Taibah University
College of Computer Science Engineering
Computer Engineering Department

Mosaned — Navigation and Guidance System for Indoor Environments for Blind and Visually Impaired Individuals

A Project Submitted in partial fulfilment of the requirements for the Bachelor Degree in Computer Engineering

Submitted By

Mohammed Alharbi	**4200174**
Firas Alsadeq	**4200625**
Mousa Alenzi	**4106574**

Project Advisor
<Title><Moteb Alghamdi>

<2024-2025>-<1st Semester>

ABSTRACT

Visually impaired and blind individuals might face difficult challenges when navigating through indoor environments, especially if they where not familiar with them. Indoor environments can be very unpredictable and filled with obstacles, and they are rich of important visual information which only designed for eyes consumption. These difficulties rise the need of these individuals to assistants and guiders whether by asking strangers, or by taking companions. But these solutions will lead to decreasing the independence of these individuals. We are trying to offer a solution in this project, a wearable and easy to use system that guides and assists the users after determining their positions and scanning their surrounding environments using only a camera and smart phone.

The proposed system is an integration of three systems: Objects Detection, Localization, and Customizable Guidance System. These components rely on analyzing the images captured by a camera and extracting the important visual information from them. The Objects Detection System can detects the surrounding objects and describe them to the users if they asked to, and informusers about obstacles, and users can ask it to alarm them when they face an object with certain characteristics, such as a man wearing red T-Shirt. The localization system will use the camera to detect artificial landmarks to determine the user's location. And finally, the Customizable Guidance System reads to user custom information and instructions after entering specific areas.

- the result obtained

- the significance of the result or finding

KEYWORDS: Localization; Obstacles Avoidance; Objects Detection, Customization; Guidance; Pose; Calibration

ACKNOWLEDGEMENTS

The content of this single page is left to the preference of the student. It is suggested however that the page makes reference to guidance received by the student from his or her supervisor and project Committee members. Reference should also be made to any financial assistance received to carry out the project. Any extraordinary assistance received by the student for example in word processing, data collection, data analysis, and so on, should be properly acknowledged. The acknowledgements should not exceed 250 words.

Contents

List of Tables

List of Figures

Glossary and List of Abbreviations

Glossary

Term 1 Definition of term 1.

Term 2 Definition of term 2.

List of Abbreviations

Abbr. Full form of the abbreviation.

Chapter 1

Introduction

1.1 Overview

Navigation in indoor areas presents considerable difficulties for those with visual impairments. Conventional GPS systems, while proficient in outside navigation, are inadequate for indoor environments due to diminished satellite signals and the intricate designs of spaces such as retail malls, hospitals, and educational institutions. Researchers have investigated alternate ways, including the use of artificial landmarks, to facilitate indoor navigation.

Artificial landmarks, including QR codes, offer a cost-effective and readily implementable alternative for localization and navigation in interior settings. These landmarks are identifiable by cameras or sensors, enabling visually challenged individuals to obtain audio or tactile feedback for navigational assistance. This project proposes a QR code-based navigation system to ensure precise localization and dependable obstacle detection for those with visual impairments. This project aim to develop a low-cost indoor navigation system that will combine accurate indoor localization, reliable obstacle detection, and customizable, for indoor environment using (... Tools and Software and QR code) to help with visual impairments.

1.2 Motivation

This project is motivated by the need to enhance the quality of life and autonomy for visually impaired individuals, especially in interior settings where conventional navigation aids such as GPS are inadequate. Visually impaired individuals encounter significant obstacles when navigating intricate indoor environments, which can restrict their mobility and independence. Existing options, such as guide dogs and human aid, although beneficial, may prove inadequate or unworkable in all situations. The motivation also is to develop a prototype solution utilising commonly accessible technology (such as QR codes and mobile devices) enhances the system's accessibility to a broader audience (Cost and Accessibility). This differs from expensive or complex systems that may require specialized hardware.

1.3 Problem Statement

Navigating indoor environments is challenging for visually impaired individuals, directly affecting their independence and daily activities. While some existing technologies have developed separate algorithms for indoor localization, they are often not integrated into a user-friendly device that individuals can easily use.

Moreover, many solutions rely on expensive sensors, increasing the overall cost of the device. Additionally, the majority of current aids do not provide a cohesive approach that combines accurate localization with obstacle detection, and they often lack context-specific, non-visual instructions tailored to the unique needs of visually impaired users. This highlights the urgent need for an affordable, integrated indoor navigation system that enhances mobility and independence in everyday settings.

1.4 Project Objectives

1.5 Project Outline

The remainder of this report is organised as follows: Chapter 2 provides the Background and the related works. Chapter 3 provides the project system design, and Chapter 4 covers the implementation. Chapter 5 provides the results obtained by the system, and Chapter 6 concludes the report.[1]

[1]Next Term. But we can mention it from now.

Chapter 2

Background

2.1 Introduction

This chapter is organized as follows: Section 2 discusses the role of landmarks in navigation, emphasizing artificial landmarks such as QR codes. Section 3 explores the structure, versions, and encoding of QR codes, while Section 4 focuses on QR code-based localization systems. Section 5 covers object detection, followed by Section 6, which addresses customizable guidance systems. Finally, Section 7 reviews related work in the field.

2.2 Landmarks (Natural & Artificial)

Landmarks can be categorized as either natural or artificial. Natural landmarks are formed by nature, such as mountains, rocks, trees, or any other natural formations. While on the other hand, artificial landmarks are human-made structures such as traffic signs, statues, and QR codes. both categorizes can be used in various important fields in computer science. For example, computer vision algorithms leverage landmarks by detecting, identifying, and tracking them to determine the precise location of an object, create a 3D map of the surrounding environment, or to monitor the state of an object.

While both categorizes can be very useful, natural landmarks are much harder to recognize, very diversified and do not have uniform shapes, difficult to customize, and they do not encode data. All of these characteristics can be opposite in the artificial landmarks, which make them better in terms of control, precision, reliability, and adaptability. Also there are a lot of libraries that support encoding, detecting, decoding, and tracking artificial landmarks. There are also a wide variety of artificial markers such as QR, ArUco, Topotag. More information on the other kinds of markers are present on reference [1].

2.3 QR code

Quick Response (QR) codes are a type of matrix barcode designed for efficient data encoding and rapid scanning. QR Codes first developed in 1994 by Denso Wave, to track automotive parts in manufacturing. but have since become widely used across various applications due to their versatility, simplicity, and capacity to store significant amounts of information compared to traditional barcodes.

2.3.1 QR Structure

QR Codes can be detected and decoded at various angles and distortion levels due to their unique structure. The code consists of black modules (squares) arranged on a white background in a grid pattern, allowing for rapid and error-resistant scanning. All QR Codes have standard structure as shown in Figure 2.1. This structure is made out of the following parts:

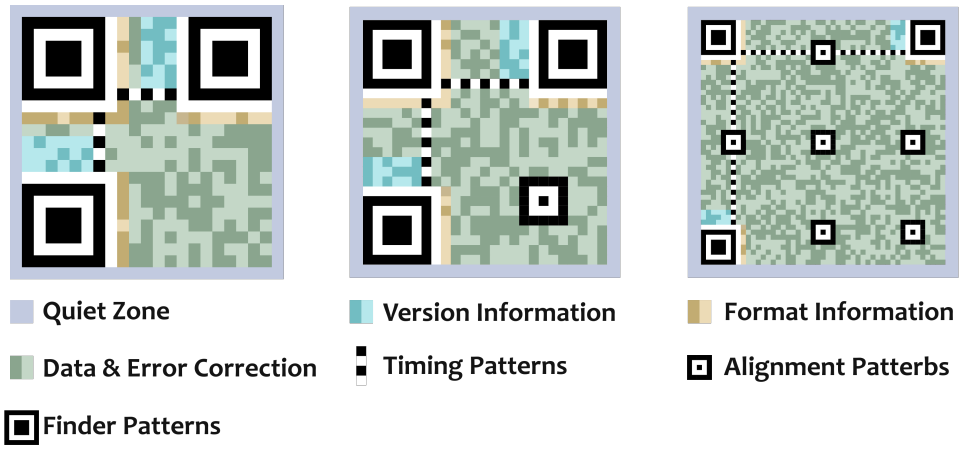

Figure 2.1: These are three QR Codes that store texts. The texts' lengths get larger going from the left to the right. Notice that the alignment pattern only appears at the bigger QR Codes, and there are multiple ones at the biggest QR Code.

- **Quiet Zone:** This is a white empty area surrounds the QR Code that helps distinguishing it from its surroundings.
- **Version Information:** QR Codes have different versions/sizes which specified by these two areas.
- **Format Information:** This part provides details about the error correction level and data mask pattern used.
- **Data & Error Correction:** This is where both encoded data & error correction are. data and error correction are stored together enabling the QR code to recover and reconstruct the stored data, even if up to 30% of the code is damaged or obscured. Data have different types, such as text, URLs, or other.
- **Timing Patterns:** This part is essential for defining the grid's structure and assists the scanner in establishing the size and coordinate system of the code.

- **Alignment Patterns:** The Alignment Patterns help correct distortion and skewing of the QR code when viewed from different angles. It is especially crucial for larger QR codes that may be prone to bending or misalignment.
- **Finder Patterns:** These patterns assist the scanning device in rapidly locating and orienting the QR code, regardless of its rotation or angle.

See [15], for more information.

2.3.2 QR Code Versions and Types

QR codes come in 40 versions, each representing a different size and data capacity. Version 1 contains 21×21 modules, while Version 40 has 177×177 modules. As the version number increases, so does the data capacity and the complexity of the code, making it capable of storing more information or supporting higher levels of error correction [15].

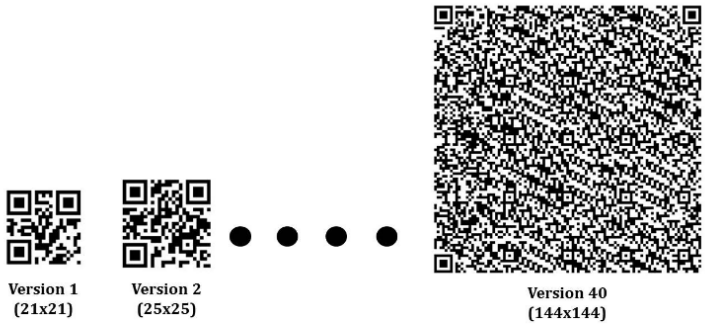

Figure 2.2: QR codes Versions (adapted from [15]).

There are also specialized QR code types designed for specific applications:

- **QR Code Model 1 and 2**: Model 1 is the original version of the QR code, developed in 1994 by Denso Wave. While model 2 is an improved version of model 1, and it is the one most commonly used today. These models are used for daily basis.
- **Micro QR Code**: Designed to be smaller and simpler, it uses only one Finder Pattern, making it more compact than traditional QR codes. It is often used when space is limited [15].
- **Logo QR Code**: Allows logos or images to be embedded within the QR code, enhancing the code's visual appeal for marketing or branding purposes [15].

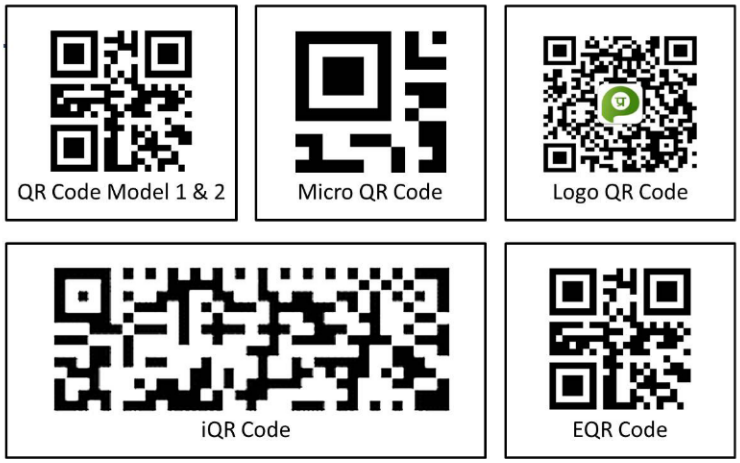

Figure 2.3: QR code types (adapted from [15]).

- **iQR Code**: A flexible matrix-type QR code capable of being printed in various sizes and configurations, from small, high-capacity codes to large codes. It can store more data than standard QR codes and can be inverted or turned into dot patterns for direct part marking [15].
- **Encrypted QR Code**: Uses encryption techniques to secure the information encoded within, making it suitable for applications requiring data confidentiality [15].

2.3.3 QR Code Encoding

The encoding process for a QR code involves converting data into a matrix of black and white modules. First, the input data is analyzed to determine the appropriate encoding mode, such as numeric, alphanumeric, byte, or Kanji. The data is then transformed into binary code, and error correction is added using Reed-Solomon algorithms, ensuring that the code can still be scanned if partially damaged [15].

Popular online tools for generating QR codes include:

- **Online Tools**: These free online tools offer simple and fast solutions for both generating and decoding QR codes. QRickit allows users to decode QR codes from uploaded images, while GOQR.me provides various output formats, such as PNG, SVG, and EPS, making it versatile for different use cases [17][18].

For generating QR codes programmatically, popular libraries include:

- **Python - PyQRCode**: A Python library that simplifies QR code creation, allowing output in SVG, PNG, and other formats [19].

- **Java - ZXing (Zebra Crossing)**: An open-source library widely used for encoding QR codes in Java and Android applications [20].
- **iOS - Core Image**: iOS provides native QR code generation capabilities via the ‘CIQRCodeGenerator‘ filter in the Core Image framework [21].

2.3.4 QR Code Decoding

Decoding a QR code starts by scanning the Finder Patterns, which allow the scanner to properly align the code. Next, the Format Information is read to apply the correct error correction and mask pattern. Once decoded, the binary data is translated back into the original format (e.g., text, URL) [15].

Common online tools for decoding QR codes include:

- **Online Tools**: These tools offer versatile QR code decoding solutions across platforms. ZXing is an open-source decoder integrated into Android and web applications, QRickit provides online QR code decoding from uploaded images, and Google Lens allows users to scan and decode QR codes directly via smartphone cameras [20, 25].

For decoding programmatically, useful libraries include:

- **Python - Segno**: A versatile Python library for generating and reading both QR and Micro QR codes [22].
- **Java - ZXing**: Provides decoding functionality along with encoding, and is widely used for mobile and web apps [20].
- **iOS - AVFoundation**: iOS provides native support for scanning QR codes using the ‘AVCaptureMetadataOutput‘ class in the AVFoundation framework [23].
- **Google ML Kit**: temporarily empty...

2.4 Indoor Localization

Indoor localization refers to the process of determining the precise position and orientation of an object or individual within indoor environments, where traditional Global Positioning System (GPS) signals are often unreliable. This technology is critical for a wide range of applications, including navigation within large buildings and asset tracking in warehouses.

Various technologies are employed for indoor localization, each with distinct advantages and limitations. Wi-Fi, Bluetooth, Radio-Frequency Identification (RFID), and Ultra-Wideband (UWB) are among the most widely used methods. Some systems rely on complex and expensive hardware, such as sensors and Inertial Measurement Units (IMUs), while others utilize more cost-effective solutions, including Bluetooth beacons and pose estimation using Quick Response (QR) codes. The selection of the appropriate technology depends on the specific requirements of the application, taking into account factors such as accuracy, cost, and ease of implementation [16].

As the field of indoor localization continues to evolve, new techniques and innovative approaches are emerging, such as the use of QR codes for precise positioning and tracking.

2.4.1 Indoor Localization with QR Codes

Indoor localization using QR codes enables position determination by detecting QR codes strategically placed throughout an indoor space. These codes can be affixed to floors, walls, ceilings, or suspended on hanging panels, each containing encoded positional information.

2.4.1.1 QR Code Placement

The placement of QR codes plays a critical role in ensuring effective indoor localization. Codes can be positioned in various locations:

- **Ceilings**: QR codes on ceilings are often out of the way and can provide an unobstructed view for overhead cameras, such as those mounted on hats or handheld devices. This placement is ideal for applications where the user's line of sight remains upward.
- **Walls**: QR codes can also be positioned on walls at different heights to accommodate various camera angles. This setup is particularly useful when the user or device is at eye level with the code.
- **Floors**: Placing QR codes on floors can be beneficial in environments where overhead cameras or downward-facing sensors (such as those on a robot or mobility aid) are used. However, codes on floors might be more prone to wear and tear and may require periodic maintenance.
- **Hanging Panels**: In environments where flexibility is needed, QR codes can be placed on hanging panels suspended from the ceiling. This allows

for better visibility while keeping the codes elevated from foot traffic or other obstacles.

2.4.1.2 Data Encoding in QR Codes

Each QR code encodes essential information for localization purposes, including:

- **Coordinates**: The most important data that QR codes can encode are the precise coordinates of their position within the environment. These coordinates are predefined and allow the system to accurately calculate the user's or object's location when the QR code is detected. The coordinates might be in terms of x, y, z positions or based on a grid system specific to the environment.
- **Unique Identifier (ID)**: In addition to coordinates, each QR code will have a unique ID that differentiates it from others in the system. The ID can be referenced in the system's database to retrieve additional information, such as the room name or floor level.
- **Orientation Data**: QR codes can also encode information about their orientation, which helps in determining the user's orientation in relation to the environment (e.g., the angle of the code in reference to a global axis).
- **Additional Metadata**: If needed, QR codes can encode further metadata, such as room names, nearby landmarks, or points of interest. This is especially useful in environments where additional contextual information enhances the user's navigation experience.

2.4.1.3 Approaches to Localization with QR Codes

Several methods exist for implementing indoor localization using QR codes, each with its own benefits and trade-offs. Two common approaches include the divided environment method, which offers simplicity and low computational requirements, and the pose estimation method, which provides higher precision but at the cost of increased computational complexity. These approaches are discussed below.

The Divided Environment Method In the divided environment method, the space is partitioned into distinct pieces(e.g., squares, triangles, or hexagons), with each piece containing a QR code encoding its position. QR codes can be placed on the floor, ceiling, walls, or hanging panels depending on the system's configuration. A similar approach has been demonstrated by Zhang et al. [3].

To illustrate this method, consider a room measuring 4 meters in both width and height, divided into 16 equal squares, each with an area of $1m^2$. If we customized a hat for example, that embeds a camera in its top, if we put the QR codes at the ceiling, then the user's position will get determined while navigating in the room wearing the hat.

Although this method is computationally efficient, it has limitations. The position values are discrete, which may not provide the continuous and precise localization required in certain applications. Nonetheless, this method can be highly effective for specific use cases, as discussed in Appendix A.

Pose Estimation Method Another important and interesting way to calculate the precise and continues values of the user's pose(position & orientation) is by detecting and decoding a QR Code using a camera to gain its global pose, then calculate its relative pose to the camera. After that, we will be able to use these two poses to estimate the user's precise pose as follows:

$$user\_global\_pose = QR\_global\_pose - QR\_relative\_pose$$

See [7], where they used this method(but different implementation) for their localization system.

But calculating the relative pose is not a straight forward process. First, let us explain how cameras work. The basic idea of cameras is to project the real world 3D points into a 2D image, see Figure 2.4 for illustration. The following equation describes the relation between the real world points and the image points:

$$x = PX \tag{2.1}$$

Where:

- $\mathbf{x} = [\mathbf{u},\ \mathbf{v},\ \mathbf{1}]^T$ is the homogeneous coordinate of the 2D point in the image plane.
- $\mathbf{X} = [\mathbf{x}_w,\ \mathbf{y}_w,\ \mathbf{z}_w,\ \mathbf{1}]^T$ is the homogeneous coordinate of the 3D point in the world coordinate system.
- $\mathbf{P} = \mathbf{K}[R|t]$ is the camera projection matrix.

  With:

  - $[R|t]$ is the camera's extrinsic matrix, which "is a transformation matrix from the world coordinate system to the camera coordinate system" - Aqeel Anwar [11].

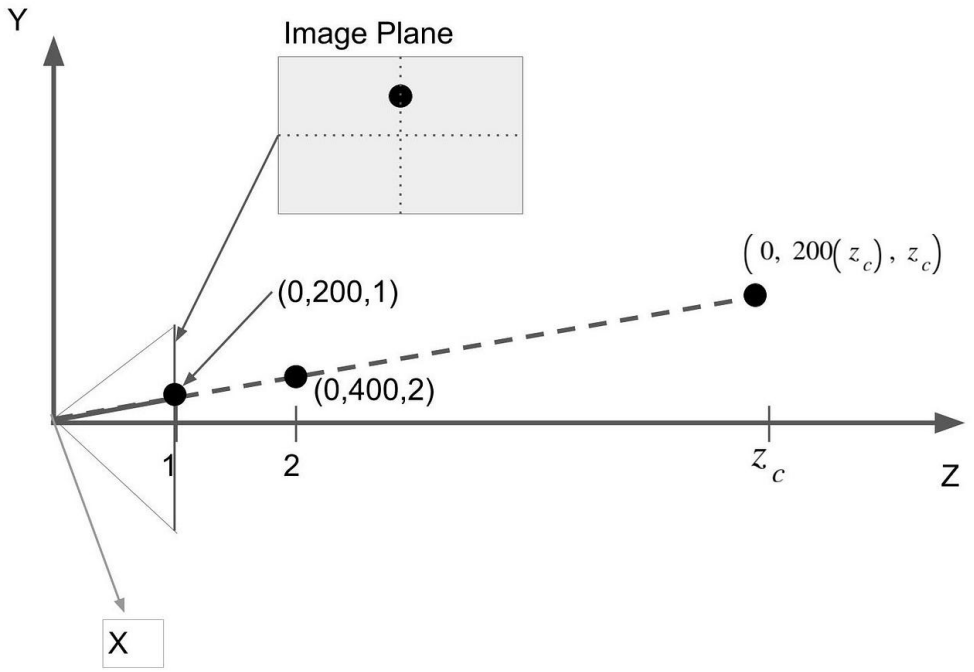

Figure 2.4: https://eugene-chian.medium.com/a-single-camera-3d-functions-fdec7ffa9a83

R is a 3x3 matrix that represents the rotation of the object's with respect to the camera. t is 3x1 vector describes the object's position relative to the camera.

$$[R|t] = \begin{bmatrix} R_{3\times3} & t_{3\times1} \end{bmatrix}_{(3\times4)} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

– **K** is the camera's intrinsic matrix, which "is a transformation matrix that converts points from the camera coordinate system to the pixel coordinate system" - Aqeel Anwar [11]. It contains camera's focal length(f_x, f_y) and the principal point(c_x, c_y) as follows:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Notice how the camera's intrinsic matrix cares only about the camera's internal parameters, while the extrinsic matrix does not.

Now we can rewrite equation 2.1 as follows:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{2.2}$$

where s is a scale factor due to the homogeneous coordinates.

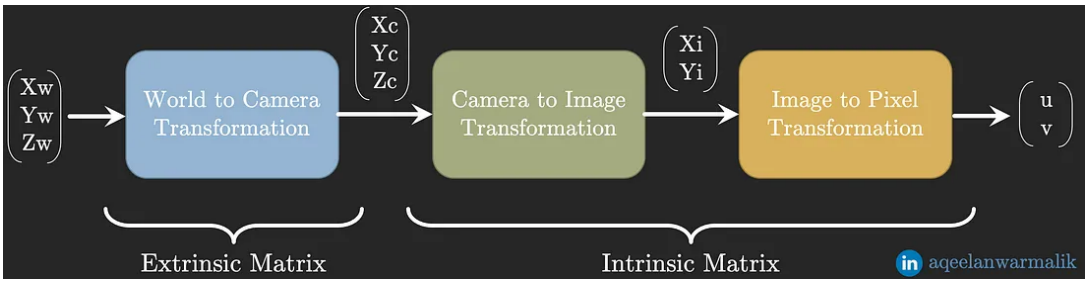

Figure 2.5: Simple illustration for the camera's projection matrix made by Aqeel Anwar [11].1

After we understood the basics of how cameras project 3D world points into 2D images, we can use this knowledge to calculate the pose of an object, such as a QR code. At least four 3D points at the object need to be known, such as the four corners of a QR code in the world coordinate system. Then, the corresponding 2D projections in the image plane of these points need to be known. See figure 2.6 for illustration.

Figure 2.6: This is a picture of a QR code that has the width/height of 16.8cm and its four corners points in the image coordinate system in pixels are known as it appears.

For more illustration, let us take the QR code at figure 2.6 as an example. The QR code's width/height is equal to 16.8cm in the real world coordinate system. Using this value, we will be able to extract the four corners 3D points in the real world coordinate system as follows:

$$\begin{gathered} top-left-point = (-16.8/2, 16.8/2, 0) = (-8.4, 8.4, 0) \\ top-right-point = (8.4, 8.4, 0) \\ bottom-right-point = (-8.4, 8.4, 0) \\ bottom-left-point = (-8.4, -8.4, 0) \end{gathered} \tag{2.3}$$

For the sake of simplicity, the values at 2.3 are for a QR code with a global position value of (0,0,0).

After choosing some pairs of 3D points with their corresponding 2D projections, use equation 2.2 for each pair as follows:

$$s \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_{w,i} \\ y_{w,i} \\ z_{w,i} \\ 1 \end{bmatrix}$$

This yields two equations per point:

$$u_i = \frac{f_x(r_{11}x_{w,i} + r_{12}y_{w,i} + r_{13}z_{w,i} + t_x)}{r_{31}x_{w,i} + r_{32}y_{w,i} + r_{33}z_{w,i} + t_z} + c_x$$

$$v_i = \frac{f_y(r_{21}x_{w,i} + r_{22}y_{w,i} + r_{23}z_{w,i} + t_y)}{r_{31}x_{w,i} + r_{32}y_{w,i} + r_{33}z_{w,i} + t_z} + c_y$$

Finally, use the later equations to solve for R and t by using one of the PnP(Perspective-n-Point) algorithms. At the end, we will have the values of R and t which is the camera's extrinsic matrix that describes an object's rotation and translation relative to the camera. So, this is how to get the relative pose.

PnP PnP is a classic approach in computer vision for determining the pose of a calibrated camera with respect to a set of known 3D points in the real world coordinate system and their corresponding 2D projection points. So in a nutshell a PnP algorithm takes three inputs: 3D points, 2D projection points, and intrinsic matrix to solve the camera's pose. There are several different PnP algorithms such as Direct Linear Transformation(DLT), Efficient PnP(EPnP), and RANSAC-PnP.

Camera Calibration Camera Calibration is the process of calculating the camera's focal length, principal point, and distortion coefficients. The basic

idea behind camera calibration is to take multiple photos by the camera to a known pattern, then try to accurately locate some points at these photos. Then compare these estimated points to their physical locations of points in the calibration pattern (usually on a flat surface) in the real-world coordinate system. By comparing the 2D points in the images to their known 3D locations in the real-world coordinate system, we will be able to calculate accurate values of the camera's focal length, principal point, and distortion coefficients.

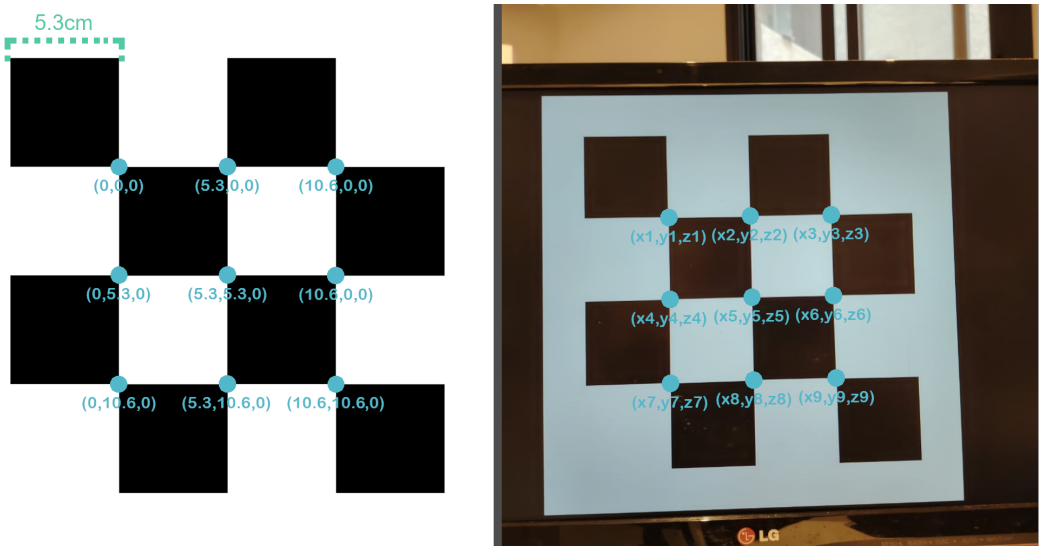

Figure 2.7: The left image represents the calibration pattern's real world inner points locations. While on the other hand, the right image shows the estimated inner points locations.

Libraries:

- **Camera Libraries**: There are several libraries for cameras in Android devices, such as Camera2, and CameraX. It is recommended for most developers to use CameraX library, since it supports the vast majority of Android devices, and provides a simple and easy to use API - unlike Camera2 - that spares users from dealing with compatibility issues and other low level details. See [14] for more info.

- **Computer Vision Libraries**: temporarily empty...

While the pose estimation approach offers significantly higher accuracy, it requires greater computational resources due to additional processes, such as camera calibration to determine intrinsic parameters. This increased complexity makes the method more resource-intensive compared to the divided environment approach. However, it is suitable for applications where continuous and precise localization is essential.

2.5 Application Programming Interface

An Application Programming Interface (API) is a set of rules that allows different software applications to communicate with each other. It defines the methods and data formats that applications can use to request and exchange information, simplifying the process for developers to build compatible software. For instance, when using a social media login to access a new app, an API handles the authentication process.

APIs are essential in various technology domains, including web and mobile applications, where they enable diverse services to work together seamlessly. For example, a travel booking website may utilize APIs to aggregate data from various airline and hotel services, allowing users to compare prices and availability in one place. Similarly, a weather app may leverage APIs to retrieve real-time weather data from different sources, providing accurate forecasts based on user location [5].

Base URL: The base URL is the starting point for accessing an API. It usually includes the protocol (like HTTP or HTTPS) and the domain name where the API is hosted. For example, if the base URL is `https://api.example.com`, all API requests will start from this address.

Endpoints: Endpoints are specific paths added to the base URL that direct requests to particular resources or functions within the API. For example, if you want to access user data, you might use an endpoint like `/users`. The full URL for this request would be `https://api.example.com/users`.

HTTP Methods: APIs commonly use HTTP methods to specify the action to be performed. The most common methods are:

- `GET`: Retrieve data from the server.
- `POST`: Send new data to the server.
- `PUT`: Update existing data on the server.
- `DELETE`: Remove data from the server.

- ADD frameworks example, including what will be used in the next semester.

2.6 Databases

A **database** is an organized system for storing, retrieving, and managing data. It enables efficient data handling, storage, and access for various applications,

from small personal projects to large-scale enterprise solutions. Databases are essential for maintaining consistency, accuracy, and accessibility in data storage. They use structures such as tables to organize data into columns (fields) and rows (records). Two common key elements in databases are **Primary Keys (PK)** and **Foreign Keys (FK)**. The Primary Key uniquely identifies each record within a table, ensuring that data remains unique, while the Foreign Key links tables, establishing relationships between them to enforce data integrity.

There are various types of databases, including:

- **Relational Databases** – Organize data into tables and use structured query language (SQL) for data manipulation. Examples include MySQL, PostgreSQL, and SQLite.
- **NoSQL Databases** – Store unstructured data, often in formats like JSON. They are designed for scalability and high performance in handling large datasets, as in MongoDB and Cassandra.

2.6.1 Entity-Relationship Diagram (ERD)

The **Entity-Relationship Diagram (ERD)** is a visual representation of the data model for a system. It illustrates the entities involved, their attributes, and the relationships between them. Each entity corresponds to a table in the database, while attributes represent the fields within those tables. Relationships indicate how entities interact, showcasing cardinality (e.g., one-to-many or many-to-many) and participation (e.g., optional or mandatory). The ERD serves as a crucial tool in the design phase, promoting clear communication about the data structure and its constraints.

2.6.2 Structured Query Language (SQL)

SQL (Structured Query Language) is the standard language used to interact with relational databases. SQL provides commands for data insertion, query, updating, and deletion. It also includes commands for schema creation and modification. Some fundamental SQL commands include:

- `SELECT` for retrieving data,
- `INSERT` for adding new data,
- `UPDATE` for modifying existing data,
- `DELETE` for removing data,

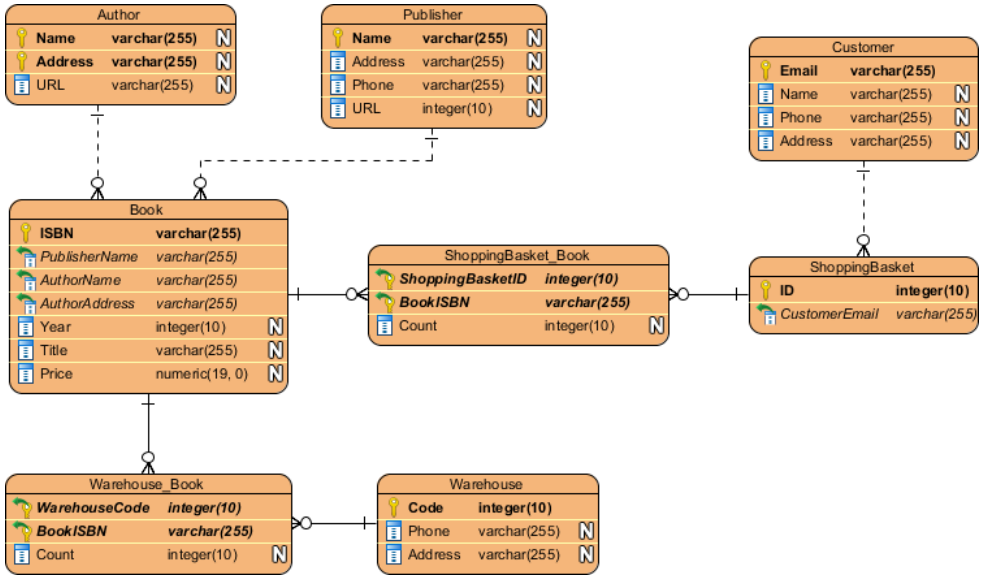

Figure 2.8: This ERD, adapted from [6], represents a bookstore database, with entities like Author, Publisher, Book, Warehouse, Customer, and ShoppingBasket. It shows relationships between entities, such as customers adding books to shopping baskets and warehouses storing book inventory, linked by primary and foreign keys.

- `CREATE TABLE` for defining new tables,
- `JOIN` for combining data from multiple tables.

SQL enables efficient, standardized access to data, supporting complex queries and data manipulation tasks.

2.6.3 SQLite3

SQLite3 is a lightweight, serverless, and self-contained relational database engine, commonly used for applications requiring minimal configuration and overhead. Unlike other SQL databases, SQLite does not require a separate server process. Instead, it stores data in a single file, making it highly portable and simple to integrate. SQLite is ideal for embedded systems, mobile apps, and small to medium-scale projects due to its reliability, compact size, and ease of use.

2.7 Customizable Guidance

2.8 Objects Detection

2.9 Related Work

Navigation solutions for visually impaired individuals utilize a range of technologies, including Bluetooth Low Energy (BLE), LiDAR, and artificial landmarks. Each of these technologies offers unique advantages and limitations that impact their effectiveness in real-world applications. In this section, we explore different solutions that incorporate various technological approaches, highlighting how each addresses the challenges faced by visually impaired users in navigating indoor environments.

PathFinder, proposed by Kuribayashi et al. [12], is a map-less navigation system developed to assist blind individuals in navigating unfamiliar indoor environments. Unlike other systems that rely on prebuilt maps, PathFinder uses real-time detection of intersections and signs to guide users. The system includes a suitcase-shaped robot that blind users control through a handle interface, receiving audio feedback about their surroundings. PathFinder's core navigation capabilities are powered by LiDAR for detecting intersections and image processing via a high-resolution camera to recognize directional and textual signs. Developed using a participatory design approach with blind users, the system focuses on addressing the most relevant challenges of navigating unknown spaces. In a user study involving seven blind participants, PathFinder demonstrated significant effectiveness in helping users navigate unfamiliar environments with greater confidence than when using traditional aids like canes or guide dogs. Despite requiring more user effort than map-based systems, participants appreciated PathFinder's flexibility and adaptability to environments without prior mapping. The study concluded that PathFinder offers a valuable solution for blind individuals, particularly in situations where prebuilt maps are unavailable.

Ahmetovic et al. [4] presents NavCog, NavCog is a smartphone-based navigation system designed to help visually impaired individuals with turn-by-turn guidance in indoor and outdoor environments. The system uses Bluetooth Low Energy (BLE) beacons for accurate localization, employing a K-nearest neighbor (KNN) algorithm to compare current signal strengths with previously recorded beacon signal fingerprints. NavCog provides real-time auditory navigation instructions to guide users along a predetermined path and also notifies them of nearby points of interest (POI) and accessibility issues such as stairs or obstacles.

One of NavCog's key features is its ability to be easily deployed in large, com-

plex environments without requiring significant infrastructure modifications. In a study conducted with six visually impaired participants on a university campus, NavCog successfully guided users through both familiar and unfamiliar spaces, receiving positive feedback on its navigation features. The study emphasizes NavCog's flexibility and ease of use, making it a promising tool for visually impaired individuals navigating new environments.

The work by Fraga et al. [13] presents an indoor navigation system has been developed for visually impaired individuals. This system uses QR code markers and computer vision techniques to provide real-time navigation instructions. Users scan QR codes placed in different locations to receive guidance along optimal paths. The system cross-references the scanned information with a prebuilt database. Audio feedback informs users of their current location and the next steps in the navigation process. If the user deviates from the planned route, the system recalculates the path. Importantly, this system does not require internet connectivity, making it practical for offline use. Additionally, the system includes a collision avoidance feature based on a monocular depth estimation algorithm, which predicts distances to obstacles using 2D images. Experimental results in a controlled environment demonstrated that the system accurately guided users and effectively detected obstacles with acceptable depth estimation margins. This study highlights the potential of QR code-based navigation for enhancing mobility for visually impaired individuals and suggests future integration into smartphones or AI accelerators for improved usability.

In conclusion, our solution distinguishes itself by employing QR codes as a cost-effective navigation tool that requires only a standard smartphone camera. This straightforward approach minimizes the need for additional hardware, thereby increasing accessibility for users. Unlike systems such as PathFinder and NavCog, which depend on specialized technologies like LiDAR and Bluetooth beacons, our method aims to simplify navigation by leveraging existing smartphone capabilities. Furthermore, we intend to integrate optional remote or Wi-Fi cameras to enhance the navigation experience without imposing significant costs. Our primary objective is to deliver a practical and user-friendly solution specifically tailored for visually impaired individuals, ensuring ease of use in real-world settings through thorough testing in various indoor environments.

Chapter 3

Methodology

3.1 System Architecture Overview

In this section, you can present the overall structure of your indoor navigation system and how its components interact. It should include:

- **High-level view of components:** Briefly describe the components of the system, such as the user interface, QR code scanner, navigation logic, backend/database, and the device (mobile app).
- **Data flow:** Explain how data moves between components. For instance, when a QR code is scanned, how the information is sent to the navigation system and then displayed to the user.
- **Technology Stack:** List the technologies and tools you are using for each component, such as the development framework (e.g., Flutter for a mobile app), libraries for QR code scanning, and the backend services (e.g., Firebase for storing data).
- **System diagram:** Include a diagram illustrating how different parts of the system interact.

3.2 Database System

The database system serves as the foundational framework for managing data storage, processing requests, delivering instructions in response to QR code scans, and it provides GUI tools for managing the database. It is essential for the functionality of various components, ensuring accurate and timely responses to user interactions. For instance, the localization system relies on the database to retrieve from it a QR code's global positions, while the customizable guidance system utilizes it to retrieve the latest instructions tailored for users. Several parts of the project depends on the database to keep data accurate and up-to-date. The database system is composed of the following components:

3.2.1 Database

The database serves as the backbone of the system, storing all data related to QR codes, building layouts, and navigation instructions. It is designed to organize QR code data in a manner that ensures each QR code is uniquely identifiable, allowing the mobile application to efficiently retrieve the correct information upon each scan.

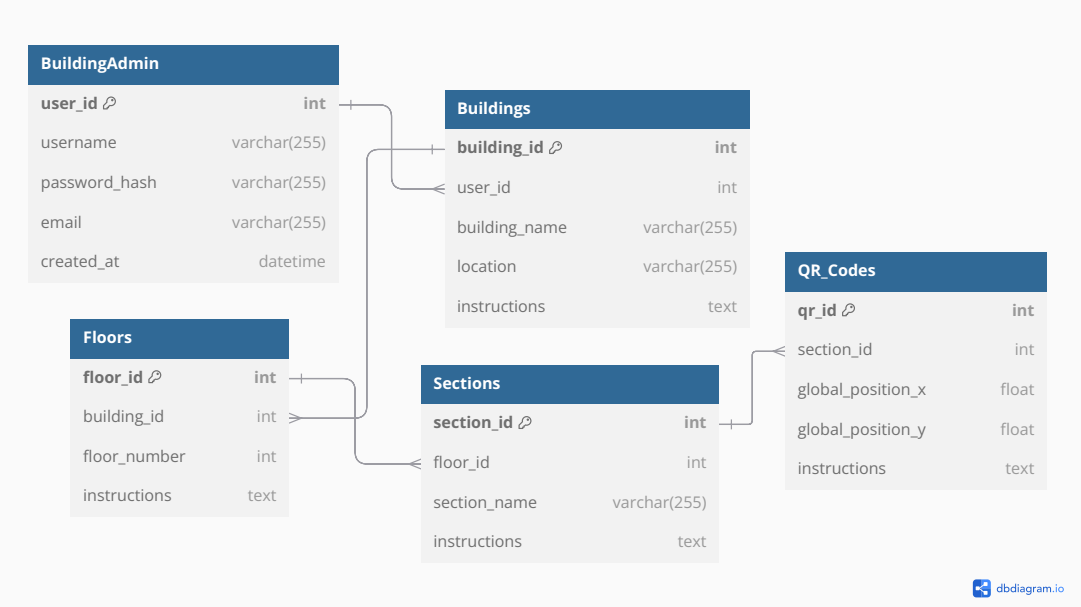

Figure 3.1: This Entity-Relationship Diagram (ERD) of Mosaned database

Figure 3.1 illustrates our database structure which is composed out of 5 main tables as follows:

- **BuildingAdmin:**
  - **user_id (Primary Key)**: A unique identifier for each user of the system, enabling individual management of buildings and associated data.
  - **username**: The name chosen by the user for login purposes.
  - **password_hash**: A secure hash of the user's password used for authentication.
  - **email**: The user's email address, utilized for notifications and account recovery.
  - **created_at**: A timestamp indicating when the user account was created.
- **Buildings:**
  - **building_id (Primary Key)**: A unique identifier for each building within the system.
  - **user_id (Foreign Key)**: Links the building to the associated user in the `BuildingAdmin` table, ensuring that each building is managed by the correct user.
  - **building_name**: The name of the building.

  - **location**: The physical address or description of the building's location.
  - **instructions**: General instructions related to the building, providing contextual information for navigation and management.

- **Floors:**
  - **floor_id (Primary Key)**: A unique identifier for each floor within a building.
  - **building_id (Foreign Key)**: Associates the floor with its corresponding building, establishing a clear hierarchical structure.
  - **floor_number**: Indicates the number of the floor (e.g., 1 for the first floor, 2 for the second, etc.).
  - **instructions**: Specific instructions related to that floor, assisting users with effective navigation.

- **Sections:**
  - **section_id (Primary Key)**: A unique identifier for each section on a floor.
  - **floor_id (Foreign Key)**: Links the section to its respective floor, maintaining the organizational hierarchy.
  - **section_name**: The name or description of the section within the floor.
  - **instructions**: Instructions specific to the section, enhancing the user's ability to navigate the area.

- **QR_Codes:**
  - **qr_id (Primary Key)**: A unique identifier for each QR code, ensuring no duplicates exist within the system.
  - **section_id (Foreign Key)**: Connects the QR code to its specific section.
  - **global_position_x**: The X-coordinate representing the global position of the QR code.
  - **global_position_y**: The Y-coordinate indicating the QR code's position, further aiding in navigation.
  - **instructions**: Specific instructions directly linked to the QR code, providing targeted guidance for users when the QR code is scanned.

3.2.2 Application Programming Interface (API)

The API serves as the bridge between the mobile app, the database, and the building management dashboard. Its primary responsibilities are:

- **Data Retrieval and Delivery:** When a QR code is scanned, the API retrieves the associated information from the database and sends it to the mobile app for processing.
- **Instruction Updates:** The API also handles data updates from the dashboard, ensuring the latest QR code instructions are accessible to users in real time.
- **User Authentication:** Manages secure access to both the dashboard and the mobile app, allowing only authenticated users to modify or retrieve data.

- ADD one example of API Endpoints, and the rest on Appendix

3.2.3 Web Server

The Web Server hosts the building management dashboard and processes web requests, allowing administrators to view and manage QR code data and building layouts. It communicates with the API to send updates to the database and deliver real-time data to the mobile app.

- ADD What tools/framework used to host the dashboard on 3.6

3.2.4 Building Management Dashboard

The Building Management Dashboard is a web-based tool that allows building administrators to manage the indoor navigation system effectively. This dashboard provides an interface to control and update QR code instructions and manage QR codes within the building remotely.

Key features include:

- **User Authentication:** Allows administrators to securely log in and access only their managed buildings.

- **Building and Floor Management:** Displays a list of buildings and their respective floors, allowing administrators to organize and view all QR codes assigned to each area.
- **QR Code Instruction Editing:** Administrators can add, update, or delete instructions for each QR code in the building, tailoring guidance to specific areas and ensuring data accuracy.

- Web Dashboard interfaces to be ADDED

The dashboard enables efficient oversight and maintenance of QR code-based navigation, allowing real-time updates without modifying the physical QR codes.

3.3 Mobile Application

The Mobile Application serves as the primary interface for visually impaired users, offering intuitive features that facilitate navigation in indoor environments. Designed with accessibility in mind, the app implements both the customizable guidance and localization systems, ensuring that users receive accurate and real-time navigation assistance.

The application functionality includes:

- **QR Code Scanning:** The app's camera detects and decodes QR codes, automatically initiating the customizable guidance system. Upon scanning a QR code, the application identifies its unique ID, which is crucial for retrieving the corresponding navigation instructions from the central database.
- **Customizable Guidance Imp.:**
- **Localization Imp.:**
- **User-Friendly Interface:** The app features an easy-to-navigate interface that simplifies the interaction for visually impaired users. It allows them to focus on their surroundings while receiving timely navigation instructions through audio feedback and alerts.

  – Mobile-APP interfaces to be ADDED

This design ensures that visually impaired users have access to accurate, location-specific guidance without the need for manual data input or app configuration except for the one-time camera calibration process. The seamless integration of the customizable guidance and localization systems within the mobile application empowers users to navigate indoor environments confidently and independently.

3.4 Localization System

The basic idea behind our localization system implementation is to capture a frame at real time by a camera, then detect and decode the QR code in it. After that, we will be able to restore the QR code's global position from the server using the decoded data, which is described at...(write where) . Now, we only need to calculate the camera's position relative to the QR code, and then add the result together with the QR code's global position as follows:

$$user\_global\_position = QR\_global\_position + camera\_relative\_position$$

This is the pose estimation approach which is mentioned previously at section 2.4.1.3.

As we just mentioned, the QR code's global position is stored at a server so there is nothing to calculate here, but this is not the case of the camera's relative position. The process of calculating the later is somewhat complicated and not short, and there are several things that need to be calculated first, such as the QR code corners locations at the image frame, camera matrix, and the camera distortion coefficients. There are multiple libraries for localization purposes as we mentioned previously at 2.4.1.3, and we choose OpenCV due to its Android devices support, ease of use, simple API, and performance.

Everything related to the localization will be running at a separate thread from the main thread. This enables the System to perform other tasks simultaneously and independently.

3.4.1 Camera Calibration

We mentioned at...(write where) that users can chose the camera they want to use from the settings after connecting it to the phone. This approach rise the need of calibration system that users can use since the cameras' parameters are distinct and not known. Each camera need to be calibrated at least once at the first time. Then the camera parameters will be stored at the phone storage for reusing in the future. At first, users need to specify the number of rows and

columns at the pattern and the width/height of each square in real world units. Then they need to take multiple photos to the pattern at different angles and distances and start the calibration process. After this point, everything else will be done automatically without the need of the users actions no more.

The camera calibration system's implementation is the same as what was mentioned previously at section 2.4.1.3. First the program will iterate through each photo trying to estimate their inner corners points locations using the following OpenCV function:

```
boolean findChessboardCorners(
Mat image,
Size patternSize,
MatOfPoint2f corners
)
```

Params:

- **image:** The current photo of the pattern.
- **patternSize:** Number of inner corners per a chessboard row and column.
- **corners:** Output array of detected corners.

After that, we used the OpenCV function cornerSubPix to refine the detected points. Finally, we used the following OpenCV function for calibration:

```
double calibrateCamera(
List<Mat> objectPoints,
List<Mat> imagePoints,
Size imageSize,
Mat cameraMatrix,
Mat distCoeffs,
List<Mat> rvecs,
List<Mat> tvecs
)
```

Params:

- **objectPoints:** List of real world inner points locations for each image.
- **imagePoints:** List of estimated inner points locations for each image.
- **imageSize:** Size of the calibrated pattern photo.

- **cameraMatrix:** Output matrix for the camera intrinsic values.
- **distCoeffs:** Output matrix for the camera distortion coefficients.
- **rvecs:** Output list contains the rotation of the calibration patterns for each image.
- **tvecs:** Output list contains the translation of the calibration patterns for each image.

3.4.2 Camera's Relative Pose

The first step is to capture frames using a camera at real time. We used cameraX to do so since it is the best for Android devices as we mentioned at 2.4.1.3. Then, each captured frame will be passed to a function for QR code detecting, decoding, and pose estimation, so finally we can calculate camera's relative position. For QR code detecting, there are bunch of useful libraries we can for QR code detecting and decoding as we mentioned previously at 2.3.4. We choose Google's ML kit due to its simplicity and good performance. After successfully detecting the QR code, Google's ML kit will calculate the QR code corners locations for us.

Finally, we use OpenCV's solvePnP to calculate the QR pose function as follow:

```
boolean solvePnP(
MatOfPoint3f objectPoints,
MatOfPoint2f imagePoints,
Mat cameraMatrix,
MatOfDouble distCoeffs,
Mat rvec,
Mat tvec
)
```

Params:

- **objectPoints:** The QR code's four corners locations relative to its center in real world units, such as cm, mm, etc.
- **imagePoints:** The locations of the QR code's corners inside the captured frame in pixels.
- **cameraMatrix & distCoeffs:** These two are crucial for calculating the QR code's pose since they represent the intrinsic values of a camera and its distortion coefficients.

- **rvec & tvec:** These are output vectors for the code's rotation(rvec) and translation(tvec) relative to the camera.

imagePoints is calculated after successfully detecting the QR code by Google's ML kit.

Now we got the QR code's translation vector relative to the camera, we just multiply it with -1 to get the camera translation relative to the QR code.

3.5 Customizable Guidance

The Customizable Guidance system provides tailored navigation assistance to visually impaired users, implemented directly within the mobile application. This system enables the mobile app to retrieve specific navigation instructions from a central database upon scanning a QR code, offering personalized guidance to the user. Each QR code corresponds to a unique set of instructions, allowing users to receive context-aware information as they move throughout the building.

The guidance system operates as follows:

- **QR Code Scanning:** When a user scans a QR code, the application identifies the QR code's unique ID.
- **Instruction Retrieval:** Based on the ID, the system fetches the relevant instructions from a remote database.
- **Audio or Visual Feedback:** The retrieved instructions can be provided as audio guidance, but the system can also present information in other forms, depending on user preferences and context.
- **Voice Commands:** Users can interact with the system using voice commands through the mobile application. For instance, they can request notifications for specific scenarios, such as "notify me when someone with a red shirt is nearby."

This approach ensures that users receive accurate and timely navigation information tailored to their environment while also allowing for interactive communication with the system.

Chapter 4

Results and Discussion

4.1 Results

This chapter presents the results obtained from the system.

Scenario	**Expected Result**	**Actual Result**
Scenario 1	Success	Success
Scenario 2	Failure	Success

Table 4.1: Results of Various Scenarios

Table 4.1 shows the results obtained during testing in various scenarios.

Chapter 5

Conclusion and Future Work

5.1 Conclusion and Future Work

The project concludes by summarizing the findings and suggesting future work.

References

[1] Michail K., Brennan C., Sabrina C., Anand A., Camden W., & Nikolaos V. (2021). Fiducial Markers for Pose Estimation. Journal of Intelligent & Robotic Systems (2021) 101:71. https://link.springer.com/article/10.1007/s10846-020-01307-9

[2] Tiwari, V., "QR Code Technology and Its Applications," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 6, no. 4, pp. 123-127, 2016.

[3] H. Zhang, C. Zhang, W. Yang and C. -Y. Chen, "Localization and navigation using QR code for mobile robot in indoor environment," 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, China, 2015, pp. 2501-2506, doi: 10.1109/ROBIO.2015.7419715.

[4] D. Ahmetovic, C. Gleason, C. Ruan, K. M. Kitani, H. Takagi and C. Asakawa, "NavCog: a navigational cognitive assistant for the blind," *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, Florence, Italy, 2016, pp. 90-99, doi: 10.1145/2935334.2935361.

[5] IBM, "What Is an API (Application Programming Interface)?," *IBM Topics*, 2024. [Online]. Available: https://www.ibm.com/topics/api. [Accessed: 28-Oct-2024].

[6] Visual Paradigm, "What Is Entity Relationship Diagram?," *Visual Paradigm*, 2024. [Online]. Available: https://www.visual-

paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/. [Accessed: 28-Oct-2024].

[7] Luca C., Gionata C., Francesco F., Alessandro F., Gianluca I., Andrea M. Federica V. (2017). A QR-code Localization System for Mobile Robots: Application to Smart Wheelchairs. https://ieeexplore.ieee.org/document/8098667

[8] J.-I. Kim, H.-S. Gang, J.-Y. Pyun, and G.-R. Kwon, "Implementation of QR Code Recognition Technology Using Smartphone Camera for Indoor Positioning," Energies, vol. 14, no. 10, p. 2759, 2021, doi: 10.3390/EN14102759.

[9] S. -J. Lee, G. Tewolde, J. Lim and J. Kwon, "QR-code based Localization for Indoor Mobile Robot with validation using a 3D optical tracking instrument," 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Busan, Korea (South), 2015, pp. 965-970, doi: 10.1109/AIM.2015.7222664.

[10] M. Kalaitzakis, B. Cain, S. Carroll, A. Ambrosi, C. Whitehead, and N. Vitzilaios, "Fiducial Markers for Pose Estimation: Overview, Applications, and Experimental Comparison of the ARTag, AprilTag, ArUco, and STag Markers," Journal of Intelligent & Robotic Systems, vol. 101, no. 71, pp. 1-26, 2021, doi: 10.1007/s10846-020-01307-9.

[11] Aqeel Anwar, "What are Intrinsic and Extrinsic Camera Parameters in Computer Vision?", A detailed explanation of intrinsic and extrinsic camera parameters with the help of visualizations. https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec

[12] M. Kuribayashi, T. Ishihara, D. Sato, J. Vongkulbhisal, K. Ram, S. Kayukawa, H. Takagi, S. Morishima and C. Asakawa, "PathFinder: Designing a Map-less Navigation System for Blind People in Unfamiliar Buildings," *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*, New York, NY, USA, 2023, Article 41, pp. 1–16, doi: 10.1145/3544548.3580687.

[13] A. L. Fraga, X. Yu, W.-J. Yi, and J. Saniie, "Indoor Navigation System for Visually Impaired People using Computer Vision," *2022 IEEE International Conference on Electro Information Technology (eIT)*, Grand Forks, ND, USA, 2022, pp. 257-260, doi: 10.1109/eIT53694.2022.9800294.

[14] https://developer.android.com/media/camera/choose-camera-library

[15] S. Tiwari, "An Introduction to QR Code Technology," 2016 International Conference on Information Technology (ICIT), Bhubaneswar, India, pp. 39-44, 2016, doi: 10.1109/ICIT.2016.021.

[16] S.G. Leitch, Q.Z. Ahmed, W.B. Abbas, M. Hafeez, P.I. Laziridis, P. Sureephong, and T. Alade, "On Indoor Localization Using WiFi, BLE, UWB, and IMU Technologies," Sensors, vol. 23, no. 20, pp. 8598, 2023, doi: 10.3390/s23208598.

[17] "QRCode Monkey - Free QR Code Generator," QRCode Monkey, Available: `https://www.qrcode-monkey.com`.

[18] "QR TIGER - Free QR Code Generator," QR Tiger, Available: `https://www.qrcode-tiger.com`.

[19] "PyQRCode Documentation," PyQRCode, Available: `https://pyqrcode.readthedocs.io`.

[20] "ZXing Decoder," ZXing Project, Available: `https://github.com/zxing/zxing`.

[21] "Core Image Framework," Apple Developer Documentation, Available: `https://developer.apple.com/documentation/coreimage`.

[22] "Segno - Python QR Code Generator," Segno Documentation, Available: `https://segno.readthedocs.io`.

[23] "AVFoundation Framework," Apple Developer Documentation, Available: `https://developer.apple.com/documentation/avfoundation`.

[24] https://www.youtube.com/watch?v=-L-WgKMFuhE&list= PLFt AvWsXl0cq5Umv3pMC9SPnKjfp9eGW

[25] "QRickit QR Code Decoder," QRickit, Available: `https://qrickit.com`.

Appendix A

A.1 The divided environment method

The idea of dividing an environment into smaller pieces is widely used in some industries in different ways and purposes. One of these industries is the video game industry. A huge amount of video games divide the game world into a chunk of squares, triangles, hexagons, and other shapes depending on the game's need and performance. But the difference here is that they do not use this technique for localization since the positions for all the objects in the games are known and stored at the RAMs already. They use the technique to categorize the pieces such as walkable ground, water, lava, rocks, and so on. Then these pieces are used along with their categories to find a proper path between two points. Different path finding algorithms can be used, but the most popular and simple one is A* algorithm. See this [24] incredible youtube tutorial made by Sebastian Lague, that explain an implementation of the A* algorithm and how it can be optimized.