Maurader's Map Final Report

Eric Minor, Supriya Naidu, Ashwin Sankaralingam, Nelson Mitchell, Talia Krause

Project Desctiption

The goal is to use computer vision to track people in a room or rooms and place them onto a 2D grid, marking their location. This technology would be useful for anyone trying to find someone else in room/building, plus it would be fun to have a real life version of the Marauder's Map from Harry Potter.

A common problem in large buildings is trying to figure out the location of certain people who you need to talk to or deliver things too. Using cameras situated throughout a building, a map with the location of everyone in the building could be constructed, making locating individuals easier.

Specifically, this technology would be used in a collaborative workspace where people want to be available to help others. Some jobs require people to spend time in a variety of places. For instance, a researcher might spend time in a lab, in a machine shop creating equipment, or in a computer room analyzing data. Providing a real-time map of workers and their locations would make it easier for colleagues to find one other. This would not compromise privacy as a worker is supposed to be available while at work and not hidden. It would merely save a colleague a few minutes of searching around to find the person they needed to talk to. The technology could also be used to keep track of which professionals are currently available in a given helproom or other place where assistance is expected to be given. If a person marks themselves to be tracked in such a room, it is expected that they are there to render assistance and thus would not want to be hidden.

Collaboration focused workspaces rely upon being able to quickly and easily access the expertise of your coworkers. Messaging apps allow for quick digital communication, but it is often more efficient to simply have a conversation in person. The system we developed allows for consenting individuals to train a webcam system to identify and track their face in order to create a map of everyone's location in a collaborative workspace.

Using a simple interface, a python script records images of a person's face, extracts features, and uses transfer learning to create a recognizer. This can be done with multiple computers and cameras, which all connect via websockets to a Node.js server that aggregates the data and keeps a dictionary containing everyone's position. The node server also serves a website that displays a map of everyone's location.

Challenges Faced in the project

We had 2 main problems in the entire project:

- Face tracking and identification
- Depth estimation

Following are some of the techniques we tried, and will be explaining why we chose some.

Face tracking and identification :

In order to get a personalized effect for the user to use our system, we planned to identify each person seperately instead of tagging them as person x. This was quiet challenging to do in real time. After some digging we planned to use transfer learning over resnet layers to utilize the convolutional layer to produce the abstract representation of the face.

1) Tranfer learning using resnet: Renet [?] is one state of the art object recognition network. Using Resnet weights we tried train the last couple of fully connected layer with our trainable faces. Although it was accuracte, and capable of identifying if a person is present in the scene, it was not able to localize the presence, and it was computation intesive everytime we need to retrain a new person. It also had a very bad fps, compared to other models we tried.

2) Single Shot detector: OpenCV has released a new way of identifying objects and localizing those objects using transfer learning principles on facenet architecture. Facenet is one of the state-of-art facial recognition networks, capable of identifying face unaffected by pose, illumination and contrast.

* After loading the facenet architecture with pretrained weights, we can simply perform a single forward propogation to get a list of localized regions of interst with a probability score depicting how probable it is face. This stage is usefu

* Facenet model computes a 128 dimension embedding that quantifies as an abstract representation of the face. Each person's 128 dimension would differ from another person, and it would be possible to linearly seperate these embeddi

* Using SVM and linear classifier, we classify the detections based on the person's embeddings. As only the regions of interest is passed through the neural network, it is quiet fast in most of the CPU, and real time.

* For each user, we have a trainable module, where they can enter their name and capture upto 5 pics, that will then undergo the embedding process for classification.

//insert image after review

3) Centroid Tracking: Due to the use of SVM for classification, the classification was jittery as the faces that lie near to the margin was often confused. In order to increase the confidence in the classification, polling method was introduced to ID a particular face after a majority. In 40 consecutive frames, if a face has a majority name, then that face gets that name for the next 400 frames. Once again on the 400th frame, we flush the registered faces, as we wanted to the classification system to be active and correct itself if it got the face wrong the first time.

* Centroid Tracking uses a simple method of comparing euclidean distances between various centroid of object on the scene. It is a effective algorithm, to track multiple moving objects in the scene without loosing track.

* For every face in the scene, a centroid is identified and kept in memory.

* When a person moves in a subsequent frame, the euclidean distance will be smaller compared to other objects in the scene. If there was a new object in a subsequent frame, the system will assign a new ID. If an object is not present in

//insert image after review

Depth Estimation :

1) Kinect Depth: We planned to use Kinect 360 to get the depth details on the face we had already localized using SSD algorithm. The depth map generated by kinect was not very accurate with respect to real world measures, which led us to drop the kinect to use for depth. Moreover since the discontinue of Kinect by Microsoft, the support for kinect seems to have reduced amongst the community.

2) Face size and angle: Depth can be used with the angle from central of a face to calculate coordinates in 2D space. When combined with camera position coordinates and angle, this allows the absolute position of each detected person to be estimated. * Depth ∝ (Actual Face Width)/(Face Width in Frame) *

Visualization and Localization

A single camera often cannot see each face in an area, so multiple cameras are necessary to accurately detect every face in a room. The position of each camera is manually set so it can calculate the absolute location in the real world. After absolute coordinates have been calculated for each detected face, the data is aggregated on the node server and passed on to clients, which connect to the server via a website. The website uses websockets to constantly update positions.

//add picture after review

Technologies and Libraries

1) Python * imutils (for video interfacing) * OpenCV (for face detection and webcam) * Caffe (for loading models) * websocket-client (for sending messages to website) * tkinter (for user input) 2) Javascript * sockets (for sending details) * p5.js (for drawing on canvas)

Risks faced and mitigated

- Real time face detection: We were facing the problem of the system to be slow to run as well as train new faces. To avoid that we introduced SSD(Single Shot Detector) utilizing facenet weights, that speeded up the detection to real-time.
- Personalized identification: Utilizing the facenet embeddings, and SVM classification, we were able to classify multiple of faces.
- Jittery face detections: Using SVM, we faced a problem of faces being confused when it was closer to the margin. To reduce the sensitivity we added a polling mechanism with Centroid tracking to avoid jitter in detection.
- Multi-Camera Data: With multiple cameras sending data to the server, we had to add a unique configuration file to every camera to avoid incorrect placement on the map.

Project Outcomes

We feel we learnt to develop an user end-to-end trainable computer vision system capable of training on each user and capture their world coordinates efficiently.

- Utilizing transfer learning to reuse weight from a trained classifier.
- Using SVM to classify an embedding input into multiple new classes to provide user trainable features.
- On creating a system capable of loading, training, inferencing computer vision models in real time.
- Setting up heroku and git submodules for communicating with multiple projects
- Using websockets for communication.

References

[1]Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[2] J. C. Nascimento, A. J. Abrantes and J. S. Marques, "An algorithm for centroid-based tracking of moving objects," 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258), Phoenix, AZ, USA, 1999, pp. 3305-3308 vol.6.

[3] Tutorials from https://www.pyimagesearch.com/