COMP4801_FITE4801 Final year project

FYP23010: A mobile application for navigating HKU visitors with computer vision

Supervisor: Dr. Luo Ping

Group Members:

Liu Kan Man, 3035790733

Wong Riley Hoi-Kiu, 3035829039

Ng Enoch, 3035781536

Date of Submission: 21/01/20244

Abstract

Indoor positioning and navigation remain a considerable challenge for engineers. This engineering problem especially applies to the campus of the University of Hong Kong (HKU) due to its complex building layout. The common approach is to use GPS-based navigation applications (apps), such as Google Maps. However, the existing apps could only guide users to buildings rather than specific rooms or lecture venues. Additionally, research suggests that GPS is unreliable for indoor usage. This project aims to develop a vision-based navigation system to guide visitors within the HKU campus. This paper demonstrates the designs of the system structure and the computer vision (CV) model structure. The client-server structured system was designed as a combination of a mobile app, a backend service, a mapping service, and a CV model. The development of the mobile application and backend service is approaching the stage of completion, providing a strong foundation for the project. An object detection model has been selected as the initial implementation. To increase the model's accuracy, various solutions are proposed to be added. Thus, apart from integrating the mapping and navigation function into the app, the next step of the project will be evaluating a cost-effective approach to enhance the CV model's performance. The analysis in model building will provide insights into CV model training for positioning purposes.

Acknowledgment

We are grateful for the guidance and support from our group supervisor, Dr. Luo Ping, and the Department of Computer Science.

Table of Content

Table of Figures

Table of Tables

Abbreviations

app	Application
CV	Computer Vision
FPS	Frame per second
HKU	The University of Hong Kong
UI	User Interface
YOLO	You Only Look Once
SGD	Stochastic Gradient Descent
Adam	Adaptive Momentum Estimation
Adamax	Adaptive Momentum Estimation with Infinity Norm
AdamW	Adaptive Momentum Estimation with Weight Decay
NAdam	Nesterov-accelerated Adaptive Momentum Estimation
RAdam	Rectified Adaptive Momentum Estimation
RMSProp	Root-mean-square Propagation

1 Introduction

1.1 Background

As one of the well-recognized universities, the campus of the University of Hong Kong (HKU) experiences a significant daily influx of individuals. Statistics [1] state that HKU has over 13,000 new students a year. Moreover, the university hosts a multitude of events, including talks, ceremonies, and workshops, occurring with great frequency. These events serve as a significant draw for visitors to the campus. Newcomers to the university often encounter confusion when they are presented with venue names and codes, leaving them uncertain about how to navigate their way to the designated venues before they get familiar with the campus. Nevertheless, the university currently lacks a dedicated navigation tool. Instead, visitors are compelled to rely on generic GPS-based navigation apps such as Google Maps, along with text-based guides sourced from the university's websites. Regrettably, these existing apps fail to provide visual representations of the indoor environments within the campus. Furthermore, research findings [2] indicate the inherent limitations of GPS technology in accurately determining precise indoor locations. As a result, there is a noticeable demand for a system to provide a substantial number of visitors with enhanced navigation experiences.

1.2 Existing Approach: GPS-based Navigation Applications

As delineated in Section 1.1, individuals depend on GPS-based navigation apps as their primary means of navigating within the HKU campus. However, these existing apps exhibit several limitations, including restricted functionality, deficient visualization capabilities, and unreliable navigation performance. Their reliance on satellite and aerial imagery to generate maps renders them incapable of comprehensively representing indoor environments. These apps are only able to identify the buildings but lack the capacity to ascertain the specific locations of individual rooms within said buildings. Consequently, users are left devoid of any visual depiction and information of the indoor spaces. Furthermore, papers [3], [4], [5] have substantiated the insufficiency and inaccuracy of GPS location information for indoor navigation. GPS signals are susceptible to interference from obstacles and adverse atmospheric conditions, thereby rendering their accuracy unreliable. The intricate building complex of the HKU campus exacerbates this issue, amplifying the challenges faced by users.

1.3 Motivation on developing vision-based navigation system

The motivation behind developing a vision-based navigation app stems from the limitations of the existing solutions mentioned in Section 1.2. A new navigation system can provide HKU visitors with user-friendly experiences, along with additional functionality.

The computer vision approach stands out among indoor navigation technologies due to its high accuracy, robustness, infrastructure flexibility, and additional functionalities.

Vision-based indoor positioning can leverage features like object recognition, depth sensing, and motion tracking to provide precise location estimates. The estimations are less susceptible to interference from environmental factors like signal attenuation, multi-path propagation, or radio frequency interference [3]. Furthermore, vision-based systems do not require the installation of additional infrastructure or hardware, like beacons or RFID tags [4]. Also, vision-based systems can provide additional functionalities beyond just positioning, like object tracking, gesture recognition, activity monitoring, and augmented reality overlays, enhancing the overall user experience [6]. Considering the rapid development and growing popularity of computer vision applications, a vision-based system has a large extendibility and a great potential to combine with different devices and technologies.

1.4 Objectives

The primary objective of this project is to deliver a dedicated mobile app that caters to the navigation needs of both students and visitors at HKU. The app should be able to facilitate precise detection of users' locations, efficient path navigation, and intuitive visualization of detection results and navigational guidance.

Another objective is to evaluate the cost-effectiveness of the computer vision method in comparison to other navigation approaches. This evaluation will encompass the optimization of each step involved in model development, including data collection, data processing, model training, and model fitting. By conducting a comprehensive cost-effectiveness analysis, this project aims to provide valuable insights and contribute to the existing body of knowledge on computer vision applications, particularly for Visual Positioning System (VPS).

1.5 Outline

The report is structured into 6 chapters. Chapter 1 outlines the project's motivation and objectives. In Chapter 2, the design details of the system, app, and model are presented. Chapter 3 focuses on the early-stage results obtained thus far. Chapter 4 highlights the encountered difficulty and the corresponding responses formulated. Chapter 5 discusses the proposed schedule and the coming tasks. Finally, Chapter 6 concludes the report, summarizing the project's progress up to the current stage.

2 Methodology

This chapter provides a detailed discussion of the project's design and implementation. It begins with the system architecture (Section 2.1) and app design (Section 2.2), followed by an exploration of the mapping and navigation component (Section 2.3) and the backend service (Section 2.4). Moreover, the chapter introduces the model design (Section 2.5) and the data processing flow (Section 2.6).

2.1 System Architecture

The section presents an overview of the system architecture, followed by the selection of the backend development tool and server hosting.

2.1.1 System Overview

As shown in Figure 1, the system uses the client-server structure. The client side has the mobile app user interface (UI). Further details regarding the app UI are discussed in Section 2.2. The app is linked to Google Maps Service and FengMap Service via corresponding APIs. The mapping and navigation services are further explained in Section 2.3. The server handles the backend service and the CV model. More information about the backend service and the CV model is discussed in Section 2.4 and Section 2.5, respectively. Data for training the model is processed with the assistance of Roboflow, which is presented in Section 2.6.

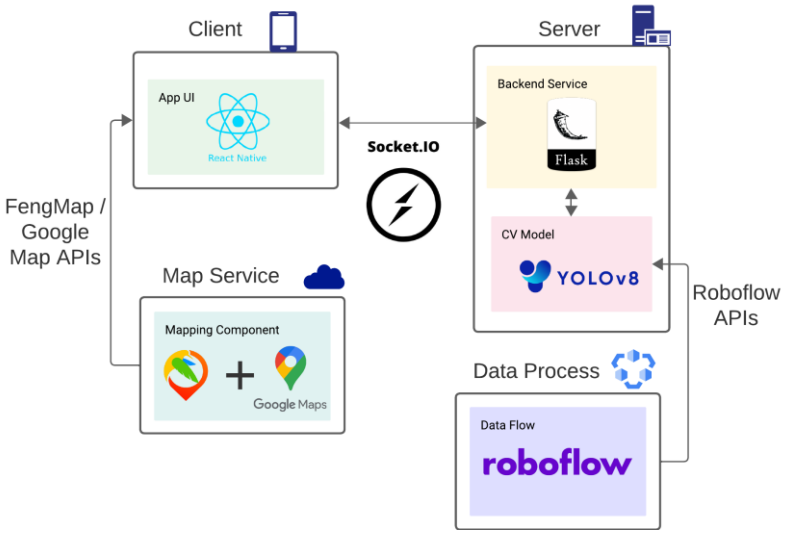


Figure 1 System structure diagram

2.1.2 System Flow

2.1.2.1 Client Side

As illustrated in Figure 2, the client-side functionality of the system follows the subsequent steps. Upon launching the app, the connectivity status is verified. If an active connection to the server is detected, the app initiates the streaming of videos to the server and awaits a response. Conversely, in offline mode, users can set their current location using manual input. Once the current location is established and the destination is provided by the user, the mapping algorithm is executed to optimize the generated path. The resulting optimized path is then rendered and displayed on the app UI.

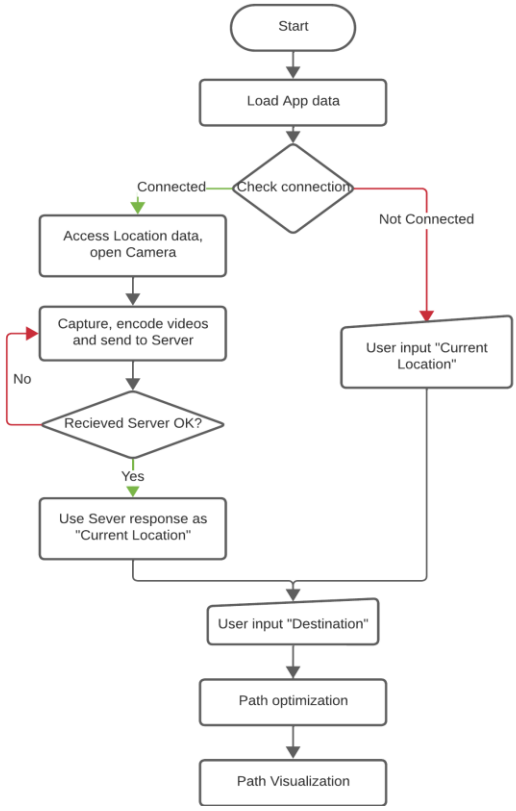


Figure 2 Flowchart of the client side

2.1.2.2 Server Side

The server-side workflow, as depicted in Figure 3, commences upon receiving a request from the client. The server establishes a connection and begins to receive the streamed photos. Subsequently, the received videos are sequentially decoded and processed by the object detection model. The

model outputs the location estimation and associated scores. The server transmits the location result back to the client when the model returns the result.

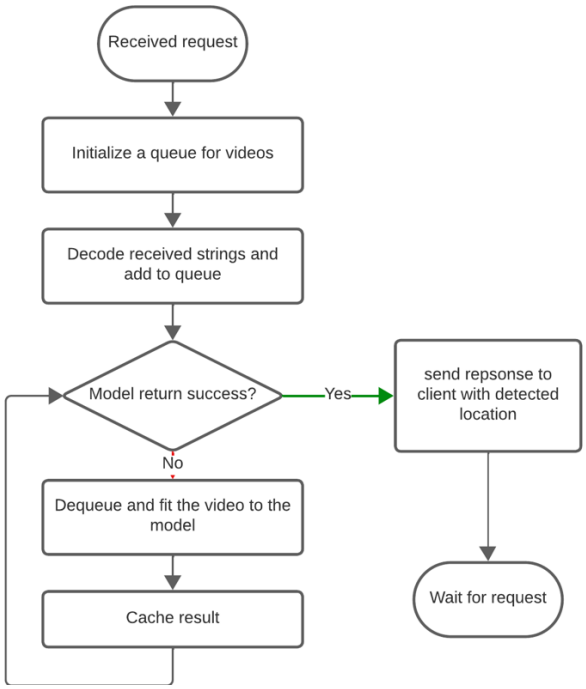


Figure 3 Flowchart of the server side

2.2 Mobile Application

This section starts with a brief introduction to the mobile app, followed by the features of the app. Additionally, the chapter discusses the selection of the development framework, considering the factors that influenced this decision.

2.2.1 App Introduction

The proposed app encompasses a UI that facilitates the visualization of a virtual map, the camera feed, and the inclusion of user controls. Additionally, the app is responsible for implementing pathfinding functionality, and providing information about the campus.

2.2.2 App Features

The app offers 2 major features: vision-based positioning and path finding. Some additional features are implemented to facilitate users' navigation experience in HKU.

2.2.2.1 Vision-based Positioning

The app employs the camera of the user's smartphone to capture images of the surroundings at a predefined frame rate. These images are securely transmitted to the server and processed using the VC model. Once a sufficient number of images have been analyzed, and the model's confidence level has been met, the resulting location is displayed on the map. This location information can be utilized for path finding purposes as well.

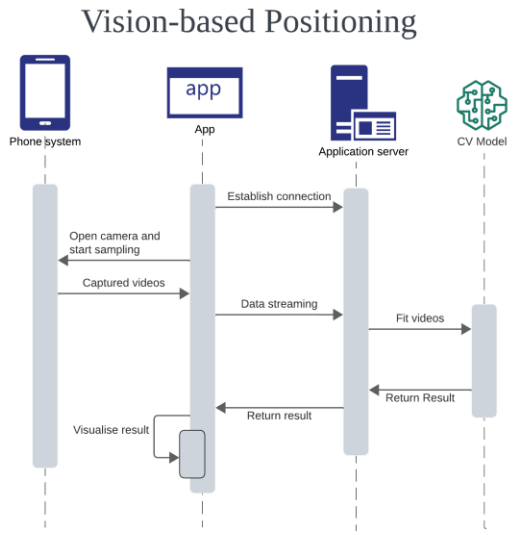


Figure 4 Sequence diagram for Vision-based Positioning

2.2.2.2 Path Finding

The app enables users to use their location and desired destination to navigate within the HKU campus. It then optimizes the best route and displays it on the map, providing navigational cues for guidance. This versatile approach ensures seamless navigation within the HKU campus, regardless of indoor or outdoor environment.

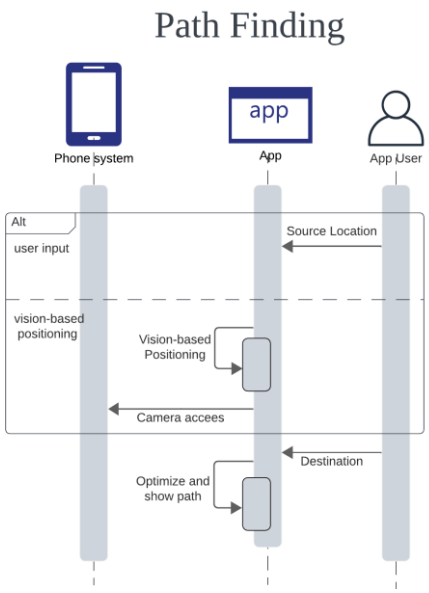


Figure 5 Sequence diagram for Path Finding

2.2.2.3 Additional Feature: Accessing HKU Campus information

The app offers users a page to access comprehensive information about the HKU campus. This page provides details about various buildings, including pictures, locations, departments, websites, and available facilities for visitors. Users can effortlessly explore the campus while navigating with the app, enhancing their overall experience.

2.2.3 Selection for Frontend Development Framework: React Native

React Native is chosen for the mobile app UI development. React Native is elected for its cross-platform nature, the vast ecosystem, and the large community. It allows developers to write code once and deploy the code on both Android and iOS, saving time and effort compared to developing separate apps for each platform using other frameworks. Moreover, the extensive library of pre-built components available in the React Native ecosystem can be easily integrated into applications, reducing the need for building UI elements from scratch and accelerating development speed. Additionally, React Native has a large and active community. The strong community support adds value to React Native by providing a rich pool of resources and making it easier for developers to find solutions to challenges during the app development process.

2.3 Mapping and Navigation Component

This section describes the use of the mapping and navigation component and explains the selection of the development tool for the component.

2.3.1 Mapping and Navigation Component Introduction

The mapping and navigation component is responsible for mapping users' locations onto the campus map and optimizing the paths to the destination. The map should cover the scope of the project, that is the Main Building. This component will be using online mapping services to ensure the campus map is standardized and up to date.

2.3.2 Selection of Development Tool

There are a number of choices for the mapping development tool, including Google Maps, FengMap, Mapbox, OpenStreetMap, IndoorAtlas, MappedIn, and ArcGIS. Considering factors for the tool selection are accessibility, money cost, development complexity, and performance. Table 1 shows the comparisons of the shortlisted tools. More details about the development tools are included in Appendix A – E.

Tool	Accessibility	Money Cost	Development Complexity	Performance
Google Maps	✓	Map: Free API: Pay As You Go	- Easy - Dependent to Google Maps Team	- Integrated with the map outside campus - Place to Place navigation - No Control to Map - No visualisation for indoor environment
FengMap	✓	Map: Fiexed Cost API: Free to Use	- Draw map from scratch - Well support for mapping and navigation	- Poorly integrated with the map outside campus - Point to Point navigation - Good Indoor Map Visualisation
Mapbox	✓	Expensive	- Draw map from scratch - Well support for mapping and navigation	- Integrated with the map outside campus - No visualisation for indoor environment

OpenStreetMap	✓	Free (Open Source)	- Draw map from scratch - Poor support for mapping - Required an additional navigation solution	- No Navigation - No visualisation for indoor environment (Floor plan)
IndoorAtlas	✓	Expensive	- Draw map from scratch - Well support for mapping and navigation	- Poorly integrated with the map outside campus - Point to Point Navigation - Good Indoor Map Visualisation
ArcGIS	✓	Expensive	- Draw map from scratch - Well support for mapping and navigation	- Poorly integrated with the map outside campus - Point to Point Navigation - Good Indoor Map Visualisation
MappedIn	**No Access in Hong Kong**	NA	NA	NA

Table 1 Comparisons of development tools for mapping and navigation

2.3.2.1 Google Maps

Google Maps services offer user-friendly interfaces and reasonable costs, making them easily accessible. Google Maps provides an indoor map for the Main Building, which saves effort for the project. Moreover, users are generally already familiar with the Google Maps interface. Implementing Google Maps services in the app would eliminate the need for users to acquaint themselves with a new map, resulting in a seamless user experience.

However, it should be noted that the completion of building navigation within the Main Building is a task that extends beyond the scope and timeline of this final-year project. Achieving this would require providing additional information to the Google Maps team and collaborating closely with them. Consequently, considering the limitations and constraints of the project, Google Maps may not be the optimal solution for fulfilling the project requirements.

2.3.2.2 FengMap, Mapbox, IndoorAtlas, and ArcGIS

FengMap, Mapbox, IndoorAtlas, and ArcGIS are notable providers of specialized services for indoor map development and navigation. These platforms offer robust support for creating visually appealing indoor maps, optimizing paths, and incorporating features like floor selectors.

However, it should be noted that building indoor maps from scratch using these tools can be time-consuming, requiring significant development efforts. Another aspect to consider is the cost associated with these development tools, which unfortunately exceeds the project's allocated budget. While all four platforms provide similar services, FengMap stands out as the most cost-effective option among them.

2.3.2.3 OpenStreetMap

OpenStreetMap is an open-source resource that can be used to develop and view indoor maps. However, it provides the least support for development, and it must work with various additional tools to provide mapping and navigation functions, rendering an extremely high development complexity.

2.3.2.4 Proposed Solution

The shortlisted development tools exhibit varying performance in outdoor and indoor scenarios. To address this, a hybrid approach is proposed, allowing users to seamlessly switch between an outdoor map and an indoor map. For the outdoor map, Google Maps is chosen due to its extensive maturity and widespread user familiarity. On the other hand, for the indoor map, FengMap is selected as the most cost-effective option among specialized indoor mapping tools.

However, it should be noted that the allocated budget remains insufficient to develop the entire indoor map for the Main Building. Consequently, only 2 floors will be included in the map, and there may be limitations in achieving an accurate scale due to budget constraints. Nevertheless, it is important to highlight that the scale of the map does not impact the visualization or other essential features provided by the map.

By combining Google Maps for the outdoor map and FengMap for the indoor map, the proposed hybrid approach aims to strike a balance between user familiarity, functionality, and cost-effectiveness. While limitations exist, this approach optimizes the available resources to provide users with a comprehensive navigation experience encompassing both outdoor and limited indoor areas of the Main Building.

2.4 Backend Service

This section discusses the development framework and the transmission technology selected for the backend service of the project. This section also describes the implementation of the backend service.

2.4.1 Selection of Backend Development Framework: Flask

The chosen backend development framework is Flask. Flask is a lightweight and flexible Python framework specifically designed for building RESTful APIs and backend services. Its simplicity and ease of use make it ideal for rapid prototyping and development of backend functionality in the system. Flask's extensive ecosystem of extensions and libraries further enhances its capabilities, enabling seamless integration with the CV model and data management requirements of the project. By utilizing Flask, the system can leverage the robustness and efficiency of Python while benefiting from Flask's simplicity and flexibility.

2.4.2 Selection of Transmission Technology: WebSocket

WebSocket is the chosen communication protocol for the project. WebSocket provides a persistent, full-duplex communication channel over a single TCP connection. This enables real-time, bidirectional communication between clients and the server. WebSocket has low latency and eliminates the need for frequent HTTP request-response cycles, resulting in faster and more efficient communication. WebSocket also offers enhanced scalability and resource utilization. Its event-driven model reduces unnecessary requests and server load, allowing servers to handle a larger number of concurrent connections with lower resource consumption. Furthermore, WebSocket supports cross-origin communication, enabling clients from different domains to establish secure connections and exchange data.

Socket.IO is used to establish the WebSocket connection. Socket.IO simplifies the implementation of real-time, bidirectional communication using WebSocket. It offers compatibility with various browsers and environments, automatic reconnection, and event-based messaging, which align with the project requirement.

2.4.3 Backend Service Implementation

The WebSocket connection is initiated upon launching the app. During active camera usage by users to scan their surroundings, short video clips captured by the phone camera are transmitted to the server in base64 format. Upon receipt of these video messages, the server proceeds to process the encoded data strings and fit them into the model. Upon deriving the output of the model, denoting the detected location, the server transmits the result back to the client through the existing WebSocket connection. The resulting message encapsulates a unique message ID, the success state, the coordinates, and the corresponding location name.

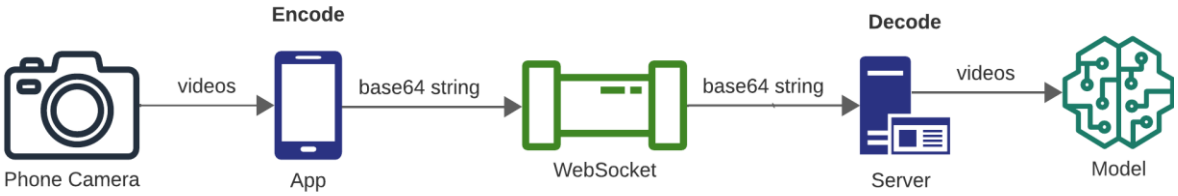


Figure 6 Flow chart for the flow of captured videos to the model

2.5 Model Design

This section explains the decision made for the model structure, followed by the selection of the building tools.

2.5.1 Model Selection and Structure Design

For the computer vision model, three kinds of algorithms are shortlisted for selection: image similarity algorithm, image classification algorithm and object detection algorithm.

Image classification algorithms and image similarity algorithms consider detection from the image content level while object detection algorithms rely on the object features. Moreover, both image classification and image similarity algorithms are unable to provide localization information from the detection.

On the other hand, object detection algorithms can handle multiple detection at the same time. Object detection algorithm can provide real time tracking features on video, with location information such as coordinates on image, while image classification algorithm and image

similarity algorithm detect perform detailed analysis on image content. As our project primarily aims to provide localization and navigation clues to users with the computer vision approach, object detection algorithms are the best option as our model algorithms.

The object detection layer serves to detect buildings, markers, and other features within the images. However, single-layer model yielded suboptimal outputs in terms of accuracy. Therefore, a supplementary image similarity layer is proposed. By filtering out irrelevant objects, such as humans, with object detection layer, the accuracy of the model is expected to improve. Furthermore, the object detection layer aids in the selection of images for subsequent comparison within the image similarity layer. This strategic image selection shall significantly reduce the computational burden and running time of the additional layer. This refined architecture may demonstrate a more comprehensive approach to the vision-based navigation system, improving its overall accuracy and efficiency.

The finalized selection for the object detection model is discussed in Section 2.5.2. Shortlisted frameworks for building the additional image similarity layer are OpenCV and ResNet. The final selection will be determined after testing.

2.5.2 Object Detection Model Selection

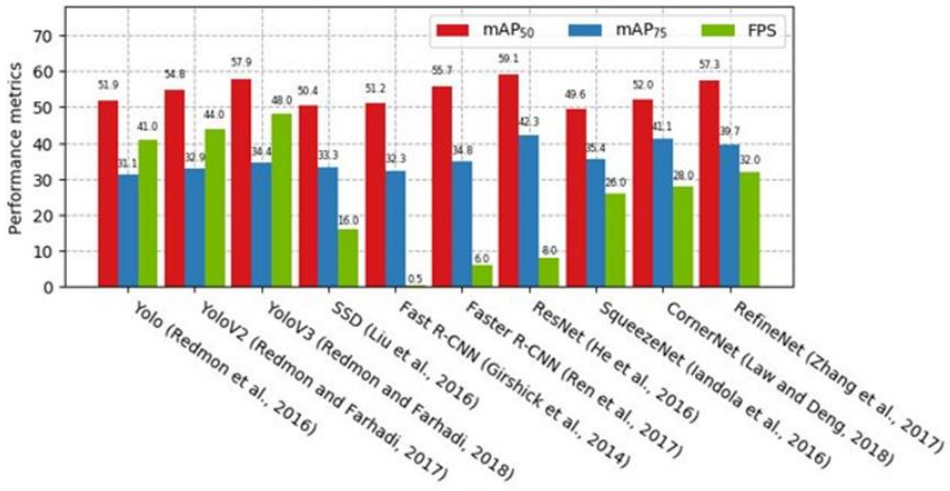


Figure 7 Performance metrics chart for different object detection models

For the object detection model, numerous algorithms have been introduced to the public. In Figure 7, a comparison of object detection and recognition performance that has been conducted by Sorin [7], You-only-look-once (YOLO) algorithms have a better performance in high frames-per-

seconds (FPS) rate. YOLO being a single-stage detector, has a relatively lower complexity than second-stage detectors such as RefineNet, CornerNet, etc. This allows YOLO to provide faster detection results, supporting real-time detection. This feature aligns with our project requirements – to provide real-time localization and navigation assistance.

YOLO is selected for implementing the object detection layer of the model.

2.5.2.1 YOLO

YOLO is a real-time object detection model that uses a single neural network to directly predict class labels, enabling the model to recognize multiple objects within an image at a faster pace. The fast-paced real-time navigation reduces the time needed for computing the location result. This computing efficiency becomes a compelling justification for selecting YOLO, as it ensures a responsive user experience during navigation tasks.

In our project, the latest version of YOLO (YOLOv8), provided by Ultralytics, has been selected for implementation. Ultralytics is a software company that publishes open-source YOLO models to the public. The application interface (API) provided extensive and efficient functionality for users to build customized object detection models. Contributing to the wide community of YOLO users, guidelines and support are available online. In addition, according to official documents released by Ultralytics, YOLOv8 has a low hardware requirement: GPU with a minimum of 8GB of memory. The low hardware requirement is favorable for the implementation of our project under the scope of Final Year Project.

2.5.2.2 Optimizer

API provided by Ultralytics allows users to select suitable optimizer algorithms, such as Stochastic Gradient Descent (SGD), Adaptive Momentum Estimation (Adam), Adam with Infinity Norm (Adamax), Adam with Weight Decay (AdamW), Nesterov-accelerated Adam (Nadam), Rectified Adam and Root-mean-square Propagation (RMSprop). The decision of the optimizer algorithm will be based on the results, further discussed in section Section 3.2, after testing. To evaluate the effect of the corresponding optimizer algorithm, testing with the same hyperparameters (listed in Appendix F) and the same dataset will be used in the model training process with 300 epochs.

2.6 Data Flow

To prepare quality data for training the CV model, every data must go through these 4 phases: data collection, data processing, data labelling, and data augmentation, before fitting into the model.

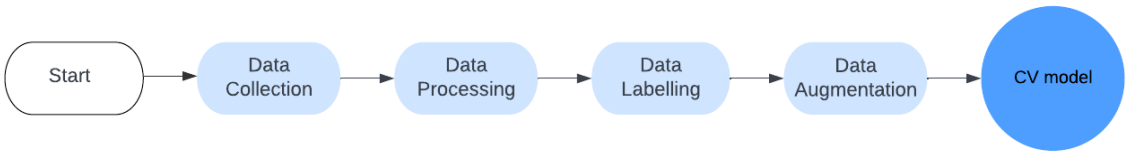


Figure 8 Demonstration of the 4 phases data workflow

2.6.1 Data Collection

In order to collect data for our research, we considered several sources including online sources, past research, internal sources, and primary data sources. Upon examination, we found that the image quality of online sources was poor for our topic and past research internal sources were not disclosed to the public for easy access, making it difficult to gather up-to-date information. Therefore, we decided that the primary data source would be the best option as it provides us with better control over both the quantity and quality of the images.

In the data collection phase, image data is collected manually through phone cameras, taking photos one by one. However, we soon discovered that this method was inefficient. Therefore, we conducted a data experiment and found that a more efficient method was needed to improve the data collection process.

To enhance the effectiveness of data collection, a more efficient method is proposed and implemented: videos are taken and used to extract images framewise. This approach generated hundreds of images quickly, significantly shortening the time it took to complete the data collection process.

2.6.2 Data Processing

Data processing includes data cleaning, image compression, and image formatting. Data cleaning involves the removal of noise, outliers, and artifacts from the acquired data to ensure its quality

and integrity. Image compression techniques are applied to reduce the storage and computational requirements. Image formatting is performed to standardize the images and facilitate seamless integration with the 2 layers, ensuring consistent results.

2.6.3 Data Labelling

Data labelling generally requires a large amount of human effort and time. To speed up the development, we chose to label our data on RoboFlow, a modern data workflow management tool [8]. It not only allows annotating on images but also reviewing the summary of the annotated dataset and updating the data whenever needed. The integration of RoboFlow in the data labelling process streamlines the annotation workflow, enhances efficiency, and maintains the quality and consistency of the labelled data.

2.6.4 Data Augmentation

Data augmentation is employed in the project to increase the diversity and robustness of the training data. By applying transformations such as cropping, flipping, rotation, scaling, and adjusting brightness, the augmented dataset introduces variations and simulates real-world conditions. This technique enhances the models' ability to generalize and accurately classify objects in different scenarios, improving the overall reliability and robustness of the navigation system.

3 Interim Results

This chapter discusses the progress for the data collected, optimizer algorithm evaluation, the mobile application implementation, and the indoor map development.

3.1 Data Collection

After capturing videos among 40 places within the HKU main building, thousands of images are extracted from it. Images with low quality are filtered and the remaining are labelled with the corresponding location tags. Examples of tag names are “room201”, “room202”, “room203”, etc. Lastly, by applying some data augmentation techniques, namely rotation variation and brightness

variation, a total of 5978 images are generated from 2264 raw images. These images are then split into training dataset, validation dataset and testing dataset in the ratio of 8:1:1 respectively.

3.2 Optimizer Algorithm

An experiment was conducted to investigate the impact of optimizer algorithms on the accuracy of object detection during model training. Table 2 below shows the overall class accuracy (Precision) using different optimizer algorithms.

Optimizer	Precision
SGD	0.965
AdamW	0.958
Adamax	0.956
RAdam	0.944
NAdam	0.942
Adam	0.934
RMSProp	0.517

Table 2 Overall class precision for YOLOv8 training with different optimizer algorithms

Based on the overall class detection accuracy, RMSProp has the worst performance among all. The precision of 0.517 indicates that the model trained with RMSProp in the given conditions is not accurate enough to support precise and fast object detection. On the contrary, other optimizer algorithms have a better performance in terms of overall class accuracy, ranging from 0.934 to 0.965. Of the seven optimizer algorithms tested, SGD performed the best. A more detailed evaluation of the different optimizer algorithms is provided in the following sections.

3.2.1 Evaluation Metrics

In the following sections, we will evaluate each optimizer's performance in the model training process based on corresponding Precision curve, Recall Curve and Precision-Recall Curve.

The precision curve allows us to have an understanding of how precision (ability to make positive predictions) varies as the detection threshold changes, providing insight into the ability to make positive predictions.

The recall curve illustrates the variation of recall (ability to make true positive value comparing with positive instances, true positives and false negatives) value at different detection threshold values, establishing the performance of the model in correctly identifying objects.

Precision-Recall curve showcases the trade-offs between precision and recall at varied confidence thresholds. From the precision-recall curve, we may derive the value of mean average precision at an Intersection over Union (IoU) threshold of 0.5 (mAP@0.5). This metric evaluates the average performance in terms of the trade-off between precision and recall when the level of overlapping predicted bounding boxes and ground truth bounding boxes exceeded the threshold of 0.5.

3.2.2.1 SGD

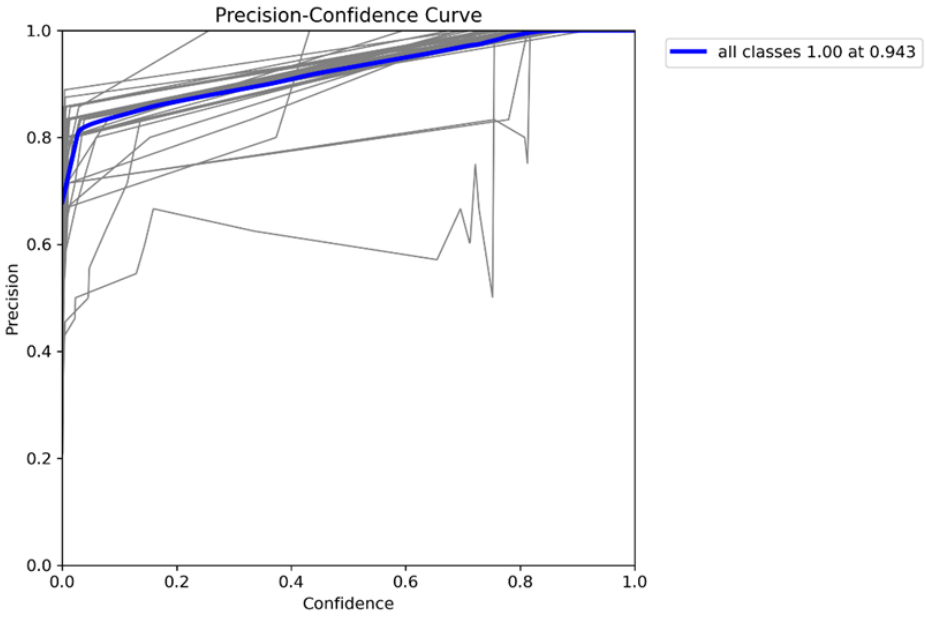


Figure 9 Precision-Confidence Curve for model using SGD

From Figure 9, precision curves for each class are establishing an overall upward trend at a high precision value. The blue curve indicates the overall class precision curve for the model. As the confidence threshold value increases, the precision of the model increases correspondingly and achieves maximum precision (precision = 1.0) when the confidence level is set as 0.943.

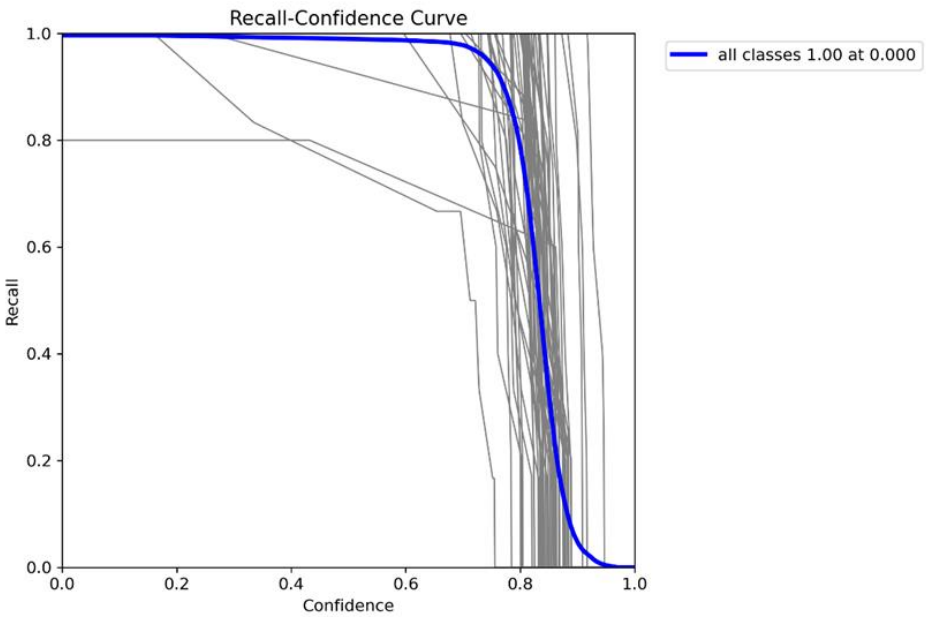


Figure 9 Recall-Confidence Curve for model using SGD

The presented recall curve, as depicted in Figure 10, corresponds to the performance of a model that has employed the stochastic gradient descent (SGD) optimizer algorithm. The curve shows a perfect recall value of 1.0 when the confidence threshold is set between 0.0 and 0.4, indicating that the model's performance is optimal when the threshold is lenient. As the confidence threshold value increases, a slight decline in the recall curve is observed. However, when the confidence threshold value reaches 0.8, the recall curve exhibits a gradual decline, indicating that the model's ability to make true positive predictions decreases as the threshold becomes stringent.

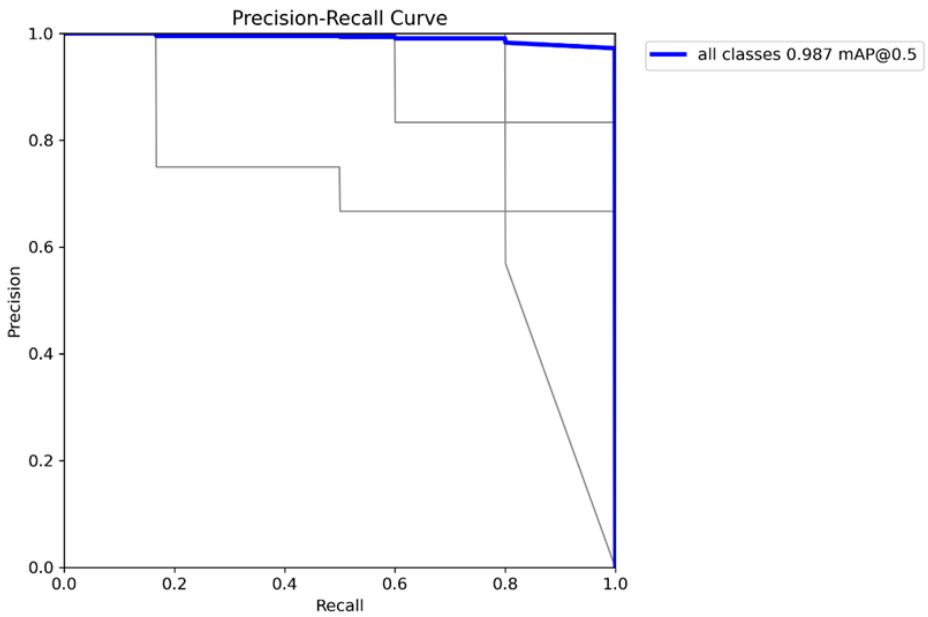


Figure 11 Precision-Recall Curve for model using SGD

Figure 11 illustrates the trade-off for precision and recall for the model trained with SGD as the optimizer algorithm. The overall class precision-recall curve indicated that the model has a high precision value and high recall value at most of the threshold values. From the graph, we obtained mAP@0.5 is equal to 0.987 which shows that the model has a very good performance in terms of the trade-off between precision and recall. The precision-recall curve for the model is almost identical to the precision-recall curve of an ideal detector, high precision value and high recall value at all confidence thresholds.

3.2.2.2 Adam

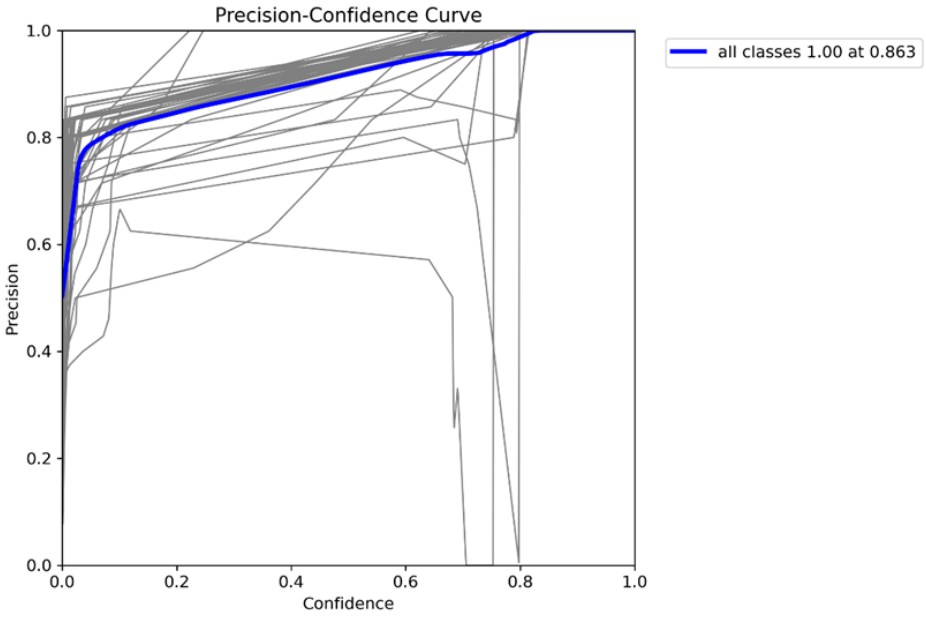


Figure 10 Precision-Confidence Curve for model using Adam

From the precision curve (Figure 12), the precision curve of the overall classes established a general upward trend and achieved 1.0 precision at a 0.863 confidence level. By looking at the precision curve for individual classes, a fluctuating and inconsistent pattern is observed. Precision curve of individuals classes reaches 0.0 precision at confidence score threshold ranged from 0.7-0.8. It indicates that the model performance in terms of precision for individual classes is not consistently reliable and precise.

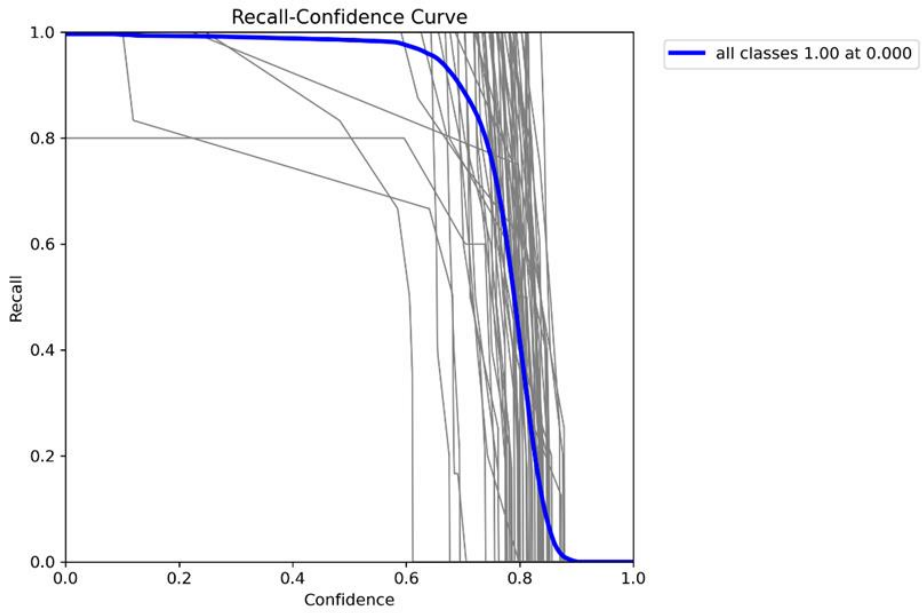


Figure 11 Recall-Confidence Curve for model using Adam

Above is the Recall curve obtained after the model training using Adam as the optimizer algorithm. It has a very similar trend to the recall curve for the model using SGD (Figure 10). However, the perfect recall value is only maintained when the confidence is set between 0.0 – 0.3. The recall curve declined as the confidence threshold value increased and drastically dropped when the confidence threshold was set between 0.75 - 0.85. From the graph, some classes even result in 0.0 recall at a 0.62 confidence threshold. It shows that the model may not be able to be detected in a stringent confidence threshold.

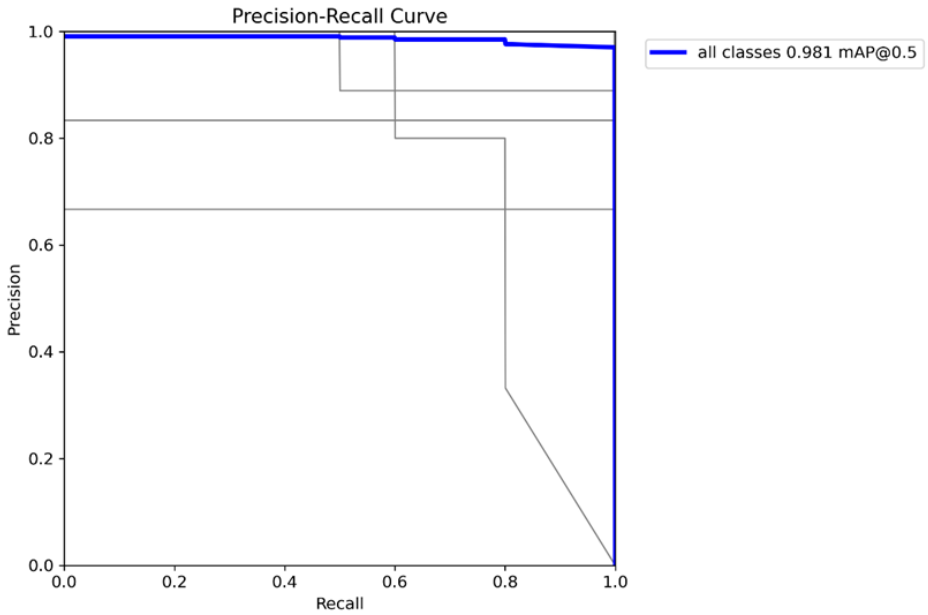


Figure 12 Precision-Recall Curve for model using Adam

Figure 14 depicts a similar precision-recall curve pattern as shown in Figure 11. The overall class precision-recall curve indicated that the model has a high precision value and high recall value at most of the threshold values. The mAP@0.5 value obtained from the precision-recall curve is 0.981, indicating the model has an acceptably good performance in terms of precision-recall trade-off.

3.2.2.3 AdamW

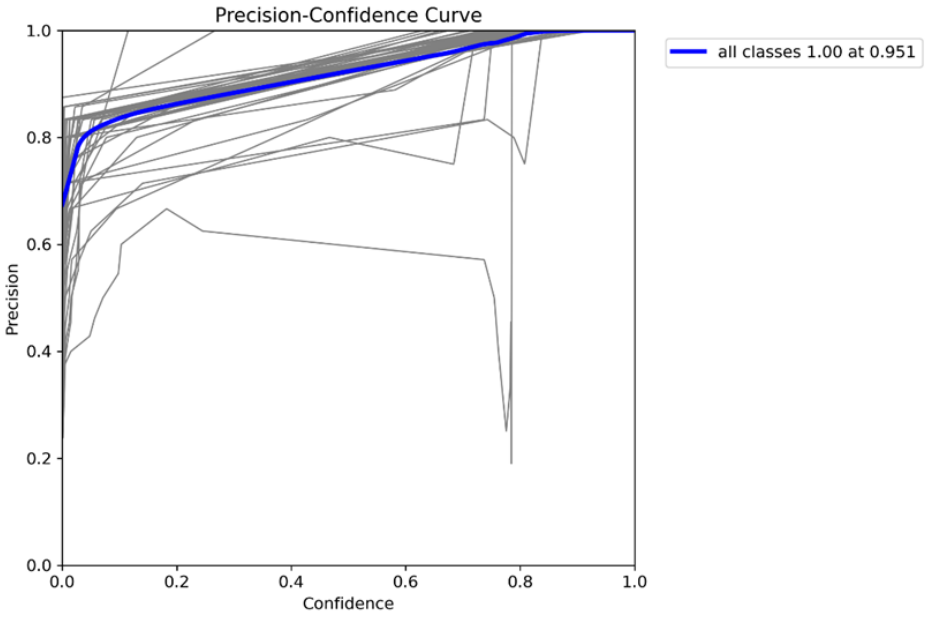


Figure 13 Precision-Confidence Curve for model using AdamW

In Figure 15, the blue curve representing the precision curve for overall classes establishes an upward trend. The model obtained a precision score of 1.0 when the confidence score threshold was set to 0.951. However, for some classes, the precision fluctuated as the confidence score threshold increased.

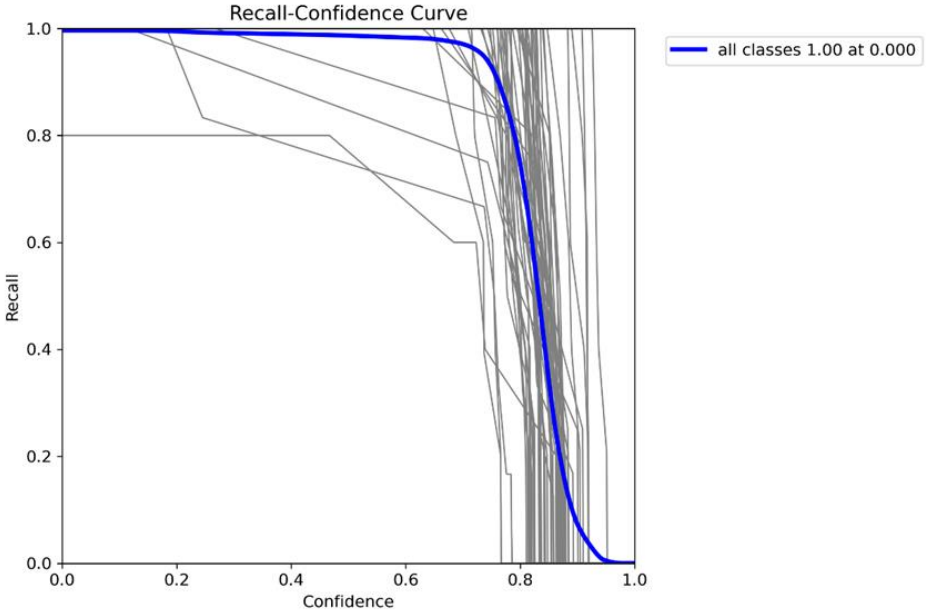


Figure 14 Recall-Confidence Curve for model using AdamW

By comparing the above figure with the recall curve for the model using SGD (Figure 10), a similar pattern is observed. Both curves display comparatively high recall values at lenient confidence threshold levels. As the confidence threshold becomes stricter, the recall value drops drastically from the 0.75-0.9 confidence threshold level. The recall curve demonstrates that the model has an overall good performance in terms of recall at different confidence thresholds.

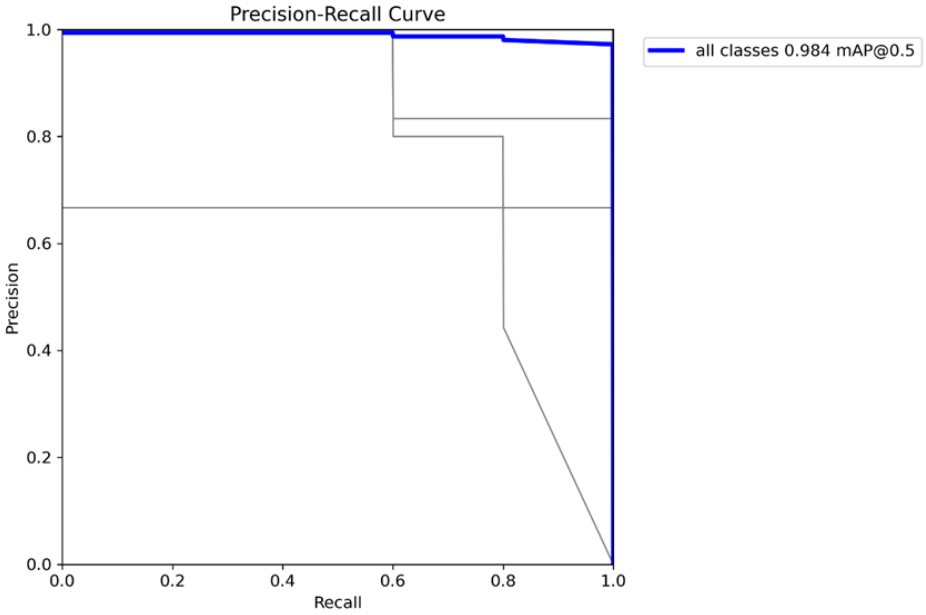


Figure 15 Precision-Recall Curve for model using AdamW

The presented analysis depicts a precision-recall curve pattern in Figure 17 that is analogous to the one shown in Figure 10. The overall precision-recall curve reveals high precision and recall values for the model at most threshold values. The mAP@0.5 value obtained from the precision-recall curve is 0.984, indicating the model has an acceptably good performance in terms of precision-recall trade-off.

3.2.2.4 Adamax

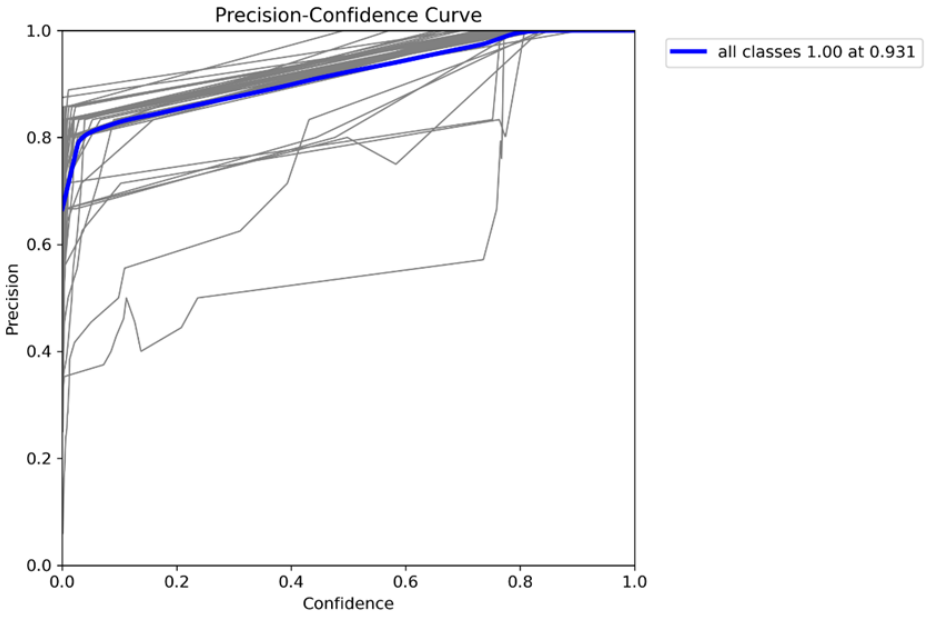


Figure 16 Precision-Confidence Curve for model using Adamax

In Figure 18, the blue curve demonstrates an upward trend for the overall class precision. Moreover, the perfect value of the overall class precision is retained when the confidence score threshold is equal to or greater than 0.931. It is noteworthy that as the confidence score threshold value increases, there is a steady improvement in precision observed for each class.

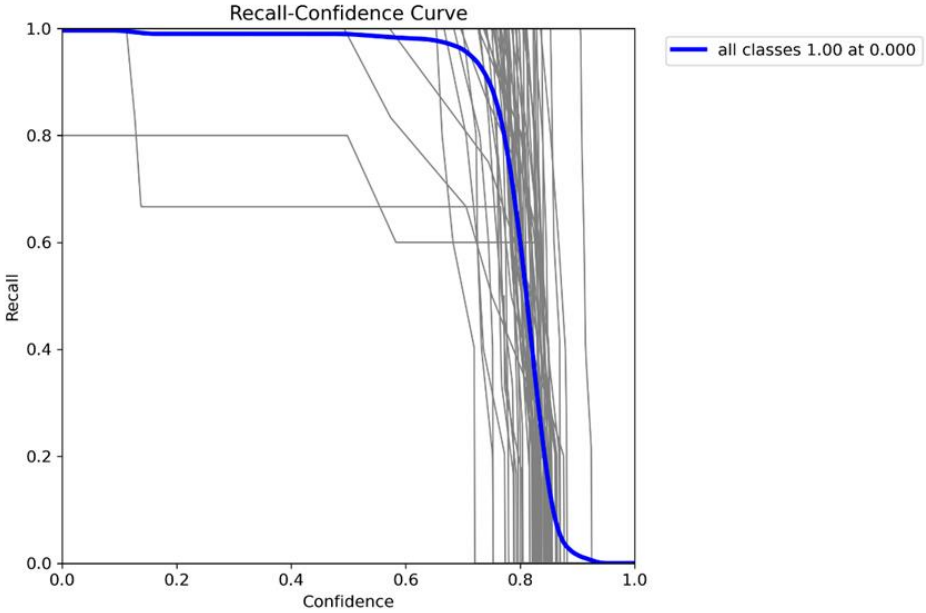


Figure 17 Recall-Confidence Curve for model using Adamax

From Figure 19, high recall values are obtained when the confidence threshold value is lenient (0.0 – 0.6). The recall curve gradually declined when the confidence threshold value became stringent, decreasing when the confidence was set between 0.75 - 0.9. Moreover, some classes, represented by the grey curve, result in a relatively low recall value at lenient confidence threshold values. It indicates that, for some of the classes, the detection may not be able to discover targeted features successfully.

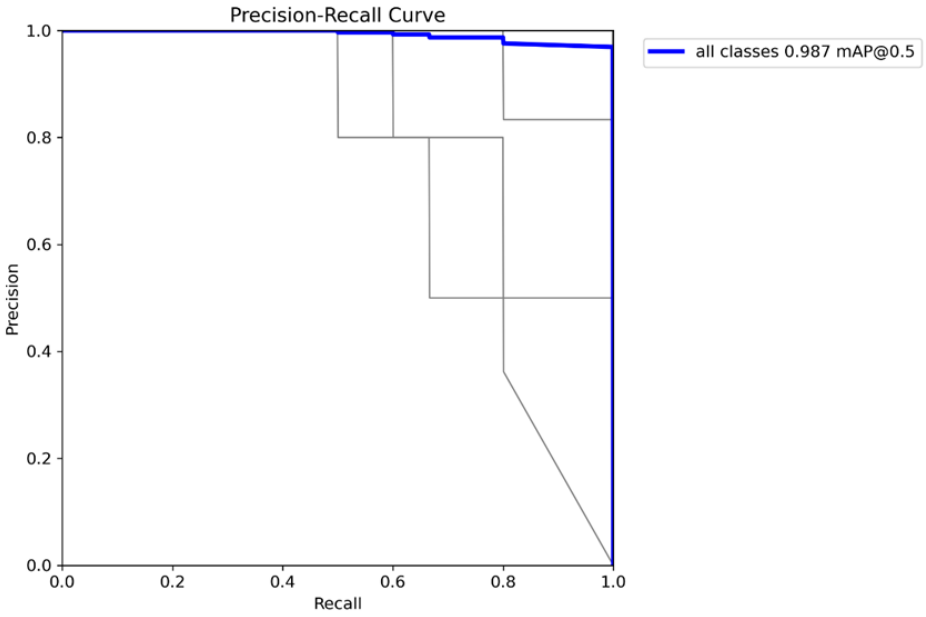


Figure 18 Precision-Recall Curve for model using Adamax

Figure 20 depicts the precision-recall curve for the model using Adamax. The mAP@0.5 value is 0.987, which is the same as the mAP@0.5 score for the model using SGD (see Figure 11), for the overall class performance. Moreover, by comparing the pattern of the individual class's precision-recall curve from both figures, the model using Adamax as an optimizer algorithm demonstrates a better performance in terms of the trade-off between precision and recall.

3.2.2.5 RAdam

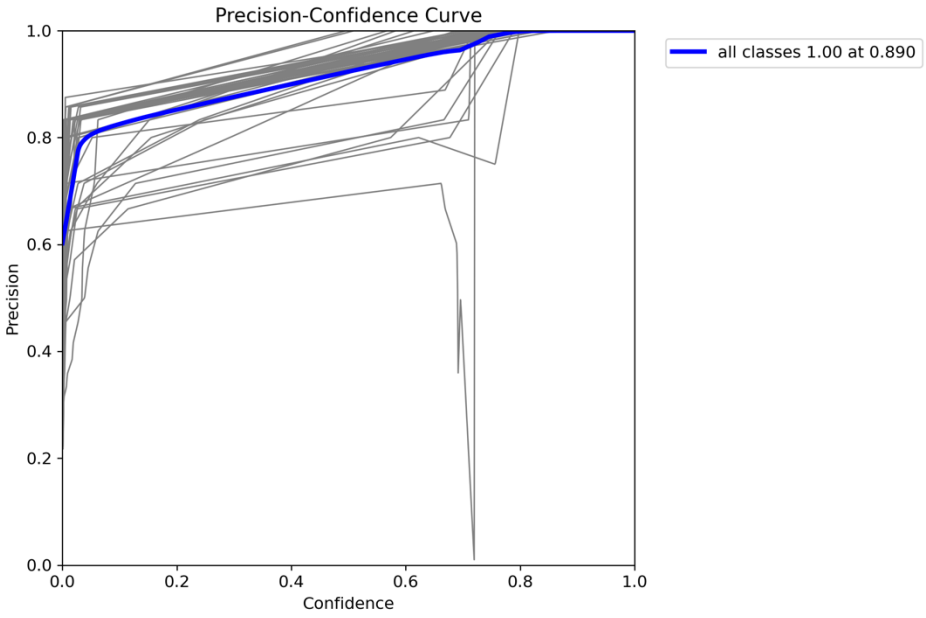


Figure 19 Precision-Confidence Curve for model using RAdam

The precision curve shown in Figure 21 illustrates an upward trend. The precision for all classes achieves 1.0 when the confidence threshold value is equal to or greater than 0.89. Further looking into the precision performance of individual classes, abnormal cases, where the precision fluctuated drastically, are observed. The presence of the pit in the precision-confidence curve suggests that the model is struggling with making correct predictions in a certain range of confidence score threshold for some of the classes. Therefore, there is still room for improvement in terms of prediction accuracy.

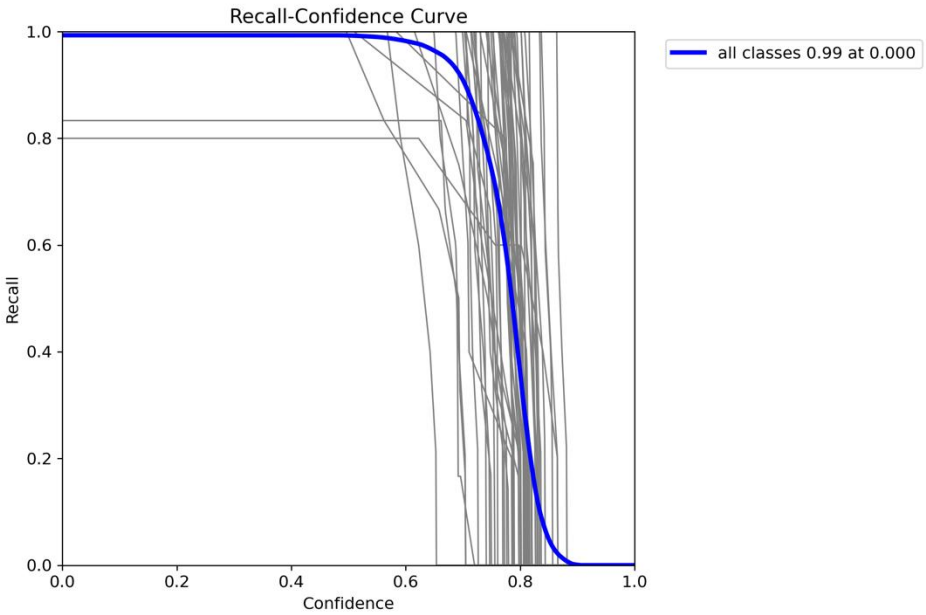


Figure 20 Recall-Confidence Curve of model using RAdam

The recall-confidence curve shown in Figure 22 follows a similar pattern as the ones obtained from models that use different optimizers. The performance, measured in terms of overall class recall, is good for lenient confidence threshold levels. As the confidence level becomes more stringent, the recall value decreases. Notably, the overall class recall value is not as high as that of other optimizers, where maximum recall value is 1.0, whereas the model has a maximum recall value of 0.99. The slight difference suggests that both the model and dataset may require further improvement to provide better results.

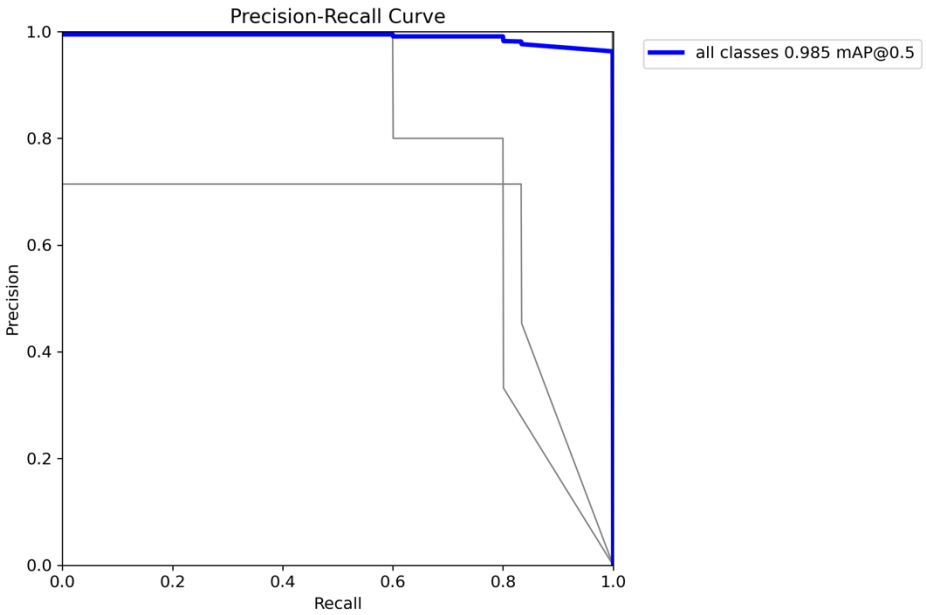


Figure 23 Precision-Recall Curve of model using RAdam

The precision-recall curve in Figure 23 provides an approximately ideal model performance in terms of the trade-off between precision and recall. The precision value remains high for most of the recall values. The mAP@0.5 value obtained from the figure is 0.985.

3.2.2.6 NAdam

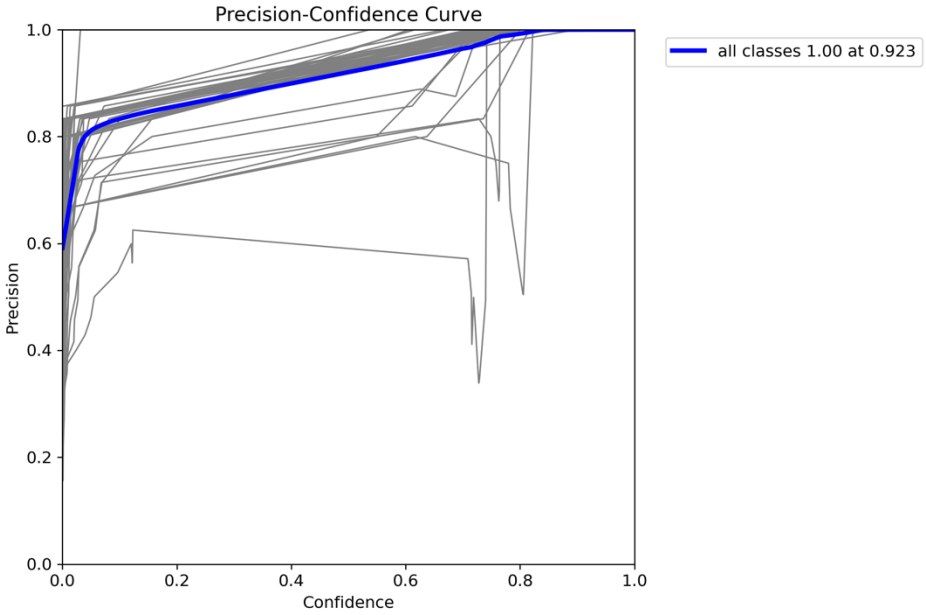


Figure 21 Precision-Confidence Curve of model using NAdam

The overall class precision observed in Figure 24 shows an upward trend as the confidence threshold value increases. The model achieved a precision value of 1.0 at a confidence threshold value equal to 0.923 for all classes. However, the precision-confidence curve for individual classes fluctuates. This suggests that the accuracy of precision provided by the model in different classes can be inaccurate and unreliable.

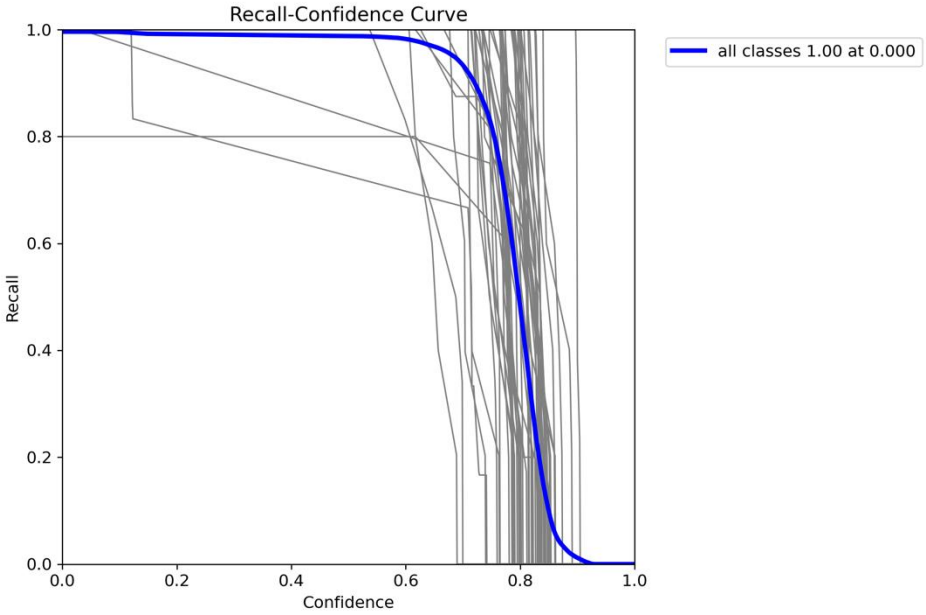


Figure 22 Recall-Confidence Curve of model using NAdam

The recall-confidence curve in Figure 25 drops slowly from confidence threshold 0.0 to confidence threshold 0.6. It declines drastically from confidence threshold 0.75 – 0.9. Additionally, the grey curves that represent the recall-confidence curves for individual classes display distinct fluctuations. Some of the classes exhibit a noteworthy decline, achieving only 0.8 recall at lenient confidence threshold level. These observations suggest that the machine learning model may not provide accurate predictions for some of the classes.

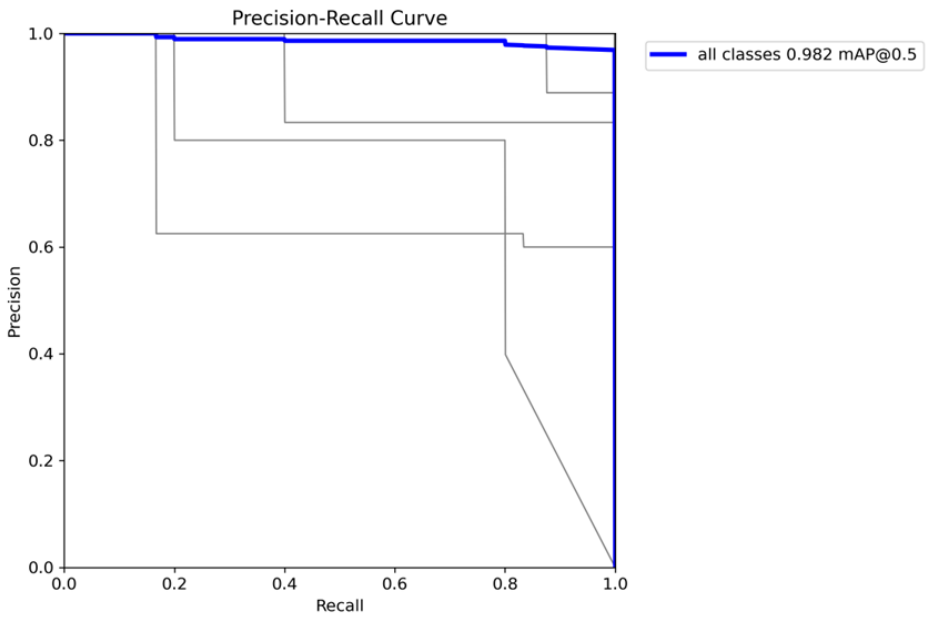


Figure 23 Precision-Recall Curve of model using NAdam

The above figure showcases the precision and recall trade-off of the model using NAdam. The representing precision-recall curve demonstrates a good performance: high precision value at different recall values. The mAP@0.5 obtained is equal to 0.982. However, the precision-recall curves for individual classes established a diversified and dispersed pattern. For some individuals, the precision value dropped at a recall level of 0.2. It implies that the trade-off between precision and recall for some of the classes may be dissatisfying. In other words, the detection results will be unreliable as the precision and recall is affected.

3.2.2.7 RMSProp

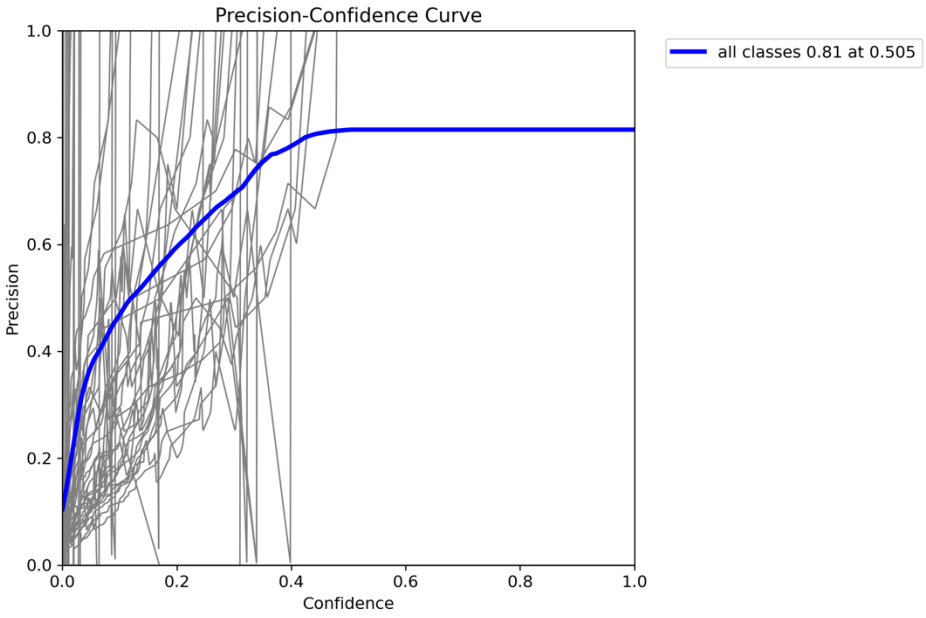


Figure 24 Precision-Confidence Curve of model using RMSProp

In Figure 277, the overall precision performance is demonstrated. The overall precision achieves a maximum of 0.81 at the confidence threshold value of 0.505. It indicates that the model is struggling to make highly confident and precise results. The grey curves representing individual classes display an unstable and fluctuating precision-confidence pattern, further indicating that the model training with RMSProp under the given conditions may not be capable of delivering accurate predictions.

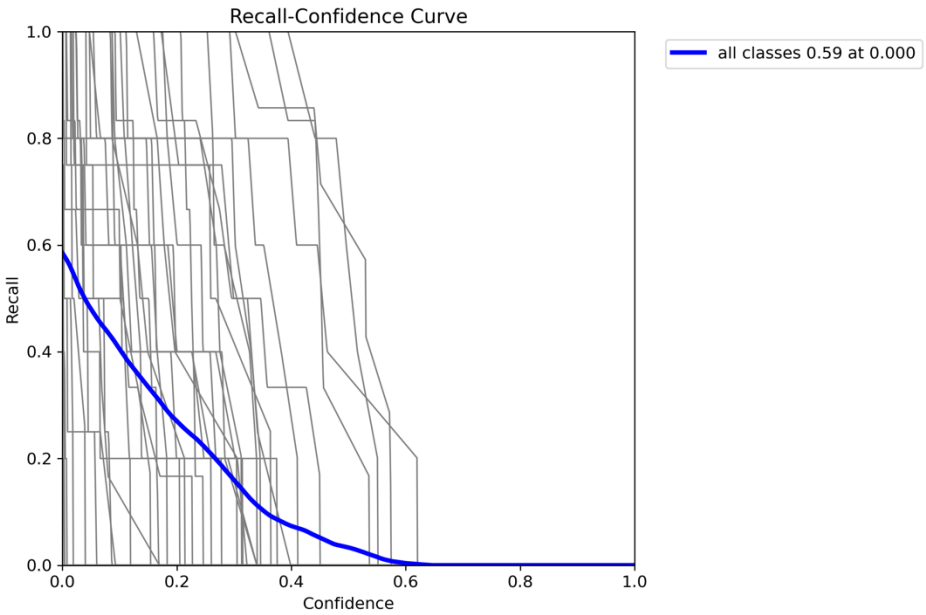


Figure 25 Recall-Confidence Curve of model using RMSProp

The Recall-Confidence Curve in Figure 28 has an exponential decay pattern. The overall class recall value dropped from 0.59 to 0.0 across the confidence threshold value. This suggests that the model is incapable of identifying or capturing positive instances and making accurate predictions.

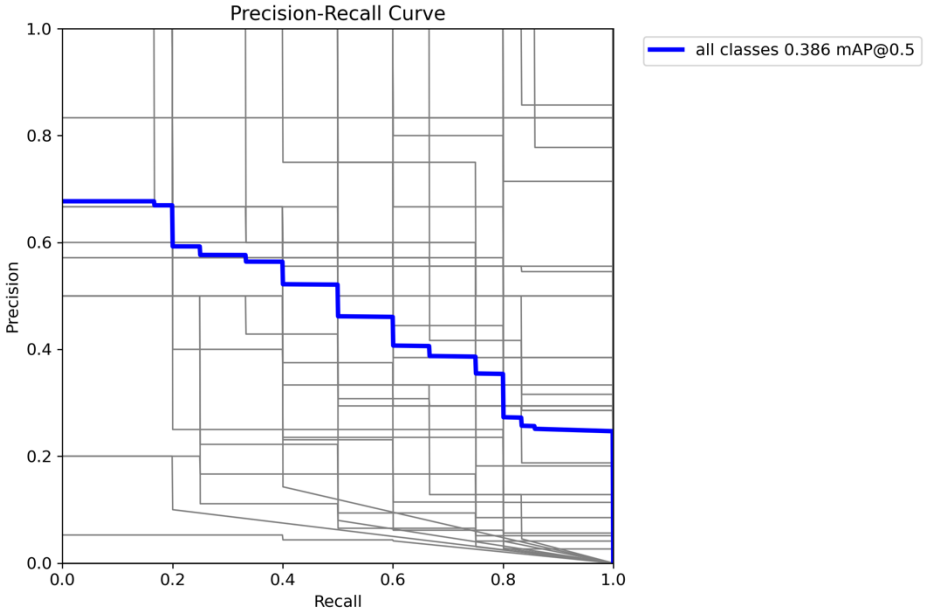


Figure 26 Precision-Recall Curve of mode using RMSProp

Figure 29 displays a mAP@0.5 value of 0.386 and a precision-recall trade-off curve that is non-smooth and exhibits low values for precision and recall. This indicates a poor performance of the

model, as reflected by the low mAP@0.5 value. The model's predictions do not meet the desired performance standards, being both inaccurate and incomplete.

3.2.2.8 Decision

After testing different optimizer algorithms, it was found that the model using SGD achieved the best results during training. Based on Table x, the model that used SGD attained the highest overall class precision score of 0.965. Both the precision and recall curves showed that the model using SGD was able to make accurate and sensitive predictions. Moreover, the precision-recall curve demonstrated an excellent trade-off between precision and recall. Therefore, we have decided to use SGD as the optimizer algorithm for our object detection model. Further hyperparameter fine-tuning will be performed in the future.

3.3 Mobile Application

The UI consists of 3 pages, implementing vision-based positioning, path finding, and accessing campus information respectively.

3.3.1 Vision-based positioning

As shown in Figure 30 to Figure 32, the app accesses the phone camera for image capturing. If the connection to the server fails, an error message will be prompted, as demonstrated in Figure 30. Figure 31 shows the reminder message to the users if the location detection continues to fail. Upon receipt of the location result from the server, as shown in Figure 32, a message will be prompted to indicate the detected location. Users can choose to keep capturing or go to the "Map" page with the detected location. For the latter option, the UI would direct to the "Map" page and set the detected location as the current location.

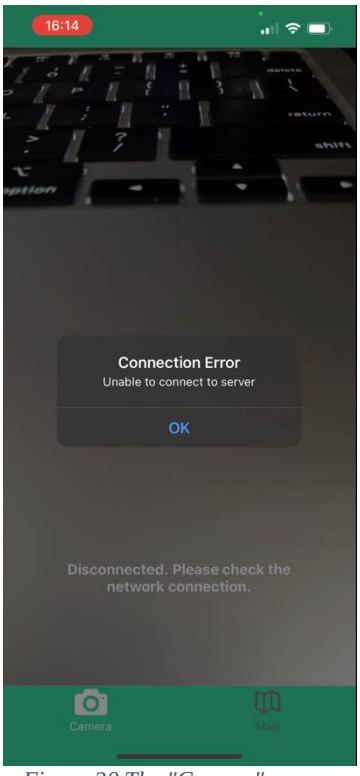


Figure 30 The "Camera" page – Connection error

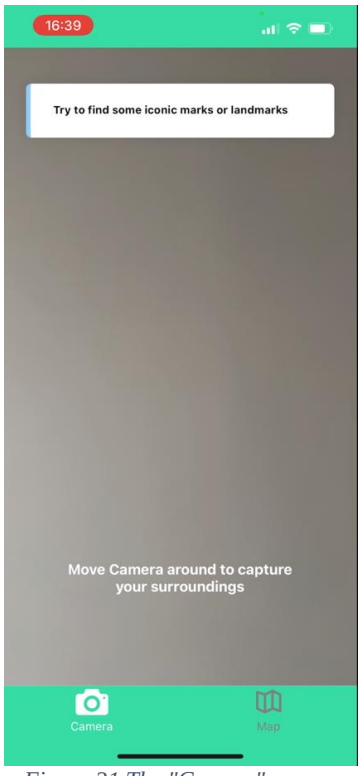


Figure 31 The "Camera" page – Reminder message

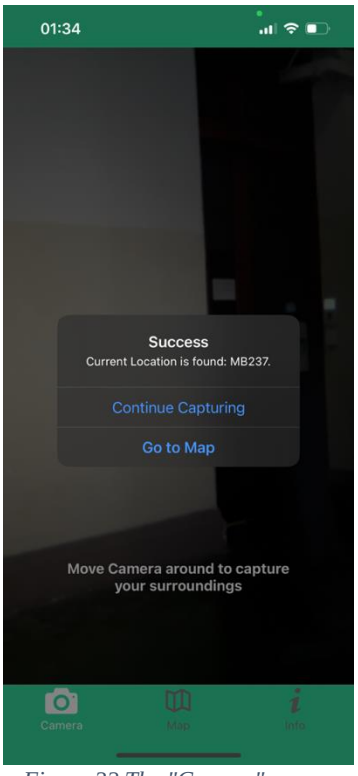


Figure 32 The "Camera" page – Successful detection

3.3.2 Path finding

As illustrated in Figure 33, the “Map” page allows users to input source location and destination. When users are inputting in the boxes, a suggestion list of venue names will be provided. Users have to choose from the suggestion list to finish inputting. This serves as a way of input validation. The navigation service will be integrated into the app to visualize the optimized path on the page.

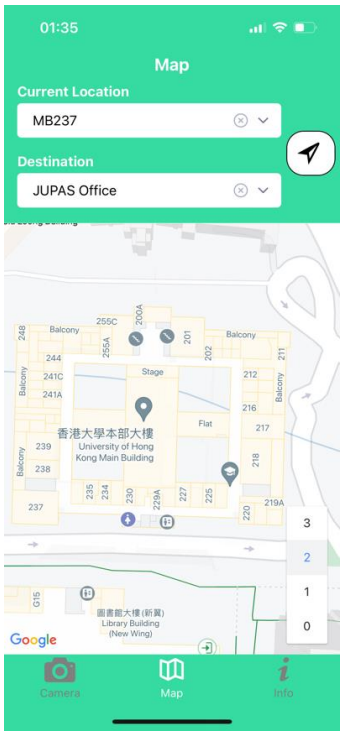


Figure 27 The "Map" page

3.3.3 Access to campus information

Figure 34 shows the “Info” page that provides users with various information about the campus of HKU. Basic information about the campus, including locations of buildings, departments, facilities, and transport, can be accessed through this page.

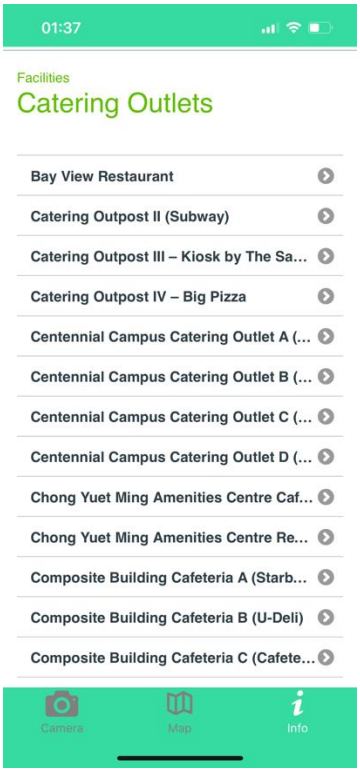


Figure 28 The "Info" page

3.4 Campus Map

The indoor map for the main building has been structured. A floor selector is provided for selecting the floor to display. Routes are added so that indoor navigation powered by FengMap can be used. The remaining works are labelling room names, optimizing layouts, and integrating the mapping and navigation service into the app.

Figure 29 Demonstration of an indoor map built using FengMap (2/F, Main Building)

4. Limitations and Difficulties

This chapter delves into the dataset size overload issue identified in the early stages, data collection inefficiency problem, and the location generalization issue, examining both the nature of the challenges and the responses devised to address them.

4.1 Dataset Size Overload

4.1.1 Problem of Dataset Size Overload

The large data size required for the project imposes significant challenges on its development. In particular, the capturing of images from various angles, heights, and times at the same node on the map adds to the complexity. Furthermore, the quest for higher accuracy necessitates a substantial number of images for model training. Consequently, the development of an accurate positioning model for the entire HKU campus may require an extensive dataset comprising millions of images. The magnitude of data involved imposes rigorous demands on computational power, storage capabilities, and human resources. Notably, these requirements surpass the scope of a typical final year project, warranting careful consideration and resource allocation.

4.1.2 Response to Dataset Size Overload: Limiting Scope

The primary mitigation strategy is to limit the scope of the project. Instead of encompassing the entire HKU campus, the project will concentrate on a specific section. The Main Building of HKU has been selected as the designated testing site for the project. By focusing on the Main Building, the project aims to achieve the goal of effectively navigating visitors within its premises. By adopting a focused approach and considering potential future expansions, the project aims to strike a balance between the limited resources available and the aspiration to deliver an effective navigation solution within the Main Building of HKU.

4.2 Inefficiency of initial Data Collection

4.2.1 Problem of Initial Data Collection

An experiment was conducted to examine the efficiency of the initial data flow design. Table 1 below shows the times used for different numbers of images in the 4 phases.

	No. of Image	Time used			
		Data Collection	Data Processing	Data Labelling	Data Augmentation
Set I	40	5	5	10	3
Set II	117	14	7	15	4
Coefficient	2.93	2.80	1.40	1.50	1.33

Table 3 Time used for 40 images and 117 images in the 4 data flow phases

The findings demonstrate that the time spent on the data collection phase exhibits the most substantial growth. The coefficients associated with the number of images and data collection, highlighted in Table 3, are closely aligned, suggesting that the time required for data collection gradually increases as the dataset size expands. Leveraging existing tools aids in mitigating the proportional growth of time in data processing, data labelling, and data augmentation. However, due to the manual image acquisition on an individual basis, a linear relationship emerges between the number of images and the time spent on data collection. The experiment highlights the inefficiency of the original data collection method, consequently diminishing the value of the vision-based approach.

4.2.2 Response to Inefficient Data Collection: Videos instead of Photos

The initial data collection process encountered inefficiency as images were gathered on an individual basis. However, given the importance of both image quality and quantity, it became evident that obtaining data from primary sources was necessary. Consequently, a new and improved method was implemented, which involved capturing videos and subsequently extracting frames from these videos. This approach ensures a more efficient and comprehensive collection of data for the project, allowing for a greater variety of images to be obtained while maintaining the desired level of quality.

Using the new data collection method, the time used for capturing training data gradually reduced. Table 4 shows the improved time used for data collection.

	No. of Image	Time used in Data Collection	Image per minute
Initial Method	117	14	8.36
New Method	165	3	55

Table 4 Time cost comparison of the initial and new data collection methods

4.3 Location Generalization

4.3.1 Problem of Location Generalization

To improve the accuracy of object detection, we have designed a model that recognizes specific and unique features in images which can provide a brief idea of the location of the user. However, this approach poses a limitation on the direct prediction of location. Multiple objects can be detected in the same frame, and the detection results can be misleading and cannot be associated with the exact location.

4.3.2 Proposed Response to Location Generalization: Additional Layer

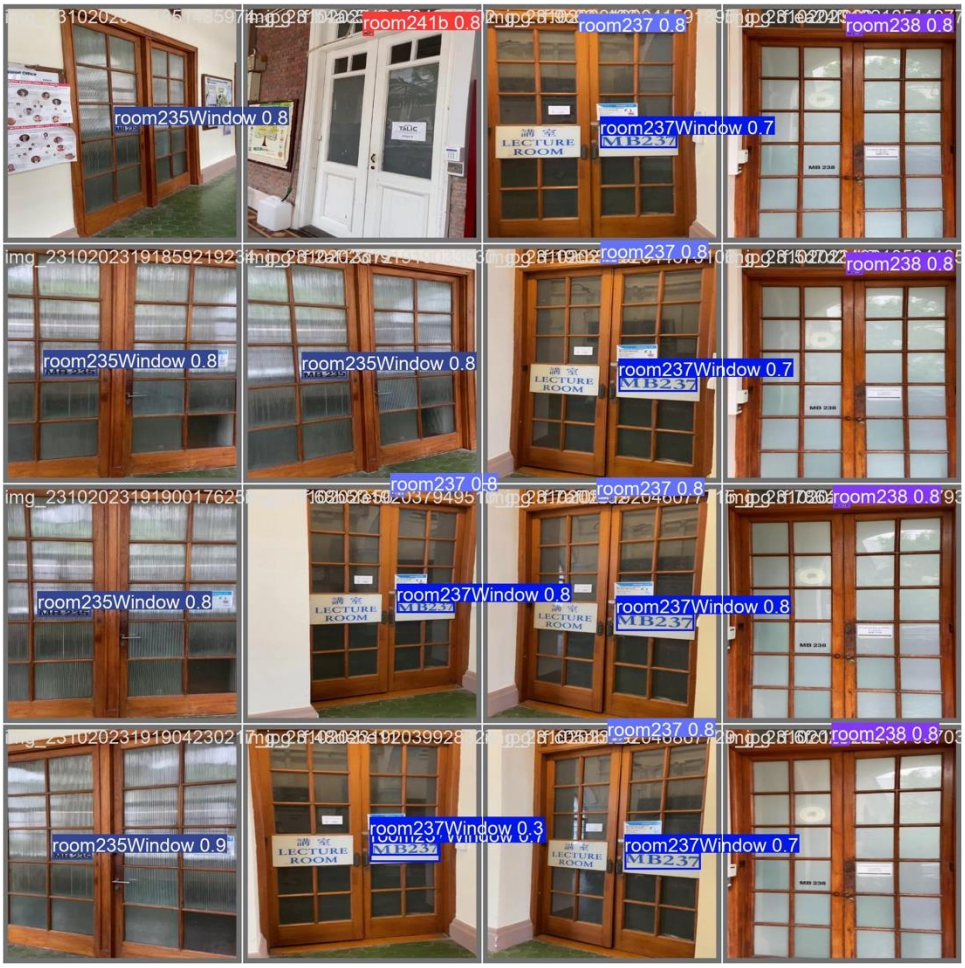


Figure 30 Detection results example of using trained model

One of the improving methods is to impose confidence score filtering on the object detection model. As seen in Figure 36, multiple objects can be detected in the same image with different confidence scores. By applying confidence score filtering, some predictions with low confidence scores (i.e.

room237window 0.3) can be screened, reducing the misleading detection results and the complexity of understanding the location of the user.

Another mitigation will be creating a specialized function that integrates various factors such as confidence scores, detected class names, and appearance on frames. This function will analyze and summarize the user's location information and provide a generalized view of their whereabouts. This approach will help us to mitigate potential errors and improve the overall quality of our predictions.

5 Schedule and Future Plan

5.1 Schedule

The project progress, as depicted in Table 5, generally aligns with the proposed schedule. The initial implementations of the mobile application and the CV model have been completed as planned. The construction and optimization of the indoor map are still in progress. Prioritizing indoor map development and integration, along with exploring various approaches to improve model's accuracy, are crucial for the upcoming stages of the project.

Period	Work Description	Progress
Sep - Oct	Analyse different proposed ideas' feasibility and effectiveness	Done
	Research and literature review of computer vision-based positioning	Done
	Design interface for mobile app	Done
Oct – Nov	Collect data in Main Building (2nd Floor)	Done
	Research in different computer vision models	Done
	Develop mobile app (both frontend & backend)	Done
	Data processing	Done
Nov – Dec	Build and train shortlisted machine model	Done
	Evaluate and select model with best performance	Done
Dec - Jan	Prepare interim report and presentation	Done
	Prepare rough demo for interim presentation	Done
Jan - Feb	Select and implement suitable navigation mechanism	In Progress
	Integrate machine learning model and map & navigation mechanism to mobile application	Starting Soon
	Fine tuning the model	In Progress
Feb - Mar	Debug and improve mobile application and model	Starting Soon
	Deploy mobile application for alpha testing	Starting Soon
Mar - Apr	Prepare final report	Starting Soon

	Source code cleanup	Starting Soon
	Prepare for final presentation	Starting Soon
	Prepare for the project exhibition and project competition	Starting Soon

Table 5 Project Schedule Table

5.2 Future Plan and Directions

For the object detection model, the next goal is to achieve at least 97% accuracy with further hyperparameter tunning, given that the current best model can achieve approximately 95% accuracy. To prove the reliability of our model, our next goal is to maintain high accuracy during the dataset expansion. In order to have a direct insight on the ability of our model, setting up a chatGPT4 model to compare with our model is a feasible next step.

In terms of the navigation map, a more complete indoor building map is required for accurate positioning. One of the potential implementations is to cooperate with ManiFold Tech Limited, a company with experience and equipment for object scanning, and explore the extendibility of our project with an Augmented Reality (AR) map.

For the application development, a few additional features are possible to integrate with the app. For example, an image upload recognition function. It allows users to upload images for positioning instead of real time location rendering.

6 Conclusion

This project aims to develop an effective vision-based navigation and contribute to the research field of computer vision applications. The navigation system, once fully developed, will enhance visitor experience, providing accurate and efficient navigation within the HKU campus. This system will greatly benefit individuals unfamiliar with the building's layout and those with special navigation needs. Moreover, the project contributes to the advancement of indoor navigation technologies, particularly within building complexes, serving as a valuable reference for future endeavours in this domain.

The paper presented an in-depth analysis of the methodologies employed and the outcomes achieved in designing the system, app, and model. The implementation of the mobile application and backend service has been finalized, ensuring a solid foundation for the project. However, careful considerations and cost-effectiveness studies are required for the development of the mapping and navigation functionality. Utilizing a dataset comprising 2264 collected images, an object detection model has been successfully trained using the SGD optimizer. To enhance accuracy for navigational purposes, a proposed approach involves an additional layer to the CV model. This is because object detection models often overlook important positioning factors such as distance and capturing angle. The availability and efficiency of data collection play significant roles in determining the cost-effectiveness of the project. As the dataset size increases, the time spent in the data collection phase can become a dominant factor. Therefore, it is important to carefully manage data acquisition and minimize the required dataset size to optimize project costs. Overall, the project's progress is generally in line with the schedule, signifying a positive advancement toward the desired objectives.

Looking ahead, the immediate next steps involve improving functionality and implementation of the app and the backend service. Another important step is prioritizing the completion and integration of the indoor map with the outdoor map and the mobile application. Due to the model deficiency, attention will also be given to research and comparing different proposed model implementation. Additionally, research in optimizing the dataset size for model training will continue to address the dataset size overload issue.

References

[1]Communications and Public Affairs Office, “HKU Quick Stats 2022”. https://www.cpao.hku.hk/qstats/files/Archive/2022.pdf (accessed Sep 19, 2023).

[2] H. Motte, J. Wyffels, L. De Strycker, and J. P. Goemaere, “Evaluating GPS data in indoor environments,”.https://www.researchgate.net/publication/269978666_Evaluating_GPS_Data_in_Ind oor_Environments. Advances in Electrical and Computer Engineering, vol. 11, no. 3, pp. 25–28, Jan. 2011, doi: 10.4316/aece.2011.03004. (accessed Sep 21, 2023).

[3] T. Wu, L.-K. Chen, and Y. Hong, A Vision-Based Indoor Positioning Method with High Accuracy and Efficiency Based on Self-Optimized Ordered Visual Vocabulary, https://lightweb.ie.cuhk.edu.hk/api/publication/1658736041110- 07479682.pdf (accessed Sep. 21, 2023).

[4] D. Khan, Z. Cheng, H. Uchiyama, S. Ali, M. Asshad, and K. Kiyokawa, “Recent advances in vision-based indoor navigation: A systematic literature review,”.https://www.sciencedirect.com/science/article/abs/pii/S0097849322000371. Computers & Graphics, vol. 104, pp. 24–45, May 2022, DOI: 10.1016/j.cag.2022.03.005. (accessed Sep 22, 2023).

[5] C. Wiegand, “Achieving Blue Dot: Best types of indoor positioning systems,” Mar. 22, 2023. https://www.inpixon.com/blog/what-is-the-best-system-for-achieving-blue-dot- indoors

(accessed Sep 22, 2023).

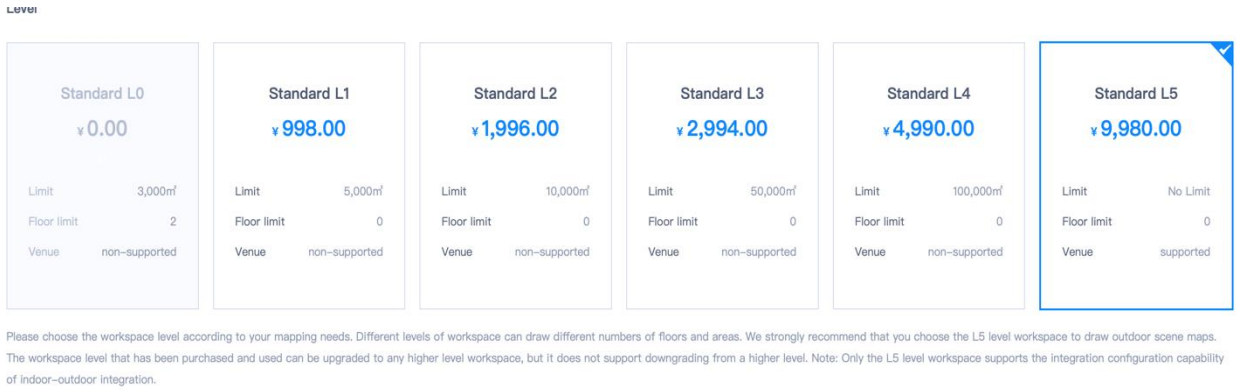
[6] S. Ahmad, “How is Mobile Computer Vision Changing the World?,” Mobisoft Infotech, Dec. 01, 2020. https://mobisoftinfotech.com/resources/blog/how-is-mobile- computer-vision-changing-the-world/ (accessed Sep 23, 2023).

[7] Grigorescu, Sorin & Trasnea, Bogdan & Cocias, Tiberiu & Macesanu, Gigel. (2019). A survey of deep learning techniques for autonomous driving. Journal of Field Robotics. 37. 10.1002/rob.21918. (accessed Dec 21, 2023)

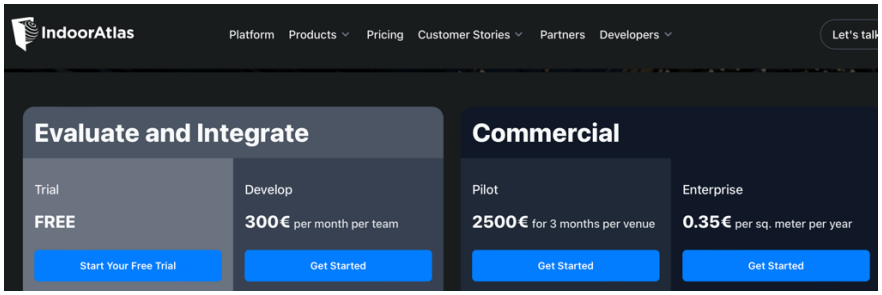
[8] Q. Lin, G. Ye, J. Wang, & H. Liu, “RoboFlow: a Data-centric Workflow Management System for Developing AI-enhanced Robots.” Proceedings of the 5th Conference on Robot Learning, PMLR 164:1789-1794, 2022. https://proceedings.mlr.press/v164/lin22c.html. (accessed Nov 30, 2023).

Appendices

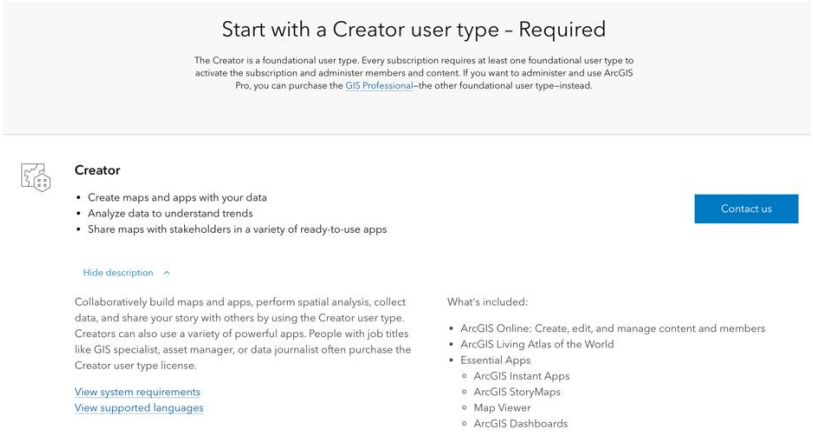
Appendix A – FengMap Pricing



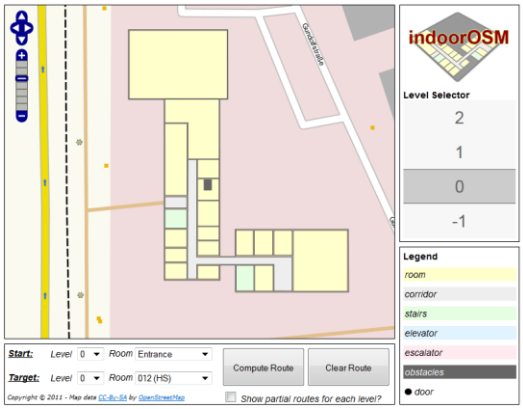
Appendix B – IndoorAtlas Pricing



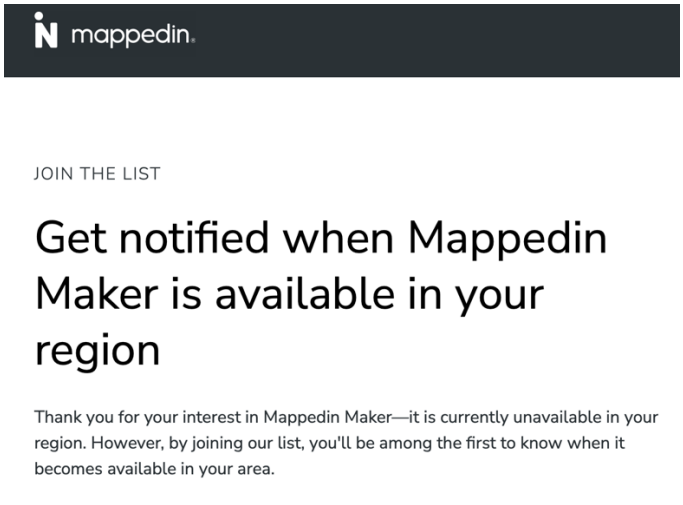
Appendix C – ArcGIS Pricing



Appendix D – Indoor Map of Demonstration of OpenStreetMap



Appendix E – Inaccessibility of MappedIn



Appendix F – Used parameters of model training

```
task: detect
mode: train
model: yolov8n.pt
epochs: 300
patience: 20
batch: 16
imgsz: 640
save: true
save_period: -1
cache: false
device: 0
workers: 8
project: null
name: train
exist_ok: false
pretrained: true
verbose: true
seed: 0
deterministic: true
single_cls: false
rect: false
cos_lr: false
close_mosaic: 10
resume: false
amp: true
fraction: 1.0
profile: false
freeze: null
overlap_mask: true
mask_ratio: 4
dropout: 0.0
val: false
split: val
save_json: false
save_hybrid: false
conf: null
iou: 0.7
max_det: 300
half: false
dnn: false
plots: true
source: null
vid_stride: 1
stream_buffer: false
visualize: false
augment: false
agnostic_nms: false
classes: null
retina_masks: false
show: false
save_frames: false
save_txt: false
save_conf: false
save_crop: false
show_labels: true
show_conf: true
show_boxes: true
line_width: null
format: torchscript
keras: false
optimize: false
int8: false
dynamic: false
simplify: false
opset: null
workspace: 4
nms: false
lr0: 0.01
lrf: 0.01
momentum: 0.937
weight_decay: 0.0005
warmup_epochs: 3.0
warmup_momentum: 0.8
warmup_bias_lr: 0.1
box: 7.5
cls: 0.5
dfl: 1.5
pose: 12.0
kobj: 1.0
label_smoothing: 0.0
nbs: 64
hsv_h: 0.015
hsv_s: 0.7
hsv_v: 0.4
degrees: 0.0
translate: 0.1
scale: 0.5
shear: 0.0
perspective: 0.0
flipud: 0.0
fliplr: 0.5
mosaic: 1.0
mixup: 0.0
copy_paste: 0.0
cfg: null
tracker: botsort.yaml
save_dir:
runs\detect\train
```