MediaPipe Objectron (Deprecated)

Introduction

MediaPipe Objectron is a mobile **real-time 3D object detection** solution for everyday objects. It detects objects in 2D images and estimates their poses through a machine learning model, trained on the Objectron dataset.

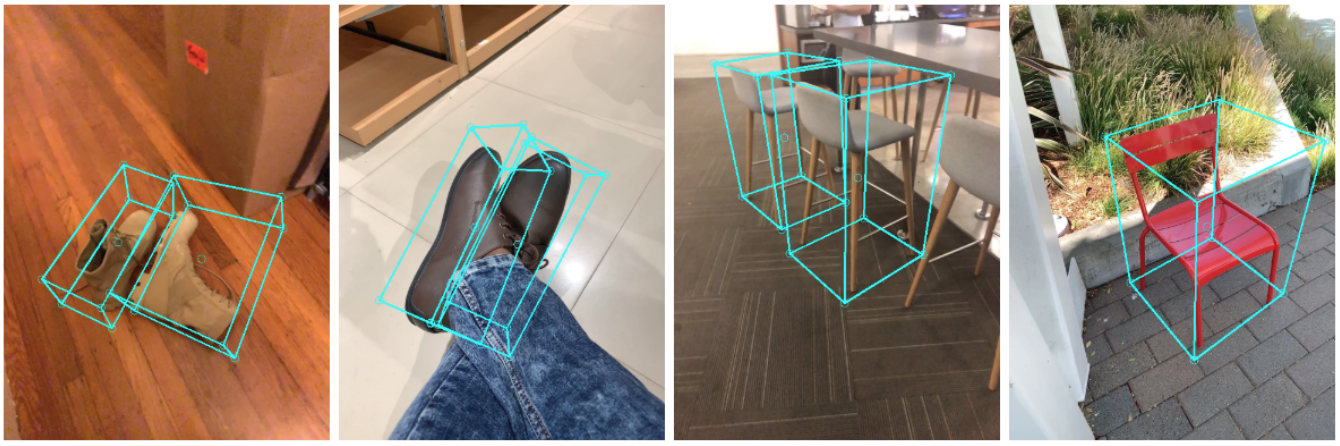
Object detection is an extensively studied computer vision problem, but most of the research has focused on 2D object prediction. While 2D prediction only provides 2D bounding boxes, by extending prediction to 3D, one can capture an object's size, position and orientation in the world, leading to a variety of applications in robotics, self-driving vehicles, image retrieval, and augmented reality. Although 2D object detection is relatively mature and has been widely used in the industry, 3D object detection from 2D imagery is a challenging problem, due to the lack of data and diversity of appearances and shapes of objects within a category.

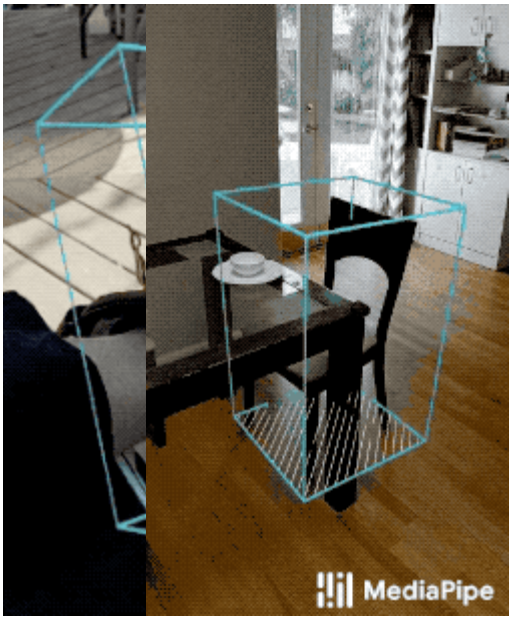
MediaPipe Objectron was considered since this project requires extracting estimated 3D coordinates of objects detected via visual input. Most object detection solutions such as YOLO only focus on segmenting objects from 2D images and videos (still treated as streaming image frames). This was discussed with our client who suggested it is possible to use existing 3D object detection solutions like MediaPipe Objectron to directly estimate the coordinates in the real world.

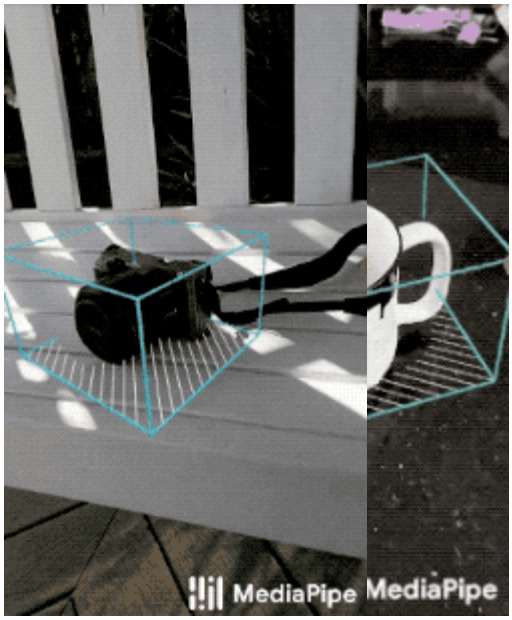
Detection and Tracking

When the model is applied to every frame captured by the mobile device, it can suffer from jitter due to the ambiguity of the 3D bounding box estimated in each frame. To mitigate this, we adopt the same detection + tracking strategy in our 2D object detection and tracking pipeline in MediaPipe Box Tracking. This mitigates the need to run the network on every frame, allowing the use of heavier and therefore more accurate models, while keeping the pipeline real-time on mobile devices. It also retains object identity across frames and ensures that the prediction is temporally consistent, reducing the jitter.

The **Objectron 3D object detection and tracking pipeline** is implemented as a MediaPipe graph, which internally uses a detection subgraph and a tracking subgraph. The detection subgraph performs ML inference only once every few frames to reduce computation load, and decodes the output tensor to a FrameAnnotation that contains nine keypoints: the 3D bounding box's center and its eight vertices. The tracking subgraph runs every frame, using the box tracker in MediaPipe Box Tracking to track the 2D box tightly enclosing the projection of the 3D bounding box, and lifts the tracked 2D keypoints to 3D with EPnP. When new detection becomes available from the detection subgraph, the tracking subgraph is also responsible for consolidation between the detection and tracking results, based on the area of overlap.







Python Solutions

MediaPipe offers ready-to-use yet customizable Python solutions as a prebuilt Python package. MediaPipe Python package is available on PyPI for Linux, macOS and Windows.

Easily install the MediaPipe Python package using pip:

```
pip install mediapipe
```

Import in Python and ready to use:

```
import mediapipe as mp
mp_face_mesh = mp.solutions.face_mesh
```

Supported configuration options:

- static_image_mode
- max_num_objects
- min_detection_confidence
- min_tracking_confidence
- model_name focal_length
- principal_point image_size

```
import cv2
import mediapipe as mp
mp_drawing = mp.solutions.drawing_utils
mp_objectron = mp.solutions.objectron

# For static images:
IMAGE_FILES = []
with mp_objectron.Objectron(static_image_mode=True,
                            max_num_objects=5,
                            min_detection_confidence=0.5,
                            model_name='Shoe') as objectron:
  for idx, file in enumerate(IMAGE_FILES):
    image = cv2.imread(file)
    # Convert the BGR image to RGB and process it with MediaPipe Objectron.
    results = objectron.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

    # Draw box landmarks.
    if not results.detected_objects:
      print(f'No box landmarks detected on {file}')
      continue
    print(f'Box landmarks of {file}:')
    annotated_image = image.copy()
    for detected_object in results.detected_objects:
      mp_drawing.draw_landmarks(
          annotated_image, detected_object.landmarks_2d, mp_objectron.BOX_CONNECTIONS)
      mp_drawing.draw_axis(annotated_image, detected_object.rotation,
                           detected_object.translation)
      cv2.imwrite('/tmp/annotated_image' + str(idx) + '.png', annotated_image)

# For webcam input:
cap = cv2.VideoCapture(0)
with mp_objectron.Objectron(static_image_mode=False,
                            max_num_objects=5,
                            min_detection_confidence=0.5,
                            min_tracking_confidence=0.99,
                            model_name='Shoe') as objectron:
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      print("Ignoring empty camera frame.")
      # If loading a video, use 'break' instead of 'continue'.
      continue

    # To improve performance, optionally mark the image as not writeable to
    # pass by reference.
    image.flags.writeable = False
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = objectron.process(image)

    # Draw the box landmarks on the image.
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    if results.detected_objects:
        for detected_object in results.detected_objects:
            mp_drawing.draw_landmarks(
              image, detected_object.landmarks_2d, mp_objectron.BOX_CONNECTIONS)
            mp_drawing.draw_axis(image, detected_object.rotation,
                                 detected_object.translation)
    # Flip the image horizontally for a selfie-view display.
    cv2.imshow('MediaPipe Objectron', cv2.flip(image, 1))
    if cv2.waitKey(5) & 0xFF == 27:
      break
cap.release()
```

Deprecation

Updated on 2023-9-18

For the past week, we looked into MediaPipe Objectron to see if we can utilise its ability to process 3D object detection. But we realised that the official support for Objectron has ended from 1st March 2023 and only a few documentations have been preserved with lots of details missing. In the end, we made the decision to deprecate the use of Objectron is our project due to reasons listed below:

- Official support ended
- Missing documentation
- Can only detect and process **one object** in the scene
    - Doesn't support multiple object estimation
- Can only detect **4 object classes**
    - Shoe, Chair, Camera, Cup
    - Anything other than these four categories can not be recognised and correctly processed

Updated on 2023-9-20

We have borrowed two types of depth camera from our client:

1. **Microsoft Azure Kinect**
2. **Intel RealSense Depth Camera**

As the alternative approach, it is decided to use depth camera for capturing real-time depth information regarding the target objects based on the customised TOF sensor. The extracted depth information can be combined with the 2D coordinates extracted from YOLO so that a more precise object position in the real world can be estimated to support real-time decision making and robot movement.

Objectron 3D Detection Demo