Presence Detection with OAK-D

Max-Felix Müller

2021
March

Contents

List of Figures

1 Motivation

Since our team was selected as a Phase 1 winner of the OpenCV AI Competition 2021 we received multiple OAK-D cameras. In this project, the goal is to use the supplied neural network models to make a simple, but useful project. This is mainly to get started with using the OAK-D.

2 Requirements

- Limit the frame rate to 1 FPS
- Detect persons in the room and count them
- Push the output to a database to visualize the results in Grafana

3 Introduction

The OAK-D is a 4k camera comprising three image sensors and an on board processor. Two monochrome cameras, one on either side, allow for depth detection while the center camera is full color and with higher resolution. The on board processor from Intel allows on the edge inferencing of neural networks, offloading the image processing from the host to the camera itself. This allows for low cost devices like a Raspberry to run the OAK-D at it's full potential. Also the bandwidth of the output data can be significantly decreased, compared to uploading the full image to the cloud for processing.

The API comes with a processing pipeline. For the introduction, mobilenet is used as a neural network. The input resolution of the network is 300 x 300.

The existing demo code can be used an easily modified to accomplish these tasks.

4 Limiting the Frame Rate

Limiting the frame rate can easily be achieved by setting the FPS when adding the camera to the pipeline.

```
cam_rgb = pipeline.createColorCamera()
cam_rgb.setPreviewSize(300, 300)
cam_rgb.setInterleaved(False)
cam_rgb.setFps(1)
```

5 Detecting People

The mobilenet is used to detect persons. Increasing the confidence threshold is necessary, since later on there is no further differentiation. After adding it to the pipeline, its output can be used.

The reasoning behind limiting the framerate is to reduce data output to the database. This could also be handled in different ways, but this was the simplest one. Also power consumption is reduced.

```
detectionNetwork = pipeline.createMobileNetDetectionNetwork()
detectionNetwork.setConfidenceThreshold(0.8)
detectionNetwork.setBlobPath(mobilenet_path)
detectionNetwork.setNumInferenceThreads(2)
detectionNetwork.input.setBlocking(False)
cam_rgb.preview.link(detectionNetwork.input)
```

To reduce overhead, the frame is not drawn. Instead only the label is checked to find whether it is a person or not. Counting the persons in the frame is then only a matter of counting the number of detected objects with a matching label.

```
for bbox in bboxes:
    if bbox.label == 15:
    person_count = person_count + 1
    crop_frame = frame_depth[y1:y2, x1:x2]
   #cv2.imshow("depth_crop", crop_frame)
    distances.append(cv2.mean(crop_frame)[1])
```

5.1 Calculating the Distance

To calculate the distance, the depth is used. The OAK-D has two monochrome cameras, one on either side. The disparity between the two images is used to calculate the depth in the pipeline on the device itself.

```
depth = pipeline.createStereoDepth()
depth.setConfidenceThreshold(200)
depth.setOutputRectified(True) # mirror image
depth.setRectifyEdgeFillColor(0) # black on the edges
cam_left.out.link(depth.left)
cam_right.out.link(depth.right
```

The output is then cropped to the bounding box of the person detected with the neural network. Averaging the green channel of the image allows an estimation of the distance of the person. The higher the number, the closer. This is only a relative measurement and nothing absolute. Also a bigger person would propably appear closer, since the bounding box is filled to a larger degree.

6 Storing in a Database

On the Raspberry Pi 4 of my Homeserver a MariaDB is running. To upload the data to it, the python module "mariadb" is used.

For a secure login, the credentials are put in a separate file. It is then included in the main script. The credentials file is not put on Github.

After logging in, a sql querry is sent to include a new data record. Included as data is the time, the number of persons detected and the "distance" to the closest person. To have a unique identifier for each, the server adds an automatically increasing id.

6.1 Visualizing in Grafana

The data can be visualized using Grafana to get a nice web interface.

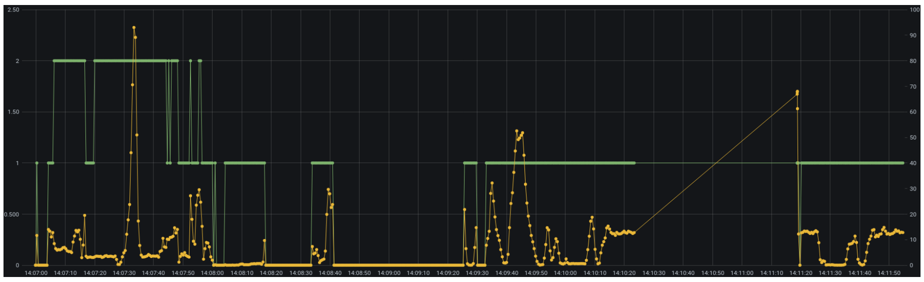

Figure 1: Visualizing in Grafana

The green line represents the number of persons detected. In yellow and on the right y-axis, the average distance of this person can be seen.

7 Possible Improvements

It is possible to add multiple neural networks detecting the same thing. For example all three cameras could be used to detect persons. If there is a large deviation in the detection or the number of people detected by each camera, there is propably a fault somewhere. This redundancy can be used to increase the confidence in the result. This was tested once, but it comes with a large hit on the framerate.

The network could be trained to distinguish between adults and children.

Instead of only finding the closest person, the distance to each person could be calculated individually.

The distance calculation can be improved. Instead of cropping the whole box, only the person itself could be used to calculate the average distance. Also the way distance is turned into a color does not only affect the green channel, but only it is used at the moment.

8 Links

OAK-D:
https://store.opencv.ai/products/oak-d

OpenCV AI Competition 2021:
https://opencv.org/opencv-ai-competition-2021/

This porject and more on Github:
$https : //github.com/MxFxM/TGMB/blob/main/code/04_persons_depth.py$