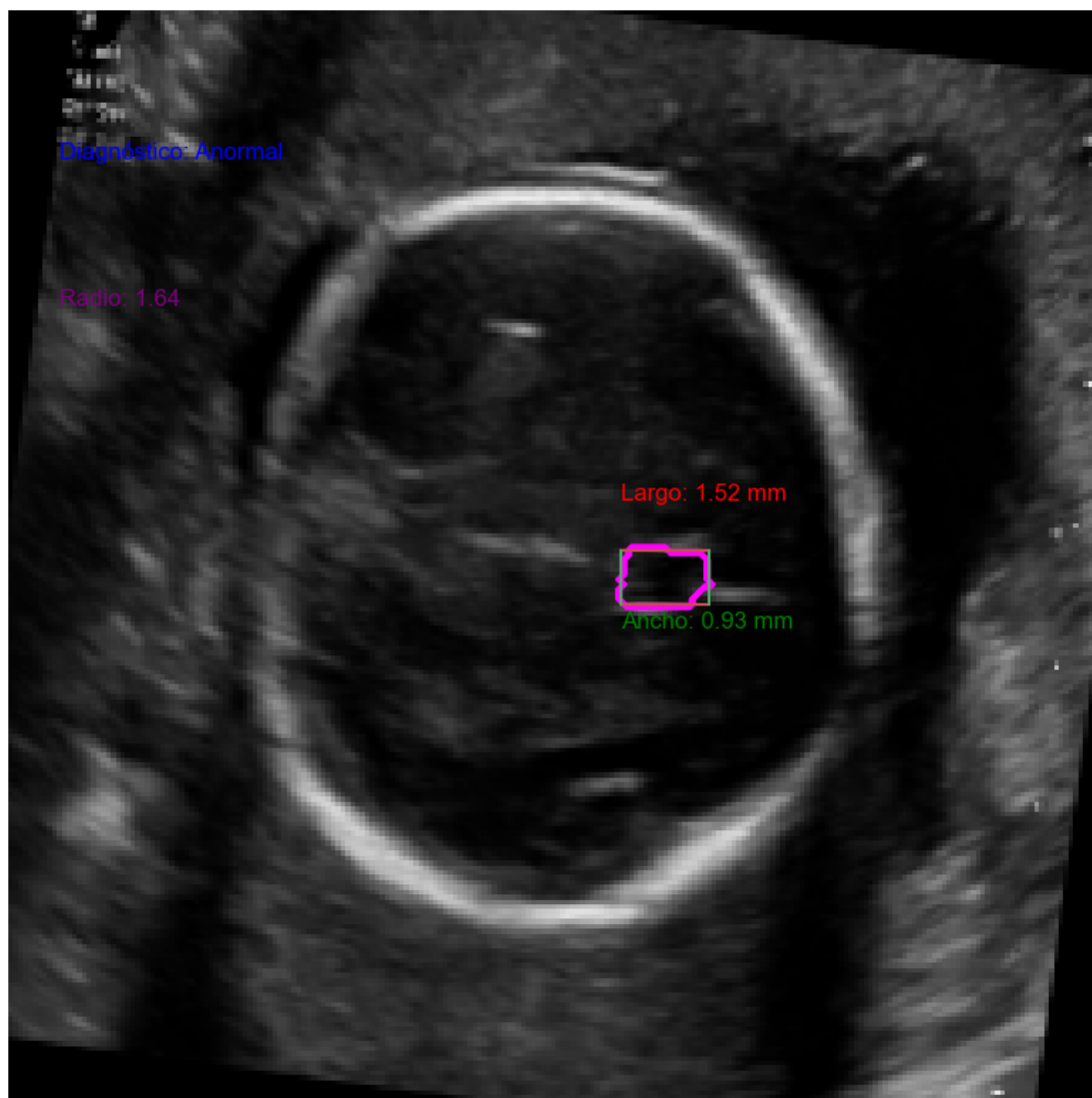


Imagen Patient00216_Plane3_1_of_5_6_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00216_Plane3_1_of_5_6_con_mascara.png

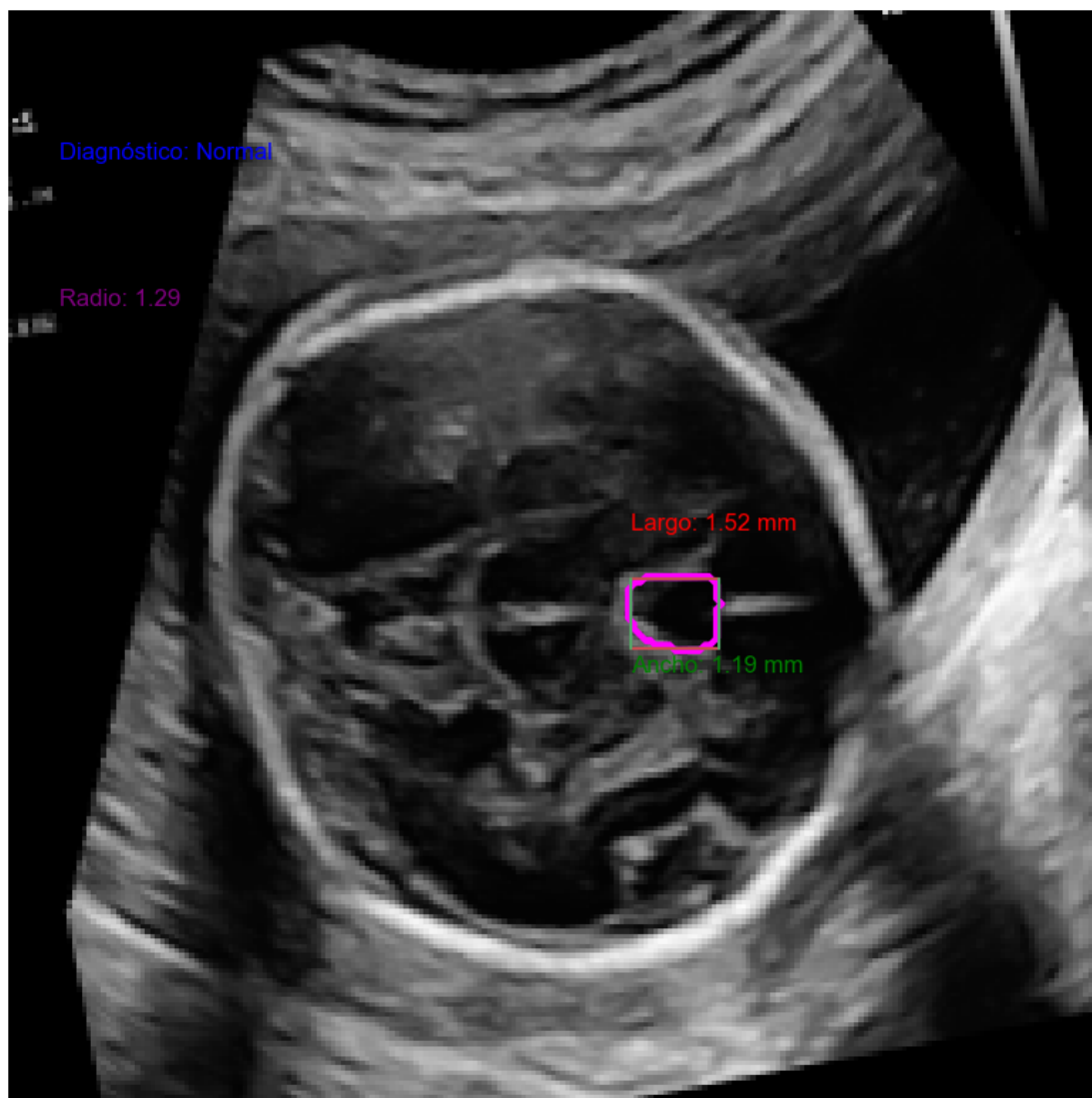


Imagen Patient00305_Plane3_5_of_5_2_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00305_Plane3_5_of_5_2_con_mascara.png

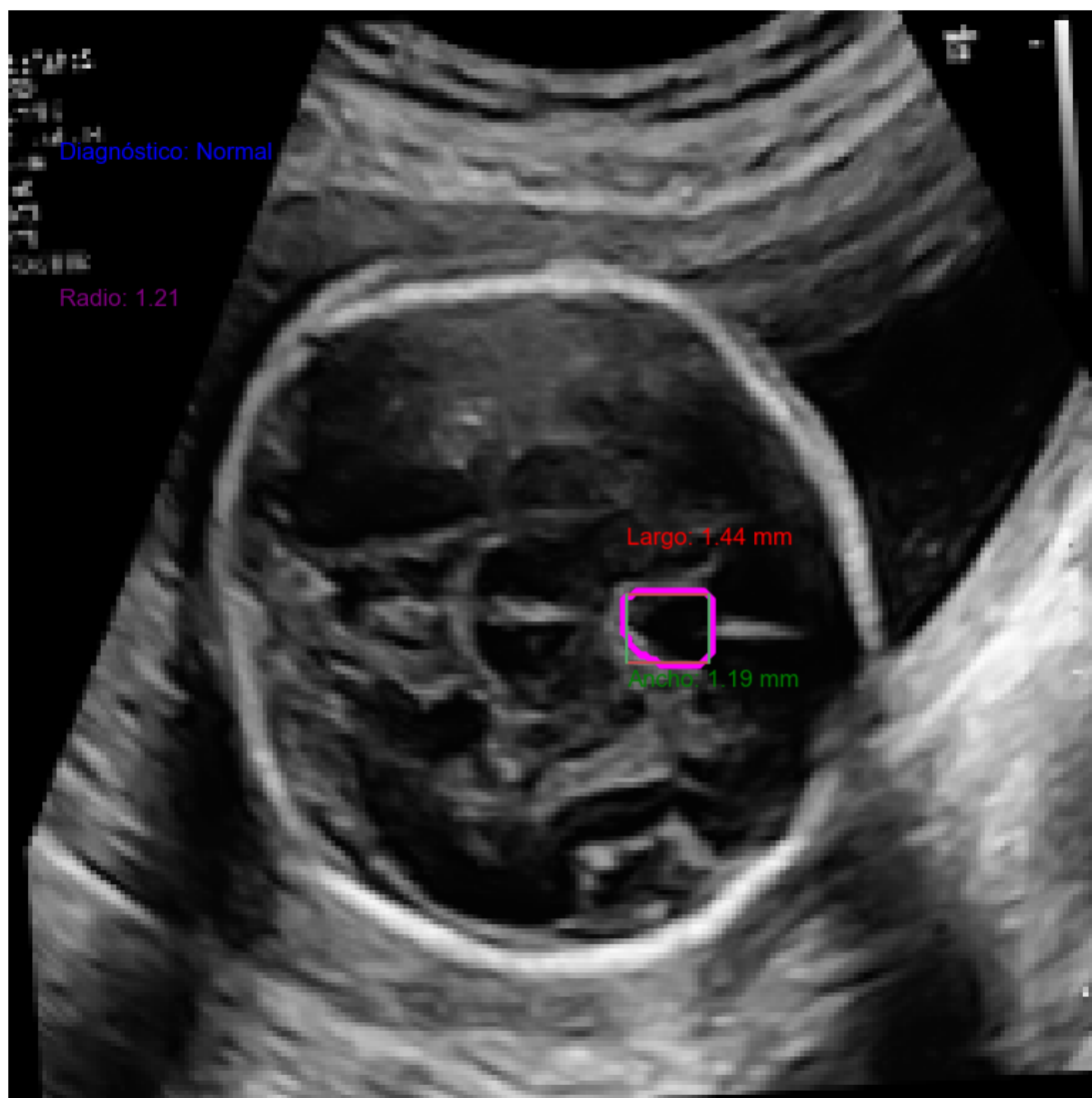


Imagen Patient00305_Plane3_5_of_5_3_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00305_Plane3_5_of_5_3_con_mascara.png

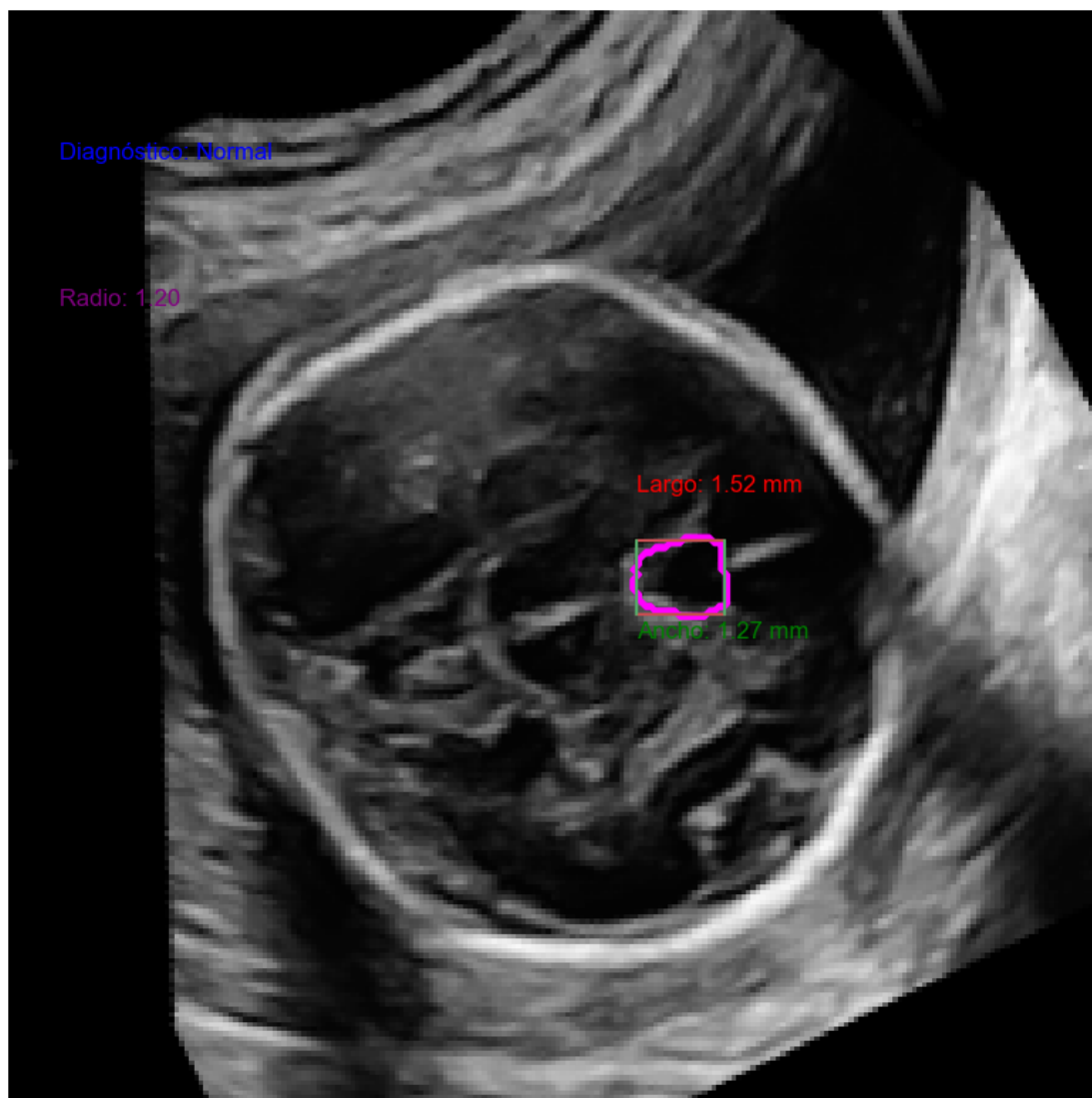


Imagen Patient00305_Plane3_5_of_5_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00305_Plane3_5_of_5_con_mascara.png

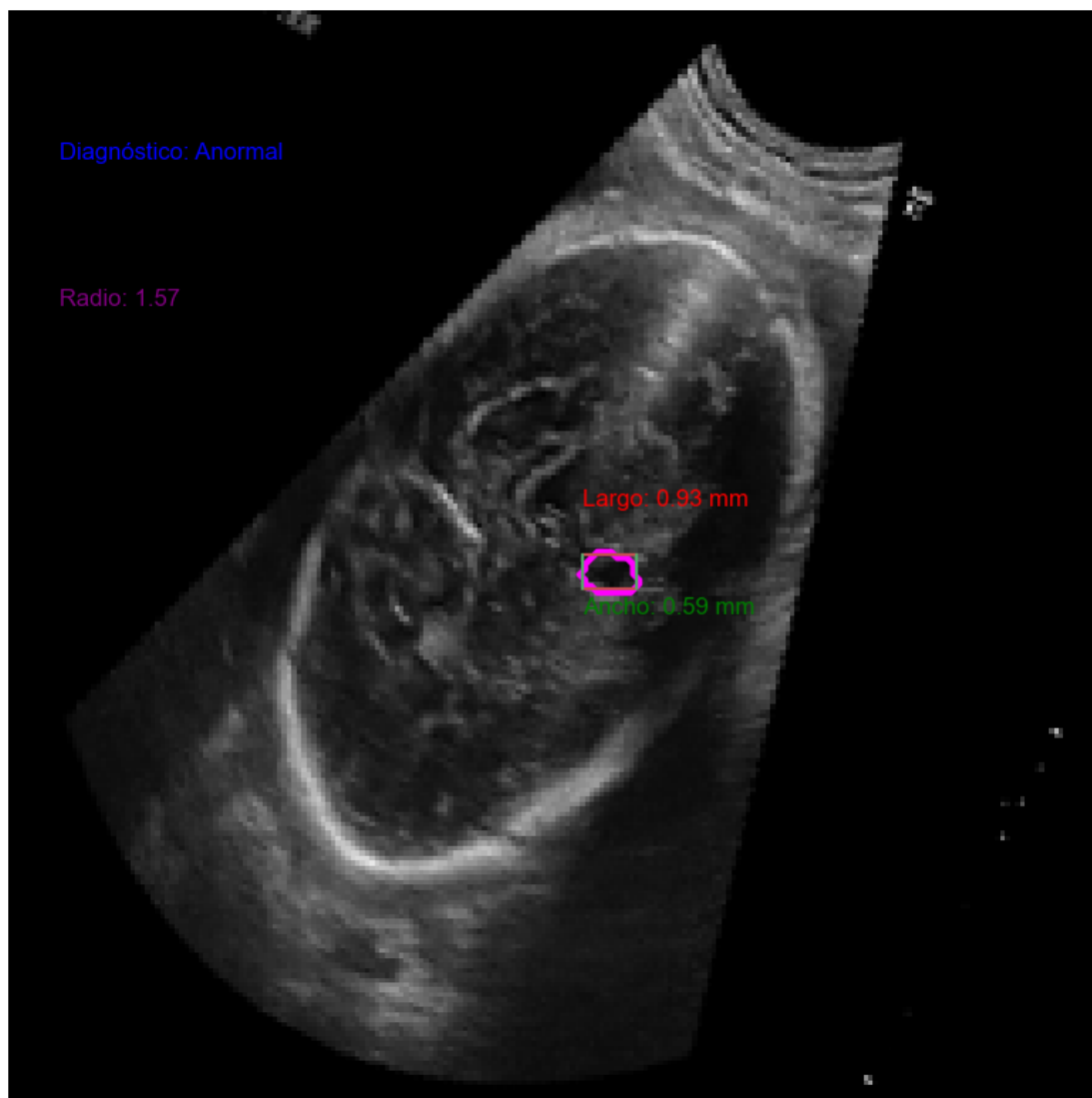


Imagen Patient00640_Plane3_1_of_1_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00640_Plane3_1_of_1_con_mascara.png

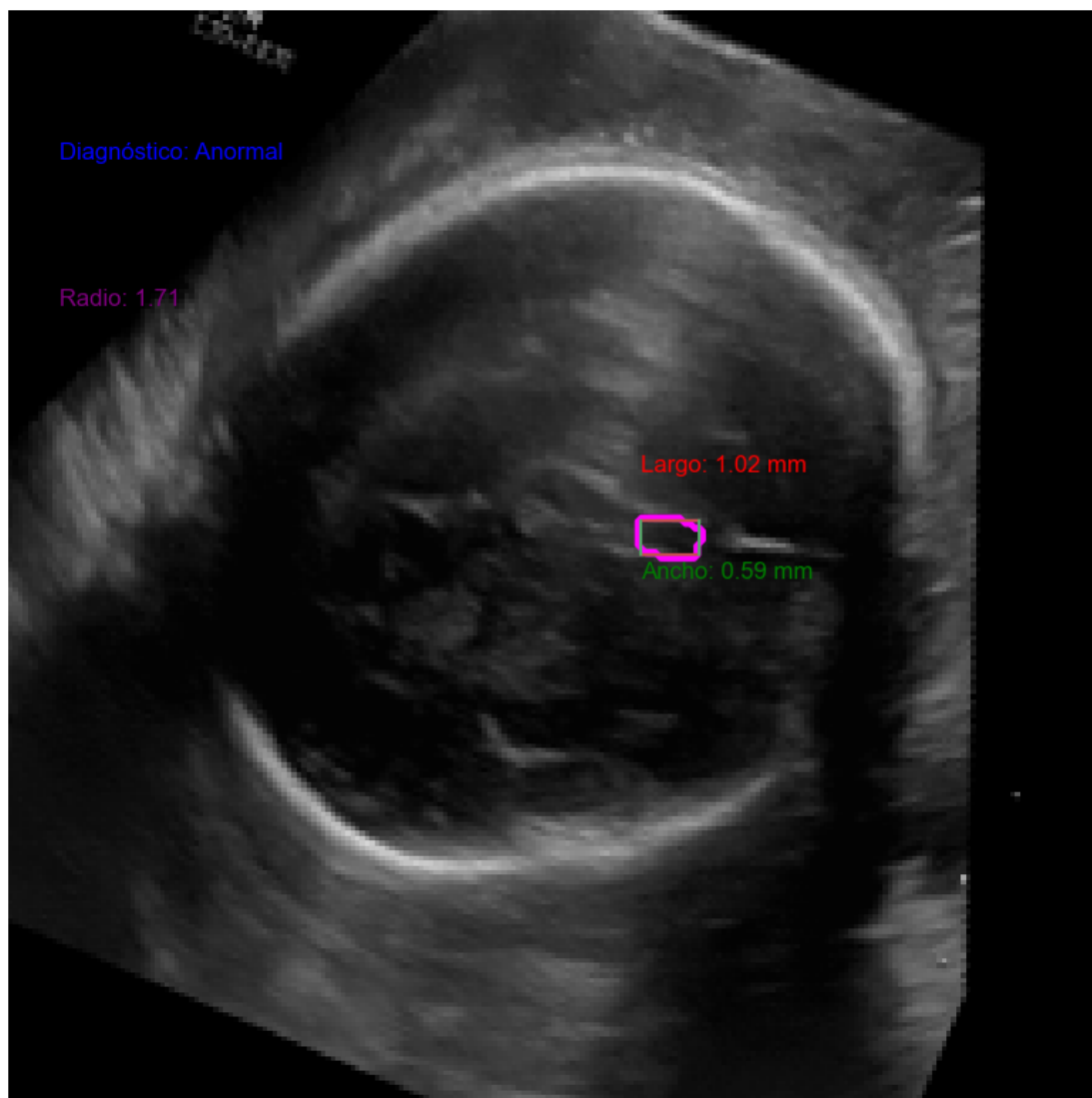


Imagen Patient00647_Plane3_1_of_1_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00647_Plane3_1_of_1_con_mascara.png

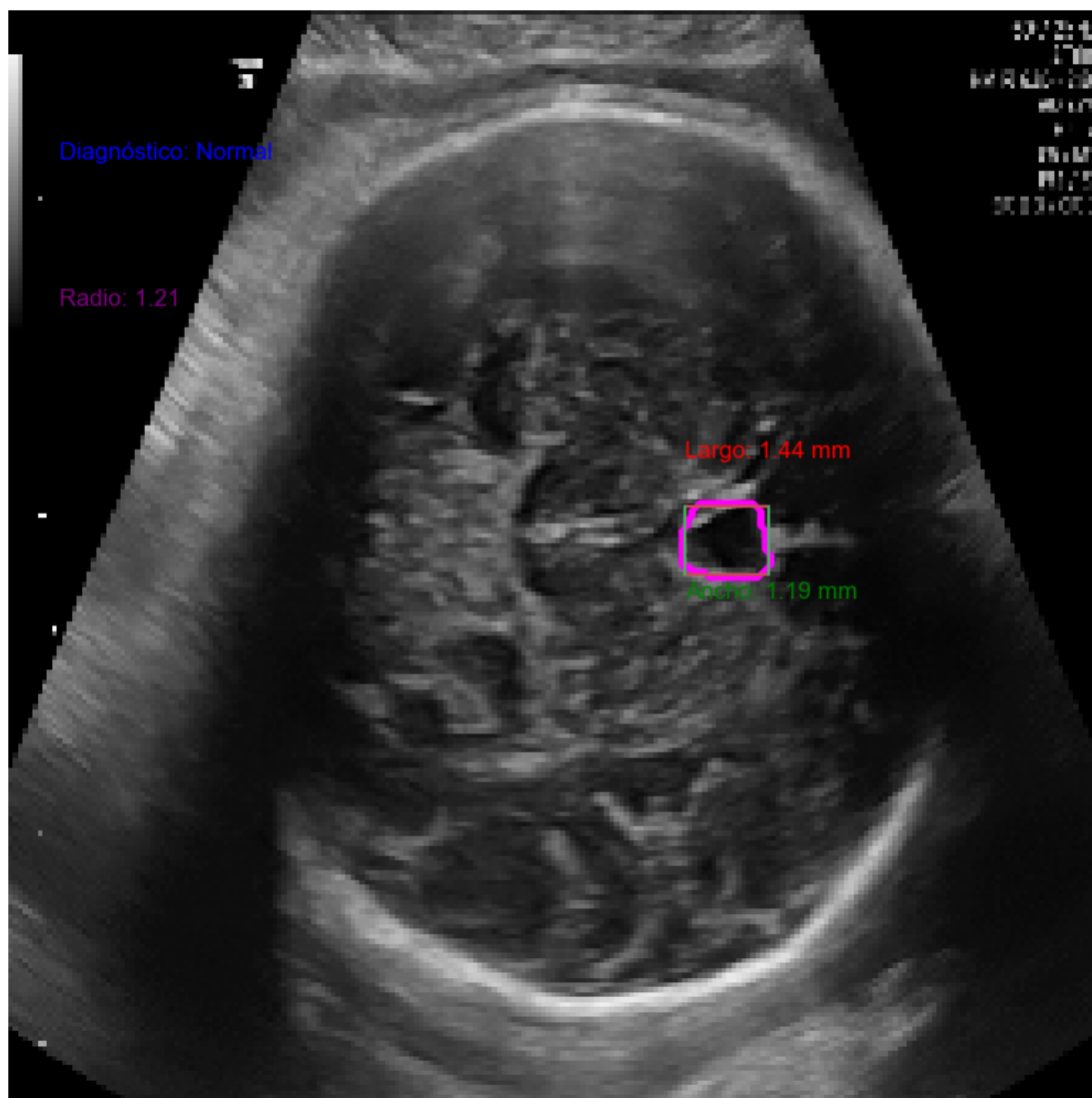


Imagen Patient00658_Plane3_1_of_1_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00658_Plane3_1_of_1_con_mascara.png

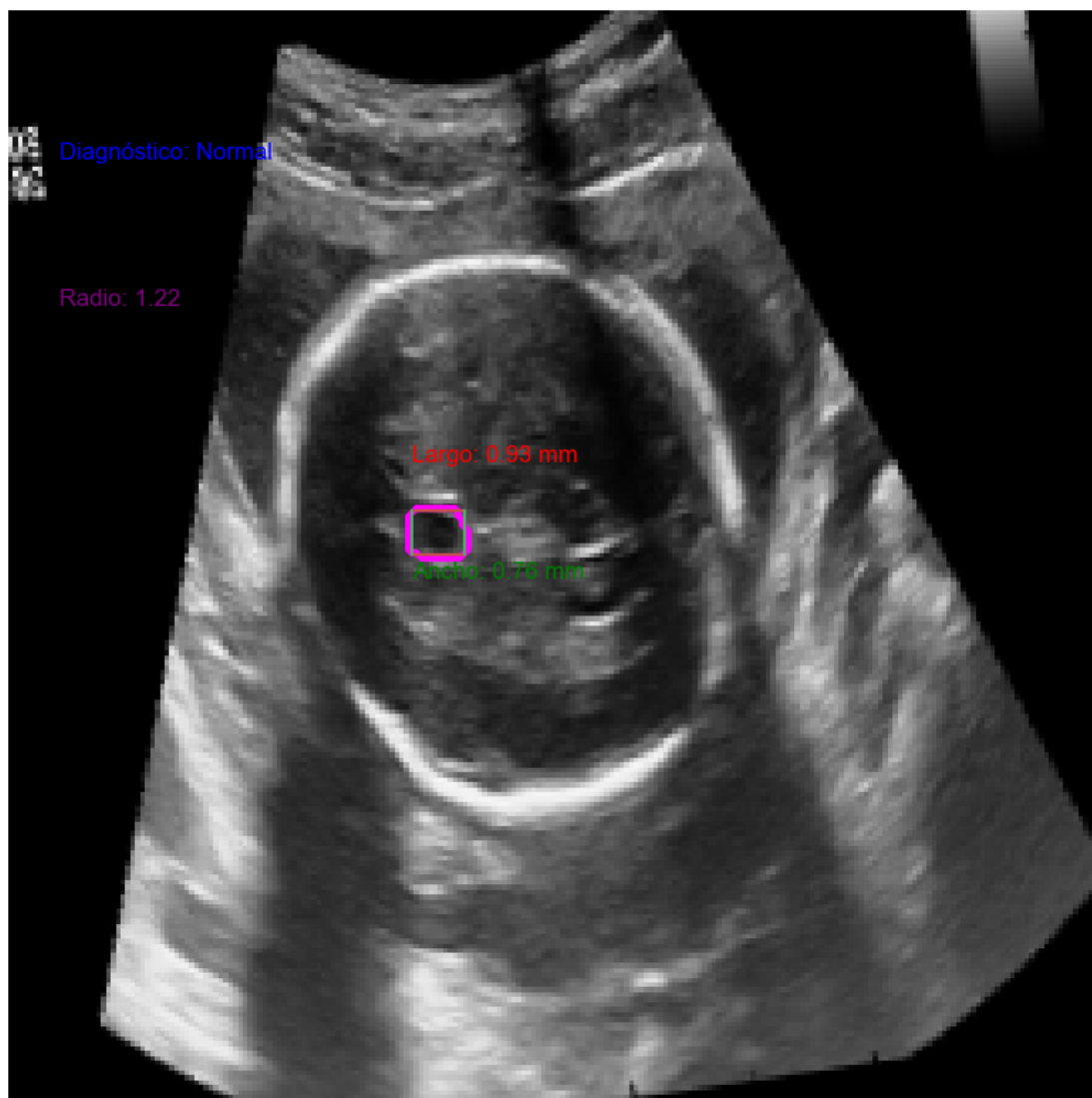


Imagen Patient00188_Plane3_2_of_3_6_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00188_Plane3_2_of_3_6_con_mascara.png

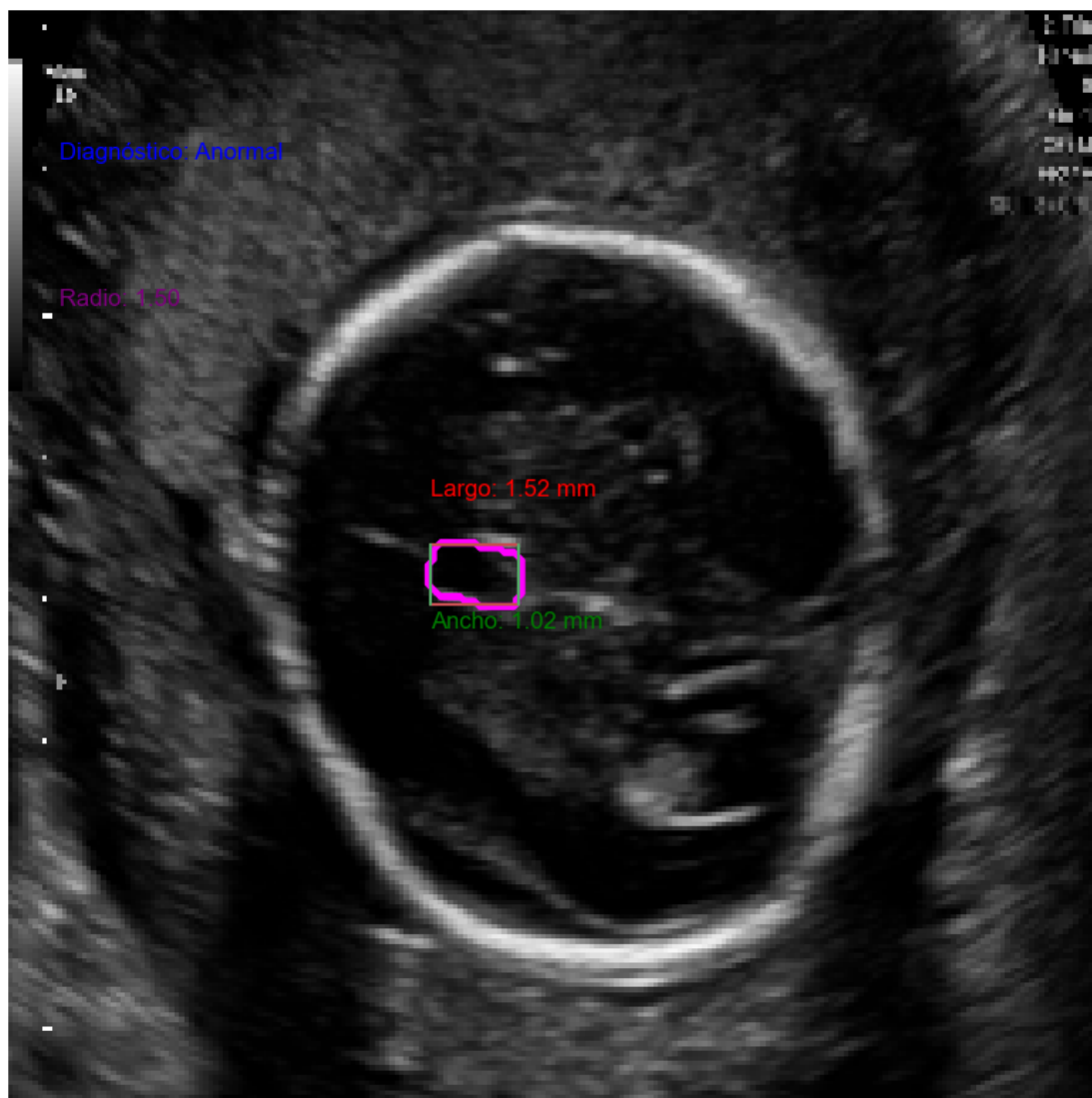


Imagen Patient00216_Plane3_2_of_5_2_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00216_Plane3_2_of_5_2_con_mascara.png

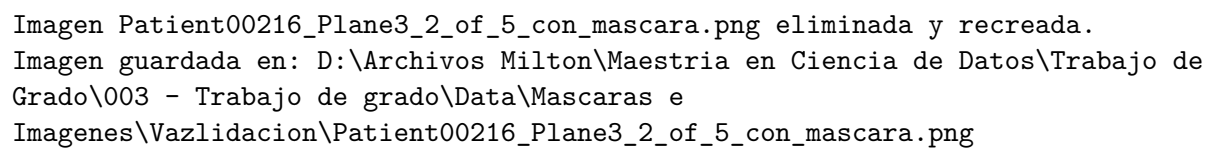


Imagen sin Máscara Predicha

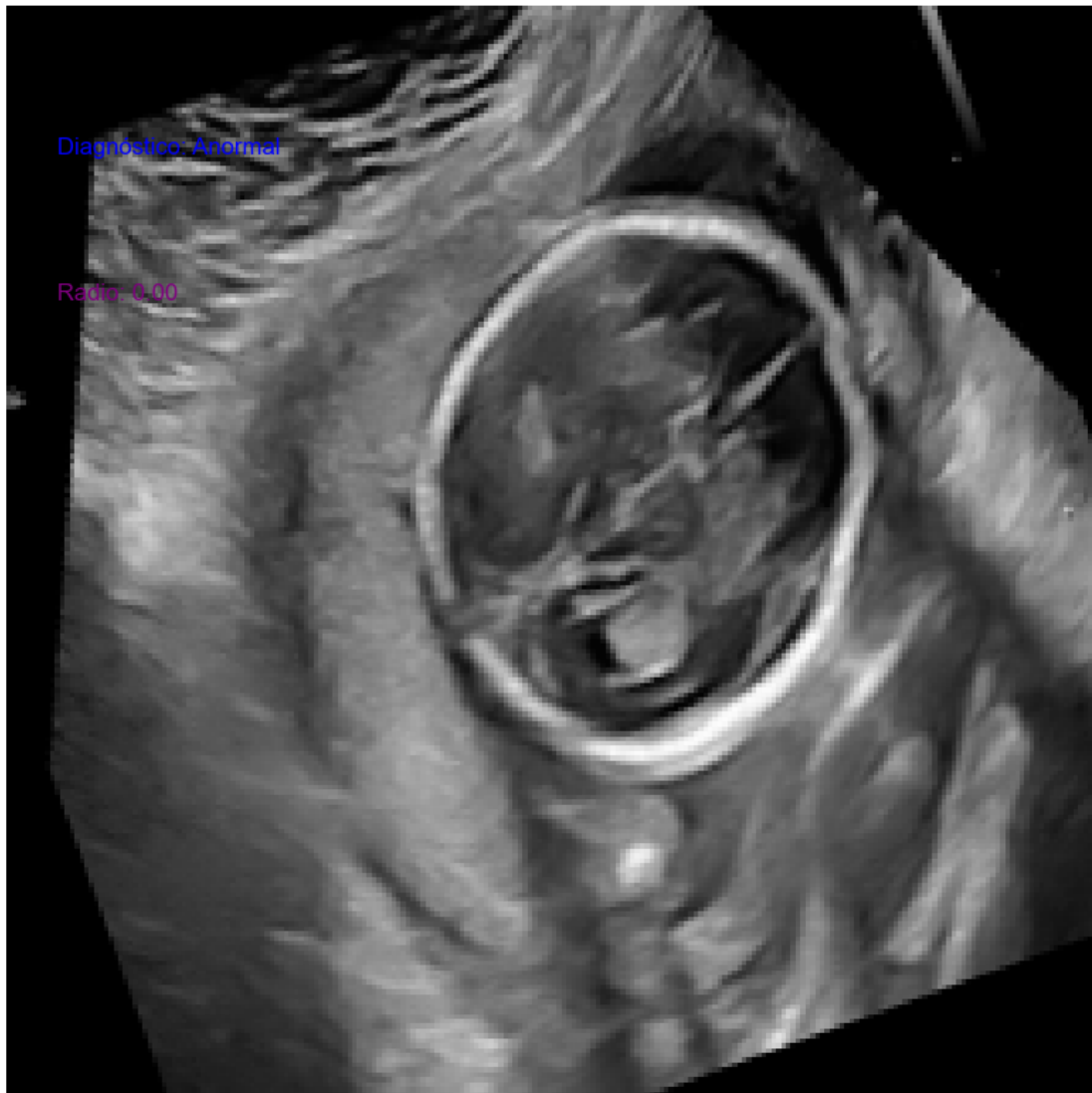


Imagen Patient00305_Plane3_1_of_5_2_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00305_Plane3_1_of_5_2_con_mascara.png

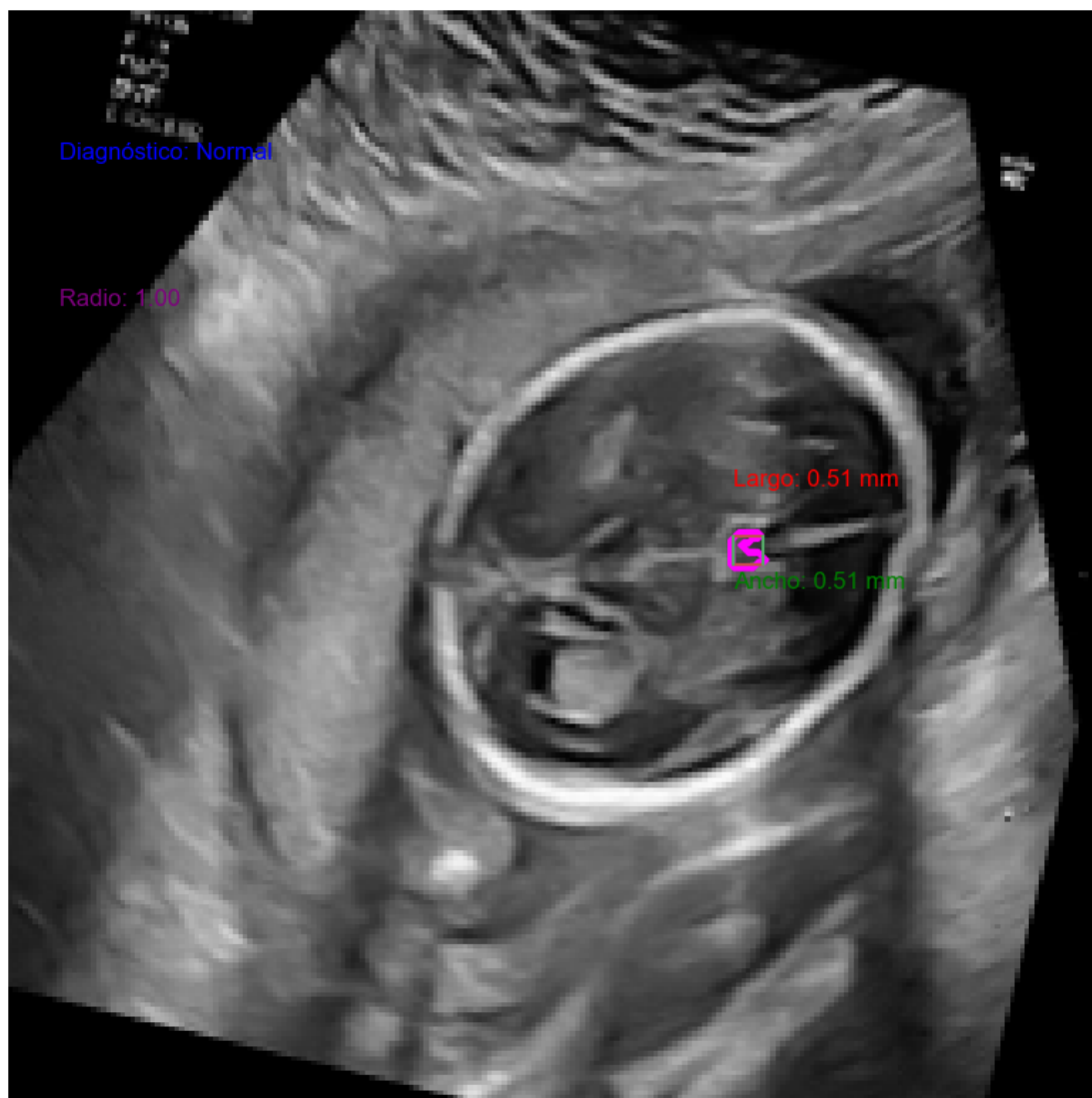


Imagen Patient00305_Plane3_1_of_5_4_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00305_Plane3_1_of_5_4_con_mascara.png

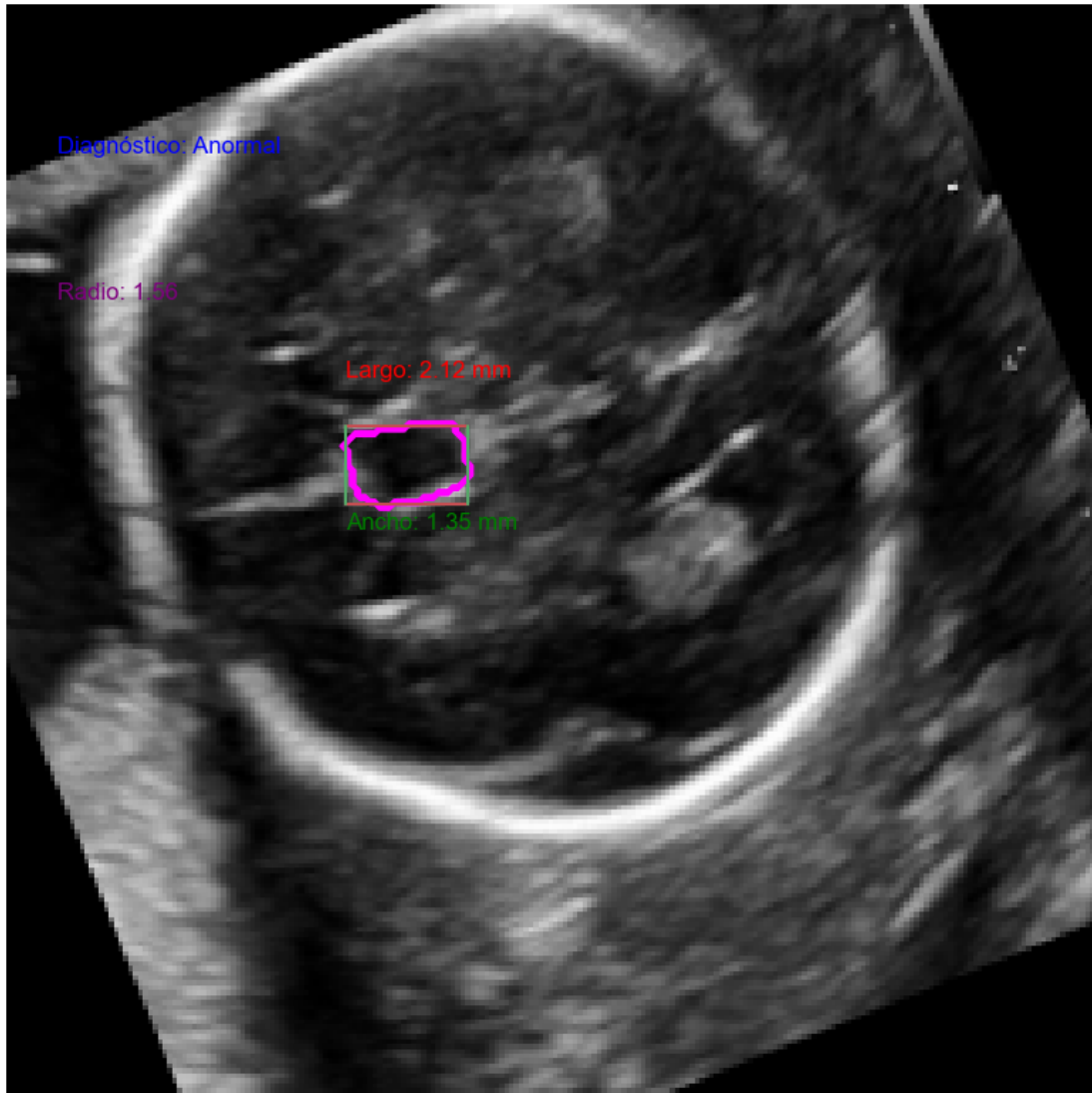


Imagen Patient00644_Plane3_1_of_3_2_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00644_Plane3_1_of_3_2_con_mascara.png

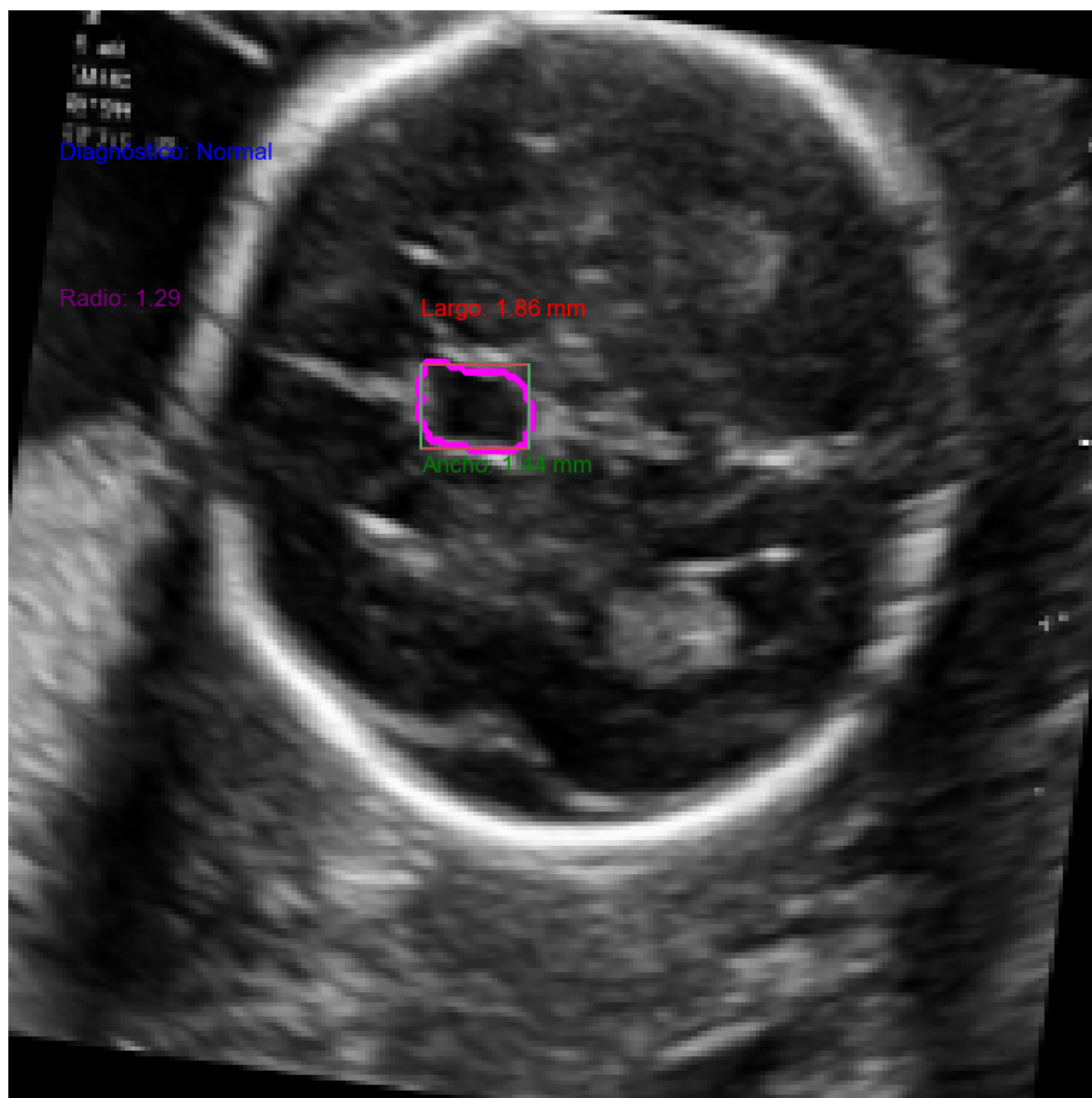


Imagen Patient00644_Plane3_1_of_3_5_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00644_Plane3_1_of_3_5_con_mascara.png

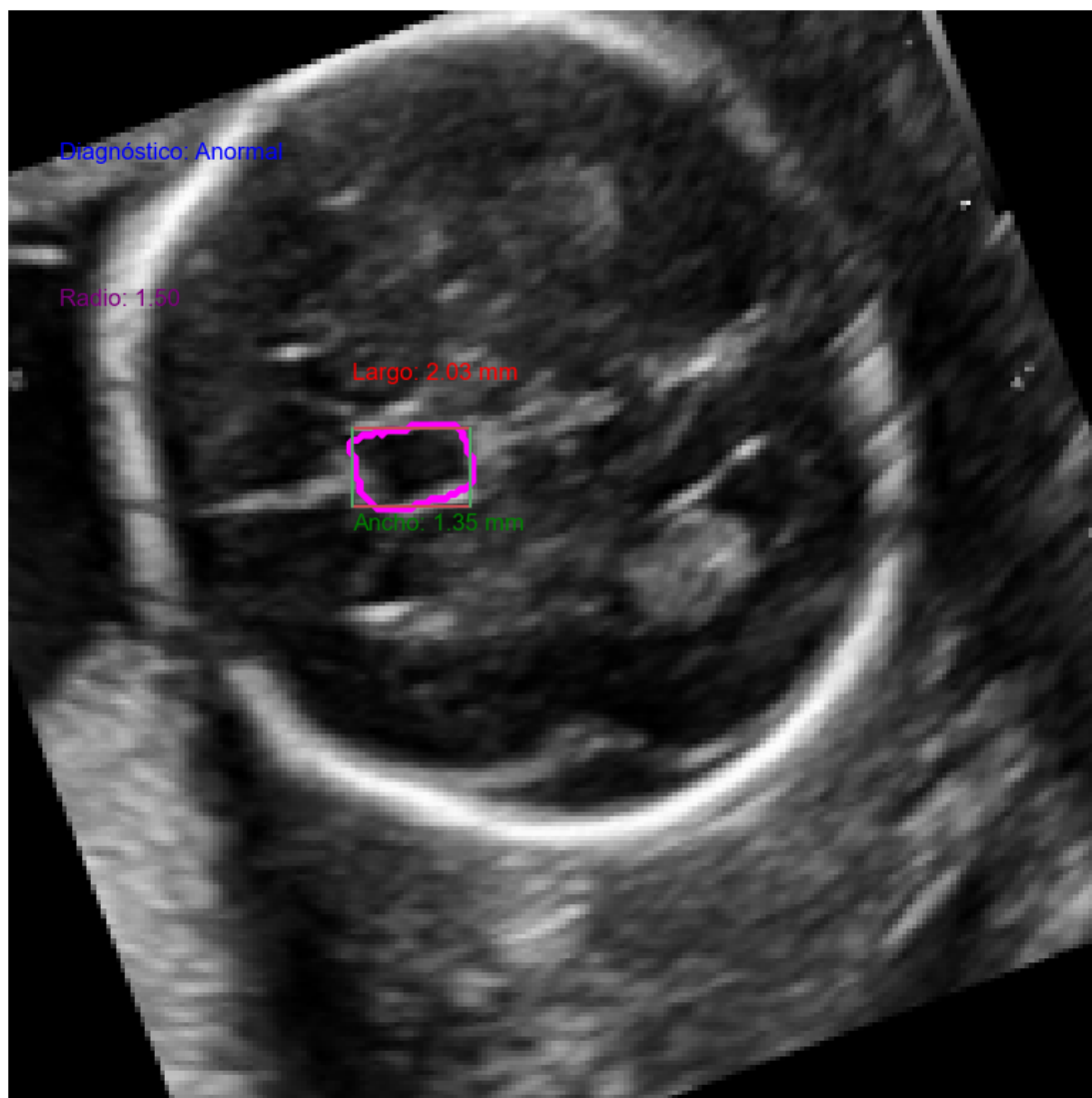


Imagen Patient00644_Plane3_1_of_3_8_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00644_Plane3_1_of_3_8_con_mascara.png

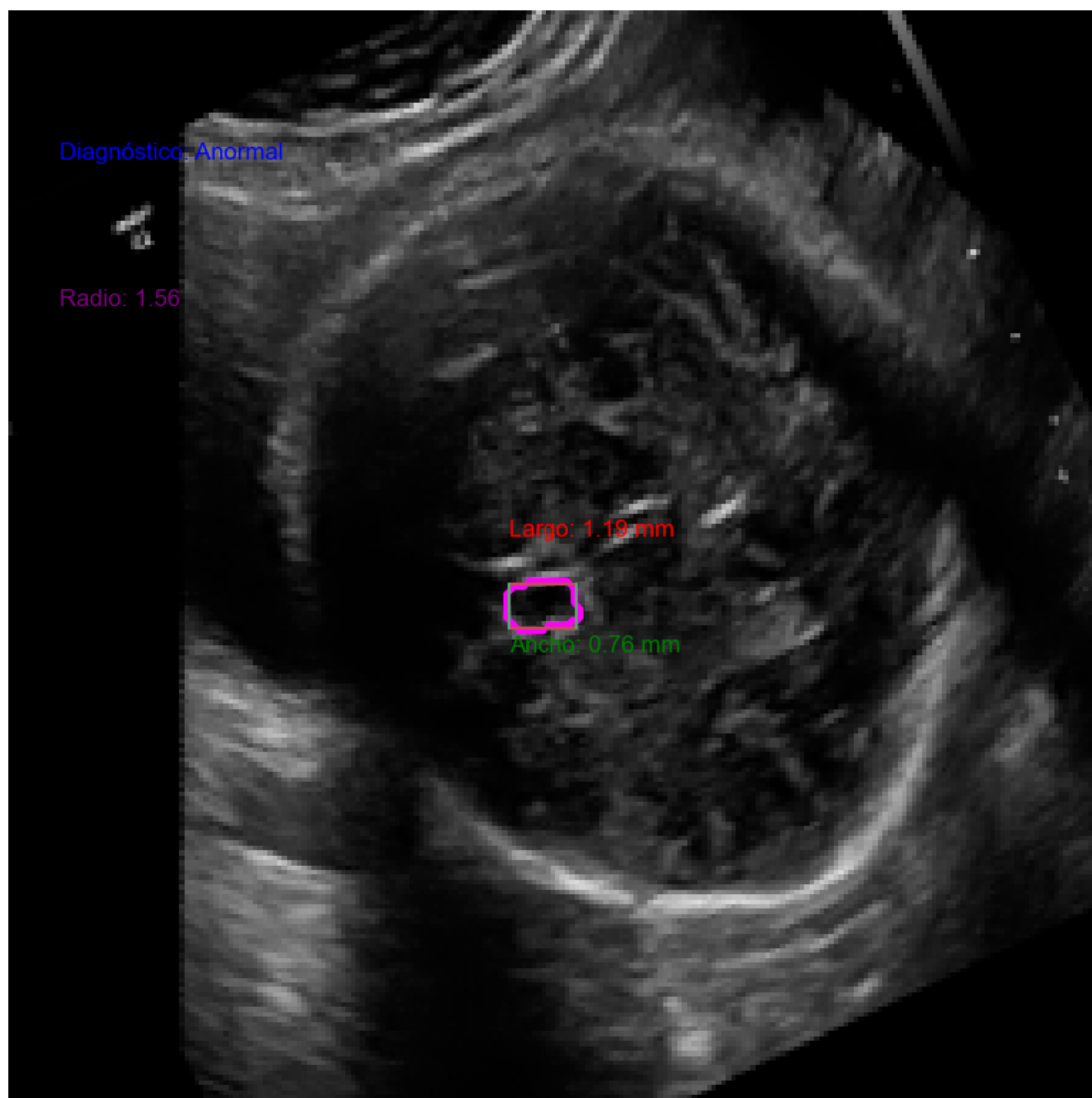


Imagen Patient00688_Plane3_1_of_1_7_con_mascara.png eliminada y recreada.
Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
Grado\003 - Trabajo de grado\Data\Mascaras e
Imagenes\Vazlidacion\Patient00688_Plane3_1_of_1_7_con_mascara.png

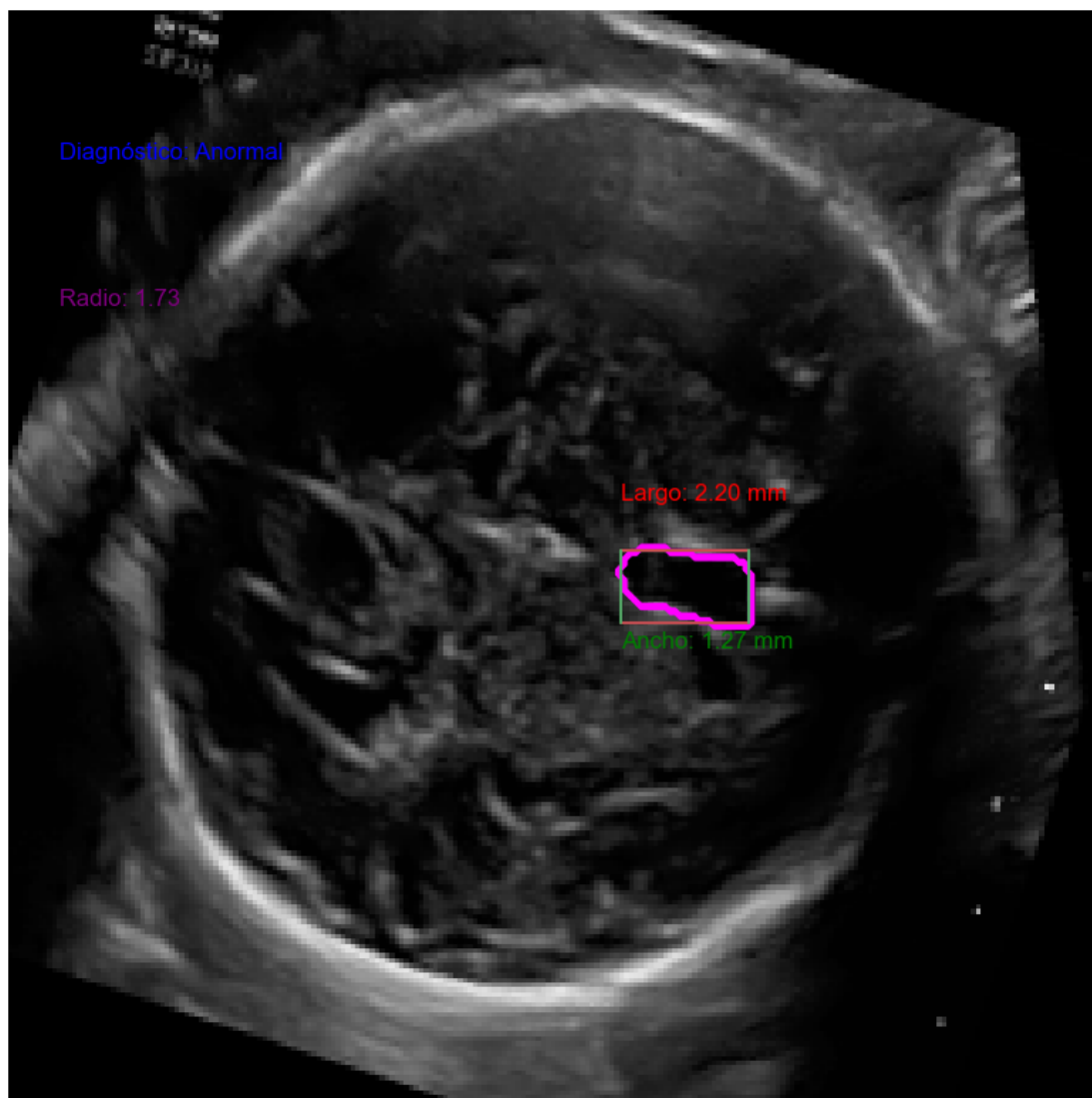


Imagen Patient00700_Plane3_2_of_2_3_con_mascara.png eliminada y recreada.
 Imagen guardada en: D:\Archivos Milton\Maestria en Ciencia de Datos\Trabajo de
 Grado\003 - Trabajo de grado\Data\Mascaras e
 Imagenes\Vazlidacion\Patient00700_Plane3_2_of_2_3_con_mascara.png

5 4.6. Determinación de Anormalidad

Luego de realizado el proceso de determinación y teniendo en cuenta que se realizó una revisión sistemática del tema se determina que para nuestro caso podrían existir dos formar para calcular la normalidad o anormalidad de la estructura:

1. en el primer caso lo tomaremos por el tamaño de la estructura, esta decisión se toma debido a que las imágenes sufrieron un proceso de redimensionamiento para ser tomadas en cuenta

por los modelos, lo que conlleva a que ninguno de los rangos establecidos concuerde con las medidas actuales que tenemos. Teniendo en cuenta lo anterior hemos decidido determinar una normalidad en caso de que el largo de la estructura no sea un 30% mayor al ancho de la misma. la desicion de tomar el 30% es debido a que el modelo cuenta con porcentajes de exactitud de un poco mas del 70%.

2. por otro lado al momento de presentar las imagenes a nuestra especialista en perinatologia, nos informa que en la mayoria de los casos revizados las imagenes correspondieron a casos sin anomalías por lo que podriamos tomar la media de las medidas como un tamaño estandar para una anomalía y todo lo que este por fuera de este valor seria considerado anormal.

```
[40]: # Renombrar la columna "Carpeta" a "Plano"
df_resultados.rename(columns={'Carpeta': 'Plano_cerebral'}, inplace=True)

# Cálculos estadísticos por cada categoría de "Plano"
estadisticos_por_plano = {}
for columna in ['Largo_Geometrico_mm', 'Ancho_Geometrico_mm']:
    estadisticos_por_plano[columna] = df_resultados.
    ↳groupby('Plano_cerebral')[columna].agg(
        Media='mean',
        Mediana='median',
        Moda=lambda x: x.mode().tolist(),
        Desviacion_Estandar='std',
        Varianza='var'
    )

# Mostrar resultados
for columna, stats in estadisticos_por_plano.items():
    print(f"\nEstadísticos para {columna}:")
    print(stats)
```

Estadísticos para Largo_Geometrico_mm:

	Media	Mediana	Moda \
Plano_cerebral			
Trans-Cerebellum	1.659467	1.693333	[1.6933333333333331]
Trans-Thalamic	1.320800	1.397000	[1.5239999999999998]
Trans-Ventricular	1.388533	1.524000	[1.5239999999999998]

	Desviacion_Estandar	Varianza
Plano_cerebral		
Trans-Cerebellum	0.127470	0.016248
Trans-Thalamic	0.222938	0.049701
Trans-Ventricular	0.731821	0.535562

Estadísticos para Ancho_Geometrico_mm:

Media	Mediana \
-------	-----------

Plano_cerebral		
Trans-Cerebellum	1.371600	1.354667
Trans-Thalamic	0.999067	1.016000
Trans-Ventricular	0.948267	1.016000

Moda \

Plano_cerebral	
Trans-Cerebellum	[1.27, 1.3546666666666667]
Trans-Thalamic	[1.016, 1.1853333333333333]
Trans-Ventricular	[0.7619999999999999, 1.016, 1.3546666666666667]

	Desviacion_Estandar	Varianza
Plano_cerebral		
Trans-Cerebellum	0.148267	0.021983
Trans-Thalamic	0.238807	0.057029
Trans-Ventricular	0.452964	0.205177

Ahora generamos el diagnostico teniendo en cuenta la media y la desviación estándar por cada plano Cerebral

```
[41]: # Calcular estadísticas para cada plano
estadisticas = df_resultados.groupby('Plano_cerebral').agg(
    Media_Largo=('Largo_Geometrico_mm', 'mean'),
    Std_Largo=('Largo_Geometrico_mm', 'std'),
    Media_Ancho=('Ancho_Geometrico_mm', 'mean'),
    Std_Ancho=('Ancho_Geometrico_mm', 'std')
).reset_index()

# Verificar las estadísticas generadas
print("Estadísticas calculadas:")
print(estadisticas)

# Limpiar columnas previas relacionadas con las estadísticas
columnas_estadisticas = ['Media_Largo', 'Std_Largo', 'Media_Ancho', 'Std_Ancho']
df_resultados = df_resultados.drop(columns=[col for col in
    ↪columnas_estadisticas if col in df_resultados.columns], errors='ignore')

# Unir estadísticas al DataFrame original
df_resultados = pd.merge(df_resultados, estadisticas, on='Plano_cerebral')

# Función para determinar el diagnóstico
def diagnostico(row):
    # Calcular rangos normales para Largo
    rango_largo_min = row['Media_Largo'] - row['Std_Largo']
    rango_largo_max = row['Media_Largo'] + row['Std_Largo']
    es_normal_largo = rango_largo_min <= row['Largo_Geometrico_mm'] <=
    ↪rango_largo_max
```

```

# Calcular rangos normales para Ancho
rango_ancho_min = row['Media_Ancho'] - row['Std_Ancho']
rango_ancho_max = row['Media_Ancho'] + row['Std_Ancho']
es_normal_ancho = rango_ancho_min <= row['Ancho_Geometrico_mm'] <=
↪rango_ancho_max

# Diagnóstico final
if es_normal_largo and es_normal_ancho:
    return 'Normal'
else:
    return 'Anormal'

# Crear la nueva columna con el diagnóstico
df_resultados['Diagnostico_por_estandares'] = df_resultados.apply(diagnostico,
↪axis=1)

# Mostrar el DataFrame resultante
print("\nDataFrame con diagnóstico:")

df_resultados[['Base_Nombre', 'Plano_cerebral', 'Largo_Geometrico_mm',
↪'Ancho_Geometrico_mm', 'Diagnostico_por_estandares', 'diagnostico_tamaño',
↪'Media_Largo', 'Std_Largo', 'Media_Ancho', 'Std_Ancho']]

```

Estadísticas calculadas:

	Plano_cerebral	Media_Largo	Std_Largo	Media_Ancho	Std_Ancho
0	Trans-Cerebellum	1.659467	0.127470	1.371600	0.148267
1	Trans-Thalamic	1.320800	0.222938	0.999067	0.238807
2	Trans-Ventricular	1.388533	0.731821	0.948267	0.452964

DataFrame con diagnóstico:

```

[41]:
      Base_Nombre      Plano_cerebral  Largo_Geometrico_mm \
0  Patient00644_Plane3_2_of_3_5  Trans-Cerebellum      1.693333
1  Patient00662_Plane3_1_of_1_2  Trans-Cerebellum      1.439333
2   Patient00662_Plane3_1_of_1  Trans-Cerebellum      1.439333
3  Patient00675_Plane3_1_of_1_2  Trans-Cerebellum      1.778000
4  Patient00675_Plane3_1_of_1_3  Trans-Cerebellum      1.778000
5  Patient00687_Plane3_1_of_1_8  Trans-Cerebellum      1.693333
6  Patient00706_Plane3_5_of_5_2  Trans-Cerebellum      1.693333
7  Patient00706_Plane3_5_of_5_4  Trans-Cerebellum      1.693333
8  Patient00706_Plane3_5_of_5_6  Trans-Cerebellum      1.608667
9  Patient00706_Plane3_5_of_5_8  Trans-Cerebellum      1.778000
10 Patient00168_Plane3_2_of_3     Trans-Thalamic      1.100667
11 Patient00216_Plane3_1_of_5_3     Trans-Thalamic      1.354667
12 Patient00216_Plane3_1_of_5_5     Trans-Thalamic      1.354667
13 Patient00216_Plane3_1_of_5_6     Trans-Thalamic      1.524000
14 Patient00305_Plane3_5_of_5_2     Trans-Thalamic      1.524000

```


15	Patient00305_Plane3_5_of_5_3	Trans-Thalamic	1.439333
16	Patient00305_Plane3_5_of_5	Trans-Thalamic	1.524000
17	Patient00640_Plane3_1_of_1	Trans-Thalamic	0.931333
18	Patient00647_Plane3_1_of_1	Trans-Thalamic	1.016000
19	Patient00658_Plane3_1_of_1	Trans-Thalamic	1.439333
20	Patient00188_Plane3_2_of_3_6	Trans-Ventricular	0.931333
21	Patient00216_Plane3_2_of_5_2	Trans-Ventricular	1.524000
22	Patient00216_Plane3_2_of_5	Trans-Ventricular	1.524000
23	Patient00305_Plane3_1_of_5_2	Trans-Ventricular	0.000000
24	Patient00305_Plane3_1_of_5_4	Trans-Ventricular	0.508000
25	Patient00644_Plane3_1_of_3_2	Trans-Ventricular	2.116667
26	Patient00644_Plane3_1_of_3_5	Trans-Ventricular	1.862667
27	Patient00644_Plane3_1_of_3_8	Trans-Ventricular	2.032000
28	Patient00688_Plane3_1_of_1_7	Trans-Ventricular	1.185333
29	Patient00700_Plane3_2_of_2_3	Trans-Ventricular	2.201333

	Ancho_Geometrico_mm	Diagnostico_por_estandares	diagnostico_tamaño \
0	1.354667	Normal	Normal
1	1.270000	Anormal	Normal
2	1.270000	Anormal	Normal
3	1.270000	Normal	Anormal
4	1.185333	Anormal	Anormal
5	1.693333	Anormal	Normal
6	1.354667	Normal	Normal
7	1.354667	Normal	Normal
8	1.524000	Anormal	Normal
9	1.439333	Normal	Normal
10	1.016000	Normal	Normal
11	1.016000	Normal	Anormal
12	1.016000	Normal	Anormal
13	0.931333	Normal	Anormal
14	1.185333	Normal	Normal
15	1.185333	Normal	Normal
16	1.270000	Anormal	Normal
17	0.592667	Anormal	Anormal
18	0.592667	Anormal	Anormal
19	1.185333	Normal	Normal
20	0.762000	Normal	Normal
21	1.016000	Normal	Anormal
22	1.016000	Normal	Anormal
23	0.000000	Anormal	Anormal
24	0.508000	Anormal	Normal
25	1.354667	Normal	Anormal
26	1.439333	Anormal	Normal
27	1.354667	Normal	Anormal
28	0.762000	Normal	Anormal
29	1.270000	Anormal	Anormal

	Media_Largo	Std_Largo	Media_Ancho	Std_Ancho
0	1.659467	0.127470	1.371600	0.148267
1	1.659467	0.127470	1.371600	0.148267
2	1.659467	0.127470	1.371600	0.148267
3	1.659467	0.127470	1.371600	0.148267
4	1.659467	0.127470	1.371600	0.148267
5	1.659467	0.127470	1.371600	0.148267
6	1.659467	0.127470	1.371600	0.148267
7	1.659467	0.127470	1.371600	0.148267
8	1.659467	0.127470	1.371600	0.148267
9	1.659467	0.127470	1.371600	0.148267
10	1.320800	0.222938	0.999067	0.238807
11	1.320800	0.222938	0.999067	0.238807
12	1.320800	0.222938	0.999067	0.238807
13	1.320800	0.222938	0.999067	0.238807
14	1.320800	0.222938	0.999067	0.238807
15	1.320800	0.222938	0.999067	0.238807
16	1.320800	0.222938	0.999067	0.238807
17	1.320800	0.222938	0.999067	0.238807
18	1.320800	0.222938	0.999067	0.238807
19	1.320800	0.222938	0.999067	0.238807
20	1.388533	0.731821	0.948267	0.452964
21	1.388533	0.731821	0.948267	0.452964
22	1.388533	0.731821	0.948267	0.452964
23	1.388533	0.731821	0.948267	0.452964
24	1.388533	0.731821	0.948267	0.452964
25	1.388533	0.731821	0.948267	0.452964
26	1.388533	0.731821	0.948267	0.452964
27	1.388533	0.731821	0.948267	0.452964
28	1.388533	0.731821	0.948267	0.452964
29	1.388533	0.731821	0.948267	0.452964

Ahora realizamos una gráfica para validar como dio la distribución entre normales y anormales, teniendo en cuenta el plano cerebral y el método de toma de la decisión

```
[42]: #pip install seaborn
```

```
Requirement already satisfied: seaborn in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages
(0.13.2)Note: you may need to restart the kernel to use updated packages.
```

```
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages (from
seaborn) (2.2.0)
```

```
Requirement already satisfied: pandas>=1.2 in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages (from
seaborn) (2.2.3)
```

Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages (from
seaborn) (3.10.0)

Requirement already satisfied: contourpy>=1.0.1 in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.3.1)

Requirement already satisfied: cycler>=0.10 in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)

Requirement already satisfied: fonttools>=4.22.0 in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (4.55.3)

Requirement already satisfied: kiwisolver>=1.3.1 in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.4.7)

Requirement already satisfied: packaging>=20.0 in
c:\users\lenovo\appdata\roaming\python\python313\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (24.2)

Requirement already satisfied: pillow>=8 in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (11.0.0)

Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (3.2.0)

Requirement already satisfied: python-dateutil>=2.7 in
c:\users\lenovo\appdata\roaming\python\python313\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)

Requirement already satisfied: pytz>=2020.1 in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages (from
pandas>=1.2->seaborn) (2024.2)

Requirement already satisfied: tzdata>=2022.7 in
c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages (from
pandas>=1.2->seaborn) (2024.2)

Requirement already satisfied: six>=1.5 in
c:\users\lenovo\appdata\roaming\python\python313\site-packages (from python-
dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)

```
[46]: # Configuración de estilo general
import seaborn as sns
sns.set_theme(style='darkgrid')

# Función para crear la gráfica
def crear_grafica(df, columna_diagnostico, titulo):
    planos = df['Plano_cerebral'].unique()
    diagnosticos = ['Normal', 'Anormal']

    # Crear tabla cruzada con los conteos
```

```

    conteos = df.groupby(['Plano_cerebral', columna_diagnostico]).size().
↳unstack(fill_value=0)

# Crear el gráfico
fig, ax = plt.subplots(figsize=(12, 6))

# Configurar posición de las barras
x = np.arange(len(planos)) # Posiciones en el eje X para cada plano
bar_width = 0.4 # Ancho de las barras

# Barras para "Normal"
bars_normal = ax.bar(x - bar_width / 2, conteos['Normal'], width=bar_width,
↳label='Normal', color='skyblue')
# Barras para "Anormal"
bars_anormal = ax.bar(x + bar_width / 2, conteos['Anormal'],
↳width=bar_width, label='Anormal', color='salmon')

# Añadir los valores al centro de las barras
for bar in bars_normal:
    height = bar.get_height()
    if height > 0: # Evitar etiquetas en barras con altura 0
        ax.text(bar.get_x() + bar.get_width() / 2, height / 2, str(height),
            ha='center', va='center', fontsize=10, color='black')
for bar in bars_anormal:
    height = bar.get_height()
    if height > 0: # Evitar etiquetas en barras con altura 0
        ax.text(bar.get_x() + bar.get_width() / 2, height / 2, str(height),
            ha='center', va='center', fontsize=10, color='black')

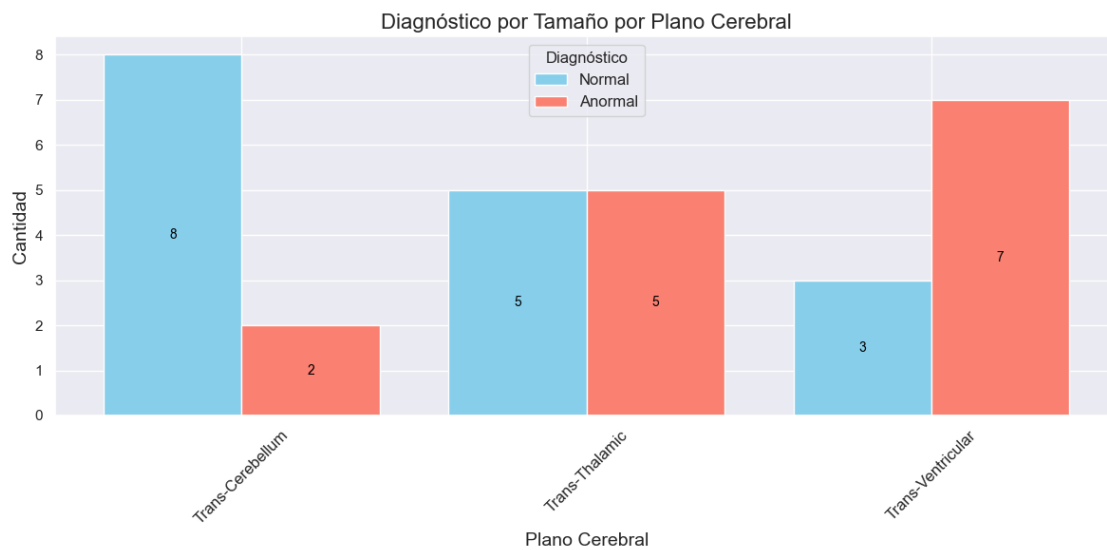
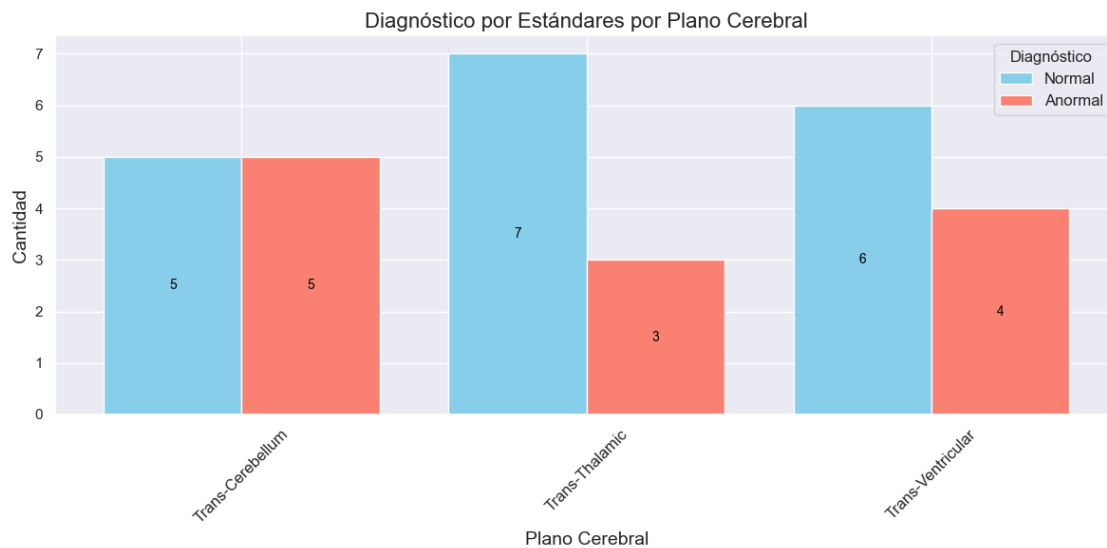
# Configurar etiquetas y título
ax.set_title(titulo, fontsize=16)
ax.set_xlabel('Plano Cerebral', fontsize=14)
ax.set_ylabel('Cantidad', fontsize=14)
ax.set_xticks(x)
ax.set_xticklabels(planos, rotation=45, fontsize=12)
ax.legend(title='Diagnóstico', fontsize=12)

# Mostrar la gráfica
plt.tight_layout()
plt.show()

# Crear gráfica para 'Diagnostico_por_estandares'
crear_grafica(
    df_resultados,
    columna_diagnostico='Diagnostico_por_estandares',
    titulo='Diagnóstico por Estándares por Plano Cerebral'
)

```

```
# Crear gráfica para 'diagnostico_tamaño'
crear_grafica(
  df_resultados,
  columna_diagnostico='diagnostico_tamaño',
  titulo='Diagnóstico por Tamaño por Plano Cerebral'
)
```



como podemos observar el diagnostico de planteado por estándares es más balanceado y se identifican menos imágenes con anomalías por plano cerebral, mientras que el diagnostico por tamaño el plano ventricular presenta muchos más casos anormales.

La razón principal para tomar la desviación estándar es debido a que es más practica por lo que estas expresada en las mismas unidades que los datos originales, así que de esta manera nos facilita entender de mejor manera los datos de las 10 imagen por plano que creamos.