1. Introduction

The motion detection program is used for determine the movement relation between one frame and the next frame, respects of each block (sub block of a frame). It has a simple structure. We can either use a piece of video or consecutive pictures as the input. The output is the motion vector of each block in a single frame. The algorithms used for motion detection are as follows: Brutal force, Hexagonal Search (with low pass filter version and Simulate annealing version).

1. Input
   1. Video

We use the matlab library function aviread to input the video in the memory. Thus, only limited types of the video format are supported. Nevertheless, there is a software called Winavi 9.0 can convert almost every video into the readable format for matlab. The video coding format is ZJMedia Uncompress RGB24. Still, one thing should be notified, the resolution of the video cannot be too high (such as 720p or more). The primitive format is born to deal with low resolution videos. Otherwise, error may occur.

The software is also included in this package.

* 1. Consecutive picture

If we don’t have the camera equipment, we can use the live camera to capture the moving cloud. The following link offers a good quality and view of the cloud. <http://65.121.113.114:8080/view/index.shtml>. Some networks may not support the video stream. However, it can be guaranteed that the video stream works fine when UCLA VPN is connected.

The next thing is to screenshot automatically when the live camera is all set up. We can use two pieces of software; one is zoneminder, which is really hard to set up; the other is automatically take screenshot software. The latter one is a green software, easy to install, however, it cannot configure the area you want to screenshot. After screenshot, one has to cut the picture to the certain size manually which takes time. But as a prototype, I choose this software, for the reason that the zoneminder is too hard to configure.

1. Algorithms

We divided the frames into small square blocks. For example. Here, we use the picture size of 360\*600. Then, we can use the 20\*20 pixels for each small block. In addition, there are in total 18\*30 blocks. Our goal is to determine the movement of these blocks according to the next frame.

* 1. Brutal force

In this situation, we search the neighborhood of each block then determine the movement vector of the block. The search range is 4 times greater than the area of the block. For instance, if the block size is 20\*20 with center coordinator (0,0). We then search the range of x, y both from (-20, 20).

Intuitively, the results seem to be the ground truth in the experiment. However, when the whole video is still, the noise will greatly affect the result. Take the bird fly as an example, the pure sky may considered to be moving fast by the flying bird. So how to overcome the noise in the video is a problem need to be solved in the future.

The function called in the main file is: BrutalMovDetector(P1,P2,block\_size, blur\_flag,Blur\_range). The last two arguments are the application of low pass filter in the next HEXBS algorithm. However, we can introduce the idea into the brutal force method to see the improvement.

* 1. Hexagonal base search

Hexagonal base search is a much faster way of determining (O(n)) the movement compared to the brutal force(O(n^3)). The algorithm is attached in the paper. The problem of the original algorithm is that it has a great chance stuck in the local optimal. In other word, if the object moves away from the center, the algorithm is hard to track the object. There are mainly to ways to solve the problem. First one is to impose a low pass filter onto the picture so that the trace of the moving object can be seen by the center point. The other way is to apply simulate annealing into the algorithm. In addition, enlarge the size of the block can mitigate the issue, because the block can see further.

So there are two version of HEXBS algorithm in the HexMovDetector(P1,P2,block\_size, blur\_flag,Blur\_range) and HexMovDetectorSA(P1,P2,block\_size, blur\_flag,Blur\_range). The simulate annealing function also combines the low pass filter part to see if the idea works.

* 1. Other functions

We use the MAD to determine the difference between the blocks. The input is (seg, P2, position, block\_size). Seg is the tested block, P2 is the next frame, position contains the information of the compared block coordinator in the P2 and block\_size is the block size.

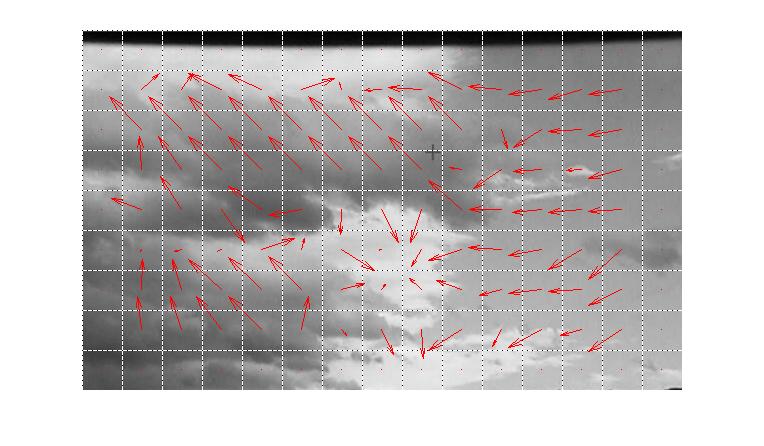
The ImagePlot function plot the picture and superimposing the motion vectors onto the picture. The effect will be shown in part four.

1. Results



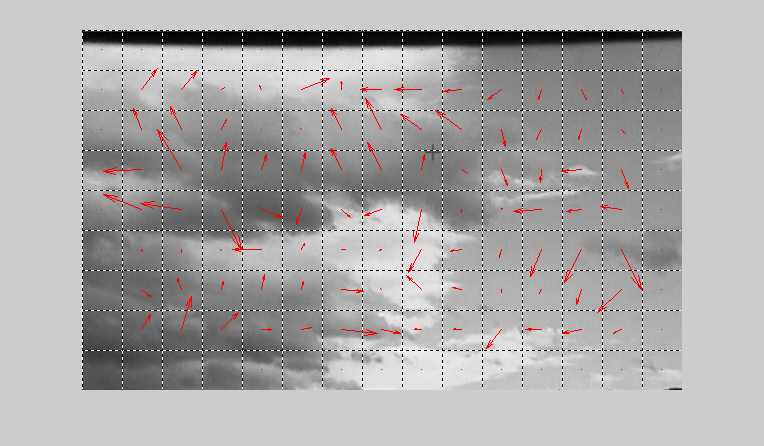
the left picture is frame 1 and the right one is frame 2. From the pictures, we can see the cloud moves from right to left.

The result of the brutal force is the following:

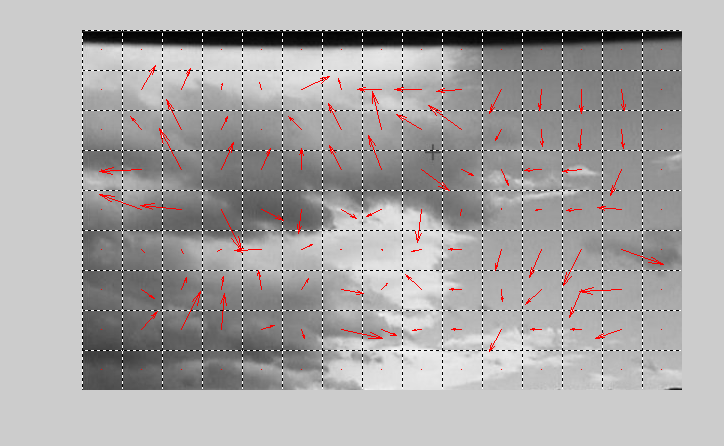


The overall trend of the cloud is correct. However, it seems that each block has the overshoot problem. The movement of the block is exaggerated, especially when the cloud is a big piece. This is because the noise, not the pattern, determines the movement. In addition, the speed difference of the front view and back view confuses the algorithm. The white part in the center of the picture is a good example.

The results of the HEXBS are worse. The results vary greatly according to the low pass index. The low pass is the blur range. Here the range means the area which the center pixel does the average.



The above picture is the HEXBS no SA version when set the blur range to 7. When the low pass filter is turned on in the SA version, ther is no difference between the SA and noSA version. The following is the SA version.



The difference of the two is not that large. The overshoot problem is not as severe as the brutal force one. However, only the boundary blocks of the cloud shows the correct trend. When the block is mainly white or dark, it will fail to track.

1. Further discussion

I think there are mainly two aspects need to be investigated.

The first one is how to overcome the noise. It causes overshooting in the brutal force algorithm and confusion in the HEXBS.

The second one is how to tune the parameters in the program. This includes the interval of the two frames, low pass filter parameters and SA parameters. For interval issue, since the movement has a great coherent between the two frames. If the interval is too small, then the noise will have large effect. On the other hand, when the interval is large, the algorithm is hard to trace the movement and causes confusion. So finding the balance is important, this will involve the speed of the cloud. For low pass filter and SA parameters issues, they are tightly connected to the correlation of the two frames. If the correlation is high, then the range of low pass filter only need small number. The larger the range is, the lower resolution you will get. So never try the range with a big number. Also, as the correlation increase, the SA doesn’t need to be that aggressive.

So how to fuse all this consideration is what I want to know.

There is one potential solution comes to my mind that what if we only determine the boundaries blocks of the cloud.