CS x476 Project 6

Bipin Koirala
bkoirala3@gatech.edu
9037.15.285

detect_3d_box(): Code (10 points)

Please screenshot your "**detect_3d_box()**" here (only the two steps you implemented, not the whole function)

```
################################################################################
# TODO: YOUR CODE HERE
################################################################################
# Step-1: Call inference and get the result.
heatmap, displacement = inference(image,model_path)
# Step-2: Decode bounding boxes from inference result .
boxes = decode(heatmap, displacement)
################################################################################
#                                END OF YOUR CODE
################################################################################
```

Part 1: (8 points)

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

There are two pipelines to predict 3D bounding box of an object from one colored image i.e., two-stage pipeline and single-stage pipeline.

- Two-stage pipeline
  First stage: This stage uses an object detector over a still image in order to obtain the 2D crop of the object of interest.
  Second stage: It takes the cropped image as an input and predicts the 3d bounding box around it. Furthermore, it also processes the 2D crop of the object for the next successive frame. This is done to avoid redundancy of having to run objectron for every frames. Two stage pipeline is good for estimating a single dominant object.

- Single-stage pipeline
  This model has an encoder-decoder architecture and uses multi-task learning approach to jointly predict object's shape with detection and regression. Object's shape is predicted based on available ground-truth annotation, e.g., segmentation. Annotated bounding box is fitted with a Gaussian for detection task. Here the Gaussian helps in locating the peak that corresponds to the object's center point. Then the regression helps in estimating the 2D projections of the eight vertices of bounding box. Finally, the 3D coordinates for bounding box(es) are estimated with the help of EPnP algorithm.

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)?

Unlike stereo image, monocular image has less information and about the depths of fields. Therefore, we cannot recover a single 3D point from a 2D point of monocular image. To recover a 3D point we need to determine the location of camera in the space, which is lacking in the case of a monocular image.

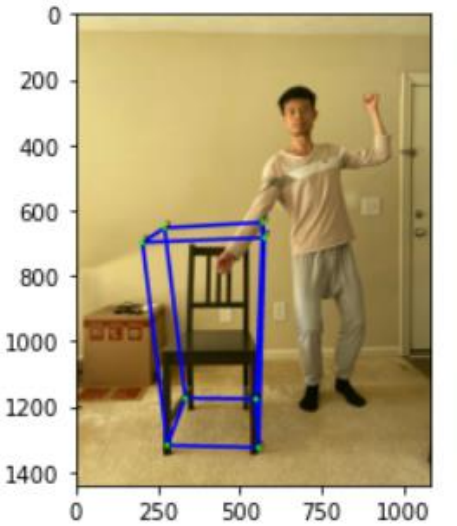
Part 1: (4 points)

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.

We can project the world coordinates to 2D image coordinates. We can reverse this technique to obtain 3D world coordinates from a 2D image pixels coordinates. This allows us to estimate the 3D object in the image frame with certain level of uncertainties.
Some assumptions needed to accomplish this results are:

- We need to have information about the depth of object i.e., distance between camera and the object.
- Furthermore, having multiple images of the object from various angles could help in a better estimation of 3D object
- Measurement of actual size of the object would result in a better estimation of 3D object as well.

Part 1: (4 points)



Put your annotated picture generated from part 1 here

See slide 2

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

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?

In utils.py, optimal intrinsic matrix of camera is obtained using the following command:

cv2.getOptimalNewCameraMatrix(mtx,dist,(w,h),1,(w,h))

where;

mtx = input camera matrix

dist = lens distortion coeffieicent (in the form of an array)

(w,h) = input image size

1 = alpha value i.e., free scaling parameter ranging from 0 to 1 corresponding to when all the pixels in the undistorted image are valid and when all the source image pixel are retained in the undistorted image respectively.

(w,h) = new image size (same size in our case)

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 imply?

In an intrinsic camera matrix, fx and fy correspond to x-axis and y-axis focal length for a pin hole geometry camera respectively. They are commonly measured in pixels (image) or millimeters (camera).

For an ideal pin hole camera, the values for both fx and fy are equal. However, in practice these values are not the same. There are numerous reasons behind this odd phenomena and they include:

- non-uniform scaling of image in post-processing
- Faulty camera calibration
- Lens distortion

Part 2: (6 points)

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

1) Previously what we did was 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 will change. 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)

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

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

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

1) The world to camera transformation is denoted by ${}^{C}t_{w}$ such that $P_c = {}^{C}t_{w} \times P_w$
2) P indeed achieves two coordinate transform i.e., world coordinate to camera coordinate and from camera coordinate to image pixel coordinate. P represents a matrix with camera intrinsics and extrinsics put together. Therefore, it first transforms 3D world coordinate to camera coordinate through extrinsic parameters which is followed by intrinsic transformation to a 2D points on image plane.

Part 2: (4 points)

According to the box and camera plot at the end of Part 2. Where is the camera? Where is it facing?

Camera center is at (1.91410572, 0.44133695, 1.40106129).

It is facing downwards (-y, -z axis at an angle with vertical)

hand_pose_img(): Code (10 points)

Please screenshot your "**hand_pose_img()**" here (only the two steps you implemented, not the whole function)

```
################################################################################
# TODO: YOUR CODE HERE
################################################################################
# Reminder: check the mediapipe documentation first!
# Step-1: Pass the image to the pose model.
pose = mp.solutions.pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5)
results = pose.process(image)
# Step-2: Get landmark from the result.
landmark = results.pose_landmarks
################################################################################
#                                END OF YOUR CODE
################################################################################
```

Part 3: (4 points)

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

>> Human pose estimation is useful in numerous applications. For example, pose estimation model can be deployed in order to monitor the activity of patients in hospital bed for their well being purpose.

Similarly, pose estimated is also useful in developing augmented reality applications such as video games, or training robots by capturing human motion. Tracking human motion helps in rendering graphics (in video games, movies etc.) or motion (in robots) that seem more natural.

Part 3: (4 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.

>> Personally, I would use **AlphaPose** estimation on tracking motion of football (soccer) players in real-time during matches and get post-match analysis on scoring chances based on the player's position in the field and their perceived momentum. With enough analysis, I would then like to build a model to predict scoring probability of a team. The probability would then help me to play safe on live betting websites which would allow me to go on a vacation to the Caribbean coasts.

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’s pose detection uses two step detector-tracker ML pipeline. The first step of the pipeline i.e., detector locates the person/pose region-of-interest (ROI) within the camera frame followed by the second step of the pipeline (tracker). Tracker predicts the pose landmarks and segmentation mask with the region-of-interest using the ROI cropped frame as the input.
To avoid having to run the detector in every frame, it is only invoked when the tracker can no longer identify body pose presence in the previous frame.

Part 3: (4 points)

Put your annotated picture after pose detection here

See slide 11

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

Part 4: (6 points)

How would you estimate depth information from a 2D image, given that the person's feet and the chair are on the same floor and are both visible in the image?

>> The only information provided here is that we have a single image with a person's feet and the chair both on the same floor and both visible. We need extra information such as length of feet and chair dimensions in order to perform regression analysis to estimate the depth information. With this knowledge we could perform similar triangle analysis to estimate the ratio of real-world size to the image pixel size for an object. If the ratio for chair is larger than the ratio for the person, I would suggest that the person is closer to the camera than the chair and vice versa.

check_hand_inside_bounding_box(): Code (10 points)

Please screenshot your “**check_hand_inside_bounding_box()**” here (only the steps you implemented, not the whole function)

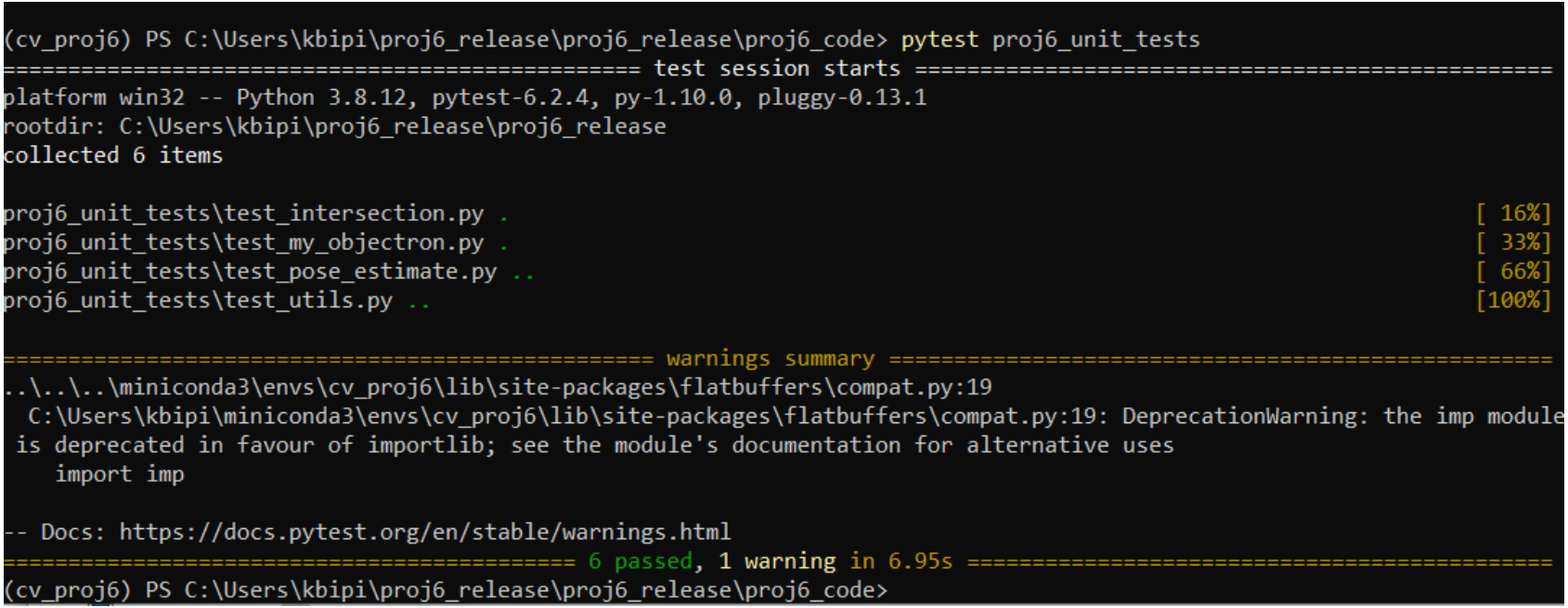
```
    inside = None
    ###########################################################################
    # TODO: YOUR CODE HERE
    ###########################################################################
    minimum_point = pts[0,:]
    maximum_point = pts[7,:]
    if hand[0]>=minimum_point[0] and hand[0]<=maximum_point[0] and hand[1]>=minimum_point[1] and hand[1]<=maximum_point[1] and
hand[2]>=minimum_point[2] and hand[2]<=maximum_point[2]:
        inside = 1
    else:
        inside = 0
    #raise NotImplementedError('`check_hand_inside_bounding_box` function in '
    #                          + 'intersection.py needs to be implemented')
    ###########################################################################
    #                           END OF YOUR CODE
    ###########################################################################
```

Part 5: (6 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 are given 3D coordinates of eight vertices of box in space. From this we can extract the minimum and maximum values for each coordinate of the bounding box. Now if the corresponding coordinates of the hand falls between the minimum and maximum values for each of the three coordinates of the bounding box, we can say that the hand is inside the box. If any of the coordinate falls outside this limit, it is said to be outside the box.

Screenshot the result of running “pytest” on this page:



Extra Credit (for all): Your own image

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

Extra Credit (for grad students): Interaction Video

<Screenshot the code you implemented in **process_video()** here>

Extra Credit (for grad students): 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 (for grad students): Interaction Video

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