W = 848

H = 480

# Configure depth and color streams

pipeline = rs.pipeline()

config = rs.config()

# config.enable\_device('220222066259')

config.enable\_device('151322061880')

config.enable\_stream(rs.stream.depth, W, H, rs.format.z16, 30)

config.enable\_stream(rs.stream.color, W, H, rs.format.bgr8, 30)

# print("[INFO] start streaming...")

pipeline.start(config)

aligned\_stream = rs.align(rs.stream.color) # alignment between color and depth

point\_cloud = rs.pointcloud()

# get a stream of images

while True:

# for i in range(50):

# ----- added from other method

current\_pose = fa.get\_pose()

frames = pipeline.wait\_for\_frames()

frames = aligned\_stream.process(frames)

color\_frame = frames.get\_color\_frame()

depth\_frame = frames.get\_depth\_frame().as\_depth\_frame()

points = point\_cloud.calculate(depth\_frame)

verts = np.asanyarray(points.get\_vertices()).view(np.float32).reshape(-1, W, 3) # xyz

# Convert images to numpy arrays

depth\_image = np.asanyarray(depth\_frame.get\_data())

# skip empty frames

if not np.any(depth\_image):

print("no depth")

# continue

# print("\n[INFO] found a valid depth frame")

color\_image = np.asanyarray(color\_frame.get\_data())

bw\_image = cv2.cvtColor(color\_image, cv2.COLOR\_BGR2GRAY)

# camera parameters [fx, fy, cx, cy]

cam\_param = [realsense\_intrinsics.fx, realsense\_intrinsics.fy, realsense\_intrinsics.cx, realsense\_intrinsics.cy]

cv2.imshow("Image", color\_image)

k=cv2.waitKey(1)

if(k==27):

# Wait for the next set of frames from the camera

colorized = colorizer.process(frames)

# Create save\_to\_ply object

ply = rs.save\_to\_ply("1.ply")

# Set options to the desired values

# In this example we'll generate a textual PLY with normals (mesh is already created by default)

ply.set\_option(rs.save\_to\_ply.option\_ply\_binary, False)

ply.set\_option(rs.save\_to\_ply.option\_ply\_normals, True)

print("Saving to 1.ply...")

# Apply the processing block to the frameset which contains the depth frame and the texture

ply.process(colorized)

print("Done")