CS 6476 Project 5

Name Shen-Yi Cheng
GT Email scheng98@gatech.edu
GT ID 903514405

Part 1: (8 points)

Briefly describe your understanding about the pipeline of mediapipe's objectron detection. Describe the stages and there required input/outputs

<text answer here>

Part 1

Used cv2 to import image

Converted to RGB from BGR

Detect 3d box and bounding

Part 2

Measured the size of chair to get 3d world vertices of chair

Calculated the intrinsic matrix and camera pose of the photo

Constructed the scene of objects and camera 3d location

Part 3

Used mediapipe to estimate human pose in the photo and detect 25 landmarks

Part 4

Used the estimate depth of photo scene to project 2d to 3d

Part 5

Checked the landmarks detected in part 4 are inside or outside the vertices space or not

Part 1: (4 points)

Is it possible to recover a single 3D point from a 2D point of a monocular image (which means a single image taken by a single camera)?

<text answer here>

It is possible to recover a single 3D point from a 2d point of an image as long as we provide the estimate depth of the scene (depth calibration by Nicolas Burrus http://nicolas.burrus.name/index.php/Research/KinectCalibration)

Part 1: (4 points)

Why is it possible to estimate a 3D object from a monocular image (like mediapipe's objectron)? What other assumptions or data is needed to accomplish this.

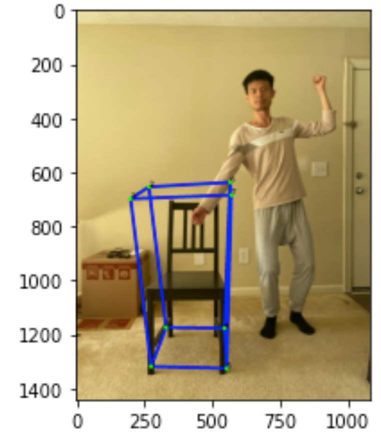
<text answer here>

By building a novel annotation tool for use with AR session data, which allows annotators to quickly label 3D bounding boxes for objects. This tool uses a split-screen view to display 2D video frames on which are overlaid 3D bounding boxes on the left, alongside a view showing 3D point clouds, camera positions and detected planes on the right. Annotators draw 3D bounding boxes in the 3D view, and verify its location by reviewing the projections in 2D video frames. For static objects, we only need to annotate an object in a single frame and propagate its location to all frames using the ground truth camera pose information from the AR session data, which makes the procedure highly efficient. Another popular approach is to complement real-world data with synthetic data in order to increase the accuracy of prediction.

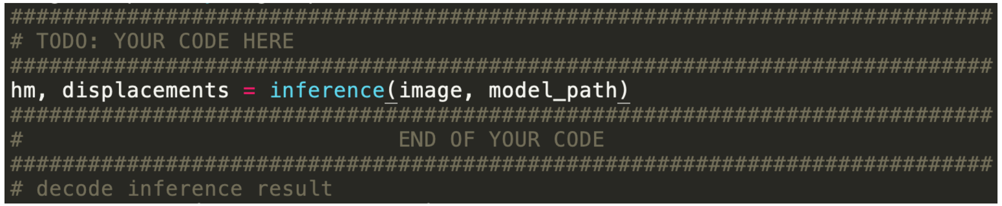
Part 1: (4 points)

Put your annotated picture generated from part 1 here

```
[[1, 3, 640, 480]] [[1, 16, 40, 30], [1, 1, 40, 30]]
0.9636402130126953
```



Copy and paste the code you fill in "detect_3d_box()" in "my_objectron.py()"
Note: Only paste the code you fill. Do not add the whole function



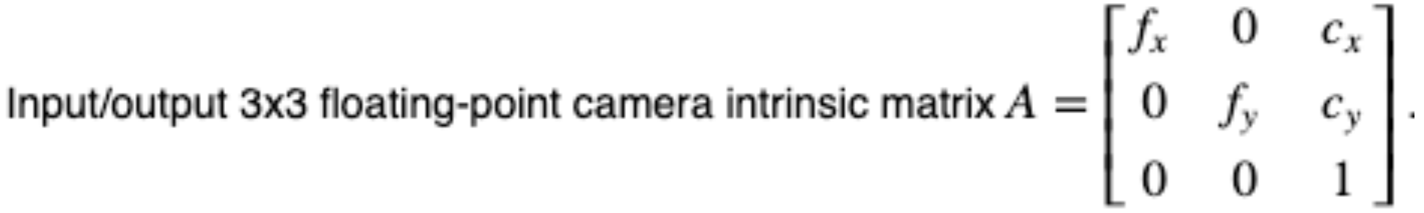
```
##########################################################################
# TODO: YOUR CODE HERE
##########################################################################
hm, displacements = inference(image, model_path)
##########################################################################
#                              END OF YOUR CODE
##########################################################################
# decode inference result
```

Part 2: (5 points)

After you did camera calibration, you get a more accurate K, the intrinsic matrix of the camera, can you describe what is the meaning of the five non-zero parameter in K?

<text answer here>

In calibration function, we use cv2.cameraCalibration to get the intrinsic and matrix. The five nonzero parameter in K are f_x, f_y, c_x, c_y and 1. The function returns the average re-projection error. This number gives a good estimation of precision of the found parameters. This should be as close to zero as possible. Given the intrinsic, distortion, rotation and translation matrices we may calculate the error for one view by using the cv::projectPoints to first transform the object point to image point. f_x and f_y are the focal length in terms of pixel. c_x and c_y are the center point of the image. The right bottom element of K should equal 1 in order to get the correct (u, v)



Input/output 3x3 floating-point camera intrinsic matrix $A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$.

Part 2: (5 points)

In the K (intrinsic matrix), there is one value representing fx and another one representing fy, what is the unit of those two values? Why? In practice, when fx is not equal to fy, what does this mean in physical?

<text answer here>

The unit of f_x and f_y is pixel. They are focal length of the camera intrinsic matrix. Because a camera matrix is used to denote a projective mapping from world coordinates to pixel coordinates. There are many reason cause fx is not equal to fy, such as flaws in the camera sensor, image has been non-uniformly scaled in post-processing, camera's lens introduces unintentional distortion.

Part 2: (10 points)

You also performed the transformation from world to camera by using the equations below.

1) Previous what we did is from world coordinate to camera coordinate. If we perform the inverse, which is from camera coordinate to world coordinate, will also have a similar equation1, but the w and c changes. Then there will be a ctw in the change equation, what does ctw represent?

2)Using the equation2 and equation3 below, can we describe why the P matrix can project 3D points in world coordinate to 2D points on image plane? (Hint: the P matrix achieves two coordinate transform)

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} {}_{\mathbf{w}}\mathbf{R}_{\mathbf{c}}^{\mathbf{T}} & -{}_{\mathbf{w}}\mathbf{R}_{\mathbf{c}}^{\mathbf{T}}\ {}_{w}\mathbf{t}_c \\ 0^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad \text{eq.1}$$

$$\mathbf{P} = \mathbf{K}\ {}^{w}\mathbf{R}_c^{\mathsf{T}} [\mathbf{I} \mid -{}^{w}\mathbf{t}_c] \quad \text{eq.2}$$

$$z \begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \quad \text{eq.3}$$

<text answer here>

1.

Ctw represent camera to world coordinate

P is the extrinsic matrix

H = K * [r1, r2, t]

R1 and r2 are the rotation matrix and t is the translation matrix

Projection = H * P (H is Hartley and Zisserman equation 8.1 in the paper)

Projnorm = projection / p(z)

2. Because in the coordinate transformation, Wx and Yw would be divided by Zw to Xp and Yp. And then add up camerr center to get the 2d coordinates on 2d image plane.

Part 3: (3 points)

Please describe an application situation for pose estimation and explain why it is useful there.

An AI fitness coach app.
During the pandemic time, we should avoid social contact. Although we still want to take some fitness course to maintain body shape, we can use virtual fitness coach app to monitor our pose during workout. Moreover, by using human pose estimation, this app can provide feedback and teaching in real-time. So it is very useful to the development of such app.

Part 3: (3 points)

If you are going to do a pose detection project, what kind of pose do you want to detect and explain why these pose are important for you.

I would like to implement a Just Dance gaming pose detection project. The original Just Dance on switch is based on the motion sensor controller. To improve the gaming experience. We should add pose detection while playing this game. By estimating user pose in real time to give the credits to users. It is more accurate to score the player when they are dancning.

Part 3: (6 points)

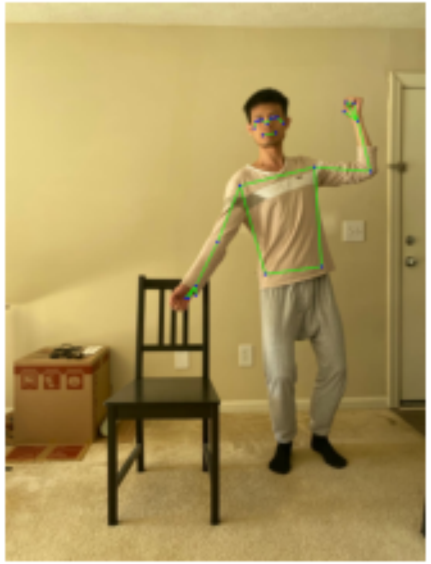
What are the two main steps associated with pose detection used in mediapipe (Hints, read the blog post of mediapipe's pose detection)?

Mediapipe utilizes a two step detector tracker machine learning pipeline, proven to be effective in Mediapipe Hands and Mediapipe Face Mesh solutions. Using a detector, the pipeline first locates the pose region-of-interest (ROI) within the frame. The tracker subsequently predicts the pose landmarks within the ROI using the ROI-cropped frame as input. Note that for video use cases the detector is invoked only as needed. The pipeline is implemented as a Mediapipe graph that uses a pose landmark subgraph from the pose landmark module and renders using a dedicated upper body pose renderer subgraph.

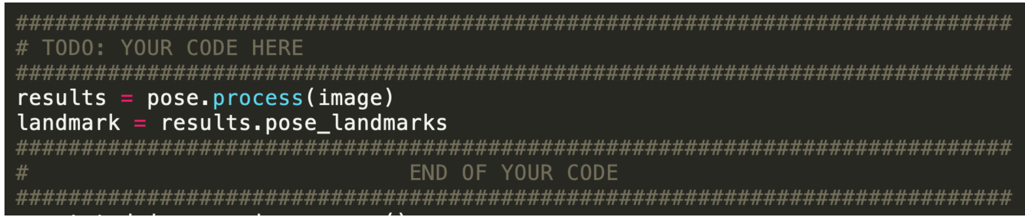
Part 3: (4 points)

Put your annotated picture after pose detection here

```
Detected landmark numbers:  25
```



Copy and paste the code you fill in "hand_pose_img()" in "pose_estimate.py"
Note: Only paste the code you fill. Do not add the whole function



```
############################################################################
# TODO: YOUR CODE HERE
############################################################################
results = pose.process(image)
landmark = results.pose_landmarks
############################################################################
#                              END OF YOUR CODE
############################################################################
```

Part 5: (4 points)

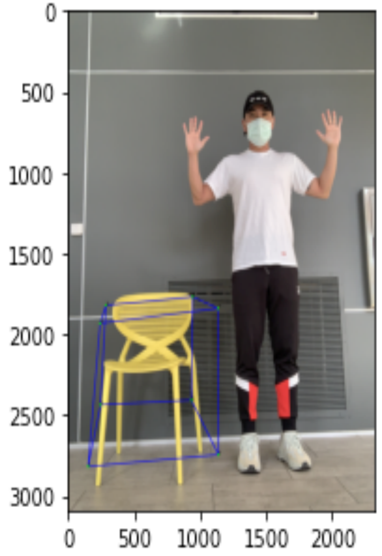
Given the 3D coordinates of eight vertices of a box in space, and one 3D point, describe how do you detect whether this point is inside or outside the box?

I use brute force to detect the point is in/out the box by finding the range of X, Y and Z coordinates. Then, I checked the 3d coordinate if all of them are inside the range.

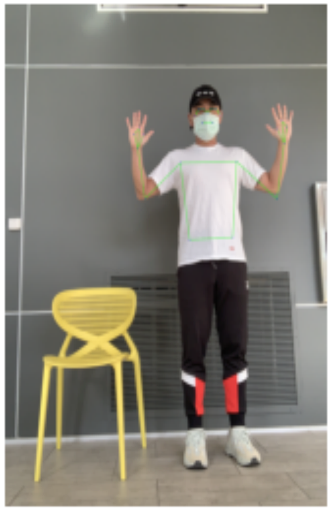
Part 6: (10 points)

Insert your picture before interacting with the object and other picture after the interaction happens (the bounding box changes color)

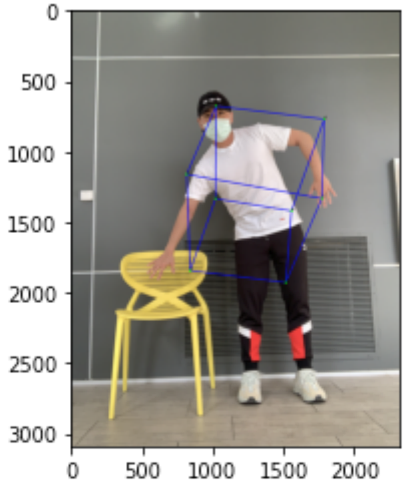
```
(3088, 2316, 3)
```



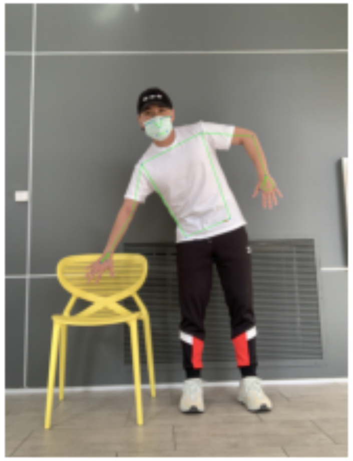
```
Detected landmark numbers:  25
```



```
(3088, 2316, 3)
```



```
Detected landmark numbers:  25
```



Extra Credit: Interaction Video

<Tell us where to access your final video of part 1 or 2, Discuss what you found out. If you have a two-chair video, you don't have to record the one-chair video again. >

The video is in the data directory.
In my result, if I intersect with the right chair, the bounding box will jump to my body just as the photo in the last slide. However, the if I intersect with the left chair, the bounding box just distort the shape and include my arms. I do not know why there are two different results. I guess maybe the hieght of the camera is not hight enough. I put my camera on a chair. It caused elevation angle to display the viedeo.

Extra Credit: Interaction Video

<Were your results shaky? If so, why/what did you have to do to fix it? >

Its kind of shaky, I guess because I transform the video format from MOV to mp4. The resolution changed. It caused the mediapipe is hard to detect the object. Also the background color may distort the detection. Next time I should stand in front of a white wall with a black chair plus sufficient light source to reduce the error.

Extra Credit: Interaction Video

< What kind of factors determined how accurate the intersection detection was?>

The resolution of the image/video
The depth of the video or image. The farther object would be compressed more.
Severe motion in the video might loss the accuracy of the intersection detection.
The objects color or background color are too close/similar might cause the accuracy problem.