Using GPS Data to Control an Agent in a Realistic 3D Environment

Matti Pouke
University of Oulu, Center for Internet Excellence
Erkki Koiso-Kanttilan katu
P.O. Box 1001
FI-90014 University of Oulu
Email: matti.pouke@cie.fi

Abstract—In this paper we focus on controlling an agent in a virtual environment using only GPS information collected with off-the-shelf devices. GPS data in itself is too noisy to be used as waypoint data, as even minor errors in waypoints may cause agents to wander into illegal locations or get stuck between obstacles. To improve the inaccurate waypoint coordinates, we created a method based on map matching and A* pathfinding utilizing a navigation mesh. To demonstrate the competence of our method we compare it with two other methods, which are map matching without pathfinding and a classifier based GPS averaging without a navigation mesh. We found that using a navigation mesh with map matching and A* based pathfinding, safe routes can always be guaranteed for the agent even though some unnaturalness is sometimes introduced in the paths.

Keywords—Path planning, intelligent agents, virtual reality

I. INTRODUCTION

3D environments portraying real-world locations are becoming increasingly popular. Modern map services such as Google Maps and Nokia Maps already contain representative 3D models of major cities. Realistic 3D models of real-world locations allow new type of information visualisation possibilities such as acquiring real-world sensor data with geographical information and representing information provided by the data in its corresponding virtual environment location. This research belongs to a project attempting to display real-world phenomenon and human activity in a virtual environment. In this paper we focus on controlling an agent in a virtual environment using only GPS information collected with off-the-shelf devices. GPS data has been used with many augmented and virtual reality applications such as [1] and [2]. However according to our current literary review there are very few works in which an agent is controlled through GPS data. The work of Rakkolainen et al. [3] is probably one of the first articles to describe using GPS data for character control in a virtual environment; they suffer from the inaccuracy of the early GPS devices but compensate the inaccuracy using radio differential correction. Mantoro et al. [4] are using GPS data for a navigation system which simultaneously shows a representation of the user in a 3D virtual environment. However, they do not describe how they cope with an inaccurate GPS signal.

Some modern videogames use navigation meshes to describe navigable and innavigable areas in virtual environments. Navigation meshes can easily be used with pathfinding algorithms such as the A* [5], which can find a proper route through a complex environment [6]. Map matching is a method used especially in traffic analysis and means the process of matching inaccurate GPS or user positions in a road network or a digital map. An example of map matching for vehicles can be seen in the work of Lou et al. [7], whereas map matching for pedestrians is mentioned in the work of Martin et al. [8]. Information from massive amounts of GPS data can also be clustered into singular paths. An example of visual clustering of large amounts of spatio-temporal data can be found in [9]. We investigated map matching, path finding and clustering of GPS data in order to improve the use of noisy GPS data as waypoint information for agents. We found that combining map matching and pathfinding utilizing a navigation mesh can guarantee that a logical route is always found for an agent despite the noisiness of the GPS data. To demonstrate the effectiveness of our approach we compare our proposed method with plain map matching withouth pathfinding and with a clustering based waypoint improvement method which does not utilize a navigation mesh. For this purpose we created an automatic trajectory clustering method using a k-Nearest Neighbor classifier. There are also local avoidance methods for dodging objects and other agents such as [10], but they are beyond the scope of this work as in this paper we focus on a single agent moving in a static environment.

II. SYSTEM OVERVIEW

Our system consists of the following components: virtual environment, GPS device, navigation mesh, and teaching set. The GPS device is used to gather the positional data to guide an agent in the virtual environment. The navigation mesh is used to represent the navigational information for the agent, allowing map matching and pathfinding through a complex environment. The teaching set is only used with path averaging in our comparative analysis.

Fig. 1. Map of Oulu and the 3D scene.

1©OpenStreetMap contributors http://www.openstreetmap.org/copyright
A. Virtual Environment

We use the RealXtend Tundra [11] as our virtual environment platform. The RealXtend Tundra is an entity-component based virtual environment platform. It uses the Ogre Open Source Graphics Engine as its renderer. The city of Oulu 3D scene is based on 9 blocks of the actual Oulu downtown. It is oriented along compass points, as shown in Figure 1, uses the real Oulu map as its basis and is scaled so that one scene coordinate unit corresponds to one meter in the real world.

B. Navigation Mesh

The backbone of our system is a navigation mesh created from the Oulu 3D model as seen in Figure 2. The navigation mesh is constructed semi-automatically using the open source Recast Navigation Application \(^2\). As we did not want to integrate the entire Recast Navigation system to RealXtend, we used a custom build \(^3\) which allows the exporting of a navigation mesh as an obj file. After the .obj export, we adjusted the navigation mesh by deleting all isolated meshes using the Blender 3D modelling application. The node sizes within the mesh depend on their geographical location. Open squares are represented with larger nodes whereas smaller areas such as paths and roads have smaller nodes. The largest nodes can have an area over 400 \(m^2\), however typically the node areas lie between 10 \(m^2\) and 40 \(m^2\). We then used the navigation mesh with our own map matching and navigation code.

C. GPS Use

We gathered the GPS data using a Garmin eTrex 10 device and mobile phones with Endomondo and Sports Tracker applications. With the eTrex device and the Endomondo application, the sampling rate of the GPS was around a point per 10-60 seconds. When using Sports Tracker, the sampling rate was one point per second which is plenty for walking speed. We used the lower sampling rate Endomondo and eTrex devices for preliminary experimentation and gathering the teaching set needed for path averaging method. In our primary tests (later described) we used a cell phone with the Sports Tracker application.

Because of the proper orientation and scale of the Oulu 3D model, the conversion from GPS data into the 3D model coordinate system is easy. We acquire the displacement from the scene origin by subtracting the measured GPS latitude and longitude points from the latitude and longitude of the scene origin as seen in equations 1 and 2. We then convert the displacements to meters (serving as coordinates \(x\) and \(z\) in the 3D model). As the city of Oulu 3D model is very small, we did not use great circle distance but the simpler equations 3 and 4 in the conversion. By comparing our conversion with the GPS map visualisations provided by the Endomondo and Sports Tracker applications, we concluded that our conversion was accurate enough for our purposes. Where visible differences could be seen between the navigation mesh and the map visualisations, they were within 1 or 2 meters range which is acceptable taken the noise in the original GPS signal is much larger.

\[
\Delta \text{lat} = \text{lat}_c - \text{lat} \quad (1)
\]
\[
\Delta \text{lon} = \text{lon}_c - \text{lon} \quad (2)
\]
\[
x = 1000(111.28\Delta \text{lat}) \quad (3)
\]
\[
z = -1000(111.28\Delta \text{lon} \cos \text{lat}) \quad (4)
\]

III. MATERIALS AND METHODS

Even though the GPS coordinates received from the device can be transformed accurately into the virtual environment, the GPS data itself is not accurate enough to be used as waypoints. Depending on the conditions, the coordinates given by GPS devices can be off by several or even tens of meters which would easily direct the agents towards obstacles and off the map. Using GPS is especially problematic in urban environments as the GPS signal is easily blocked by high buildings. In this section we describe these three methods which we refer as: path averaging, map matching and path generation.

A. Path Averaging

The path averaging attempts to smooth the waypoint trail using previously gathered GPS data for noise cancellation.

---

2http://code.google.com/p/recastnavigation/

3http://et1337.wordpress.com/2010/05/08/meshes-of-navigation/
It uses the $K$ Nearest Neighbor algorithm with GPS data transformed into waypoints as its teaching set. The GPS data for the teaching set was generated partly by ourselves for this experiment and partly acquired from a dataset of related research. We settled on to the $K$ value of 12 based on the amount of teaching data and experimentation. When a new trail is smoothed, for its every waypoint, 12 closest neighbor points are sought from the teaching set and their centroid is used as the new waypoint.

B. Map Matching

The basic idea behind map matching is to “snap” inaccurate GPS points into closest known locations [7]. In our method, we use nodes of the navigation mesh to represent the known walkable locations. Our method differs from the work of Martin et al. [8] in the way that we use map matching to snap GPS points inside of the nodes instead of vice versa. For each waypoint in the trail, a point-in-triangle test is made if the waypoint falls inside one of the nodes within the navigation mesh. If the point is inside a node, we use it as is. If the point is not inside of a node and thus falls outside of the navigation mesh, we check its 6 closest neighboring nodes. From these neighbors we choose the one which is the closest to the previous waypoint. By favoring matched waypoints close to each other, we can create smoother paths than always assigning the inaccurate waypoint to its closest node. From Figure 4 A we can see an example where waypoints next to a river are first matched to their closest nodes and then by favoring nodes close to previous waypoints in Figure 4 B. Using closest nodes to the inaccurate waypoints leads into waypoints assigned to both sides of the river and thus, an illegal path.

C. Path Generation

Our proposed method is the path generation, in which we first preprocess the waypoints with map matching. However, we don’t use the resulting waypoint trail for controlling the agent but downsample it and use the resulting waypoints as start and destination points for the A* pathfinding algorithm [5]. The A* algorithm can search the shortest path between two points through the navigation mesh nodes and generate new waypoints as it goes. The A* algorithm initially calculates the waypoint trail through each nodes midpoints, resulting in an unrealistically winding trail. We straighten the resulting trails with a funneling algorithm which calculates straight paths between consecutive nodes and lessens the amount of waypoints, thus producing a straighter, more realistic trail. In addition to navigation mesh based pathfinding, the funneling algorithm is also well described in [6].

D. Experiment

To validate our results we performed three trial walks around the Otto Karhi park of Oulu downtown (Figure 3). A research assistant was carrying a cell phone with a Sports Tracker application recording GPS points with the sampling rate of 1 point per second. We videotaped the walk from a nearby rooftop while the assistant also took mental notes of the exact trajectory of his path. The first route (colored green in Figure 3) was 620 meters using the sidewalks going around the park. The second and the third route (colored red in Figure 3) was 560 meters long using the path through the park on the way back. The park is one of the more complex areas within the Oulu 3D model so it is suitable as a testing ground. We then transformed the acquired GPS points into three waypoint trails and applied the three waypoint improvement methods into each one of them. The accuracy of the recorded GPS points vary greatly between the routes. We use three metrics to inspect the effectiveness of each method. Firstly, we calculate the distance between each waypoint trail and the route reported by the research assistant at eight locations. We average these distances for each trail to be used as one of our error metrics for the waypoint improvement methods (Figure 5). We also visually inspect if the waypoint trails contain illogical choices and critical errors. By illogical choices we mean seemingly absurd winding and sudden u-turns in the path. By critical errors we mean trails that would lead the agent inside of a physical obstacle, such as a building, or into the river running in the middle of the park. We consider the critical errors to be our most important error metric.

![Fig. 4. Figure A: Waypoints matched without favoring earlier points. Figure B: Waypoints matched with favoring earlier points (red dots are raw waypoints).](image1)

![Fig. 5. Average error in meters in comparison to the reported routes](image2)

IV. RESULTS

Figure 5 shows the average error of each of the improvement methods as well as the raw waypoint data in comparison to the routes reported by the research assistant (the reported routes are shown in Figure 3). The results of each methods waypoint improvement can be seen in Figures 6, 7 and 8. In the first route (Figure 6), the GPS points are the most accurate while still deviating greatly from the actual path of the assistant. In the second route (Figure 7), the error is slightly larger while the third one (Figure 8) barely hits any roads or paths in its course. The best way to see how each method improves the waypoints is to look at Figures 9, 10 and 11. The unchanged waypoints are shown in red whereas the improved paths are shown in green. To save space we show images of each methods waypoint improvement only for the most inaccurate Route 3.
1) Path Averaging: We can see the results of path averaging in Figure 9. While path averaging evenly shortens the distance between the waypoints and the actual route, the waypoints are moved only a little and there is a large amount of critical errors. Of the described methods, path averaging seems to produce the smoothest paths in terms of illogical choices but the critical errors render the waypoint trails unusable.

2) Map Matching: Using map matching the inaccurate waypoints are matched within the navigation mesh nodes attempting to keep the waypoints close to each other. We found that map matching occasionally increases the distance error at some points of the path but still decreases the error on the average. The map matching does take all the waypoints off critical locations, but there still exists numerous instances where following a straight line between the waypoints would lead to a critical error. Especially in the northwestern part of the trail, map matching produces illogical winding to both sides of the river which are also critical errors.

3) Path Generation: We downsampled the experiment data by taking every 10th map matched waypoint and generating the routes in between. Waypoint trails improved with path generation have the smallest average distance error between them and the reported routes. Occasionally the waypoints can however be even further away than when using raw GPS waypoints. One example can be seen in the southern part of Route 3 (Figure 11) when path averaging cuts to the left side of the river when the actual route was at the right side. Path generation however, is the only method where there exists no critical errors. Path generation seems to produce larger amounts of illogical route choices than the other methods, such as the u-turns at the western part of the park and at the bridge. Large errors in the original waypoint trail and erroneous map matchings can cause the illogicalities, but the A* pathfinding guarantees that a sane route is always found if the navigation mesh is unbroken so there are no critical errors in the paths.

A. Comparison between video and virtual reality
We videotaped the three trial runs reported in the previous section. We then compared an agent using waypoints improved with path generation with the video data. Even with occasional dissimilarities and sometimes different choices of path, it can be seen that the agent roughly follows the same trail as the research assistant. Minor details are lost, such as choice of sidewalks. For example, in Figure 12 we can see the research assistant walking at the left side of the road and using proper sidewalks. In Figure 13 we see the virtual character following somewhat similar trail but in the middle of the road.

V. Discussion
Of the methods described in this paper, it is clear that path generation is the only usable method to control an agent with GPS data. With A* pathfinding it can be guaranteed that the agent does not wander into illegal areas (at least if
there are no other forces affecting the agent). The downside is that calculating the paths with pathfinding loses more of the original GPS information than using the other methods. As path generation relies entirely on the underlying navigation mesh, small details are lost even when the GPS reception is excellent and the waypoint data is accurate. Using more points from the preprocessed data and thus calculating shorter intermediate routes compensates this to some extent. In real-time use, the application developer would choose the time interval for waypoint polling. When a new waypoint is received from the user, the application would calculate the route from the previous location.

Theoretically, the path averaging allows the finding of possible real world routes which are not taken into account when generating the navigation mesh. However, in our tests the noise of the existing trails was too large and non-uniformly distributed to create accurate routes as the result of averaging. It would be interesting to see if the results would be different from extremely large existing datasets and large k values. Of course the teaching set for the path averaging could also be preprocessed by ignoring points which fall outside of the navigation mesh. However the generated waypoint trails would still contain same problems as with using only map matching (a straight path might not exist between two waypoints). Also, in our comparison we wanted to include a method which is not dependent of the navigation mesh. The path averaging is also quite slow (taking several seconds to calculate with brute-force search) whereas the A* based pathfinding is almost instantaneous and thus more suitable for possible real-time use.

A drawback of this work is that despite the agent staying on walkable areas with path generation, the overall result can still be inaccurate with poor GPS reception. One solution could be the combination of cell phone camera for additional feature tracking such as used in Augmented Reality applications [12].

VI. CONCLUSION

The contribution of this paper is to present a foolproof method to control an agent in a 3D virtual environment using noisy waypoint data as the source for agent control. The method can be used for example in videogames using GPS as an avatar controller or in information visualisation. While we did not test our system real-time, the method we propose is fast enough to use in real-time applications. Future work should focus on improving the path generation accuracy further by tracking location from other sources in addition to the GPS data. The future system should be evaluated in real-time at various locations instead of the single park used in the current study. The performance of the system with dynamic and adaptive sampling rates of GPS devices should also be tested.

ACKNOWLEDGMENT

This work has been carried out in the TEKES Chiru project.

REFERENCES


