Outdoor Navigation with Handheld Augmented Reality

Lianyao Wu[1,2], Xiaoqing Yu[1,2*]

[1]*School of Communication and Information Engineering*
[2]*Institute of Smartcity, Shanghai University*
Shanghai, China

hi501@foxmail.com

***Abstract*—Recently, outdoor augmented reality (AR) has become readily available. Many smart phone based navigation applications provide AR capabilities. However, most of them are to show virtual points of interest (POIs) in the map or add virtual information to the real scenes captured by the camera. We found these applications are helpful when users are looking for interesting places, but it is inconvenient for users who intent to go to the destination. Also, arrows for indicating directions on navigation maps are not intuitive when users are on the way. In this paper, we proposed an outdoor navigation system combined with Baidu map using AR technique where a virtual model can guide users to their destinations.**

Keywords—Augmented reality; Baidu map; outdoor navigation; global positioning system

I. INTRODUCTION

Since the 1990s, with the rise of virtual reality technology, augmented reality (AR) technology as a branch of virtual reality (VR) has attracted more and more attention of researchers.

Nowadays, AR has many applications in real life. One of the most popular applications is handheld AR navigation system. In this area, there are two research directions, indoor and outdoor. Indoor navigation is difficult to use global positioning system (GPS) to locate user in the buildings. In [1,2] both use markers to show the direction of movement. Reference [1] just put the marker on the ground and show the route on the virtual model of the building. And [2] is more flexible by using natural markers seen everywhere such as escape icon. But from the description of the two articles, it is impossible for either of them to achieve the effect of indoor navigation without markers. Because of large space, indoor methods cannot be applied outdoors. Therefore, marker-less methods are considered into outdoor cases. Simultaneous localization and mapping (SLAM) algorithm [3] is one of the marker-less methods which is mainly used in robotic area. But it cannot guarantee real-time and long-term performance on handheld devices. So, a hardware based method which takes less time and complexity would be a better choice. This method can run in real time on handheld devices although it is not stable enough than SLAM.

In this paper, we proposed an outdoor AR navigation system based on GPS and inertial measurement unit (IMU). It is different from the traditional AR navigation system in [4,5]. They both focus on annotations of outdoor AR. Reference [4] proposed an online AR annotation framework. And [5] uses the Metaio package to develop its AR function to learn raw images possible. This proposed system is more like Pokémon GO. It is developed on Android and it mainly depends on Baidu map which needs GPS to get high accuracy position data and Unity3D which renders virtual model.

*Corresponding author.

II. BAIDU MAP

Baidu map is a well-known navigation application. Baidu offers a software development kit (SDK) based Baidu map for developers. Its main features include maps, POI, geocoding, path planning, location and offline maps. Based on this, users can use the interface of Baidu map to conveniently realize the function of map positioning and navigation on the smart phone, and the desired positioning navigation information can be conveniently obtained through the provided monitoring function.

The entire navigation process is divided into two threads, one is the map interface update, and the other is the backstage data processing. The backstage receives and processes positioning data and obtains real-time updated path planning information and POI information. After getting the latest data, the backstage passes the data to the front desk to update the map.

A. Positioning Acquisition and Processing

Real-time positioning is based on GPS, Wi-Fi and base stations [6]. Wi-Fi and base station positioning is used here to compensate for GPS positioning whose error accuracy is within 1 meter. These positioning data are obtained from application programming interface (API) Baidu map provides.

Real-time positioning sometimes has a large delay or no signal, so sometimes there will be a large gap between the two positioning points. To make up for the gap between them, it is necessary to interpolate between the gaps to make points appear to be moving smoothly. The idea of smoothing algorithm follows the following three steps and its flow chart is shown in Fig.1.

1) Obtain positioning data: After the listener obtains location data, the data are discriminated whether it is not empty or obtained from one of the Wi-Fi, base station and GPS. If the condition is met, we declare a variable to store those data. This variable stores the positioning point and positioning time. Meanwhile, we define a list to store these variables.

2) Data storage and processing: When the list previous step defines stores less than two data, we return to the listener and

wait for new data. In addition, when the data is greater than 5, the first element of the list is removed according to the first-in first-out (FIFO) principle. Then we get average speed by dividing the distance between each point in the list and the current point by the time gap.

3) Data analysis: We multiple the set of speed data in the second step with Earth weight data provided by Baidu map to get the corresponding score and accumulate each point's score. Finally, we compare final score with the experience value 0.00000999~0.00005. If final score is within the range, we calculate an intermediate point between the current point and the previous point and store it in the list. If it is outside the range, it will be directly stored in the list.

The above step *2)* and *3)* can be summarized as (1).

$$\text{Score} = \sum_{i=1}^{5} \frac{dis_i}{time_i} * coef_i \tag{1}$$

where dis_i and $time_i$ are respectively the distance between each point and current point and time gap. $coef_i$ is Earth weight data provided by Baidu map and Score is the final score.

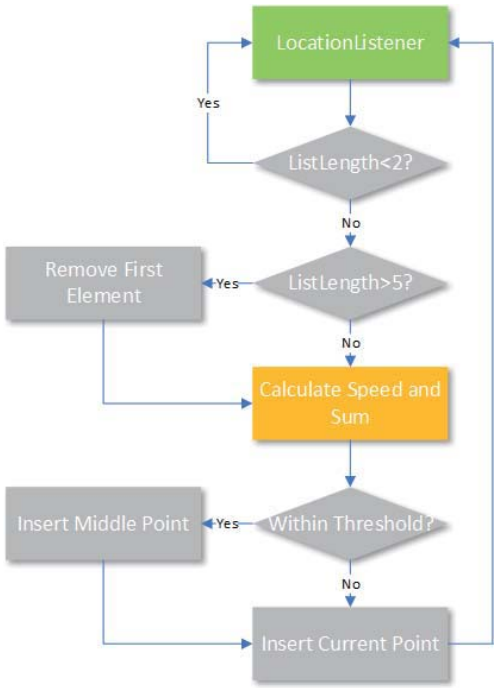

Fig.1 Positioning point smoothing process

Through the above three steps, we will eliminate the sudden jump between two positioning points and get a smooth point motion. Thanks to this algorithm, the virtual model movement is more stable in the AR interface. In addition, the distance between two positioning points is calculated based on Baidu's latitude and longitude which exists a conversion relationship with GPS's. The positioning time depends on the time device that comes with Android itself. The starting time is January 1, 1970, and its unit is milliseconds.

B. Path Planning Acquisition

The main path planning method is based on A* algorithm [7] which is a heuristic algorithm. Its formula is as follows.

$$F(n) = G(n) + H(n) \tag{2}$$

where n is a path node, $G(\cdot)$ is the actual cost of the initial node to n, $H(\cdot)$ is the estimated cost of the best path from n to the target node which has many optimization methods, and $F(\cdot)$ returns the total cost form the initial node to the target node. This algorithm is applied to Baidu map to achieve its path planning function which has been encapsulated.

Like positioning acquisition, path planning data is obtained from API Baidu Map provides. We get listeners through path planning search interface and path planning results. There are several path planning methods available, but what we need is only walking route information. At the same time, it should be noted that path planning needs real-time updates because of the real-time changes of users' position. Therefore, we set up a callback function that performs a path planning when the users' position changes.

Path planning is a very important part of the entire system. When users set a destination, path planning provides a path from the current location to the destination. This also provides data support for the virtual model in the AR interface to guide users to their destination.

C. Phone Direction Acquisition

Baidu map provides a large error in the orientation of the smart phone when turning off GPS. This issue affects the user experience greatly. Meanwhile, to avoid large errors may occur in the case of opening GPS, the orientation direction is forced to use an alternative whose orientation angle information obtained by the smart phone direction sensor.

The direction sensor not only just gets the orientation of the phone, but it can reflect slight changes in the three physical axes of the phone on the data. By this way, we can precisely adjust the movement of the virtual model so that it does not show a significant drift. We specifically set up a listener for the orientation sensor and register it listening angle information in the backstage.

D. POI Acquisition

Acquiring POI information is to make users more convenient find their destination. When user enters the first few letters of the place, the input box pops up a list of matching places and a related introduction drop-down box. Then user simply selects the correct place in the drop-down box as a destination.

III. Model Rendering and Control

In this section, we will introduce the main part of this AR system, virtual model rendering and control. The most common rendering library is OpenGL which occupies a large part of virtual-actual combination tools. It applies to lightweight model rendering, but its support for complex model animation is not good. Therefore, we turn to use the more powerful rendering

engine called Unity3D. Unity3D provides model animation switching and model control capabilities that greatly simplify our development steps. Before continuing to introduce AR interface, we must first understand the message mechanism of the entire system as shown in Fig.2.

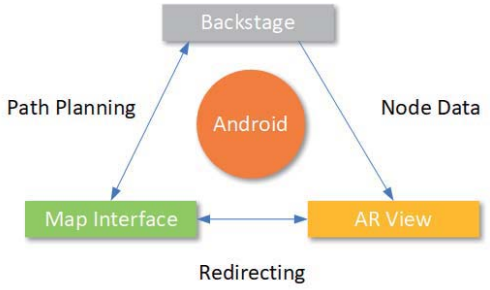

Fig.2 The message mechanism of the entire system

In Fig.2, we clear the links between the map interface, AR interface and backstage. The backstage passes data to both interfaces throughout its life cycle and the two interfaces redirect to each other [8].

A. Model Rendering and Animation

Unity3D natively supports the import of 3D models in various formats, especially supporting import of animations in .obj format. So, we no longer care about the concrete realization of the animation, just focusing on the mutual switching of the animation.

The model has a total of six actions between which we need to establish a connection. For example, the model is not always show the status of running. It will stop to the status of idling and switch to the status of walking. These actions have clear connection relationships definitely showed in Fig.3.

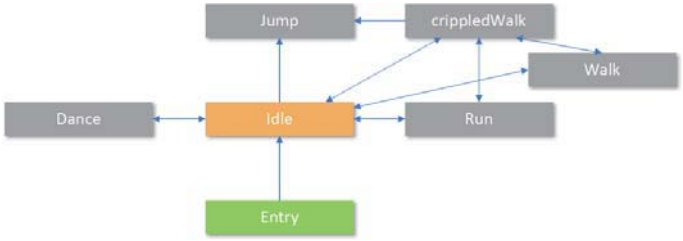

Fig.3 Model animation FSM

The animation clips in Fig.3 represent the different actions of the model, and the change of the model action is represented by the animation switching in Unity3D [9]. This process can be described by the finite state machine (FSM). This FSM structure helps us clearly understand the transition relationship between the states of the model action. The pointing arrow between two actions in Fig.3 is the transition to the next associated action under certain conditions, and Unity3D provides a smooth transition between the two actions to prevent inconsistencies.

B. Model Initialization

Before we display virtual model, it is necessary to layout scene. Firstly, we bind the main camera in Unity3D and phone camera with its mobile gyroscope. Secondly, we place a plane at the (0,0,0) point for the area where the model is placed. Finally, we put a collision box on the plane to prevent model out of the sight.

After preparing scene, we use the intersection of the ray emitted from the camera and the plane as the initialization position of the model. In order to direct the model toward the front of the ray correctly, our derivation process is as follows and Fig.4 is a schematic of the model placement process.

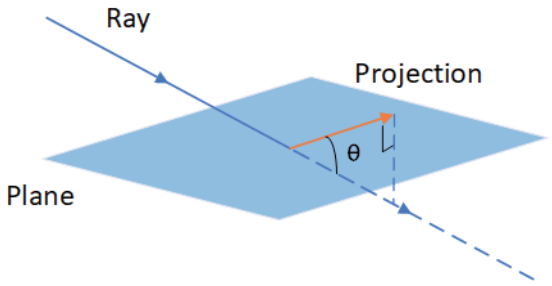

Fig.4 Projection of the ray on plane

$$\sin\theta = -\vec{d}\cdot\vec{n} \tag{3}$$

$$\vec{d}_{proj} = \vec{n}\sin\theta + \vec{d} \tag{4}$$

where $\vec{d}$ is the unit direction vector of the ray, $\vec{n}$ is the unit normal vector of the plane and it is perpendicular to the plane and upward, $\vec{d}_{proj}$ is the projection vector of ray on the plane and θ is the angle between $\vec{d}_{proj}$ and $\vec{d}$.

After calculation of (3) and (4), we get the direction of the projection of the ray on the plane. This direction will determine the orientation of the model during initialization and at the beginning, the model is back to us.

C. Model Movement

The rotation of the model relies on quaternion which eliminates universal lock generated by Euler angle. Quaternion is simple super-complex number that can be expressed as follows.

$$\mathrm{T} = \mathrm{w} + \mathrm{x}\vec{\imath} + y\vec{\jmath} + z\vec{k} = \left(\vec{v}\sin\frac{\theta}{2},\ \ \cos\frac{\theta}{2}\right) \tag{5}$$

where w is a coefficient, $\vec{v}$ is the point (x, y, z) and θ is rotation angle. Rotating a point T by quaternion is as following steps. Firstly, T is converted into quaternion $\mathrm{t}(\mathrm{T},0)$. Next, rotating this point θ degrees around a unit vector $\vec{m}(a,b,c)$ means following formula in quaternion.

$$t' = qtq^{-1} \tag{6}$$

where q is quaternion $(\vec{m},\ w)$ and q^{-1} is the inverse of q. Finally, t' is the quaternion corresponding to the new point after rotation. According to (5) and (6), we can transform Euler angle to quaternion. By this way we can know how to rotate model in Unity3D.

The deflection angle transmitted by Baidu map is based on the north as the starting point and rotates in the clockwise direction within 0~360 degrees, so the geographical North direction in Unity3D must be determined. Fortunately, Unity3D provides related API. And it should be noted that this API must be used when receiving GPS signals, otherwise the geographical North will constantly shift and finally undermine the correctness of navigation.

After determining North, we next discuss how the model moves in the space of Unity3D. Actually, the backstage will send the information of the next node on the route including deflection angle, route length, etc. Then the model moves the deflection angle θ on a circle with an initial radius and its final direction turns to the direction along the outward radius of its position. The entire process is shown in Fig.5 below.

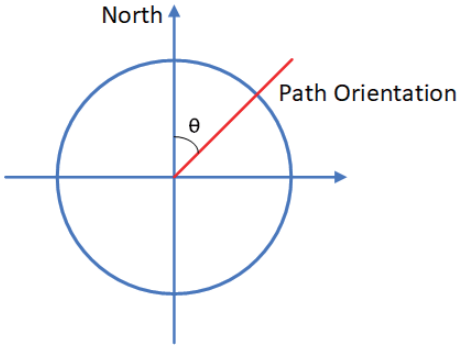

Fig.5 Model movement trajectory

In order to control the movement of the model that goes left, right, and continues when the user stops, we should use quaternion to store path orientation first. Then we determine the direction in which the model needs to be rotated based on the model position and path orientation by (7).

$$Y = Y(\vec{d} \times \vec{P}) \tag{7}$$

where $\vec{P}$ is unit vector of path orientation and the function $Y(\cdot)$ returns y which is y-axis coordinate value of the vector. When y is greater than zero, it means the path is on the right side of the model and when y is less than zero, the path is on the left side. When knowing the position of the path relative to the model, we move the model to the position of the path and turn its orientation. And when the model moves in one direction for a while, the model will trigger a running animation but when the user stops, the model will stop after advancing a certain distance.

D. Model Jitter Smoothing

In the process of users' movement, the video image will fluctuate with users' jitter, but the model's jitter is not obvious due to the Unity3D space. In order to make the model more real, the acceleration sensor of smart phone is used here [10]. Judging from the threshold whose range is $(-0.06, -0.008) \cup (0.008, 0.06)$ obtained by the test of the sensor values, when the jitter of the mobile phone is less than or greater than this range, it indicates that the smart phone's jitter is caused by user's walking. Within this range, we perform smoothing and move the model slightly to the left or right.

IV. Demonstration Results

In this section, the map interface and the AR interface are introduced. The model movement is also displayed here.

A. Map Interface

The map interface mainly includes the functions of selecting a destination, displaying the current location, and the path planning results. The map interface is built on Android system as Fig.6 (a) shows. In Fig.6 (b), the destination is marked with a red pattern.

(a) (b)

Fig.6 (a) current positioning function, (b) path planning function

B. AR interface

The AR interface implements core AR functions. In this interface, the virtual model is rendered on the camera frames. Furthermore, some interactive buttons on the screen control model scaling, path information display and model deletion. This entire interface is shown in Fig.7 below.

(a) (b)

Fig.7 (a) AR interface, (b) interactive buttons on the screen

C. Model Navigation Effect

This part demonstrates the application navigation effect. As Fig.7 shows, the model guides the user to the pre-set destination. But in Fig.7 it only shows the case that the user goes straight on the road. The key problem is the performance of the model when the user turns at the corner. On this issue, the system has good performance results.

In Fig.8, it shows the effect of the model's correct rotational movement at the corner. By the correct rotation, the model guides the user arrive at the destination finally.

Fig.8 The model turns left at the corner

The three parts detailed above constitute the complete handheld AR navigation system. First of all, the user inputs a destination and presses the start button. The system will show the path planning results in the map interface. Then, when the user presses AR button, the interface will redirect to the AR interface. In this interface, the model displayed in Fig.7 will guide the user to the destination. Finally, after the user arrives, the model stops guiding and waits for next instruction.

V. Conclusion

In this paper, the system we proposed basically achieves AR navigation by using Baidu map, Unity3D, etc. But there are still some drawbacks in this system. For example, the problem that the model sometimes turns around when turning at the corner directly affects the user experience of navigation. Another problem is that the smart phone screen shakes badly when the user is walking, and how to reduce the poor user experience caused by the screen jitter is also a concern. In addition to these two problems, it is impossible to judge whether the user is moving in time. Now it is relying on every 3.5 seconds to detect total distance to determine whether the user is moving which has a relatively large delay causing the model retention.

In general, although there are some problems, the entire system has achieved the effect of AR navigation and can be used normally in outdoor environments.

References

[1] A. Mulloni, H. Seichter and D. Schmalstieg. “Handheld augmented reality indoor navigation with activity-based instructions,” Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, Stockholm, Sweden, 2011, pp.211-220.

[2] C. Koch, M. Neges, M. König, M. Abramovici. “Natural markers for augmented reality-based indoor navigation and facility maintenance,” Automation in Construction, vol. 48, pp.18-30, December 2014.

[3] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte and M. Csorba. “A solution to the simultaneous localization and map building (SLAM) problem,” IEEE Transactions on Robotics and Automation, vol. 17, no. 3, pp.229-241, June 2001.

[4] J. Wither, S. DiVerdi and T. Höllerer. “Annotation in outdoor augmented reality,” Computers & Graphics, vol. 33, no. 6, pp.679-689, December 2009.

[5] KM. Yu, JC. Chiu, MG. Lee and SS. Chi. “A mobile application for an ecological campus navigation system using augmented reality,” 2015 8th International Conference on Ubi-Media Computing (UMEDIA), Colombo, Sri Lanka, 2015, pp.17-22.

[6] M. Hazas, J. Scott and J. Krumm. “Location-aware computing comes of age,” Computer, vol. 37, no. 2, pp.95-97, February 2004.

[7] G. Nannicini, D. Delling, L. Liberti and D. Schultes. “Bidirectional A $*$ Search for Time-Dependent Fast Paths,” Experimental Algorithms, 2008, pp.334-346.

[8] E. Chin, AP. Felt, K. Greenwood and D. Wagner. “Analyzing inter-application communication in Android,” Proceedings of the 9th international conference on Mobile systems, applications, and services, Bethesda, Maryland, USA, 2011, pp.239-252.

[9] WA. Mattingly, D. Chang, R. Paris, N. Smith, J. Blevins and M. Ouyang. “Robot design using Unity for computer games and robotic simulations,” 2012 17th International Conference on Computer Games (CGAMES), Louisville, KY, USA, 2012, pp.56-59.

[10] G. Bieber, J. Voskamp and B. Urban. “Activity Recognition for Everyday Life on Mobile Phones,” International Conference on Universal Access in Human-Computer Interaction, 2009, pp.289-296.