Implementation of Augmented Reality to Ease Visualization for Real Estate Firms

Akshay Deore[1] Shivani Dongre[2] Vrushali Pagare[3] Karan Sethi[4]
[1,2,3,4]BE Student
[1,2,3,4]Department of Computer Engineering
[1,2,3,4]AISSMS College of Engineering, Pune, India

Abstract— World over the markets has opened up to immense possibilities that augmented reality devices holds, their applications span across the industries and already there is a lot of buzz happening around the world. And within few years the way we interact with the technology is about to change forever. In this project we are trying to solve the problems faced by the architects due to visualization right from the idea stage to implementation since what an architect thinks, a builder builds and a customer imagines are completely different views of the same project, for this we are using our own Random GPS Position Locking Algorithm using augmented reality to developing an application through which they would be able to see their projects at actual site in real world.

Key words: Augmented Reality (AR), Global Positioning system (GPS), eXtensible Markup Language (XML), Unified Modeling Language (UML)

I. Introduction

The year 1992 is believed to be the birth of the term augmented reality. This term first appeared in the work of Caudell and Mizell [1992] at Boeing, which sought to assist workers in an airplane factory by displaying wire bundle assembly schematics in a see-through HMD. AR became a lot more popular in more recent times, and especially after 2013. Google started working on the Glass and just a few months after Google opened their Glass to developers, car manufacturers Audi and Volkswagen showcased their own AR mobile applications relating to instruction manuals and remote assistance services. And even though the Glass was not a successful project, it was a very important step into the future. Wearable AR technology is surely a part of our future.

In the next decade AR technology got a lot more popular and was used in various fields. Louis Rosenberg developed the Virtual Fixtures on 1992 which became one of the earliest functioning AR systems followed by team of people from Columbia University who built an HMD called KARMA.

II. Objectives

To increase the transparency and understanding of the project, to increase the Productivity and to obtain more efficient result on less given resources.

III. Research Methodology

Previously before AR (Augmented Reality) many methods were used to conquer the problem of visualization such as :- Displaying of a small 3D model of the whole project but through this the customer was only able to visualize the whole project and not the flat he/she was going to buy to overcome this they started to show the 3D model of their flats through 3D softwares through this the users were able to visualize their flats up to some extent but not about the whole project so they started building demo/ sample flats but the same problem emerged due to which they started showing video walkthroughs of the project due to which visualization of the flat and the project in real world became an issue which still prevails in VR walk-through since the VR walk-through takes you completely into a virtual world.

A. Theoretical framework

The overall system design consists of following:

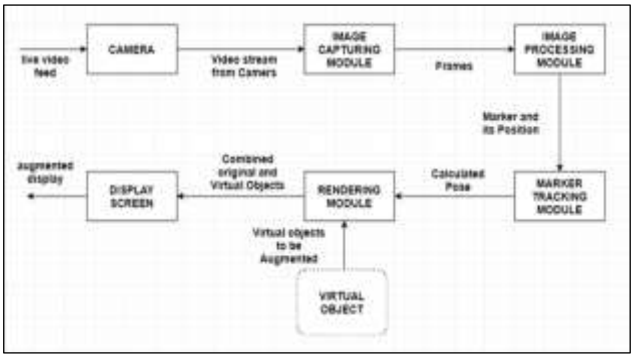
1) Image selection.
2) Scan the QR code/ Lock GPS co-ordinates.
3) Generate AR view using mobile application.

We propose a system to automatically generate AR view that have good structure and content quality than traditional methods. The architecture of our system is shown in Figure 3.1. We are using android application to assign an important view for each co-ordinates in the given QR code or locked GPS co-ordinates, where the model is stored on the cloud. Then, we generate AR view from the given co-ordinates by using QR code or locked GPS co-ordinates. In Figure 3.1 Current method generally extract view from the points to construct the model. In contrast the co-ordinates can be divided into an ordered sequence of parts. Each part addresses a specific co-ordinate and these co-ordinates are also relevant to each other. AR view usually not only have 2D view but also 3D view such as figures and tables. But our work focuses on the 3D AR view only.

In this paper, we propose an Augmented Reality example to generate well-structured AR view for architects. Augmented reality is no longer science fiction. As we move forward, this has now become possible through CAD software, 3D data from software, a smartphone or tablet having a camera and the right amount of computing power. With this technology, while you point your device (which includes a camera) on an object in the real world while computers adds (augment), align and scale the 3D models with your view. The computer models appear in the real world, where they would be if completed. A target, I.e a printed QR code or locked GPS co-ordinates, is used to sync the location of the digital 3D data with the real world. You can walk around the object and look up or down, just as you would if it was there in real life.

B. Abbreviations and Acronyms

QRR - QR Code Reader,
OpenGL - Open Graphics Library,
SRS - Software Requirement Specification,
GUI - Graphical User Interface.
GPGGA - Global Positioning System Fix Data



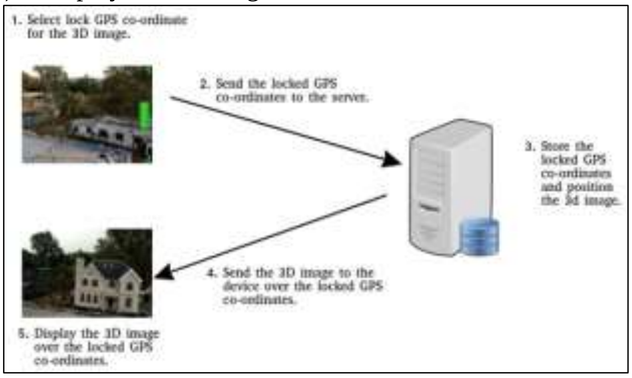
C. Random GPS Position Locking Algorithm

A GPS co-ordinate given by a device is described below by using this we can lock the desired GPS location to a 3D image which could be then displayed into the device:-
$GPRMC,124205.000,A,3002.8030,N,00401.4681,W,0.08, 033.64,030001,,*08$GPGGA,124206.000,3002.8028,N,004 01.4682,W,0,07,0.5,0466.8,M,- 10.6,M,,0000*4F$GPGSA,A,2,11,07,10,05,04,08,13,04,,,,,1 .4,0.3,0.8*2E
The GPGGA sentence that is shown above contains the following:

- Time: 124206.000 is 12:42 and 06.000 seconds in Greenwich mean time
- Longitude: 3002.8030,N is latitude in degrees.decimal minutes, north
- Latitude: 00401.4681,W is longitude in degrees.decimal minutes, west
- Number of satellites seen: 07
- Altitude: 0466 meters

Steps for locking Random GPS location from the application:

1) Click on Lock GPS Location.
2) Send the GPS co-ordinates i.e. altitude, latitude, longitude to the server.
3) Store the locked GPS co-ordinates and position the 3D image.
4) Send the 3D image to the device over the locked GPS co-ordinates.
5) Display the 3D image over the locked GPS co-ordinates.



1) Random GPS Position locking algorithm:

S= {Q; I; P;C; S; PP; FC; F;DD;NDD}
where, S=System
Q= States, Q={q0, q1, q2, q3, q4}
Where,
q0= Initial state
q1= Object selection and Create Tracker
q2= Direct Mode
q3= VR Mode
q4= Display and Gesture Control.(Final state)

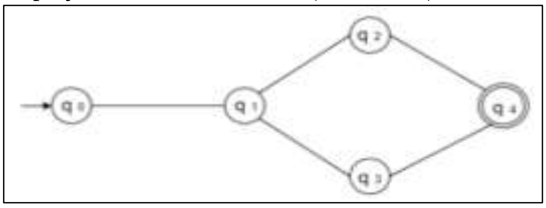


Fig. 3.4.2: Mathematical Model

I=Input:
Image selection
I = Input : f u1, u2, u3, ..., ung; Where, I is a scanning of marker.
u1; u2; u3, ..., un are the number of markers.
Process Augmented Object
P = Process :fp1, p2, p3, ..., png; Where, P is represented as image Co-ordinates conversion process to detect the physical structures of environment, sections and sections of number of markers. p1, p2, p3, ..., pn are the number of Parsing modules.
Failure Condition(FC):ffc1, fc2, fc3,.,fcng; Where, FC is represented as a unsuccessful conditions. fc1, fc2, fc3,.., fcn are number of final output.
Failure(F): image is not generated.
Deterministic Data(DD): image with extensions
Non-deterministic Data(NDD): Unstructured images

D. Coding and implementation

1) pose tracking of 3d object:

```
public class MutablePose3D : Pose3D {
  /// Sets the position and orientation from a Vector3 + Quaternion.
  public new void Set(Vector3 position, Quaternion orientation) {
    base.Set(position, orientation);
  }

  /// Sets the position and orientation from a Matrix4x4.
  public new void Set(Matrix4x4 matrix) {
    base.Set(matrix);
  }

  /// Sets the position and orientation from a right-handed Matrix4x4.
  public void SetRightHanded(Matrix4x4 matrix) {
    Set(FlipHandedness(matrix));
  }
}
```

2) registering tracker:

```
/// <summary>
/// Register this instance with the Tracking Method class for event handling.
/// </summary>
protected override void Register()
{
    if (_tracker != null)
    {
        _tracker._updateMarkerEvent.AddListener(OnTrackingUpdate);
    }
    this.gameObject.SetActive(false);

}
```

3) Rotation:

```
protected virtual void Update()
{
    // If we require a selectable and it isn't selected, cancel rotation
    if (RequiredSelectable != null && RequiredSelectable.IsSelected == false)
    {
        return;
    }

    // Get the fingers we want to use
    var fingers = LeanTouch.GetFingers(IgnoreGuiFingers, RequiredFingerCount);

    // Calculate the rotation values based on these fingers
    var center  = LeanGesture.GetScreenCenter(fingers);
    var degrees = LeanGesture.GetTwistDegrees(fingers);

    // Perform the rotation
    Rotate(center, degrees);
}
```

4) Scaling:

```
protected virtual void Update()
{
    // If we require a selectable and it isn't selected, cancel scaling
    if (RequiredSelectable != null && RequiredSelectable.IsSelected == false)
    {
        return;
    }

    // Get the fingers we want to use
    var fingers = LeanTouch.GetFingers(IgnoreGuiFingers, RequiredFingerCount);

    // Calculate the scaling values based on these fingers
    var pinchScale   = LeanGesture.GetPinchScale(fingers, WheelSensitivity);
    var screenCenter = LeanGesture.GetScreenCenter(fingers);

    // Perform the scaling
    Scale(pinchScale, screenCenter);
}
```

5) track n transform 3dobject:

```
/// <summary>
/// Method called every frame ArbiTrack is running.
/// Updates the position and orientation of the trackable.
/// </summary>
/// <param name="trackable">Trackable.</param>
///

public void OnTrackingUpdate(Trackable trackable)
{
    this.transform.localPosition = trackable.position;
    this.transform.localRotation = trackable.orientation;

    this.gameObject.SetActive(trackable.isDetected);
}
```

6) translate:

```
protected virtual void Update()
{
    // If we require a selectable and it isn't selected, cancel translation
    if (RequiredSelectable != null && RequiredSelectable.IsSelected == false)
    {
        return;
    }

    // Get the fingers we want to use
    var fingers = LeanTouch.GetFingers(IgnoreGuiFingers, RequiredFingerCount, RequiredSelectable);

    // Calculate the screenDelta value based on these fingers
    var screenDelta = LeanGesture.GetScreenDelta(fingers);

    // Perform the translation
    Translate(screenDelta);
}
```

7) unregistering tracker:

```
/// <summary>
/// Unregister this instance with the Tracking Method class for event handling.
/// </summary>
protected override void Unregister()
{
    if (_tracker != null)
    {
        _tracker._updateMarkerEvent.RemoveListener(OnTrackingUpdate);
    }
}
```

8) gps data:

```
void Start(){
    //get distance text reference
    distanceTextObject = GameObject.FindGameObjectWithTag ("distanceText");
    //start GetCoordinate() function
    StartCoroutine ("GetCoordinates");
    //initialize target and original position
    targetPosition = transform.position;
    originalPosition = transform.position;

}
```

9) gps earth algo:

```
//calculates distance between two sets of coordinates, taking into account the curvature of the earth.
public void Calc(float lat1, float lon1, float lat2, float lon2)
{

    var R = 6378.137; // Radius of earth in KM
    var dLat = lat2 * Mathf.PI / 180 - lat1 * Mathf.PI / 180;
    var dLon = lon2 * Mathf.PI / 180 - lon1 * Mathf.PI / 180;
    float a = Mathf.Sin(dLat / 2) * Mathf.Sin(dLat / 2) +
        Mathf.Cos(lat1 * Mathf.PI / 180) * Mathf.Cos(lat2 * Mathf.PI / 180) *
        Mathf.Sin(dLon / 2) * Mathf.Sin(dLon / 2);
    var c = 2 * Mathf.Atan2(Mathf.Sqrt(a), Mathf.Sqrt(1 - a));
    distance = R * c;
    distance = distance * 1000f; // meters
    //set the distance text on the canvas
    distanceTextObject.GetComponent<Text> ().text = "Distance: " + distance;
    //convert distance from double to float
    float distanceFloat = (float)distance;
    //set the target position of the object, this is where we lerp to in the update function
    targetPosition = originalPosition - new Vector3 (0, 0, distanceFloat * 12);
    //distance was multiplied by 12 so I didn't have to walk that far to get the object to show up closer

}
```

10) gps get co ordi n calc:

```
IEnumerator GetCoordinates()
{
    //while true so this function keeps running once started.
    while (true) {
        // check if user has location service enabled
        if (!Input.location.isEnabledByUser)
            yield break;

        // Start service before querying location
        Input.location.Start (1f,.1f);

        // Wait until service initializes
        int maxWait = 20;
        while (Input.location.status == LocationServiceStatus.Initializing && maxWait > 0) {
            yield return new WaitForSeconds (1);
            maxWait--;
        }

        // Service didn't initialize in 20 seconds
        if (maxWait < 1) {
            print ("Timed out");
            yield break;
        }

        // Connection has failed
        if (Input.location.status == LocationServiceStatus.Failed) {
            print ("Unable to determine device location");
            yield break;
        } else {
            // Access granted and location value could be retrieved
            print ("Location: " + Input.location.lastData.latitude + " " + Input.location.lastData.longitude + " " + Input.locat

            //if original value has not yet been set save coordinates of player on app start
            if (setOriginalValues) {
                originalLatitude = Input.location.lastData.latitude;
                originalLongitude = Input.location.lastData.longitude;
                setOriginalValues = false;//[illegible]
            }

            //overwrite current lat and lon everytime
            currentLatitude = Input.location.lastData.latitude;
            currentLongitude = Input.location.lastData.longitude;

            //calculate the distance between where the player was when the app started and where they are now.
            Calc (originalLatitude, originalLongitude, currentLatitude, currentLongitude);

        }
        Input.location.Stop();
    }
}
void Update(){
    //Linearly interpolate from current position to target position
    transform.position = Vector3.Lerp(transform.position, targetPosition, speed);
    //rotate by 1 degree about the y axis every frame
    transform.eulerAngles += new Vector3 (0, 1f, 0);

}
```

IV. RESULTS AND DISCUSSION

This method is used to automatically generate AR view considering the important point from the QR code or locked GPS co-ordinates. Augmented reality provides a better way of understanding for architectural firms by using 3D model increasing firms productivity, transparency and understanding. Thus, augmented reality will solve a huge gap of visualization in the architecture industry in an efficient way reducing the resources required.

ACKNOWLEDGMENT

We would like to extend our sincere gratitude and thanks to our guide “Prof. S. V. Shinde”, for his invaluable guidance and for giving us useful inputs and encouragement time and again, which inspired us to work harder. Due to his forethought, appreciation of the work involved and continuous imparting of useful tips, this report has been successfully completed. We are also grateful to “Prof. D.P.Gaikwad”, Head of the Department of Computer Engineering, for his encouragement during our project work.

“We also take this opportunity to thank all our staff of Computer Engineering Department, classmates, friends and all those who have directly or indirectly provided their overwhelming support during our project work and the development of this report.”

V. APPLICATION DESIGN

REFERENCES

[1] Max Krichenbauer, Goshiro Yamamoto, Member, IEEE, TakafumiTaketom, Member, IEEE, Christian Sandor, Member, IEEE, and Hirokazu Kato, Member, IEEE on Augmented Reality vs Virtual Reality for 3D Object Manipulation. IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, VOL. 14, NO. 8, AUGUST 2015.

[2] 2011 International Symposium on Ubiquitous Virtual Reality ARWand: Phonebased 3D Object Manipulation in Augmented Reality Environment Taejin Ha and Woontack Woo GIST U-VR Lab. 500-712, S. Korea tha@gist.ac.kr and wwoo@gist.ac.kr Taejin Ha and Woontack Woo GIST U-VR Lab. 500- 712, S. Korea tha@gist.ac.kr and wwoo@gist.ac.kr Taejin Ha and Woontack Woo GIST U-VR Lab. 500-712, S. Korea tha@gist.ac.kr and wwoo@gist.ac.kr.

[3] Yuan Lin, Shuang Song and Max Q.-H. Meng on The Implementation of Augmented Reality in a Robotic Teleoperation System. Proceedings of The 2016 IEEE International Conference on Real-time Computing and Robotics June 69, 2016, Angkor Wat, Cambodia.

[4] An Empirical Evaluation of Virtual Hand Techniques for 3D Object Manipulation in a Tangible Augmented Reality Environment Taejin Ha* andWoontack Woo GIST U-VR Lab., 500-712, S. Korea.

[5] The Research Bulletin of Jordan ACM, ISSN : 2078 - 7952, Volume II (III) Hand Gesture Interaction with a 3D Virtual Environment Nasser H. Dardas University of Ottawa 800 King Edward Avenue, Ottawa, ON K1N 6N5, Canada, Mohammad Alhaj Carleton University 1125 Colonel By Drive, Ottawa, ON K1S 5B6, Canada.

[6] 2016 IEEE International Symposium on Mixed and Augmented Reality Adjunct Proceedings PoLAR: a Portable Library for Augmented Reality PierreJeanPetitprezErwanKerrien Pierre-Frederic Villard Universite de Lorraine, LORIA, UMR 7503, Vandoeuvre-les-Nancy F-54506, France Inria, Villers-lesNancy F-54600, France CNRS, LORIA, UMR 7503, Vandoeuvre-les-Nancy F54506, France.

[7] ICACT Transactions on Advanced Communications Technology (TACT) Vol. 2, Issue 5, September 2013 A Stereo-Vision Approach for a Natural 3D Hand Interaction with an AR Object Seonho Lee, Junchul Chun Department of Computer Science, Kyonggi University, San 94-6 Yiui-Dong, Yeongtong-Gu, Suwon, S. Korea.

[8] Synchronous location-aware Media and Augmented visualization for Real world Tourist (SMART) An application for Khalifatabad heritage site, Bagerhat, Bangladesh HafizurRahaman * School of Architecture and Built Environment University of Newcastle, Australia hafizur.rahaman@newcastle.edu.au.

[9] The 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN06), Hatfield, UK, September 6-8, 2006 Direct-Manipulation Interface for Collaborative 3D Drawing in the Real World by Akihiro Osaki, Hiroyuki Taniguchi, Yoshiyuki Miwa.