The Chinese University of Hong Kong

Intelligent Car Park System

Chow Wai Kwong, Lam Lee Shan

CSCI 4998, Final Year Project

Supervisor: Prof. Shengyu Zhang

4th December, 2016

*Abstract*

This project is trying to using existing equipment to improve car park management system. Appling the computer vision technology and algorithm design, we hope that the new management system can automate management processes by computer.

*Background*

CSE has several FYP car park management systems developed in previous years. They concern about the in-out records and checking empty slot. To tell driver where is the empty slot rely on human to assign the parking slot. In this project, an intelligent car park management system is developed to replace the human assignment. In other words, the system should recognize the size of incoming car and determine the best fit location for the car based on the on site situation.

*Purpose*

In the 1st semester, we would like to figure out what methods can we only using static camera to approximately distinguish the size of car so that we can automatically assign the position to the car in the following semester.

In fact, this project consists a lot of technical constraints to achieve the goal. For example, the unstable brightness of car park entry, object detection and tracking, height and distance of camera setting, movement of cars (Determine car is moving or not) etc. Hence, we have focusing on a lot of computer vision experiments to investigate and figure out the suitable solution in the 1st semester.

*Object Detection and tracking*

How to detect an object in the camera is one of the problems we need to solve. In fact, we may not need to consider the whole screen *(Figure 2a)*, we can only focus on the screen area that we need, then it can filter out some useless information or vision noise. In the following, we would like to call that area as “Interested Area” *(Figure 2b)*.

The general idea is using MOG2 algorithm provided openCV to compare the background image *(Figure 2f)* and the current image *(Figure 2a)*. And output the foreground image *(Figure 2c)*.

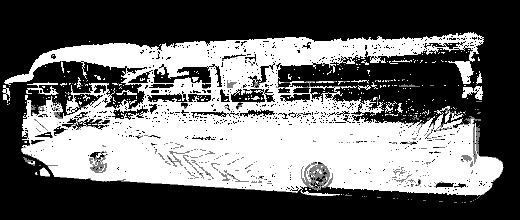


*Figure 1 (General Idea of MOG2 Algorithm)*

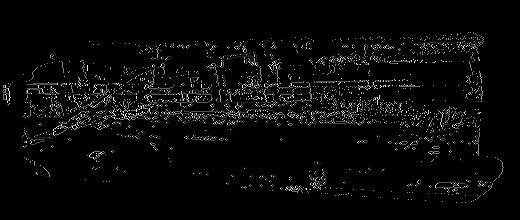


*Figure 2b (Interested Area)*

*Figure 2a (Original/Current Image)*



*Figure 2c (MOG2 algorithm Mask Image)*



*Figure 2e (Canny edge Processed image from (2c) )*



*Figure 2f (Background Image)*

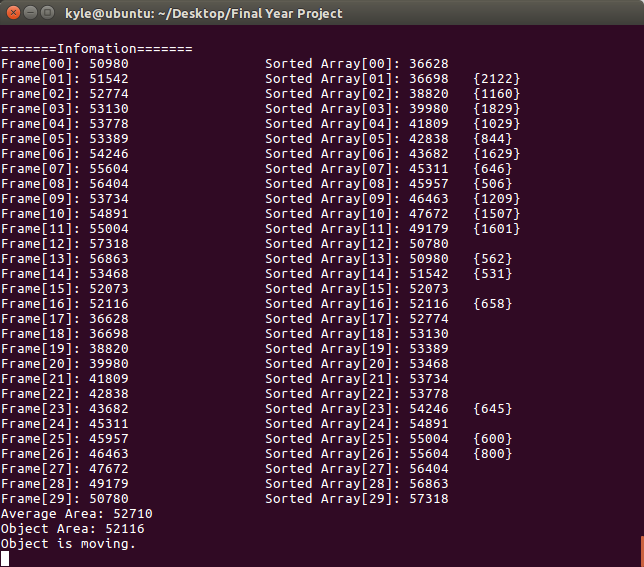
*Figure 2d (subtracted foreground Image by MOG2)*

MOG2 algorithm can only threshold the object image. The following step is finding contours of all object detected in the image. Canny edge detection algorithm *(Figure 2e)* is going to show the frontier edge and it helps calculating object contours in the screen. After finished all kind of processing, we can locate where the object detected in the screen. The following will be the result of processing:



*Figure 2g (object detected image)*

In *Figure 2b*, there is only 1 rectangle to indicate one object because of we based on object size in the screen to filter out unwanted information. The reason why there are inaccurate object detection *(Figure 2g)* is that the algorithm of Canny edge detection is not 100% correct to draw a continuous edge of the object contours. Some line may be represented in some dispersed dots. And it may let computer misunderstand this is another object even thought this is a part of the object. To compute the Edges or dots is belonging to the same whole object which is a subgraph isomorphism problem (NP-Complete). However, it is not necessary in our project, we only need to consider the whole object detected is enough. Hence, we can only focus on the object that it’s area is the biggest in the screen and in acceptable range. For example, object area detected is less than 1000 we simply can ignore it. The school bus’s object area calculated by openCV in *Figure 2f* is 52116 pixels.



*Figure 2h (Biggest object in (2b))*

Since there are no efficient algorithms to automatically subtract object from background perfectly. For MOG2 algorithm, it also subtracts the shadow of the object and determine it belongs to that object *(Figure 2d)*. Hence, the definition of of object area would be:

*Areatotal  = AreaObject + AreaObject Shadow*

However, the object shadow can be ignored since every car detection we would also add the area of object shadow for consideration. In that situation, the object shadow can be treated as constant variable. And it would not have a large effect to determine the size of car approximately.

Besides, the area detected in different timeslot will also different because of the brightness and distance to object. Hence, the area detected in different time slot will be different, but it is in the acceptable range. For example, there following are 3 different timeslot frame:

|  |  |
| --- | --- |
| ../../../tmp/VMwareDnD/5840b9 | ../../../tmp/VMwareDnD/1735239 |
| ../../../tmp/VMwareDnD/5acbbe | ../../../tmp/VMwareDnD/5849b9f |
| ../../../tmp/VMwareDnD/cde3407f/Screenshot%20from%202016-11-22% | ../../../tmp/VMwareDnD/d7646b8f/Screenshot%20from%202016-11-22% |

Although the area in different will not be the same even they are the same object, but their area is floating in a stable range. The object area in above figure are 53468, 52073, 52116 respectively, the range is around 52000~53500.

*Object Motion*

1. *Object movement determination:*

For determining what status of object is a minestrone of this project. Our algorithm is trying to compare the object area with past n frame. General idea is finding out the average object area in past n frame.

And then compare the current frame detected object area to the average area.

If the differences between them is greater than the acceptant range:

*| Areacurrent - AreaAverage | > Acceptant Range*

we can determine the object in that frame seems like moving.

Otherwise, If

*| Areacurrent - AreaAverage | < Acceptant Range*

and continuously within the range in 5 frame. We can determine the object in that frame seems like no movement.

1. *Object area exception:*

In sometimes, the object area detected may be abnormal because of light change, camera affected by wind / others, unexpected object detected in the screen, rainy day etc. The implementation to handle abnormal object area detected is that try to eliminate the object area which is not in the acceptant range suddenly. Hence, we would ignore those frame for calculation which is abnormal. And the sum of object area in all normal frames would be:

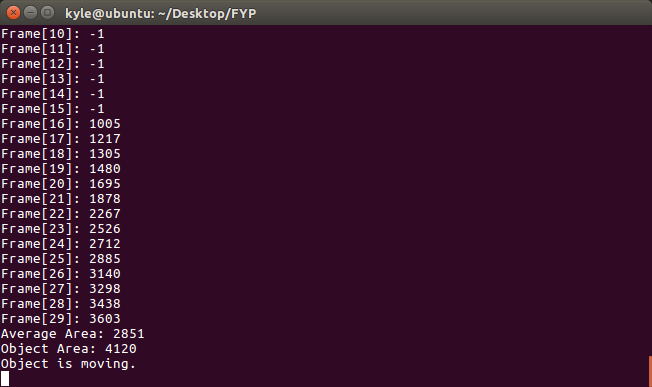
*Constraints*

Computer vision is one of the main area involved in this project. After doing researches and experiment, we discovered some factors would affect the camera reading. And it is really difficult to calculate the actual size of cars only using one single camera without any extra equipment, but the approximate size of object is enough for this project.

1. *Distance:*

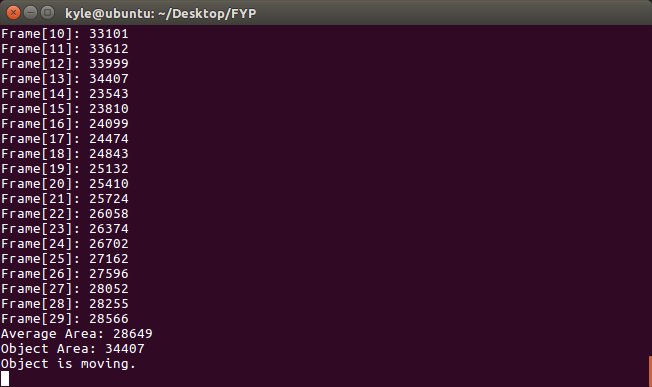
As mentioned in the above, object area means that the object occupied how many pixels in the screen. The area is directly proportional to the distance.

|  |  |  |
| --- | --- | --- |
| C:\Users\Kyle\Desktop\Screenshot from 2016-11-24 23-41-37.png  *Figure 3a (Origin image)* |  | C:\Users\Kyle\Desktop\Screenshot from 2016-11-24 23-41-45.png  *Figure 3b (Detected image by MOG2 )* |



*Figure 3c (Counted Object Area)*

|  |  |  |
| --- | --- | --- |
| C:\Users\Kyle\Desktop\Screenshot from 2016-11-24 23-52-36.png  *Figure 3d (Origin image)* |  | C:\Users\Kyle\Desktop\Screenshot from 2016-11-24 23-52-30.png*Figure 3e (Detected image by MOG2 )* |



*Figure 3f (Counted Object Area)*

In *Figure 3a* and *3d*, the size of the 4 seats private car is more or the less same as the taxi. The distance in *3a* between the camera and the car is around 30m, but in *3d* it is only 4m. Due to the different of distance, their area calculated are 4120 and 34407 pixels respectively. It can be concluded that the distance between the camera and object is directly proportional. And if we want to successfully approximately distinguish the size of cars in this project, we have to find a distance which is suitable for camera recording.

1. *Camera angle:*

*Technology*

Operating System:

* Linux Ubuntu 16.04 64bit

Programming language:

* C++

Tools/library:

* OpenCV

*Conclusion*

This is the first time of us touching technical skill in computer vision area. Both of us have no knowledge in that area, and we don’t know what method can we achieve our goal. What we can do is that only do a lot of computer vision experiment in this semester and hoping that we can have an efficient method for us to determine the size of car.

In fact, there are so many difficulties for our experiment. For example, finding a place and getting a permission to do our experiment. The video recording is needed to get permission by the security team of CHUK because of privacy issue. And also it is impossible for to set a camera 24 hours to get image since there are no place and we do not have equipment to do so. As mentioned in the above of this report, most likely the difficulties and constraint basically come from physical factor. The ideal situation is that the camera will not be shaken by wind, the brightness of the whole area is stable. In other words, it will be easy to implement in some car park which is like industry car park.

However, what we can do is trying to fix some physical parameter and minimize factors affect the computer reading. For example, the height and distance of camera setting, camera setting angle, camera vision angle etc. If the car park entry in outdoor, the effect came from brightness and wind we cannot be controlled. Then the next step maybe adds other method to assist us to do so. For example, after we detected there an object, using cascaded classifier to recognize which type of car it is. But it involves some machine learning knowledge such that we need to take some time to understand it.

*Reference*

*http://docs.opencv.org/3.1.0/d1/dc5/tutorial\_background\_subtraction.html*