Pull and Push: Proximity-Aware User Interface for Navigating in 3D Space Using a Handheld Camera

Mingming Fan and Yuanchun Shi

Department of Computer Science &Technology, Tsinghua University, P.R. China
fmmbupt@yahoo.com.cn, shiyc@tsinghua.edu.cn

Abstract. In the 3D object controlling or virtual space wandering tasks, it is necessary to provide the efficient zoom operation. The common method is using the combination of the mouse and keyboard. This method requires users familiar with the operation which needs much time to practice. This paper presents two methods to recognize the zoom operation by sensing users' pull and push movement. People only need to hold a camera in hand and when they pull or push hands, our approach will sense the proximity and translate it into the zoom operation in the tasks. By user studies, we have compared different methods' correct rate and analyzed the factors which will affect the approach's performance. The results show that our methods are real-time and high accurate.

1 Introduction

Many 3D interaction tasks need the zoom operation. Suppose that if we want to wander in the 3D campus, we may need to go ahead to watch the landscapes. In order to satisfy this requirement, we can use the mouse to control the moving direction and the up arrow key to move ahead. The disadvantages of the method are as following. First, the operation needs relatively complex combinations of keyboard shortcuts with mouse movement and clicks. This usually operates with two hands. Second, it gives a low level of naturalness and is not a good choice for the children or people who are not familiar with the keyboard and mouse operations. In order to crack the above two disadvantages, we propose a method that users could simply pull or push their hands to move in or out by holding the camera. When they want to go ahead, they just need push their hands forward. When they would like to go back, they just need pull their hands back. Our approach needs only people's natural movement and almost need no study. Besides the naturalness, the operation only needs one hand and people may use the other hand for other operations.

Some researches [2, 5, 6] have done the familiar studies, such as the Harrison and Dey [2] try to recognize the people proximity by the camera in the computer. However, during their mode, the camera is still and the approach is not proper for the 2D or 3D interaction tasks such as the object control or the virtual space navigation. IsseU [5] is similar to our approach. IseeU tries to calculate the change in the standard deviation of the positions of feature points, which are selected in the image captured by the camera, and transform it into a zooming message. However, we analyze that it is not enough to give a high accuracy.

After having studied the previous works, we first give two methods to recognize the zooming message. After that we test the accuracy rates of them and analyze the factors which have an effect on the accuracy. Then by taking more factors into consideration, such as how to support large distance zoom, we modify the methods to make them more efficient.

2 Framework of the Algorithm

The handheld camera is just a tool for interaction. Because the camera is hold steadily in user's hand, the camera's movement will reflect the hand's movement. In order to detect the camera's movement, first, we detect some corner points in the image frames captured by the handheld camera, then by analyzing the geometric characters of the corner points' positions, we try to decide whether the movement is zooming or not. During the following part, we propose two methods to detect the zoom and then compare them with each other to see which one is better.

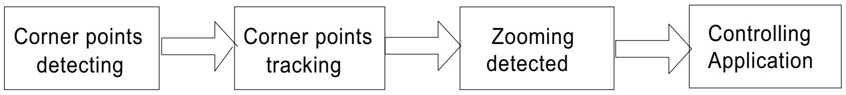

Fig. 1. The whole framework of processing

3 Corner Points Detecting and Tracking

Corner-like points [4] which are corners, with big eigenvalues in the image, are easy to find on incoming frames and are relatively stable while being tracked. Tracking the points means finding the new positions of the corner points, which appeared in the last frame, in the frames. Our approach tracks the feature points by implementing sparse iterative version of Lucas-Kanade optical flow in pyramids [1] .

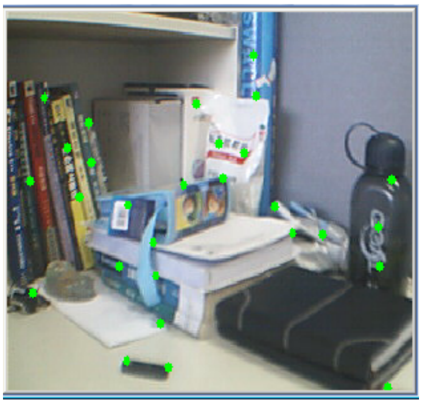

Fig. 2. The green points are the corner points

4 Zoom Detecting Algorithms

In this part, we will discuss two algorithms to detect the zoom in detail and then compare their performance.

4.1 Algorithm One: Sensing the Distance

As the figure3 shows, A, B, C, D are positions of the corner points in the last frame, the A', B', C' D' are the positions in the frame. The average distance between the corner points and their centers becomes farther when the camera zooms in, since the distance between the camera and background are shortened.

According the above analysis, first, we calculate the positions of the corner points in the last and now frames. Then, calculate the average distances among them and their centers. Finally, calculate the rate of the new distance and old distance. If the rate is over 1.0, it means that the camera pulls back. If the rate is less 1.0, it means that the camera pushes forward. In the real experiment, due to the hand's jitter, the camera may have the slight movement. In order to reduce the jitter's interference, we set a threshold to instead of the above number 1.0.

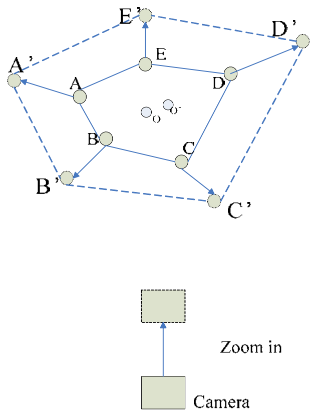

Fig. 3. Corner points' positions before and after the camera zoom in. o and o' are the centers of the old corner points in the last frame and new corner points in now frame.

4.2 Algorithm Two: Sensing the Change of the Area

As the figure 3 shows, the corner points form a polygon ABCDE. After the zoom in operation, the polygon becomes the $A^{'}B^{'}C^{'}D^{'}E^{'}$, the area of the polygon ABCDE changes to be larger. Through sensing the change of the area, we can decide whether a zoom in or out happens.

4.3 The Accuracy of the Two Algorithms

Participants

Seven participants, six male and one female, take part in the test. They use a webcamera with frame rate 30 fps(frames per second) and a pentium 4 PC with the main frequency 3.2GHz. Each of them takes the experiment for about five minutes. Before the test, they are given no more than five minutes to be familiar with the camera.

Experiment

We have rendered a 3D virtual space with DirectX 3D(see figure4). People are asked to use the camera to go ahead or back in the virtual space. We count the total decisions and the right decisions, then calculate the accuracy rate. (We ask the testers to do zoom in operation, then we count the total judgment and the actural zoom in times).

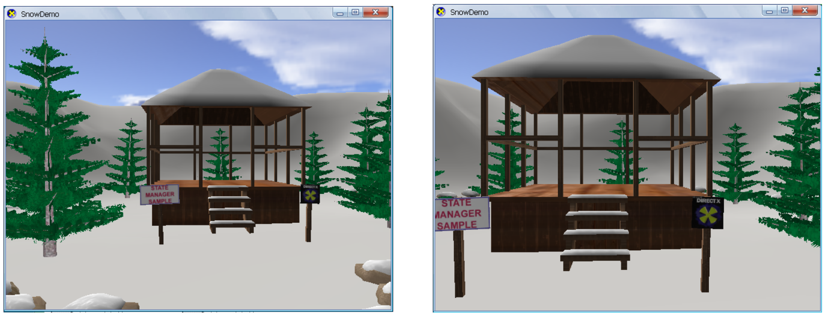

Fig. 4. The left one is the last image. When users push the camera forward, our view goes forward and the house becomes bigger than ever.

Accuracy Rate

Participants are asked to do the zoom in and out movements to test two algorithms' accuracy rates.

According to the Fitts law [3], the distance between the camera and positions of the corner points in the real world will have an effect on our algorithms. So in order to test how the distance factor affects our algorithms' performances, we do the experiments at different distances, such as 0.6m, 2~2.5m, 5m.

We calculate the seven participants' results and give the average accuracy rates at different distances in figure 5.

Discussion

From the figure5, we can conclude that:

- The average accuracy rate of the algorithm2, which senses the proximity by calculating the changes of areas, is higher than the algorithm1, which detects the zoom by calculating the changes of distances.
- As the distance between the camera and the corner points' real world positions increase, the accuracy rate declines rapidly. At the distance about five meters, the accuracy rate of algorithm one has been below 50% and the accuracy rate of algorithm two is almost equal 50%.
- The zoom detecting algorithms are totally sensitive about the distance. Within the five meters, the algorithm can keep the accuracy rate above 50%. Within the

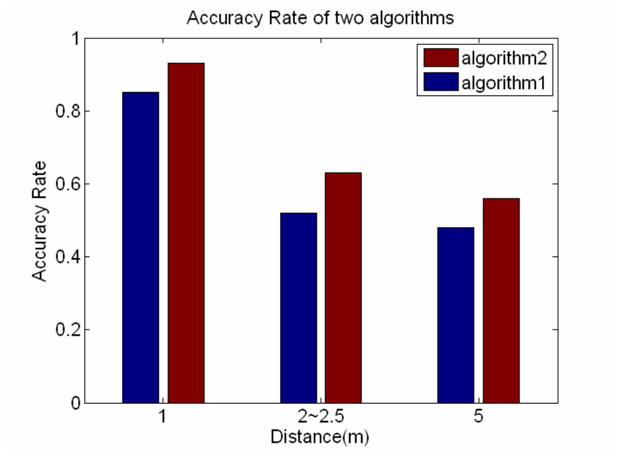

Fig. 5. The average accuracy rates of seven participants' results

distance 1~2m, the two algorithms can keep the accuracy rate over 80%. This results guide us that our hands had better push or pull the camera in a direction in which there are some objects with in 1~2m.

- The experiments are taken by seven participants who only use the camera to operate less than five minutes. The results show that they can operate the zooming easily and need less time to practice.

4.4 Large Distance Zooming Support

Using a Finite State Machine

From the questionnaire, they reflect that the approach is not suitable for moving a long distance at one time. If they want to go ahead in the virtual space for a long time, they must keep pushing or pulling the camera for a long time. This is impossible due to users' moving space is limited. In order to crack this hard nut, we give the strategy that we first detect the movement. If we continuous detect the zoom in movement for two times, then we simply think that users want to zoom in. In this situation, we output the zoom in. if users want to stop zooming in, they can pull back the camera. If our approach detects the zoom out movement for two times, we think that users want to pull back. The whole procedure can be described as a finite state machine (FSM)(Figure 6). During the zoom in / out state, our approach will output the "zoom in/out" decision. Suppose the current state is "zoom in", and current judgment of the algorithm is "zoom out", then the counter Count1 will add by one, then we examine whether Count1 is two or not. If the count1 is two, then the state will change to "zoom out" and the output is "zoom out". But if the count1 is not two, then the state will still be "zoom in" and the output is "zoom in". if the current state is "zoom in" and the current judgment is "zoom in", our approach will output "zoom in" and set the count1 as zero.

The reason why we use the counters when the state is changing is to make our algorithm stable. Since the camera is held in people's hand and the hand will be

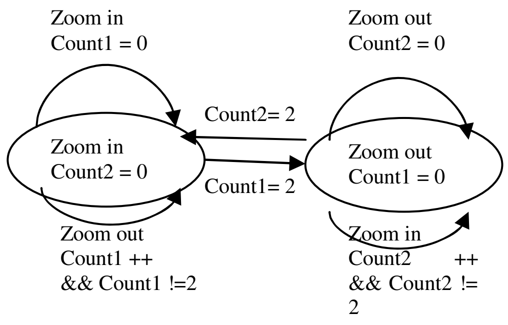

Fig. 6. The finite state machine of zoom in and zoom out

naturally jitter while suspending in the air. And this jitter maybe causes some zoom in or out motion. If the approach does not use the state machine or two counters, then only a slight noise will cause false decisions.

Experiment for Testing the Effect of Finite State Machine

Participants and the hardware conditions are the same as the above experiment. We have done two zoom detecting algorithms, one uses the finite state machine and the other does not. All participants have required to do zoom in and out movements alternatively for about five minutes. Each have done the experiments twice, one time is without the finite machine and the other time with the finite machine. The average distance between the corner points' positions and users' hands' positions is about one meter which is good for our algorithm to work. The average accuracy rates are calculated.

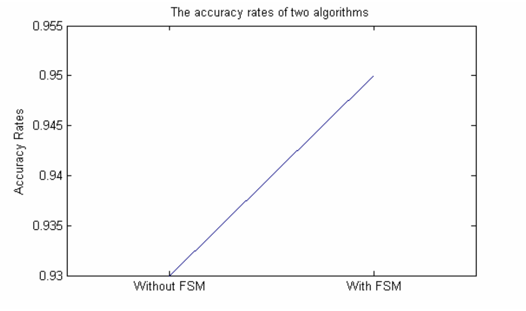

Fig. 7. The accuracy rates of the algorithms. One is 0.93, the other is 0.95. The result shows that the FSM improves the algorithm's performance.

After the experiments, participants give us some valuable feedbacks, based on which we conclude the following

- With the finite state machine, they can continuously zoom in or out. While they use for the object controlling, they can magnify or reduce the size of objects. While

using in the virtual space wandering, they can continuously go ahead or back in the scene.
- They can switch between the zoom in and zoom out movement with higher accuracy rate.
- Before the hand's motion state changes to the other one, the counter must count to two. Since the frame rate is 30fps, then the delay time is 6.6 milliseconds which is almost real-time to our eyes. By using the finite state machine and the counter, the accuracy rate is improved.

5 Applications

As we have claimed that the given proximity-aware algorithm can be used in the object controlling and the virtual space navigation. In the object controlling task, people can magnify or reduce the virtual object by pushing or pulling the camera. The application is shown in Figure8. In the virtual space navigating tasks, the algorithm can be used for going forward or back in the scene which is especially useful for the games(Figure4).

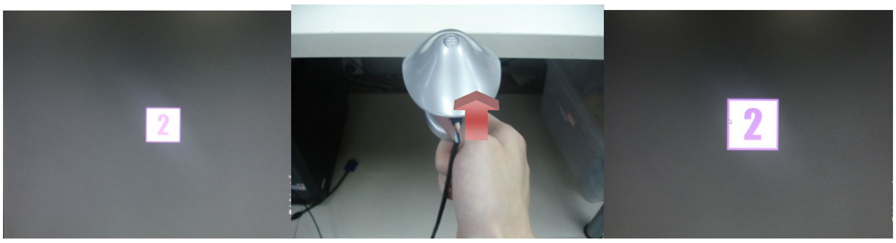

Fig. 8. The left one is the former image of a cube, when the user pushes the camera forward, the cube's size will increase as is shown in the right picture

6 Conclusions

In this paper, we have proposed and compared two proximity-aware algorithms. From the experiment's results, we conclude that the algorithm two has a better performance. In order to support the large distance zoom and improve the accuracy rate, we take in a finite state machine. Comparing to the traditional mouse and keyboard operation, our methods are much more natural and easier to learn. Our approaches are real-time and have high accuracy rate. Our methods can be used in the object control and virtual space navigating tasks to fulfill the zoom function.

Acknowledgements

Specialized Research Fund for the Doctorial Program of Higher Education, China, No.20050003048 and this research is also supported by the Nokia Research Center.

References

1. Bouguet, J.V.: Pyramidal implementation of the Lucas Kanade Feature Tracker Description of the algorithm. Intel. Corporation Microprocessor Research Labs (1999)
2. Harrison, C., Anind, K.D.: Lean and Zoom: Proximity-Aware UserInterface and Content Magnification. In: Proc. CHI, pp. 507–510 (2008)
3. ISO. Ergonomic requirements for office work with visual display terminals (VDTs) - Requirements for nonkeyboard input devices. ISO 9241-9 (2000)
4. Shi, J., Tomasi, C.: Good features to track. In: Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn., pp. 593–600 (1994)
5. Sohn, M., Lee, G.: ISeeU: camera-based User interface for a handheld computer. In: ACM MobilCHI 2005, pp. 299–302 (2005)
6. Wang, J., Canny, J.: TinyMotion: Camera Phone Based Interaction Methods. In: Proc. CHI 2006, pp. 339–344 (2006)