Ingeniería en Robótica y Telecomunicaciones

Departamento de computación, electrónica y mecatrónica.

Course Name:
Visión en Róbotica O24-LRT4012-1

Jonathan Eliasib Rosas Tlaczani – 168399

Examen 2

17/10/24

Reconstrucción 3D de una Silla Usando una Imagen

Para este código se utilizo el Entorno Anaconda y su IDE Spyder con 5 librerías llamadas pytorch, pillow, transformers, matplotlib y open3d.

Descripción General del Código

Este código implementa un proceso de reconstrucción 3D a partir de una única imagen de una silla. Utiliza un modelo de estimación de profundidad preentrenado para generar un mapa de profundidad desde la imagen, el cual se convierte en una nube de puntos 3D. Posteriormente, se realiza un procesamiento para eliminar puntos atípicos, y finalmente se genera y visualiza una malla 3D que se puede exportar a un archivo OBJ.

Se pueden observar más comentarios en el código.

Posibles Errores

1. **Error de Archivo No Encontrado**:

Descripción: Si la ruta del archivo de imagen no es válida o el archivo no existe, el programa lanzará un FileNotFoundError. Este error puede ocurrir si se especifica una ruta incorrecta o si el archivo ha sido movido o eliminado.

Solución: Asegúrate de que la ruta de la imagen sea correcta y que el archivo exista en esa ubicación. Se ha agregado un bloque try-except para manejar este error de forma más elegante.

2. **Error en el Modelo de Estimación de Profundidad**:

Descripción: Si el modelo de estimación de profundidad no se puede cargar o si hay un problema de compatibilidad con la versión de transformers o torch, se puede generar un error al intentar realizar las predicciones.

Solución: Verifica que las bibliotecas estén actualizadas y que el modelo esté disponible en la ruta especificada.

Código

```
#%% Importar bibliotecas necesarias

import matplotlib
```

```
matplotlib.use('TkAgg')  # Establecer el backend de Matplotlib para la visualización
from matplotlib import pyplot as plt
from PIL import Image
import torch
from transformers import GLPNImageProcessor, GLPNForDepthEstimation
import numpy as np
import open3d as o3d

#%% Cargar el modelo preentrenado para estimación de profundidad
# Se instancia el extractor de características y el modelo para la estimación de profundidad
depth_feature_extractor = GLPNImageProcessor.from_pretrained("vinvino02/glpn-nyu")
depth_model = GLPNForDepthEstimation.from_pretrained("vinvino02/glpn-nyu")

#%% Cargar y preprocesar la imagen
# Ruta de la imagen de entrada
image_path = r"C:\Users\JONATHAN\Documents\SpyderCodes\Vis_ex2\Code\silla1.jpg"
# Cambiar por la ruta de la imagen
try:
    original_image = Image.open(image_path)  # Intentar cargar la imagen
except FileNotFoundError:
    print(f"Error: El archivo no fue encontrado en la ruta {image_path}")
    exit(1)  # Salir del programa si no se puede encontrar la imagen

# Redimensionar la imagen manteniendo la relación de aspecto, ajustando a múltiplos de 32
max_height = 480  # Altura máxima permitida
new_height = min(original_image.height, max_height)  # Ajustar la altura máxima
new_height -= (new_height % 32)  # Asegurar que la altura sea múltiplo de 32
```

```
new_width = int(new_height * original_image.width / original_image.height)  # Calcular nueva anchura

new_width -= new_width % 32 if new_width % 32 < 16 else new_width % 32 - 32  # Asegurar que la anchura sea múltiplo de 32


resized_image = original_image.resize((new_width, new_height))  # Redimensionar la imagen


#%% Preparar la imagen para el modelo

# Convertir la imagen al formato requerido por el modelo

model_inputs = depth_feature_extractor(images=resized_image, return_tensors="pt")


#%% Realizar predicciones utilizando el modelo de estimación de profundidad

with torch.no_grad():  # Desactivar cálculo de gradientes para mejorar el rendimiento

    model_outputs = depth_model(**model_inputs)  # Pasar la imagen al modelo

    predicted_depth_map = model_outputs.predicted_depth  # Obtener el mapa de profundidad predicho


#%% Post-procesar el mapa de profundidad predicho

# Eliminar el padding y convertir el mapa de profundidad a un formato utilizable

padding = 16  # Cantidad de padding a eliminar

depth_output = predicted_depth_map.squeeze().cpu().numpy() * 1000.0  # Escalar el mapa de profundidad a mm

depth_output = depth_output[padding:-padding, padding:-padding]  # Recortar el mapa de profundidad

cropped_image = resized_image.crop((padding, padding, resized_image.width - padding, resized_image.height - padding))  # Recortar la imagen


# Visualizar la imagen original y el mapa de profundidad predicho

fig, ax = plt.subplots(1, 2)  # Crear subgráficas
```

```
ax[0].imshow(cropped_image)  # Mostrar la imagen recortada

ax[0].tick_params(left=False, bottom=False, labelleft=False, labelbottom=False)  # Eliminar ticks

ax[1].imshow(depth_output, cmap='inferno')  # Mostrar el mapa de profundidad

ax[1].tick_params(left=False, bottom=False, labelleft=False, labelbottom=False)  # Eliminar ticks

plt.tight_layout()

plt.pause(5)


#%% Preparar la imagen de profundidad para Open3D

image_width, image_height = cropped_image.size  # Obtener dimensiones de la imagen

depth_image_scaled = (depth_output * 255 / np.max(depth_output)).astype('uint8')  # Escalar valores de profundidad a 0-255

image_array = np.array(cropped_image)  # Convertir imagen a matriz numpy


# Crear imagen RGBD en Open3D a partir de las imágenes de color y profundidad

depth_o3d_image = o3d.geometry.Image(depth_image_scaled)  # Crear imagen de profundidad

color_o3d_image = o3d.geometry.Image(image_array)  # Crear imagen de color

rgbd_image = o3d.geometry.RGBDImage.create_from_color_and_depth(color_o3d_image, depth_o3d_image, convert_rgb_to_intensity=False)  # Crear imagen RGBD


#%% Definir las intrínsecas de la cámara

camera_intrinsic = o3d.camera.PinholeCameraIntrinsic()  # Crear objeto de intrínsecas de cámara

focal_length = 500  # Definir longitud focal (ajustar según el caso)

# Configurar parámetros de la cámara

camera_intrinsic.set_intrinsics(image_width, image_height, focal_length, focal_length, image_width / 2, image_height / 2)
```

```
#%% Crear nube de puntos a partir de la imagen RGBD

raw_point_cloud = o3d.geometry.PointCloud.create_from_rgbd_image(rgbd_image, camera_intrinsic)  # Crear nube de puntos a partir de la imagen RGBD


# Visualizar la nube de puntos sin procesar

o3d.visualization.draw_geometries([raw_point_cloud])


#%% Post-procesar la nube de puntos 3D

# Eliminar outliers de la nube de puntos

clustering_indices, inliers_indices = raw_point_cloud.remove_statistical_outlier(nb_neighbors=20, std_ratio=20.0)  # Eliminar outliers

filtered_point_cloud = raw_point_cloud.select_by_index(inliers_indices)  # Seleccionar solo inliers


# Estimar normales para la nube de puntos filtrada

filtered_point_cloud.estimate_normals()  # Estimar normales

filtered_point_cloud.orient_normals_to_align_with_direction()  # Orientar normales para mejorar la generación de malla


# Visualizar la nube de puntos procesada

o3d.visualization.draw_geometries([filtered_point_cloud])


#%% Generar una malla a partir de la nube de puntos

mesh = o3d.geometry.TriangleMesh.create_from_point_cloud_poisson(filtered_point_cloud, depth=10, n_threads=1)[0]  # Crear malla a partir de la nube de puntos


# Rotar la malla para una mejor visualización
```

```
rotation_matrix = mesh.get_rotation_matrix_from_xyz((np.pi, 0, 0))  # Rotar malla 180 grados en el eje X

mesh.rotate(rotation_matrix, center=(0, 0, 0))  # Aplicar rotación


# Visualizar la malla generada

o3d.visualization.draw_geometries([mesh], mesh_show_back_face=True)


#%% Exportar la malla generada a un archivo

output_mesh_path = 'C:/Users/JONATHAN/Documents/SpyderCodes/Vis_ex2/RESULTSMESH/office.obj'

o3d.io.write_triangle_mesh(output_mesh_path, mesh)  # Guardar la malla en el archivo especificado
```

Resultados

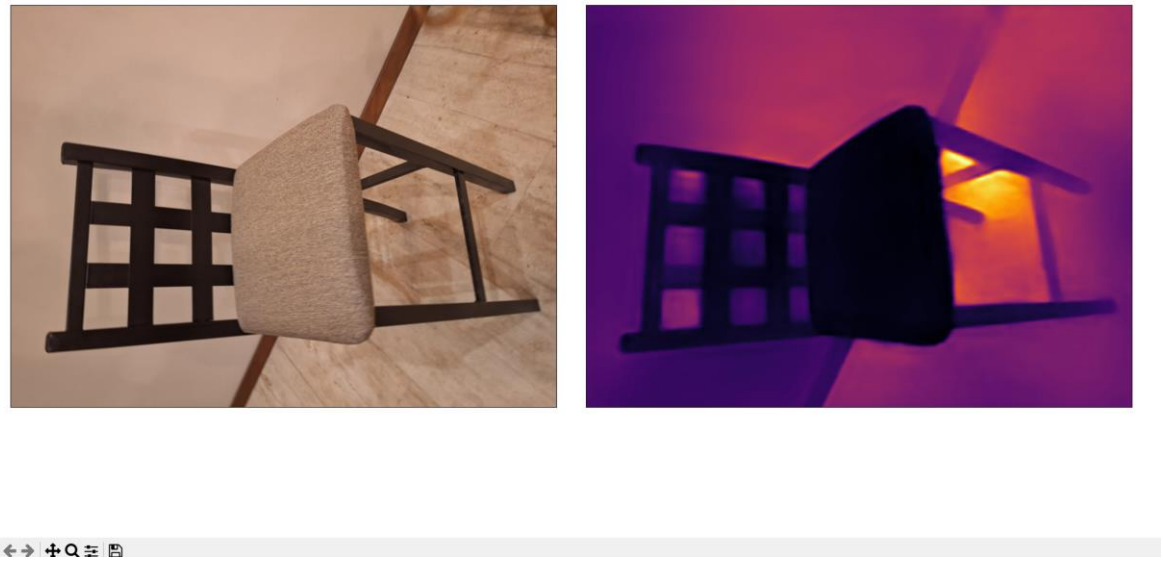

Figura 1. Silla.

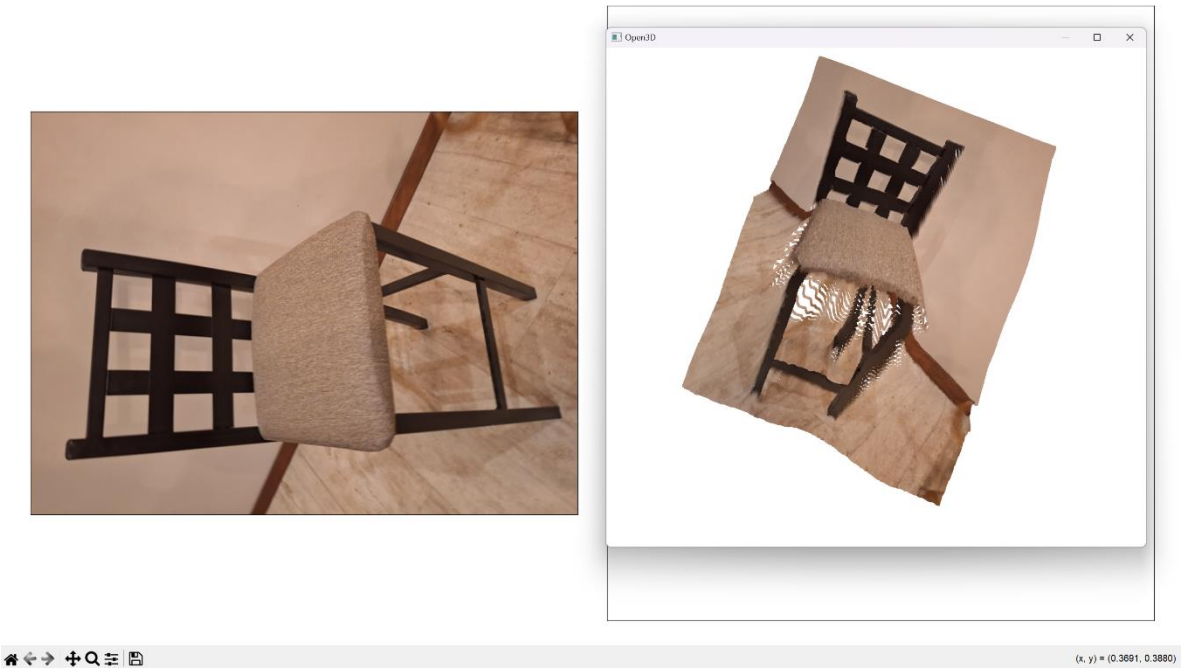

Figura 2. Impresión de silla de frente.

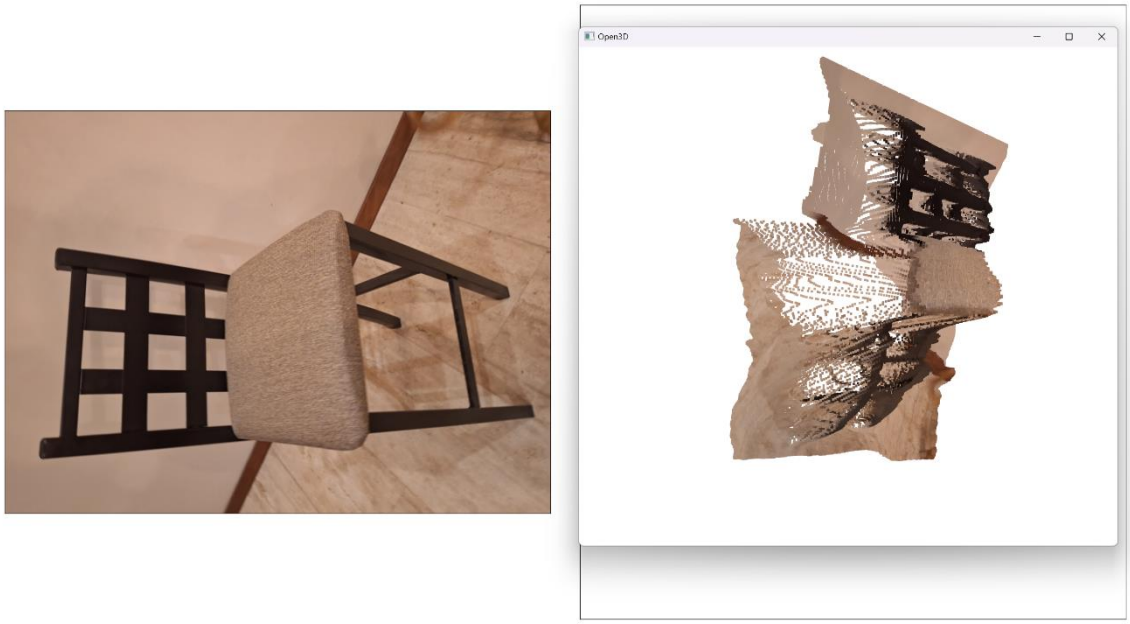

Figura 3. Impresión de silla hacia la derecha.

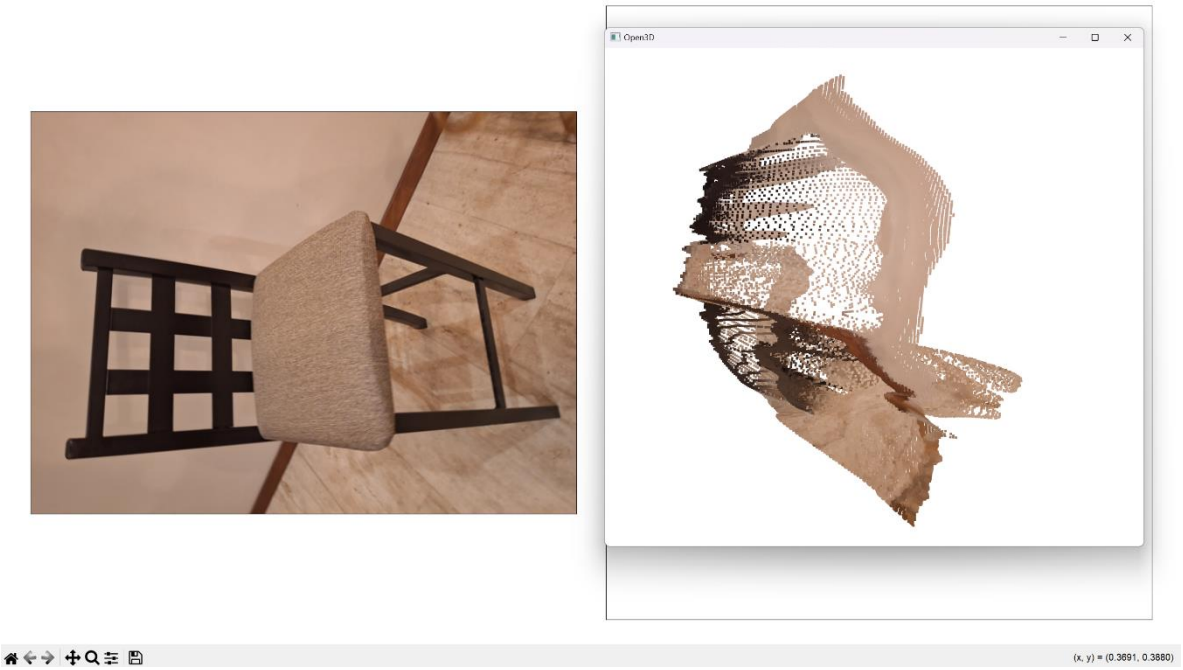

Figura 4. Impresión de silla hacia la izquierda.

Conclusión

El código presentado proporciona una solución efectiva para la reconstrucción 3D a partir de una sola imagen de una silla. A través de un modelo de estimación de profundidad y técnicas de procesamiento de nubes de puntos, se logra obtener una representación 3D de la silla. Sin embargo, es fundamental manejar adecuadamente los posibles errores que pueden surgir durante la ejecución para mejorar la robustez del código.