CS 6476 Project 5

Sandeep Dasari
sdasari38@gatech.edu
903540744

Part 1: (8 points)

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

The model features an Encoder-Decoder architecture made popular by MobileNetv2. The network uses inverted residual blocks sandwiched between a convolutional layer and a deconvolutional layer after the bottleneck. The output of the encoder-decoder is 2-D annotated bounding boxes with 4 vertices only.

After this layer, the detection: finding the gaussian and the regression: finding 8 2-d vertices of the bounding box are performed. Finally we use the EPnP algorithms to lift the vertices from 2D to 3D co-ordinates.

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)?

Yes, but with some human help

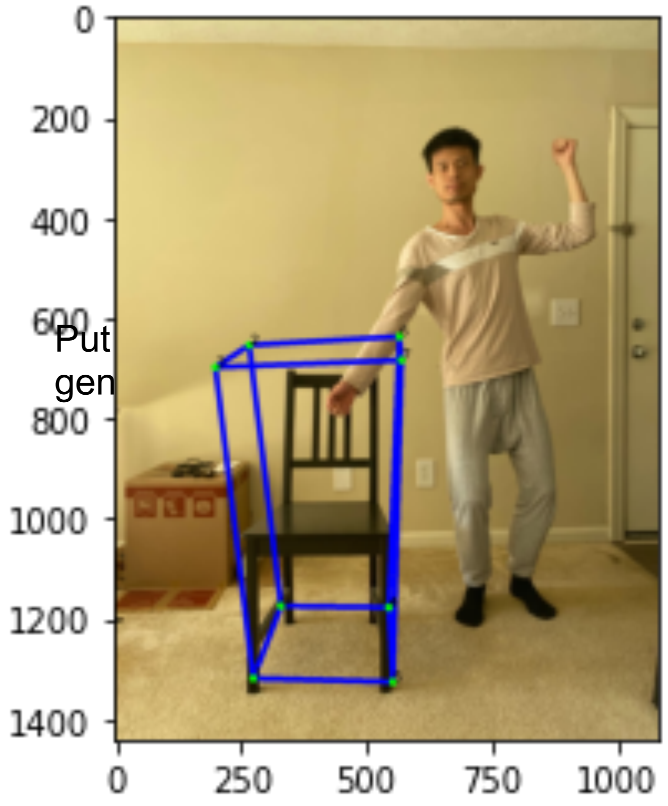
Part 1: (4 points)

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.

3D estimation of an object is usually done with multi-view geometry, but it’s possible with some prior knowledge or human input as it is considerably difficult to estimate the depth information in an image. By setting the depth externally, it is possible to estimate 3D objects from a monocular image.

We can also assume plane symmetry and surface smoothness to infer some depth information by playing with the volume of the object in the image.

Part 1: (4 points)



Put

gen

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

```
hm, displacements =
inference(image, model_path)
```

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?

The five non-zero parameter in K

1. Fx – focal length in pixels along X = F/px

2. Fy – focal length in pixels along Y = F/py

3. Cx,cy – optical center in pixels

4. Finally we have 1 at index (3,3) in the K matrix, if we change this to 0, the resultant 2-d image coordinate will be sT = [u,v,0] in homogenous coordinates is a point at infinity!

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?

fx and fy are the focal lengths in pixels. In calibration, it is essential that we convert the world measurement units (metres, cms) to camera units (pixels). Thus, they are in pixels.

fx = F/px; fx = F/py where F is focal length of the camera in metres and px , py have units of metres/pixel.

If fx is not equal to fy, it just means that the camera plane is rectangular and not a square ie. The image resolution is rectangular.

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)

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

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

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

1. ctw represents the origin of our object (in our context the chair) in the camera coordinate system
2. Eq 2 performs rotation with wrc_T while the center of the camera is wtc. The K matrix is the calibration matrix that provides for the required scaling to convert the 3D world coordinate to image coordinates.

In Eq 3, the shape of the P matrix is [3x4] while the shape of 3D homogenous points is [4x1]. The matrix product results in a matrix of shape [3x1] which is a 2D homogenous coordinate on image plane.

Part 3: (3 points)

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

Pose estimation can be heavily used in space travel and autonomous docking at space stations.

The space shuttle is aligned perfectly with the space station using the 3D pose and orientation of the station and the shuttle. In this case, pose of the station is calculated initially. Then, rotation and translation of the shuttle have to be calculated at a high frame rate to align the axes of the shuttle with the docking station.

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.

Pose detection in performing arts is an area of my interest. I would try to detect human body poses that differentiate between sitting, standing and jumping, thus using poses of the detected limbs.

The 3d projection of this data has rich possibilities in interactive art and music and is a relatively unexplored area of computer vision and multimedia.

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)?

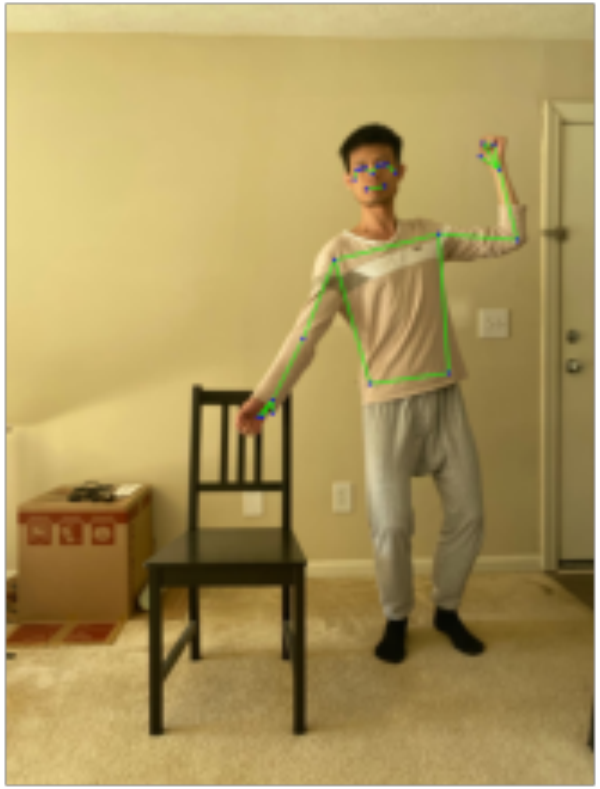
Mediapipe uses a two-stage pipeline for pose detection

1. Uses object detection to find the 2D crop of the object ie. a 2D bounding box is drawn around the image with 4 vertices
2. This stages takes the image crop and estimates the 3D bounding box

For the pose detection, two steps are taken

1. Detection: We fit a Gaussian to the 2D bounding box drawn with the center at the box centroid. We try and predict the distribution with it's peak at the center of the box
2. Regression: This step estimates the 2D projections of the eight bounding box vertices. Then we can lift these vertices into 3D using EPnP

Part 3: (4 points)



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

```
results = pose.process(image)
landmark = results.pose_landmarks
```

Part 5: (4 points)

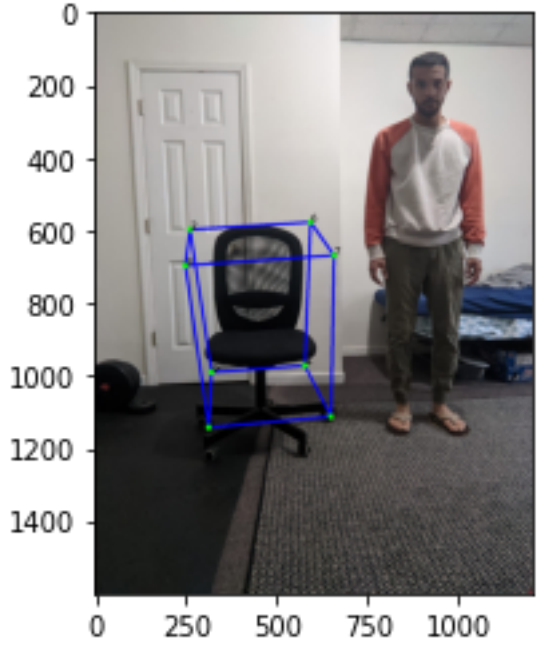
Given the 3D coordinates of eight vertices of a box in space, and one 3D point, describe how do you detect whether this point is inside or outside the box?

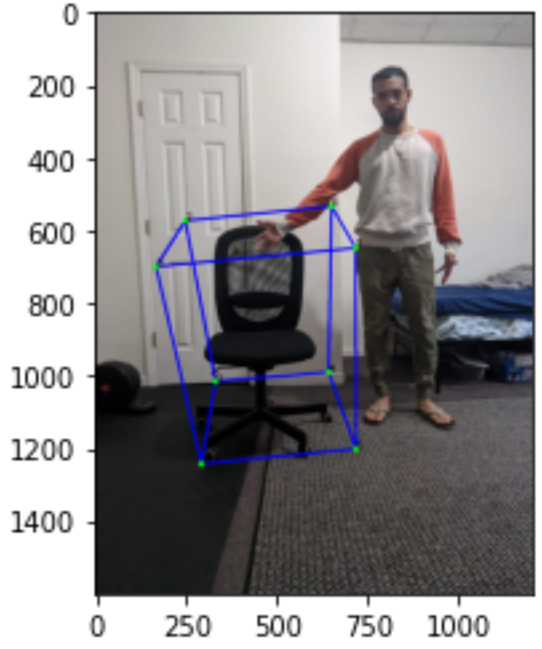
We initially establish the world frame around the bounding box of the chair. That is we lay out the origin and the direction of the axes in advance. So we now have the origin at (0,0,0) and 7 other vertices of the box in space.

The new 3D point's x, y and z coordinate values should be <= to the max of the 8 vertices in each dimension.
We use this as a checking criteria with a logical_and in numpy to to ensure

Part 6: (10 points)

The intersection here wasn't successful to detect the hand in the box. This was difficult to debug and i tried measuring the chair dimensions and the depth of the camera accurately, but the hand 3d coodinates were always outside the box.





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. >

Extra Credit: Interaction Video

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

Extra Credit: Interaction Video

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