In [ ]:
```
from rembg import remove
import pyrealsense2 as rs
import numpy as np
import pickle
import time
import cv2
```

In [ ]:
```
# Load camera calibration data
with open('cameraCalibration_srcdata/calibration_data_RSd435i.pkl', 'rb') as f:
    calibration_data = pickle.load(f)
    cameraMatrix = calibration_data["cameraMatrix"]
    dist = calibration_data["dist"]

# Initialize the camera and the background removal model
pipe = rs.pipeline()
cfg = rs.config()

cfg.enable_stream(rs.stream.color, 640, 360, rs.format.bgr8, 30)
cfg.enable_stream(rs.stream.depth, 640, 360, rs.format.z16, 30)

pipe.start(cfg)

stop_display = False

# Initialize: capture the initial image and use it to generate a mask for backgr
init_frame = pipe.wait_for_frames()
init_color_frame = init_frame.get_color_frame()
if init_color_frame:
    init_color_image = np.asanyarray(init_color_frame.get_data())
    init_bg_removed = remove(
        init_color_image,
        alpha_matting=True,
        alpha_matting_foreground_threshold=240,
        alpha_matting_background_threshold=5,
        alpha_matting_erode_size=15,
        alpha_matting_base_size=1000,
    )
    # Extract the Alpha channel from the RGBA image as the mask
    if init_bg_removed.shape[2] == 4:
        init_mask = init_bg_removed[:, :, 3]
        init_mask = cv2.cvtColor(init_mask, cv2.COLOR_GRAY2BGR)
        init_mask = init_mask.astype(np.uint8)

# Precompute the undistortion map
h, w = init_color_image.shape[:2]
newCameraMatrix, roi = cv2.getOptimalNewCameraMatrix(cameraMatrix, dist, (w, h),
map1, map2 = cv2.initUndistortRectifyMap(cameraMatrix, dist, None, newCameraMatr

# Initialize variables for FPS calculation
fps = 0
frame_count = 0
start_time = time.time()

while not stop_display:
    frame = pipe.wait_for_frames()
    color_frame = frame.get_color_frame()
    depth_frame = frame.get_depth_frame()
```

```
    if not color_frame:
        continue
    color_image = np.asanyarray(color_frame.get_data())
    depth_image = np.asanyarray(depth_frame.get_data())

    bg_removed = cv2.bitwise_and(color_image, color_image, mask=init_mask[:, :,

    depth_cm = cv2.applyColorMap(cv2.convertScaleAbs(depth_image, alpha=0.5), cv

    alpha = np.sum(bg_removed, axis=-1) > 0
    alpha = np.uint8(alpha * 255)
    bg_removed = np.dstack((bg_removed, alpha))

    h, w = bg_removed.shape[:2]
    newCameraMatrix, roi = cv2.getOptimalNewCameraMatrix(cameraMatrix, dist, (w,
    bg_removed = cv2.undistort(bg_removed, cameraMatrix, dist, None, newCameraMa

    # Calculate and display FPS
    frame_count += 1
    elapsed_time = time.time() - start_time
    if elapsed_time > 1:
        fps = frame_count / elapsed_time
        frame_count = 0
        start_time = time.time()

    # Display FPS on the image
    cv2.putText(bg_removed, f"FPS: {fps:.2f}", (10, 30), cv2.FONT_HERSHEY_SIMPLE

    cv2.imshow('bg_removed', bg_removed)

    saved = False
    if not saved:
        saved = True
        cv2.imwrite('bg_removed2.png', bg_removed)

    key = cv2.waitKey(1)
    if key == ord('q'):
        stop_display = True

pipe.stop()
cv2.destroyAllWindows()
```