University of Nottingham
UK | CHINA | MALAYSIA

COMP3065.Computer Vision Coursework Report

May 3rd 2024

Yue Zhang

20215287

Contents

Abbreviations **1**

Chapter 1 Introduction **1**

Chapter 2 Methodology **3**

2.1 Feature Descriptor 3

2.2 Optical Flow 4

2.3 Other Techniques 5

Chapter 3 Result Analysis **9**

3.1 Video1 9

3.2 Video2 10

3.3 Video3 11

3.4 Video4 12

Chapter 4 Performance evaluation **14**

Chapter 1

Introduction

Among the three topics, I choose to work on the second project 'Person tracking from videos'. My method can create boxes on moving people in the original video and update their motion trajectories. Additionally, it can also display the number of people detected in the upper left corner of the output video. Figures 1.1 and 1.2 show examples of the results of detecting single and multiple people from a video.

Figure 1.1: An example of detecting one person from the video

Figure 1.2: An example of detecting multiple person from the video

Chapter 2

Methodology

2.1 Feature Descriptor

Feature point detection is extracting the important feature points in the image and provide location for object tracking and position prediction. In a word, the movement trajectory of the people in the video can be captured by observing the feature points it drew.

First of all, the first frame of image will be converted to a grey scale image for feature point detection. Then the 'cv2.goodFeaturesToTrack' has been utilized to perform the feature point detection. Next , detected feature points will be added to the track list.It is a Shi-Tomasi Corner Detector integrated in the OpenCV. The core idea of the Shi-Tomasi corner detector is calculating the eigenvalues of the local region around the pixel (usually the smallest eigenvalue), and then decides whether it is a corner point based on these eigenvalues. The following is the detailed steps.

1. Compute the gradient values I_x and I_y of the image in the x and y directions, typically computed using Sobel filters.

2. Define a second-order derivative matrix M within the neighborhood of pixel (x, y), where M elements are:

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

3. Compute the eigenvalues λ_1 and λ_2 of M.

4. Calculate the corner response function R, typically using the minimum eigenvalue $\lambda_{\min} = \min(\lambda_1, \lambda_2)$:

$$R = \lambda_{\min}$$

The parameters of it has been set to 100, 0.1 and 20 which means that the maximum number of corner points to be detected is 100, the quality level threshold of the corner points is 0.1, and the minimum Euclidean distance between detected feature points is 20. Additionally, in order to track the motion trajectory of humans entering new video frames, feature points are re-extracted every 15 frames.

2.2 Optical Flow

Optical flow is a method that uses the variation of pixels in an image sequence and the correlation between neighbouring frames to find the correspondence that exists between the previous frame and the current frame,

so as to compute the information about the motion of an object between neighbouring frames. Because it can calculate the movement of pixels between adjacent frames to infer the trajectory of the object so that it is used to track the trajectory of people in the video for this project.

Lucas-Kanade has been applied here to perform optical flow. Lucas-Kanade is a classic optical flow estimation algorithm used to compute local motion between consecutive frames of an image sequence. The core idea of it is to estimate motion parameters by minimising an error function, where the error function is the sum of squares of the residuals. The local motion is the vector (u, v), where u and v denote the displacement of the pixel in the horizontal and vertical directions, respectively. The Lucas-Kanade algorithm estimates the local motion vector (u, v) by solving a system of equations with zero partial derivatives of the function 2.1. This system of equations can be implemented by solving the equation 2.2. It is perform by 'cv2.calcOpticalFlowPyrLK()' which is intergrated in the openCV library.

$$E(u, v) = \sum_{x,y} (I_x(x, y)u + I_y(x, y)v + I_t(x, y))^2 \tag{2.1}$$

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \tag{2.2}$$

2.3 Other Techniques

2.3.1 Delete Useless Feature Point

Unlike machine learning, the feature descriptor used here is a traditional technique that can not only detect people in a video, which leads to some

other objects, such as walls or plants, also being assigned feature points. Figure 2.1 illustrates an example of this weakness.

To solve this problem, I developed a measure that removes feature points that move less than a certain distance within a specified number of frames. The number of frames and distance have been set to 30 and 10 in this project. This step successfully deletes useless feature points and significantly increase the accuracy of the detection. Figure 2.2 shows the result after deleting invalid feature point.

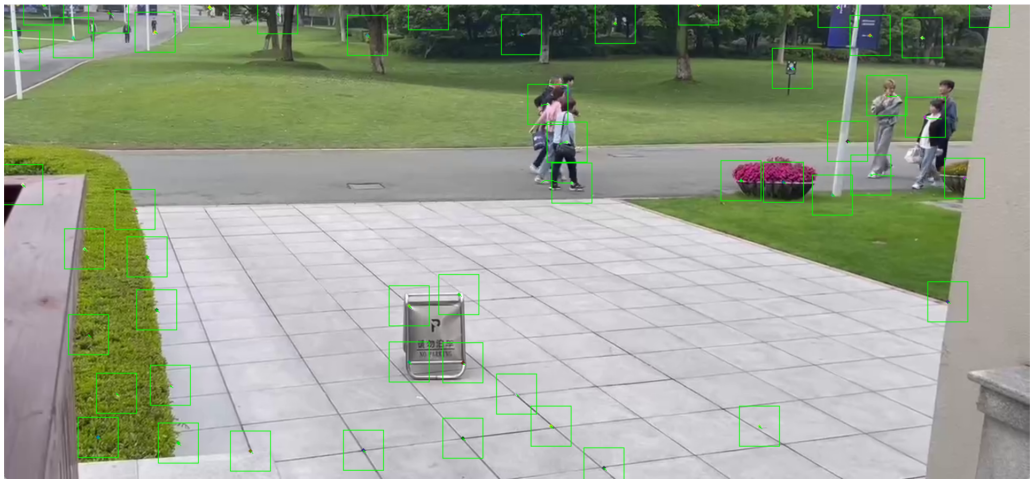

Figure 2.1: An illustration of the result without deleting deleting useless feature points

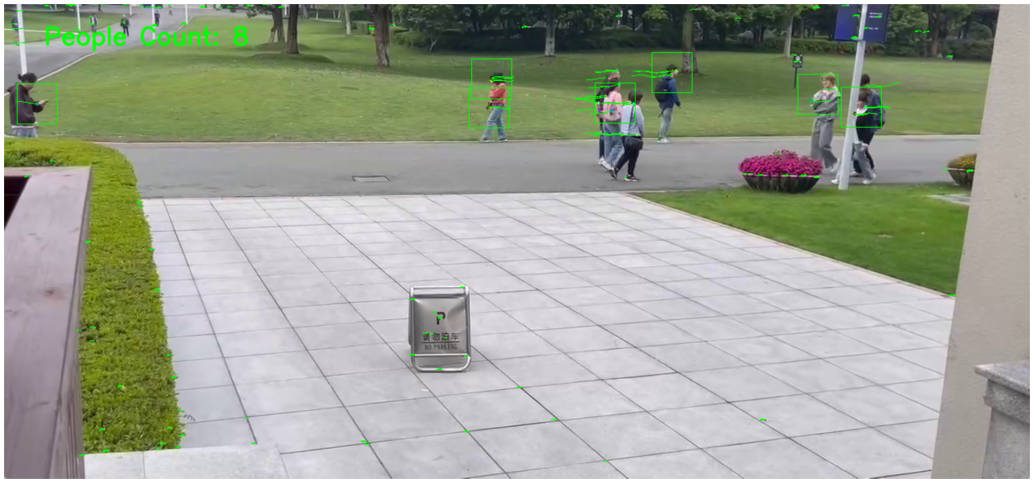

Figure 2.2: An illustration of the result after deleting invalid feature points

2.3.2 Box merging

Although most of the invalid feature points have been removed, there is still the problem that there will always be more than one feature point on a person. Because this method is tracking and framing people based on feature points, this issue results in not being able to accurately draw a box on a person.

In order to solve this problem, a function has been proposed to calculate the Intersection over Union (IoU) between two bounding boxes. This is a commonly used method to calculate the overlapping area of two boxes. The function first calculates the area of each of the two boxes, then calculates the area of the intersection. Finally, it will calculate and return Intersection over Union (IoU) of two bounding boxes. Subsequently, a threshold is set, and if the overlap ratio of the two boxes exceeds this threshold, they will be merged and return a mean point. The number of people in the video is obtained by counting the number of boxes in each frame of the output video. Figure 2.3 and 2.4 show the results before and after performing the merging of feature points, respectively. It can be concluded that this method can greatly reduce the occurrence of multiple boxes on one person and make the detection of the number of characters more accurate.

Figure 2.3: An illustration of the result without merging points

Figure 2.4: An illustration of the result of video1

Chapter 3

Result Analysis

There are four videos in total to test the performance. All of them are recorded by myself except video4. In order to test on a video with a pure background environment, I found video 4 on the internet for testing. If you want to replace the video used for testing just make sure the video and the code file locating in the same folder and then you should change the path in the figure to the name of the video you want to use.

```
path = 'video3.mp4'
## path of output video
out = 'result.mp4'
```

Figure 3.1

3.1 Video1

video1 is captured outdoors with a cluttered background. Figure 3.1 shows the result when processing video1. The people and their trajectories in the video can be successfully tracked. However, there is a problem that sometimes multiple people are stacked on top of each other. Consequently, in many scenarios, only one person is effectively tracked when attempting

to track a group of people. This issue also results in inaccurate counting of the number of people in the video. The box merging mentioned above leads to this problem, with multiple feature points on a person and a group of people walking together all result in many boxes stacked on top of each other. So there is a need to find a suitable threshold to merge only the former cases. But this seems too difficult to achieve.

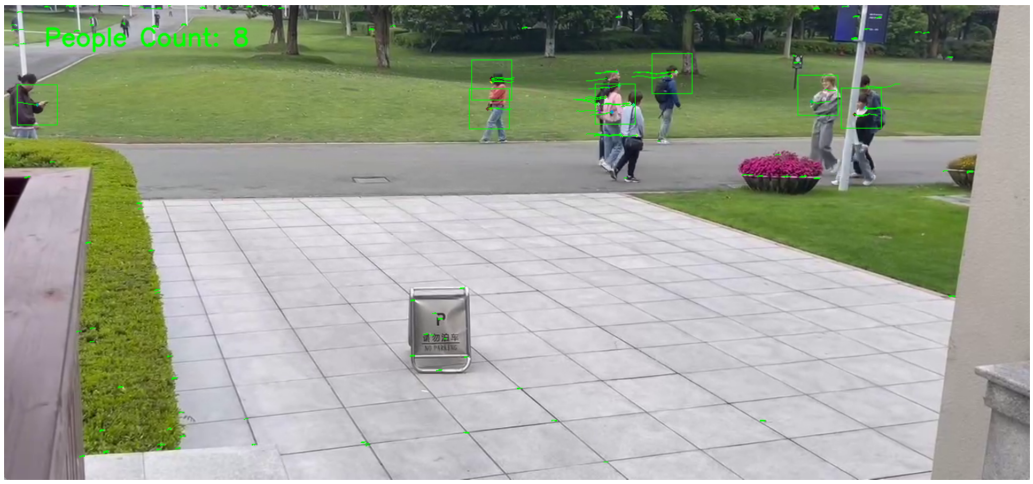

Figure 3.2: An illustration of the result of video1

3.2 Video2

There is only one people moving in the video2. This method succeeds in accurately tracking the people and outputting the number of people in the video. Figure 3.2 shows the result when processing video2.

Figure 3.3: An illustration of the result of video2

3.3 Video3

Video3 reveals a significant weakness of this method. The motion trajectories and amount detection of people in the video have significant errors. In this video, since my camera is moving, some feature points in the background are also perceived as moving under the optical flow tracking. This results in those invalid feature points not being removed when trying to delete useless feature points. I searched for some possible solutions, but ultimately could not address this issue. Figure 3.3 illustrates this issue. However, when the camera stops moving and the content in the video becomes stable, the trajectories and the amount of people in the video can be successfully detected. Figure 3.4 illustrates the result when the camera stops moving.

Figure 3.4: An illustration of the result of video2

Figure 3.5: An illustration of the result of video2

3.4 Video4

The video was shot against a single background with no complex surroundings. Figure 3.5 shows the result when processing it. This method successfully tracked most of the people in the video. However, it also revealed a

problem: if individuals in the video do not move, they cannot be accurately tracked and counted.

Figure 3.6: An illustration of the result of video4

Chapter 4

Performance evaluation

Overall, after testing with the four videos mentioned above, this method can successfully track the trajectories of people in most cases and display the number of people from the video. However, there are still significant errors in some situations, such as when people are not moving or when the camera undergoes significant movement. I think this is because this method detects movement based on variation of pixels not the entire person. This problem could be addressed by using machine learning methods, where only objects classified as humans would be monitored.

'cv2.CascadeClassifier(cv2.data.haarcascades' + 'haarcascade_fullbody.xml') and body_cascade.detectMultiScale() are used to test the performance of machine learning method, it is a pre-trained human detection model. Figure 4.1 shows that this machine learning method is still able to only track humans and ignore other background factors despite the camera movement. However, it also suffers from a significant drawback: despite the people are moving in the video, due to the considerable diversity in human appearances, this method may fail to identify and track them accurately in certain

cases. Figure 4.2 is an example illustrating this method is highly inaccurate in identifying human bodies. Moreover, machine learning typically involves greater time and money costs. Therefore, one advancement of this traditional computer vision method is it involves small time and money costs.

In conclusion, although sometimes this method can't completely track and count the number of people in the current video without error and it has a very significant error when the camera is moving. It can still give a relatively accurate prediction in most cases.

Figure 4.1: The result of video when using machine learning method

Figure 4.2: The result of video when using machine learning method