2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE) 978-1-7281-3066-8/20/$31.00 ©2020 IEEE 10.1109/ICISCAE48440.2019.221579

AR Navigation Application based on iOS Platform

Xin Wang
College of Computer Science and Technology Jilin University, 130012, Changchun, China;
Key Laboratory of Symbolic Computation and Knowledge Engineer of Ministry of Education, Jilin University, 130012, Changchun, China
w_x@jlu.edu.cn

Risong Zhang
College of Computer Science and Technology Jilin University, 130012, Changchun, China
zhangrs18@mails.jlu.edu.cn

***Abstract*—In this paper, an Augmented Reality Navigation Application based on iOS platform is designed. This application is composed of five modules: the Location Module, the Digital Map Module, the Approximate Module, the Estimate Module and the AR Module. The Location Module requests current location via GPS in real-time and obtains the details of destination from map server. The Digital Map Module provides a Satellite Imagery View which superimposes navigation routes with marks such as current location, starting and destination. The Approximate module produced an approximation to fix the bias when converting the geographical coordinates between WGS-84 and GCJ-02 system. The Estimate Module produced a 3D position estimation according to a geographical coordinate provided by GPS or map server. This estimation will be used for placing the marks and routes in AR view. The AR Module provides an Augmented Reality scene superimposed destination and the route on a real-world background with a real-world scale.**

Keywords—Augmented Reality, Navigation Application, GPS, Geographic Coordinate Processing, Digital Map.

I. Introduction

The navigation application provides user a convenient and efficient travel experience. Furthermore, the release of walking navigation applications gives a brand new navigation experience which is quite different from a vehicle or a vessel navigation.

The Augmented Reality(AR) is an interactive experience of real-world environment, which is augmented by virtual objects. An AR environment is created by overlaying the computer-generated objects on a real-world background. This environment provides an interactive visual, auditory or somatosensory experience.

AR is used in variety fields, such as video games, education, medical and navigation. For example, the AR mobile phone game *Pokémon GO*[1] builds a Pokémon world based on the digital map information, players look for *Pokémons* by walking through the street and catch them by a series of simple interactions. Dilek et al. developed a novel application which can be used for detecting position in physics experiments with *ARKit*[2]. AccuVein developed the *AV400 Vein Viewing System*, which supports a blood vessel displaying for people of different skin and ages[3]. HG Debarba et al. present a visualization tool for human motion analysis in augmented reality[4], J Albouys-Perrois et al. designed an Augmented Reality Map for Blind and Low Vision People[5].

Nowadays, the AR SDKs based on smartphone platform such as *fAR-Play*[6], *Vuforia Augmented Reality SDK*, *ARCore* and *ARKit* has matured. Combining the AR with the navigation system, the design and developing of an *AR Navigation Application* becomes feasible.

The AR Navigation Application designed in this paper provides an AR experience for walking navigation — overlaying the destination and the route on a real-world background with a real-world scale.

II. General Design

As shown in Fig. 1 and Fig. 2, the navigation application consists of two parts, which are the Navigation Part and the AR Part. The Navigation Part is based on *MapKit* and *Core Location* framework of Apple App Frameworks. The AR Part is based on *SceneKit* and *ARKit* [7] framework.

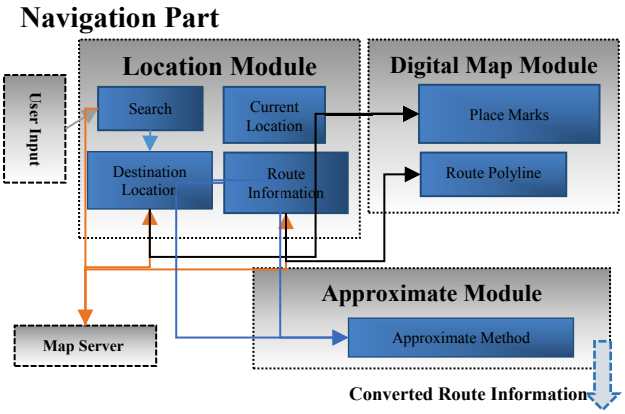

Fig. 1. Components of Navigation Part

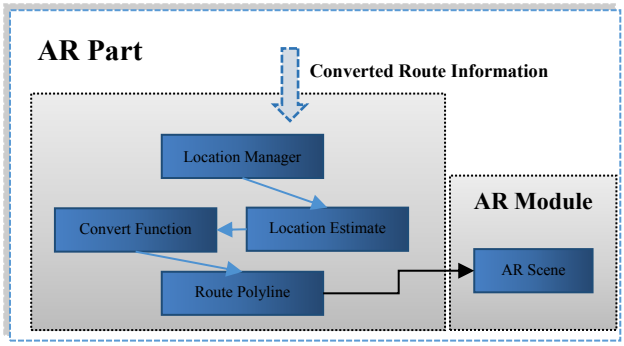

Fig. 2. Components of AR Part

The application was developed with Swift 4.1 and Xcode9.

Due to the limitation of ARKit, this application requires an iOS device with an A9 or later processor.

The Navigation Part can be divided into Location Module, Digital Map Module and Approximate Module. Furthermore, the AR part can be divided into Estimate Module and AR Module.

In Navigation Part, the *Amap* server is used as our map server. The system obtains user's current location via GPS, request the location's details from map server. The system gets the destination with the toponym user input and generate the navigation route from current location to destination.

GCJ-02 (*Mars Coordinates*) is a geodetic datum based on *WGS-84* and formulated by the Chinese State Bureau of Surveying and Mapping. With the alleged goal of improving national security, it uses an obfuscation algorithm, which adds apparently random offsets to both the latitude and longitude.

Moreover, *WGS-84* [8] is the last revision of *World Geodetic System*(*WGS*). The *WGS* is a standard for use in cartography, geodesy and satellite navigation including GPS.

It is remarkable that *Amap* uses *GCJ-02* system to indicate the geographical coordinate. In contrast, the GPS uses *WSG-84* system. There is a non-ignorable bias when using a *GCJ-02* coordinate directly in *WSG-84* system, but this bias can be fixed in Approximate Module by the method proposed in open source GitHub project *wandergis /coordTransform*.

In addition, the Navigation Part provides a *Satellite Imagery View* which is superimposed the start point mark, the *Destination Mark* and the polyline of the navigation route.

The AR Part can be divided into Estimate Module and AR Module. It converts the key points' geographical coordinate of routes into 3D position in Estimate Module using the method proposed in open source GitHub project *ProjectDent/ARKit-CoreLocation* based on *Haversine Formula*.

Moreover, AR Module shows the route transformed in AR scene with animated blue arrows and cylinders. It also provides a *Destination Mark* in AR scene with the distance in meters.

The description of each module will be expanded in detail in next session.

III. Module Design

A. Location Module

The Location Module is based on *Core Location* Framework and *MapKit* Framework [9]. To achieve this module, the *Core Location* Framework was configured with the configuration shown in Table I.

TABLE I. The configurations of Core Location Framework

Configuration	Value
delegate	*self*
desierdAccuracy	*kCLLocationAccuracyBest*
distanceFilter	10
requestWhenInUseAuthorization()	-

Due to the accurate requirement of location by AR, we set the parameter desiredAccuracy to *kCLLocationAccuracyBest*, even though it will lead a huge loss of electricity. For economy, we only put the Location Authorization in use and update the location every ten meters.

This module also processes the information user inputted. When perform the *Search* action, the Location Module transform current location to an MKPlacemark object as start point. Furthermore, the module requests map server with an MKLocationsearch object to get a list of MKPlacemark object for results. Each item of the list will be shown on a table view. After user chooses an item, the corresponding place mark will be used as destination.

Then obtain a route between the start point and the destination from map server with an MKDirectionRequest object. The route will be used by Estimate Module to generate the polyline which is used in *Approximate Module*.

B. Digital Map Module

The Digital Map Module is based on *MapKit* Framework. It shows a *Satellite Imagery View* superimposed with marks and route. To achieve this module, the *MapKit* Framework was configured with the configurations shown in Table II.

TABLE II. The configurations of MapKit Framework

Configuration	Value
delegate	*self*
userTrackingMode	*FollowWithHeading*
mapType	*.hybird*
showsUserLocation	*true*

The Digital Map Module provides a *Satellite Imagery View* as an overall view of the navigation route to provide a more intuitive navigation experience.

To make the *Satellite Imagery View* rotate with direction user forward, set the parameter userTrackingMode to *FollowWithHeading*, set the showsUserLocation to *true*. For an intuitive experience, set the maptype to *.hybrid* which overlays the streets and other interest points on the *Satellite Imagery View*.

When application run, the current location and heading mark will be shown on the center of *Satellite Imagery View*. When the details of navigation route is responded by map server, the module converts the route into a MKMapPolyline object and superimpose it on the *Satellite Imagery View*.

C. Approximate Module

When the *Location Module* and the *Digital Map Module* requests the user's current location and details of place marks, there is a bias due to the differences between *WGS-84* and *GCJ-02* system occurred. In addition, this bias could be up to hundreds of meters.

The Estimate Module is based on *WGS-84* system. In contrast, the route generated by Location Module is based on GCJ-02 system. For transforming geographical coordinates into 3D position correctly, we need to use Approximate Module to provide a method which makes the coordinate and position can be convert to each other.

To fix this bias, we used an Approximate Method proposed in the open source GitHub project *wandergis/coordTransform*. The method provides an experiential approximation when convert the coordinates

between *GCJ-02* and *WGS-84*, and the workflow is shown in Fig. 3.

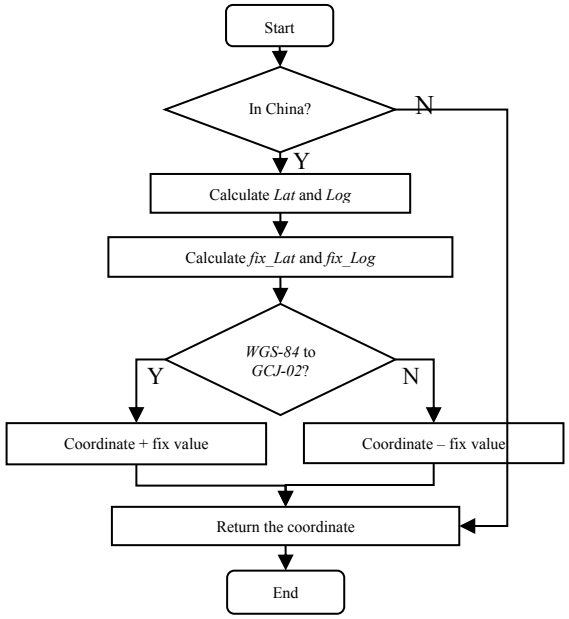

Fig. 3. Approximate Method workflow

In this method, the coordinate defined using (*Latitude*, *Longitude*) and the latitude correction value is defined by Equation (1).

$$Lat = -100 + 2x + 3y + 0.2y^2 + 0.1xy + 0.2\sqrt{|x|} + \frac{2}{3}\left\{[20\sin 6\pi x + 20\sin 2\pi x] + \left[20\sin y\pi + 40\sin\left(\frac{\pi y}{3}\right)\right] + \left[160\sin\left(\frac{\pi y}{12}\right) + 300\sin\left(\frac{\pi y}{30}\right)\right]\right\} \quad (1)$$

The longitude correction value is defined by Equation (2).

$$Log = 300 + x + 2y + 0.1(x^2 + xy) + 0.1\sqrt{|x|} + \frac{2}{3}\left\{[20\sin 6\pi x + 20\sin 2\pi x] + \left[20\sin x\pi + 40\sin\left(\frac{\pi x}{3}\right)\right] + \left[150\sin\left(\frac{\pi x}{12}\right) + 300\sin\left(\frac{\pi x}{30}\right)\right]\right\} \quad (2)$$

Where x and y are calculated according to the (*Latitude*, *Longitude*), the x and y can be obtained by Equation (3).

$$\begin{cases} x = Longitude - 105.0 \\ y = Latitude - 35.0 \end{cases} \quad (3)$$

Then we can obtain the fix value (fix_Lat, fix_Log) by Equation (4) and (5).

$$fix\_Lat = \frac{180Lat}{\frac{a(1-ee)\pi}{\sqrt[2]{magic^3}}} \quad (4)$$

$$fix\_Log = \frac{180Log}{\frac{a\pi}{\sqrt{magic}\cos(radLatitude)}} \quad (5)$$

Where a refers to the length of the earth's long axis, and ee is the square of the eccentricity of the earth's ellipsoid. In addition, $magic$ and $radLatitude$ are the intermediate parameters.

$$\begin{cases} radLatitude = \frac{Lat\pi}{180} \\ magic = 1 - ee\sin^2 radLatitude \\ a = 6378245.0 \\ ee = 0.00669342162296594323 \end{cases} \quad (6)$$

Finally, use the (fix_Lat, fix_Log) to fix the (*Latitude*, *Longitude*). If we want to convert *GCJ-02* to *WGS-84*, we have to subtract the fix value from coordinate. To convert *WGS-84* to *GCJ-02*, add the fix value, shown in Equation (7).

$$(postLat,\ postLog) = \begin{cases} (latitude + fix\_Lat, longitude + fix\_Log), WGS84\ to\ GCJ02 \\ (latitude - fix\_Lat, longitude - fix\_Log), GCJ02\ to\ WGS84 \end{cases} \quad (7)$$

With the method above, the *WGS-84* coordinates used in *Core Location* and the *GCJ-02* coordinates used in *MapKit* can be convert to each other.

D. Estimate Module

The Estimate Module implements a method to convert the geographical coordinate fixed by Approximate Module into 3D position.

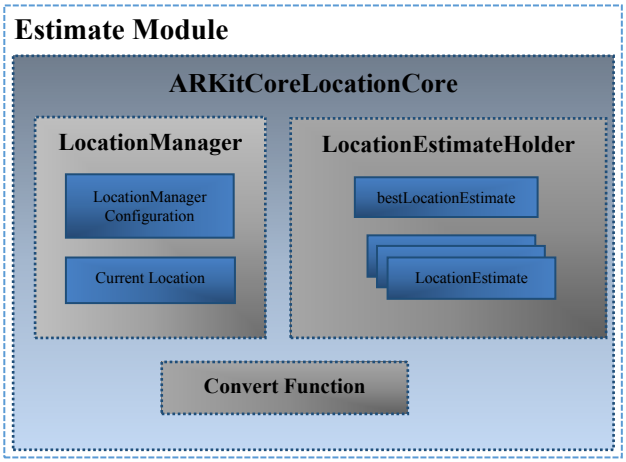

Fig. 4. The components of Estimate Module

The components of Estimate Module are shown in Fig. 4. An ARKitCoreLocationCore object is used to convert the geographical coordinates to 3D positions (where the Y is 0). Moreover, the ARKitCoreLocationCore have two indispensable members, which are locationManager and locationEstimatesHolder.

1) Location Manager and Location Estimate:

The LocationManager object keeps the current location and positioning configuration. The configuration is provided by a LocationManagerConfiguration object, which contains information such as positioning accuracy and relocation distance.

A LocationManager object has a member called LocationManagerListener, which is used to monitor and respond the change of authorization mode and location coordinate.

The locationEstimateHolder object holds the speculation results for multiple locations and selects the best one by constantly updating and comparing each location. In addition, it has two subclasses—BasicEstiHolder and

AdvancedEstiHolder. (10)

BasicEstiHolder compares the current best estimation with the estimation newly added, and AdvancedEstiHolder determines whether the new estimation can be replaced by previous estimation at first.

The rule to comparing two Location Estimations is presented.

Select the estimate with the higher horizontal accuracy. If they have the same horizontal accuracy, the newer estimate will be selected.

2) Convert Function

The Estimate Module uses a *Convert Function* to convert geographical coordinates into 3D positions. Using this Function, 2D coordinates will be convert, and return a 3D vector of type SCNVector3. Fortunately, the unit of the coordinate used by *ARKit* Framework is meter. In other words, one unit in *ARKit's* coordinate means one meter.

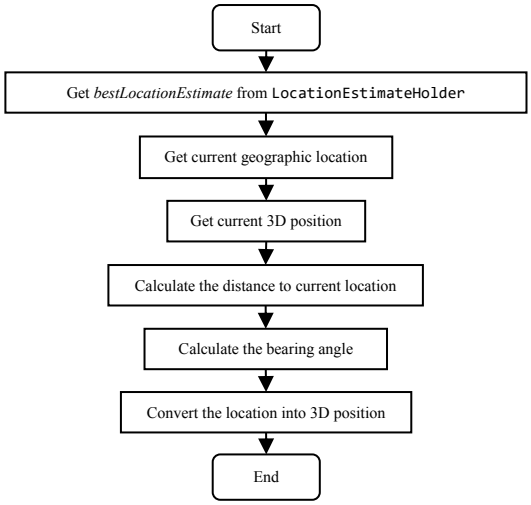

Fig. 5. Convert Function workflow

The *Convert Function* is based on the *Haversine Formula* [10] and the workflow shown in Fig. 5. The *Haversine Formula* uses fewer trigonometric cosine functions than the Great-circle distance formula, and reduces the error due to processing longitude problems and guarantees sufficient effective numbers for dealing with the two close points.

The *Haversine formula* is defined by Equation (8) and (9).

$$\text{Haversine}\left(\frac{d}{\mathrm{R}}\right) = \text{haversin}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\,\text{haversin}(\lambda_2 - \lambda_1) \tag{8}$$

$$\text{haversin}(\theta) = \sin^2\frac{\theta}{2} = \frac{(1-\cos\theta)}{2} \tag{9}$$

Where R is the radius of the earth, generally taking an average of 6371km. Where φ_1, φ_2 represents the latitude in radians of the first point and the second point, and λ_1, λ_2 represents the longitude in radians of the first point and the second point.

It is necessary to make sure the angle between to coordinates. The bearing angle defined by Equation (10).

$$\text{Bearing} = \arctan\left(\frac{\sin\Delta\lambda\cos\varphi_2}{\cos\varphi_1\sin\varphi_2 - \sin\varphi_1\cos\varphi_2\cos(\Delta\lambda)}\right)$$

With the *Convert Function*, we can convert a geographical coordinate into a 3D position. The navigation route provided in Location Module and Approximate Module converts to a 3D polyline with Estimate Module, and the polyline will be used by AR Module to show the navigation route in an AR scene.

E. AR Module

The AR Module provides an AR scene to show a navigation route with a real-world scale and a *Destination Mark* to show the distance from user's current location. The AR Module provides a more intuitive and vivid navigation experience. Fig. 6 shows the components of AR Module.

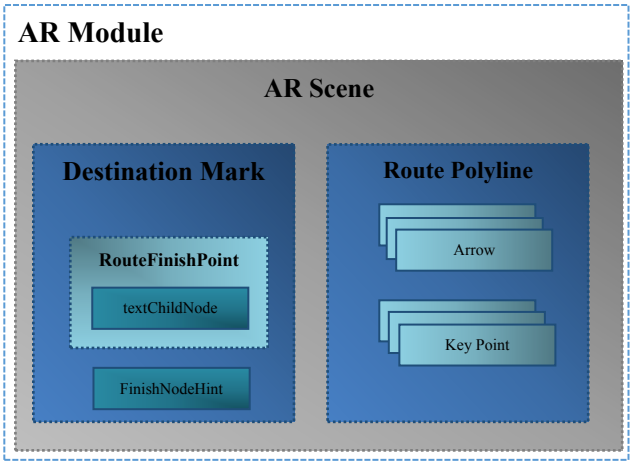

Fig. 6. The components of AR Module

1) Destination Mark

The *Destination Mark* is used to show the distance between current location and destination in AR Scene. This mark implemented by a RouteFinishNode object, which is a subclass of SCNNode. A RouteFinishNode object contains a member named textChildNode to display the distance, and a pulseAnimationNode member to display the mark.

When the destination is out of view, the system uses a FinishNodeHint object named *Hint Mark* to prompt the destination. When the destination returns to the field of view, the *Destination Mark* will be displayed normally.

It is remarkable that the *Hint Mark* is not an NSNode object, but a UI view, which is a way to display the destination's position through the UI.

As shown in Fig. 7, this UI interface is hidden by default. The `FinishNode` object determines whether the coordinates of the destination are in the camera view at first. If it is in the view, the UI is hidden (left). If it is not, the *Hint Mark* will be shown on the edge of the camera view (right).

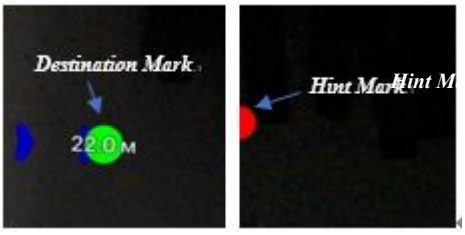

Fig. 7. *Destination Mark* and *Hint Mark*

2) Route Polyline

The route polyline consists of arrows and key points, where the arrows indicate the direction which the user should follow.

A `RoutePointNode` object implements the key points of the route. After Location Module generates the route, the key points of route will be stored in an array named `routePointNodes`. Then the Estimate Module will convert the key points into 3D positions.

Each of these arrows is defined by a `RouteGeometryFactory` object. Then use *createPolyline method* to show them in AR scene.

The *createPolyline method* splits the path and generates animations for each segment of the path (including direction, start point and distance). Once get the animation, use this animation to animate arrows.

With the arrows and the *Destination Mark*, the AR Module provides an AR based navigation experience. User can reach the destination by following the arrows simply.

IV. RESULTS

This section describes the results of modules when the application is in use. In this section, we use an iPad6 to test the Approximate Module and Estimate Module individually. Then test the overall application process including AR scene and UI.

We take the surrounding area of South Qianwei Campus of Jilin University as the test range. The range[1] is shown in Fig. 8.

Fig. 8. The range to test the application

A. Approximation

Approximate Module provided the Location Approximates with *WGS-84* system. To contract, we choose five coordinates and show them with *WGS-84* and *GCJ-02* system, then regard the *GCJ-02* coordinate as a *WGS-84* coordinate directly, and calculate the distance between the coordinates.

As shown in Table III, the average bias in meter is 584m. The biases are non-ignorable and nonlinear. We cannot regard a *GCJ-02* coordinate as a WGS-84 coordinate directly.

TABLE III. THE BIAS WITHOUT APPROXIMATE

WGS-84	GCJ-02	Bias in meter
(43.822983, 125.279227)	(43.825363, 125.285640)	579
(43.819479, 125.262431)	(43.821849, 125.268817)	576
(43.835890, 125.390787)	(43.838427, 125.397434)	603
(43.959022, 125.341596)	(43.961422, 125.348167)	590
(43.817547, 125.269656)	(43.819919, 125.275979)	572

As shown in Table IV, with the approximate method we used in Approximate Module, the bias can be fixed into several meters. The average of the biases fixed is 2m. It is negligible when using a navigation system.

TABLE IV. THE BIAS WITH APPROXIMATE

WGS-84	Approximation	Bias in meter
(43.822983, 125.279227)	(43.822978, 125.279224)	1
(43.819479, 125.262431)	(43.819480, 125.262428)	0
(43.835890, 125.390787)	(43.835886, 125.390782)	1
(43.959022, 125.341596)	(43.959007, 125.341572)	3
(43.817547, 125.269656)	(43.817543, 125.269580)	5

However, the biases still exist, and non-ignored to AR navigation. A meter level bias still can be vital when we use AR to a navigation application. Because the AR requires an accurate location or position to superimpose a virtual object onto a real-world background.

B. Convert method

The method we used in the Estimate Module provides a 3D position estimate of the key points of the navigation route. With this method, the polyline shown in AR scene can be a correct position with a real-world scale.

We choose five locations randomly and convert into 3D positions with the Convert Method. The distributions of location and 3D position are shown in Fig. 9.

TABLE V. RESULT OF THE CONVERT MODULE

Location Coordinate	3D Position
(43.818851, 125.276214)	(525.79, 0, -145.53)
(43.821947, 125.273575)	(314.05, 0, -489.53)
(43.820716, 125.270818)	(92.85, 0, -352.65)
(43.820306, 125.268886)	(-62.16, 0, -307.06)
(43.819510, 125.262216)	(-597.31, 0, -218.55)

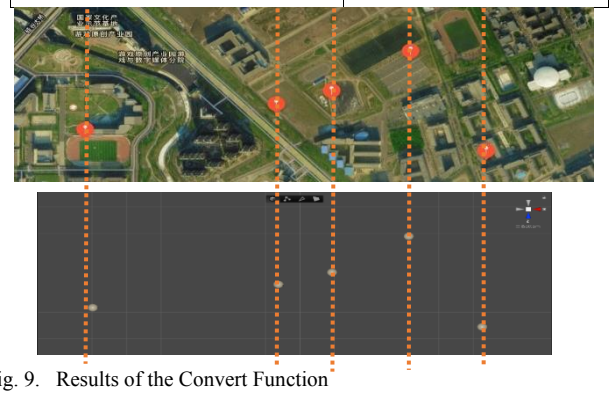

Fig. 9. Results of the Convert Function

The distributions of the locations and the distributions of 3D positions converted by *Convert Function* are roughly the same. In addition, with the advantages of *ARKit*, we can use

[1] Data source: Google Maps

these 3D positions to show a polyline which is the same as the navigation route.

C. Navigation Application

The tests above manifest how the modules convert the locations to 3D positions. With these modules, the application provides an AR navigation experience in an AR scene with a series of simple interactions.

1) UI and Satellite Imagery View

The UI interface of this application is shown in Fig. 10. The search bar above the interface is used to send user's input to Location Module and shows the place marks Location Module found with a table view. In addition, the toponym of user's location will be shown at the right corner of the caption bar.

A *Satellite Imagery View* which is at lower right corner of the interface shows the user's current location, place mark of start point and destination, route, streets and interested points (Fig. 10 right). When the destination has been chosen, the route and the place marks provide an overall view if the route in AR scene is confused.

Fig. 10. UI Interface

2) Augmented Reality

As shown in Fig. 11, the AR Scene superimposes an AR route with blue arrows on a real-world background and the user can follow these arrows to get to the destination.

With combining the route and marks on *Satellite Imagery View* and AR scene, the application can navigate users to the destination correctly with a more intuitionistic, vivid and imaginal experience.

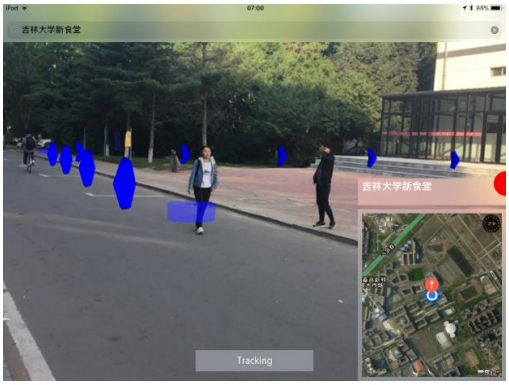

Fig. 11. AR Scene

V. Conclusion

This paper provided a set of design of an augmented reality navigation application based on iOS platform. This application is composed of five modules: the Location Module, the Digital Map Module, the Approximate Module, the Estimate Module and the AR Module. Meanwhile, the application complement a Location Approximate method and a Position Convert method which are used to generate a navigation route in AR scene and overlay this route on a real-world background. The Approximate method can convert the *GCJ-02* coordinate and *WGS-84* coordinate to each other. The Position Convert method can convert a geographic coordinate to a 3D position. With these modules, the application can provide a more intuitionistic, vivid and imaginary navigation experience.

Acknowledgment

Supported by Scientific and Technological Development Program Foundation of Jilin Province, China (No. 201604054YY; No. 20170414006GH)

References

[1] Jess Denham, Pokémon Go praised by gamers for introducing gender fluid avatars [DB/OL]. https://www.independent.co.uk/arts-entertainment/pokemon-go-praised-by-gamers-for-introducing-gender-fluid-avatars-characters-players-lgbt-style-a7132536.html. 2016-7-12.

[2] Dilek, Ufuk|Erol, Mustafa. Detecting Position Using ARKit [J]. Physics Education, 2018, 53(2):025011.

[3] Law, Kyle W., et al. "Use of the AccuVein AV400 during RARP: an infrared augmented reality device to help reduce abdominal wall hematoma." The Canadian journal of urology 25.4 (2018): 9384-9388.

[4] Debarba H G, Oliveira M E D, Lädermann A, et al. Augmented Reality Visualization of Joint Movements for Physical Examination and Rehabilitation[C]. IEEE Virtual Reality. 2018.

[5] Albouys-Perrois J, Laviole J, Briant C, et al. Towards a Multisensory Augmented Reality Map for Blind and Low Vision People: a Participatory Design Approach[C]. CHI Conference. 2018:1-14.

[6] Gutierrez L, Nikolaidis I, Stroulia E, et al. fAR-PLAY: A framework to develop Augmented/Alternate Reality Games[C]. IEEE International Conference on Pervasive Computing and Communications Workshops. 2011:531-536.

[7] Dilek U, Erol M. Detecting position using ARKit II: generating position-time graphs in real-time and further information on limitations of ARKit[J]. Physics Education, 2018, 53(3):035020.

[8] Joe Keeley, iOS MapKit and Core Location LiveLessions (Developer Talks), Downloadable Version [DB/OL]. http://www.informit.com/store/ios-mapkit-and-core-location-livelessons-developer-9780321958570. 2013-8-28.

[9] Wang J, Wang J, Lu C. PROBLEM OF COORDINATE TRANSFORMATION BETWEEN WGS-84 AND BEIJING 54[J]. Crustal Deformation & Earthquake, 2003.

[10] Winarno E, Hadikurniawati W, Rosso R N. Location based service for presence system using haversine method[C]. International Conference on Innovative and Creative Information Technology. 2017:1-4.