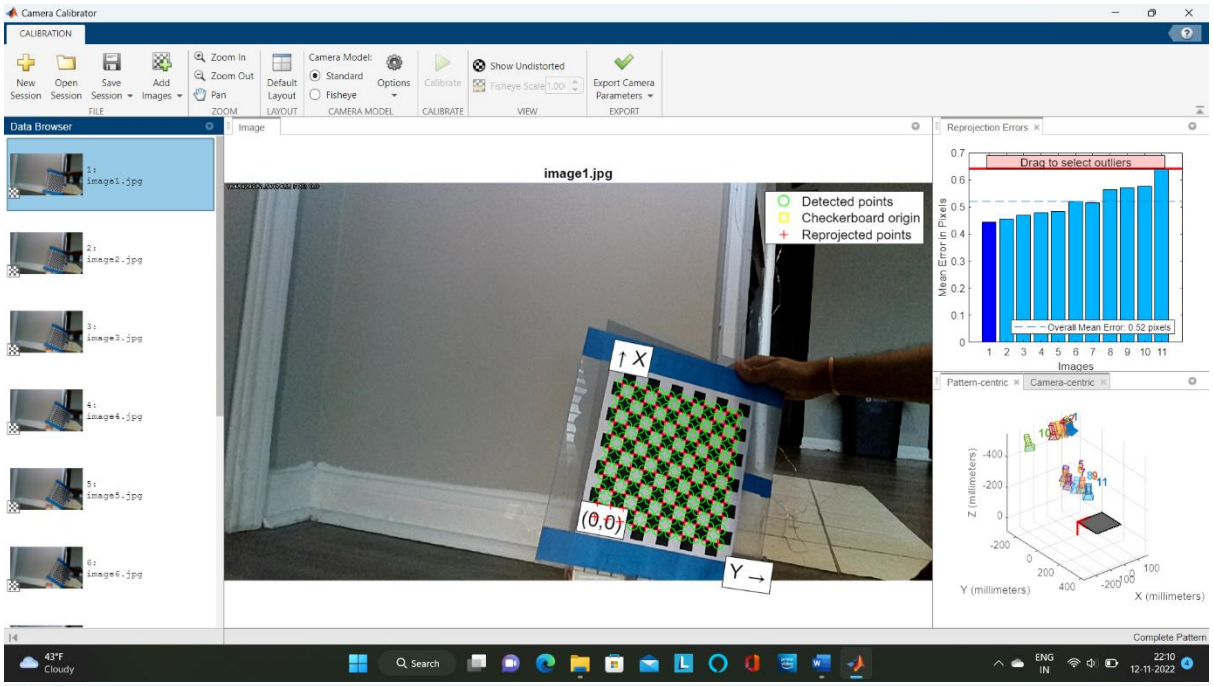
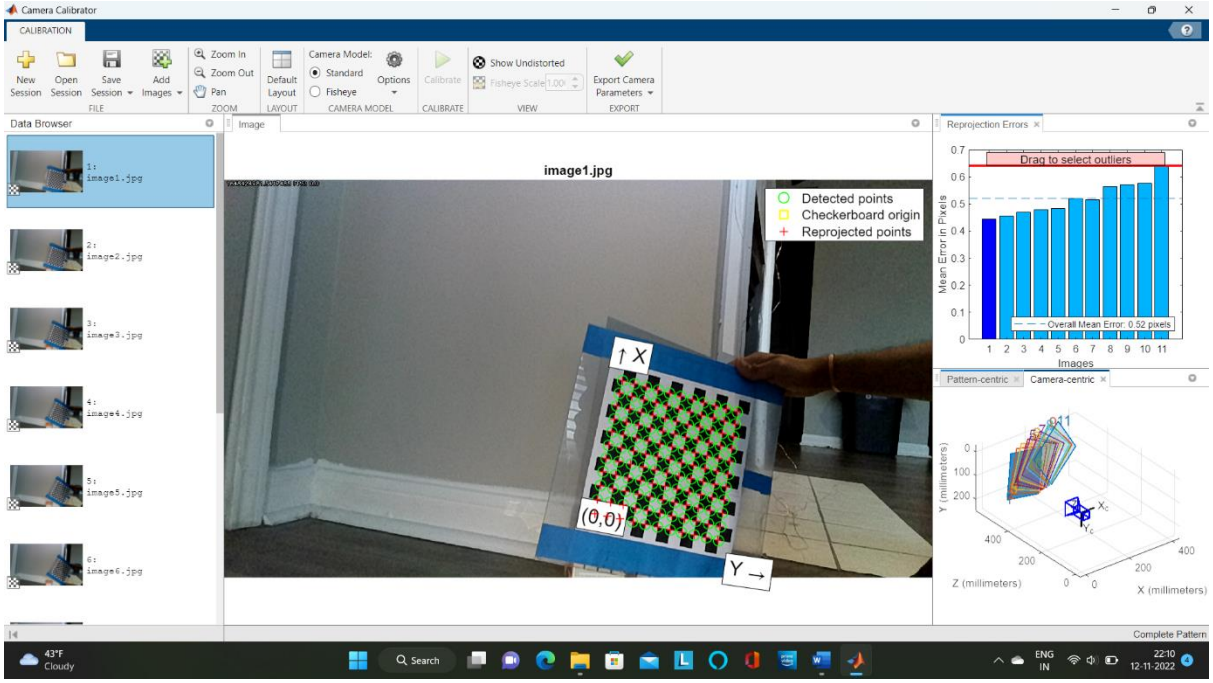
CV-ASSIGNMENT-1

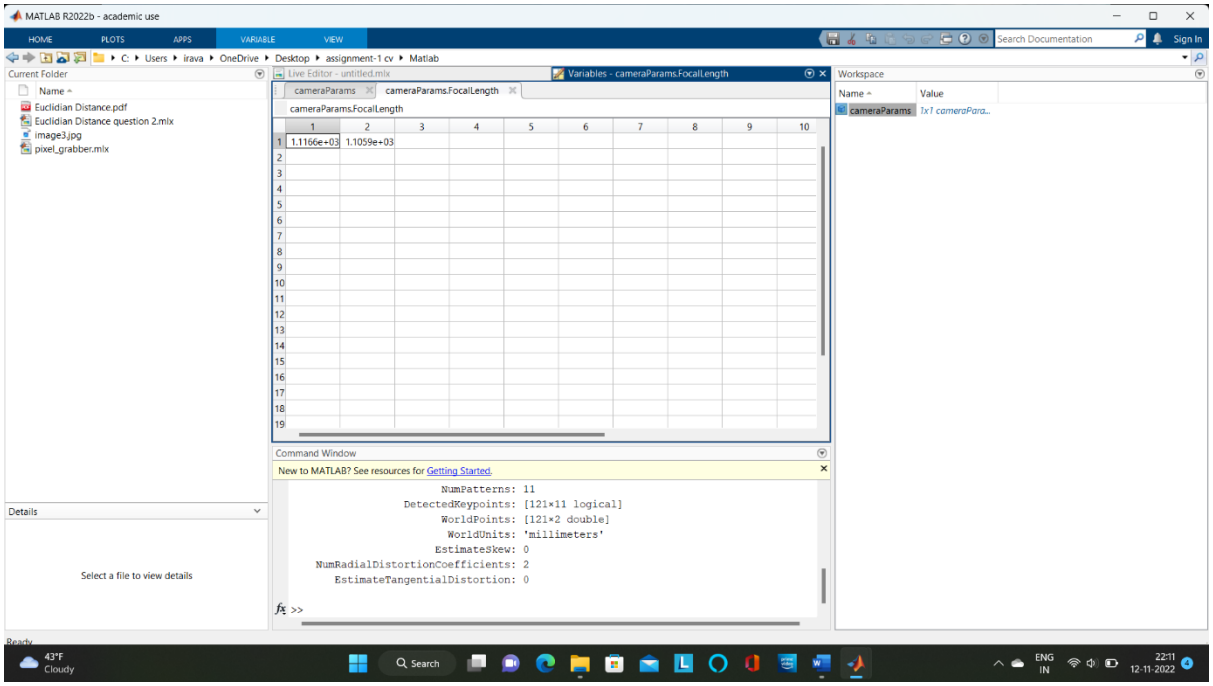
Q1:

```
import cv2
import depthai as dai
import time
from depthai_sdk.fps import FPSHandler
# Create pipeline
pipeline = dai.Pipeline()

# Define source and output
camRgb = pipeline.create(dai.node.ColorCamera)
xoutVideo = pipeline.create(dai.node.XLinkOut)

xoutVideo.setStreamName("video")

# Properties
camRgb.setBoardSocket(dai.CameraBoardSocket.RGB)
camRgb.setResolution(dai.ColorCameraProperties.SensorResolution.THE_1080_P)
camRgb.setVideoSize(1280,720)

xoutVideo.input.setBlocking(False)
xoutVideo.input.setQueueSize(1)

# Linking
camRgb.video.link(xoutVideo.input)
# Connect to device and start pipeline
start_time = time.time()
x = 1
counter = 0
```

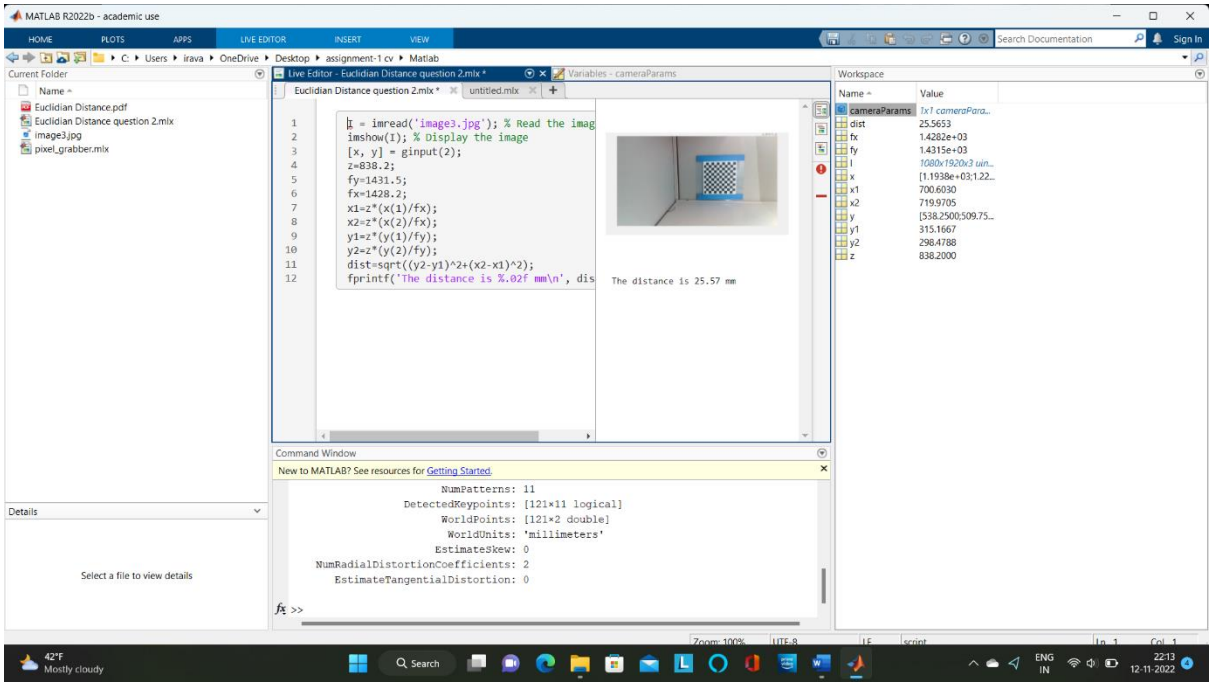
```
count=0
with dai.Device(pipeline) as device:

    video = device.getOutputQueue(name="video", maxSize=1, blocking=False)

    while True:
        videoIn = video.get()
        Frame=videoIn.getCvFrame()
        counter+=1
        font = cv2.FONT_HERSHEY_SIMPLEX
        if (time.time() - start_time) >= 1 :
                fps=counter
                counter = 0
                start_time = time.time()
        cv2.putText(Frame, str(fps)+' '+str(camRgb.getVideoSize()), (7, 70), font, 1, (100, 255,
0), 1, cv2.LINE_AA)
        cv2.imshow("video", Frame)
        if cv2.waitKey(1)==ord('p'):
            count+=1
            cv2.imwrite('image'+str(count)+'.jpg',Frame)
        if cv2.waitKey(1) == ord('q'):
            break
```


Camera Calibrator
CALIBRATION
New Session
Open Session
Save Session
Add Images
Zoom In
Zoom Out
Pan
Default Layout
Camera Model:
Standard
Fisheye
Options
Calibrate
Show Undistorted
Fisheye Scale
Export Camera Parameters
Data Browser
1: image1.jpg
2: image2.jpg
3: image3.jpg
4: image4.jpg
5: image5.jpg
6: image6.jpg
Image
image1.jpg
Detected points
Checkerboard origin
Reprojected points
↑ X
(0,0)
Y →
Reprojection Errors
Drag to select outliers
Mean Error in Pixels
Overall Mean Error: 0.52 pixels
Images
Pattern-centric
Camera-centric
Z (millimeters)
Y (millimeters)
X (millimeters)
Complete Pattern
43°F
Cloudy
Search
ENG
IN
22:10
12-11-2022


Camera Calibrator
CALIBRATION
New Session
Open Session
Save Session
Add Images
Zoom In
Zoom Out
Pan
Default Layout
Camera Model:
Standard
Fisheye
Options
Calibrate
Show Undistorted
Fisheye Scale
Export Camera Parameters
Data Browser
1: image1.jpg
2: image2.jpg
3: image3.jpg
4: image4.jpg
5: image5.jpg
6: image6.jpg
Image
image1.jpg
Detected points
Checkerboard origin
Reprojected points
↑ X
(0,0)
Y →
Reprojection Errors
Drag to select outliers
Mean Error in Pixels
Overall Mean Error: 0.52 pixels
Images
Pattern-centric
Camera-centric
Y (millimeters)
Z (millimeters)
X (millimeters)
Complete Pattern
43°F
Cloudy
Search
ENG
IN
22:10
12-11-2022


MATLAB R2022b - academic use
Variables - cameraParams.FocalLength
cameraParams.FocalLength
1.1166e+03 1.1059e+03
Workspace
cameraParams 1x1 cameraPara...
Command Window
New to MATLAB? See resources for Getting Started.
NumPatterns: 11
DetectedKeypoints: [121×11 logical]
WorldPoints: [121×2 double]
WorldUnits: 'millimeters'
EstimateSkew: 0
NumRadialDistortionCoefficients: 2
EstimateTangentialDistortion: 0
Select a file to view details

Q2:


MATLAB R2022b - academic use
Live Editor - Euclidian Distance question 2.mlx *
I = imread('image3.jpg'); % Read the imag
imshow(I); % Display the image
[x, y] = ginput(2);
z=838.2;
fy=1431.5;
fx=1428.2;
x1=z*(x(1)/fx);
x2=z*(x(2)/fx);
y1=z*(y(1)/fy);
y2=z*(y(2)/fy);
dist=sqrt((y2-y1)^2+(x2-x1)^2);
fprintf('The distance is %.02f mm\n', dis
The distance is 25.57 mm
Workspace
dist 25.5653
fx 1.4282e+03
fy 1.4315e+03
x1 700.6030
x2 719.9705
y1 315.1667
y2 298.4788
z 838.2000
Command Window
NumPatterns: 11
DetectedKeypoints: [121×11 logical]
WorldPoints: [121×2 double]
WorldUnits: 'millimeters'
EstimateSkew: 0
NumRadialDistortionCoefficients: 2
EstimateTangentialDistortion: 0

Q3:

```
#!/usr/bin/env python3

import cv2
import numpy as np
import depthai as dai
import time

# Weights to use when blending depth/rgb image (should equal 1.0)
rgbWeight = 0.6
depthWeight = 0.4



def updateBlendWeights(percent_rgb):
    """
    Update the rgb and depth weights used to blend depth/rgb image

    @param[in] percent_rgb The rgb weight expressed as a percentage (0..100)
    """
    global depthWeight
    global rgbWeight
    rgbWeight = float(percent_rgb)/100.0
    depthWeight = 1.0 - rgbWeight



# Optional. If set (True), the ColorCamera is downscaled from 1080p to 720p.
# Otherwise (False), the aligned depth is automatically upscaled to 1080p
downscaleColor = True
fps = 30
# The disparity is computed at this resolution, then upscaled to RGB resolution
monoResolution = dai.MonoCameraProperties.SensorResolution.THE_400_P
```

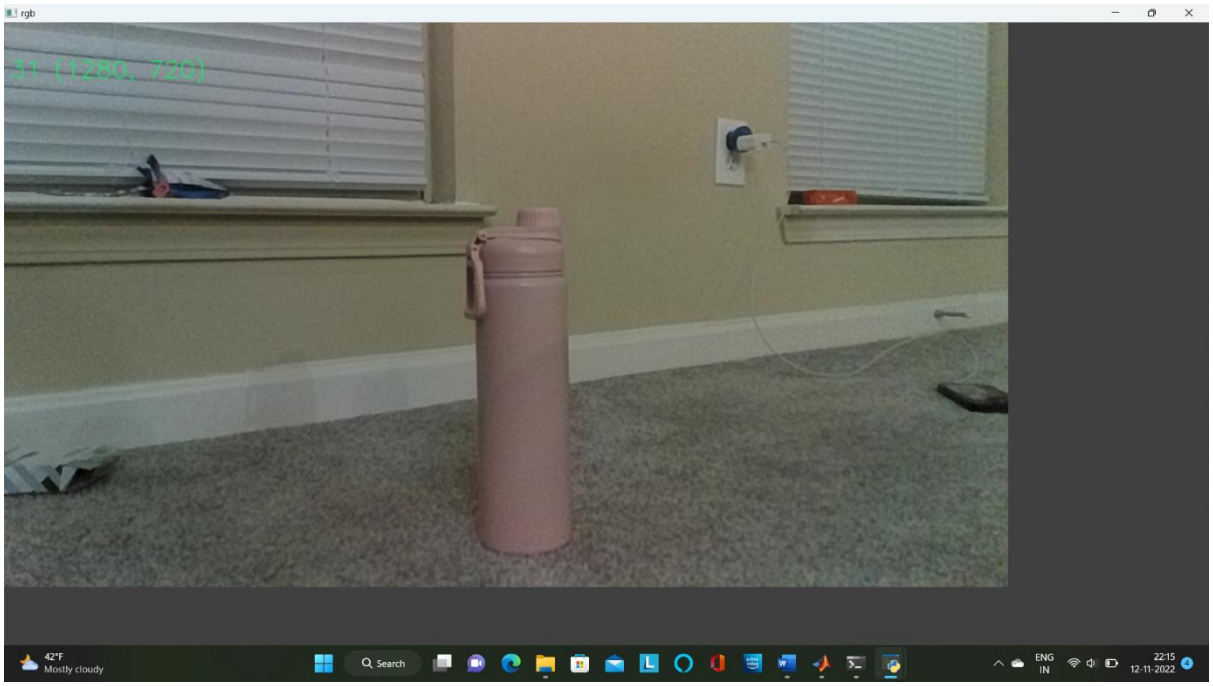
```
# Create pipeline
pipeline = dai.Pipeline()
queueNames = []

# Define sources and outputs
camRgb = pipeline.create(dai.node.ColorCamera)
left = pipeline.create(dai.node.MonoCamera)
right = pipeline.create(dai.node.MonoCamera)
stereo = pipeline.create(dai.node.StereoDepth)

rgbOut = pipeline.create(dai.node.XLinkOut)
depthOut = pipeline.create(dai.node.XLinkOut)

rgbOut.setStreamName("rgb")
queueNames.append("rgb")
depthOut.setStreamName("depth")
queueNames.append("depth")

#Properties
camRgb.setBoardSocket(dai.CameraBoardSocket.RGB)
camRgb.setResolution(dai.ColorCameraProperties.SensorResolution.THE_1080_P)
camRgb.setFps(fps)
if downscaleColor: camRgb.setIspScale(2, 3)
# For now, RGB needs fixed focus to properly align with depth.
# This value was used during calibration
camRgb.initialControl.setManualFocus(130)

left.setResolution(monoResolution)
left.setBoardSocket(dai.CameraBoardSocket.LEFT)
left.setFps(fps)
```

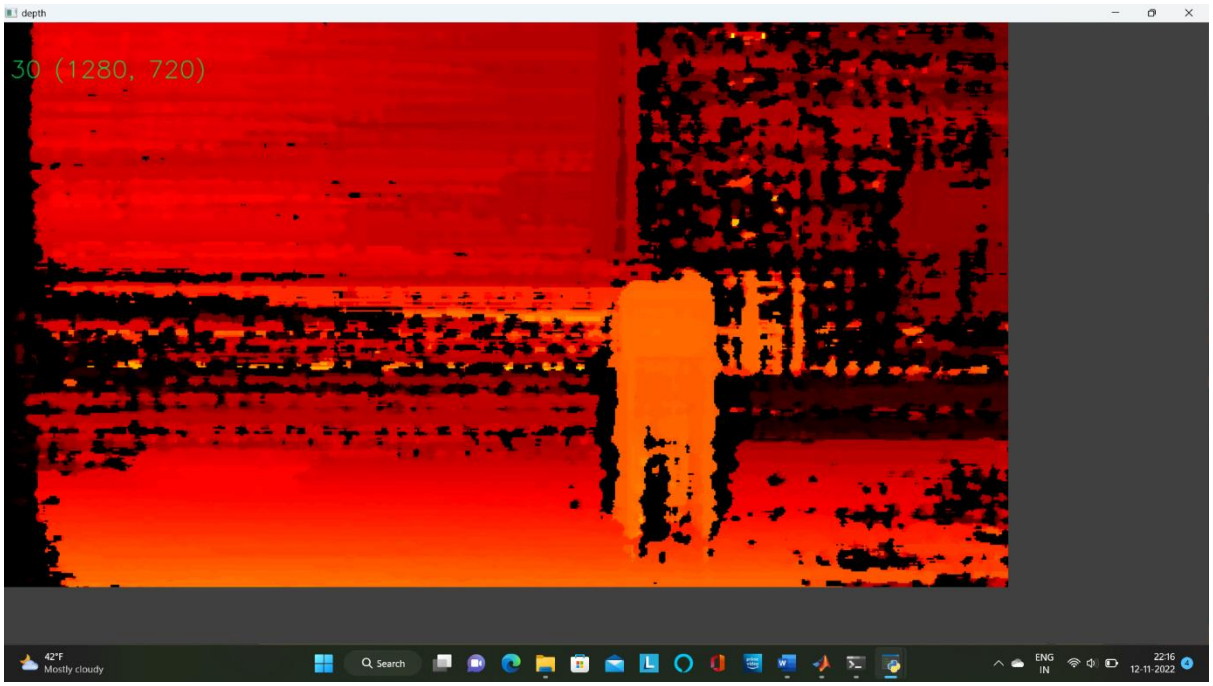
```
right.setResolution(monoResolution)
right.setBoardSocket(dai.CameraBoardSocket.RIGHT)
right.setFps(fps)

#stereo.setDefaultProfilePreset(dai.node.StereoDepth.PresetMode.HIGH_DENSITY)
# LR-check is required for depth alignment
stereo.setLeftRightCheck(True)
stereo.setDepthAlign(dai.CameraBoardSocket.RGB)

# Linking
camRgb.isp.link(rgbOut.input)
left.out.link(stereo.left)
right.out.link(stereo.right)
stereo.disparity.link(depthOut.input)
start_time1 = time.time()
start_time2 = time.time()
start_time3 = time.time()
x = 1
counter1 = 0
counter2 = 0
counter3 = 0
# Connect to device and start pipeline
with dai.Device(pipeline) as device:

    device.getOutputQueue(name="rgb",   maxSize=4, blocking=False)
    device.getOutputQueue(name="depth", maxSize=4, blocking=False)

    frameRgb = None
    frameDepth = None
    font = cv2.FONT_HERSHEY_SIMPLEX
```

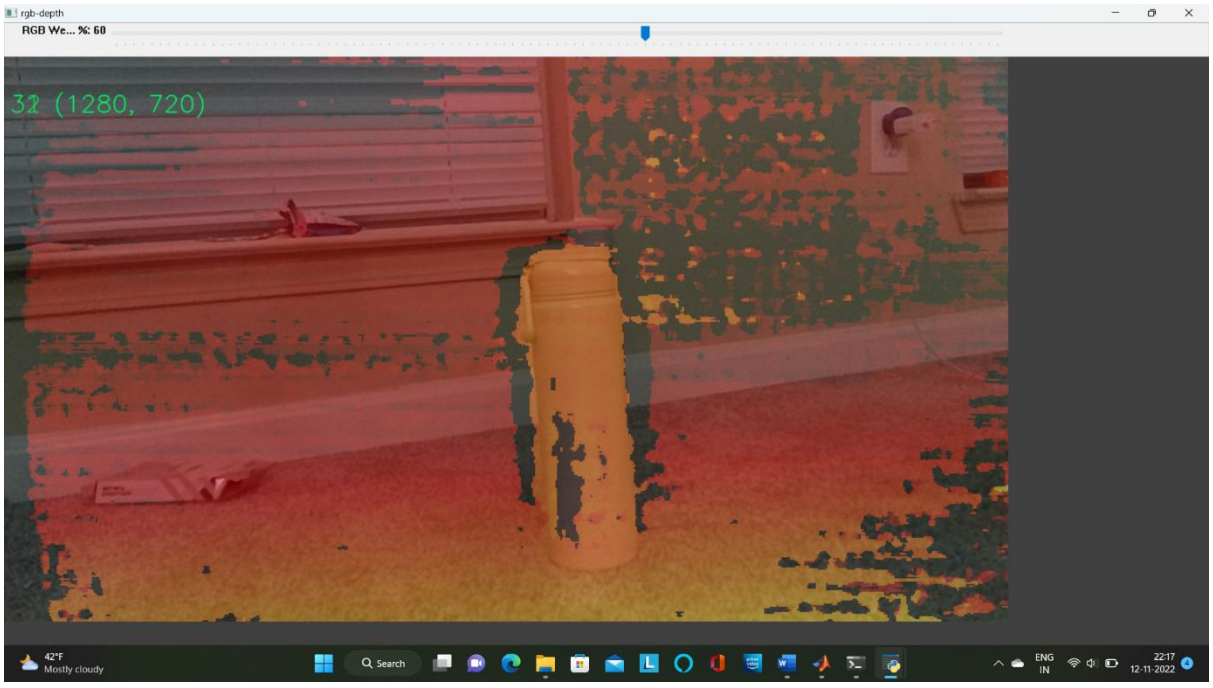
```
    # Configure windows; trackbar adjusts blending ratio of rgb/depth
    rgbWindowName = "rgb"
    depthWindowName = "depth"
    blendedWindowName = "rgb-depth"
    cv2.namedWindow(rgbWindowName)
    cv2.namedWindow(depthWindowName)
    cv2.namedWindow(blendedWindowName)
    cv2.createTrackbar('RGB Weight %', blendedWindowName, int(rgbWeight*100), 100,
updateBlendWeights)


    while True:
        counter1 += 1
        counter2 += 1
        counter3 += 1
        latestPacket = {}
        latestPacket["rgb"] = None
        latestPacket["depth"] = None


        queueEvents = device.getQueueEvents(("rgb", "depth"))
        for queueName in queueEvents:
            packets = device.getOutputQueue(queueName).tryGetAll()
            if len(packets) > 0:
                latestPacket[queueName] = packets[-1]


        if latestPacket["rgb"] is not None:
            if (time.time() - start_time1) >= 1 :
                fps1=counter1
                counter1 = 0
                start_time1 = time.time()
            frameRgb = latestPacket["rgb"].getCvFrame()
            cv2.putText(frameRgb, str(fps1)+' '+str(camRgb.getVideoSize()), (7, 70), font, 1, (100, 255, 0),
1, cv2.LINE_AA)
```

```
            cv2.imshow(rgbWindowName, frameRgb)


        if latestPacket["depth"] is not None:
            frameDepth = latestPacket["depth"].getFrame()
            maxDisparity = stereo.initialConfig.getMaxDisparity()
            # Optional, extend range 0..95 -> 0..255, for a better visualisation
            if 1: frameDepth = (frameDepth * 255. / maxDisparity).astype(np.uint8)
            # Optional, apply false colorization
            if 1: frameDepth = cv2.applyColorMap(frameDepth, cv2.COLORMAP_HOT)
            frameDepth = np.ascontiguousarray(frameDepth)
            if (time.time() - start_time2) >= 1 :
                fps2=counter2
                counter2 = 0
                start_time2 = time.time()
            cv2.putText(frameDepth, str(fps2)+' '+str(camRgb.getVideoSize()), (7, 70), font, 1, (100, 255,
0), 1, cv2.LINE_AA)
            cv2.imshow(depthWindowName, frameDepth)



        # Blend when both received
        if frameRgb is not None and frameDepth is not None:
            # Need to have both frames in BGR format before blending
            if len(frameDepth.shape) < 3:
                frameDepth = cv2.cvtColor(frameDepth, cv2.COLOR_GRAY2BGR)
            blended = cv2.addWeighted(frameRgb, rgbWeight, frameDepth, depthWeight, 0)
            if (time.time() - start_time3) >= 1 :
                fps3=counter3
                counter3 = 0
                start_time3 = time.time()
            cv2.putText(blended, str(fps3)+' '+str(camRgb.getVideoSize()), (7, 70), font, 1, (100, 255, 0), 1,
cv2.LINE_AA)
            cv2.imshow(blendedWindowName, blended)
```

```
    frameRgb = None
    frameDepth = None


if cv2.waitKey(1) == ord('q'):
    break
```




depth
30 (1280, 720)
42°F
Mostly cloudy
Search
ENG
IN
22:16
12-11-2022


rgb-depth
RGB We... %: 60
32 (1280, 720)
42°F
Mostly cloudy
Search
ENG
IN
22:17
12-11-2022