International Institute of
Information Technology Bangalore

Project Progress Report
DS/NC/ESD 863 Machine Perception

Vehicle Driving Assistant

Akanksha Dwivedi - MT2016006
Anoop Toffy - MT2016016
Athul Suresh - MT2016030
Tarini Chandrashekhar - MT2016144

Group 7

Guide :
Prof. Dinesh Babu Jayagopi

May 11, 2017

Contents

List of Figures

1 Problem Statement

The project aims to provide a comprehensive set of assistance features to aid the driver (or autonomous vehicle) to drive safely. This includes a number of indicators about the environment, the major cue being detection of potholes in the road ahead.

2 Motivation

Autonomous vehicles has been a common term in our day to day life with car manufacturers like Tesla shipping cars that are SAE Level 3. While these vehicles include a slew of features such as parking assistance and cruise control, they've mostly been tailored to foreign roads. Potholes, and the abundance of them, is something that is unique to our Indian roads. We believe that successful detection of potholes from visual images can be applied in a variety of scenarios. Moreover, the sheer variety in the color, shape and size of potholes makes this problem an apt candidate to be solved using modern machine learning techniques.

3 Related Works

In Nienaber et al (2015) [1], a system using basic image processing techniques in a constrained environment without relying on any machine learning techniques is used for pothole detection. It presents a good preliminary method for detecting potholes using a single camera within an range of 2 - 20m from a vehicle moving at a speed of not more than 60km/hr. The method separates a rectangular area of interest just above the hood of the vehicle which contains road surface, assuming that driver maintains a safe distance from the front vehicle. The rectangular area of interest is separated by connecting the various farthest region of interest using convex hull algorithm.

The work presented by Ajit Danti et al (2012) [2], presents a comprehensive approach to address the acute problems of Indian roads such as faded lanes, irregular potholes, improper and invisible road signs. Instead of using image processing techniques for pothole detection as done by Nienaber et al (2015), Ajith Danti et al (2012) uses K-Means clustering based algorithm to detect potholes. By addressing the acute problems above mentioned in the paper it makes automated driving safer and easier in Indian roads.

4 Progress

We started out with the naive approach - try to detect the potholes directly from pictures using image processing techniques. We collected a small dataset from Google images and applied classic techniques like edge and contour detection. The problem with Google images of potholes is that they're heavily biased - the potholes are shown in such a way that it is the center of focus in the picture, with good lighting and clear demarcation from the environment. Hence our initial set of technique worked for a limited subset, the results were far from satisfactory. We got too much noise and too little usable data to go on. Detecting potholes, which may be large or small, circular or quadrilateral, filled with liquid or dust and overall varied in nature, was not a straightforward problem.

The following list of figures demonstrates the approaches we did initially to detect potholes using image processing techniques:

- Blob Detection Using LoG, DoG, DoH

  In the Blob Detection Technique, the Laplacian of Gaussian(LoG), Difference of Gaussian(DoG) and the Determinant of Hessian(DoH) was found out with arguments max_sigma as 30, num_sigma as 10 and threshold as .1 in the input frames. It was observed that the LoG and DoG was detecting the potholes, additional to the noise in the frames and the DoH methodology detected only a fraction of the potholes.

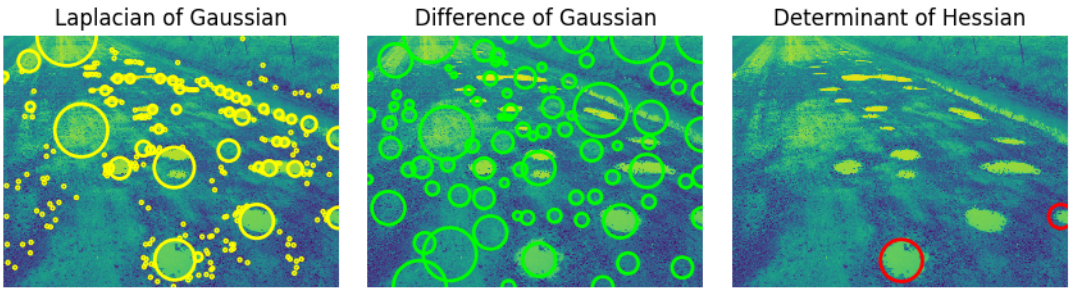

Figure 1: Output of the three Blob Detection Techniques on Google Images

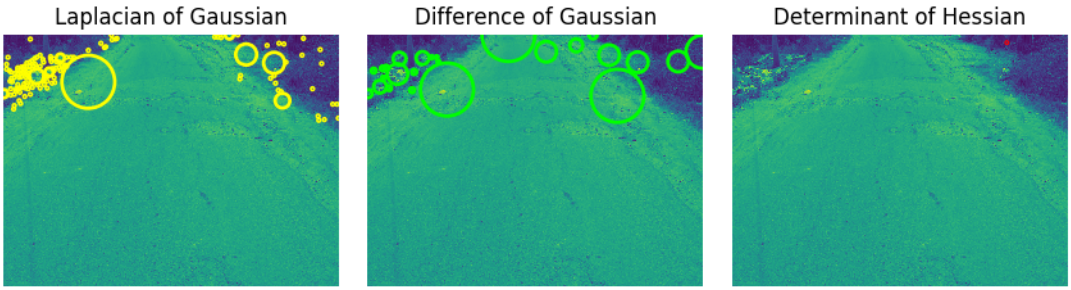

Figure 2: Output of the three Blob Detection Techniques on Dataset

- Contour Detection with K-Means

  In the contour detection method with K-Means, we used the K-Means algorithm with 30 iterations and 3 clusters. The Gaussian Blur was applied to the output of the K-Means to remove any noise in the frames. Finally, the Canny Edge Detection algorithm was used to find out the edges and contours around potholes in the frames. This technique detected potholes and some of the noise in the Google Images dataset but it did not work with our dataset. It missed the contours for potholes and detected shadows in some of the frames.

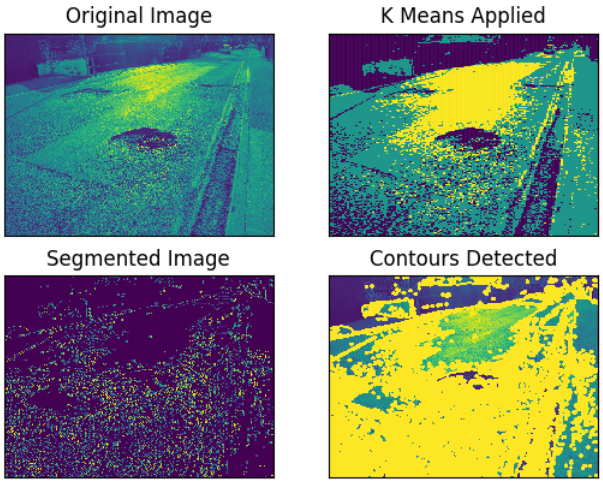

Figure 3: Output of Contour Detection using K-Means

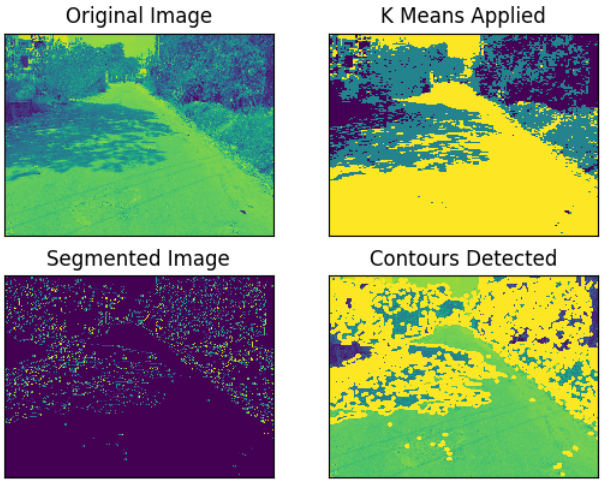

Figure 4: Output of Contour Detection using K-Means

- Contour Detection with Meanshift

  Using the idea of the previous methodology of Contour Detection, we used Meanshift as the clustering algorithm instead of the K-means algorithm. The inbuilt function pyrMeanShiftFiltering was used to filter and segment the image with arguments Spatial Window Radius as 25, Color Window Radius as 45 and Maximum Level Of Segmentation as 10. Similar to the previous methodology, Gaussian Blur and Canny Edge Detection were applied to find the contours in the frames.This method worked perfectly on the Google Images Dataset but detected the potholes partially in our dataset with the additional noise.

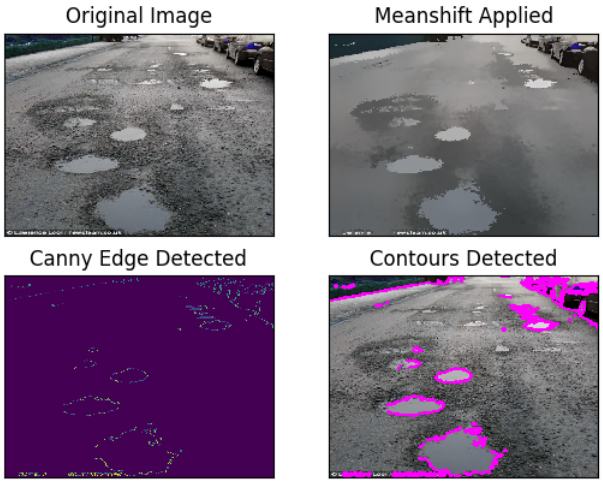

Figure 5: Output of Contour Detection using Meanshift

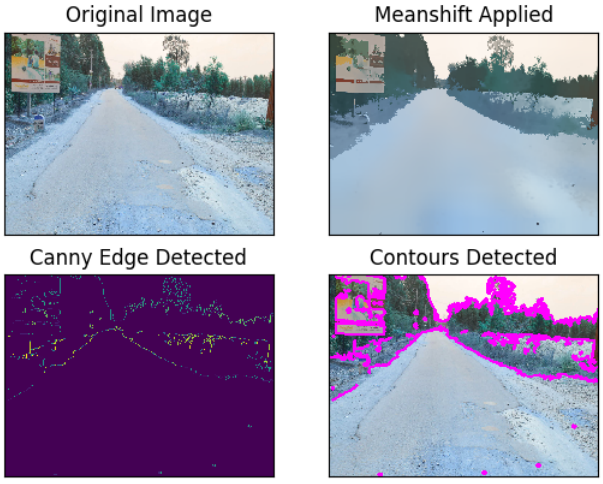

Figure 6: Output of Contour Detection using Meanshift

- Contour Detection with Morphological Transformation

  In the Contour Detection method with Morphological Transformations, we used erosion to detect contours around potholes in the images. Gaussian Blur and thresholding was applied on the input frames to remove the noise signals from the image. The results of Image Gradients: Laplacian, Sobel X, Sobel Y were compared, after applying them on the blurred image. Among all, Sobel Y, performed the best. Then, erosion was applied to this output frame, with the medium sized kernel, to narrow down the search for potholes in the frames. Contours were being detected in the final frame. This technique worked on the Google Images dataset with partial noise being detected but detected too much noise in our dataset along with a fraction of potholes.

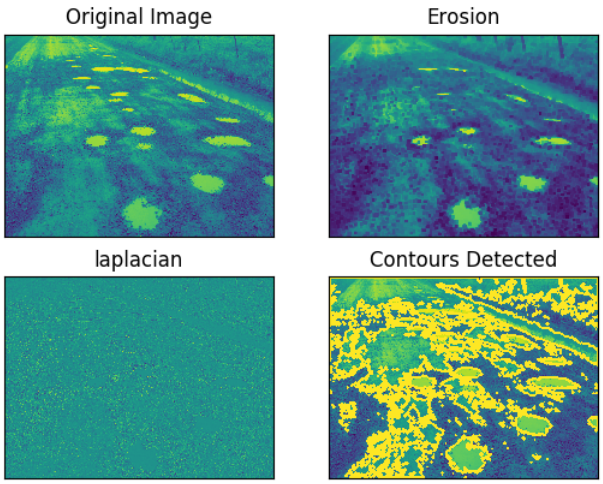

Figure 7: Output of Contour Detection using Morphological Transformation

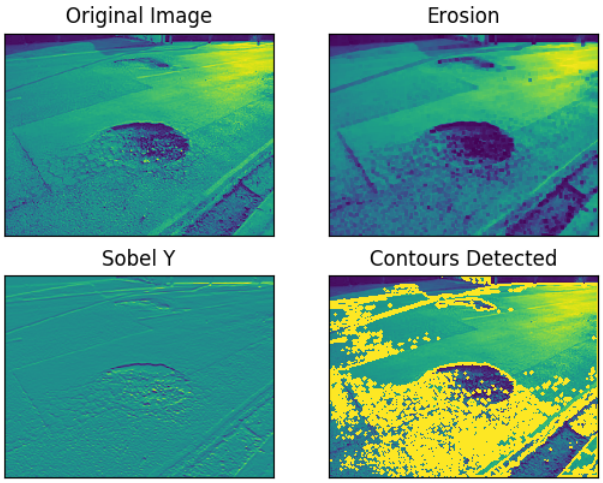

Figure 8: Output of Contour Detection using Morphological Transformation

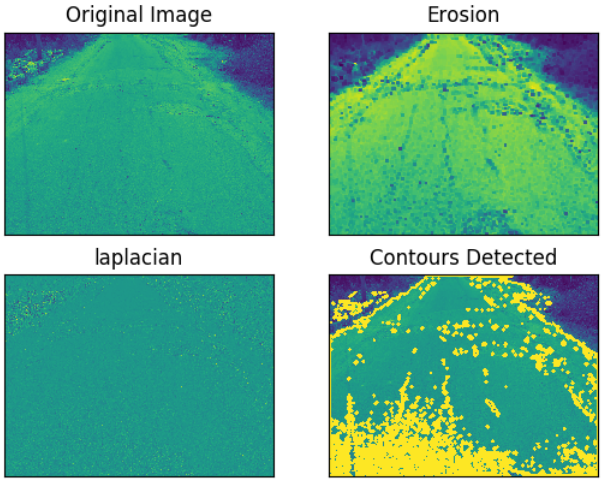

Figure 9: Output of Contour Detection using Morphological Transformation

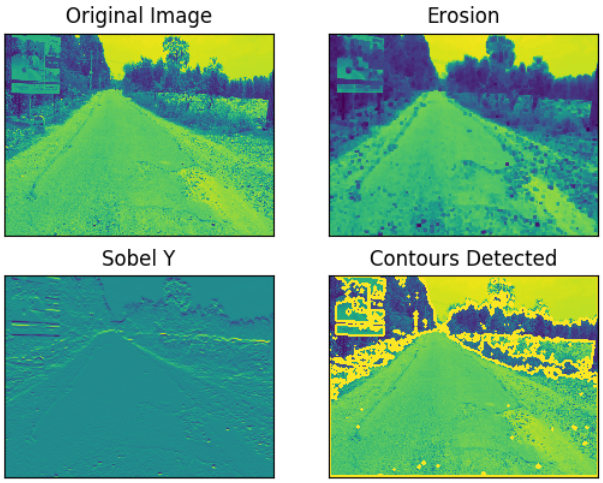

Figure 10: Output of Contour Detection using Morphological Transformation

- Classic Watershed Algorithm

  In this methodology, we used the classic Watershed Algorithm for the segmentation. The Watershed Algorithm works as follows: The input image, being converted into the grayscale, is thresholded with Binary Inverse and Otsu's thresholding.The morphology transformations, opening and dilating, are applied to extract the background area from the image. To find the foreground area in the image, Distance Transform is being applied with L2 Transform as argument. Both the background and the foreground regions are marked and this output is fed to the watershed function to get the output of segmentation. This algorithm worked absolutely fine on the Google Images Dataset.

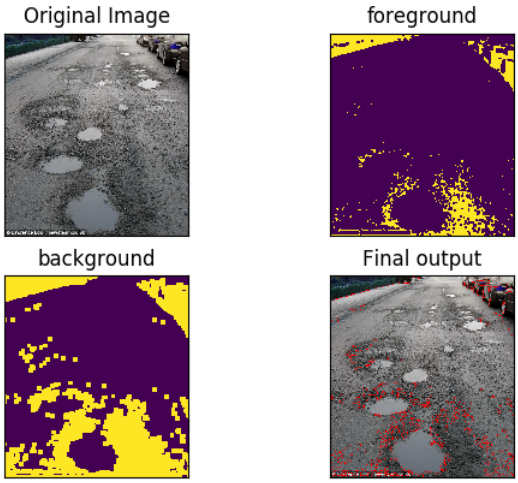

Figure 11: Output of Watershed Algorithm

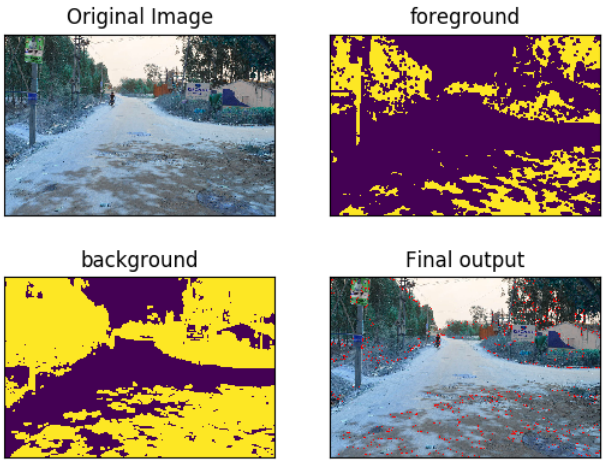

Figure 12: Output of Watershed Algorithm

- Potholes Detection using Nienabar's Paper

  This technique referred one of the papers we followed, by the author Nienabar et al [1]. The input frame was to be converted to one of the color channels for the next steps. The results of Grayscale, Hue,

saturation and variance channels were compared. Among these, the results came out best with the saturation channel. Gaussian Filter and Canny Edge Detection were applied to detect the edges. Finally, dilation was performed on the output frames, in 4 iterations, to bring out the edges/contours around potholes.

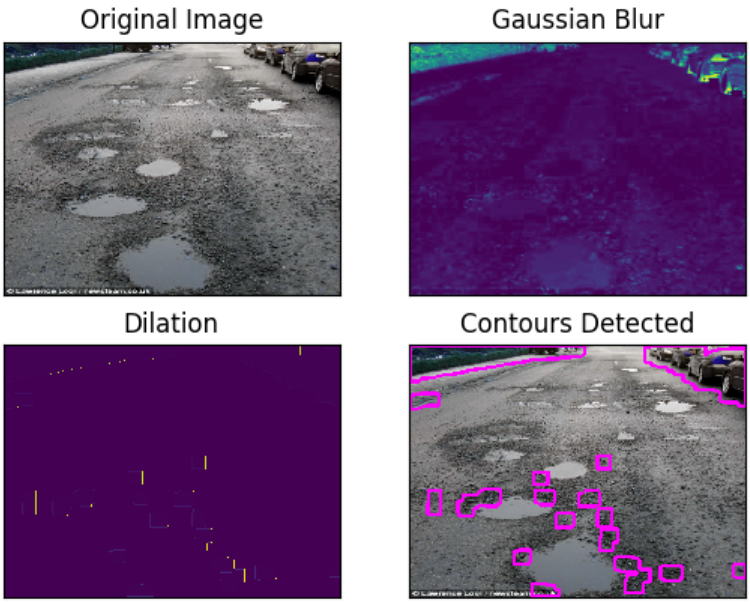

Figure 13: Output of Nienabar's Algorithm

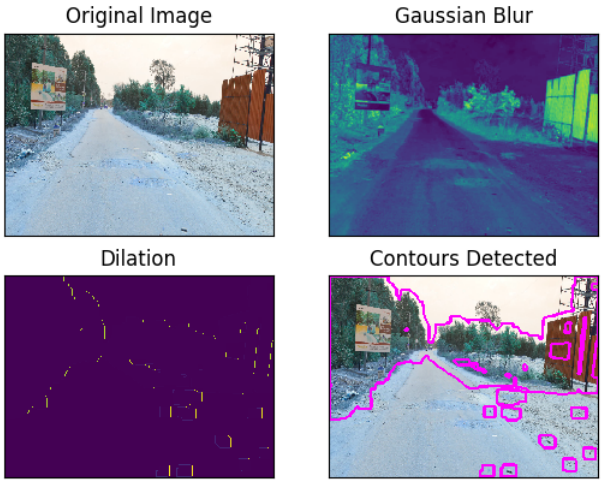

Figure 14: Output of Nienabar's Algorithm

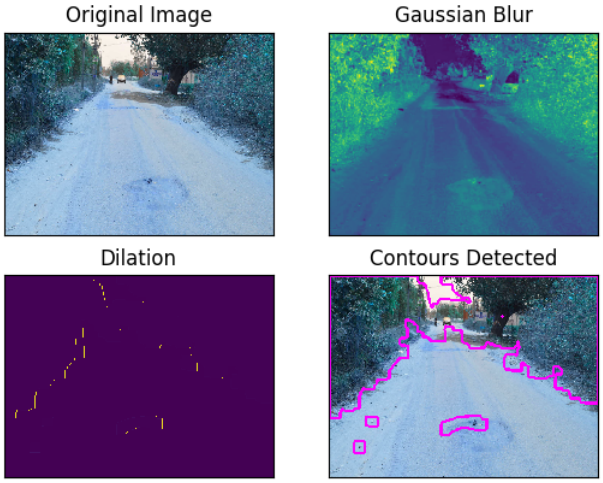

Figure 15: Output of Nienabar's Algorithm

In order to get dataset that is rooted in the real world, we went out and recorded 360p video of the roads in and around Electronic City phase 2. We managed to get about 20 minutes of footage of varying terrains with diverse features that are commonly encountered on Indian roads. The footage was preprocessed to reduce jitter induced in the video due to it being recorded

by the pillion rider of a moving motorcycle.

Collected data: `https://youtu.be/hmeBmdZzlLU`

We expected the detection of potholes to be simplified if a localized area in the image can be found where the chance of occurrence of the pothole is relatively high. In essence, we decided to extract the road from the surrounding environment.

In order to remove the external environment and extract the road from the given frame, we tried a variety of techniques.

- Choose a channel

  We had to come up with an input representation where the difference between the road and the surrounding was accentuated. We tried Grayscale images but the contrast wasn't enough to facilitate extraction. Much better results were found by converting the frame to HSV colorspace. We tried each channel in isolation and got satisfactory results with the S and V channels.

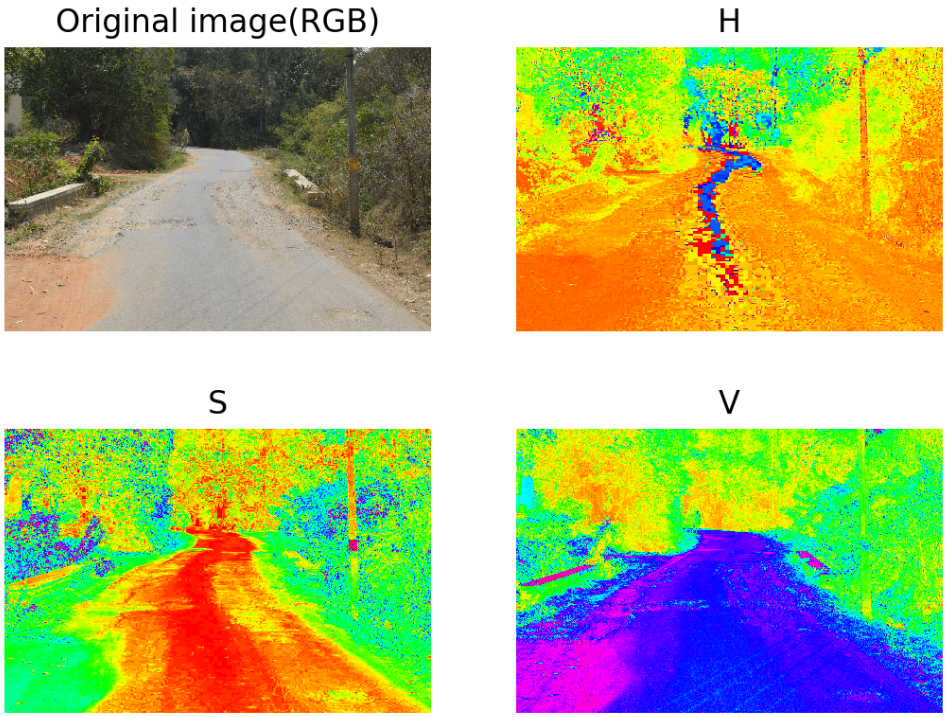

Figure 16: HSV channels for classic image

- Remove Noise

  Now that we had a channel to work on, wee had to reduce the amount of noise in the frame. Between Gaussian Blur and Median Blur, we got better results with Median Blur with a relatively large kernel size to reduce the noise inherent in the picture.

- Threshold to get a suitable mask

  Now that we had a denoised version of the channel with maximum variance between the road and surrounding, we applied Otzu's thresholding to extract the part of the image containing the road. We got a relatively good output but there was still noise in the threshold mask.

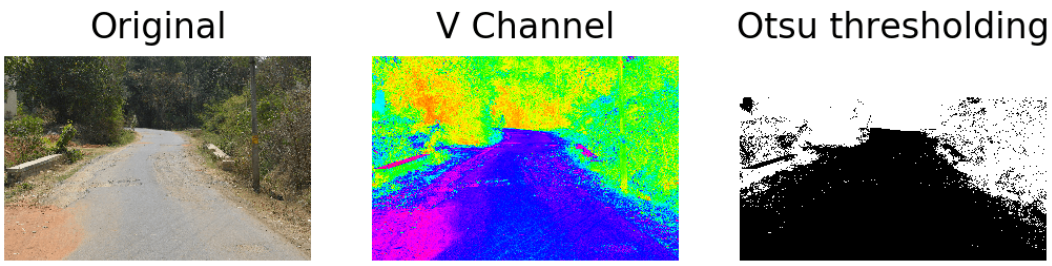

Figure 17: Thresholding to create a mask

- Clean up the mask

  Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is the original image, second one is called structuring element or kernel which decides the nature of operation. Two basic morphological operators are Erosion and Dilation. ** Erosion: The basic idea of erosion is to erode away the boundaries of foreground object. It is useful for removing small white noises, detach two connected objects etc. ** Dilation: It is just opposite of erosion. It increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So, we dilate it. Since noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object.

  There were still stray islands in the mask, i.e small localized black(white) regions within the white(black) areas. In order to even these out, we used the following two morphological transformation methods in succession - ** Closing : It is dilation followed by erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object. ** Opening : It is erosion followed by dilation. It is useful in removing noise from the black areas. We used a relatively large kernel for both the operations to get a fairly clean mask.

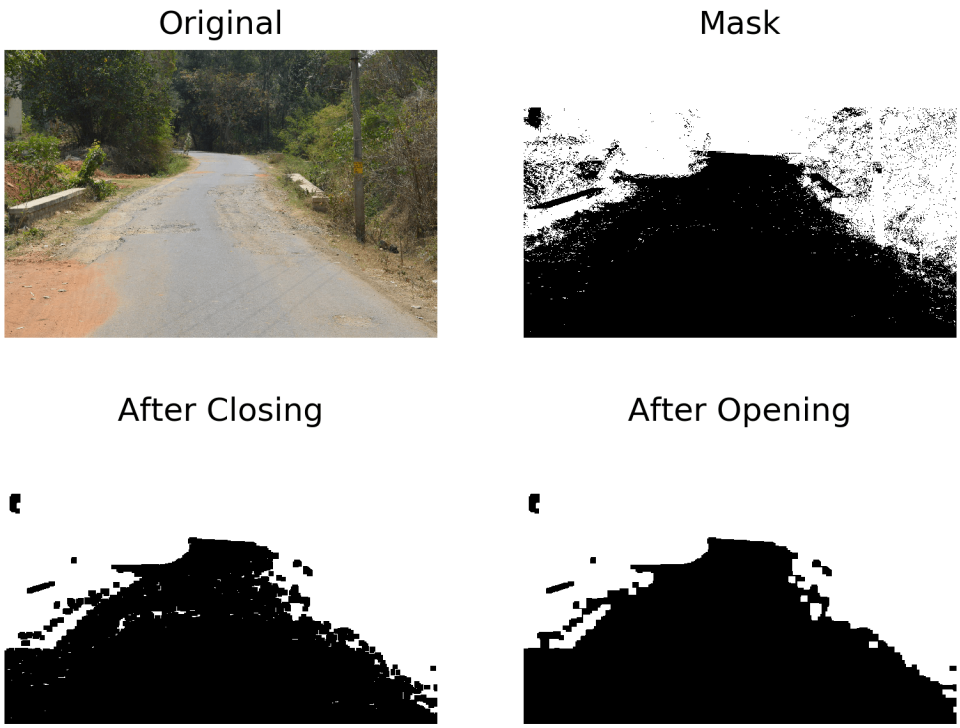

Figure 18: Opening and Closing to clean up the mask

- Application of mask

  The obtained mask was applied on the original image and the road was extracted out.

Figure 19: Application of mask

Mask applied on collected dataset: `https://youtu.be/hmeBmdZzlLU`.

5 Future Work

To analyse the frames more precisely we will be collecting additional dataset with improved frame rate and resolution.One which we will carry out a set of image preprocessing techniques to clear the jitter and apply a set of methods to detect potholes precisely. Future enhancements of this work is explained briefly in the following paragraph.

5.1 Supervised Learning

We will be generating manually labelled data from the images collected. Using the labelled image data we apply classical supervised learning techniques to find out the presence of potholes in the image. We will be extracting each frame separately from the video and segregate the frames that has potholes. SURF or SIFT detectors are used to extract features from the images which is used for learning.

5.2 Classification

After the features are extracted we will be using various machine learning techniques to classify whether or not the given frame from a video contains potholes. We use methods like Naive Bayes, KNearest Neighbour, SVM and the like to train the model and predict the existence of the potholes.

5.3 Comparative analysis

A study is then done based on the result obtained from the classical machine learning methods to find out which method works best in a given scenario.

References

[1] S Nienaber*, M Booysen* AND R Kroon**. *Detecting potholes using simple image processing techniques and Real-world Footage, 2015.* *Department of E&E Engineering, Stellenbosch University. **Computer Science Division, Stellenbosch University `http://scholar.sun.ac.za/handle/10019.1/97191`

[2] Ajit Danti, Jyoti Y. Kulkarni, and P. S. Hiremath, Member, IACSIT *An Image Processing Approach to Detect Lanes, Pot Holes and Recognize Road Signs in Indian Roads, December 2012* `http://www.ijmo.org/papers/204-S3015.pdf`