



GUÍA DE USO DE HERRAMIENTAS DE ESTIMACIÓN DE BIOMASA

Optimización y Funcionalidades Avanzadas



14 DE DICIEMBRE DE 2023
TECNOLÓGICO NACIONAL DE MÉXICO
CAMPUS TUXTLA GUTIÉRREZ

Contenido

INFORMACION DE AUTORES	1
INTRODUCCIÓN	2
RECOMENDACIONES	3
CODIGO PROCESS PREDICTIONS	4
Link para descargar el modelo:	5
INTERFAZ PARA CAMARA INTEL REALSENSE D435	9
GRAFICADOR PLANTA 3D	12
VISOR DE IMÁGENES DEPTH	15
VISOR DE OBJETOS DE ARCHIVO NPZ	17
PROGRAMA PARA EXTREAR DATOS DE UN ARCHIVO NPZ	20
MODELO 3D PARA HOJAS	23
CÁLCULO VOLUMEN 3D	26
CÁLCULO ÁREA 2D	28
PROGRAMA PARA REDIMENSIONAR TAMAÑO IMAGEN	30
CONCLUSIONES	33
CONTACTOS	34

INFORMACION DE AUTORES



Emmanuel Bonilla Balbuena, Estudiante del noveno semestre de la carrera de Ingeniería en sistemas computacionales del Instituto Tecnológico de Tuxtla Gutiérrez, trabaja en proyecto de residencia titulado “Estimación no invasiva de biomasa vegetal mediante el fenotipado de la Fragaria basado en la segmentación SuperVoxel a partir de una nube de puntos 3D”



Ángel Andrés López Espinosa, Estudiante del noveno semestre de la carrera de Ingeniería en sistemas computacionales del Instituto Tecnológico de Tuxtla Gutiérrez, trabaja en proyecto de residencia titulado “Estimación no invasiva de biomasa vegetal mediante el fenotipado de la Fragaria basado en la segmentación SuperVoxel a partir de una nube de puntos 3D”

INTRODUCCIÓN

Este manual representa un recurso fundamental para aquellos inmersos en el mundo de la tecnología aplicada a la visualización tridimensional y el procesamiento de datos provenientes de cámaras Intel RealSense. Con un enfoque práctico y detallado, este compendio ofrece una guía exhaustiva para el uso de programas desarrollados en Python.

Estos programas, meticulosamente creados, abarcan una amplia gama de funcionalidades. Desde la activación y control de cámaras Intel RealSense hasta la generación tridimensional de plantas, proporcionan herramientas especializadas como visores de imágenes en profundidad y objetos, así como extractores y manipuladores de datos de archivos npz.dat.

Además, este manual facilita la comprensión y aplicación de modelos 3D para la representación gráfica de hojas, el cálculo preciso de volúmenes tridimensionales y áreas bidimensionales, y la optimización de imágenes mediante un redimensionador integrado.

Sin dejar de lado la implementación del código para el modelo YOLOv7, esta guía se convierte en un recurso integral, capacitando a los usuarios para la predicción y clasificación de objetos en imágenes.

En resumen, este compendio no solo enseña a utilizar programas específicos desarrollados en Python, sino que permite explorar y potenciar las capacidades de las cámaras Intel RealSense, la representación tridimensional, el procesamiento de imágenes y la manipulación de datos de una manera eficiente y efectiva.

RECOMENDACIONES

Para ejecutar programas que involucren procesamiento de imágenes, visualización tridimensional, y ejecución de modelos de aprendizaje automático, es recomendable tener una PC con ciertos componentes que garanticen un rendimiento adecuado. Aquí hay algunas recomendaciones:

Procesador (CPU)

Potencia de Procesamiento: Un procesador de al menos 4 núcleos físicos y una frecuencia base alta ayudaría en la ejecución de operaciones intensivas de procesamiento.

Memoria RAM

Capacidad: Un mínimo de 8 GB de RAM es esencial para manejar el procesamiento de imágenes y datos tridimensionales.

Velocidad: Una RAM con alta velocidad ayuda a la agilidad en el procesamiento de datos.

Tarjeta Gráfica (GPU)

GPU Dedicada: Es beneficiosa para acelerar tareas de procesamiento gráfico, visualización tridimensional y ejecución de modelos de aprendizaje automático.

Memoria VRAM: Una GPU con una cantidad considerable de memoria VRAM facilita la manipulación de imágenes en alta resolución y operaciones de aprendizaje automático.

Almacenamiento

SSD: Se recomienda un disco de estado sólido (SSD) para almacenamiento, ya que ofrece tiempos de carga más rápidos y lectura/escritura eficiente, lo que mejora la velocidad de acceso a datos.

Sistema Operativo

Compatibilidad: Asegúrate de utilizar un sistema operativo WINDOWS 10, 11 o LINUX, compatible con las herramientas y bibliotecas específicas requeridas por los programas (como versiones de Python, bibliotecas de visión por computadora, etc.).

Otros Componentes

Conectividad: Una conexión de red estable y rápida puede ser necesaria si los programas requieren acceso a recursos en la nube o descargas de datos.

Puertos USB: Si se utilizan dispositivos externos como cámaras Intel RealSense u otros dispositivos periféricos, asegúrate de tener suficientes puertos USB disponibles.

Estos son lineamientos generales; los requerimientos específicos pueden variar según los programas y la complejidad de las operaciones que desees realizar. Siempre es útil revisar los requisitos de sistema recomendados por los desarrolladores de los programas en cuestión para obtener un rendimiento óptimo.

CODIGO PROCESS PREDICTIONS

1. Configuración inicial en Google Colab:

- *Abre Google Colab desde tu navegador.*
- *Selecciona la opción de conectar con entorno de ejecución.*
- *Verifica que tengas acceso a los archivos necesarios (modelos pre-entrenados, datos de prueba, etc.).*

2. Carga del modelo YOLOv7:

- *Descarga el modelo YOLOv7 o utiliza la implementación existente a través de repositorios como GitHub.*
- *Carga el modelo en tu entorno de Colab utilizando las bibliotecas y herramientas necesarias de Python.*

3. Procesamiento de imágenes y segmentación semántica:

- *Utiliza el modelo YOLOv7 para realizar la detección de objetos en las imágenes.*
- *Una vez que se detecten los objetos, realiza la segmentación semántica para generar las máscaras correspondientes a cada objeto detectado.*

4. Almacenamiento de máscaras en un archivo npz.dat:

- *Guarda las máscaras generadas en un archivo npz.dat utilizando funciones o librerías de manejo de archivos en Python, como NumPy por ejemplo.*
- *Asegúrate de organizar y etiquetar adecuadamente las máscaras dentro del archivo npz.dat para su fácil recuperación posterior.*



5. Verificación y almacenamiento final:

- Verifica que las máscaras se hayan almacenado correctamente en el archivo npz.dat y que contengan la información requerida para su uso futuro.
- Guarda el archivo npz.dat en un directorio o sistema de almacenamiento accesible desde Colab para su uso posterior o descarga.

Recomendaciones:

- Revisa documentación adicional, tutoriales o ejemplos específicos de YOLOv7 en Google Colab para entender mejor su implementación en este entorno.
- Asegúrate de tener el conjunto de datos adecuado y los scripts necesarios para procesar las imágenes y generar las máscaras.

Este proceso puede requerir el uso de librerías y herramientas específicas, así como una comprensión detallada de cómo funcionan el modelo YOLOv7 y la manipulación de imágenes en Python. Además, la organización y estructuración de las máscaras dentro del archivo npz.dat es crucial para su utilidad posterior.

 DETECCION_POR_SEGMENTACION_YOLOV7_0923 

Link para descargar el modelo:

<https://drive.google.com/file/d/1B3Ob5tqr6xXEsM0B0unJpOlzzX4aV8ON/view?usp=sharing>

```
for i, det in enumerate(pred): # per image
    seen += 1
    if webcam: # batch_size >= 1
        p, im0, frame = path[i], im0s[i].copy(),
dataset.count
        s += f'{i}: '
    else:
        p, im0, frame = path, im0s.copy(),
getattr(dataset, 'frame', 0)

    p = Path(p) # to Path
    save_path = str(save_dir / p.name) # im.jpg
```

```

        txt_path = str(save_dir / 'labels' / p.stem) + (' ' if
dataset.mode == 'image' else f'_{frame}') # im.txt
        s += '%gx%g ' % im.shape[2:] # print string
        gn = torch.tensor(im0.shape)[[1, 0, 1, 0]] #
normalization gain whwh
        imc = im0.copy() if save_crop else im0 # for
save_crop
        annotator = Annotator(im0, line_width=line_thickness,
example=str(names))
        if len(det):
            masks = process_mask(proto[i], det[:, 6:], det[:,
:4], im.shape[2:], upsample=True) # HWC

            nombre = f"matriz_{cantidad}.dat"
            cantidad += 1

            # Guardar la matriz en un archivo .dat
            file_path = os.path.join(save_dir, nombre)
            np.savez(file_path, matriz=masks.cpu().numpy())

```

Este fragmento de código es un bucle que itera sobre los resultados de detección (`pred`) para cada imagen procesada. Aquí está la explicación línea por línea:

- `for i, det in enumerate(pred):`: Este bucle `for` itera sobre las detecciones realizadas para cada imagen. `i` representa el índice de la imagen en el conjunto de datos, mientras que `det` contiene la información de las detecciones (coordenadas, clases, etc.).

- `seen += 1`: Incrementa un contador para cada imagen vista.

- `if webcam:` y el bloque de código relacionado: Verifica si se está utilizando una cámara web (en contraposición a una imagen estática). En el caso de una cámara web, el código procesa las imágenes de manera específica, registrando los datos correspondientes.

- `p = Path(p)`, `save_path`, `txt_path`: Maneja las rutas de archivo para guardar las imágenes procesadas y los archivos de texto relacionados con las detecciones.

- `s += '%gx%g ' % im.shape[2:]`: Agrega información sobre el tamaño de la imagen a una cadena de texto (`s`), mostrando el ancho y alto de la imagen.

- ``gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]``: Calcula un tensor que representa el ajuste de normalización para las coordenadas de las detecciones.

- ``imc = im0.copy() if save_crop else im0``: Prepara una copia de la imagen original (``im0``) para la manipulación, dependiendo de si se va a guardar un recorte de la imagen o la imagen completa.

- ``annotator = Annotator(im0, line_width=line_thickness, example=str(names))``: Inicializa un objeto para la anotación gráfica de la imagen original.

- ``if len(det):``: Verifica si se detectaron objetos en la imagen actual. Si hay detecciones:

- ``masks = process_mask(proto[i], det[:, 6:], det[:, :4], im.shape[2:], upsample=True)``: Procesa las máscaras de los objetos detectados a partir de los resultados de detección (``det``). Esta línea parece utilizar una función llamada ``process_mask`` para generar máscaras a partir de las detecciones.

- ``nombre = f'matriz_{cantidad}.dat'`` y ``cantidad += 1``: Genera un nombre de archivo para guardar la matriz resultante de las máscaras en un archivo ``dat``.









- ``file_path = os.path.join(save_dir, nombre)``: Crea la ruta del archivo de salida ``dat``.

- ``np.savez(file_path, matriz=masks.cpu().numpy())``: Guarda la matriz generada (``masks``) en un archivo ``dat`` utilizando la biblioteca NumPy, convirtiendo las máscaras a formato numpy antes de guardarlas.

CON ESTOS ARCHIVOS EL CUAL NOS GUARDA COLAB EN NUESTRO GOOGLE DRIVE PODEMOS USARLOS PARA LOS SIGUIENTES PROCESOS.

Los datos estarán guardados en la ruta. H:\Mi unidad\yolov7\seg\runs\predict-seg

En esa dirección encontraremos los archivos npz.dat con los cuales trabajaremos

Nombre	Fecha de modificación	Tipo	Tamaño
 imagen_rgb_2_png.rf.0f5b63233c7472de...	19/09/2023 11:20 p. m.	Archivo JPG	61 KB
 imagen_rgb_15_png.rf.b66a8b92bd6a138...	19/09/2023 11:20 p. m.	Archivo JPG	58 KB
 imagen_rgb_17_png.rf.6edcd67dac56e9d...	19/09/2023 11:20 p. m.	Archivo JPG	58 KB
 imagen_rgb_24_png.rf.6a06f05e9c689186...	19/09/2023 11:20 p. m.	Archivo JPG	59 KB
 matriz_0.dat.npz	19/09/2023 11:20 p. m.	Archivo NPZ	59,201 KB
 matriz_1.dat.npz	19/09/2023 11:20 p. m.	Archivo NPZ	59,201 KB
 matriz_2.dat.npz	19/09/2023 11:20 p. m.	Archivo NPZ	65,601 KB
 matriz_3.dat.npz	19/09/2023 11:20 p. m.	Archivo NPZ	78,401 KB

INTERFAZ PARA CAMARA INTEL REALSENSE D435

Librerías Utilizadas

- ``tkinter``: Para crear la interfaz gráfica.
- ``cv2``: OpenCV para procesamiento de imágenes.
- ``time``, ``os``, ``numpy``, ``datetime``: Librerías para manejar tiempo, archivos, matrices y fechas.
- ``pyrealsense2``: Librería para interactuar con la cámara RealSense.
- ``PIL.Image``, ``ImageTk``: Para manipulación y visualización de imágenes.

Funcionalidad Principal

Inicio de la Cámara: La función ``start_camera()`` inicializa la cámara RealSense con configuraciones específicas para obtener imágenes en color y en profundidad.

Vista Previa de Imagen: La función ``show_image_preview()`` muestra una ventana emergente con vistas previas de la imagen en color y en profundidad, permitiendo al usuario ingresar un nombre y una fecha para guardar la imagen.

Captura de Imágenes: La función ``take_photo()`` captura imágenes en tiempo real tanto en color como en profundidad. También proporciona la funcionalidad para guardar las imágenes capturadas.

Guardado de Imágenes: La función ``save_image()`` guarda las imágenes capturadas con un nombre y fecha especificados en directorios separados para imágenes en RGB y en profundidad, junto con archivos numpy (`` .npy``) que contienen los datos de las imágenes en formato de matriz.

Actualización de la Interfaz: La función ``update_camera_frame()`` actualiza continuamente las vistas de la cámara en tiempo real en la interfaz.

Interfaz Gráfica: Se crea una interfaz con botones para iniciar la cámara, tomar fotos y salir de la aplicación, junto con etiquetas que muestran las vistas de la cámara en la interfaz.

Utilización del Programa

1. **Inicio:** Se inicia el programa, se accede a la cámara RealSense y se muestra una interfaz con vistas previas en tiempo real de la cámara.
2. **Operación:** Al presionar los botones "Iniciar Cámara" y "Tomar Foto", se activa la cámara y se capturan imágenes en color y profundidad respectivamente.
3. **Guardar Imágenes:** Se proporciona una ventana emergente para ingresar un nombre y fecha para guardar la imagen capturada.

Proceso de Guardado de Imágenes

Cuando se toma una foto con la cámara RealSense:

Las imágenes en color y en profundidad se almacenan en las carpetas RGB y Profundidad respectivamente, con nombres específicos según la fecha y hora de captura.

Además, los datos de estas imágenes se guardan como archivos .npy en las carpetas RGB_npy y Profundidad_npy.

Ubicación de Almacenamiento

Las imágenes se almacenan localmente en el sistema de archivos en las carpetas mencionadas anteriormente.

La estructura de carpetas se crea dentro de la carpeta desde donde se ejecuta el programa. Por ejemplo, si el programa se ejecuta desde el escritorio, estas carpetas se crearán allí.

Verificación y Sobrescritura

Antes de guardar una nueva imagen con el mismo nombre y fecha, el programa verifica si ya existe un archivo con ese nombre.

Si existe un archivo con el mismo nombre y fecha, se pregunta al usuario si desea sobrescribirlo o no.

Ubicación de la Aplicación









La aplicación se puede ejecutar desde cualquier directorio, y la estructura de carpetas se creará dentro del directorio actual desde donde se ejecute el programa.

4. Finalización: Al presionar el botón "Salir", se cierra la aplicación y se detiene la cámara.

Instalación de Librerías

Asegúrate de tener instaladas las librerías listadas al inicio del código, especialmente ``tkinter``, ``cv2``, ``pyrealsense2``, ``numpy``, y ``PIL``. Puedes instalarlas usando ``pip``, por ejemplo: ``pip install opencv-python``.

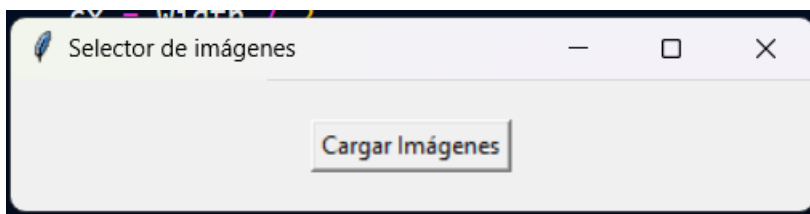


 CapturasIntelRealSense	25/09/2023 01:49 p. m.	Carpeta de archivos
 20230802	02/08/2023 11:46 a. m.	Carpeta de archivos
 20230803	03/08/2023 11:59 a. m.	Carpeta de archivos
 20230810	17/08/2023 12:46 p. m.	Carpeta de archivos
 20230814	17/08/2023 12:47 p. m.	Carpeta de archivos
 20230911	11/09/2023 12:58 p. m.	Carpeta de archivos
 20230919	19/09/2023 10:56 a. m.	Carpeta de archivos
 20230925	25/09/2023 01:49 p. m.	Carpeta de archivos

Profundidad	02/08/2023 11:56 a. m.	Carpeta de archivos
Profundidad_npy	02/08/2023 11:56 a. m.	Carpeta de archivos
RGB	02/08/2023 11:56 a. m.	Carpeta de archivos
RGB_npy	02/08/2023 12:07 p. m.	Carpeta de archivos

GRAFICADOR PLANTA 3D

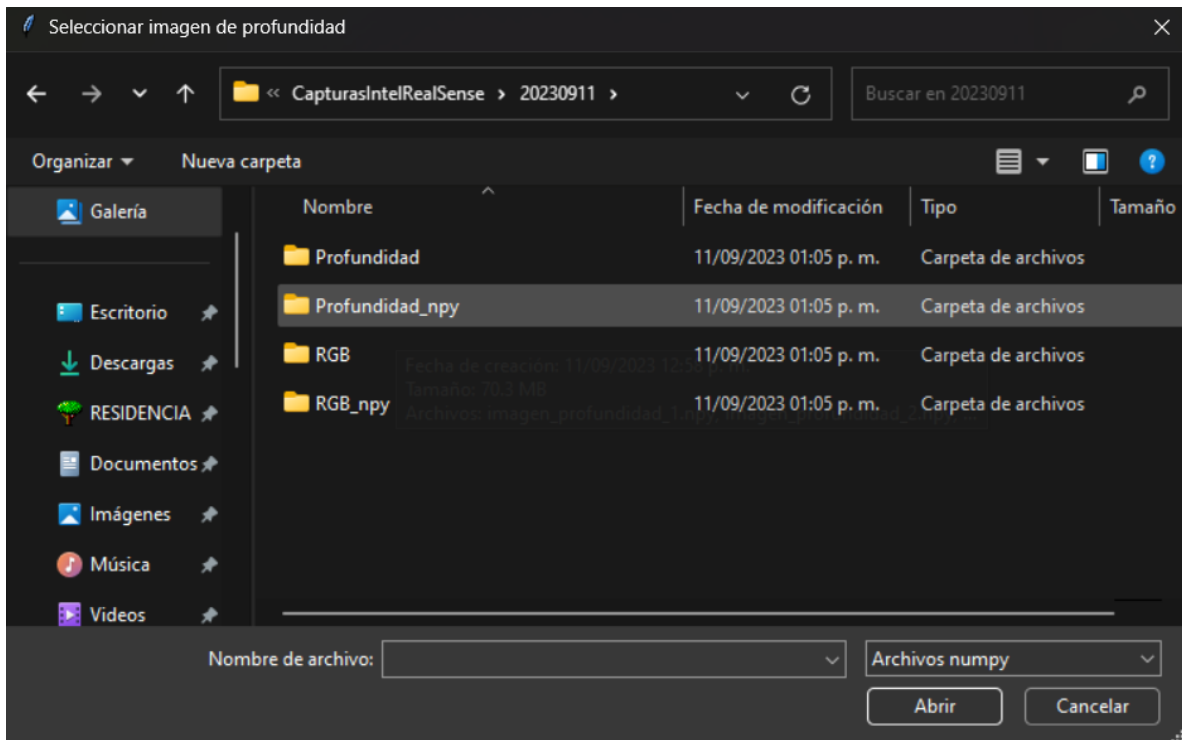
Funcionamiento del Programa:



1. Interfaz Gráfica: Al ejecutar el programa, aparecerá una ventana con un botón llamado "Cargar Imágenes".

2. Selección de Imágenes: Al hacer clic en "Cargar Imágenes", se abrirán dos ventanas emergentes para que selecciones dos archivos:

- Imagen de Profundidad: Selecciona un archivo de imagen de profundidad en formato `.npy`.
- Imagen RGB: Selecciona un archivo de imagen en formato RGB (también en `.npy`).



3. Procesamiento de Imágenes: Una vez seleccionadas ambas imágenes, el programa procesará estas imágenes para crear el modelo 3D.

4. Visualización del Modelo 3D: El programa generará una representación tridimensional basada en las imágenes de profundidad y color seleccionadas. Esta representación se visualizará en una ventana gráfica que mostrará puntos 3D coloreados según la información de las imágenes.

Imágenes Necesarias:

- Imagen de Profundidad: Debe ser un archivo de imagen de profundidad en formato `.npy`.
- Imagen RGB: Debe ser un archivo de imagen en formato RGB (también en `.npy`).

Librerías Necesarias:

1. ``tkinter``: Para la creación de la interfaz gráfica.
2. ``filedialog`` de ``tkinter``: Para abrir las ventanas de selección de archivos.
3. ``skimage.util``: Para procesamiento de imágenes.
4. ``matplotlib.pyplot``: Para la visualización de gráficos y el modelo 3D.
5. ``mpl_toolkits.mplot3d.Axes3D``: Para manejar gráficos 3D.
6. ``numpy``: Para operaciones matriciales y manejo de datos.

Proceso de Generación del Modelo 3D:

- Las imágenes seleccionadas (profundidad y RGB) se cargan y se procesan para generar puntos en un espacio tridimensional basado en la información de profundidad.
- Estos puntos se grafican en una ventana 3D utilizando colores RGB para representar los valores de las imágenes RGB, creando así una representación visual del modelo tridimensional.

Uso del Manual:

1. Ejecución del Programa:** Ejecuta el programa desde tu entorno de Python.
2. Selección de Imágenes:** Selecciona una imagen de profundidad y una imagen RGB en formato `.numpy`.
3. Visualización del Modelo: Observa la representación tridimensional generada en una ventana gráfica.

```
for y in range(0, height, 1):  
    for x in range(0, width, 1):  
        z = depth_image[y, x] * k  
        if 0 < z < 500 :
```

Modificación de los Valores de Z:

Si modificas los valores de z:

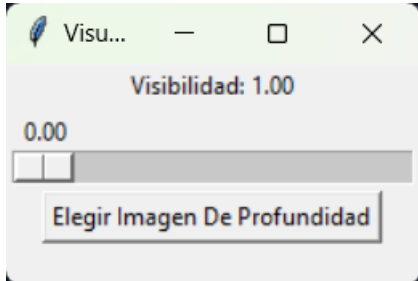
Disminución de Z: Reducir los valores de z puede acercar los puntos generados al origen del espacio 3D. Esto puede dar la sensación de que el objeto detectado está más cerca.

Aumento de Z: Incrementar los valores de z puede alejar los puntos generados del origen del espacio 3D, dando la impresión de que el objeto está más lejos.

Para garantizar que el programa funcione correctamente, asegúrate de tener las librerías mencionadas instaladas en tu entorno de Python. Puedes instalarlas utilizando `pip`, por ejemplo: `pip install matplotlib`.

VISOR DE IMÁGENES DEPTH

Funcionamiento General del Programa:



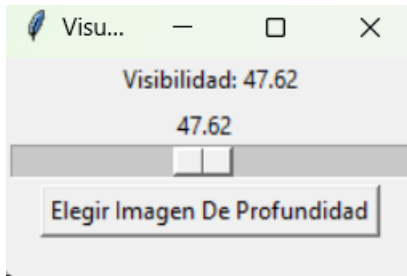
1. Interfaz Gráfica:

Al ejecutar el programa, se abre una ventana que contiene una etiqueta que muestra la visibilidad actual, una barra deslizante para ajustar la visibilidad, un botón para elegir la imagen de profundidad y una etiqueta para mostrar mensajes.

2. Acciones:

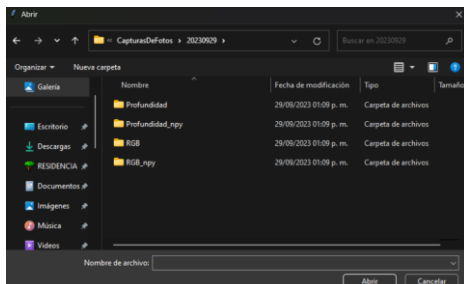
Ajuste de Visibilidad:

La barra deslizante permite ajustar la visibilidad de la imagen en un rango de 0.0 a 100.0.



Elegir Imagen de Profundidad:

El botón "Elegir Imagen de Profundidad" abre una ventana para seleccionar un archivo de imagen (ya sea `.npy`, `.png` o `.jpg`) que contiene datos de profundidad.



Visualización de Imágenes:**

Después de elegir una imagen, la aplicación ajusta la visibilidad de la imagen de acuerdo con el valor seleccionado en la barra deslizante.

Luego, guarda temporalmente la imagen ajustada como un archivo `.png` y la abre en el visor de imágenes predeterminado del sistema operativo.

La imagen temporal se elimina después de un segundo para mantener la limpieza.

Uso del Programa:

1. Ejecución del Programa:

- Ejecuta el programa desde tu entorno de Python.

2. Ajuste de Visibilidad:

- Utiliza la barra deslizante para cambiar la visibilidad de la imagen.

3. Elegir Imagen de Profundidad:

- Presiona el botón "Elegir Imagen de Profundidad" para seleccionar un archivo de imagen.

4. Visualizar Imágenes:

- Después de seleccionar la imagen, se ajusta su visibilidad según el valor de la barra deslizante y se muestra temporalmente en el visor de imágenes predeterminado del sistema operativo.

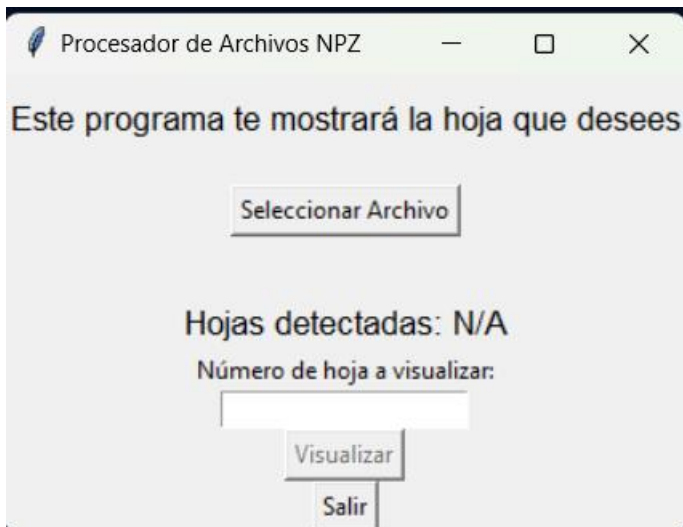
Librerías Necesarias:

- ``tkinter``: Para la interfaz gráfica.
- ``filedialog`` de ``tkinter``: Para seleccionar archivos de imágenes.
- ``subprocess``: Para abrir el visor de imágenes predeterminado del sistema operativo.
- ``cv2`` (OpenCV): Para manejar imágenes.
- ``numpy``: Para operaciones con matrices.
- ``os``: Para operaciones con archivos y directorios.
- ``time``: Para añadir un retraso antes de eliminar el archivo temporal.

Este programa proporciona una forma sencilla de ajustar la visibilidad de las imágenes de profundidad y visualizarlas en el visor de imágenes del sistema operativo de forma rápida y conveniente.

VISOR DE OBJETOS DE ARCHIVO NPZ

Funcionamiento General del Programa:



1. Interfaz Gráfica:

- Se inicia la ventana principal que contiene:
- Un mensaje introductorio.
- Un botón "Seleccionar Archivo" para cargar archivos `.npz`.
- Una etiqueta para mostrar el resultado del procesamiento.
- Una etiqueta para mostrar la cantidad de hojas detectadas.
- Un campo de entrada para ingresar el número de hoja a visualizar.
- Un botón "Visualizar" para mostrar la hoja seleccionada.
- Un botón "Salir" para cerrar la aplicación.

2. Acciones:

- Seleccionar Archivo:
- Al hacer clic en "Seleccionar Archivo", se abre una ventana para elegir un archivo `.npz`.
- Los datos del archivo se procesan para encontrar coordenadas de hojas.
- Se crea una estructura de carpetas para guardar los datos procesados.

- Visualizar Hoja:

- Después de cargar y procesar el archivo `.npz`, se puede ingresar un número de hoja para visualizarla.

- Al hacer clic en "Visualizar", muestra la imagen de la hoja seleccionada en escala de grises utilizando `matplotlib`.

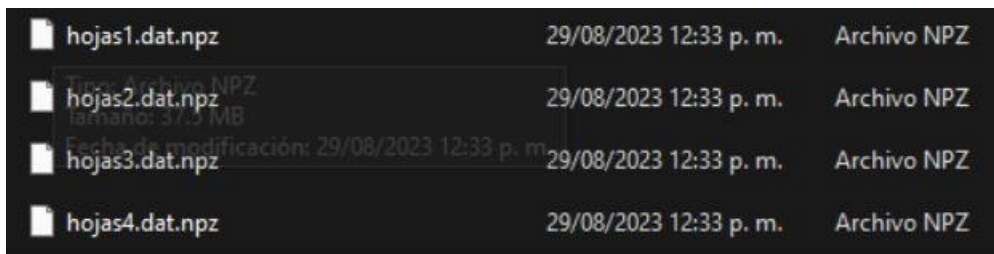
Uso del Programa:

1. Ejecución del Programa:

- Ejecuta el programa desde tu entorno de Python.

2. Cargar Archivo `.npz`:

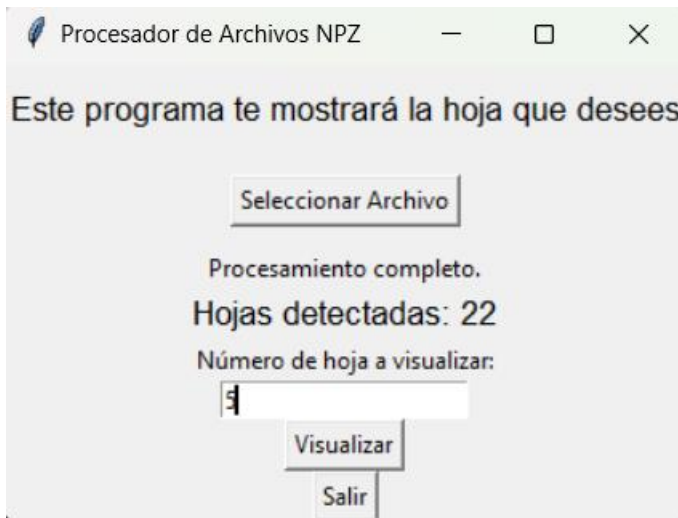
- Haz clic en "Seleccionar Archivo" para elegir un archivo `.npz`.

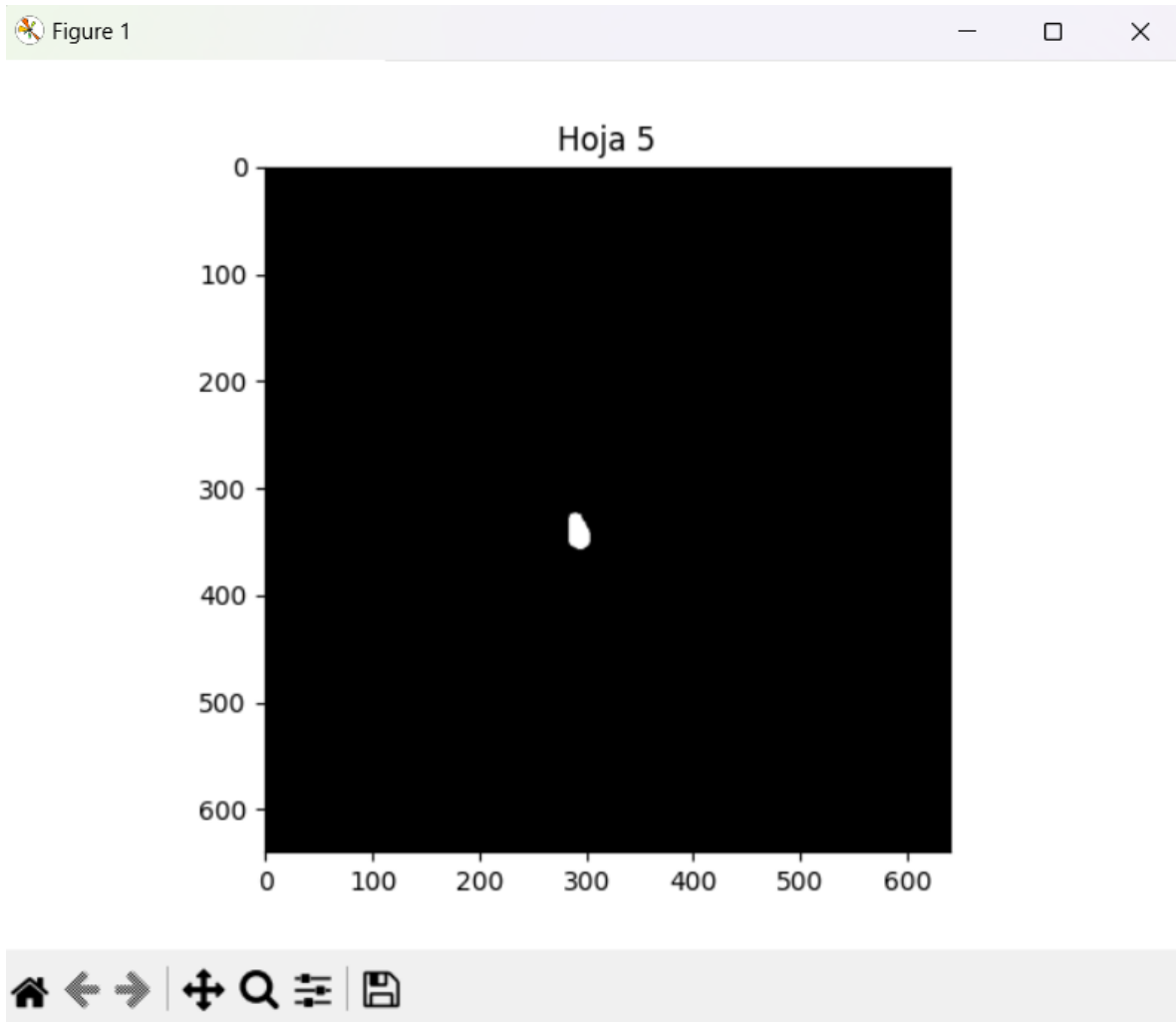


3. Visualizar Hojas:

- Ingresa el número de hoja en el campo de entrada.

- Haz clic en "Visualizar" para mostrar la hoja seleccionada.





4. Cerrar la Aplicación:

- Utiliza el botón "Salir" para cerrar la aplicación.

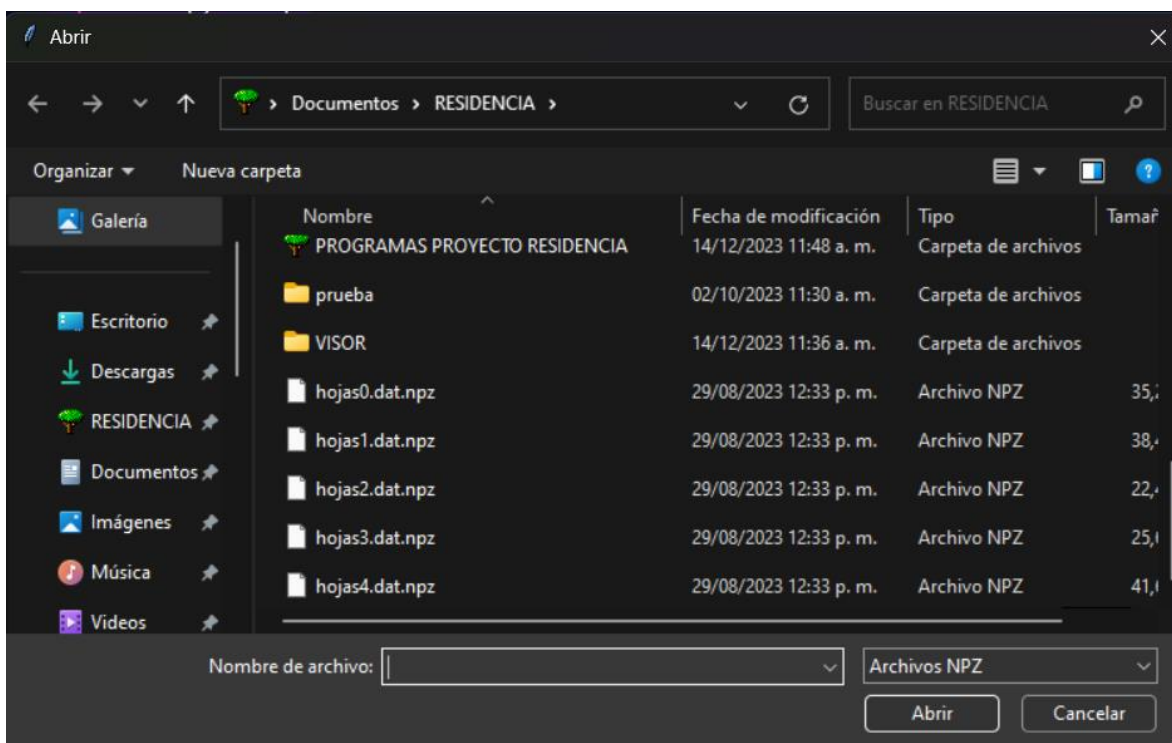
Librerías Necesarias:

- ``tkinter``: Para la interfaz gráfica.
- ``filedialog`` de ``tkinter``: Para seleccionar archivos ``.npz``.
- ``numpy``: Para manipulación de datos.
- ``os``: Para operaciones con archivos y directorios.
- ``datetime``: Para trabajar con fechas y horas.
- ``matplotlib.pyplot``: Para mostrar imágenes.

Este programa proporciona una manera de cargar archivos ``.npz``, encontrar coordenadas de hojas y visualizar hojas específicas según la entrada del usuario, lo que facilita el análisis y la exploración de estos archivos de datos.

PROGRAMA PARA EXTREAR DATOS DE UN ARCHIVO NPZ

Este programa está diseñado para procesar archivos `.npz` y extraer coordenadas de píxeles de matrices almacenadas en estos archivos. Aquí tienes una explicación de su funcionamiento:



Funcionamiento General del Programa:

1. Interfaz Gráfica Oculta:

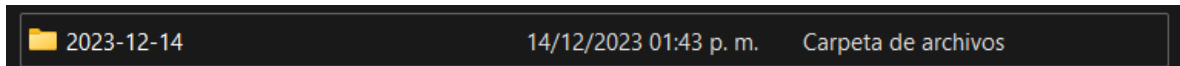
- Se crea una ventana de `tkinter` (`root`) que está oculta (`root.withdraw()`). No se muestra al usuario, pero se utiliza para aprovechar el diálogo de selección de archivos (`filedialog`).

2. Selección de Archivo:

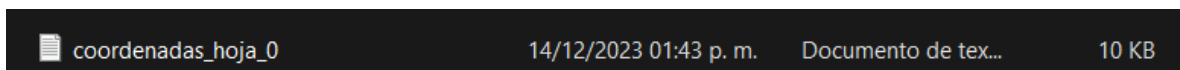
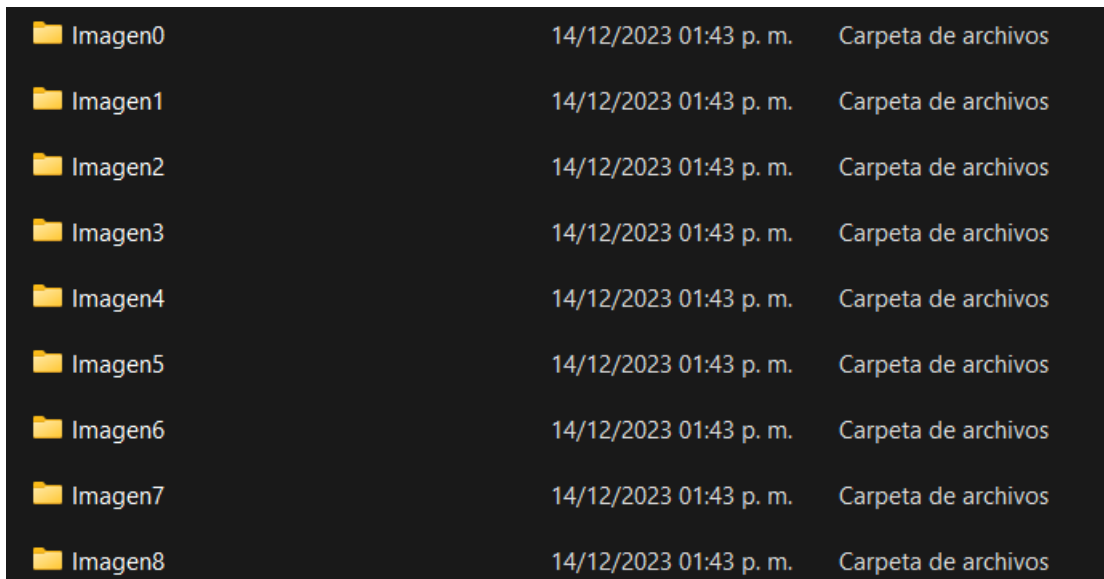
- Se abre un cuadro de diálogo para seleccionar un archivo `.npz`.
- Si se elige un archivo, se procede con el procesamiento; de lo contrario, se muestra un mensaje indicando que no se seleccionó ningún archivo.

3. Procesamiento de Archivo Seleccionado:

- Si se selecciona un archivo:
- Los datos se cargan desde el archivo `.npz`.
- Se extrae la matriz `matriz` del archivo.
- Se crea una carpeta con la fecha actual en formato `YYYY-MM-DD`.
- Se crea una carpeta con el nombre del archivo (sin extensión) dentro de la carpeta de la fecha.



- Se itera a través de las matrices y se encuentran las coordenadas de los píxeles distintos de cero.
- Se almacenan las coordenadas de cada hoja en una lista `coordenadas_por_hoja`.
- Se crean carpetas individuales para cada hoja y se guarda un archivo de texto (`coordenadas_hoja_{i}.txt`) con las coordenadas.



Uso del Programa:

1. Ejecución del Programa:

- Ejecuta el script en tu entorno de Python.

2. Seleccionar Archivo `.npz`:

- Selecciona un archivo `.npz` mediante el cuadro de diálogo que aparece.

- El programa procesará el archivo seleccionado y guardará las coordenadas de píxeles en archivos de texto organizados en carpetas según la estructura mencionada anteriormente.

Librerías Necesarias:

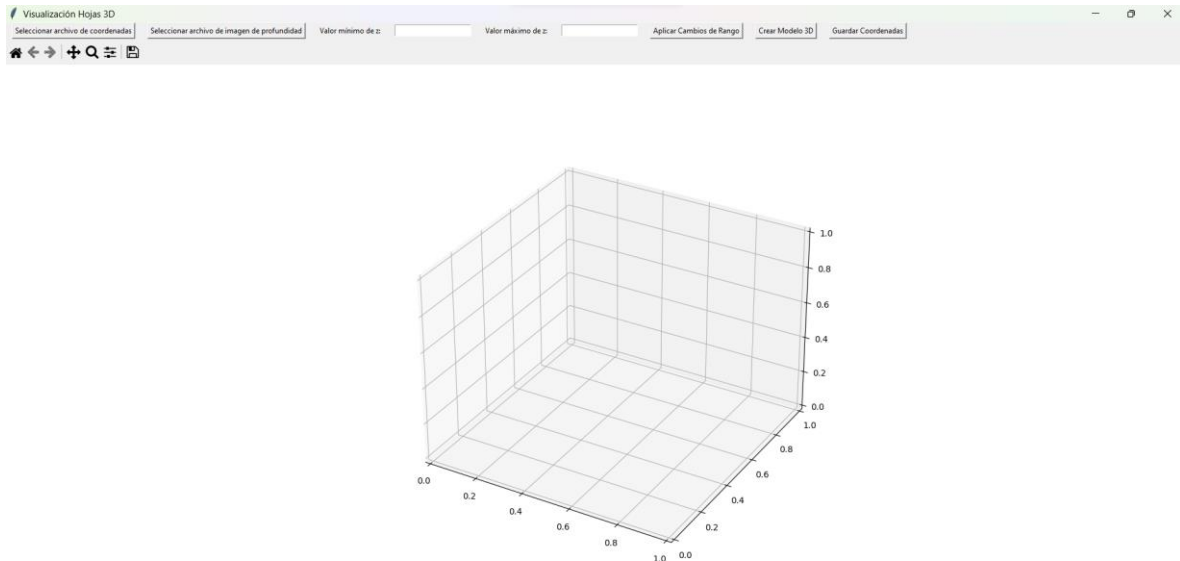
- ``numpy``: Para la manipulación de datos y el procesamiento de archivos ``npz``.
- ``os``: Para realizar operaciones con archivos y directorios.
- ``datetime``: Para trabajar con fechas y horas.
- ``tkinter`` (usado solo para el diálogo de selección de archivos): Para la selección de archivos ``npz`` a través de ``filedialog``.

Este programa facilita la extracción y organización de datos de píxeles almacenados en archivos ``npz`` en carpetas individuales con archivos de texto que contienen coordenadas de píxeles para cada hoja del archivo ``npz``.

MODELO 3D PARA HOJAS

Este código permite visualizar un modelo 3D a partir de un conjunto de coordenadas y una imagen de profundidad. Aquí te explico su funcionamiento:

Funcionamiento General del Programa:



1. Interfaz Gráfica:

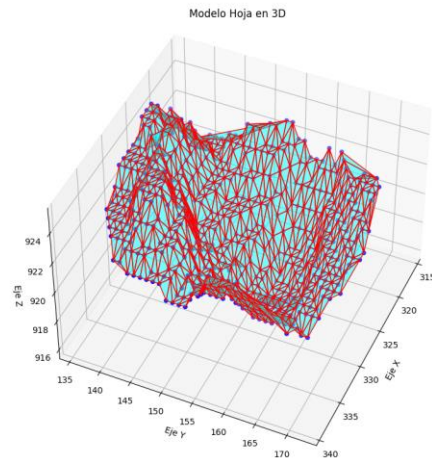
- Se utiliza `tkinter` para crear una interfaz gráfica.
- Se incluyen botones para seleccionar archivos de coordenadas y de imagen de profundidad.
- Hay campos de entrada para definir el rango mínimo y máximo de z, así como un botón para aplicar los cambios.



2. Visualización 3D:

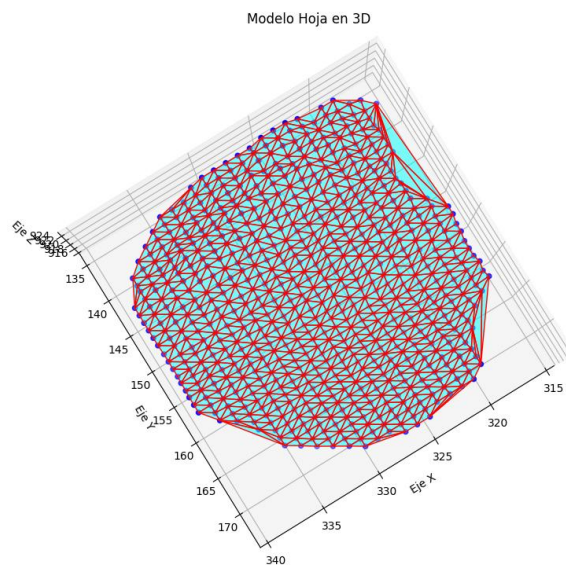
- Se grafica un modelo 3D basado en las coordenadas y la imagen de profundidad seleccionadas. El archivo de coordenadas lo encontramos en la ruta donde utilizamos el programa anterior después de ejecutarlo.

- Se muestra un gráfico interactivo en 3D usando `matplotlib` y `mpl_toolkits`.



3. Guardar Coordenadas:

- Se incluye un botón para guardar las coordenadas visualizadas en un archivo de texto.



Uso del Programa:

1. Ejecución del Programa:

- Ejecuta el script en tu entorno de Python.

2. Selección de Archivos:

- Utiliza los botones "Seleccionar archivo de coordenadas" y "Seleccionar archivo de imagen de profundidad" para cargar tus archivos respectivos.

3. Ajuste del Rango de Z:

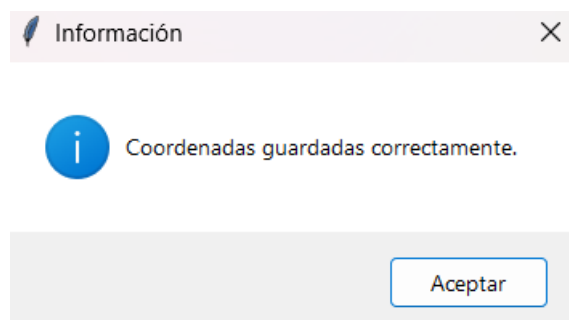
- Completa los campos "Valor mínimo de z" y "Valor máximo de z" con los límites deseados para el eje Z.
- Pulsa el botón "Aplicar Cambios de Rango" para ajustar el rango.

4. Creación del Modelo 3D:

- Haz clic en "Crear Modelo 3D" para generar la visualización basada en los archivos seleccionados y el rango Z definido.

5. Guardar Coordenadas:

- Al utilizar "Guardar Coordenadas", se almacenarán las coordenadas visualizadas en un archivo de texto.



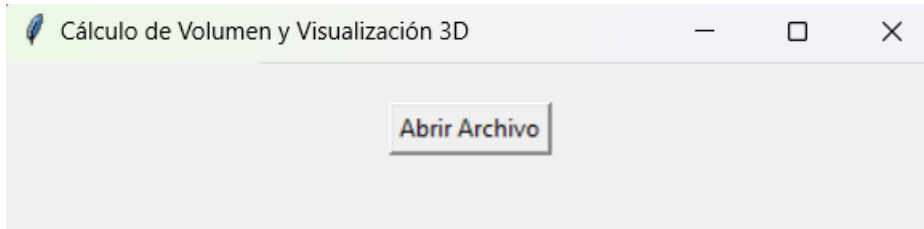
Librerías Necesarias:

- ``numpy``: Para el manejo de datos y operaciones matemáticas.
- ``matplotlib``: Para visualizaciones en 2D y 3D.
- ``tkinter``: Para la creación de la interfaz gráfica y la interacción con el usuario.
- ``scipy.spatial``: Para la triangulación y operaciones espaciales.
- ``matplotlib.backends.backend_tkagg``: Para integrar los gráficos de Matplotlib en la interfaz de usuario de Tkinter.

Este programa facilita la visualización interactiva en 3D de modelos basados en coordenadas y permite guardar estas coordenadas después de la visualización.

CÁLCULO VOLUMEN 3D

Este programa tiene la finalidad de calcular el volumen a partir de un conjunto de coordenadas tridimensionales y crear una representación 3D del objeto. Aquí te explico su funcionamiento:



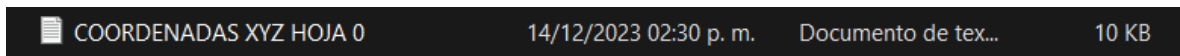
Funcionamiento General del Programa:

1. Interfaz Gráfica:

- Se utiliza `tkinter` para crear una ventana con un botón para abrir un archivo de coordenadas.

2. Cálculo de Volumen:

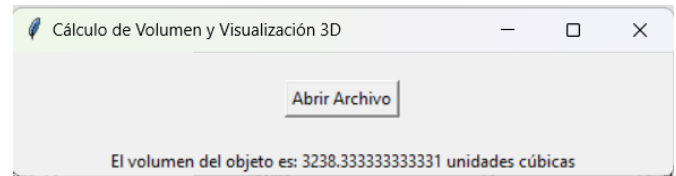
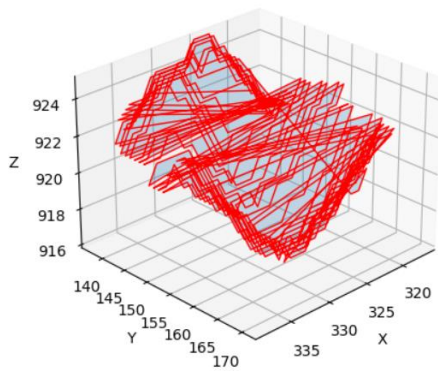
- Lee las coordenadas desde un archivo seleccionado por el usuario. Este archivo a seleccionar será el archivo que generamos después de dar click en la opción guardar coordenadas del programa anterior para generar el modelo 3d de hojas.



- Calcula el volumen del objeto representado por estas coordenadas.

3. Visualización 3D:

- Crea una representación gráfica del objeto tridimensional utilizando `matplotlib` con `mpl_toolkits`.



Uso del Programa:

1. Ejecución del Programa:

- Ejecuta el script en tu entorno de Python.

2. Selección de Archivo de Coordenadas:

- Haz clic en el botón "Abrir Archivo" para seleccionar un archivo de coordenadas.

3. Cálculo de Volumen:

- Tras seleccionar el archivo, el programa calculará automáticamente el volumen del objeto basado en las coordenadas proporcionadas.

4. Visualización 3D:

- Se generará una visualización tridimensional del objeto con la estructura de coordenadas leída.

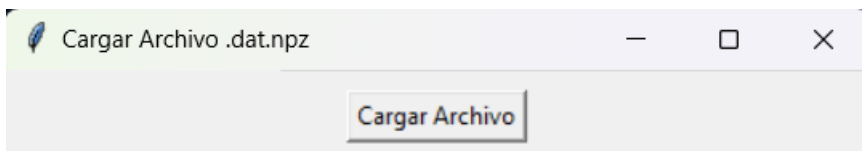
Librerías Necesarias:

- `tkinter`: Para la creación de la interfaz gráfica y la interacción con el usuario.
- `numpy`: Para el manejo de datos y operaciones matemáticas.
- `matplotlib`: Para visualizaciones en 2D y 3D.

Este programa brinda una forma sencilla de calcular el volumen de un objeto tridimensional a partir de un conjunto de coordenadas y ofrece una representación visual 3D del objeto utilizando una representación de malla de triángulos.

CÁLCULO ÁREA 2D

Este programa tiene como objetivo calcular y guardar las áreas de los objetos en imágenes contenidas en un archivo de extensión `.dat.npz`. Aquí está la descripción de su funcionamiento:



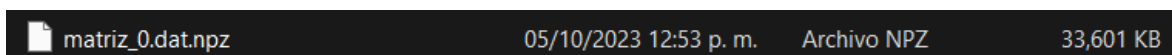
Funcionamiento General del Programa:

1. Interfaz Gráfica:

- Se usa `tkinter` para crear una ventana con un botón para cargar un archivo `.dat.npz`.

2. Cálculo de Áreas:

- Lee el archivo `.dat.npz` seleccionado por el usuario.
- Extrae las matrices de datos del archivo y procesa cada matriz como una imagen.
- Calcula el área del objeto en cada imagen y la guarda junto con la imagen procesada.



Uso del Programa:

1. Ejecución del Programa:

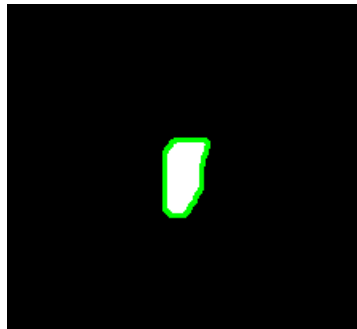
- Ejecuta el script en tu entorno de Python.

2. Carga del Archivo .dat.npz:

- Haz clic en el botón "Cargar Archivo" para seleccionar un archivo `.dat.npz`.

3. Procesamiento y Cálculo de Áreas:

- Tras seleccionar el archivo, el programa procesará las matrices de datos contenidas en él.
- Convertirá estas matrices en imágenes y calculará las áreas de los objetos presentes en esas imágenes.
- Guardará las imágenes procesadas con los contornos de los objetos y las áreas calculadas en una carpeta con la fecha y hora actual.



```
Matriz matriz-0: Área -> 999.0
Matriz matriz-1: Área -> 1438.0
Matriz matriz-2: Área -> 1185.0
Matriz matriz-3: Área -> 928.0
```

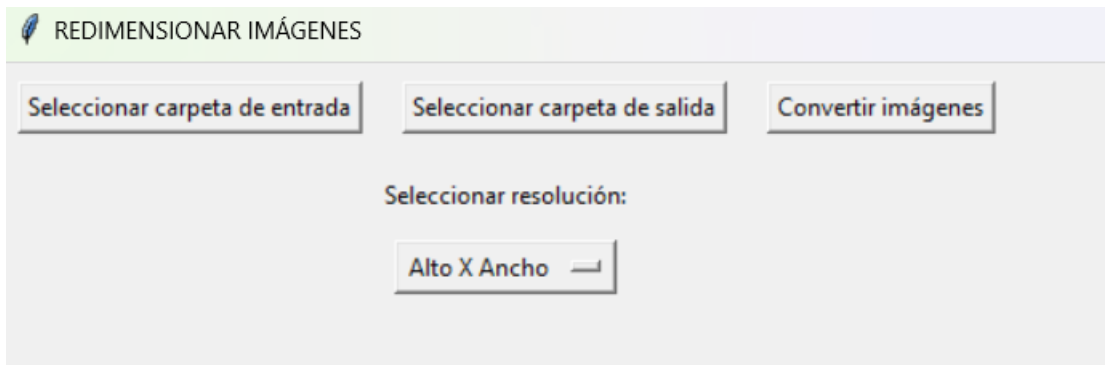
Librerías Necesarias:

- ``tkinter``: Para la creación de la interfaz gráfica y la interacción con el usuario.
- ``numpy``: Para el manejo de datos y operaciones matemáticas.
- ``PIL`` (Image): Para la manipulación de imágenes.
- ``cv2`` (OpenCV): Para el procesamiento de imágenes y cálculo de áreas.
- ``os``: Para operaciones relacionadas con el sistema operativo.
- ``datetime``: Para obtener la fecha y hora actual.

Este programa proporciona una manera rápida de calcular y guardar áreas de objetos en imágenes contenidas en archivos ``dat.npz``. Las áreas se guardan junto con imágenes procesadas que muestran los contornos de los objetos.

PROGRAMA PARA REDIMENSIONAR TAMAÑO IMAGEN

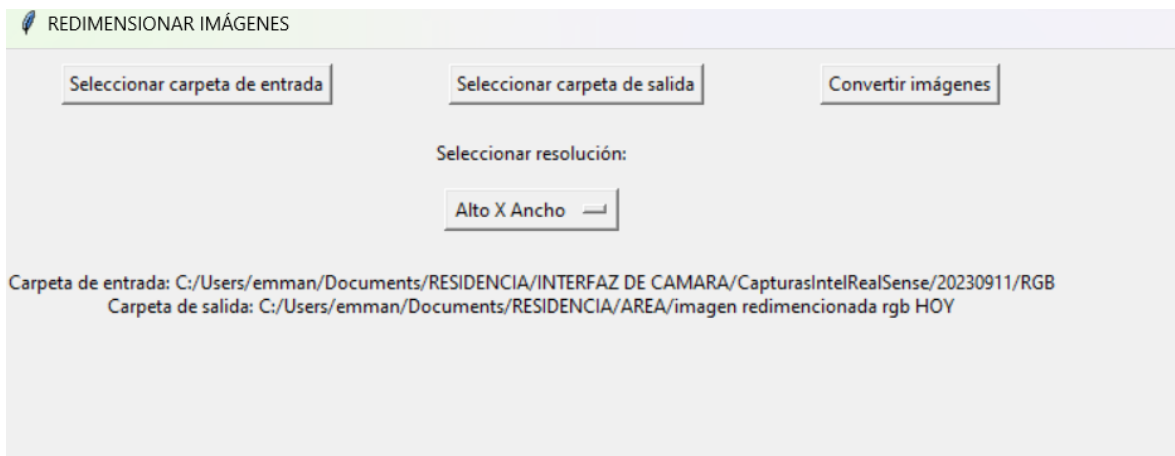
Este programa facilita la redimensión de imágenes seleccionadas y su guardado en una carpeta de salida. Aquí está el análisis de su funcionamiento:

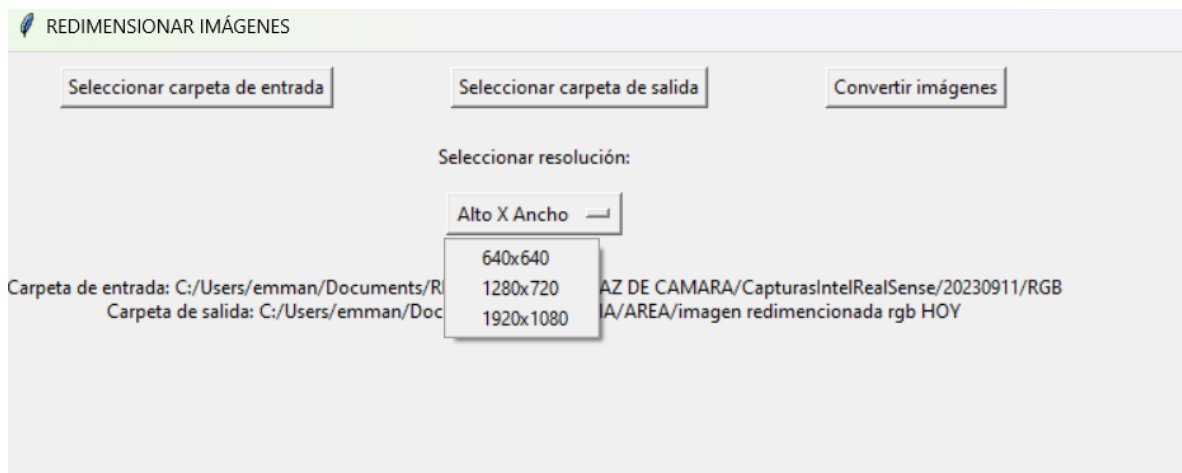


Funcionamiento General del Programa:

1. Interfaz Gráfica:

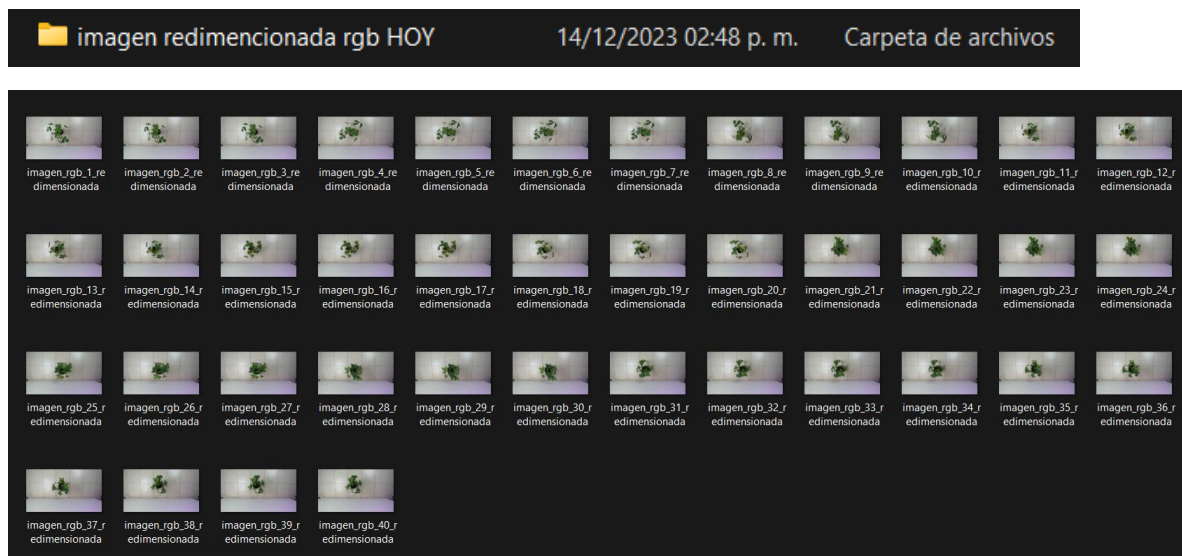
- Utiliza `tkinter` para crear una ventana con botones para seleccionar carpetas de entrada y salida.
- Ofrece una opción para elegir la resolución de salida.
- Muestra las rutas de la carpeta de entrada y la carpeta de salida seleccionadas.





2. Redimensionamiento y Guardado:

- Selecciona una carpeta de entrada que contiene imágenes en formatos `.npz`, `.png`, `.jpg`, o `.jpeg`.
- Redimensiona las imágenes a la resolución especificada y las guarda en la carpeta de salida con el mismo formato que la imagen original.



Uso del Programa:

1. Ejecución del Programa:

- Ejecuta el script en tu entorno de Python.

2. Selección de Carpetas:

- Haz clic en "Seleccionar carpeta de entrada" para elegir la carpeta que contiene las imágenes.
- Haz clic en "Seleccionar carpeta de salida" para especificar la carpeta donde se guardarán las imágenes redimensionadas.

3. Redimensionar y Guardar:

- Selecciona la resolución de salida de la lista desplegable.
- Haz clic en "Convertir Imágenes" para redimensionar y guardar las imágenes.

Librerías Necesarias:

- ``tkinter``: Para la creación de la interfaz gráfica y la interacción con el usuario.
- ``numpy``: Para el manejo de matrices y operaciones numéricas.
- ``PIL`` (Image): Para la manipulación de imágenes.
- ``os``: Para operaciones relacionadas con el sistema operativo.

Este programa ofrece una manera fácil de redimensionar y guardar imágenes en diferentes formatos, todo desde una interfaz gráfica intuitiva.

CONCLUSIONES

Este manual representa un hito esencial en nuestro proyecto de residencia, destinado a ofrecer un recurso completo y práctico para aquellos inmersos en la aplicación de tecnologías relacionadas con cámaras Intel RealSense y la visualización tridimensional, empleando herramientas desarrolladas en Python.

Nuestro objetivo radica en abrir oportunidades para que otros puedan beneficiarse de los programas meticulosamente elaborados. Desde el control de cámaras hasta la generación tridimensional de plantas, estos programas ofrecen una gama diversa de funcionalidades. Facilitan la visualización de imágenes en profundidad, la manipulación de archivos npz.dat y la representación gráfica precisa de volúmenes y áreas.

Este manual sirve como una guía completa, permitiendo no solo la utilización eficaz de programas específicos, sino también el aprovechamiento de las capacidades de las cámaras Intel RealSense y la potenciación de habilidades en procesamiento de imágenes y manipulación de datos. Además, la implementación del código para el modelo YOLOv7 amplía aún más las posibilidades, capacitando a los usuarios para la predicción y clasificación de objetos en imágenes.

Agradecemos profundamente el apoyo invaluable de nuestros asesores, cuya orientación y asistencia han sido fundamentales en el desarrollo de este recurso integral.

En resumen, este compendio no solo enseña el uso práctico de programas específicos en Python, sino que también habilita una exploración exhaustiva y eficiente de las capacidades tecnológicas y visuales en un contexto aplicado.

CONTACTOS

Emmanuel Bonilla Balbuena (1). Estudiante Tecnológico Nacional de México, Instituto Tecnológico de Tuxtla Gutiérrez. 119270548@tuxtla.tecnm.mx.

Ángel Andrés López Espinosa (2)*. Estudiante I. T. de Tuxtla Gutiérrez. 119270568@tuxtla.tecnm.mx.

Néstor Antonio Morales Navarro (3). I. T. de Tuxtla Gutiérrez, nestor.mn@tuxtla.tecnm.mx.

Madaín Pérez Patricio (4). I. T. de Tuxtla Gutiérrez, madain.pp@tuxtla.tecnm.mx.

Osvaldo Brindis Velázquez (5), I. T. de Comitán, osvaldo.bv@comitan.tecnm.mx.

*corresponding author.