

Software Design Document (SDD)

for

Sidewalk Slope Monitoring System

Prepared by: Aquil Alam, Alejandro Chanocua, Omar Eclicerio, Ernesto Garcia,
Francisco Gastelum, Henry Gonzales, Gui He, Perla Ramirez, Rishi Shah, Daniel
Zeng

Department of Public Works and Bureau of Engineering, LA City

CSULA Senior Design 2021-2022 / LA City Bureau of Engineering

Table of Contents	
1.0 Introduction	5
1.1 Purpose	5
1.2 Document Conventions	5
1.3 Intended Audience and Reading Suggestions	5
1.4 System Overview	5
2.0 Design Considerations	7
2.1 Assumptions and Dependencies	7
2.2 General Constraints	7
2.3 Goals and Guidelines	8
2.4 Development Methods	9
3.0 Architectural Strategies	11
3.1 Task 1	11
4.0 System Architecture	13
4.1 Task 1	13
4.2 Task 2	14
4.3 Task 3	16
4.3.1 Rover UI	16
4.4 Task 4	16
4.4.1 Database Data Diagram	16
4.4.2 Sidewalk Table	17
4.4.3 Rover data table	20
4.4.4 Rover sidewalk	22
4.4.5 Sidewalk GoPro Metadata	23
5.0 Policies and Tactics	25
5.1 Choice of which specific products used	25
5.2 Plans for ensuring requirements traceability	26
5.3 Plans for testing the software	26
6.0 Detailed System Design	27
6.1 Raspberry Pi 3	27
6.1.1 Responsibilities	27
6.1.2 Constraints	27
6.1.3 Composition	27
6.1.4 Resources	27
6.1.5 Interface/Exports	27
6.2 Rover User Interface	27
6.2.1 Responsibilities	27
6.2.2 Constraints	27
6.2.3 Composition	27
6.2.4 Uses/Interactions	27
6.2.5 Resources	27
6.2.6 Interface/Exports	28
6.3 Image Processing	28
6.3.1 Responsibilities	28
6.3.2 Constraints	28

6.3.3 Composition	28
6.3.4 Uses/Interactions	28
6.3.5 Resources	28
6.3.6 Interface/Exports	28
6.4 Web application and mapping files	28
6.4.1 Responsibilities	28
6.4.2 Constraints	28
6.4.3 Composition	29
6.4.4 Uses/Interactions	29
6.4.5 Resources	29
6.4.6 Interface/Exports	29
7.0 Rover UI Design	30
7.1 Rover UI Input Field and Buttons	30
7.2 Rover UI JoyStick	30
8.0 Database Design	31
8.1 Relational Schema	31
8.2 Diagram Description	31
8.3 Azure blob Storage implementation	31
8.4 Data Collection	32
8.5 Future Implementation	32
9.0 User Interface	33
9.1 Overview of User Interface	33
9.1.1 Task 1, Web Application	33
9.2 Screen Frameworks or Images	33
9.2.1 Task 1, Web Application	33
9.2.2 Task 3, Rover UI	35
9.3 User Interface Flow Model	36
9.3.1 Task 1, Web Application	36
10.0 Requirements Validation and Verification	37
11.0 Glossary	38
12.0 References	38

Revision History

Name	Date	Reason For Changes	Version
Aquil Alam	10/4/2021	Fall 2021 Draft Prepare.	1
Perla Ramirez	12/5/2021	Added basic descriptions to sections 1 and 2	1
Perla Ramirez, Ernesto Garcia Alejandro Chanocua, Aquil Alam	12/5/2021	Fixed formatting for headers, changed table of contents for automatic updates. Updated descriptions, contents/explanations for Sections contributed by each person, details on project elaborated.	1
Daniel Zeng, Omar Eclicerio	12/5/2021	Modified section 2.3,2.4, 4, 5.1, 6.3, and updated the table of contents. Content updated.	2
Francisco Gastelum, Gui He, Rishi Shah	12/9/2021	Modified sections 2, 3, 4, 5, 6 from a task 1 perspective.	2
Henry Gonzales	12/9/2021	Updated task 1 for each section	2

1.0 Introduction

1.1 Purpose

The Sidewalk Slope Monitoring System project is an effort to develop the necessary databases, user interfaces, and automation scripts to aid the City of Los Angeles, Bureau of Engineering (BOE) in maintaining over 11,000 miles of sidewalk. Based on their prioritization and scoring system, BOE can assign a numerical score to each sidewalk segment to determine which segments require immediate attention or repair for their Sidewalk Repair Program.

The system developed by our team is designed to help BOE in their data collection by building a robot user interface to control the robot BOE uses during their field analysis, image processing scripts to extract image information, a web application to display and map image data, and a database to hold the data collected and extracted by the system.

This document will outline the design the system explained above. See [System Overview](#) for an exact breakdown of the system into tasks.

1.2 Document Conventions

This Software Design Document (SDD) uses a standard documentation template provided by the California State University, Los Angeles Computer Science department.

1.3 Intended Audience and Reading Suggestions

This document is intended to aid the team members listed in the first page in scoping and developing their initial software design requirements that will later be refined in the Software Requirements Specification (SRS) document. This document is intended to breakdown BOE's initial project requirements into tasks to aid the team in planning work. The entire document should be read by the team. Although team members are ultimately held responsible for their specific tasks, team members should be aware of the entire team's efforts.

This document is intended to provide BOE and the project advisor an overview of the expected work to be completed by the team. Sections 2, 6, and 10 should be read by BOE and the project advisor.

This document is intended to provide a high-level overview of design planning and concepts for Computer Science students involved in future years' efforts. Sections 2, 3, 4, 6, 9, and 10 should be read by Computer Science students who will continue this project beyond Spring 2022.

1.4 System Overview

BOE requires sidewalk data to evaluate sidewalk segments for their Sidewalk Repair Program. Because there are over 11,000 miles of sidewalk, it is valuable to BOE to leverage a process that minimizes overhead by automating image and data processing.

To collect the necessary data to assign scores, BOE plans to utilize a Leo Rover robot, with a GoPro camera mounted on it, to collect images of sidewalk segments in the city of Los Angeles. The

system as a whole is broken down into four major components or tasks. The process for BOE to use our system is as follows:

1. The robot will be accompanied by a field worker that will control it using a user interface developed by the team. The user interface is hereby referenced to as Task 3.
2. Scripts will be produced in efforts to parse Exchangeable Image File (EXIF) data from the GoPro images collected. Image processing scripts will be applied to the images to extract additional data necessary for BOE's prioritization and scoring system. The image processing scripts will hereby be referenced to as Task 2 and begin in the Spring 2022 semester.
3. A web application will display the collected images, one at a time, with its related EXIF and image processing data. In addition, Task 1 will develop scripts to create mapping files that can be hosted on BOE's mapping application, NavigateLA. The web application and the mapping files will hereby be referenced to Task 1.
4. A database developed by the team will contain tables for the processed GoPro EXIF data, rover data, and GeoJSON ready shapes based on collected sidewalk data. The database is hereby referenced to as Task 4.

2.0 Design Considerations

2.1 Assumptions and Dependencies

- Rover expectations
 - Cracks and holes on sidewalks shall be minimal.
 - Battery 4 hrs of nominal driving or 8 hrs of video streaming.
 - Hardware is reliable.
 - User will protect hardware from damage.
 - System is waterproof.
 - Operator will clear sidewalk before measurement.
- Task 1
 - Web application
 - Web application will be used to view specific or individual sidewalk segment images and slope data.
 - Web application will be used to view damage and assign numerical score to sidewalk segments.
 - Mapping files
 - Mapping files will be viewed by the user by importing them to NavigateLA, if and only if, mapping files are not hosted on NavigateLA.
 - Mapping files will contain data for many sidewalk segments.
 - Mapping files will be viewed by the user by selecting a layer loaded on NavigateLA, if and only if, mapping files are kept on a database managed by BOE.
- Task 2
 - System will be used during the day to take optimal photos.
 - Sidewalk will always be captured as the center of the image
 - Rendered images requires user-input before processing measurements
- Task 4
 - Rover
 - User will use GoPro provided app to render 360 Degree Images
 - User will keep an eye on the rover at all times.
 - User will manually extract data from rover's memory cards
 - User will manually control the Rover using the Rover UI.
 - Database
 - User will manually input data into Azure DB
 - User will manually input sidewalk asset ID after every rover scan.
 - User will manually input location data fields after every use.
 - User will denoise the data to get more accurate readings

2.2 General Constraints

- Task 1
 - Web application
 - Web application can use a local instance for development and testing purposes.

- Web application must have access to the database containing sidewalk data.
- Web application must use database when displaying sidewalk data and cannot store or use data locally.
- Web application must use Azure Storage Blob to access GoPro images.
- Web application must use a BOE web server for production.
- Web application must be accessible to BOE, either using Internet or Intranet access.
- Web application must be accessible to BOE regardless of their preferred browser.
- BOE must provision a web server for web application production use.
- BOE must maintain the web server hosting the web application.
- Web application backend will be done using Python/Django.
- Web application frontend will be done using HTML/CSS and Bootstrap.
- Mapping files
 - 3rd party application must use Python language to develop automation scripts.
 - Mapping files must be generated by scripts using the 3rd party application..
 - Mapping files must be importable in NavigateLA, if and only if, mapping files will not be hosted on a database managed by BOE.
 - Mapping files must maintain all design features, such as color schemes, annotations, polygons, lines, shapes, when hosted in NavigateLA.
 - Mapping files must be hosted on NavigateLA when a substantial amount of mapping files is available for viewing.
 - A request for a database to host the mapping files can only be placed after discussing and receiving approval from the advisor and BOE.

2.3 Goals and Guidelines

- Task 1
 - Web application
 - Web application will be hosted on a web server when the frontend and major components of the backend are complete.
 - Web application's web server will be discussed with BOE to determine what languages and web server applications are available to us.
 - Mapping files
 - Mapping files will be hosted on NavigateLA after receiving approval from advisor and BOE to request the database. Database will host files that NavigateLA will reference. Users can select the files as a layer on NavigateLA.
 - Mapping files will be available to users to import to NavigateLA if a database is not available or if only a limited section of the city is mapped by the mapping files.
 - Meeting with the advisor and BOE will be required to determine how users can best view mapping files.
- Task 2
 - Image Processing

- Implement image segmentation and texture processing methods on images to easily help find the x and y displacement of a sidewalk.
 - Count pixels between two points.
 - Convert pixel distance to centimeters.
- Task 3
 - Image reading structure for the image processing.
 - Utilize sidewalk measurements
 - Conjoin the four tasks
 - Continue updating Rover UI to provide a more intuitive experience for the user
 - Fix bugs in logic of Rover UI
 - Test Ui functionality on Rover
- Task 4
 - Rover
 - Build Leo Rover
 - Implement last year's project onto the Leo Rover.
 - Implement this year's tasks on the Leo Rover
 - Database
 - Design a database schema which will include all aspects of the project. Including but not limited it too, rover, UI's, and NavLA.
 - Create a way to bridge rover data with NavLa data. This might require the team to map sidewalk coordinates into their own table and use that to correlate with the existing tables. An algorithm will then be needed to loop through all the coordinates and make that correlation.
 - Create the necessary SQL statements to create said database in an Azure environment provided by the BOE to begin storing important data.
 - Gain access to Azures Blob storage within the BOE's environment to be able to store JPG's.
 - Implement Azure tools to help with data manipulation.
 - Denoise acquired data to have more accurate estimations

2.4 Development Methods

Components to build a slope monitoring system have been divided into four applicable tasks for early stages of the system. Development methods are yet to be discussed.

- Task 1
 - Web application
 - Use prior knowledge from web development courses and projects to build frontend and backend of the site.
 - Refer to Mozilla and Django's documentation for best practices and setting up environment.
 - Refer to Google's Maps Javascript API documentation for setting up Map API on web application.
 - Mapping files

- Reference ArcGIS forums to explore other user's methods and processes when developing scripts using ArcGIS
 - Understanding ArcGIS and developing methods to map rover data and GoPro metadata
- Task 2
 - Use the Grabcut Algorithm from the OpenCV library to easily remove background noise and segment sidewalks from captured GoPro images.
 - Use Euclidean distance method when calculating the distance between two points to find x and y displacement in pixels.
 - Cleaning up the data that is being collected by the Rover using histograms and boxplots to visualize outliers that should be removed.
 - Currently using Z-Test to identify outliers and removing them from categories that are important.
- Task 3
 - Rover User Interface allows field users to control the rover and collect sidewalk data.
- Task 4
 - Database allows for access to storage from all aspects of the project.
 - Sources of data are the rover with numerical data as well as the GoPro with image and EXIF data collected upon testing; incorporated with the corresponding sidewalk data provided by BOE.
 - Organized in Azure Data Studio to efficiently manipulate and connect data from all sources including the rover as well as other image data.
 - With the large number of images involved, Azure Blob storage is implemented for easy access from other tasks.
 - Denoise collected data to remove outliers by looking through the data and finding duplicates or data with extreme differences compared to the other data.

3.0 Architectural Strategies

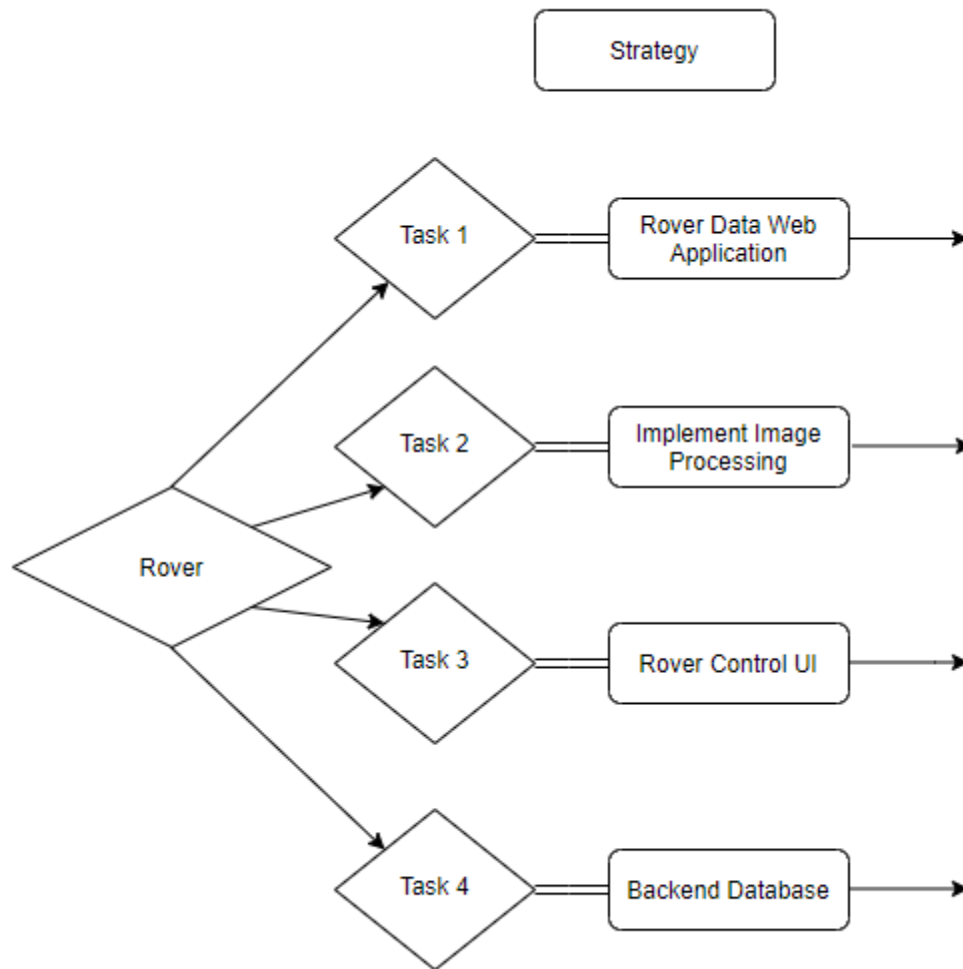
3.1 Task 1

- Web application
 - Used Django to combine HTML, CSS, Bootstrap for frontend and Python for backend.
 - Python libraries are heavily used in parsing and processing image data both in task1 and in other tasks.
 - Enables team to utilize each other's scripts and combine efforts without being concerned with language compatibility.
 - Django uses templates that make it easier in developing the backend.
 - Django testing environment can easily be created. Existing environment can easily be replicated following Django and Mozilla's documentation.
 - Django, Mozilla, and Python have an extensive set of documentation that we have heavily depended on to ensure our product is created with best practices.
 - Python used to create scripts to parse EXIF data from GoPro images
 - Python used to create scripts that extract database data and to extract images from Azure Storage Blob
 - Depending on web server availability from liaison, the Django environment will be recreated for a production environment. Dependencies on liaison's web server application may cause changes to our current use of Django to another widely supported web framework.
- Mapping files
 - Used ArcGIS to automate the creation of shapefiles that will be mapped on NavigateLA.
 - ArcGIS videos and explanations, pages regarding app use and how to use the ArcGIS program.
 - ArcGIS supports location and GPS coordinates, which allows us to use EXIF GPS data and map it easily on NavigateLA.
 - Other mapping applications such as ArcGIS are available and their file types are supported by NavigateLA.
 - ArcGIS is also available to students for free from the university, but is only accessible via Parallels Client, a remote desktop application that allows users to access campus servers containing the application. Because we are remotely accessing the server from our personal machines, performance is limited and requires an internet connection.
 - ArcGIS is also used by BOE.
 - Once many mapping files are created to cover a large portion of the city of Los Angeles, the mapping files can be hosted on NavigateLA.
 - Users would be able to select our layers on NavigateLA.

- Otherwise, users would need to import the mapping files and apply color schemes. When imported to NavigateLA, color schemes are not maintained. Any colors or annotations applied to the mapping file is forgotten.
- However, in order for the layer to be hosted on NavigateLA, large amounts of sidewalk data and fully functioning automation scripts are required. This would develop many, or perhaps a large shapefile, that covers a large portion of the city of Los Angeles. We can then request a database from the liaison to host our data so that NavigateLA users can select the layer.

4.0 System Architecture

The main functionalities of the system are divided into four tasks, then brought together at the end. The four tasks; create a web application for rover data, implement image processing, create a user interface for rover controls, and create a backend database.



4.1 Task 1

- Web Application
 - The web application will display a GoPro image and all of its related data on the console. This includes GPS data, slope data, and other data. Users can switch back and forth between images, use the auto feature so that the position in the sidewalk moves forward showing many images, or search for specific images based on GPS coordinates and image name.
 - The web application is intended to show one instance of a sidewalk segment at a time. It will not provide an overview of the entire city of Los Angeles.
 - Data shown on the web console is obtained from the Azure storage and SQL Server database. The console contains data from the image processing and database teams.

- Mapping files
 - Mapping files will pull data from the Azure database. It will map the GPS coordinates and the sidewalk to a shapefile in ArcGIS app.
 - The shapefile can then either be imported or hosted on NavigateLA.
 - The shapefile can provide an overview of the entire city of Los Angeles. Users can also zoom in to a section of the map and view annotations or additional notes creating during the automation scripts.

4.2 Task 2

Implements the functionalities to process the images rendered and stored by the rover camera. Currently, the program reads an image and applies image processing algorithms using the OpenCV library.

Algorithms in use:

Image Thresholding and Canny Edge Detection

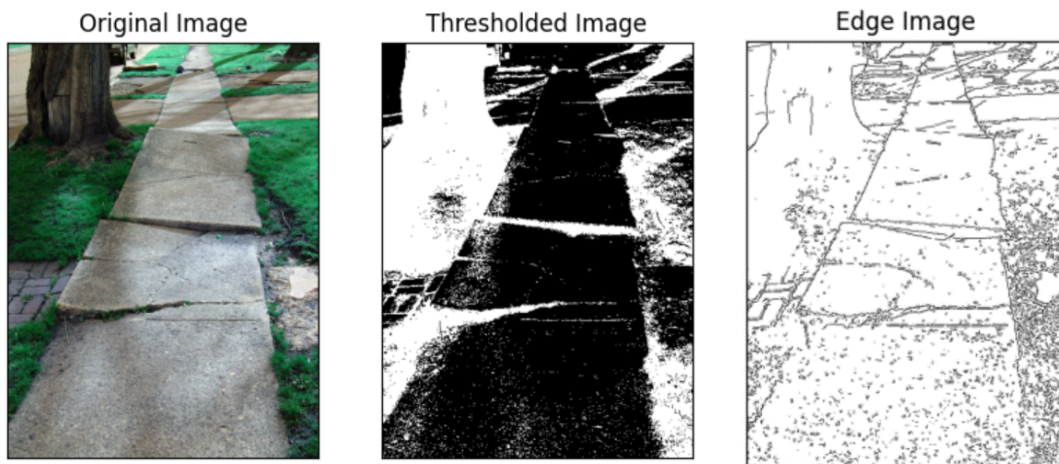
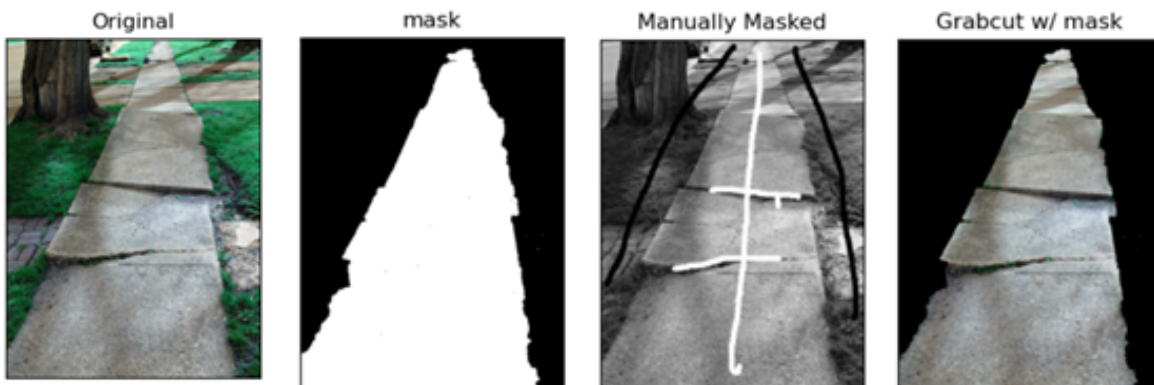


Image Segmentation and Grabcut Algorithm



Current goal of this task is to count the pixels so it can be scaled onto real unit measurement, specifically in centimeters. The above images can easily help find the edges of the sidewalk to

precisely locate the starting and ending points for each of the x and y displacements. Then we can calculate the distance between those two points to find the pixel distance.

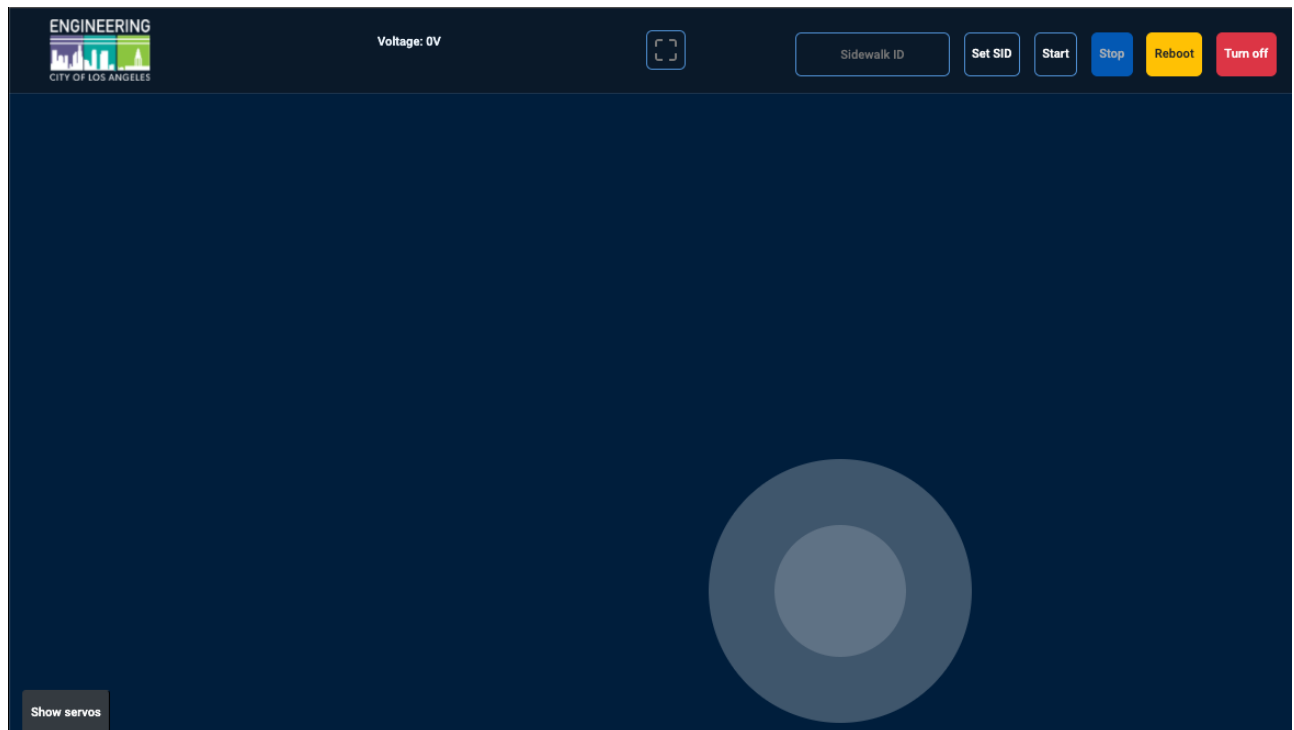


The ultimate goal is to get the measurements of the horizontal and vertical displacements on the sidewalk. Once we find the pixel distance for the x and y displacements, we convert it to centimeters. Those measurements will be used as part of the parameters in labeling the priority of the sidewalk images.



4.3 Task 3

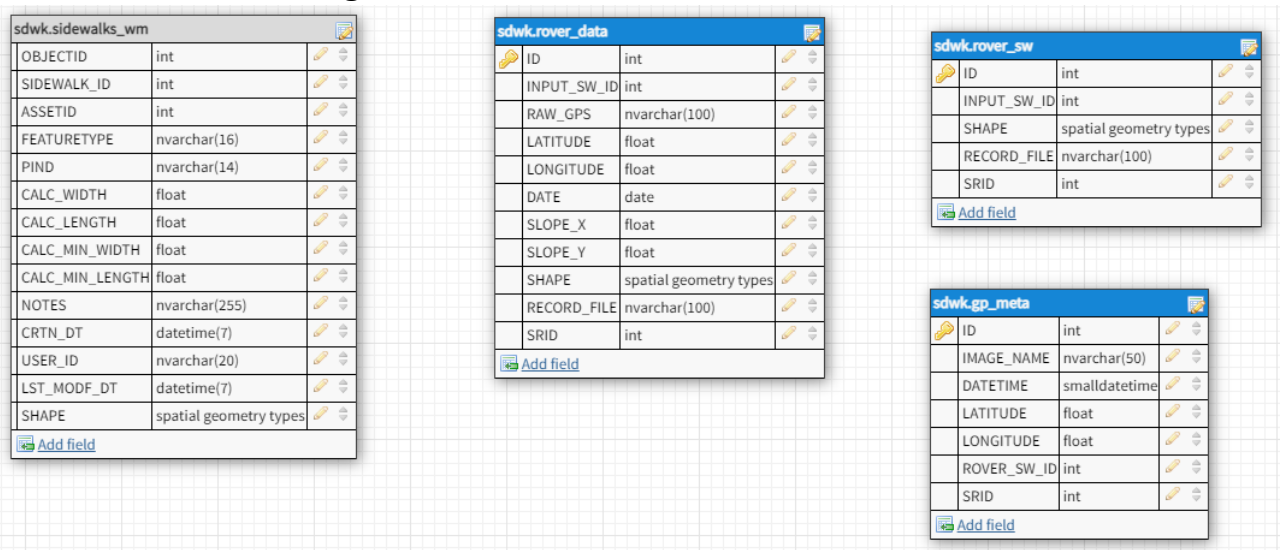
4.3.1 Rover UI



This Rover UI allows manual control of the Leo Rover. Input field for the Sidewalk ID corresponds to the sidewalk segment to start collecting data. Data is stored on the SD card on the rover to be post processed.

4.4 Task 4

4.4.1 Database Data Diagram



4.4.2 Sidewalk Table

This dataset contains an inventory of City of Los Angeles Sidewalks and was digitized using a combination of G.I.S software, aerial imagery (2014 LARIAC), and geographic dataset of property/right-of way lines. Dataset is not actively being maintained. Last updated: 2/23/2021

Table Name - sdwk.sidewalk wm

Column name	Type	Descriptive name	Allow Nulls	Description
OBJECTID	INT	Object Field Asset Identifier	NO	A unique feature identifier populated by Los Angeles City Staff for internal use.
SIDEWALK_ID	INT	Sidewalk Identifier	NO	A unique feature identifier populated by Los Angeles City staff for internal use.
ASSETID	INT	Asset Identifier	NO	A unique feature identifier populated by Los Angeles City staff for internal use.
FEATURETYPE	NVARCHAR (16)	Feature Type	NO	Describes the type of feature class(sidewalk/ramp/driveway/etc) the data belongs to.
PIND	NVARCHAR (14)	Parcel Identification Number	NO	A unique Parcel Identification Number (PIN) for all parcels

				within the City of L.A. All Sidewalk related features will be split, non-overlapping, and have one associated PIN.
CALC_WIDTH	FLOAT	Calculated Width	NO	A generalized width of the feature calculated using spatial and mathematical algorithms on the feature. Widths are rounded to the nearest whole number. In cases where there is no value for the width, the applied algorithms were unable to calculate a reliable value.
CALC_LENGTH	FLOAT	Calculated Length	NO	A generalized length of the feature calculated using spatial and mathematical algorithms on the feature. Lengths are rounded to the nearest whole number. In cases where there is no

				value for the length, the applied algorithms were unable to calculate a reliable value.
CALC_MIN_WIDTH	FLOAT	Calculated Minimum Width	NO	In almost all cases where features have variable widths, the minimum width is used.
CALC_MIN_LENGTH	FLOAT	Calculated Minimum Length	NO	In almost all cases where features have variable widths, the minimum length is used.
NOTES	NVARCHAR (255)	Notes	YES	Short notes created by
CRTN_DT	DATETIME(7)	Creation Date	NO	Indicates date feature was created
USER_ID	NVARCHAR (20)	User Identifier	NO	Name of field agent responsible for feature input
LST_MODF_DT	DATETIME(7)	Last Modified Date	NO	Indicates date feature was revised/edited
SHAPE	GEOMETRY	Shape	NO	A polygon that represents paved pedestrian walkways. The polygon is constructed by

				several GPS (Latitude, Longitude) coordinates. The first coordinate of the polygon will also always be the last coordinate.
--	--	--	--	---

4.4.3 Rover data table

The rover data table was created to store data extracted from rover output files. The table contains GPS coordinates, slope measurements, and information of the file the data originated from. All coordinates and geometry data are in Web Mercator(SR3857) format and can be used to locate a position on any map system using SR3857.

Table Name - sdwk.rover_data

Column name	Type	Descriptive name	Allow Nulls	Description
ID	INT	Identifier	NO	Incremental number used for indexing.
INPUT_SW_ID	INT	Input Sidewalk Identifier	NO	Numeric identifier assigned by field agent. Identifiers should be referenced from the sidewalk table.
RAW_GPS	NVARCHAR(100)	Raw GPS Text	NO	Text string extracted directly from rover output csv file.
LATITUDE	FLOAT	Latitude	NO	Latitude coordinates extracted from

				GPS values in SR:3857 format.
LONGITUDE	FLOAT	Longitude	NO	Longitude coordinates extracted from GPS values in SR:3857 format.
DATE	DATE	Date	NO	Date indicating the recording of GPS data
SLOPE_X	FLOAT	Slope X	NO	X slope value extracted from rover data recording.
SLOPE_Y	FLOAT	Slope Y	NO	Y slope value extracted from rover data recording.
SHAPE	GEOMETRY	Shape	NO	Spatial point geometry created from latitude and longitude referencing GPS position rover data was recorded.
RECORD_FILE	NVARCHAR(100)	Recorded File Name	NO	File name of .csv file recorded data was extracted from.
SRID	INT	Spatial Reference Identifier	NO	Spatial reference identifier denoting the format GPS

				coordinates are in.
--	--	--	--	---------------------

4.4.4 Rover sidewalk

The rover sidewalk was created to store spatial geometry generated per output file from the rover. Every GPS coordinate collected from the output file is used to create a polygon (SHAPE) that can be used to generate an outline of the portion of sidewalk traversed by the rover. The default spatial reference id is 3857 and all relative coordinate data is formatted in that manner.

Table Name - sdwk.rover sw

Column name	Type	Descriptive name	Allow Nulls	Description
ID	INT	Identifier	NO	Auto incrementing number used for indexing.
INPUT_SW_ID	INT	Input Sidewalk Identifier	NO	Numeric identifier assigned by field agent. Identifiers should be referenced from the sidewalk table.
SHAPE	GEOMETRY	Shape	NO	A polygon that represents the sidewalk positions the rover traversed. The polygon is constructed by several GPS (Latitude, Longitude) coordinates. The first coordinate of the polygon will also always be

				the last coordinate.
RECORD_FILE	NVARCHAR(100)	Recorded File Name	NO	File name of .csv file recorded data was extracted from.
SRID	INT	Spatial Reference Identifier	NO	Spatial reference identifier denoting the format GPS coordinates are in.

4.4.5 Sidewalk GoPro Metadata

The Sidewalk GoPro Metadata table was created to store metadata extracted from GoPro images. The GPS coordinates from the metadata is useful to associate the image taken to a position on the map. The IMAGE_NAME can also be used to locate the image file from the Azure Blob store. The ROVER_SW_ID references the INPUT_SW_ID from the sdwk.rover_sw table. To find the sidewalk id associated with the image, a geometric point is created from the latitude and longitude of the metadata to find the nearest geometry from the sdwk.rover_sw table to that point and assign the associating INPUT_SW_ID.

Table Name - sdwk.gp meta

Column name	Type	Descriptive name	Allow Nulls	Description
ID	INT	Identifier	NO	Auto incrementing number used for indexing.
IMAGE_NAME	NVARCHAR(50)	Image File Name	NO	Image file name where metadata was extracted from. This is used to locate the image from the Azure blob store.
DATETIME	SMALLDATET	Date time	NO	Date and time

	IME			of image recording.
LATITUDE	FLOAT	Latitude	NO	Latitude position for where the image was taken. Coordinates are in SR:3857 format.
LONGITUDE	FLOAT	Longitude	NO	Longitude position for where the image was taken. Coordinates are in SR:3857 format.
ROVER_SW_ID	INT	Rover Sidewalk Identifier	YES	The sidewalk id the image was estimated to be recorded at. A geometric point is created from the longitude and latitude values and used to find the closest rings to that point and the sidewalk id is taken from the row containing the rings.
SRID	INT	Spatial Reference Identifier	NO	Spatial reference identifier denoting the format GPS coordinates are

				in.
--	--	--	--	-----

5.0 Policies and Tactics

5.1 Choice of which specific products used

- Task 1
 - Web application
 - Django
 - Django uses Python for its backend. See Python under the mapping files bullet point to view why we select Python.
 - HTML, CSS, and Bootstrap
 - Google Map Javascript API
 - Mapping files
 - ArcGIS
 - Program available to automate mapping file creation. Forums and documentation thoroughly explains how app may be used and provides sample use cases.
 - Python
 - Existing scripts from previous years efforts use this language. Python has an extensive library that can provide parsing and data processing abilities that is not readily available in other languages.
- Task 2
 - OpenCV Library
 - OpenCV library was chosen for image processing because it provides many functions that helps with image segmentation.
 - Our advisor recommended using OpenCV.
- Task 3
 - Rover UI was implemented using HTML and Bootstrap library.
 - We choose to use a Python IDE due to the Rover we acquired, Leo Rover, uses a Raspberry Pi microprocessor. The database we choose is the Azure database based on our liaison, the city of los Angeles, using the same database. This is so we can transfer our data freely between the liaison and our project.
- Task 4
 - Rover
 - Leo Rover was chosen as a replacement for our previous rover “Robecca ” because it's more versatile, price friendly, and is a good platform to base a fleet of rovers on.
 - GoPro Fusion was chosen as our camera because it offers a 360-degree view of the rovers surrounding with only one product.

- Database
 - Azure SQL database was chosen by our corresponding liaisons because it's what they are currently using themselves.
 - Azure Blob storage was chosen as a solution for image storage because it integrates to Azure DB nicely and it's the most cost-effective method available to the BOE.

5.2 Plans for ensuring requirements traceability

- Task 1
 - Web application
 - Backend is actively extracting data from the Azure storage and SQL Server database. It is not storing or using local data.
 - Mapping files
 - Automation scripts create shapefiles that meet the design requirements. Design requirements include color schemes based on BOE's prioritization and scoring system, annotations for slope, etc.

5.3 Plans for testing the software

- Task 1
 - Web application
 - Users can view an image. The administrators can confirm that the image shown, and its pulled data are related.
 - Users can flip through the images and its related data is shown. See the first bullet point.
 - Users can auto flip through the images and all related data is shown. See the first bullet point.
 - Django local server will be created for testing purposes
 - Mapping files
 - When running the ArcGIS app, it pulls data from the Azure database and create shapefiles.
 - The shapefiles must display the shape of a sidewalk with annotations and any needed color schemes (applying color schemes based on BOE's prioritization and scoring system).
- Quality Control

6.0 Detailed System Design

6.1 Raspberry Pi 3

6.1.1 Responsibilities

The role of this module will be to compute and store data captured by the rover modules such as Camera and Accelerometer.

6.1.2 Constraints

Possible constraint of this module would be the memory size of the MicroSd Card attached to the rover.

6.1.3 Composition

A description of the use and meaning of the subcomponents that are a part of this component.

Uses/Interactions

User interaction to power the Raspberry Pi 3

6.1.4 Resources

This is the main module that will power the accelerometer and camera.

They are dependent on Raspberry Pi 3

6.1.5 Interface/Exports

CSV file will export data from the MicroSd card into the database.

Interface Linux.

6.2 Rover User Interface

6.2.1 Responsibilities

The primary responsibility of the rover interface is to allow the user to access, move, and to automate data collection using the rover.

6.2.2 Constraints

Constraints of this component would be based on the sidewalk condition, any debris would give inaccurate results.

6.2.3 Composition

Extra camera added on the rover is a GoPro. It is responsible for capturing photos of the sidewalk where data was recorded.

6.2.4 Uses/Interactions

The rover user interface will be used to control the rover movements and ability to collect data. The rover user interface will interact with the Bureau Of Engineering field workers.

6.2.5 Resources

Rover UI

6.2.6 Interface/Exports

The interface has been coded with python tkinter toolkit. The exports available will be a data CSV file.

6.3 Image Processing

6.3.1 Responsibilities

Image processing functionalities will be implemented for the images taken from the GoPro camera attached to the Rover. The functionalities include measuring the XY (vertical and horizontal) displacements on the sidewalk and eventually categorize the images according to their condition.

6.3.2 Constraints

Measuring the vertical displacement is the close to impossible. The angle of the images being take from the camera doesn't show the distance from an overhead view.

6.3.3 Composition

We use OpenCV or Open-Source Computer Vision library that is compatible with Python for image processing functionality. We currently use Pycharm IDE to utilize those components.

6.3.4 Uses/Interactions

If the strokes for Grabcut technique mentioned earlier cannot be implemented automatically, user strokes are a requirement to isolate the sidewalk, when measurements are made.

6.3.5 Resources

Beginning stages don't require any resources other than the images used to learn and test OpenCV algorithms.

6.3.6 Interface/Exports

OpenCV library built in GUI with input and processed image.

6.4 Web application and mapping files

6.4.1 Responsibilities

The web application is a console for users to view a single instance of a GoPro image and its related data. The mapping files script will create DWG files automatically that can either be imported or hosted on NavigateLA. The DWG files will contain data from the GoPro image, too.

6.4.2 Constraints

Web applications will be developed in a test environment locally. If the liaison provides a web server, the web application will be moved to a production environment. The mapping files will be imported to NavigateLA. If many mapping files are creating using sidewalk data, the mapping files can be hosted on NavigateLA. This depends on the amount of sidewalk data collected, GoPro images available, and a database to host the mapping files.

6.4.3 Composition

The web application is the web console that users interact with. The mapping files are automation scripts that create DWG files.

6.4.4 Uses/Interactions

The web application and the mapping files scripts use the image processing scripts and the Azure database.

6.4.5 Resources

Azure database, image processing scripts

6.4.6 Interface/Exports

Web application is the web console and the mapping files scripts create DWG files.

7.0 Rover UI Design

7.1 Rover UI Input Field and Buttons



Located on the nav bar:

- The input field allows the user to input the desired sidewalk ID.
- Set SID sets the imputed sidewalk ID.
- Start begins the data collection process on the Leo Rover.
- Stop ends the data collection process on the Leo Rover.
- Reboot, reboots the Leo Rover.
- Turn Off, shuts down the Leo Rover.

7.2 Rover UI JoyStick



Located on the lower half the Rover UI:

- Transfers directional movement of the joystick to direction the rover will navigate towards.

8.0 Database Design

The database is utilized by all aspects of the project, linking the existing data from NavLA to new data collected by the rover--providing access web server which displays the data in real-time. In addition, it provides access and stores the images from the GoPro which can then be directly implemented with the image processing side of things.

8.1 Relational Schema

sdwk.sidewalks_wm	sdwk.rover_data	sdwk.rover_sw
OBJECTID int	ID int	ID int
SIDEWALK_ID int	INPUT_SW_ID int	INPUT_SW_ID int
ASSETID int	RAW_GPS nvarchar(100)	SHAPE spatial geometry types
FEATUETYPE nvarchar(16)	LATITUDE float	RECORD_FILE nvarchar(100)
PIND nvarchar(14)	LONGITUDE float	SRID int
CALC_WIDTH float	DATE date	Add field
CALC_LENGTH float	SLOPE_X float	
CALC_MIN_WIDTH float	SLOPE_Y float	
CALC_MIN_LENGTH float	SHAPE spatial geometry types	
NOTES nvarchar(255)	RECORD_FILE nvarchar(100)	sdwk.gp_meta
CRTN_DT datetime(7)	SRID int	ID int
USER_ID nvarchar(20)	Add field	IMAGE_NAME nvarchar(50)
LST_MODF_DT datetime(7)		DATETIME smalldatetime
SHAPE spatial geometry types		LATITUDE float
Add field		LONGITUDE float
		ROVER_SW_ID int
		SRID int
		Add field

8.2 Diagram Description

The relational schema as seen in the diagram is organized by the various sources of data in mind.

The leftmost part is the existing data provided by BOE:SWBot, and property tables. It includes PIN IDs and asset sidewalk IDs which are essential in pinpointing specific properties and sidewalk locations. Navigate LA data will serve as geographical reference points for the GoPro data and the rover data listed.

The rightmost data, containing the two Rover Data Tables and GoPro Image Data Table, would be extracted from the rover itself after collection and uploaded in their respective tables. The rover_data table consists of collected GPS and slope data along with other identifiers such as input sidewalk ID and the spatial reference ID. In order to represent the collected data as a shape, the data is processed and stored in the rover_sw table. The GoPro metadata is contained in the gp_meta table, which holds the GPS data extracted from the taken images as well as useful identifiers such as the sidewalk ID.

8.3 Azure blob Storage implementation

Azure Blob Storage is required in order to store a large amount of JPGs--such as the front, rear, and rendered Images. Image IDs from the GoPro picture data will serve as a link for the images stored in the blob storage. This method allows us to create a much more efficient method of storing images and sharing them across different platforms, such as our UI and web applications.

8.4 Data Collection

Data sources include:

1. Rover
 - a. The main focus as it holds key information that will be used by BOE to prioritize the locations in need of repairs.
 - b. GPS Coordinates (Precise Lat and Longitude)
 - c. Timestamps
 - d. Leveler - Vertical and Horizontal displacement of the sidewalk
2. GoPro Fusion
 - a. Forward and Rear facing camera to create a 360 Degree image on the rovers surroundings
 - b. Rendered Images
 - c. Python code to exfiltrate the EXIF/METADATA from each image which provides another DB table to populate
3. Existing data - Mapped by the BOE on NavLA

8.5 Future Implementation

Future implementations consist of things such as

1. Data Manipulation
 - The way data is correlated, either by location or specific timestamps
 - Automation of data uploads
2. Data Visualization
 - Azure offers Tools which help visualize our data, which offers a unique perspective that could be implemented into our applications.
3. Image processing
 - Azure AI will be utilized to introduce severity levels into our dataset and help reduce the manual labor and reach our goal faster and more productively.

9.0 User Interface

The user interface tasks are Task 1 for the web application and Task 3 for the rover UI.

9.1 Overview of User Interface

9.1.1 Task 1, Web Application

The web interface will receive image data from Leo Rover's camera to be transferred to the website. BOE uses an Azure database for their backend, which we will integrate into our web application.

Our web application will include a page to display our data. The data to be displayed includes from the rover, longitude and longitude slope data, Global Positioning System (GPS) data as well as the image name and date when what image was taken. The images that will be displayed come from the GoPro fusion camera on the rover, and we will also be displaying the GoPro metadata which includes, longitude and latitude data. The web application will have the ability to iterate through the image with a previous button, next button, and an auto button. The auto button will iterate through the images automatically, with each image showing for 2 seconds. The user will be able to pan the image and zoom in and out of the image. The web application will have a function to search through the database of pictures by the images name or coordinate which the image was taken from.

The web application will have additional pages which will describe the purpose of our project, the overall description, the environment where our project was worked on, and the algorithms used in this project.

9.2 Screen Frameworks or Images

9.2.1 Task 1, Web Application

The following are wireframes and actual frontend images:

[Sidewalk Project](#)
[Home](#)
[Render](#)
[Database](#)
[NavigateLA](#)
[About](#)



City of LA

Sidewalk Project



The City of Los Angeles, Bureau of Engineering maintains over 11,000 miles of sidewalks. When a segment of sidewalk does not settle evenly or has been raised up by tree-root growth, the sidewalk becomes uneven. This can create pedestrian hazards. In addition, the City is obligated to ensure that its sidewalks conform to Federal ADA standards, which limit the extent to which a sidewalk may slope.

[Learn More](#)

Leo Rover



Leo Rover
 Leo Rover is an outdoor Robotics Development Kit. It is open-source and built on RaspberryPi. With video streaming and driving UI ready out-of-the-box.

[Learn More](#)


 LOGO
 200 N Spring St
 Los Angeles, CA 90012

CONTACT US
 Phone: 1-(800)-466-9999
 Email: random@gmail.com

HELPFUL LINK
[Terms of Use](#)
[Privacy Policy](#)
[FAQ](#)

Copyright © 2017. All Rights Reserved by Random Company

[Sidewalk Project](#)
[Home](#)
[Render](#)
[Database](#)
[NavigateLA](#)
[About](#)

BOE Sidewalk Project

Global Positioning System (GPS)

Latitude: (insert from database) North/South(N/S)

Longitude: (insert from database) East/West (E/W)

Validity

GoPro

Latitude: (insert from database) North/South(N/S)

Longitude: (insert from database) East/West (E/W)

Altitude




Image Name


Date: (insert from database)

SLOPE X ↔

(insert from database)

SLOPE Y ↑↓


(insert from database)



[← Prev](#)

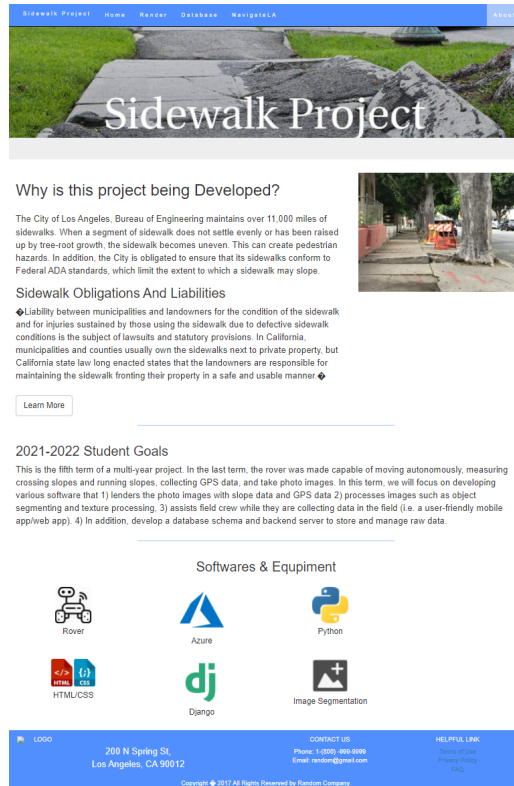
[Auto](#)

[Next →](#)

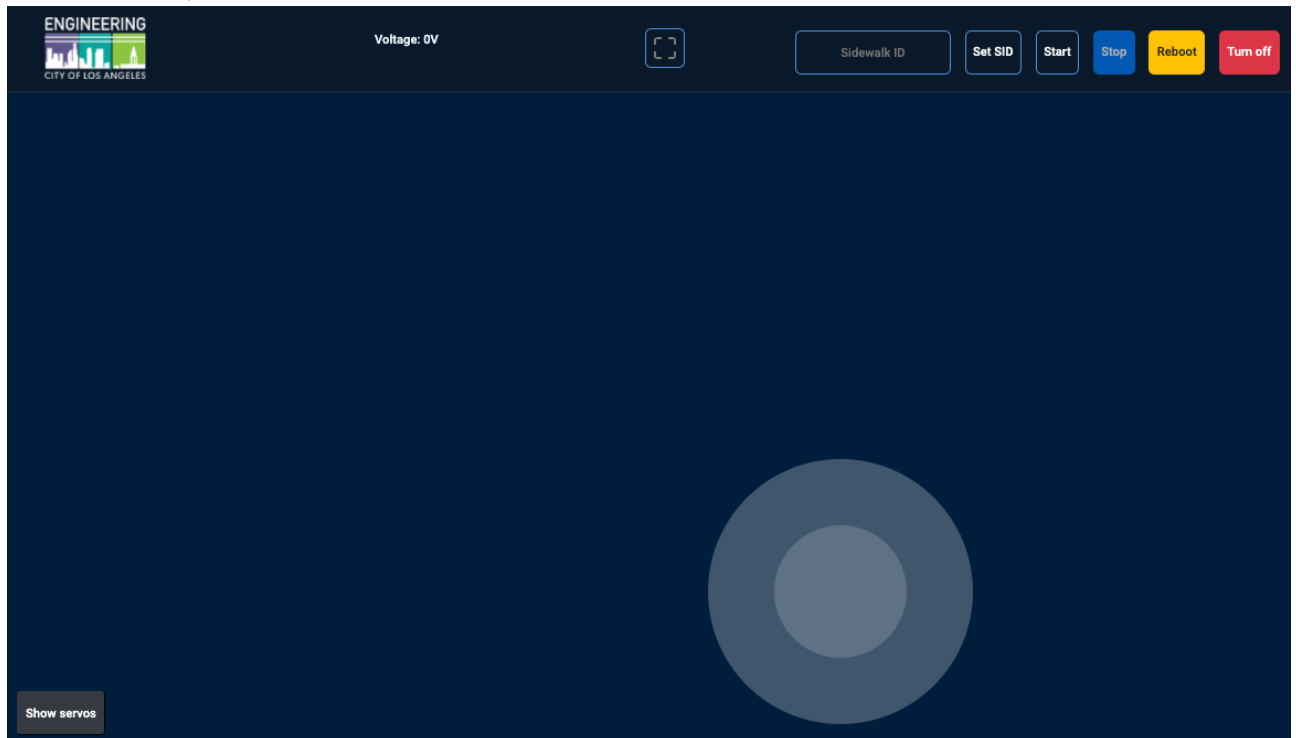

 LOGO
 200 N Spring St
 Los Angeles, CA 90012

CONTACT US
 Phone: 1-(800)-466-9999
 Email: random@gmail.com

HELPFUL LINK
[Terms of Use](#)
[Privacy Policy](#)



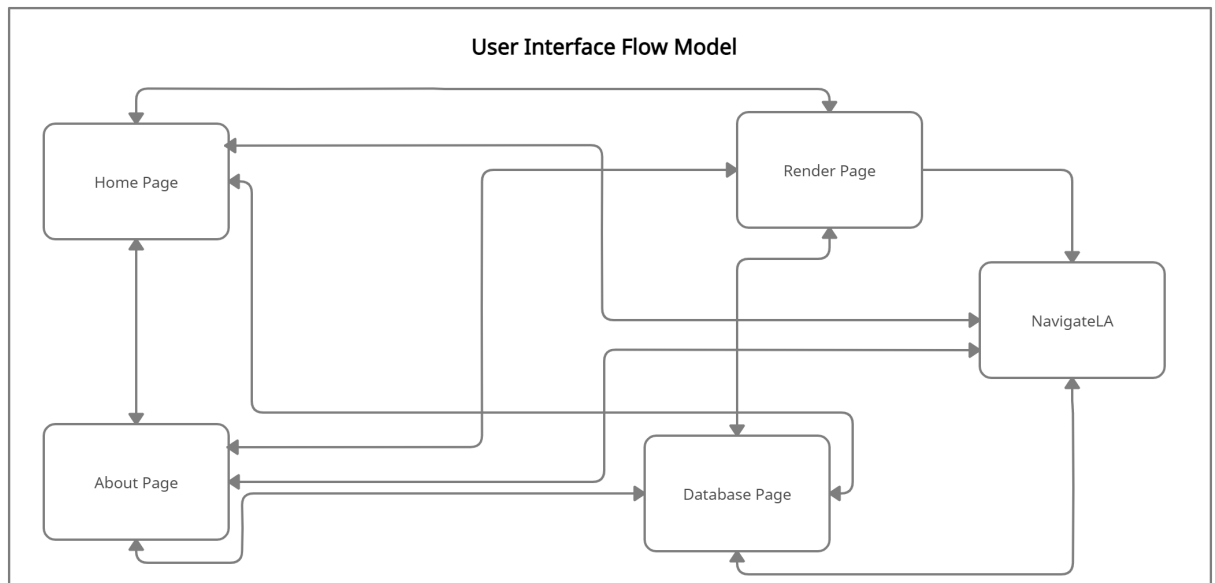
9.2.2 Task 3, Rover UI



9.3 User Interface Flow Model

9.3.1 Task 1, Web Application

The Home, Render, Database and About page can freely navigate between each other. However, the NavigateLA tab is different. It takes the user to a different webpage, <https://navigatela.lacity.org/>. Once at NavigateLA, the user cannot navigate back to the web application.



10.0 Requirements Validation and Verification

Requirement	Component	Test
	Module	
Raspberry Pi 3 shall collect data from modules and store the data.	Raspberry Pi 3	To be determined in Spring
Accelerometer shall measure the grade of sidewalk and send the raw data to Raspberry Pi 3.	Accelerometer	To be determined in Spring
Camera shall be responsible for taking pictures.	Camera	To be determined in Spring
Real-Time Kinematic GPS unit to record centimeter accurate coordinates.	GNSS Device	To be determined in Spring

11.0 Glossary

For the purposes of this project, the following terms are defined as follows:

BOE	City of Los Angeles, Bureau of Engineering
SDD	Software Design Document
SRS	Software Requirements Specification
Rover	Device with wheels that will display the GUI software.
IMU	Inertial Measurement Unit
DB	Database
NavLA	NavigateLA (Site)
EXIF	Exchangeable Image File
IDE	Integrated Development Environment
GUI	Graphical User Interface
Task 1	Web application and mapping files
Task 2	Image processing
Task 3	Rover UI
Task 4	Database

12.0References

- NavigateLA, <https://navigatela.lacity.org/navigatela/>
- Sidewalk Repair Program Prioritization and Scoring System Council File 14-0163-S3, from the City of Los Angeles, Bureau of Engineering
- Azure SQL database - <https://docs.microsoft.com/en-us/azure/azure-sql/>
- Azure Blob - <https://docs.microsoft.com/en-us/azure/storage/blobs/>
- Leo Rover - <https://www.leorover.tech/>
- GoPro Fusion - https://gopro.com/content/dam/help/fusion/manuals/Fusion_UM_ENG_REVC.pdf