COMP3419 assignment 1 techniclal report

Shuwei Zhang 470425437

Task A: Motion estimation and visualization

The K value determines the size of block. And therefore, a smaller key value results in more blocks. More blocks is definitely more accurate for motion estimation for the movement of a block can be represented more precisely, but it will also take longer time to calculate, therefore lowing the performance.

For this task, I used a K value of 45 after calculation, analysis and experiment. The value 45 will producing a 12\*16 block size, which is not much, and therefore being very easy to manipulate and has high efficiency, with its accuracy preserved.

For 2 frames Fi and Fi+1, since a block from Fi needs to calculate the SSD for all 8 blocks surrounding it to decide the minimal SSD, the Fi+1 image was padded with white(255,255,255) pixels with its height and width of K. This will make the calculation easier, and they will have no impact on actual minimal SSD calculation since the SSD between these white blocks and the vanilla blocks was so huge that they are always ignored when trying to find out the minimal SSD block.

I used a threshold of 200 SSD at K = 45, for determining whether a block belong to the object. If the minimal SSD is lower than 200, then this block is considered ‘background’ and therefore no motion estimation operation is applied to it. Therefore, I can avoid drawing many unnecessary arrows on the background, which we don’t need to care about.

The way I visualize the motion is slightly different from what was stated in the assignment documentation:

For each block Bi, all SSD values between its corresponding block Bi+1 and its surrounding block is put into a 3\*3 matrix.

|  |  |  |
| --- | --- | --- |
| 0,0 | 0,1 | 0,2 |
| 1,0 | 1,1 | 1,2 |
| 2,0 | 2,1 | 2,2 |

The number pairs shown in this sample matrix is the index for each element.

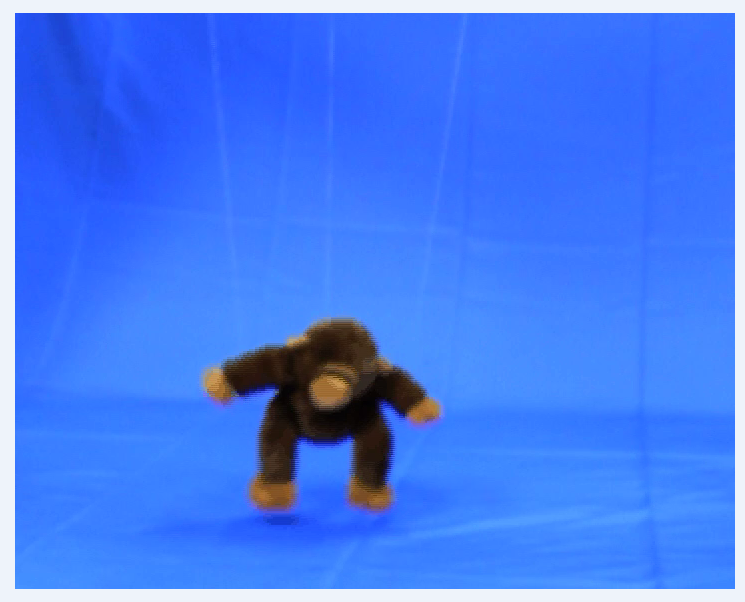
|  |  |  |
| --- | --- | --- |
| 99 | 98 | 97 |
| 96 | 99 | 45 |
| 99 | 88 | 89 |

The number shown in this sample matrix is an example of possible SSD value, while in this case ‘45’ is the minimal SSD, we can assign the ‘spatial reference’ of this element --- (1,2), to the SSD map M that has the size of (W/K, H/K) (16\*12 for K = 45) . for those minimal SSD lower than the preset threshold, the spatial reference assigned to M is (-1, -1)

This is done for each block, therefore producing a spatial reference for each block, and storing all of them into the SSD map M. next, we can use M to draw arrows on the frame Fi.

Using the same example as above, suppose for block Bi, the spatial reference for that block is (1,2), that means the block needs to draw an arrow linking its centre and the centre of the block at right. If the reference is (2,0), that means the arrow is linking its centre to the centre of the block at its bottom-left corner.

In this way, the motion can be visualized.



This is a screenshot of a sample motion visualization for 2 connecting frames. Note that the arm of second monkey has bent to the left, as the first image’s arrow shown.

Task B: Digital video processing

This task has 2 parts:

1. Dynamic background replacement
2. Object replacement

For the background replacement, the procedure I have taken is:

1. Extract every single frame of each video (monkey and water)
2. For each monkey frame, try to get non-background part of the video (monkey part)
3. Replace corresponding pixel of water video with monkey part extracted from step 2, and do this for each corresponding frame of the water video. The nzwly composited frames are placed into a new folder
4. Combine these newly composited frames into a new video.

When I was doing step 2, an issue happened: the program left out some part of original background, and determine some part of the monkey’s body as background, this primarily happens on monkey’s face:

like this

The method I used to distinguish whether a pixel belong to the background or belong to the monkey, is by looking at the ‘blue’ value from the RGB value of the pixel. If a pixel has its blue value beyond one point, like 120/255, then this pixel is considered background, and therefore not used for the composition.

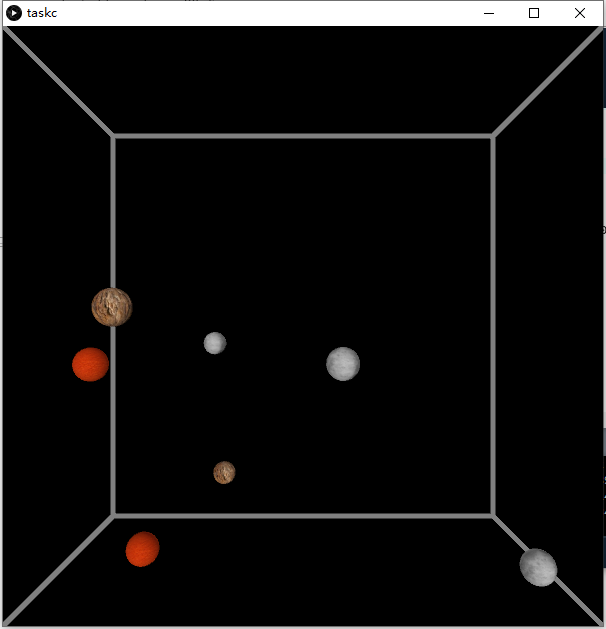
The problem raised from this method is that, for some of the left-out backgrounds, their blue value may be less than the threshold I used. To solve this, I simply lowered the ‘blue’ threshold to 80.

Another problem is the monkey’s face. After analyzing the cause, I found out that, the face of the monkey is nearly white. Since a white pixel has its RGB value (255, 255, 255), this makes a facial pixel has a blue value beyond 80, and therefore being considered as a background pixel. To solve this problem, I added another constraint: if a pixel has not only its blue value > 80, but also its red and green value > 120, then this pixel is considered ‘white’ pixel, therefore excluded from the background blue pixel. This effectively solved the problem, producing a satisfying result.

The object replacement part of this task was not done due to technical difficulty.

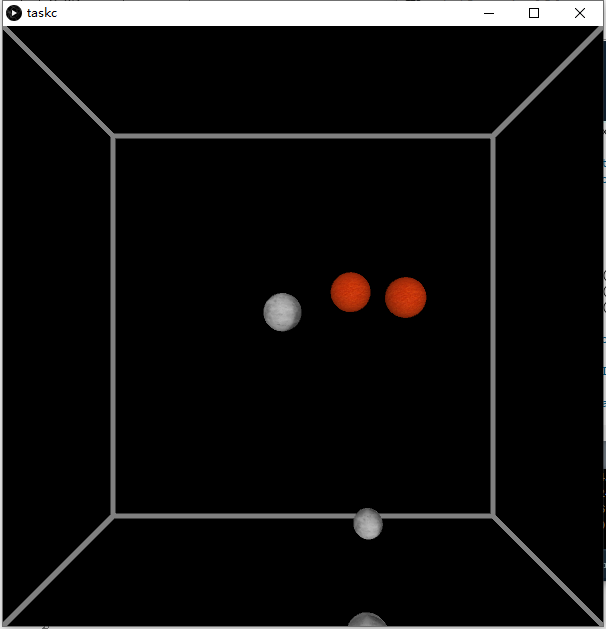
Task C. 3D animation scene

1. The balls are randomly equipped with one of the 3 textures from my texture pool.



When a ball object is created, it will choose 1 random number between0, 1, 2, each of them represents a texture.

1. The balls are launched with random speed from where the mouse clicked, and the Z value is set to 0 by default, since mouse is 2d object.



This is how it looks like after rapid clicking on 1 same point on succession. Note that these 3 balls are flying to different directions.

When the mouse click, a ball object is created with its x, y value equal to mouse position, and its speed for x, y, axis is chosen randomly from -20 to +20, while for the z axis is chosen from -40 to -1, since we don’t allow the ball to go straight ahead.

1. Whether a ball touches a wall, it’s judged by if the position of its centre has exceed the difference between the constraint of wall(600 pixel here) and its radius(20 here).

If a ball touches a wall, then it will bounce back, by changing its direction. (time the speed along that axis by -1)

1. If a ball hit another ball, both balls will have all their speeds timed by -1, therefore changing their direction completely. Whether a ball hits another ball is decided by if their distance exceed their sum of radius.