UROP Report

AI meets Big Data:

Location Sensing by Surveillance Cameras

by

SONG, Sizhe

Supervised by

Prof. Shueng-Han Gary Chan

August 9th, 2019

Abstract

Location data are highly useful in daily life, they can be applied to crowd control, security and many other areas. However, the conventional tracking methods such as GPS and Wi-Fi tracking do not have a satisfying performance indoors, GPS suffers from poor signals while Wi-Fi tracking is unstable and can be disappointingly inaccurate sometimes. This project is aimed to replace them with surveillance cameras, and in this report, we will introduce how we designed and implemented the system to track people and gather relevant data.

1 Introduction

1.1 Objectives

The system should be able to:

- ❑ Get videos' frames
- ❑ Detect people in video frames
- ❑ Distinguish trackings of different people
- ❑ Output the trackings with real-world coordinates and time

1.2 Camera installation

The camera used in this project is installed on the first floor of CYT building on HKUST campus, capturing most areas of the staircases right outside the building. The whole staircase has 5 different sections, three of them are stairs and the other two are flat.

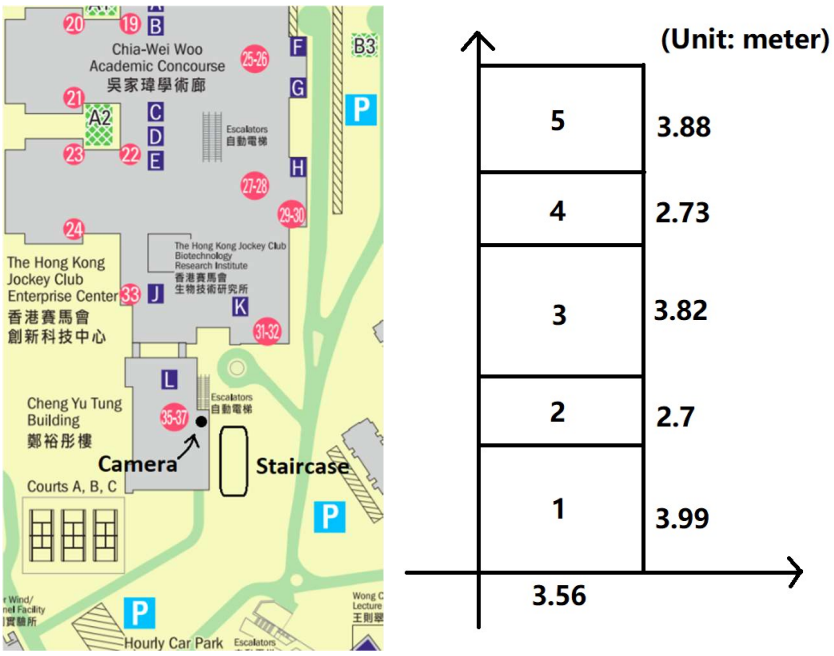

Figure1: Camera Installation Location & Staircase Info

2 Methodology

2.1 Video processing

The surveillance video captured by our camera have 25 frames per second. Although the more frames we process, the more accurate the result will be, it can also significantly increase the time cost. Moreover, it is believed that the range and the track of a person's walking within one second are simple and limited in most cases. Due to these two facts, we decide to only process 5 frames per second.

2.1.1 Detection of people

In this project, we use *Yolo* for people detection. *Yolo* is an open-source real-time object detection system, which has a relatively better performance to other similar tools. It also allows users to choose from more than 10 different models to achieve an expected balance between accuracy and speed [1]. For an input frame, *Yolo* will detect the people inside and return their locations in terms of pixel coordinates. In the

meantime, each person in the frame will be cut and stored to a new image file for later use.

Figure2: Yolo People Detection

2.1.2 Output JSON File

For each coordinate obtained in 2.1.1, we output it to a JSON file together with a timestamp and a unique index. In this way, we have represented each detected person in each processed frame by a tuple in the output JSON file and an image for further processing.

[{"h": 13, "m": 28, "s": 8, "ms": 0.639, "index": 3, "x": 0.259, "y": 2.958}]

An example of JSON output format: Time + Index + Coordinate

2.2 Coordinate transformation

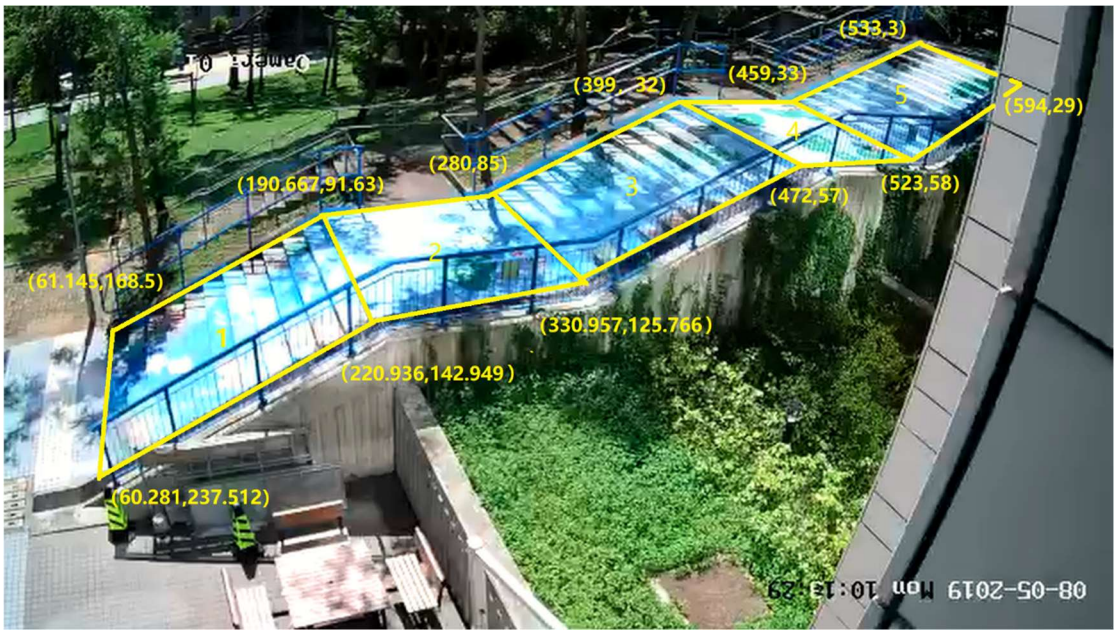

Figure3: Anchor points

So far, the coordinates are still pixel coordinates, we must transform them into real-world coordinates. First, we need to find out the coordinates of some special points, for example, the vertices of each staircase section. We call these points that we know both their pixel coordinates and real coordinate the anchor points. We will transform the pixel coordinates by their relative position to the four anchor points of the sections they belong to.

According to Perspective Principle, two parallel lines will intersect at the end point. In figure4, P is the coordinate we want to transform, $A \sim D$ is four vertices of a staircase section. The line between P and *end point* intersects with AB at E, and with DA at F. The rectangle on the right side is the coordinates in real-world. Now we are going to estimate $\frac{A'E'}{E'B'}$ using $\frac{AE}{EB}$. By the same method, we can estimate $\frac{D'F'}{F'A'}$ as well.

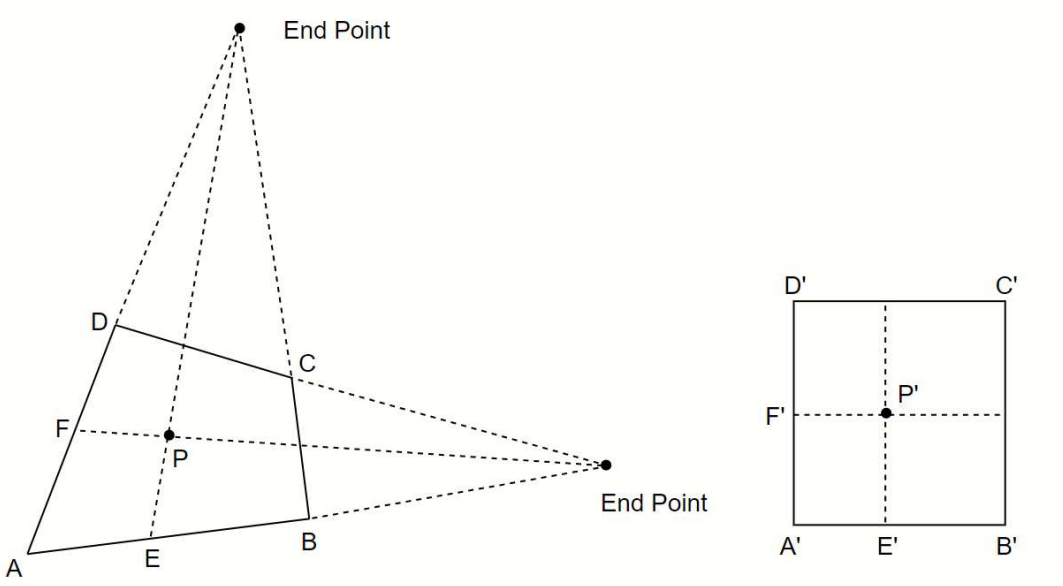

Figure4: Estimation method

In this way, we have obtained the estimation of the transformed coordinates. The error of this estimation can be represented by the α angle in figure5. Assume a line *A'B'* in the real world is *AB* in the video (the projection of *A'B'*) and *A'F* is parallel with *AB*. If the projection surface is parallel to *AB*, then the error is 0. If they are not parallel, which is common in practice, the smaller the α angle is, the more "parallel" *P'E* is with *B'F*, the smaller the error between $\frac{A'P'}{P'B'}$ and $\frac{A'E}{EF} = \frac{AP}{PB}$ is.

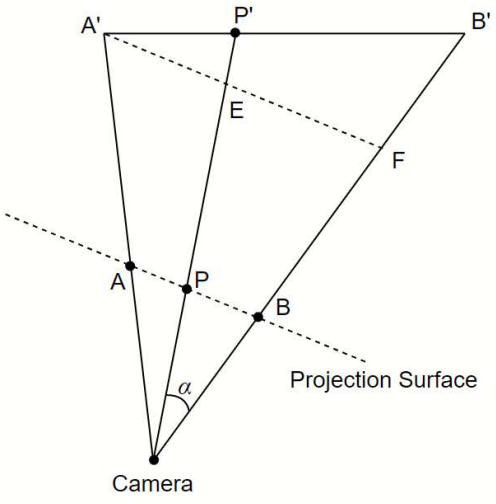

Figure5: Error Analysis

2.3 Identification

The next step is to determine which tuples are from the same person and assign a unique ID to them by comparing their coordinates and the cropped image we obtained in 2.1.1. Here is the pseudocode helping to elaborate the identification logic.

```
1     for each tuple:
2           for each person in active_list:
3                 compare the tuple with the person
4
5           if the tuple matches someone:
6                 assign the person's id to the tuple
7           else:
8                 create a new person object with a new id
9                 assign new id to the tuple
10                add new person to active_list
11
12          for each person in active_list:
13                if the latest tuple is beyond active_period:
14                      remove the person from active_list
```

Two variables are introduced here: active_list, active_period. Since *Yolo* sometimes cannot recognize a person in a specific frame, it is possible that a person may "disappear" for one or two seconds in our records. After they appear again, we believe it is better to temporarily consider them as someone new, instead of adding them to the old identified people records. In this way, we can process the records more quickly with an easier logic and leave the concatenating to 2.4. The active_period is a customized variable to define after how long we will remove a person from active_list. Once a person is inactive, we think that his track ends so that when we are identifying a tuple, we only need to compare it with those people in the active_list.

For "*compare the tuple with the person*" in line 3, we first check their coordinates. If the tuple's coordinate and the person's coordinate are too far away from each other, making it impossible for humans to move such a distance in milliseconds,

it is certain that they do not match. Then we will compare the tuple's image with the person's images. If the match rate reaches a specific threshold, which is set by the users, we consider they match. In case that there might be more than one matches, we always pick the one with the highest match rate.

2.4 Concatenating

Finally, we still need to do concatenating. Sometimes a walking person may look different even from the last second. It is possibly because of light and shadow in the video, or because the person's angle to the camera is turning all the time. All these factors make the images of the same person do not look that similar uunder our similarity algorithm. In this case, it is risky to make the judgment right away, because adding a dissimilar image to a person's images will affect all the comparison later. As is mentioned in 2.3, we will temporarily consider them as different people, and now it is time to concatenate those dashed lines. We will look into the trackings that start or end at the middle sections of the staircase, if two dashed trackings appears in a small time interval and their end and start points are close, they are very likely to be the trackings of the same person.

3 Evaluation

In this part, we mainly focus on the accuracy of the coordinates transformation. In figure6, the rectangle is the border of the staircase area, dots and balck lines are the transformed coordinates and tracks, the yellow lines are the actual tracks.

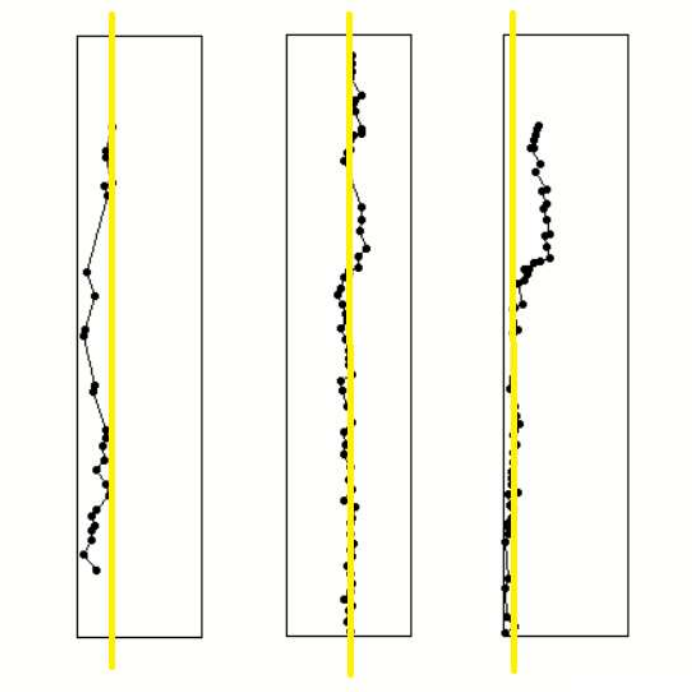

Figure6: Evaluation

4 Discussion

4.1 Error in coordinate transformation

As is explained in 2.2, the results of transforming coordinates are just estimations of the correct results. Plus, the error of this method depends on the α angle which seems not to have any practical meanings. To reduce the α angle, we may want to increase the distance between the camera and the area we observe, which is not always feasible especially where there are only limited choices of positions to install the camera, let alone this will bring a negative influence on the detection of people.

However, we notice that when a person approaches the edges, α angle also decreases. That means this method works better for areas closer to the edges. An idea is that in future we may cut each section to even smaller subsections to improve the performance. This will require more anchor points we mentioned before.

4.2 Expensive image similarity algorithm

The similarity algorithm we use is highly expensive, it contributes the most time cost is 2.3. Although we have been optimizing the code to run this algorithm as less as possible, we will have to find a better one if we want to significantly improve the performance of the whole system.

5 Conclusion

In this project, we have managed to build a system that is able to detect and track people in surveillance videos. It also has potentials to be combined with the Wi-Fi tracking to form a hybrid indoor location system which has both the accuracy of video tracking and the cost advantage of Wi-Fi tracking.

However, there are still many possible improvements to be done. Firstly, we need to either find an explicit way to transform coordinates or collect more anchor points to reduce the error. Secondly, a better image similarity algorithm is necessary, otherwise, the time cost is unacceptable.

6 References

[1] “Yolo: Real-Time Object Detection” [Online]. Available: https://pjreddie.com/darknet/yolo/ [Accessed 05 August 2019].