Detection and Classification of Cracks on Transportation Infrastructure using UAV Based Aerial Imagery

Design Document

Team: SDMay20-13
Client: Muhammad Ahmad Siddique
Advisor: Dr. Halil Ceylan

Team Members
Lauren Arner - Project Manager
Benjamin Ferreira - Testing lead
Madi Jacobsen - Data Lead
Ian Seal - Reporting Lead
John Schnoebelen - Software Developer
Jack Temple - Lead Software Developer

sdmay20-13@iastate.edu
https://sdmay20-13.sd.ece.iastate.edu

Revised: December 8, 2019 Version 3

Executive Summary

Development Standards & Practices Used

Software

- PyTorch - library for machine learning

Hardware

- UAV - for taking photographs of the pavement
- UAV mounted photography equipment - for taking photographs of pavement

Standards

- IEEE 12207 - Software life-cycle processes
- IEEE 29119-2015 - ISO/IEC/IEEE International Standard - Software and systems engineering--Software testing

Summary of Requirements

- UAV can take photographs of pavement
- Photos will be taken on sunny or overcast days with no precipitation
- Software will be able to detect cracks and joints within concrete
- Software will be able to correctly classify cracks and joints

Applicable Courses from Iowa State University Curriculum

- Com S 309: Software Development Practices
- Com S 319: Construction of User Interfaces
- Com S 311: Introduction to the design and Analysis of Algorithms
- CprE 329: Software Project Management

New Skills/Knowledge acquired that was not taught in courses

- Machine learning using PyTorch library
- Image processing with Python
- Differences between cracks and joints in concrete
- How to utilize Iowa State University's High-Performance Computing (HPC)

Table of Contents

Figures/Tables:

Definitions:

GUI - Graphical User Interface
HPC - ISU's High-Performance Computing system
UAV - Unmanned Aerial Aircraft; a drone that can fly via remote control
UI - User Interface

1 Introduction

1.1 Acknowledgement

Our team would like to acknowledge and thank our client within Iowa State's Civil, Construction and Environmental Engineering department, Muhammad Ahmad Siddique, for the assistance he has provided throughout the project. From the beginning, Ahmad has been proactive in assisting our team with design specifications, equipment, and initial data. Throughout the project, Ahmad was available to answer questions and was willing to help us in whatever manner he could.

1.2 Problem and Project Statement

Problem Statement

Due to the weather fluctuations in Iowa and throughout the Midwest, concrete and other road surfaces are constantly changing causing potholes, cracks, and other problems that create hazardous and at times, undrivable road conditions. Currently the images or videos collected by UAV must be carefully examined by an operator to manually identify the cracks over long pavements. The aim of this project is to develop advanced techniques and algorithms to detect and classify the cracks on the transportation infrastructure using the data provided by the UAV based imagers.

Project Statement

The purpose of our project is to provide a way to identify cracks and their classifications in pavement via photos taken by an Unmanned Aerial Vehicle.

This will be accomplished by using machine learning algorithms and image processing. We will train a machine learning model by creating an artificial neural network that can identify whether a given image has a crack in it or not. The model will be trained using a large dataset of open-source images of cracked/non-cracked pavement. The UAV will take photos of various concrete roads which will be provided to us by the client. Finally, these images will be run through our machine learning model and, after image processing, will be highlighted in areas where cracks are detected by our algorithm. Cracks and joints in the concrete will be classified and highlighted separately to show distinction. In addition, we will supply our client with an intuitive user interface that will help them to quickly detect cracks and joints in a collection of images.

By being able to identify cracks and their classifications, any department charged with fixing and maintaining roads will be able to quickly identify roads in the most critical conditions. As a result, crews will be able to prioritize their work and create safer driving conditions.

1.3 Operational Environment

The main operational environment for this project will consist of days with clear or overcast skies and no precipitation. While the overcast conditions will alter the pictures will still be able to run through image processing whereas precipitation could alter the photos beyond our capabilities. In addition, the UAVs that are used in this project will only fly with no precipitation.

1.4 Requirements

Economic

- Software cost will be kept minimal (use free/open source software)
- The cost to operate the UAV will stay at a minimum

Environmental

- UAV flights will not impact the surrounding environments
- Conditions for flight must be taken into account when flying

Outcome

- Software will be able to detect cracks within concrete and e roads
- Software will be able to detect different surface areas (concrete, asphalt, gravel)
- Software will be able to categorize different cracks (joints vs. cracks)

Functional Requirements

- The software will teach itself what cracks in pavement are and train itself to identify them in new images.

Non-functional Requirements

- The software will be simplified so that a user does not have to have an in depth understanding of command prompt to run the program.
- The software should be portable so that once it is trained a user does not need a powerful computer to identify cracks.
- The software should clearly highlight places of cracks for easy verification for the user.

1.5 Intended Users and Uses

This project is intended to be used by professionals such as construction engineers, researchers, and Department of Transportation employees and officials, as an additional resource to evaluate existing infrastructure including roads and bridges. This will assist in prioritizing repairs and maintenance to roads and bridges in critical conditions in order to ensure they stay safe and drivable.

1.6 Assumptions and Limitations

Assumptions

- Client will provide us with test images to verify the software works correctly.
- Surfaces will have cracks to identify (most surfaces have some form of crack or joints).
- A high-performance computer can be used to train the software.

Limitations

- UAV flying must be planned and will not always be available for use (FAA laws since ISU is within 5 miles of an airport)
- UAV can only fly to a certain height for picture to be useful (photo equipment limitation)
- UAV flight cannot be continuous (UAV hardware limitations)
- There is no budget for the project, thus, we are constrained to using Iowa State University resources.

1.7 Expected End Product and Deliverables

The end product and deliverable to the client will be a software that can intake photographs of roads and identify and classify cracks and joints. This software can be used to help those assessing infrastructure conditions and help prioritize roads for maintenance and repair. The software will be delivered in two phases. The first phase will be delivered by December 13, 2019 and will be a software that identifies any type of cracking in concrete or asphalt. The second phase will be delivered no later than May 1, 2020 and will be software that, in addition to phase 1, will also be able to classify the cracks into different types and include a user interface for the client to use.

2. Specifications and Analysis

2.1 Proposed Design

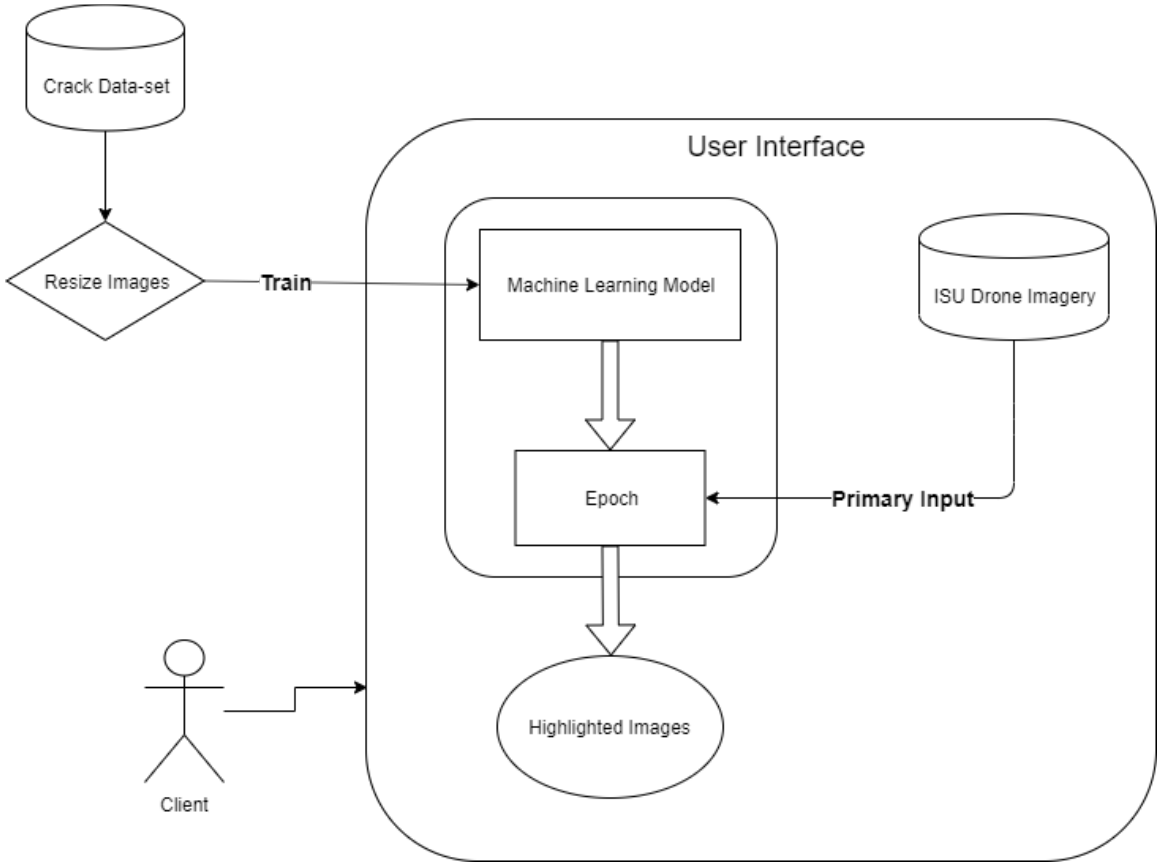

Figure 2.1 - Proposed Conceptual Design

We have decided that our project will use machine learning to identify cracks in the pavement. After looking at a few options, we decided upon using PyTorch due to its ease of use and extensive documentation. Since it is open source, it is also free to use. This meets our economic requirement of our project, as it will dramatically reduce man-hours previously required to examine every picture, and there is no cost associated with using the software. We also looked at using TensorFlow, another commonly used software library for developing machine learning programs. However, after comparing the two libraries, we found that there were more resources to help us along the way for PyTorch in contrast to TensorFlow; such as, more questions answered on discussion boards as well as more YouTube tutorials and open-source projects. Overall, we found the learning curve for PyTorch was better fitting to our level of experience. We will train the model with a dataset of 20,000 uncracked and 20,000 cracked images of concrete.

Using a dataset this large allows the model to have a very high confidence level as opposed to the 300 images we were using initially. We will take the 40,000 images and use 70% of the images for training and 30% to test the accuracy of the model. This creates an epoch with the selected dataset confidence interval. An epoch is when all images in the training set are used to train the model.

2.2 Design Analysis

After looking at different options for machine learning algorithms, we decided that PyTorch would be a better option than its closest competitor TensorFlow. This is because of PyTorch being easier to use and also having more documentation. We have also created a Python script that crops images down so that they can be easily analyzed by the software.

2.3 Development Process

We will follow an Agile-like development process. Since the majority of our project revolves around the machine learning algorithm, most of our tasks will involve removing errors and improving its accuracy. Working in 2-week sprints will also allow us to always have something new to present to our client. The following chart illustrates the project development flow. The timeline of development can be found in both Figure-4.1 and Table-4.1.

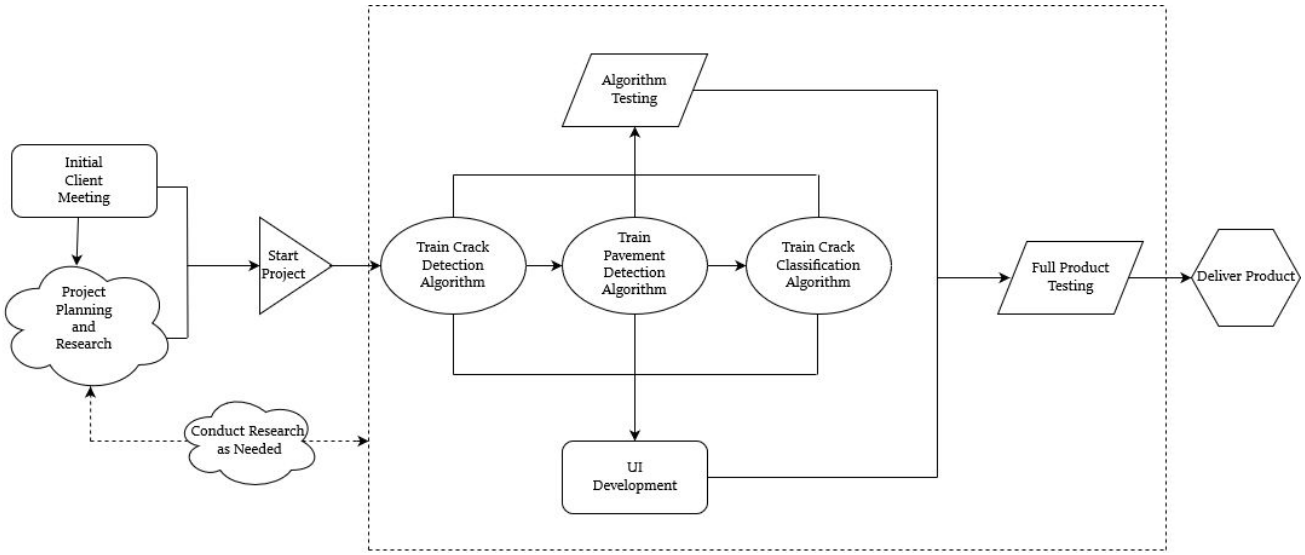

Figure 2.3 - Project Development Flowchart

2.4 Design Plan

The design for our product is relatively straightforward since the project's focus is using artificial intelligence to detect, identify, and classify cracks based on UAV imagery. All the algorithms will be developed using the PyTorch open source library for Python. This algorithm will be able to take a photo, break it down into smaller images, identify and classify cracks in the small images, and then piece the photo back together with the added data.

Our user interface will be a simple interface that will allow a client to use the algorithm without directly interacting with the script. The user should be able to simply upload a photo, wait for the algorithm to process the image, and be able to view the new photo with the added data requested.

The development flow of the work can be found in Figure 2.3 (previous section). This chart denotes the flow of the work shows how some processes can be worked on simultaneously.

3. Statement of Work

3.1 Previous Work And Literature

Our code is based off a research paper titled *Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks* by *Young-Jin Cha[1].* This paper captures, in essence, what needed to be accomplished for this design project. This research paper talks about many aspects in regards to machine learning and neural networks. It described the higher-level architecture which was needed to correctly detect concrete cracks based on imagery. It goes into detail on what each of the layers of the model should entail. There were multiple GitHub repositories implementing similar models that we could reference to create our machine learning model. Our project will use these previous projects as a starting point to machine learning for image detection. A lot of these repositories of crack detection were using TensorFlow and had a lack of documentation which made the understanding of different algorithm examples a lot harder. What expands our project beyond simple crack detection is the requirement to classify cracks and joints as separately recognized entities.

With all the information from these projects, we should be able to focus on improving the detection, as well as adding important features, such as showing the difference between cracks and joints.

3.2 Technology Considerations

The current technology used by researchers in identifying cracks in pavement involves printing off all the images and then circling by hand each crack that exists. This is very time consuming and should be automated. There have been attempts to use machine learning to detect cracks in pavement, however none of them are complete solutions. Most are not able to detect the difference between cracks in concrete and joints between different sections, and grass is frequently identified as cracks. Most examples we found also used datasets of around 4000 images to train the program. In order for a model to be trained to a higher confidence level, there must be large datasets, which is hard to come by without manually creating/gathering the images for the set.

3.3 Task Decomposition

We have divided this project into a few main parts. The first task will be to create an algorithm that is trained to identify cracks in pavement. At this point the program will only be focused on identifying anything that appears to be a crack in the pavement, which means that shadows, joints, and grass will most likely still be detected. The next task will be to have the program no longer identify shadows and grass as cracks and identify joints in a different manner. The final task will be to create a simple user interface for the program so that researchers will not have to run the program through a command line.

The following chart breaks down the milestones and necessary tasks. Some tasks will be worked on at the same time as others (ex. UI development for crack detection and training the pavement classification algorithm)

Milestone/Tasks	*Task Description*
Project Setup and Start	*Meet with client and determine scope, gather initial information for the project, and begin initial research*
Client Meeting	Meet with client to determine scope and product expectations
Initial Project Planning	Determine research that needs to be done and establish timeline with milestones and tasks

Initial Project Research	Research tools to be used, working with algorithm learning, and other research as necessary to fill the technical knowledge gap
Train Crack Detection Algorithm	*Train crack detection algorithm with 80% verified accuracy*
Create Algorithm to Breakdown Images	Develop a script to break down images to size of 128x128 pixels to begin training the algorithm
Train Algorithm	Use data sets to train the algorithm to recognize cracks. Data sets used will be open source.
Test Crack Detection Algorithm	Testing will consist of automated and manual verification of algorithm accuracy. As needed, new photos can be used to ensure the correctness of the algorithm
Train Pavement Classification Algorithm	*Train pavement classification algorithm with 80% verified accuracy*
Train Algorithm	Use datasets to train the algorithm to identify types of pavements. This will be used in the future to classify different types of cracks
Integrate Pavement Classification Algorithm	Algorithm will need to be integrated with crack detection. Both crack and pavement type will be used as factors to classify cracks later in the project
Test Pavement Classification Algorithm	Test and ensure accuracy of pavement types. This will be tested both before and after integration and will be both manual and automated.
Train Crack Classification Algorithm	*Train pavement classification algorithm with 80% verified accuracy*

Train Algorithm	This algorithm will take into account previous crack patterns and pavement to determine the correct classification for cracks.
Integrate Crack Classification Algorithm	This algorithm will use additional data that needs to be integrated with the last two algorithms. Classification will be the most complex part of the project due to the number of factors taken into account.
Test Crack Classification Algorithm	Testing will be done throughout the process. Both manual and automated testing will be used to ensure accuracy.
Develop UI	*Create a user interface that allows our client to upload a photo and obtain the results without running the algorithm directly.*
Create UI for Crack Detection	This UI will be completed first and will take the most amount of time. It will allow a user to upload a photo and obtain the data.
Integrate UI for Pavement Algorithm	This UI will be integrated as development for crack classification is happening.
Integrate UI for Crack Classification	This UI will be the final one developed for the project. This will be the one delivered to the client.
Full Product Testing	*After all the development has been completed and integrated, we will conduct full product testing using both automated and manual testing to verify overall accuracy of 80% crack classification.*

Table 3.3 - Task Decomposition

3.4 Possible Risks And Risk Management

As with any project, there will be an inherent risk. Below are the identified risks and mitigations to reduce the risk to the lowest possible severity.

Unmanned Aerial Flying

Risk

Flying UAVs require a specific skill set that not everybody on the team has. UAVs must adhere to FAA laws and since Iowa State University is within 5 miles of an airport, flight plans must be submitted in advance.

Mitigation

This risk is highly mitigated by our client's ability and experience flying UAVs. Our client has both the resources, skills, and knowledge to safely operate a UAV and has had 9 years' experience in UAV photography.

3.5 Project Proposed Milestones and Evaluation Criteria

The following major milestones will be used as a project guide to determine progress.

Crack Detection

The algorithm to detect cracks will be functioning and can accurately identify 80% of cracks in a photo. This accuracy will be determined by automated and manual testing.

Pavement Detection

The algorithm to detect cracks will be functioning and can accurately classify the pavement type in a photo with 80% accuracy. This accuracy will be determined by automated and manual testing.

Crack Classification

The algorithm is able to classify cracks found within a photo. This is considered complete when at least 80% of cracks can be correctly classified based on pavement type, shape, depth, and patterns. This will be validated through manual and automated testing.

UI Development

This will be considered complete when a user can upload a photo, start the algorithm, and receive the output image with 99% reliability.

Full Product Testing

After all the development and integration is complete, we will test the full product to ensure overall accuracy and reliability.

Deliver End Product

This will be complete when our client gets the full tested product.

3.6 Project Tracking Procedures

As stated above, our team will use an agile-like development process. Tools that will be used to track project progress will be as follows:

- Discord - Used for team member communication
- WhatsApp/Text Message - Used to communicate with client
- Email - Used to communicate between group and client
- Google Drive - Used to store documents needed for project and seamless collaboration for editing.
- GitLab/GitIssues - Repository for all code. Git Issues will be used to track all "fixes" that need to happen
- Trello - Since we are following an agile process, we will use Trello to track tasks, their progress, and completion.
- Other relevant tools will be used as needed.

3.7 Expected Results and Validation

The desired outcome for the projects will be in two stages: the identification of cracks, and the classification of cracks.

Crack Identification

The desired results for this will be that after the algorithm has been trained, it will be able to identify all cracks in concrete or asphalt. At this stage, the identification of cracks will not be dependent on the type of crack, but whether one is present there or not. To test our algorithm, we will use a series of pictures we currently have as well as new ones that we can take once the algorithm has been accurate with the current photos. At a high level, we will confirm the algorithm works when it can correctly identify at least 80% of cracks in the photos. Once we get to this accuracy, we will begin to move towards the next step while continuing the training of the algorithm to 100% of cracks.

Crack Classification

The desired result for this stage will be the algorithm being able to recognize the difference between a crack and a joint in concrete. As a stretch goal, the algorithm will also be able to identify different types of cracking patterns within concrete. To test the algorithm, we will use the same process as stated for crack identification. At a high level, we will confirm the algorithm works once it is able to correctly identify 80% of cracks and then refine the process to 100% of cracks.

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

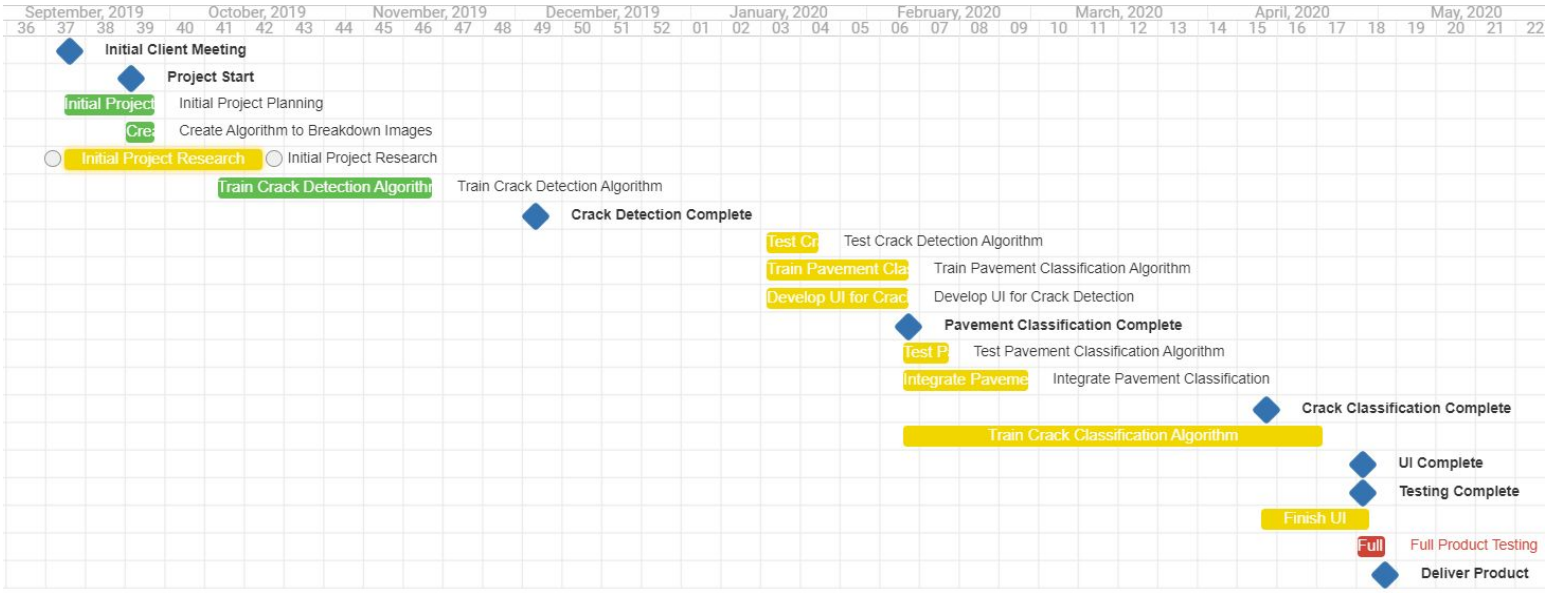

Figure 4.1 - Gantt Chart of Timeline

Task	Start Date	End Date
Client Meeting		9/12/2019
Project Start		9/22/2019
Initial Project Planning	9/12/2019	9/26/2019

Create Algorithm to Breakdown Images	9/22/2019	9/28/2019
Initial Project Research	9/12/2019	10/15/2019
Train Crack Detection Algorithm	10/9/2019	11/16/2019
Crack Detection Complete		12/3/2019
Test Crack Detection Algorithm	1/14/2020	1/21/2020
Train Pavement Classification Algorithm	1/14/2020	2/7/2020
Develop UI for Crack Detection	1/14/2020	2/7/2020
Pavement Classification Complete		2/7/2020
Test Pavement Classification Algorithm	2/7/2020	2/14/2020
Integrate Pavement Classification	2/7/2020	2/28/2020
Crack Classification Complete		4/10/2020
Train Crack Classification Algorithm	2/7/2020	4/20/2020
Finish UI	4/10/2020	4/27/2020
UI Complete		4/27/2020
Testing Complete		4/27/2020
Full Product Testing	4/27/2019	5/1/2020

Deliver Project		5/1/2020

Table 4.1 - Gantt Chart Product Timeline Key

This schedule has been made to allow time for our group to research, develop, and test each component as we build the product. As we continue to train the algorithm for accuracy, we will also be adding additional requirements. As a result, our timeline is accounting for the increased complexity for both development and testing.

In addition to the algorithm, we are also developing a user interface that will allow a user to use the algorithm without needing to run it through a command prompt. This will be developed alongside the algorithm and will be tested for reliability.

Overall, our timeline has been created to account for increasing complexity of the algorithm development, testing and remediation, experience of the team, and any additional work that will need to be completed.

4.2 Feasibility Assessment

This project should be able to be completed in the time given. Given the stages of the project, we project to have the crack detection portion finished at the end of the semester. In the spring semester, we will continue to work on identifying and classifying different types of cracks. The degree to which we are able to do this will be dependent on identifying patterns within the datasets.

4.3 Personnel Effort Requirements

The following chart is the estimated time breakdown of all the tasks. This timeline is subject to change. Task descriptions can be found in Table-3.3.

Task	Estimate Days to Complete
Project Setup and Start	48 Days
Client Meeting	1 Day
Initial Project Planning	14 Days
Initial Project Research	33Days

Train Crack Detection Algorithm	51 Days
Create Algorithm to Breakdown Images	6 Days
Train Algorithm	38 Days
Test Crack Detection Algorithm	7 Days
Train Pavement Classification Algorithm	55 Days
Train Algorithm	24 Days
Integrate Pavement Classification Algorithm	21 Days
Test Pavement Classification Algorithm	7 Days
Train Crack Classification Algorithm	73 Days
Train Algorithm	45 Days
Integrate Crack Classification Algorithm	14 Days
Test Crack Classification Algorithm	14 Days
Develop UI	40 Days
Create UI for Crack Detection	24 Days
Integrate UI for Pavement Algorithm	8.5 Days
Integrate UI for Crack Classification	8.5 Days
Full Product Testing	4 Days
Days Allocated for Project	**273 Days**

Table 4.3 - Personnel Effort Requirements

4.4 Other Resource Requirements

Outside of financial resources, our project will need access to a UAV, a High-Performance Computing (HPC), and a dataset to train the model. Our client has provided us access to the UAV photography he has previously taken and has

offered the ability to take more photos should we need it. The HPC will be available through a cluster provided by the Department of Electrical and Computer Engineering. Lastly, the dataset and library we will be using are both open source.

4.5 Financial Requirements

The financial resources for this project are limited. The products we are using are open source and the high-performance computing cost will be covered by the Department of Electrical and Computer Engineering.

5. Testing and Implementation

To test and implement our project, we need to verify that our algorithm works on a variety of photos that could include grass, cars, puddles, shadows, and joints. To test these different scenarios, we are doing functional testing. Functional testing in our case is running our trained epoch on specific hand-picked images. These images relate to each of the possible cases that could affect crack detection (shadows, puddles, cars, etc.). Next, depending on the accuracy of the detected cracks with the assigned obstacle(s), we then change our algorithm accordingly. Then, we retrain the updated epoch and check the accuracy of the crack detection in the same image. We repeat this process until the final accuracy rate of the picture meets our standards.

5.1 Interface Specifications

In the future, we will have an interface that allows us to select which epoch we want to run. We will also be able to specify a folder or a group of images that we want the epoch to be run on. The interface will then take the response (accuracy rate and edited images), create a pop-up window with buttons such as “open containing folder” so the user is easily able to access the edited images, “edit code” for when the accuracy rate is not satisfactory, “run again with different epoch” to allow the user to run the same images with a different epoch, “run again with different images” for when the user wants to run the same epoch on different images. Currently, we run our python scripts from the command prompt and use the HCP from Iowa State to train our algorithm to create epochs.

5.2 Hardware and software

No already existing software has been used in our testing phase thus far. However, we developed some useful python scripts to help assure the input images are correct for training our algorithm. One of these scripts is being used to resize any input picture to the necessary size of 128p x 128p.

5.3 Functional Testing

For our functional testing, we took some of the most common images in our test set and had our trained epochs analyze them. This took many trials. Once we were satisfied with the success rate of the crack detection on the basic set of images, we went on to have the epochs analyze the less common cases. These cases include images with certain objects that could mess with the analyzation of cracks using our algorithm.

5.4 Non-Functional Testing

Currently we have tested for testability, accuracy, timeliness, and reliability. Most of these non-functional requirements were tested while training and testing the epochs. First off, we made sure there was a way to test our program. Then, we made sure the program gives an accurate output, gives an output in a timely manner, and is reliable in the sense that it rarely crashes. In the future, we will do more non-functional testing to make sure our program meets a multitude of non-functional requirements. Some examples of other non-functional requirements that we will test for include customizability (such as inputting images of any size), maintainability (the system is able to be easily repaired), efficiency (no unnecessary loops or repeated code), evolvability (ability to modify program according to the client's needs), learnability (how easily the client is able to understand what the output program means), stability (the rate of consistently accurate results), and operability (how easily the client is able to run the program).

5.5 Process

Our testing and implementation process started by looking at how our algorithm was analyzing the test images. One after another, through trial and error, we were able to come up with certain values that seem to work best for detecting cracks in the road.

5.6 Results

Our algorithm is currently able to detect cracks using a 97% accurate trained epoch. Our most recent project milestone to be completed is identifying cracks. When the model was initially being trained, the epoch score was at 67%. Since then, the epoch score has risen to 97%. We learned, after setting up the model, that accuracy can be increased by rotating through datasets.

Figure 5.6 - Current Algorithm Output (Epoch Score of 97%)

6. Closing Material

6.1 Conclusion

Our project is to create a product that helps with identifying and classifying cracks. The end goal is to deliver a quality product to our client that will classify cracks based on the type of pavement and previously identified crack patterns. We will work to achieve the end goal by completing the major milestones according to the schedule outlined within this document.

6.2 References

[1] Cha, Young-Jin & Choi, Wooram & Buyukozturk, Oral. (2017). Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks. Computer-Aided Civil and Infrastructure Engineering. 32. 361-378. 10.1111/mice.12263.

[2] L. Zhang, F. Yang, Y. Daniel Zhang and Y. J. Zhu, "Road crack detection using deep convolutional neural network," 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, 2016, pp. 3708-3712

6.3 Appendices

Concrete Data Set:
https://data.mendeley.com/datasets/5y9wdsg2zt/2#file-cod86f9f-852e-4d00-bf45-9a0e24e3b932

GitHub Repositories:

- https://github.com/pytorch/examples/tree/master/mnist
- https://github.com/pytorch/examples/tree/master/imagenet
- https://github.com/warmspringwinds/pytorch-segmentation-detection
- https://github.com/fyangneil/pavement-crack-detection
- https://github.com/satyenrajpal/Concrete-Crack-Detection
- https://github.com/Sarthakdtu/Road-Cracks-Detection-Neural-Network-