Advanced road analysis

Graduation project 2024

Team Member

Advanced road analysis

Abdelrahman Moustafa Abd El Aziz Moussa	202000538
Mariiam Amr Fawzy Mohamed	202000881
Diaa Hany Ahmed Abd El Monam	202000475
Abdelrahman Mohamed Youssef Zahran	202000534
Toqa Osama Hassan Ahmed Badran	202000229
Reem Rada Abd EL Hamed Rashwan	202000348

Supervised by:
Prof/Hala Abd El Galil

Agenda

01
Overview

02
Problem Statement

03
Work Methodology

i. Deep Learning
ii. GIS Mapping

04
Software

05
Conclusion

06
Results

07
Future Work

01

Overview

Total description for the project workflow and included features

Our Application is aims to give the driver safe driving experience from road cracks as it cause server damges.

This Application enable user to get an alram for each upcoming crack ahead before it appears infront of him on road.

Application has also the abilty to gather crack data .

This Application used Deep learning machine learning model and Mapping technologies to achieve it capabilites.

02

Problem Statement

The problems we focus on our project

Road infrastructure cracks problems

- Road cracks are a pervasive issue that pose significant challenges to
- both infrastructure and safety.
- These cracks, which can arise from a variety of factors such as weather conditions, heavy traffic, and substandard construction materials, lead to the deterioration of road surfaces over time.
- The presence of cracks not only accelerates the wear and tear of vehicles but also increases the likelihood of accidents,
- as drivers may lose control when navigating uneven surfaces.

Road Signs Awareness problems

- Imagine a world devoid of traffic signs. Intersections would transform into scenes of anarchy, with drivers locked in a perpetual game of chicken, unsure of who has the right-of-way. Speed limits would become a forgotten concept, with reckless drivers posing a constant threat to pedestrians and other vehicles.

03

Working Methodologies

The proposed working solution to over come the problems mentioned

Deep Learning

Crack detection model

- Our goal is to make a reliable deep learning model to be our base for detecting road cracks on streets so we can locate them on Map after that .

Crack detection model

Work methodology references

YOLO V2 model

- The model used here was yolo V2
- Dataset images 7240 images
- Dataset gathered by mobile camera
- F1 score accuracy is 87 % without classification
- F1 score accuracy is 79 % with classification
- Training time 18.2 hours.

Faster RCNN

- Yolo v2 model is less 5 % than it
- mAp : 41-44% @0.5 , 35-37% @0.5-0.95.
- Speed : 7-10 FPs on Navida RTX 3080
- Inference time : 100-143 ms per image
- Two stage with separate region proposal and classification stage

Crack detection model

Work methodology reference

EfficientDet

- 1-Single stage using focal EfficientNet backbone and BiFPN
- 2-mAp : 50-52 % @0.5 , 40-43% @0.5-0.95.
- 3-Speed : 18-22 FPs on Navida RTX 3080
- 4-Inference time : 45-55 ms per image

RetinaNet

- Single stage using focal loss to improve performance.
- mAp : 49-51 % @0.5 , 39-41% @0.5-0.95.
- Speed : 12-15 FPs on Navida RTX 3080
- Inference time : 67-83 ms per image

Crack detection model

Work methodology reference

Single shot detector

- Single stage , using myti-scale feature maps
- mAp : 43-46% @0.5 , 35-38% @0.5-0.95.
- Speed : 20-25 FPs on Navida RTX 3080
- Inference time : 40-50 ms per image

YOLO V8 model

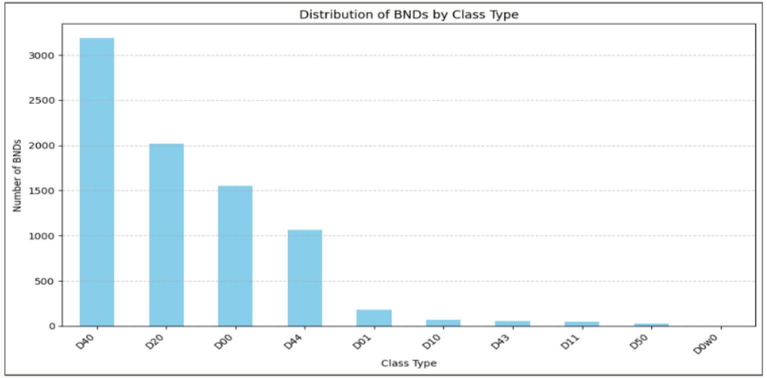
- Single stage , unified for detection and classification
- mAp : 52-55% @0.5 , 40-42% @0.5-0.95.
- Speed : 40-50 FPs on Navida RTX 3080
- Inference time : 20-25 ms per image

Crack detection model

DATASET

Analysis

Dataset consists of 10 classes has different types of cracks.
Our Focus on only the D40 and D20 which cause accidents and damages other than that was not important to us.
Dataset gathered by mount vehicle method.
Dataset has Training folder contains : 7706 images Annotated as Pascal Voc format.
Problem with this dataset that is does not Have many images to depend on so we Merged another dataset that gathered by same method but gathered from Japan



Crack detection model

DATASET

Pre-processing

The dataset has some problems that needed to be fixed :

1) Annotated by Pascal VOC format
2) Resizing
3) Data Augmentation
4) Data Ratio

Crack detection model

Experimental Training

Training parameters

1. Optimizer : Adam
2. Learning rate : lr0 by 0.001 and final 0.01
3. Batch size : 8
4. Epochs : 50 epoch
5. Image size : 704 *704
6. Intersection over union : 0.7
7. Box : 7.5
8. Class : 0.5
9. Dfl : 1.5
10. Training time : 12.5
11. Trained on CPU

Evaluation

1. F1 score : 85.4 % with classification
2. Precision : 88 %
3. Recall : 83.5%
4. mAp50 : 70%
5. mAp50-95 : 64 %

Sign detection model

Dataset

Analysis

Classifying the cracks into 15 types
Gathered by mounted vehicle method
Contains 4969 images

Data pre-processing

Image resizing
Formatting
Data ratio

Sign detection model

Experimental Training

Training parameters

1. Optimizer : Adam
2. Learning rate : lr0 by 0.001 and lrF 0.01
3. Batch size : 8
4. Epochs : 60
5. Image size : 416* 416
6. Intesection over unio : iou 0.7
7. Box : 7.5
8. Class : 0.5
9. Dfl : 1.5
10. Training time 7.45
11. Trained on GPU

Evaluation

1. F1 score :81 % with classification
2. Precison :88 %
3. Recall : 75 %
4. mAp50 : 85 %
5. mAp50-95 :73%

GIS Mapping

GIS Mapping Brief.

Choosing the Right Approach for Road Crack and Sign Visualization and Data Management

This guide explored various approaches to visualizing road cracks and signs, defining geodatabase schemas, and managing geospatial data. The best method depends on your specific needs and resources.

Key Takeaways:

- **Visualization:** Choose point-based representation for a quick overview or imagery techniques for detailed analysis of crack and sign characteristics.
- **Map Creation:** Tailor your map to the analysis goal and audience. Gather necessary geospatial data layers, ensuring accuracy, compatibility, and permissions. Prepare the data and design the map with clear symbols, labels, and essential elements.
- **Geodatabase Schema:** Define a schema to structure your data effectively. Include location information and relevant attributes for cracks and signs. Set appropriate data types and validation rules. Document everything for clarity.

GIS Mapping Brief.

Key Takeaways:

- **Feature Class Definition:** Choose a descriptive name for the feature class and define the geometry type representing crack/sign locations. Assign suitable data types for each attribute and establish a spatial reference system. Consider optional domains and validation rules for data control.
- **Adding Features:** Develop a systematic method for adding new features. This includes selecting data collection methods (visual inspection, mobile apps, or remote sensing), establishing data entry procedures with clear standards, implementing quality assurance checks, and documenting the entire process.

By following these steps and considering your specific needs, you can effectively visualize road cracks and signs, manage your geospatial data, and gain valuable insights for infrastructure maintenance and road safety improvements.

Map Construction

- Define Map Purpose and Audience
- Map Layout and Design
- Base Layer Selection
- Data Integration
- Symbology and Labeling

Survey

- Ground Survey Methodologies
- Types of Ground Surveys

Integrations

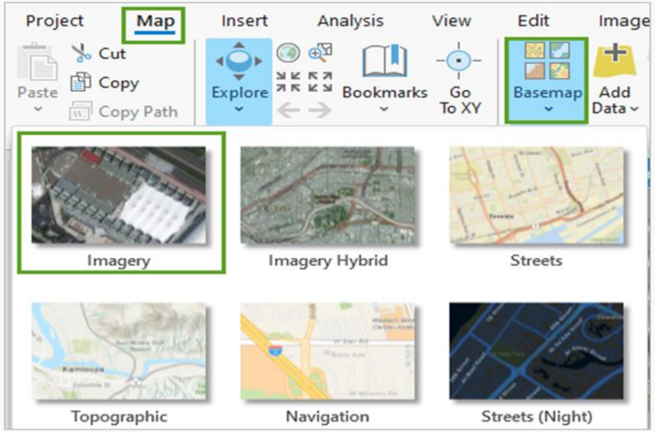
- Integration with AI
- Integration With Android App

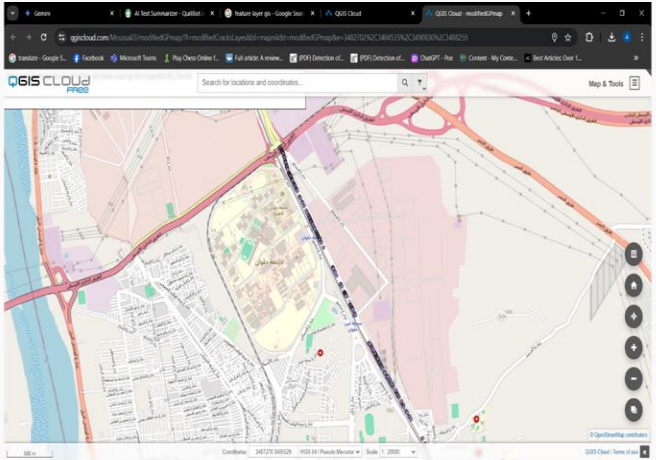
GIS Data Handling

- Importing Data to GIS software
- Processes and manipulation

Map Construction

1. Define Map Purpose and Audience:
   a. Determine the overall goal of the map.
   b. Consider the needs of the target audience.
2. Base Layer Selection:
   a. Choose a base map that provides context and aids visualization.
   b. Detailed street map for precise location reference.
3. Data Integration:
   a. Import existing geospatial data relevant to your analysis.
   b. Road network data (essential for crack and sign placement).
4. Symbology and Labeling:
   a. Design clear and informative symbology for base map elements (roads, boundaries).





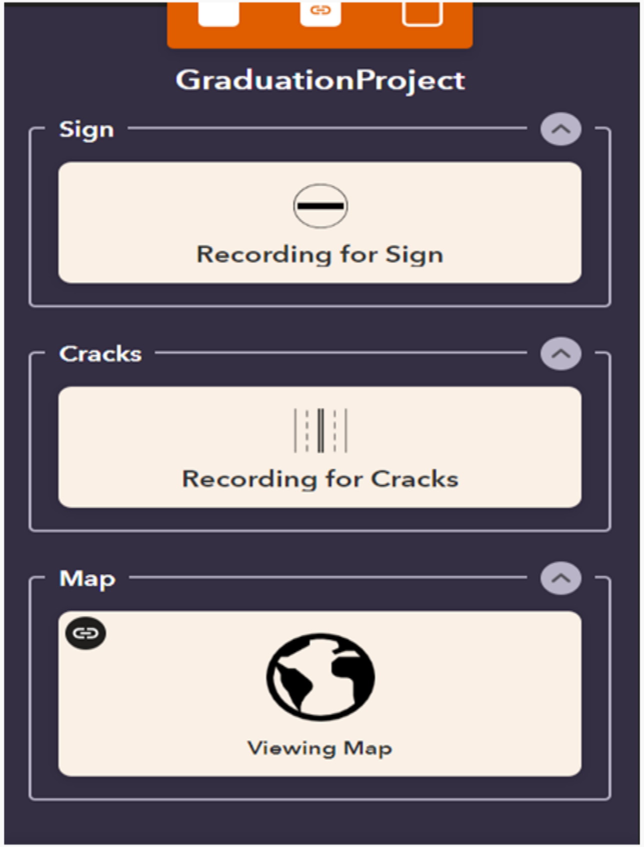
Survey

- Choosing Survey Type to Match Limited Resources.
  - Ground Survey Methodologies:
    - Traditional Manual Surveys.
    - LiDAR Surveys (Light Detection and Ranging).
    - Drone-Based Survey.

- Selection Considerations:
  - Factors include project scope, budget, terrain complexity, and data accuracy requirements.
  - Each survey type has advantages and limitations.

Survey Type And QuickCapture

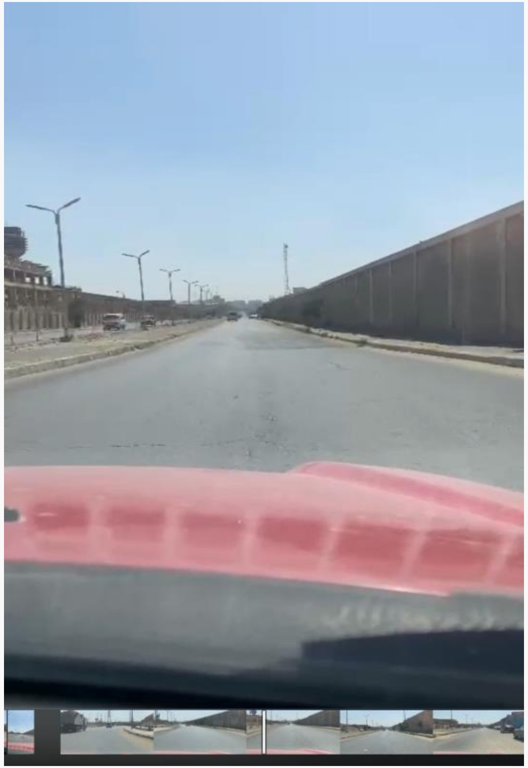
- Esri's QuickCapture app is used for efficient data collection of road crack and sign data. It considers resource limitations, provides comprehensive training, and captures essential attributes. Field teams record real-time geospatial data, ensuring accuracy through monitoring and validation measures.



Scanning Specific Road:

- Selecting a road having the same features as the the ML model's data.

- The scanned road is the industrial road west of Helwan university.



Integration With AI

CSV File for Anomalies.

The machine learning model converts road anomalies into structured data, including type, location, frame number, and time, which is then exported into a CSV file for analysis and review for accuracy

- Steps:
  - Compile detected anomalies into a structured format.
  - Include fields like anomaly type, location (coordinates), frame number, and timestamp.
  - Export this data to a CSV file.
  - Review the CSV file for completeness and accuracy.
  - Store the CSV file for further integration with GIS.

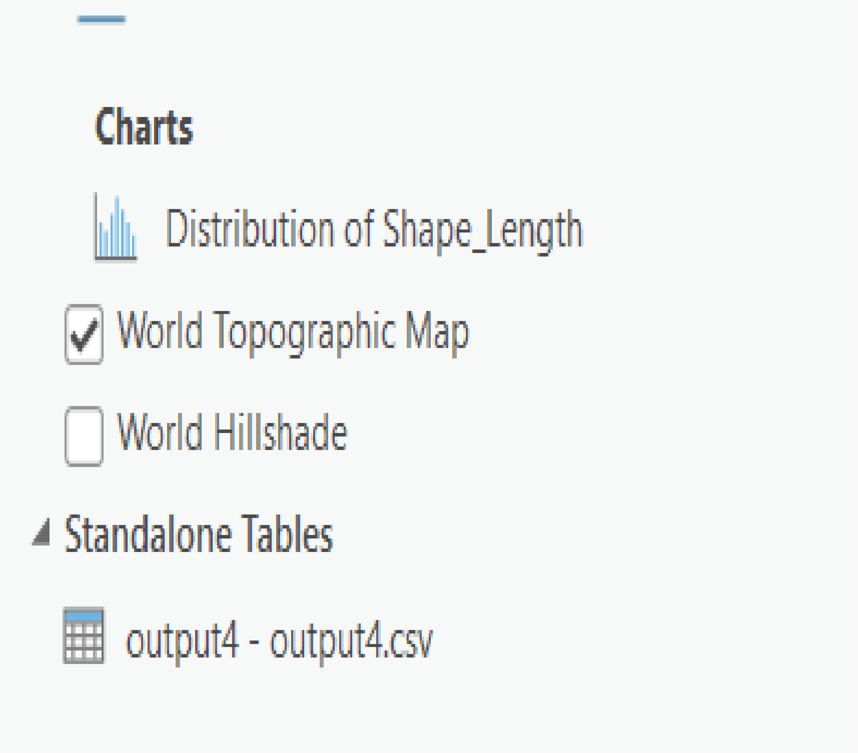
Data Handling

Data handling consists of multiple processes :

1-get a CSV having the location of the cracks or signs, type, and time.

2-uploading this CSV into ArcGIS pro .

3-we need to match the number of records of CSV with the number of records of feature class .



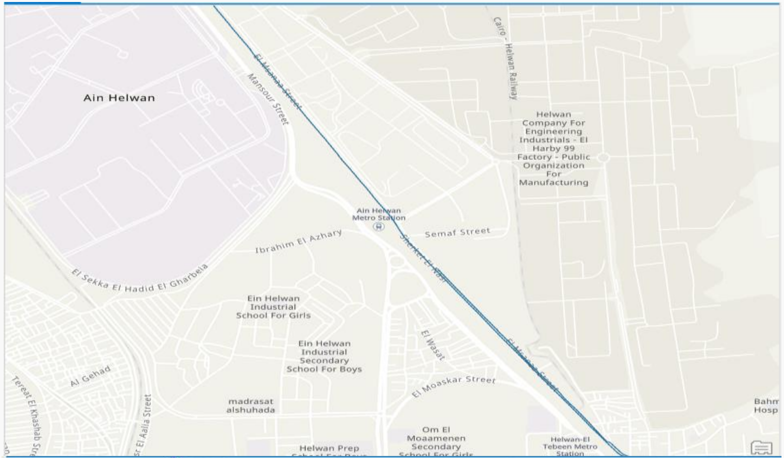
Data Handling

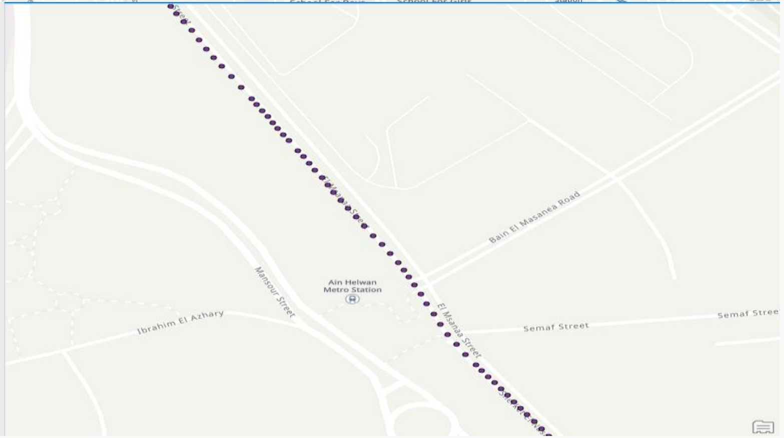
Data handling consists of multiple processes

4-increasing the number of points on the map by the following:

1-convert line layer by connecting the points to create 1 line

2-divided the created line into points but with the same number of CSV records





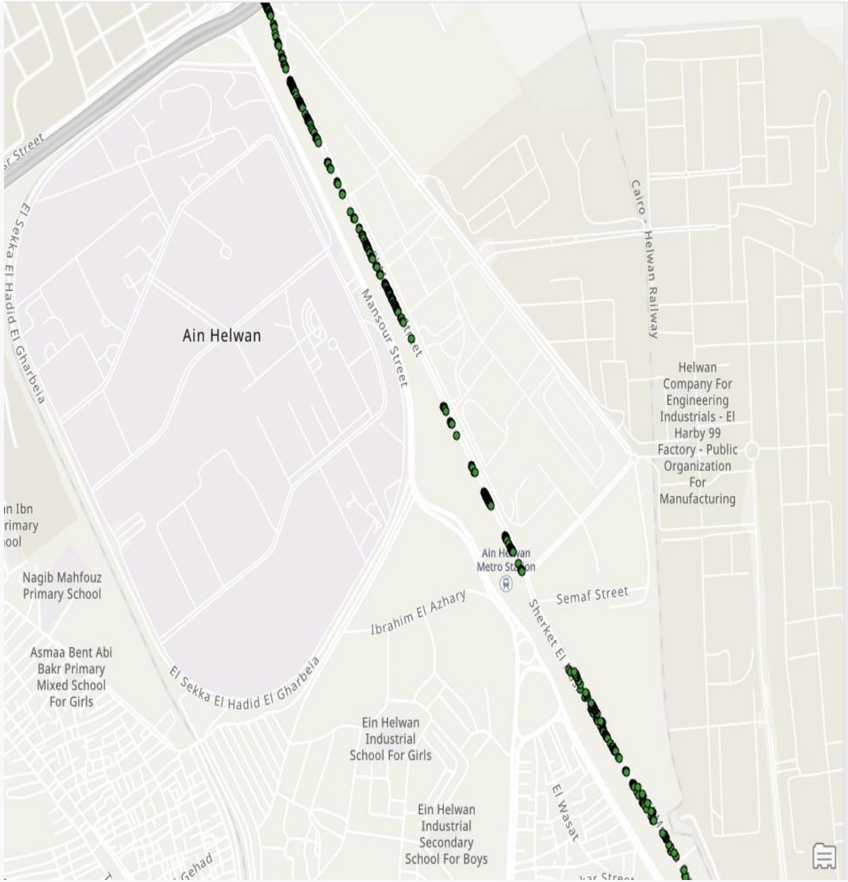
Data Handling

Data handling consists of multiple processes :

5-merging the CSV with the feature class (geodatabase) based on the calculated field having the same number and sequence in both the feature class and CSV file

6-filtering the points that contain a crack from all recorded points

7-deleting unnecessary points creating the final feature layer having the points with their detected objects (cracks and signs)



Integration With Android App

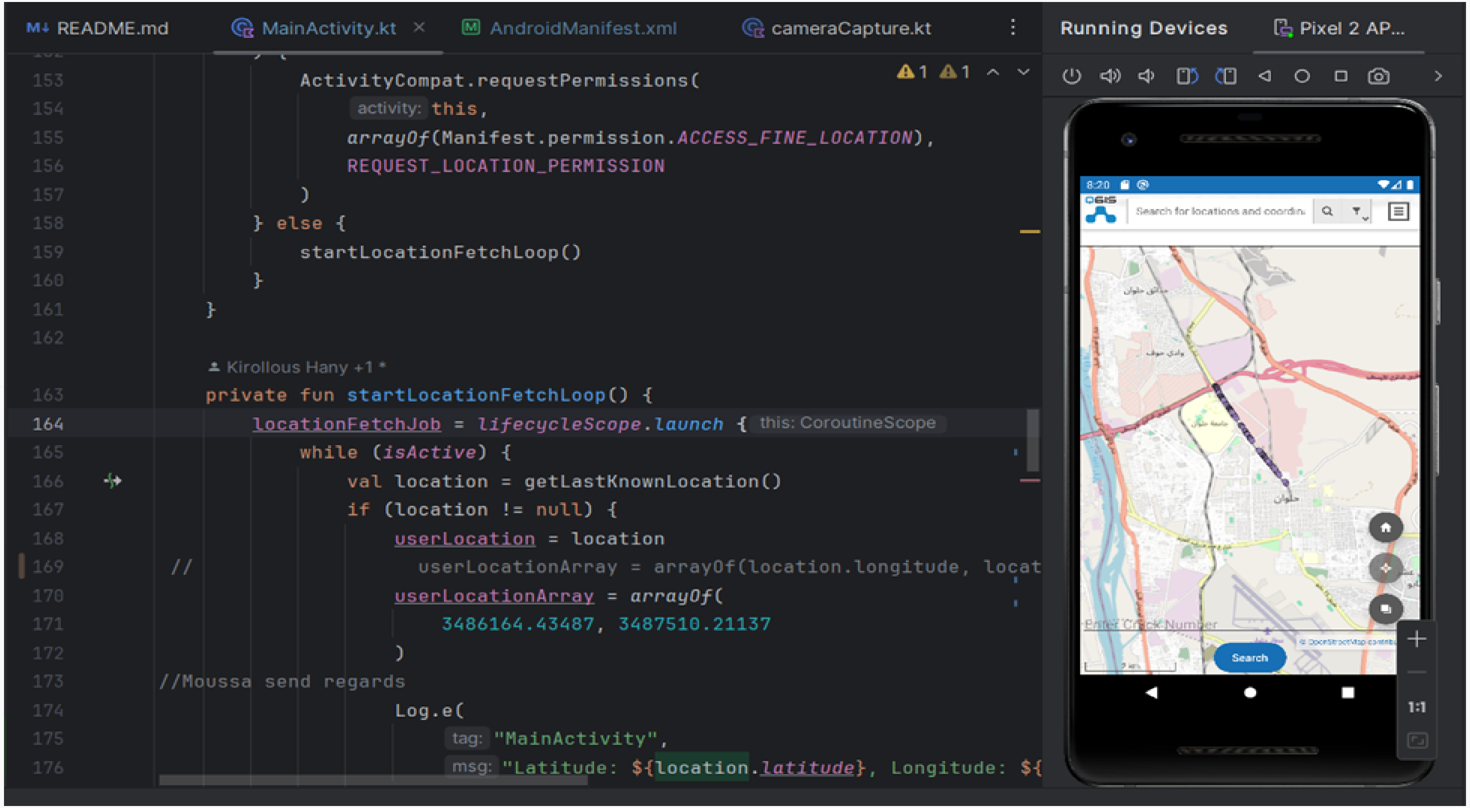
Map deployment

An Android application can integrate with a web map containing road crack and sign data for improved accessibility and interaction. APIs facilitate communication, and real-time updates and offline access functionality can be added. User-friendly navigation and intuitive interaction are crucial for a seamless experience.

Steps:

- Develop or update an Android application to interact with the web map.
- Use APIs to fetch and display the web map within the app.
- Implement features for real-time data updates and offline access.
- Ensure the app has user-friendly navigation and interaction capabilities.
- Test the app thoroughly and deploy it to users.

Integration With Android App



Integration With Android App

Data Retrieving

The feature data, which includes geographic coordinates, is extracted from the response. This data is converted into a format suitable for further processing (e.g., a 2D array of longitude and latitude values).

- **Steps:**

1. Setup the Network Client.
2. Define API Endpoints.
3. Make the API Call.
4. Handle the Response.


```json
{
    "features": [
        {
            "geometry": null,
            "id": "modifiedCracksLayes.0",
            "properties": {
                "boody": 3,
                "boody_1": 3,
                "date": "2024-03-10",
                "detection": "1 D20",
                "frame_numb": "3/29204)",
                "latitude": 3487510.21137,
                "longitude": 3486164.43487,
                "orig_fid": 1,
                "qc_id": 0,
                "time": "11:32:46",
                "timestamp": "2024-03-10"
            },
            "type": "Feature"
        }
```

Integration With Android App

User Location Retrieving

- Utilizes Location Services Client for device location access.
- Handles permissions and requests necessary permissions if necessary.
- Regularly updates user's location at regular intervals for the most recent data.
- Performs location fetching asynchronously within a coroutine to maintain main thread responsiveness.

```kotlin
    private fun startLocationFetchLoop() {
        locationFetchJob = lifecycleScope.launch {
            while (isActive) {
                val location = getLastKnownLocation()
                if (location != null) {
                    userLocation = location
                    //        userLocationArray = arrayOf(location.longitude, location.latitude)
                    userLocationArray = arrayOf(
                        3486164.43487, 3487510.21137
                    )
    //Moussa send regards
                    Log.e(
                        "MainActivity",
                        "Latitude: ${location.latitude}, Longitude: ${location.longitude}"
                    )
                    checkProximityToPoints()
                } else {
                    Log.e("MainActivity", "Failed to get location.")
                }

                delay(locationFetchInterval)
            }
        }
    }
```

Integration With Android App

Alarming

Proximity Check in App

• Calculates distance between user's current location and predefined points using Haversine formula.
• Checks if distance is within specified threshold (e.g., 500 meters).
• Triggers alarm if user is within this range.

Alarm Trigger Process

• Alarm activation: When proximity criteria are met, an alarm is triggered.
• Media Player Handling: A media player instance plays the alarm sound for a fixed duration.
• Proper management of media player to prevent memory leaks.

```kotlin
private fun checkProximityToPoints() {
    var location = 0
    for (point in pointsArray) {
        val distance = calculateDistance(
            userLocationArray[1], userLocationArray[0],
            point[1], point[0]
        )
        if (distance <= 500) {
            triggerAlarm()
            break
        } else {
            Log.e( tag: "MainActivity",  msg: "ALARM TRIGGERED")
            location++
        }
    }
}
```

04

Software

Software

Overview

- **User Experience**

  Highlight the user-friendly aspects of the map interface.

- **Data Privacy**

  Explain how the app protects user data.

- **Permission Management**

  Demonstrate how users can manage location permissions within the app.

- **Updates**

  Explain the process for updating the mapping data and software(Realtime).

- **Device Compatibility**

  Show the app running on different devices.

  Explain how the app integrates with other location-based services or APIs.

Technology

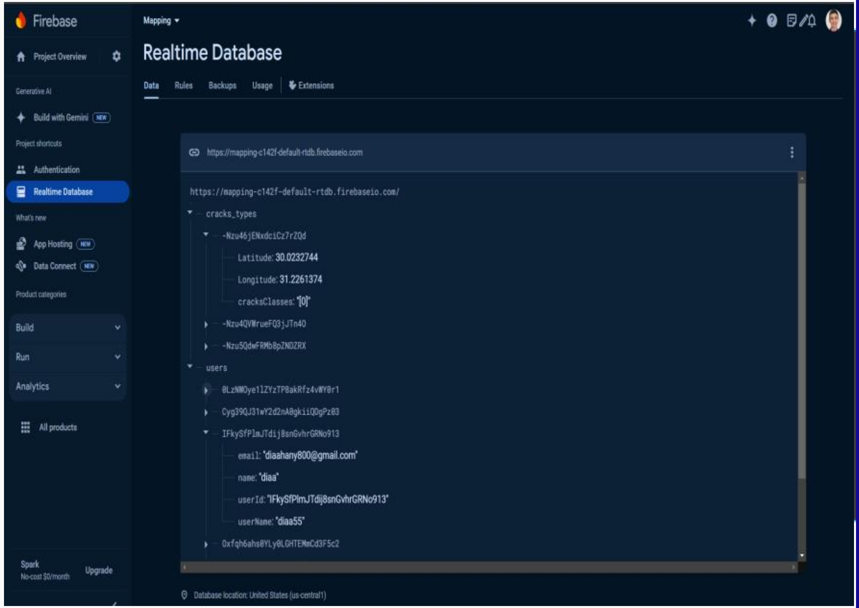
Android

Kotlin

Firebase

Firebase Intgration

- **Authentication**
  - Simplifies the process of user authentication. Supports authentication method(email/password).
- **Database**
  - Reed And Write User’s Data Using Firebase (Realtime Database).
  - Storing Cracks Types Using Firebase (Realtime Database).
  - Database Location (US)



05

Conclusion

Summary And Conclusion

- An Android application for crack and road sign detection can significantly enhance road safety by incorporating several advanced features. The app will use cameras to detect cracks on the road . When the vehicle approaches a detected crack, the app will provide an audible alert to the driver, helping them avoid potential hazards. Additionally, users can enter their location into the app, which will be displayed on an interactive map. This map will continuously update to show the user’s current position and save locations where cracks have been detected, allowing for easy navigation and reference.

Summary And Conclusion

- The app will also include a survey feature where users can report new cracks by capturing images. These images will be analyzed by a back end system to verify whether they indeed show a road crack, leveraging machine learning techniques for accurate detection. The back-end system will store data on detected cracks and user-reported issues, helping to build a comprehensive database that can be used for further analysis and improvement of road conditions. By combining real-time detection, user input, and data analysis, this app aims to enhance road safety and contribute to long-term road maintenance and planning.

06

Results

Achieved results

- This project achieved the core resulted that was our focus. Helping drivers to avoid road cracks as much as can to protect vehicles from damages and decrease traffic congestion as much as we can , and also gather more data about roads cracks.

Limitations

- This project is only for visualizing the cracks on map it does not navigate or route users to destinations.

Results reflection

- Our map does not contain any road sign location and the notification about them is not implemented here , as scanning a region was really hard because signs unlike cracks located far from each other and at least you have to scan an entire government to locate this points on Map.

07

Future Works

Future works

- Our future work plan is to add navigation and routing for user to be able to select a destination . In addition to adding a feature for the user to be able to show route statistics meaning that user could see how many cracks on this road and road speed limit. If the tools will be available we will definitely add road signs locations and notify user while approaching them .

Thank You

Graduation project 2024