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

April 17, 2017

Contents

1. 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. Then its variant forms like Opening, Closing, Gradient etc also comes into play.

   (a) **Erosion** The basic idea of erosion is to erode away the boundaries of foreground object. The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).

   So, all the pixels near boundary will be discarded depending upon the size of kernel. The thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises, detach two connected objects etc.

   (b) **Dilation** It is just opposite of erosion. Here, a pixel element is '1' if atleast one pixel under the kernel is '1'. So, 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.

   (c) **Opening** Opening is just another name of erosion followed by dilation. It is useful in removing noise.

   (d) **Closing** Closing is reverse of Opening, Dilation followed by Erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object.

   (e) **Morphological Gradient** It is the difference between dilation and erosion of an image.

1. Pre-processing Techniques:

   (a) **Method 1:Blob Detection Using LOG, DOG, DOH**

      i. 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: Original Image from the Google Images Dataset

Figure 2: Output of the three Blob Detection Techniques

Figure 3: Original Image from the Collected Dataset

Figure 4: Output of the three Blob Detection Techniques

Figure 5: Original Image from the Collected Dataset

Figure 6: Output of the three Blob Detection Techniques

(b) **Method 2: Contour Detection with K-Means**

i. 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 7: Original Image from the Google Images Dataset

Figure 8: Output of Contour Detection using K-Means

Figure 9: Original Image from the Collected Dataset

Figure 10: Output of Contour Detection using K-Means

Figure 11: Original Image from the Collected Dataset

Figure 12: Output of Contour Detection using K-Means

(c) **Method 3:Contour Detection with Meanshift**

i. 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 13: Original Image from the Google Images Dataset

Figure 14: Output of Contour Detection using Meanshift

Figure 15: Original Image from the Collected Dataset

Figure 16: Output of Contour Detection using Meanshift

Figure 17: Original Image from the Collected Dataset

Figure 18: Output of Contour Detection using Meanshift

Figure 19: Original Image from the Collected Dataset

Figure 20: Output of Contour Detection using Meanshift

(d) **Method 4: Contour Detection with Morphological Transformation**

i. 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 alongwith a fraction of potholes.

Figure 21: Original Image from the Google Images Dataset

Figure 22: Output of Contour Detection using Morphological Transformation

Figure 23: Original Image from the Google Images Dataset

Figure 24: Output of Contour Detection using Morphological Transformation

Figure 25: Original Image from the Collected Dataset

Figure 26: Output of Contour Detection using Morphological Transformation

Figure 27: Original Image from the Collected Dataset

Figure 28: Output of Contour Detection using Morphological Transformation

Figure 29: Original Image from the Collected Dataset

Figure 30: Output of Contour Detection using Morphological Transformation

Figure 31: Original Image from the Collected Dataset

Figure 32: Output of Contour Detection using Morphological Transformation

Figure 33: Original Image from the Collected Dataset

Figure 34: Output of Contour Detection using Morphological Transformation

(e) **Method 5: Classic Watershed Algorithm**

i. In this methodolgy, 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. It detected almost 90% of the potholes in our dataset along with the minor noise and the shadows.

Figure 35: Original Image from the Collected Dataset

Figure 36: Output of Watershed Algorithm

Figure 37: Original Image from the Collected Dataset

Figure 38: Output of Watershed Algorithm

Figure 39: Original Image from the Collected Dataset

Figure 40: Output of Watershed Algorithm

Figure 41: Original Image from the Collected Dataset

Figure 42: Output of Watershed Algorithm

Figure 43: Original Image from the Collected Dataset

Figure 44: Output of Watershed Algorithm

Figure 45: Original Image from the Collected Dataset

Figure 46: Output of Watershed Algorithm

Figure 47: Original Image from the Collected Dataset

Figure 48: Output of Watershed Algorithm

(f) **Method 5: Potholes Detection using Nienabar's Paper**

i. This technique referred one of the papers we followed, by the author Nienabar, Boysoon and Kroon. 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. Among all the methodologies being used, this methodology gave the best results both on the Google Images dataset and our dataset, despite of the shadow and illumination setbacks.

Figure 49: Original Image from the Google Images Dataset

Figure 50: Output of Nienabar's Algorithm

Figure 51: Original Image from the Google Images Dataset

Figure 52: Output of Nienabar's Algorithm

Figure 53: Original Image from the Collected Images Dataset

Figure 54: Output of Nienabar's Algorithm

Figure 55: Original Image from the Collected Images Dataset

Figure 56: Output of Nienabar's Algorithm

Figure 57: Original Image from the Collected Images Dataset

Figure 58: Output of Nienabar's Algorithm

Figure 59: Original Image from the Collected Images Dataset

Figure 60: Output of Nienabar's Algorithm

Figure 61: Original Image from the Collected Images Dataset

Figure 62: Output of Nienabar's Algorithm