Crowd Motion Data Capture

Shenshu Zhou <shenshu@ualberta.ca>, Justin Wu <justin5@ualberta.ca>,
Xuan Chen <xuan13@ualberta.ca>, Yue Yin <yin3@ualberta.ca>

Abstract

The purpose of this project is that to get an overview with how to capture the data from a scene with a large amount of people and make it errorless, and also make it applicable to simulation. By recording a real video from light railway transition station, the project should be able to detect all the people's walking trace and produce an output file from Haar Cascade, and use the Unity engine as simulation stage to read the output file and simulate the crowd scene. In the end, our results show that we did some improvement from another project as applying the detection output into the real scene. In addition, we did few extra work like improve the result quality to avoid the clone effect, and fixed some bugs from the last year project's code.

Introduction

The motion data capture technology are attracting more interest and drawing attention to people. It can be used on video surveillance as a method for improving both public safety and the design of areas subject to crowd traffic. Due to this increase in surveillance, there have been a variety of methods developed for analyzing how pedestrians in a crowd interact with each other and the environment.

Fig 1. Example of a crowd scene from a camera of a mall

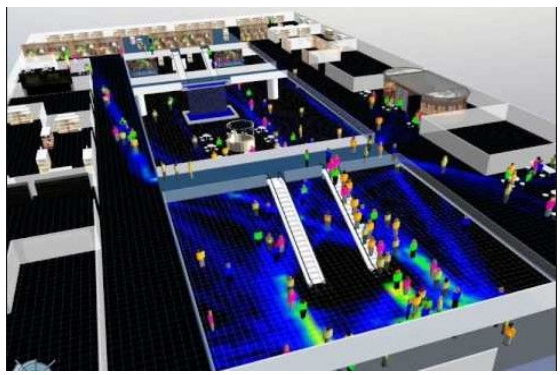


Fig 2. Example of the simulation from the malll after analyzing the detection

In this project we are getting the result work from last year's group and based on what they did we are going to improve the quality, reality and correctness for the results. In this case the LRT station are selected, as a good example of crowd scene, our purpose is to simulate the crowd scene accurately in a real world scene.

Fig 3. The result simulation from last year

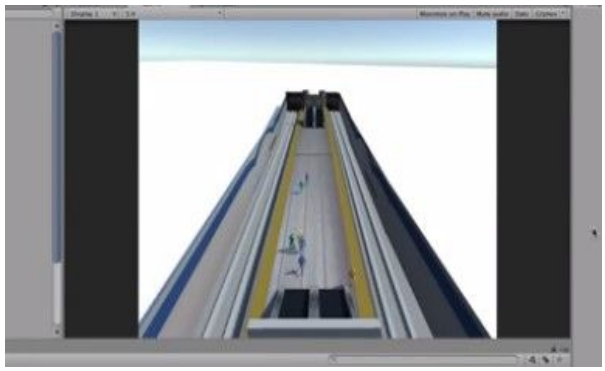


Fig 4. The result simulation from this year after make it applicaple to some real scene

The whole process can be generally separated as three steps. The first is to get an video from the real word, but due to the detection problem in this project we are only recording the video without the train in LRT station since the detection algorithm may detect the train as a person too which cause the error. Secondly, ****. In the end, we insert the output data file into the unity program and monitor the scene to check the correctness of the detection.

In addition work, to make it real, we put the LRT station models into the scene instead of the white board, which is more convincible, since the original project only used a white board as the background, and it's hard to check the correctness. After the success of putting LRT station model in this case, it is not hard that we can develop place models into the scene to make it really applicable in real world in future.

Also, McDonnell *et al.* conducted a study to test the perceptual abilities of people viewing a simulation that used cloned models and cloned motion, which cloned models were more easily perceived than cloned motion. It was found that the clones could be effectively masked by changing the colors of the models or their spatial orientation. In this case we add some extra code after unity program produced human models, we colored them with different colors to avoid the clone effect, as the quality improvement.

Related work

In this project, we did some additional research about the project.***

Implementation

- **Improvement of place models setting**

  We get the LRT station model from another project group and edit a little to make it fit with this project and place it into our scene.

  However, a new problem came out that the integration of using optical flow, it is problematic, and people are not walking on the right way. In previous project since it is a white board, the problem is not obvious since there is no real place model, now after putting the place model into the scene, the correctness of the detection is important for whole simulation.

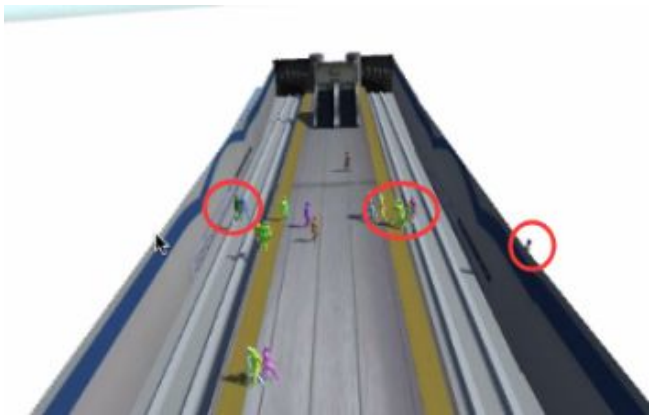


Fig 5. Integration in Scene Using Optical Flow

- **Instead of optical flow, we used moving object detection as new algorithm**.

Optical flow is based on tracking every single frame. It is counting the difference between two frames by pixels. By getting a 2D velocity vector, with the information of direction and speed of motion is assigned to each pixel in a given place of the picture.

The optical flow originally transfer the 3D world information into 2D and simply describe it as I(x,y,t) which means 2D dynamic brightness function of location and time. The expression is:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)$$

However, a problem for that is it uses a lot of system resources and it will be affected easily by camera movement, and it is not that accurate since it accounts every single frames' pixel difference.

In this case, we decided to use moving object detection instead. The difference between optical flow and object detection is that the object detection will find a object and compare the distance between current object and objects detected from pervious frame.

If there are multiple objects from the current frame, it will simply pick the pair with the smallest distance.

If there are multiple objects and one object does not have a match then we can assume it is a new object and create a new track for it.

However, if there are no new objects added into a number of frames, then it will automatically end the track to decrease the system resource uses since it is no longer to be updated with new object locations. Fig 4 is the correct result by applying the algorithm with object detection.

- **Improvement of avoiding "clone attack"**

According to McDonnell *et al,* in a large crowd simulation the clone models are very common way to reduce the resource uses. However, to make it more real as a scene simulation, we want to do some improvement in this area.

By using all same model in one scene, we changed each model's color to use it to justify the direction. People going forward will be colored blue and backward will be colored green. However it is not that simple to do in Unity. Since the model is already created, and if we want to change the color we must create a new instantiate as a copy of original model and then get into the component of skin mesh renderer and change the RGBA value in shader. The reason we are doing a copy instead of using original model is becasue the original file is read only.

As the result, all the models are correctly perfomed and audiences will be easier to see what is going on for whole simulation, the people going forward and backward are clearly marked with different color.

Fig 6. The color based on direction + track clean up

- **Improvement for moving code in Unity**
  ***

Result

We took several videos from LRT churchil station and use Haar Cascade to analyze it. The error still exist but the percentage for error rate is much lower compare to last year's project.

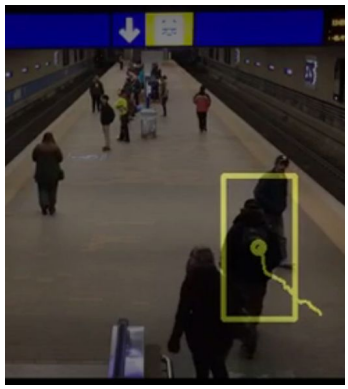
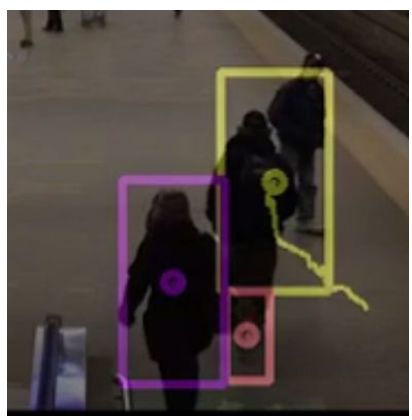


Fig 7. The Haar Cascade Object Detection

As the Fig. 7 shown, the detection will automatically follow the moving black pieces which is a person is walking into the center area of the station. In same time, all the other people are not moving so the detection is not able to track them.

After few seconds, the other people are moving. The object detection found all the moving objects between frames and match them together. However, the first error also exist. Because all the people were wearing the black clothes, once they seperated they body a part, the object detection would define that as two seperated objects, but it may combine two different



object together since their pixels are way to close to each other in 2D frames. As Fig. 8 shown, a person's foot are seperated as a independent object from object detection, and in same time, the square gets bigger because the object detection combined the people on the right together into the square.

Fig 8. The first error exist in Object Detection

In the end of the analyzation, the video wiill recored as Fig. 9 shown, each line represents a movement action from an object, in this case most of moving obects are people. By recording the frame number and each point's coordinates as a 2D vector from the line, the final output file will be recorded as Fig. 10 shown, first it will record different objects exist in different, in Fig. 10 the first line means an object will exist from frame 0 to frame 1008. After a line of space, it will start record the real coordinates of the object, like the first object, it exists for 1008 frames, it will read next 1008 lines for position. In this case it ends at (915,319), which means an object exist in (1021,317) at frame 0, and end the existance in position (1008,228) at frame 1008.

Fig 9. The track result

```
0 1008
0 1007
0 936
0 1007

1021 317
1020 317
...
1007 229
1007 229
1008 228
915 320
915 319
..
1005 381
1002 382
999 382
998 382
```

Fig 10. The output file

After getting the output file, use Unity program we made to read the output file, and it will automatically run the simulation as Fig 6 shown above. The main code can be found on Move-Pedestrians.cs script, based on last year group's work, we did a lot of improvement and extra implementation as above. The void update and void OnGui is the funtion it will run every single frame, the void start funtion is the initialization. So what we code is to set up everything in start function, then read the file and create the model or make a movement in each frame.

As the simulation runs, there is a little problem is that the person will be automatically disappeard after they finish their walking action. It is kind of unrealistic but due to the limitation from algorithm and coding, it is going to take a lot of time to fix the bug. Since there is a time limitation for project, it is not a main issue we have to consider.

Conclusion

In this project, we learned a basic concept of how the motion data capture is applied in real life. During the project, with over 30 research papers background information's support, our final project demo is running successfully as what we expected in beginning of the project. However, the project still has a lot of space to imporve, for both the algorithms, settings, and coding parts.

We have decomposed the the whole process of detecting and simulating into a three-stage working process. The first stage is to get the video clip from camera, with the camera information. Then the second stage is detecting, which is composed by many detailed

steps in algorithm, it is the key stage of the whole project and it is the most difficult part. In steps of transfering 3D world information into 2D world position, it is acceptable if error exist, but we have to choose the best algorithm which makes the least errors. After the second stage we will have an output file for the last stage, which is the simulation. It will automatically read the output file and try to reproduce the scene from the camera recording.

It is not only a practice of multimedia technology, but also a strategy planning of the future development. It is meaningful, because we may actually apply it in our life in future. For example, we can apply it into a crowd shopping mall to insure the public safety, or use it into a airport simulation to estimate the basic information they needed. We hope our results may encourage some other groups to take the project and continue work on it in future, to make the project better, and more applicable for our real life.