ME 5405 Machine Vision Project Report

Submitted by -

Khakimjon Saidov (A?????)

Utkarsh Anand (A0147589J)

Introduction

In the last 4 decades, machine vision has had an important role in industry, security, medicine, and academia. Humans vision is very diverse, for example we can distinguish between different objects and can isolate the background from the object of interest very efficiently. But what humans vision is not very good at is performing the same task repetitively as we tend to make more mistakes and lose neutrality as we do the same thing over and over again. Inspection of manufactured goods for quality control, surveillance of a traffic junction for road safety, and detection of abnormalities in a MRI scan are few of the examples where machine vision is extremely useful. While researchers have extensively been working to develop better and faster image processing algorithms to do more complex and demanding jobs, the rapid growth by the semi-conductor industry to make cheaper and faster computers has completely revolutionized machine vision. Nowadays, a lot of processes can be done fast because of the quick feedback provided by the state-of-the-art image processing algorithms running on state-of-the-art computers. Through this module we learn the basic processes involved in image processing and we apply some of this knowledge to process two different types of images.

Overview

This report comprises of two parts that describe our strategy to perform the required operations on the two different images. The image processing codes were first tested in Octave (because of its lower memory requirement), and after all the algorithms worked as expected they were checked for compatibility with MATLAB. We tried to implement most of the functions on our own, and as a result they are not optimized and take longer to run. The codes are also available on www.github.com (link).

For each image we are required to do the following operations -

1. Display the original image.
2. Create binary image using thresholding.
3. Separate and identify different characters.
4. Rotate the characters clockwise by 90 degrees.
5. Rotate the characters anti-clockwise 35 degrees.
6. Find the boundary of all the characters.
7. Find the one-pixel thin image of all the characters.
8. Rearrange the characters in a particular sequence.

Processing of Image 1

Part 1

The original image is in the form of a text file ‘charact1.txt’ (Figure 1). We use the inbuilt function ‘fileread()’ to read the contents of the input file as a vector of characters. This vector is then passed as an argument to a custom defined function ‘textToascii()’. The textToascii function creates a new empty vector, and then goes through each of the characters of the input vector one-by-one. Then the character is converted to its corresponding ASCII value using the inbuilt function ‘toascii()’ which converts the character to an integer in the range 0-255. If the ASCII character does not have a value of 10 (corresponds to new line) or 13 (corresponds to carriage return), it is valid and is appended to the empty vector. At the end of this operation we get a new vector of numbers of length 4096 elements.

Next, the problem statement says that the image is 64 pixels x 64 pixels, and the vector of numbers is reshaped accordingly. The first 64 elements of the vector form the first row of the image, elements 65 to 128 form the second row of the image, and so on. This generate a grayscale image, with 32 unique intensity values and a minimum and maximum intensity of 48 (corresponds to character 0) and 86 (corresponds to character V) respectively.



