ABSTRACT

Augmented Reality Based Indoor Navigation introduces to the concept of navigation system for smartphone which guide users to their destination inside a environment.

GPS (global positioning system) works perfectly outdoors but for indoor navigation this method doesn't work because of the obstructions that may be inside the building.

For indoor navigation Wi-Fi based positioning, Bluetooth beacons ,Visual positioning techniques are used for identifying the current location. Here visual positioning technique is used to scan QR and localize the 3D world according to that scanned position.

Integrating the system helps users to combine both physical and 3D world accordingly and help user navigate to a destination by overlaying Augmented reality cues on to the screen.

It will work on devices like smartphones, tablets based on Android which will guide the users accurately to their unknown destination.

CHAPTER 1 - INTRODUCTION

1.1 Problem Statement

To develop an android application for smartphones which helps the user to navigate through an unfamiliar indoor environment using Augmented Reality as an interface.

1.2 Introduction to SmartWay

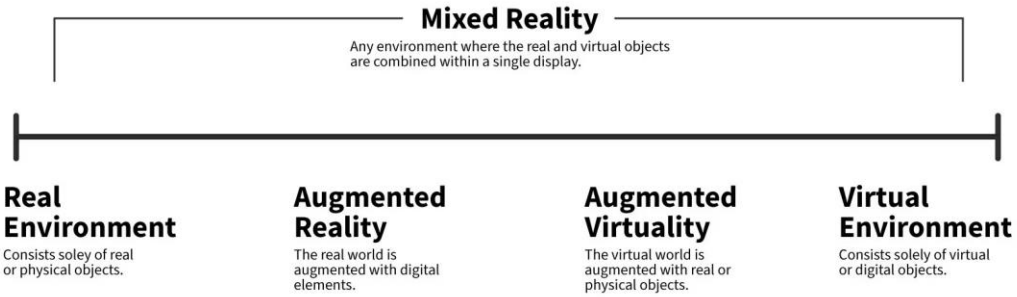
SmartWay is an advanced Augmented Reality (AR) based indoor navigation system designed to assist users in navigating complex and unfamiliar indoor environments—such as college campuses, hostels, hospitals, or malls easily. Built using AR technologies, SmartWay overlays virtual navigation cues onto the real world through mobile devices, offering user experience without the need for internet connectivity.

AR technology enhances the real-world view by superimposing digital content such as arrows, destination markers, footsteps, and labels. These virtual elements are anchored to real-world positions, making navigation as simple as following signs in the air. This makes SmartWay highly suitable for indoor spaces where traditional GPS signals are ineffective or unavailable.

The system uses image-based positioning through technologies like ARCore to accurately determine the user's location by scanning visual markers (like QR codes). Once the origin is identified, SmartWay dynamically generates a path to the selected destination and guides the user using AR overlays, animated directions, and distance indicators—all in real-time.

1.3 Augmented Reality

"The origin of the word augmented is augment, which means to add or enhance something. In the case of Augmented Reality (also called AR), graphics, sounds, and touch feedback are added into our natural world to create an enhanced user experience." In AR, we increase the usability of an interface by adding sensory information like computer generated images, sounds and in some cases, touch feedback over user's view of the real world. This enhances user's current perception of reality.



1.3.1 Types of AR

Marker-based AR

Marker-based augmented reality (also called Image Recognition) is one of the easiest type of AR for implementation. Based on image recognition, marker based AR works by recognising a visual marker in the environment and generating a digital image only when a known marker is sensed. The markers are simple and unique, such as the QR codes. It finds its applications in the manufacturing and construction industry.

Marker-less AR

In markerless augmented reality, sensors in devices are used to detect the real world environment. With the emergence of smart devices, elements such as GPS, accelerometers, velocity meter, digital compass are pre included in the device which make the existence of Markerless AR possible. A strong force behind markerless augmented reality technology is the wide availability of smartphones and location detection features they provide. It is most commonly used for mapping directions, finding nearby businesses, and other location-centric mobile applications.

Projection based AR

This is a relatively newer trend in AR, which works on the principle of advance projection technology that forecasts light onto real world surfaces and senses human interaction with the light. This process is carried out by distinguishing between the expected and altered projection. This type of AR is usually used in manufacturing companies to assist in manufacturing, assembly, sequencing and training operations.

Superimposition based AR

Superimposition based augmented reality works with object detection and recognition. It recognises an object and then replaces it by superimposing a digital image over it. Some consumer-facing examples of superimposition based augmented reality can be organising furniture in a room or virtual trial rooms for clothing.

1.4 Aim and Objectives

- The aim of the project is to develop an AR based indoor navigation system for handheld devices like smartphones, tablets etc. using unity to identify location with an intuitive user interface.
- To make indoor navigation seamless and interactive through a hand-held mobile device using AR technology.
- To determine the current user location inside the hostel by analyzing nearby visual markers (like QR codes or posters) using vision-based feature extraction and visual positioning techniques. These techniques help the app estimate the user's position in real-time by comparing the camera feed with known marker locations in the 3D environment.
- To determine of shortest path from current location to desired destination and navigate user by augmenting directions in user’s real-world view captured.

CHAPTER 2 - LITERATURE REVIEW

2.1 Augmented Reality (Base Paper)

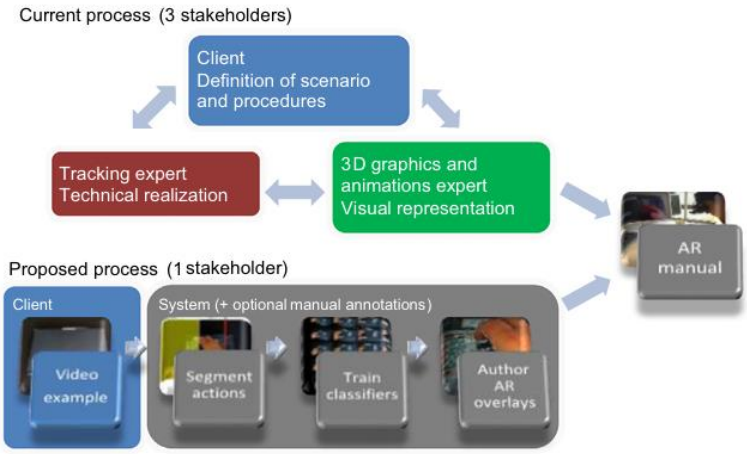
Cognitive Augmented Reality. Nils Petersen, Didier Stricker (22 August 2015)

In the early 1990s, researchers Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino introduced a concept called the reality-virtuality (RV) continuum (Milgram, 1994). While the researchers originally designed the reality-virtuality continuum to address mixed reality and the display technologies of the era, the original framework is still quite useful.

They defined mixed reality environments as those in which “real world and virtual world objects are presented together”. Their definition of mixed reality served as an umbrella term that encompassed both virtual and augmented reality technologies.

It explores and explains their Cognitive AR system for getting an AR manual, showing step-by-step instructions to a user wearing a Head-Mounted Display (HMD). It also presents a complete approach for creating augmented reality content for procedural tasks from video examples and give the details about the presentation of such content at runtime. According to the paper, today, applications face problems in giving a good demonstration due to some reasons like difficulty in content creation, ergonomic and hardware limitations and lack of “System Intelligence”.

In this paper they’ve shown that an effective approach for procedural workflows can be developed using visual observation only. An illustration of the simplified authoring and implementation process is shown.



The paper presents an overview of a novel learning-based authoring approach towards procedural task assistance using Augmented Reality. The resulting system is comprehensive and allows the fully automatic creation of Augmented Reality manuals from video examples as well as their context-driven presentation in AR. The recording of a single reference recording of a workflow is sufficient for a practical system.

With availability of additional reference recordings, the system not only improves in precision and recall but is also able to estimate certain task-specific properties, like required level of accuracy and distinction of erratic and intended actions. The presented approach is the first to combine classical AR with machine learning, classification and basic reasoning methods, leading towards a cognitive system, aware about scene state and user actions. To underscore the extension of the merely spatial paradigm of Augmented Reality with the cognitive components, we call this combination Cognitive Augmented Reality.

Beside the presented live augmentation with Head-Mounted Display, the technology and methodology proposed in this paper opens many additional fields of application, such as the automated generation of written task documentation, support for documenting error indications, or analysis of maintenance procedures on a process level.

2.2 Indoor Navigation(Base Paper)

SINS_AR:An Efficient Smart Indoor Navigation System Based on Augmented Reality

Authors:Yasmin Alkady,Rawya Rizk,Deema Mohammed Alsekait,Ala Saleh Alluhaidan,Diaa Salama Abdelminaam

The Smart Indoor Navigation System Using Augmented Reality (SINS_AR) is designed to enhance navigation within complex indoor environments, such as shopping malls, hotels, and universities. The system leverages augmented reality to provide users with visual guidance, making it easier to locate specific destinations. Here's a breakdown of the navigation process involved in SINS_AR:

Steps of Navigation

User Input:

Users begin by selecting their desired destination within the application. This could be a shop in a mall, a room in a hotel, or a lecture hall in a university

Visual Marker Detection:

The system utilizes visual markers placed throughout the indoor environment. When users scan these markers with their smartphones, the application recognizes them and begins to calculate the optimal path to the selected destination

Path Calculation:

The SINS_AR system employs the Theta* algorithm for path planning. This algorithm is advantageous as it allows for more efficient pathfinding by considering multiple potential paths and reducing unnecessary turns. Unlike the A* algorithm, Theta* can check for line-of-sight between vertices during the expansion phase, which helps in creating smoother paths

Augmented Reality Guidance:

Once the path is calculated, the application overlays guidance arrows onto the real-world view displayed on the user's smartphone. This visual assistance helps users navigate through the indoor space effectively .

Dynamic Destination Change:

Users have the flexibility to change their destination at any point during navigation. The system recalculates the path based on the new input, ensuring that users always have the most efficient route available.

User Feedback and Adjustment:

As users navigate, the system may provide real-time feedback or adjustments based on their movements and any obstacles encountered, ensuring a seamless navigation experience.

Conclusion

The SINS_AR system represents a significant advancement in indoor navigation technology by integrating augmented reality with efficient pathfinding algorithms like Theta*. This combination not only enhances user experience but also addresses the challenges posed by traditional GPS systems in indoor environments, where signal quality is often compromised

CHAPTER 3 - PROJECT REQUIREMENTS

3.1 Hardware

3.1.1 Development Hardware:

- Computer/Laptop
- Smartphone/Tablet: A modern Android or iOS device with a high-resolution camera and AR support
- Camera:A webcam or smartphone camera for testing ArUco marker detection and tracking during development.

3.1.2 Deployment Hardware:

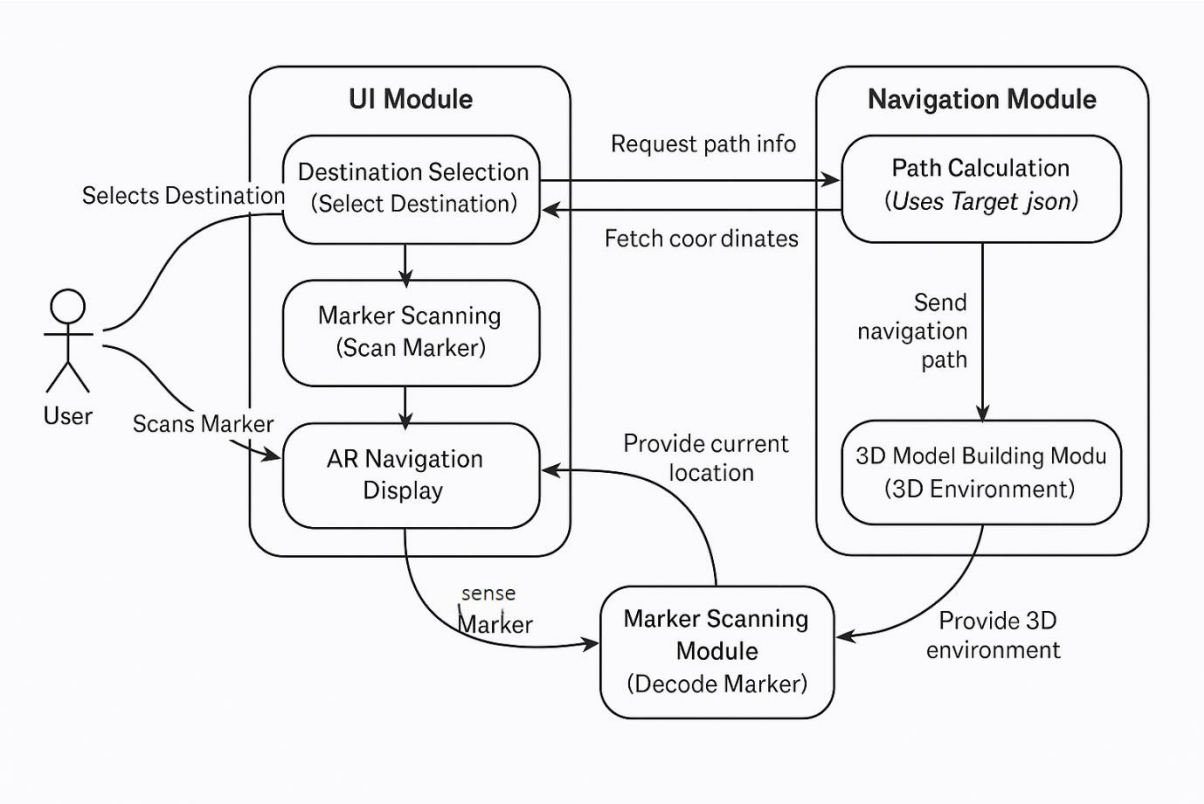
- Markers: Printed markers placed strategically throughout the Hostel to serve as reference points for tracking user position.
- User Devices:Smartphones(which supports depth API) or AR glasses for end-users to access the AR navigation system.

3.2 Software

3.2.1 Development Tools

- Unity
- C#
- VS code
- ARCore/ARKit
- AR Foundation

CHAPTER 4 - PROPOSED SYSTEM ARCHITECTURE



1.Building 3D model Module:

- Create a 3D model of the building.
- Using Unity

2.Marker Scanning Module:

- Use camera to detect markers(QR code) placed around building to calculate your current position.
- Using ZXing library to deccode QR.

3.Navigation Module:

- Calculate shortest path from current marker to selected destination.
- Using Navmesh to find the shortest path.
- Display AR cues

4.UI module:

- Interact with the system
- Buttons to navigate through panels
- Using Unity UI System

Procedure:

Step 1: User Interaction

User selects the destination through the app after scanning a QR code or using the camera to identify the current location.

Step 2: Data Fetching from JSON file

The Select Destination module sends a request to fetch location/path data from a JSON file stored locally within the app.

Step 3: Receiving Navigation Data

The JSON file returns data such as:

- Start and end point coordinates
- Destination name and info

Step 4: Navigation Processing

The Navigation Module processes:

- Current user position
- Direction

Step 5:AR Guidance Display

The app shows the path to the user using AR overlays:

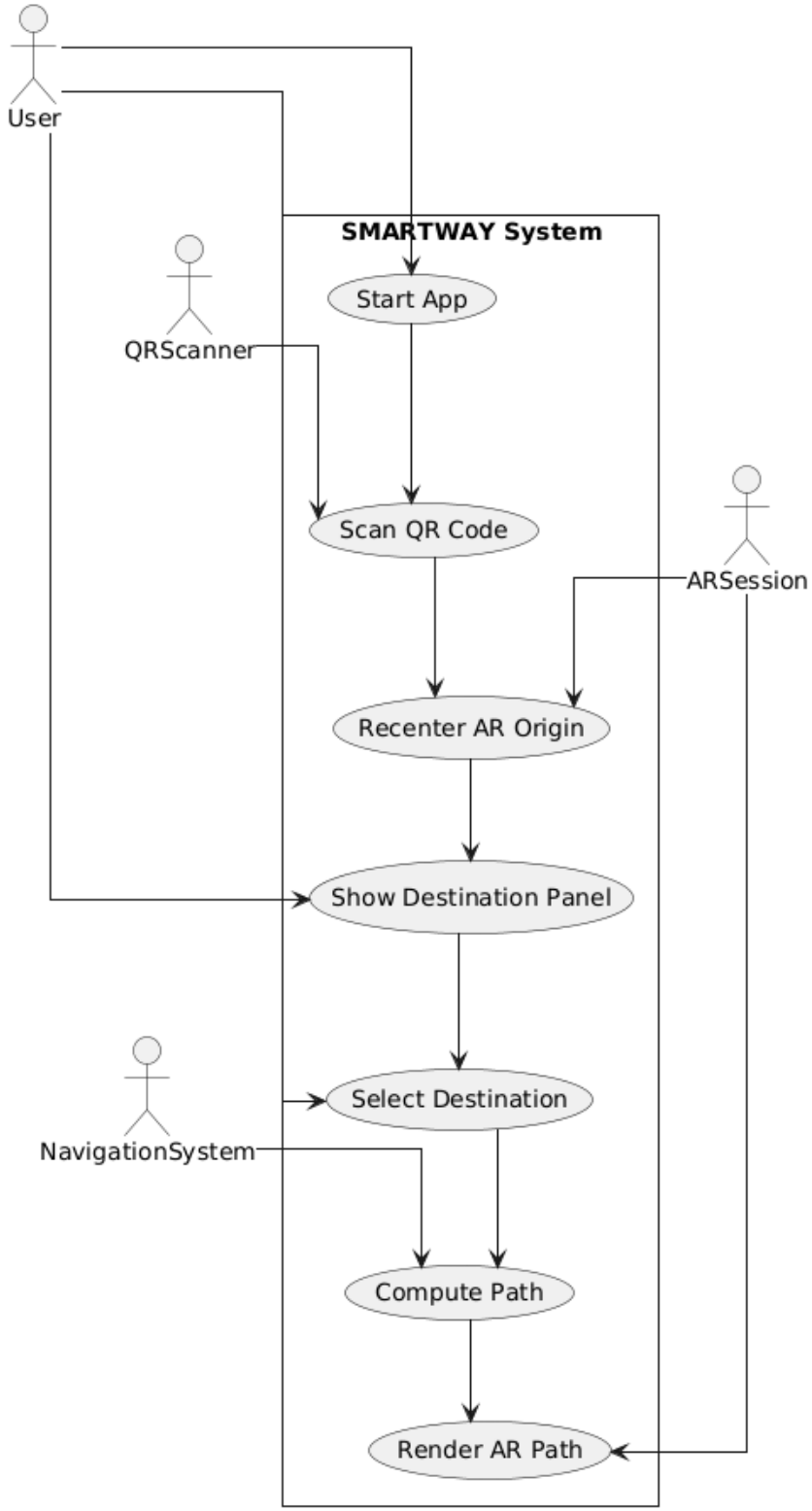
- Arrows,Footsteps,destination markers
- The overlays are aligned with real world view using AR Foundation
- N

Step 6: User follows the AR path

The user follows the visual cues in the AR interface toward the destination.

CHAPTER 5 - PROJECT DESIGN

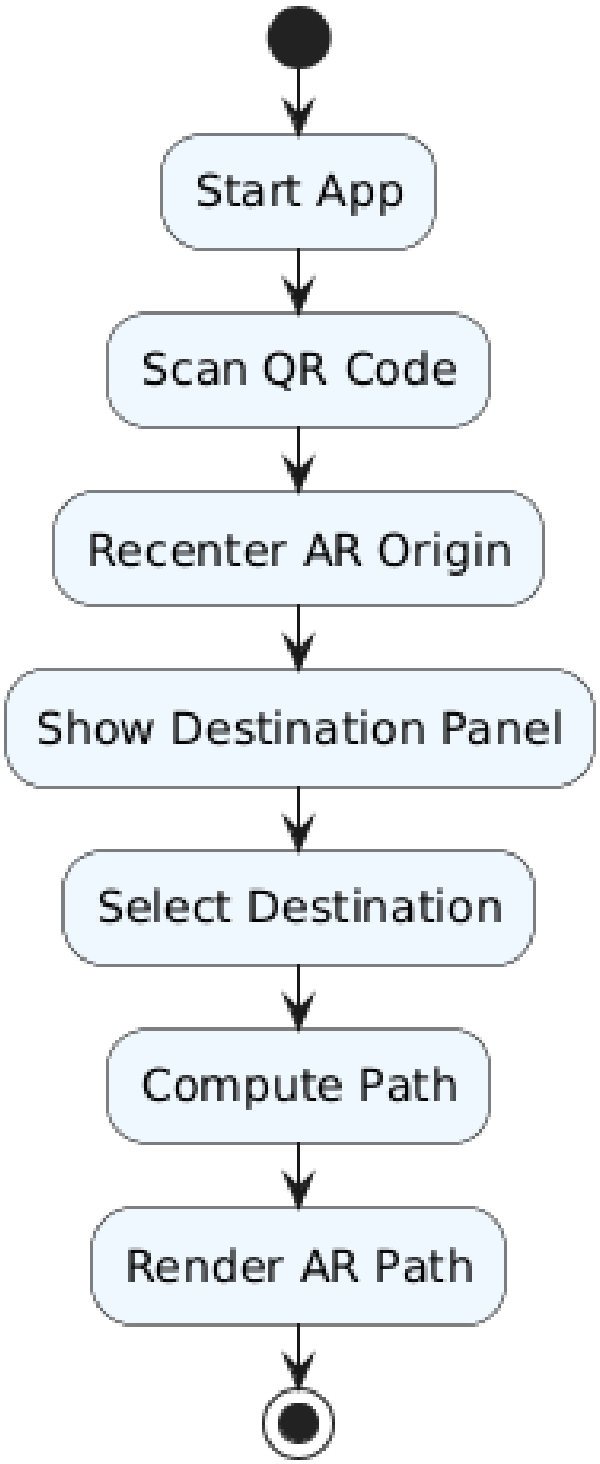
5.1 Use Case Diagram



Description:

Actor	Use Case	Description / Responsibility
User	Start App	Launches the SmartWay app and initiates navigation.
	Show Destination Panel	Views the destination selection interface after positioning is set.
	Select Destination	Chooses a location from the list of available destinations.
QRScanner	Scan QR Code	Uses the device camera to scan QR codes placed in the environment to determine current location.
ARSession	Recenter AR Origin	Resets the AR coordinate system using the scanned QR code to ensure accurate localization.
	Render AR Path	Displays AR navigation overlays (arrows, footsteps, path lines) in the physical space.
Navigation System	Compute Path	Calculates the optimal route from the user's current position to the selected destination.

5.2 Activity Diagram



Description:

1. Start App: The user opens the SmartWay app, which initializes the system and prepares it for navigation. This triggers the display of the app's main interface.
2. Scan QR Code: The QR scanner uses the device's camera to scan QR codes placed in the environment. This is a critical step for determining the user's current location within the indoor space.
3. Recenter AR Origin: Upon scanning the QR code, the system recalibrates the AR origin (coordinate system) to ensure accurate location tracking. This helps maintain precision as the user navigates
4. Show Destination Panel: Once the positioning is determined, the system presents the destination selection interface to the user. This panel allows users to choose where they want to navigate to.
5. Select Destination: The user selects a destination from the available options. The selection process is crucial for the navigation path to be calculated correctly.
6. Render AR Path: The system displays AR overlays such as arrows, footsteps, and path lines in the physical space, guiding the user along the calculated route.
7. Compute Path: The navigation system calculates the most efficient path to the selected destination, taking into account the user's position and the layout of the space.

CHAPTER 6 - SYSTEM IMPLEMENTATION

6.1 Source Detection

- Source detection here refers to determine the users current location relative to the intended region
- The prime locations are identified by markers which consist of the location coordinates.
- Marker based localisation occur here.
- The user captures the qr code using camera, it is verified by the application by extracting the text from themarker and matching it with the json file.

6.1.1 Marker Based Localisation

Localisation refers to the process of determining where something is or adapting something to a specific location.

Marker based localisation is a technique used to determine the position and orientation (pose) of a device (eg:camera,AR headset) relative to predefined visual markers placed in the environment.These markers are designed to be easily detectable by sensors (like cameras) and serve as reference points for spatial tracking.

Key Components:

- Markers
- Decoding(Using ZXing library)
- Pose estimation

How it works

- Marker Placement:placement of markers in physical environment
- Image Capture:Camera captures the Marker
- Marker Detection and Decoding:Using ZXing library
- Localization:Computes the devices 6DOF pose(x,y,z position+roll,pitch,yaw orientation)relative to the marker.

6.1.2 QR code recognition using ZXing

ZXing (Zebra Crossing) is a library that supports QR code decoding. It works across platforms (Java, Android, C++, etc.) and handles tasks like:

- Detecting QR codes in images.
- Decoding embedded data (text, URLs, contact info, etc.).
- Handling rotated, skewed, or damaged codes using error correction.

6.2 Navigation

It is the process or activity of guiding the user to the destination following an appropriate route.

6.2.1 Tracking

6.2.1.1 VIO(Visual Inertial Odometry)

Used for accurate motion tracking by combining data from visual sensors(camera) and inertial sensors(accelerometer+gyroscope)

Working of VIO

- Calibrate initial pose using camera + IMU alignment
- Track movement across frame
- Use gyroscope & accelerometer to estimate short term motion
- Combine visual tracking & inertial estimation using filtering

Filtering

Filtering in VIO refers to estimating devices current state over time by:

- Predicting the next state using a motion model
- Correcting that prediction using actual measurements

6.2.2 Pathfinding

Unity uses Navmesh to calculate all possible paths the user can move.
It calculates the shortest path to destination by using A star algorithm.

6.2.2.1 Navmesh

A mesh is a 3D object made up of vertices,edges and faces that define the shape of a model.This defines the visual geometry of objects.Used for rendering ,physics collisions and animations.

NavMesh is a simplified ,invisible 3D mesh used for AI pathfinding.It defines walkable areas where user can walk.

6.2.2.2 A star algorithm

Graph traversal and pathfinding algorithm that finds shortest path from start node to goal node.

Psuedocode:

```
1: procedure Main(void)
2:   g(sstart) ← 0
3:   parent(sstart) ← NULL
4:   open ← ∅
5:   open.Insert(sstart, g(sstart) + h(sstart))
6:   closed ← ∅
7:   while open ≠ ∅ do
8:     s ← open.Pop()  // Get node with lowest f = g + h
9:     if s = sgoal then
10:      return "path found"
11:    end if
12:    closed ← closed ∪ {s}
13:    for each s' ∈ succ(s) do
14:      if s' ∈ closed then
15:        continue
16:      end if
17:      tentative_g ← g(s) + c(s, s')
18:      if s' ∉ open or tentative_g < g(s') then
```

```
19:       parent(s') ← s
20:       g(s') ← tentative_g
21:       f ← g(s') + h(s')
22:       if s' ∉ open then
23:         open.Insert(s', f)
24:       else
25:         open.Update(s', f)
26:       end if
27:     end if
28:   end for
29:  end while
30:  return "no path found"
31: end procedure
```

6.3 Augmented Realty

Augmented reality (AR) is a type of interactive, reality-based display environment that takes the capabilities of computer generated display, sound, text and effects to enhance the user's real-world experience. Augmented reality combines real and computer-based scenes and images to deliver a unified but enhanced view of the world.

Technology used (ARCore): ARCore is Google's platform for building augmented reality experiences that seamlessly blend the digital and physical worlds. ARCore uses three key technologies to integrate virtual content with the real world as seen through your phone's camera:

1. Motion tracking allows the phone to understand and track its position relative to the world.
2. Environmental understanding allows the phone to detect the size and location of flat horizontal surfaces like the ground or a coffee table.
3. Light estimation allows the phone to estimate the environment's current lighting

CHAPTER 7 - TESTING

1.Capture QR code(Valid)

Test data:QR code with prominent coordinates

Steps	Description	Expected Result
Step 1	Open Camera of mobile from our application	Camera opens successfully
Step 2	Scan QR	QR scanned successfully, Getting location coordinates

2.Capture QR code(invalid)

Test data:QR code with no prominent coordinates

Steps	Description	Expected Result
Step 1	Open camera	Camera opens successfully
Step 2	Scan QR	Doesn't Do anything

3.Select Destination Location(valid)

Test data:201

Steps	Description	Expected Result
Step 1	Select Destination location	Selects Destination
Step 2	Proceed to navigation	Display path from current location to destination

4.Augmented Reality

Steps	Description	Expected Result
Step 1	Proceed to navigation	User gets navigated to the destination using Augmented Reality. Easy guidance for the user with the help of AR

CHAPTER 8 - APPLICATIONS

Airports

Directions directly to the gate, a specific shop or to a restaurant. Improve the traveler's experience and reduce the number of people missing their flights.

Universities

Offer students, employees and visitors indoor navigation and guide them to lecture rooms, canteens and printers.

Shopping centres

With indoor wayfinding shoppers are guided to specific shops and restaurants, which makes the shopping experience more fun and satisfying

Hospitals

Indoor navigation can improve your staff's efficiency and reduce patient and visitor stress by providing a familiar and easy-to-use wayfinding tool.

Other large venues

Indoor navigation can be applied to all large venues. Corporate headquarters, convention centres, theme parks and many more.

CHAPTER 9 - FUTURE SCOPE

1. It is possible to extend the scope of the application to multiple floors . Currently, it is devised only for a single floor.

2. Redirecting the user while navigation, in case the user takes an incorrect step can be handled in the future scope.

3. This application has been customized only for a particular indoor space, but can be later modeled dynamically for various indoor spaces.

4. The application currently works on Android devices but can be extended to iOS as well.

CHAPTER 10 - CONCLUSION

The various indoor navigation techniques currently available, such as Wi-Fi fingerprinting, Bluetooth Low Energy (BLE) beacons, and other external hardware requirements, often necessitate additional infrastructure like Wi-Fi access points or BLE beacons. However, marker-based localization eliminates these hardware dependencies by relying on the smartphone's camera and marker recognition, offering a cost-effective and scalable solution for indoor navigation. By integrating computer vision, the accuracy of the navigation system is enhanced, providing more precise positioning without the need for additional equipment.

Regarding augmented reality (AR) types and implementation methods, marker-based AR uses specific markers to anchor virtual content to physical locations within the environment. While AR can be implemented using head-mounted displays (HMDs), a smartphone-based solution offers better usability and doesn’t require the storage of distinct images or codes beyond the markers themselves. This marker-based AR system provides seamless, real-time navigation, offering an intuitive and interactive user experience without the need for external hardware.

CHAPTER 11 – REFERENCES

[1] Journal article - Nils Petersen, Didier Stricker . Cognitive Augmented Reality . 22 August 2015

[2]Research Article-Yasmin Alkady,Rawya Rizk,Deema Mohammed Alsekait,Ala Saleh Alluhaidan,Diaa Salama Abdelminaam. SINS_AR: An Efficient Smart Indoor Navigation System Based on Augmented Reality. 6 August 2024

[3] Jennifer Pearson, Simon Robinson, Matt Jones. BookMark: Appropriating Existing Infrastructure to Facilitate Scalable Indoor Navigation FIT Lab, Computer Science Department, Swansea University, Swansea, SA2 8PP, UK, 2 February 2017

[4]B Rokesh Maran,L Giridharan,RKrishnaveni .Augmented Reality – based Indoor Navigation using Unity Engine. 07 July 2023

[5]ARFoundation Documentation [online]: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html

[6] https://www.techopedia.com/definition/4776/augmented-reality-ar.

[7]Unity Discussions:ZXing Library with Unity: https://discussions.unity.com/t/zxing-library-with-unity/585903/4

APPENDIX-A:Codes

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
using UnityEngine.XR.Interaction.Toolkit;
using ZXing;
using Unity.Collections;
using Unity.XR.CoreUtils;
using System.Collections;

public class QrCodeRecenter : MonoBehaviour
{

    [SerializeField] private ARSession arSession;
    [SerializeField] private XROrigin xrOrigin;
    [SerializeField] private ARCameraManager arCameraManager;
    [SerializeField] private GameObject scanningPanel;
    [SerializeField] private GameObject successPanel;
    [SerializeField] private GameObject nextPanel;
    [SerializeField] private float successDelay = 2f;
    [SerializeField] private TargetHandler targetHandler;

    private Texture2D cameraImageTexture;
    private IBarcodeReader barcodeReader = new BarcodeReader();
    private bool isScanning = false;

    public void Initialize(ARCameraManager cameraManager, TargetHandler
handler)
    {
        arCameraManager = cameraManager;
        targetHandler = handler;
    }

    private void OnEnable() => arCameraManager.frameReceived +=
OnCameraFrameReceived;
    private void OnDisable() => arCameraManager.frameReceived -=
OnCameraFrameReceived;

    public void ToggleScanning()
    {
```

```
        isScanning = !isScanning;
        scanningPanel.SetActive(isScanning);

    }

    private void OnCameraFrameReceived(ARCameraFrameEventArgs eventArgs)
    {
        if (!isScanning || !arCameraManager.TryAcquireLatestCpuImage(out
XRCpuImage image))
            return;

        ProcessImage(image);
    }

    private void ProcessImage(XRCpuImage image)
    {
        var conversionParams = new XRCpuImage.ConversionParams
        {
            inputRect = new RectInt(0, 0, image.width, image.height),
            outputDimensions = new Vector2Int(image.width / 2,
image.height / 2),
            outputFormat = TextureFormat.RGBA32,
            transformation = XRCpuImage.Transformation.MirrorY
        };

        int bufferSize = image.GetConvertedDataSize(conversionParams);
        var buffer = new NativeArray<byte>(bufferSize, Allocator.Temp);

        image.Convert(conversionParams, buffer);
        image.Dispose();

        CreateTextureFromBuffer(buffer, conversionParams);
        buffer.Dispose();

        ScanQrCode();
    }

    private void CreateTextureFromBuffer(NativeArray<byte> buffer,
XRCpuImage.ConversionParams parameters)
    {
        cameraImageTexture = new Texture2D(
            parameters.outputDimensions.x,
            parameters.outputDimensions.y,
```

```
            parameters.outputFormat,
            false);

        cameraImageTexture.LoadRawTextureData(buffer);
        cameraImageTexture.Apply();
    }

    private void ScanQrCode()
    {
        var                          result                          =
barcodeReader.Decode(cameraImageTexture.GetPixels32(),
            cameraImageTexture.width, cameraImageTexture.height);

        if (result != null)
            HandleQrDetection(result.Text);
    }

    private void HandleQrDetection(string qrText)
    {
        var target = targetHandler.GetCurrentTargetByTargetText(qrText);
        if (target == null) return;

        RecenterARSession(target);
        ShowSuccessUI();
        ToggleScanning();
    }


    private void RecenterARSession(TargetFacade target)
{
    StartCoroutine(SmoothRecenter(target));
}

private IEnumerator SmoothRecenter(TargetFacade target)
{
    arSession.Reset();  // Reset AR tracking

    yield return new WaitForSeconds(0.1f); // give AR time to reset

    xrOrigin.transform.SetPositionAndRotation(
        target.transform.position,
        target.transform.rotation
    );
```

```
}

    private void ShowSuccessUI()
    {


        successPanel.SetActive(true);
        scanningPanel.SetActive(false);

        Invoke(nameof(ShowNextPanel), successDelay);
    }

    private void ShowNextPanel()
    {

FindFirstObjectByType<BackButtonManager>().ShowPanel(nextPanel);


        successPanel.SetActive(false);
        nextPanel.SetActive(true);
    }
}
```

QRrecenterCode

```
using System.Collections.Generic;
using System.Linq;
using TMPro;
using UnityEngine;

public class TargetHandler : MonoBehaviour {

    [SerializeField]
    private NavigationController navigationController;
    [SerializeField]
    private TextAsset targetModelData;
    [SerializeField]
    private TMP_Dropdown targetDataDropdown;

    [SerializeField]
    private GameObject targetObjectPrefab;
    [SerializeField]
    private Transform[] targetObjectsParentTransforms;

    private List<TargetFacade> currentTargetItems = new
List<TargetFacade>();

    private void Start() {
        GenerateTargetItems();
        FillDropdownWithTargetItems();
    }

    private void GenerateTargetItems() {
        IEnumerable<Target> targets = GenerateTargetDataFromSource();
        foreach (Target target in targets) {
            currentTargetItems.Add(CreateTargetFacade(target));
        }
    }

    private IEnumerable<Target> GenerateTargetDataFromSource() {
        return
JsonUtility.FromJson<TargetWrapper>(targetModelData.text).TargetList;
    }

    private TargetFacade CreateTargetFacade(Target target) {
        GameObject targetObject = Instantiate(targetObjectPrefab,
targetObjectsParentTransforms[target.FloorNumber], false);
```

```
        targetObject.SetActive(true);
        targetObject.name = $"{target.FloorNumber} - {target.Name}";
        targetObject.transform.localPosition = target.Position;
        targetObject.transform.localRotation =
Quaternion.Euler(target.Rotation);

        TargetFacade targetData =
targetObject.GetComponent<TargetFacade>();
        targetData.Name = target.Name;
        targetData.FloorNumber = target.FloorNumber;

        return targetData;
    }

    private void FillDropdownWithTargetItems() {
        List<TMP_Dropdown.OptionData> targetFacadeOptionData =
            currentTargetItems.Select(x => new TMP_Dropdown.OptionData {
                text = $"{x.FloorNumber} - {x.Name}"
            }).ToList();

        targetDataDropdown.ClearOptions();
        targetDataDropdown.AddOptions(targetFacadeOptionData);
    }

    public void SetSelectedTargetPositionWithDropdown(int selectedValue)
{
        navigationController.TargetPosition =
GetCurrentlySelectedTarget(selectedValue);
    }

    private Vector3 GetCurrentlySelectedTarget(int selectedValue) {
        if (selectedValue >= currentTargetItems.Count) {
            return Vector3.zero;
        }

        return currentTargetItems[selectedValue].transform.position;
    }

    public TargetFacade GetCurrentTargetByTargetText(string targetText)
{
        return currentTargetItems.Find(x =>
            x.Name.ToLower().Equals(targetText.ToLower()));
    }
```

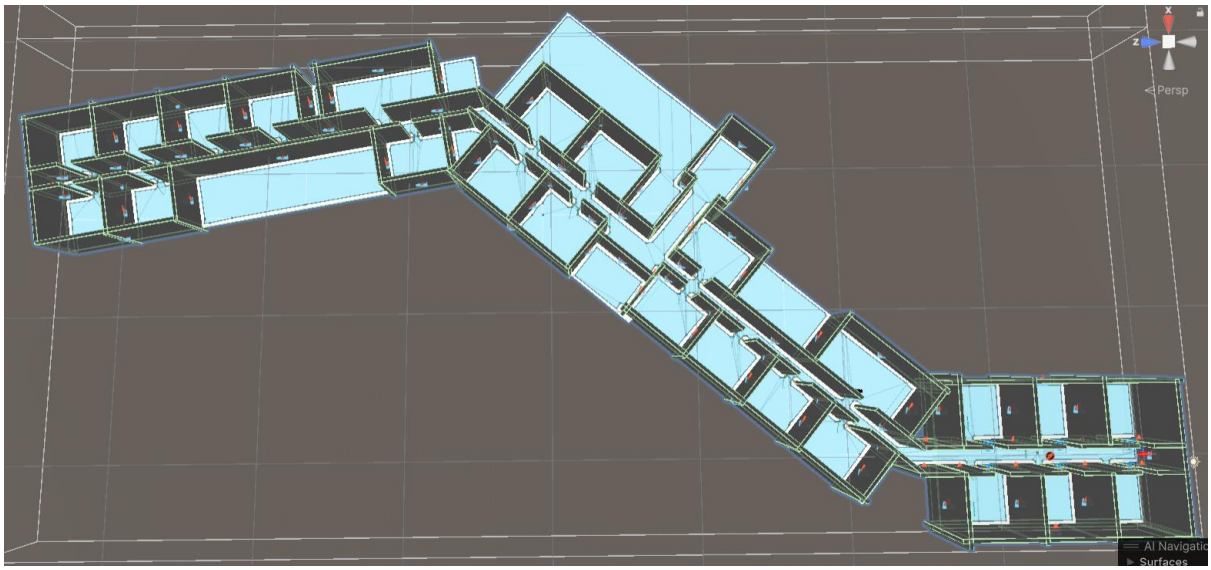
Targethandler.cs

```
using UnityEngine;
using UnityEngine.AI;

public class NavigationController : MonoBehaviour {

    public Vector3 TargetPosition { get; set; } = Vector3.zero;


    public NavMeshPath CalculatedPath { get; private set; }

    private void Start() {
        CalculatedPath = new NavMeshPath();
    }

    private void Update() {

        if (TargetPosition != Vector3.zero) {
            NavMesh.CalculatePath(transform.position,    TargetPosition,
NavMesh.AllAreas, CalculatedPath);
        }
    }
}
```

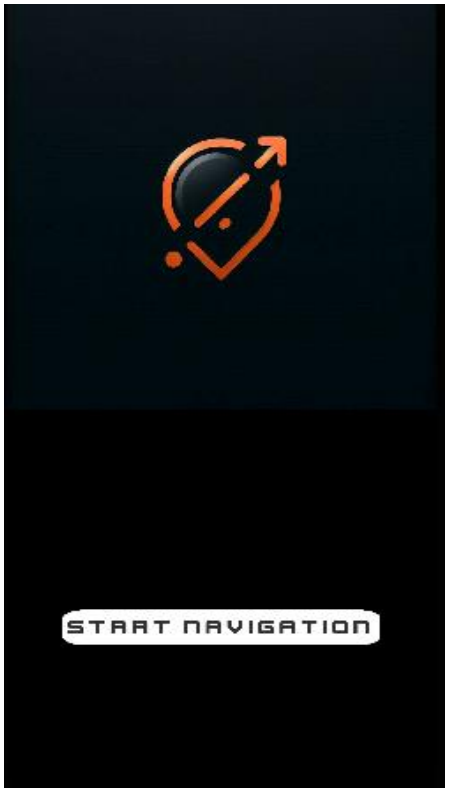
NavigationController.cs

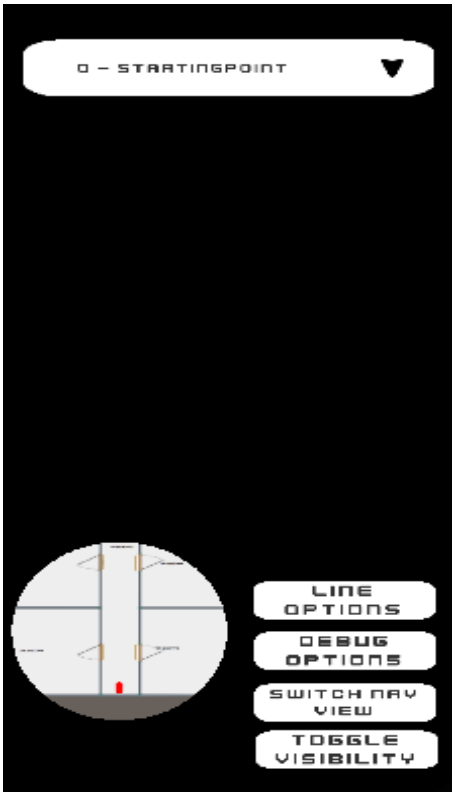
APPENDIX-B:Screenshots

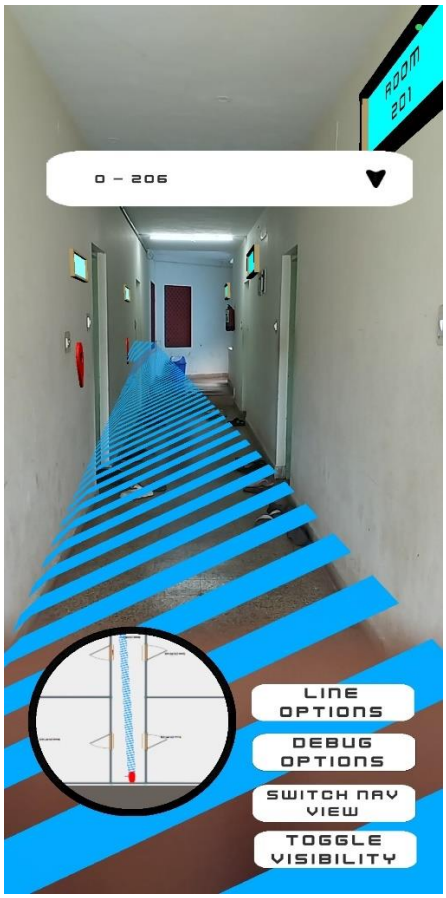
3D MODEL OF HOSTEL



UI


START NAVIGATION


0 – STARTINGPOINT
LINE OPTIONS
DEBUG OPTIONS
SWITCH NAV VIEW
TOGGLE VISIBILITY


ROOM 201
0 – 206
LINE OPTIONS
DEBUG OPTIONS
SWITCH NAV VIEW
TOGGLE VISIBILITY


ROOM 201
0 – 206
LINE OPTIONS
DEBUG OPTIONS
SWITCH NAV VIEW
TOGGLE VISIBILITY