Automated drug detection and location identification for visually impaired using image processing and voice commands

By

Utshab Roy – (13101217)
Md. Ashikur Rahman – (13101048)
Faysal Bin Hasan – (13301127)

Supervisor
Dr. Md. Ashraful Alam

Department of Computer Science and Engineering
BRAC University

Thesis report submitted to BRAC University in accordance with the requirements for the degree of Bachelor of Science in Computer Science & Engineering

Submitted in December 2017

Declaration

We, hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researchers are mentioned by reference. This thesis, neither in whole or in part, has been previously submitted for any degree.

Signature of the Supervisor

_____

Dr. Md. Ashraful Alam

Signature of Author

_____

Utshab Roy

Signature of Author

_____

Md. Ashikur Rahman

Signature of Author

_____

Faysal Bin Hasan

Acknowledgement

Our heartfelt gratitude to our thesis supervisor Dr. Md. Ashraful Alam for their generous guidance and continuous support throughout our work. Without their help and assistance, we wouldn't be able to finish our research successfully.

We are extremely thankful to our parents, family members and friends for their support and encouragement. The journey would be much harder if they weren't present for us in every moment whenever we needed them.

We would also like to take this opportunity to thank Mr. Touhid Hossain, Mr. Kazi Rezaul Karim – our Lab Technical Officers who made sure we got all the technical support we needed.

Finally we thank BRAC University for giving us the opportunity to use their resources and complete our thesis in the university.

Abstract

This report proposed a system which will aid visually impaired to detect and locate medicine (drug) using voice commands. It is a challenge to build a system which will assist a person how has eye sight difficulties. So we took that challenge and tried to build a system which will able to locate any medicine location around that person's surroundings. This system includes optical character recognition (OCR) and depth segmentation of Kinect sensor in order to process extraction of characters from frames in real time video. For extracting the text name of the medicine we tested several OCR engines which includes – Google-Tesseract-OCR, and Aspose-OCR. To process images we used emguCV which is a .NET wrapper of the Intel OpenCV image-processing library. In order to operate voice commands we used Microsoft speech recognition library. Finally, the system enables instructions which will guide a visually impaired person to find his/her medicine for daily usage.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Optical character recognition used for turn a picture into a text form, which means convert JPEG or PNG formatted image file into plain text. This method is necessary to perform an operation on images to get or extract vital information or enhance the original image quality. Nowadays, image processing is one of the rapidly growing technologies. It creates core research area within computer vision, engineering and computer science disciplines too. Optical character recognition (OCR) is a part of image processing research tool. Image processing has several application which we use concurrently in our life. For example, face and eye recognition using HAAR cascade, text extraction using google tesseract library, 2D image into 3D image, detecting license plate, X-ray images, extracting urban, forest, agriculture fields in google map aerial images. Image processing follows two types of methods to process data, which includes analog and digital processing. Analog processing generally used for scan images, hard copy of printouts and photographs. On the other hand, digital image processing includes pre-processing, enhancement, display, and information extraction. These fundamental method of image processing inspired us to work on this field. As it is a new form of technology that has an involvement in medical imaging, portrait mode edge detection, enhance color saturation, temperature sensing and many more. We designed our thesis work not only as an application but also a scope for further advance research for implantation in hardware. It is noticeable that technological firms and companies has a few sets of devices in present market for visually impaired communities in order to guide them in daily basis. Those devices are not efficiently assisting them and those are barely affordable for everyone. So, we think that our proposed solution will make an impact to society that visually impaired people need more attention for their betterment and also need technical support like we tried with in our work. Thus, we find ourselves honored and enthusiast to make a contribution into a vast research area of image processing.

1.1 Literature Review

The image processing method has reached reliable state throughout the modern technology. However engineers are trying so hard to make it more efficient. In this purpose many researcher has already developed their works which has raised significant impact on image processing. To illustrate the related work of this field, “extracted 3D location of object in real-time image recognition with existing object detection models (YOLO)” that converts 3D objects into recognized audio output to visually impaired person. (R. Jiang, Q. Lin, and S. Qu, 2016) [1]. This application represent an innovative idea of image processing that can be a good approach to aid impaired people. Another approach of image processing is “Optical Character Recognition Based Speech Synthesis System Using LabVIEW” (S. K. Singla and R.K.Yadav, 2014) [2]. They tried to build a system for visually impaired person based on sound recognition. Blind people cannot see but they can easily response to sound. Whit that idea into the mind they build a cost efficient system where any visually impaired people can gather knowledge. They have built this system using Laboratory virtual instruments engineering workbench which is called LabVIEW. Microsoft Kinect is an advanced hardware that can be used in computer vision. By this regards, there are some work implemented related to image processing. “Blind navigation support system based on Microsoft Kinect” (V. Filipe, F. Fernandesb, H. Fernandesc, A. Sousad, H. Paredese, And J. Barroso, 2012)[3] is one of them. Kinect has many built-in sensors including depth sensor and RGB (Red, Green and Blue) sensor. It can segment any picture into RGB color and depth value. Using convocational neural networks, this system extract relevant features from the scene, and detect possible obstacles. Similar approach has been done by another group. There topic is “Detecting objects using color and depth segmentation with Kinect sensor” (J. Opeza, A. Olveraa, J. ireza, F. Butandaa, M. Manzanoa, D. Ojedab, 2012)[4]. They generally work for the indoor environment because this method has the disadvantage of been very sensitive to the changes on lighting. There method improve the accuracy of the color segmentation but their limitation is that for multiple object of same color cannot be discovered at the same time. They introduced an algorithm to detect object using CIE-Lab and depth segmentation techniques.

Image processing has a positive impact on medical field. To diagnosis various types of medical disorders, “Assessment Retinal Vessel Segmentation Using Morphological Operation and Threshold” (B.Sindhu, J.B.Jeeva, May 2013) is implemented using image segmentation, applying

threshold value, RGB (Red, Green, Blue) to Green channel conversion. They actually applied image segmentation into medical image and process Green channel conversion of RGB image. Then, they applied threshold value in purpose of extracting blood vessel from those image. As an output result, they showed segmentation method accuracy sensitivity value as well as specificity value in percentages shows better realization of medical images. Another implementation of image processing relates with medical image is developed that is "Detection of Ulcer in the Gastrointestinal Tract Using the Wireless Capsule Endoscopy Images" (Shirly, Adin and C. N. Savithri, 2016). They have propose a system that is a fully automated computer aided detection system which is used to detect ulcer from the Wireless Capsule Endoscopy (WCE) images. For preprocessing of image they used multidimensional filter and their proposed method segments the image into multi-level super pixel segmentations. They used erosion and dilation to detect edge of image. In their final output, their system able to detect ulcer using Wireless Capsule Endoscopy images.

To illustrate image processing, any system need to use image segmentation approach on images. Image segmentation is briefly discussed by author (Efford. Nick, 2000) in the 10th chapter of "Digital Image Processing: A Practical Introduction Using Java." In this chapter, he wrote about the contextual and non-contextual Thresholding. He also mentioned that simple, adaptive and color Thresholding is a part of non-contextual Thresholding. Besides, contextual Thresholding focuses on pixel connectivity, regional similarity, regional growing and, split-and-merge segmentation. From these book reference one can determine its implementation approach to build a functional system. Using threshold based segmentation, this established work is completed that titled "A Survey on Threshold Based Segmentation Technique in Image Processing" (S. Jyothi &, K.Bhargavi. (2014)). Their paper focuses on reviewing and enumerating the comparison regarding performance of threshold technique as Histogram Dependent Technique (HDT) that is based on Global Threshold, Local Threshold and Adaptive Threshold.

1.2 Thesis Overview

The main goal of our research is to highlight the text from object and extract the text from that object (Drug box) in real-time video frames in purpose of giving voice assistance to a person. We consider the coordinates of that extracted text area field to determine actual distance of that

object in order to oblige the location. At first, a user will give voice command as request through the Kinect which will converted into synthesized text, the system try to identify medicine name from that voice command. Then, system will look though its database match with the converted text to check whether it is a valid medicine or not. Afterwards, the Kinect will start its camera to detect any text contained object. If it find any object with contains text, the Kinect will extract the text using OCR and emguCV library. Again, the system will check whether the extracted text from video frame identical with given medicine name. Moreover, Kinect also measure the distance of the text matched object in order to oblige the object's distance to that user. Finally, we will try to improve our OCR library to maximize the efficiency in terms of speed and interaction for betterment of the users.

1.3 Motivation

As a research group, we tried to build unique system based on some research to complete our undergraduate thesis. In modern age, we noticed that there are several innovation and services can be found which made modern daily life easier. For instance, the embedded home automation system, advanced image processing, virtual assistance, robotic elderly assistance and so many more. On the contrary, an estimated 253 million people live with vision impairment: 36 million are blind and 217 million have moderate to severe vision impairment. We believe that it's a major issue and it growing rapidly throughout the ages. This issue make us very concern about the future of the world and the way we overlook this matter, it should not in this way. The number of innovation regarding visually impaired people is quite less than general innovation for the society. Advance innovation which modernize people's life has more business impact in the tech world, whereas there is less initiative and resources to work for visually impaired people. But they are the part of the society and we should assist and take care of them. These obstacles made us inspired and enthusiastic to build a sophisticated reliable system which will able to guide a visually impaired person in a way that he/she can manage to locate any medicine surrounding around them. In medical science, eye specialist and doctors are working so hard to improve the eye sight of a person by providing them proper medication and optics (glasses). Whereas, as a computer science undergraduate student we thought if we can also contribute to provide a software which will give them a virtual assistance in order to make their life a little bit easier.

1.4 Outline

In chapter 2 we described the fundamental of image processing which is related to this topic. Our whole project is based on the image processing so that it is very important to understand the basic concept of the image processing. As long as we want to build something new and innovative the basic fundamental understanding is very important. There are several type of image processing and it has a wide area to work on. Our project is basically depend on the text extraction process and it has a huge impact on region of interest (ROI). That is why we emphasize the basic theory of Morphological image processing. This concepts are highly related to our topic. Chapter 3 describes the proposed method including system architecture, flowcharts and its components. To build the whole system we have used so many software and library. For different type of task we tried different type of approach. To implement this we first designed the system architecture. Inside the system architecture we described the hardware and software specification. Moreover, to complete the work properly and efficiently we designed a flow chart that described the whole task and guide us to achieve the ultimate result from the workflow. After that, the block diagram explain the full hardware setup and the purpose of the project. It give us the overall view of the required hardware and software setup to get the environment to work under. Therefore, comes the result and analysis that we are expecting from the experiment. In this chapter we describe different approach of our task and tool that have been used in the project. We include comparison and analysis of our work. Some screenshot of the task that explain the work and the most important part is the outcome of the project. We explain each part with example to clear the topic we are going to cover and explain it with necessary references. The last chapter of this report is the conclusion which describes limitations and future improvements of the research. No work cannot be done by one attempt but we tried our best to build a better system for the visually impaired. Though it has some limitation but it will a different approach to aid them to find their medicine.

Chapter 2

Fundamentals of Image Processing

Image can be describe as visual representation of two-dimensional image. Long before there was no digital representation of simple images but now they have. There are several types of images whom has different formats for usage. Image formats like PNG, JPGE, GIF, SVG, and TIFF are commonly used nowadays [18]. Each of these image formats has its own benefits and downsides. Vector graphics and raster graphics are two terms that describes an image. When an image stored in raster form is often called a bitmap. An image map is a file containing information that associates different locations on a specified image with hypertext links. JPEG is a graphic image file produced according to a standard from the Joint Photographic Experts Group, an ISO/IEC group of experts that develops and maintains standards for a suite of compression algorithms for computer image files. JPEGs is known for .jpg file extension [18].

2.1 Image enhancement techniques

An image refers to a 2D light intensity function f(x, y), where (x, y) denote spatial coordinates and the value of f at any point (x, y) is proportional to the brightness or gray levels of the image at that point. A digital image is an image f(x, y) that has been discretized both in spatial coordinates and brightness. The elements of such a digital array are called image elements or pixels. In simple word when we apply an operation on an image and changed its basic value to extract information or enhance it quality then that operation is called image processing. Digital image processing is the use of computer algorithms to perform image processing on digital images. Digital image processing allows the use of much more complex algorithms, and hence, can offer both more sophisticated performance at simple tasks, and the implementation of methods which would be impossible by analog means. To perform this algorithms on an image it need some steps. Fundamental steps in image processing are given below.

1. Image acquisition: The first stage of any vision system is the image acquisition stage. After the image has been obtained, various methods of processing can be applied to the image to perform

the many different vision tasks required today. However, if the image has not been acquired satisfactorily then the intended tasks may not be achievable.

2. Image preprocessing: to improve the image in ways that increase the chances for success of the other processes.

3. Image segmentation: Segmentation partitions an image into distinct regions containing each pixels with similar attributes that can be meaningful and useful for image analysis and interpretation, the regions should strongly relate to depicted objects or features of interest [12]. Segmentation techniques are either contextual or non-contextual. Thresholding is the simplest non-contextual segmentation technique [15]. With a single threshold, it transforms a greyscale or color image into a binary image considered as a binary region map [14]. A simple iterative adaptation of the threshold is based on successive refinement of the estimated peak positions [13]. It assumes that (*i*) each peak coincides with the mean grey level for all pixels that relate to that peak and (*ii*) the pixel probability decreases monotonically on the absolute difference between the pixel and peak values both for an object and background peak. The classification of the object and background pixels is done at each iteration *j* by using the threshold T_j found at previous iteration [12]. Thus, at iteration *j*, each grey level *f(x, y)* is assigned first to the object or background class (region) if $f(x, y) \leq T_j$ or $f(x, y) > T_j$, respectively [12]. Then, the new threshold, $T_{j+1} = 0.5(\mu_{j,ob} + \mu_{j,bg})$ where $\mu_{j,ob}$ and $\mu_{j,bg}$ denote the mean grey level at iteration *j* for the found object and background pixels, respectively:

Input: an image histogram $\mathbf{h} = \{h(q) : q = 0, \dots, 255\}$

Initialization: $j = 0;\ N = \sum_{q=0}^{255} \mathrm{h(q)};\quad T_0 = \frac{1}{N}\sum_{q=0}^{255} qh(q)$

while $T_{j+1} \neq T_j$ ***do***

$$\mu_{j.ob} = \frac{\sum_{q=0}^{T_j} qh(q)}{\sum_{q=0}^{T_j} h(q)};\ \mu_{j.bg} = \frac{\sum_{q=T_{j+1}}^{255} qh(q)}{\sum_{q=T_{j+1}}^{255} h(q)};\ T_{j+1} = \frac{\mu_{j.ob}+\mu_{j.bg}}{2}$$

end while

4. Image representation: to convert the input data to a form suitable for computer processing. One way to describe an image using numbers is to declare its contents using position and size of geometric forms and shapes like lines, curves, rectangles and circles; such images are called vector images. Another approach is bitmap image. Bitmap, or raster, images are "digital photographs", they are the most common form to represent natural images and other forms of graphics that are rich in detail. Bitmap images is how graphics is stored in the video memory of a computer. The term bitmap refers to how a given pattern of bits in a pixel maps to a specific color.

5. Image description: to extract features that result in some quantitative information of interest or features that are basic for differentiating one class of objects from another. Bitmap images take up a lot of memory, image compression reduces the amount of memory needed to store an image. For instance a 2.1 megapixel, 8bit RGB image (1600x1200) occupies 1600x1200x3 bytes = 5760000 bytes = 5.5 megabytes, this is the uncompressed size of the image.

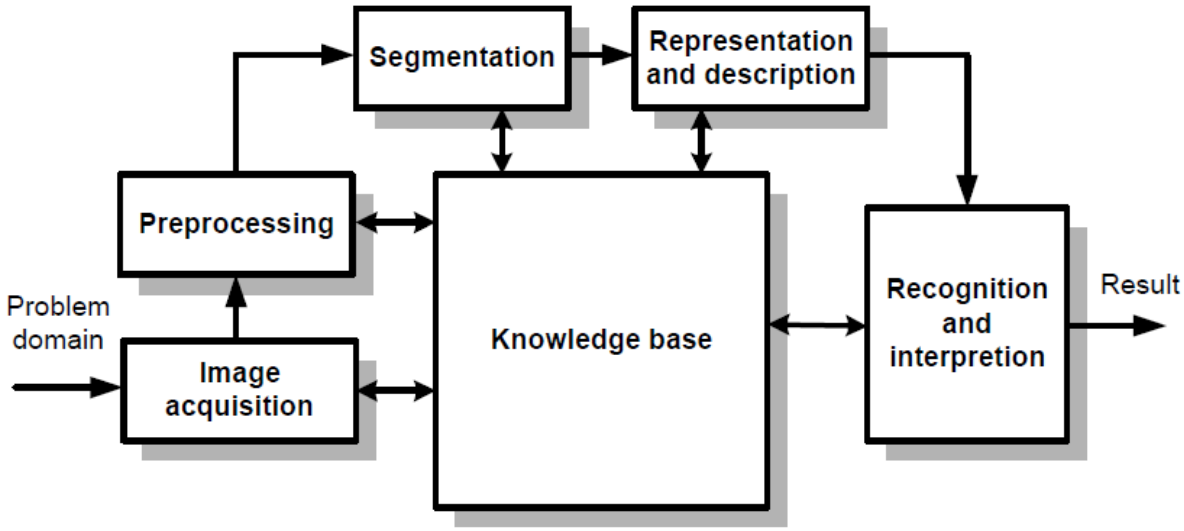

Fig 2.1: Fundamental steps in digital image processing

6. Image recognition: to assign a label to an object based on the information provided by its descriptors. Image recognition, in the context of machine vision, is the ability of software

to identify objects, places, people, writing and actions in images.

7. Image interpretation: to assign meaning to an ensemble of recognized objects. Image interpretation is defined as the extraction of qualitative and quantitative information in the form of a map, about the shape, location, structure, function, quality, condition, relationship of and between objects, etc. by using human knowledge or experience. Image reading is an elemental form of image interpretation. It corresponds to simple identification of objects using such elements as shape, size, pattern, tone, texture, color, shadow and other associated relationships. Image measurement is the extraction of physical quantities, such as length, location, height, density, temperature and so on. Image analysis is the understanding of the relationship between interpreted information and the actual status or phenomenon, and to evaluate the situation.

2.2 Morphological image processing method

Morphological image processing used to extract image components that are useful in the representation and description of region shape, such as, boundaries extraction, skeletons, convex hull, morphological filtering, thinning, pruning etc. In mathematical point of view it is tool for investigating geometric structure in binary and grayscale images. Visual perception requires transformation of images so as to make explicit particular shape information. Its main purpose is to distinguish meaningful shape information from irrelevant one. The vast majority of shape processing and analysis techniques are based on designing a shape operator which satisfies desirable properties. Morphological operation are usually two types. Erosions and dilations are the most elementary operators of mathematical morphology. More complicated morphological operators can be designed by means of combining erosions and dilations [16].

Dilation and erosion of an image f(x, y) by a structuring element b(x, y) [16].
NOTE: b and f are no longer sets, but functions of the coordinates x, y.

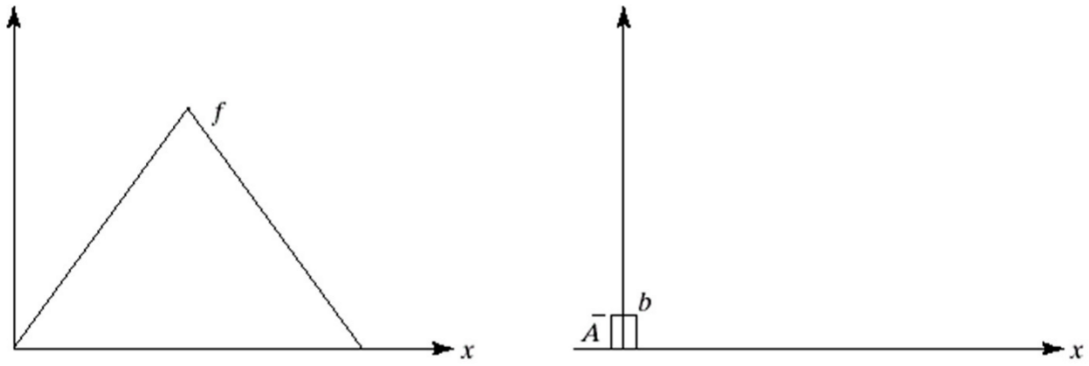

Fig 2.2: (a) A simple function [16]. (b) Structuring element of height A [16].

(f + b) (s) max { f (s - x) + b(x) | (s - x) element of D_f & x element of D_b }

Like in convolution, we can rather have b(x) slide over f(x):

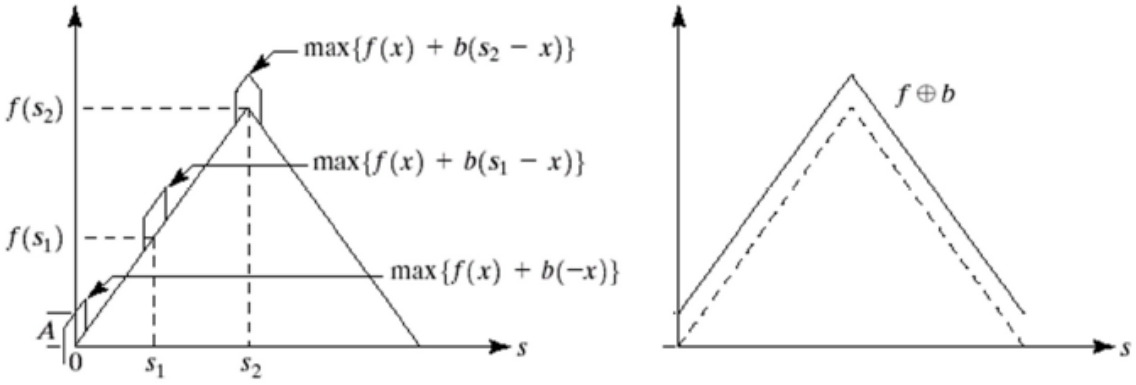

Fig 2.3 (a) Result of dilation for various positions of sliding b past f. (b) complete result of dilation (shown solid) [16].

Geometric interpretation: View the image as a 3-D surface map, and suppose we have a spherical. Opening is the roll the sphere against the underside of the surface, and take the highest points reached by any part of the sphere [16]. Closing is the roll the sphere on top of the surface, and take the lowest points reached by any part of the sphere [16].

Chapter 3

Proposed Method

To build a system, we have followed some method to complete our goal. To extract text form real time video we tried our best to perform a good accuracy. That is why we tried to solve the same problem is different ways. Some approach give us quite good result and some does not. The capability of OCR engines are different. Most common use of OCR are Aspose.OCR, Tesseract, Aforge and many other. We implemented our project applying those engines and Tesseract give us the best output. Though tesseract has some limitation but is fairly good enough to complete our task and this is the most advance text extraction tool available for out of cost. So, we emphasize this tool the most to extract the text. Moreover, it can work on the real time video frame which is an advantage because our system is running and perform in real time.

3.1 System Architecture

At first, this system takes input as voice command that has recognized and extracted by Microsoft speech recognition tool. Kinect has built in microphone hardware that allows the Microsoft speech recognition library to extract the voice commands into simple text form. Then, system will follow the text instruction as a command that has been extracted by Kinect before. These instructions usually sent request to the system to search a particular drug (medicine) in between Kinect visual area. The System observe and scan for the text contained object (medicine box) from real-time video buffer of Kinect Camera. To recognize the text area of the medicine, this system uses Tesseract OCR-engine as Optical Character Recognition (OCR) library and image processing tool named EmguCV as an OpenCV library that built for C# wrapper. Afterwards, system will try to follow the instructions by using RGB (Red, Green and Blue) color sensor and depth sensor that is already included in Kinect hardware. After successfully extracted the text from video frames using OCR toolkit, the system will analyze depth and RGB value using Kinect sensors. Using zone detection algorithm, Kinect sensor will provide the 3D coordinate values of (X, Y, Z) form that gives the exact location of the medicine box from the environment. At this point the system will put an end to its processing part and finalizing its outputs. As voice

instructions input by the user, The system will give voice output of the requested medicine along

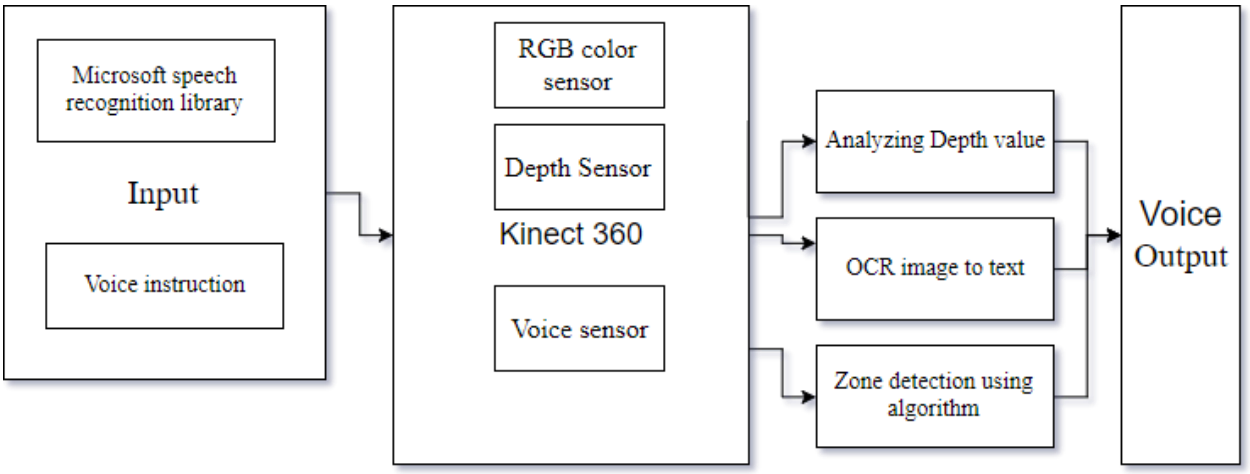

Fig 3.1 Block diagram for Proposed Method

With its location from environment area.

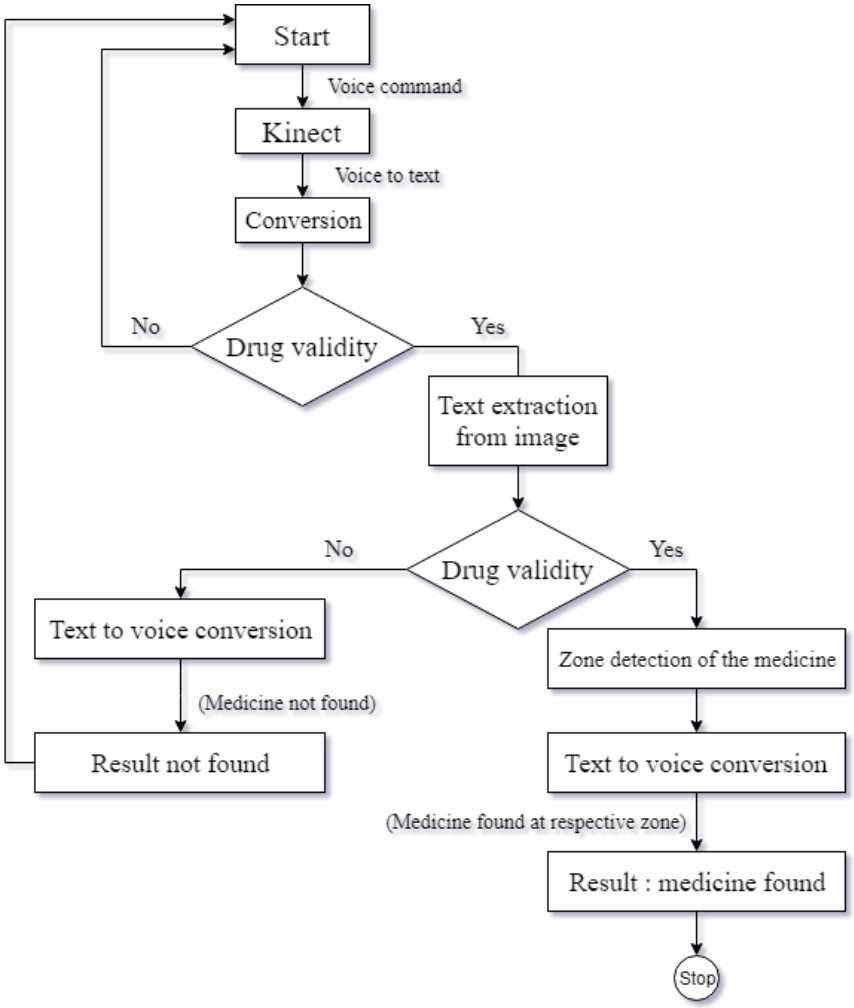

Fig 3.2 Flowchart of the Proposed Model

This is the flowchart of our system. It indicates the workflow our proposed system. First of all, the system starts with a voice command. It recognize the voice of the user and can take the necessary command to operate the program. The voice command has been captured by the Kinect microphone sensor and with the help of the Microsoft speech recognition tool it can convert the voice input to text. After that, when the system has the text input instruction then it starts searching for the desired drug which needs to find out. If the input command give a wrong name then the program wants for a valid drug name. It's a simple process to verify that the given name is actually a true drug name. Therefore, the Kinect cam turns on and scan the environment. According to the program the Kinect cam can change the threshold value of the video buffering and with the help of the region of interest it can localize the text of the environment and can work with that particular area. The other part of the video frame become black because we do not need any unwanted part that does not contain any text. Now, as Kinect locate the text area from the environment with the help of the OCR it converts the video frame which is eventually an image, it converts it to text. From that text if the given voice command has that particular drug name that needs to be find out then it gives in speaker that the desired drug has been found, otherwise it says that no drug has been found in that given voice command. Furthermore, the Kinect has many sensors and two of the mostly used sensors are depth sensor and RGB sensor. With these two sensors we can find out the 3D coordinate (x, y, and z) and color value of the object. That coordinate gives us the exact location of the drug box. Finally, when the drug is found then the speaker gives voice instruction for the visually impaired to track the drug from the environment. As the visually impaired cannot see anything so the voice instruction is the only way to guide him/her through the drug box. This is how a visually impaired can easily find the drug and the system is verifying that he/she is actually getting the right drug.

3.1.1 Tesseract OCR Engine

Tesseract is an open-source OCR engine that was developed at HP between 1984 and 1994. Like a super-nova, it appeared from nowhere for the 1995 UNLV Annual Test of OCR Accuracy [1], shone brightly with its results, and then vanished back under the same cloak of secrecy under which it had been developed. Tesseract began as a PhD research project [2] in HP Lab, Bristol, and gained momentum as a possible software and/or hardware add-on for HP's line of flatbed

scanners. Motivation was provided by the fact that the commercial OCR engines of the day were in their infancy, and failed miserably on anything but the best quality print. After a joint project between HP Labs Bristol, and HP's scanner division in Colorado, Tesseract had a significant lead in accuracy over the commercial engines, but did not become a product. The next stage of its development was back in HP Labs Bristol as an investigation of OCR for compression. Work concentrated more on improving rejection efficiency than on base-level accuracy. At the end of this project, at the end of 1994, development ceased entirely. The engine was sent to UNLV for the 1995 Annual Test of OCR Accuracy [1], where it proved its worth against the commercial engines of the time. In late 2005, HP released Tesseract for open source.

Word Recognition: Part of the recognition process for any character recognition engine is to identify how a word should be segmented into characters. The initial segmentation output from line finding is classified first. The rest of the word recognition step applies only to non-fixed-pitch text.

Joined Characters: Tesseract attempts to improve the result by chopping the blob with worst confidence from the character classifier. Candidate chop points are found from concave vertices of a polygonal approximation [2] of the outline, and may have either another concave vertex opposite, or a line segment. It may take up to 3pairs of chop points to successfully separate joined characters from the ASCII set.

Classification: Classification proceeds as a two-step process. In the first step, a class pruner creates a shortlist of character classes that the unknown might match. Each feature fetches, from a coarsely quantized 3-dimensional look-up table, a bit-vector of classes that it might match, and the bit-vectors are summed over all the features. The classes with the highest counts (after correcting for expected number of features) become the short-list for the next step. Each feature of the unknown looks up a bit vector of prototypes of the given class that it might match, and then the actual similarity between them is computed. Each prototype character class is represented by a logical sum-of-product expression with each term called a configuration, so the distance calculation process keeps a record of the total similarity evidence of each feature in each configuration, as well as of each prototype. The best combined distance, which is calculated from the summed feature and prototype evidences, is the best over all the stored configurations of the class.

Training Data: Since the classifier is able to recognize damaged characters easily, the classifier was not trained on damaged characters. In fact, the classifier was trained on a mere 20 samples of 94 characters from 8 fonts in a single size, but with 4 attributes (normal, bold, italic, bold italic), making a total of 60160 training samples. This is a significant contrast to other published classifiers, such as the Calera classifier with more than a million samples [3], and Baird's 100-font classifier [4] with 1175000 training samples.

Linguistic Analysis: Tesseract contains relatively little linguistic analysis. Whenever the word recognition module is considering a new segmentation, the linguistic module(misnamed the permute) chooses the best available word string in each of the following categories: Top frequent word, Top dictionary word, Top numeric word, Top UPPER case word, Top lower case word(with optional initial upper), Top classifier choice word. The final decision for a given segmentation is simply the word with the lowest total distance rating, where each of the above categories is multiplied by different constant. Words from different segmentations may have different numbers of characters in them. It is hard to compare these words directly, even where a classifier claims to be producing probabilities, which Tesseract does not. This problem is solved in Tesseract by generating two numbers for each character classification. The first, called the confidence, is minus the normalized distance from the prototype. This enables it to be a "confidence" in the sense that greater numbers are better, but still a distance, as, the farther from zero, the greater the distance. The second output, called the rating, multiplies the normalized distance from the prototype by the total outline length in the unknown character. Ratings for characters within a word can be summed meaningfully, since the total outline length for all characters within a word is always the same.

Adaptive Classifier: It has been suggested [5] and demonstrated [6] that OCR engines can benefit from the use of an adaptive classifier. Since the static classifier has to be good at generalizing to any kind of font, its ability to discriminate between different characters or between characters and non-characters is weakened. A more font-sensitive adaptive classifier that is trained by the output of the static classifier is therefore commonly [7] used to obtain greater discrimination within each document, where the number of fonts is limited. Tesseract does not employ a template

classifier, but uses the same features and classifier as the static classifier. The only significant difference between the static classifier and the adaptive classifier, apart from the training data, is that the adaptive classifier uses isotropic baseline/x-height normalization, whereas the static classifier normalizes characters by the centroid (first moments) for position and second moments for anisotropic size normalization. The baseline/x-height normalization makes it easier to distinguish upper and lower case characters as well as improving immunity to noise specks. The main benefit of character moment normalization is removal of font aspect ratio and some degree of font stroke width. It also makes recognition of sub and superscripts simpler, but requires an additional classifier feature to distinguish some upper and lowercase characters.

3.1.2 Aspose.OCR engine

Aspose is an OCR engine which is able to extract text from the image. It convert the image into bitmap and then segment its value to find out the region of interest in order to identify weather it contains any text or not. Aspose.OCR product family is a suite of character and optical mark recognition APIs (Application Program Interface) to add OCR and OMR functionality within all .NET and Java applications. Developers can dynamically create OMR templates directly using minimal code – open an existing one and save it using OMR template editor too. With Aspose.OCR, read characters from images and extract text from complete or part of an image. The images can also be loaded from a URL to perform OCR. It enables extracting barcode, text and OMR data from scanned images along with support of working with all advanced OCR and OMR elements. PDF files are widely used among developers as these are easy to create and manipulate with maximum security. PDF files are portable and support interactive functions that make it more popular among different file formats. Now, there might be scenarios where you need to extract text from PDF files, Aspose for Cloud allows you to extract text from Pdf files in no time. You can also extract text from images using a combination of features of two REST APIs i.e. Aspose for cloud and Aspose.OCR for Cloud. Consider a scenario where you need to extract text from an image in the PDF file. For this, first you need to extract image from PDF file using Aspose for Cloud REST API. Once you have extracted the image from PDF file, you can now extract the text from image using Aspose.OCR for Cloud REST API. Aspose.OCR for Cloud allows you to extract text from BMP and TIFF images. You can extract text and recognize characters from the documents in the cloud using Aspose.OCR for Cloud API.

3.2 System Implementation

To implement this system, we have built our own algorithms with in structural formation in order to follow the periodical steps for proper implementation. By following the algorithms, it become less difficult to process our development for this system. We wrote our algorithms in such a way that a normal reader can easily understand our execution steps one by one. It is a matter of fact that only for our particular system, the algorithms works smoothly without any types of difficulties. In addition, the algorithms we have provided in this report may or may not be efficient for implement for other types of image processing work field.

3.3 System Setup

3.3.1 Hardware Specification

CPU:

Name	AMD FX(tm)-8300
Cores	8
Clock speed (mhz)	3300
Typical TDP	95W
Socket	Socket AM3+
Microarchitecture	Piledriver
Platform	Volan
Processor core	Vishera
Core stepping	OR-C0
CPUID	600F20

Manufacturing process	0.032 micron
Data width	64 bit
Level 1 cache size ?	4 x 64 KB 2-way set associative shared instruction caches 8 x 16 KB 4-way set associative data caches
Level 2 cache size ?	4 x 2 MB 16-way set associative shared exclusive caches
Level 3 cache size	8 MB 64-way set associative shared cache

Table 3.1: CPU Specification

Memory:

Physical memory	16GB

Table 3.2: RAM

GPU:

GPU	NVIDIA GeForce GT 620

Table 3.3: GPU Specification

3.3.2 Software Specification

Name	**Type**	**Version**	**Architecture**
Visual Studio 2017	Microsoft Development	2017	64 bit
Kinect SDK(Software Development Kit)	Kinect Development	2012	64 bit
Tesseract OCR	Optical Character Recognition	3.2.0	64 bit

Aspose OCR	Optical Character Recognition	1.0.0	64 bit

Table 3.4: Software Requirements

OS:

Name	Microsoft Windows 10 Pro
Version	10.0.10586
Build Number	10586
System type	64 bit

Table 3.5: Operating System Details

Chapter 4

Experiment and Result Analysis

This chapter explains us the experimental process and the result analysis of the project. In this chapter we are going to explain all the outcomes of our project. At first, we applied all the operation on a single image. Our first approach was Aspose.OCR tool which is used for the text extraction form any image. It can extract text from any plain image but the challenge of that engine is that it cannot extract any text where there are other value present alongside the text. In a simple word that engine only extract text where there is nothing but text is presented. Moreover, it has the charter limitation. It can only extract text up to a limited number of charter. More the 25 charter cannot be determined by the Aspose.OCR. Though in a medicine box the name of the medicine is less than 25 charter but the medicine box contains some visual text and design. It sometimes has visual instruction of usage of that product and the company name and logo is also present there. So that it gave us some unwanted value that is not at all related to the medicine name. The avg. total charter error is 20.9 and the word precision error is 37.2. It's a high precision error rate and many cases we cannot even able to find the desire result which we are looking for. That is why we had to look for the second approach to solve that issue. For that purpose, we chose another OCR engine that is Google Tesseract. It is basically an open source library that has been developed and maintained by Google. As it is an open source project that is why we had the full access of that library. We change and modified that library according to our demand and changed the threshold value of the image and get almost close result compared to our expected result. According to accuracy and performance [number]. The avg. total charter error is 11.0 and the word precision error is 25.4 which is a good rate of accuracy.

4.1 Experiment Process

Proposed model is followed in order to proceed to the workflow and experimental process. This model guide us to implement the techniques of optical character recognition and text area location using depth and RGB color sensor.

In order to frames to text extraction, tesseract OCR engine takes frames for image acquisition. Then, OCR engine segments the input image for matching or recognition from the buffered frames taken from the captured environment. OCR engine return the recognized text that it process from the input image. In the meantime, this system run another process concurrently that marks the text extracted area and locate the object's (drug) location to guide the user with voice output form.

4.2 Algorithms

We have developed some algorithm based on our assumptions. The algorithms are shown and illustrated below:-

Data: Image
Result: Text
`OcrEngineProcess`(*sender*, *eventArgs*)
if *images inserted* **then**
    image ← Bitmap Conversion
    ocrProcess ← TesseractEngine(tessdata, Language)
    processedText ← ocrProcess(image)
    GetText ← processedText
end

Algorithm 1: Image to Text

In the beginning, we started our work by taking one test image file as input. We took the image file in order to convert that image into plaintext format. We followed the algorithm that is shown above figure (Algorithm 1) in purpose of convert image to text format. Same image file is processed into several OCR engines to get better accuracy and decision to pick one of the OCR engine that should be reliable for further implementation. In this algorithm, the method OcrEngineProcess (sender, eventArgs) takes an object named 'sender' and an event named 'eventArgs' where the sender object includes a reference that control an event of a program. That means whether an event will be handle an input or goes to idle when no input will be passed to this specific method. In second step, If condition will check whether image files in inserted or not. After inserting an image file, if condition gives true value and processed into next step. The image

file will be converted into bitmap at very next step of this algorithm. Then ocrProcess variable will be initialized by TesseractEngine (tessdata, Language) method that takes two input whom are 'tessdata' that is trained data of characters and 'Language' that specifies particular format of the text which will be extracted. After that ocrProcess variable takes the bitmap image and convert into text of given language format. Lastly, the processed text will be stored and shown.

After successful conversion of image to text, we build another algorithm that extracts text from real-time Kinect video. This algorithms is shown and illustrated below:-

Data: Webcam video
Result: Text
`OpenKinectVideo`(*sender, eventArgs*)
initialization KinectVideoCam
CaptureVideo ← frames to Bimap Conversion
`DetectFrames`(*sender, eventArgs*)
ExtractText(*BGRFrame*) ← BgrConversion
if *frame is not empty* **then**
    frames ← Bimap Conversion
    Fixed buffer at 30FPS
end
`ExtractText`(*BGRframe*)
if *Bgr Frame is not empty* **then**
    Edge Detection using Sobel method
    Dilation for buffered frames
    Find Contours for buffered frames
    Get Processed Text
end

Algorithm 2: Realtime Text Detection

In this algorithm, it takes input data as buffered frames from Kinect Video capture in real-time. And as an output it gives plain text by extracting text from medicine box in real-time video frames. Initially, the method OpenKinectVideo (sender, eventArgs) takes an object named 'sender' and an event named 'eventArgs' where the sender object includes a reference that control an event of a program. That means whether an event will be handle an input or goes to idle when no input will

be passed to this particular method. At the beginning of this method, Kinect Video camera setting will be initialized for the first time of this algorithm. Then, this method will be end by converting buffered frames into Bitmap and store them in CaptureVideo variable that eventually means capturing bitmap converted video frames. After this step, there will be another method would be ready for execution. This next method named DetectFrames (sender, eventArgs) also initialized like previous method whether its execution steps are not same. At the first step of this method, there will be another nested method called ExtractText (BGRframe) will be executed inside this method. The nested method ExtractText (BGRframe) takes a BGR frame as an input for execution. This method starts with If Condition that check whether a BGR frame is passed in this condition or not. When a frame of BGR format is pushed into this condition, it gives true value and processed into next step for execution. Then, the algorithm will use EmguCv library property and use Sobel method for edge detection. In following step, the method process dilation into buffered video frames. The algorithm will find contours from buffered frames in this step. At last step of this method, from all above process finally processed text contain frame will be extracted and return into previous method for further execution. By returning extracted text include frame from ExtractText method, DetectFrames method proceed into a condition that checks whether the frame contains value or not. If the frame contains value then condition will be true and process to next step that is Bitmap conversion of value contained frames. At the final step of this method, the extracted text included frames will be shown and maintained in thirty FPS (Frame per Second) buffer rate for visual perception. From this algorithm, we can able to extract text from real-time buffered video frames that is part of our proposed implementation workflow. After extracting text from Kinect Video frames, the system will use Microsoft speech recognition library in order to provide voice output from extracted text. From voice output, the system will be able to guide the user by giving information.

4.3 Result Analysis

For text extraction we have used two OCR engine with different performance. For betterment of our system we used one of them.

PERFORMANCE ANALYSIS: OPTICAL CHARACTER RECOGNITION ENGINE

Optical Character Recognition technique shows us some more points of view. Say, if data contains millions of components or attributes, then the system will take more time to compute the desired model. But the fact is, the more data is provided, the more accurate result we get.

OCR Engine	Test Language	Text extraction		Total Character Errors	Word Precision Errors
		Plain Text	RGB Colored Text		
Google Tesseract	English	Yes	Yes	11.0	25.4
Aspose OCR	English	Yes	No	20.9	37.2

Table 4.1: Comparison between Google Tesseract OCR and Aspose OCR engines [17].

From above the character and precision errors, it is noticeable that Tesseract OCR engine has less errors than Aspose OCR engine. Aspose also fails to extract text from colored image where Tesseract OCR engine can perform extraction.

For better realization image results of these two separate OCR engines are shown below:

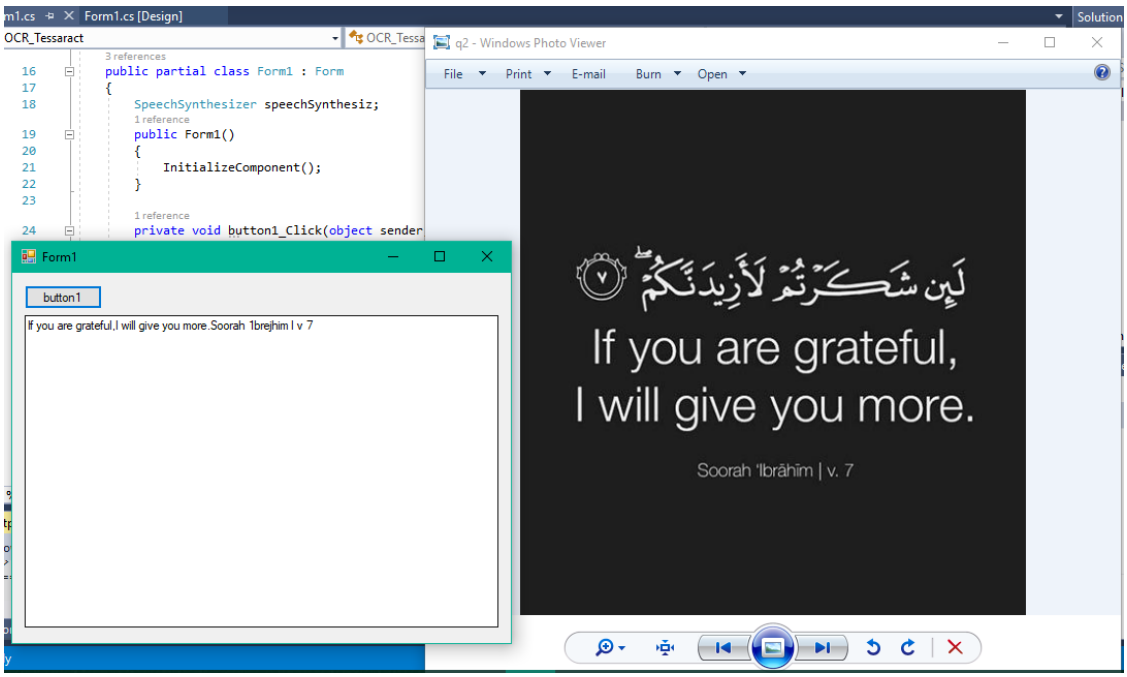

Fig 4.1: Plain-Text Result of Google Tesseract OCR Engine

This result example gives us the extracted text value from the image. The black background image is a simple image which contains some text in two different language. As we can see in the output panel the English text part has been successfully extracted which processed by OCR engine using English language's trained data and the Arabic text part has been ignored because Arabic trained data was not included in the system. That is because while building this system we made it in a way so that it can extract English word from the image. However, other language trained data will included for future extension to our proposed work. For now, we assumed that all the text will be English on the medicine box. As there is other language rather than English so that we get some unwanted value which has been shown in the output panel.

Fig 4.2: Colored-Text Result of Google Tesseract OCR Engine

This is another example of extracting text value from the image. This is a simple diary notebook with black background. The text in the diary is in silver color. In upper of the text has a logo of the company. In the output panel we can see that here also the system extracted the text successfully and the unwanted logo part has been cut off from the extraction as it was not recognized by OCR engine as characters. It gives us the accurate result that we are looking for.

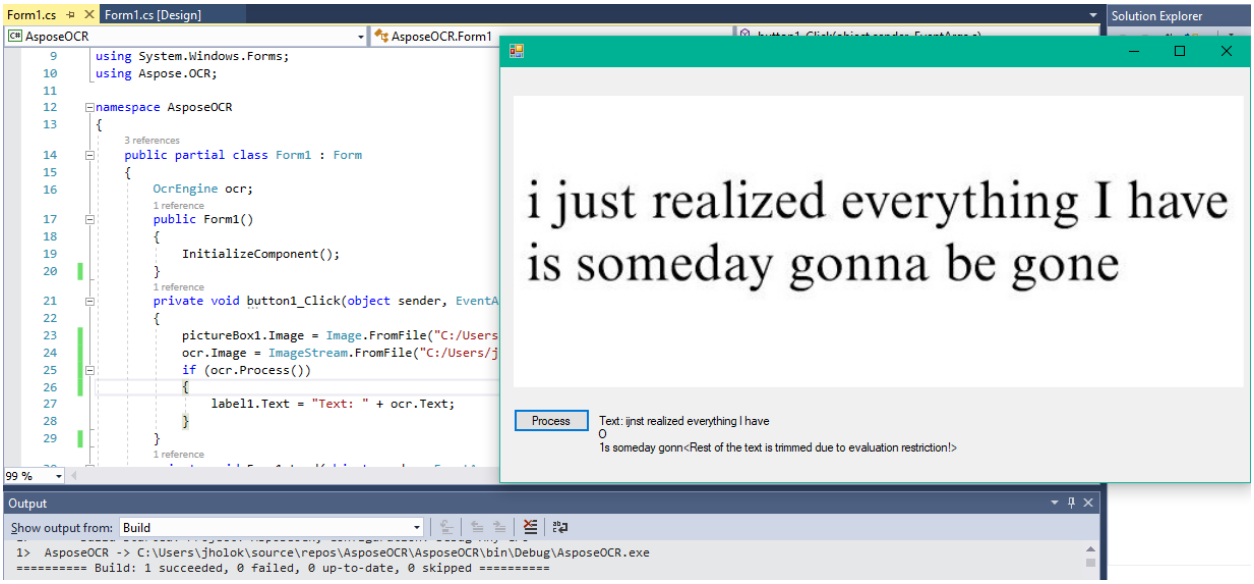

Fig 4.3: Plain-Text Result of Aspose OCR Engine

This is an example of Aspose OCR. This program has been developed by using Aspose.OCR engine. It only works perfectly when there is plain text and no background image or design. It do not even give any veiled result for handwritten text. So, when we applied this system for the medicine box we could not able to select and region of interest and extract text from any handwritten text. It extract text faster but has some serious limitation. Our task need something different and upgraded engine to solve our issue so we rejected this approach.

Concluding this, the system uses Tesseract OCR to perform better text extraction from buffered frames of the camera. It give us better performance and efficient result to extract the text. We emphasized our motive to this engine and build a better system to extract text from image as well as video frames.

BUFFERED FRAME ANALYSIS:

Using Google Tesseract OCR engine as Optical Character Recognition tool, the system extracts the drug (medicine name) from the buffered frames from Kinect Camera.

Here are some samples images of medicine recognition and extraction below:

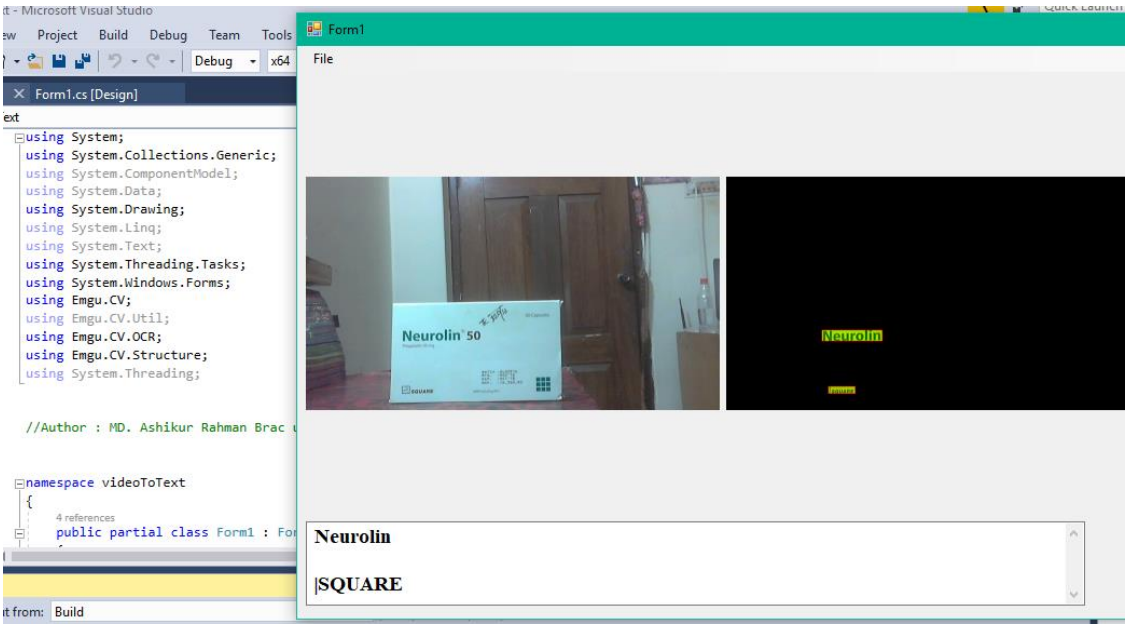

Fig 4.4: Result of Google Tesseract OCR Engine with Image Enhancement

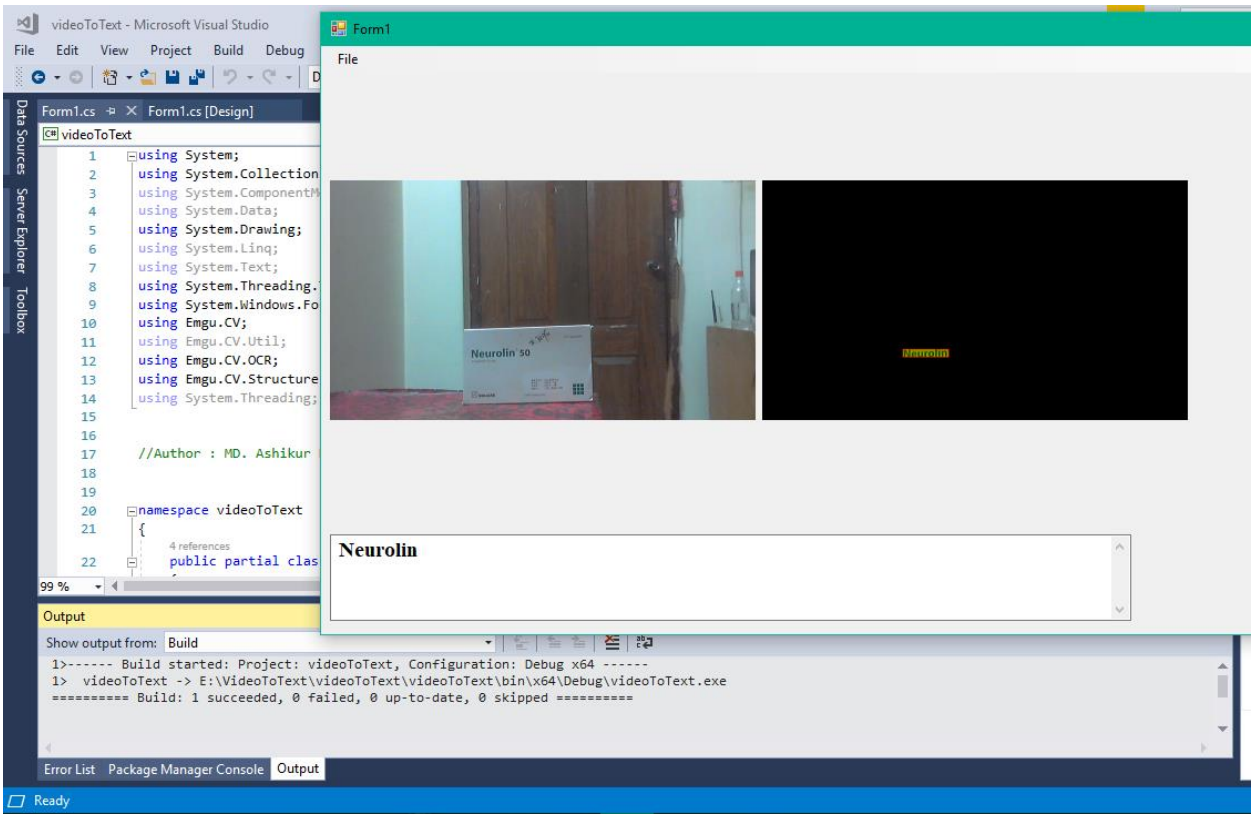

Fig 4.5: Result of Google Tesseract OCR Engine without Image Enhancement

In this example we use Google Tesseract OCR with Image enhancement. Image enhancement is a part of the image processing. First we convert the image into bitmap and then change the threshold value using the adaptive-threshold property of EmguCV that modify the threshold value that make changes into image frame. This approach enhances the image quality and ultimately it gives the better text extraction result when OCR engine is applied to extract text of that enhanced frame.

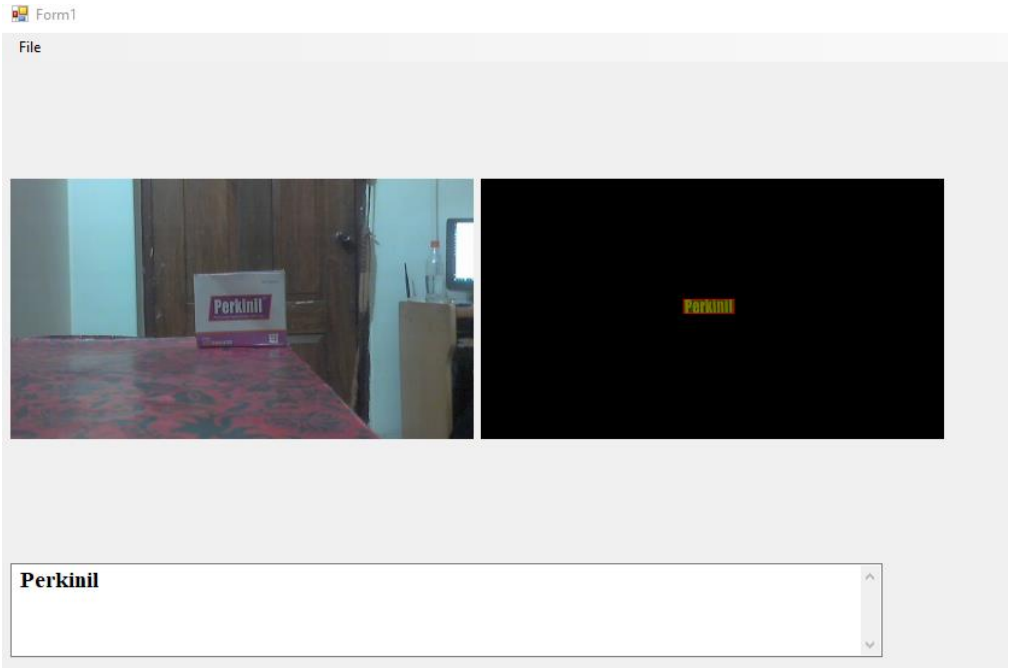

Fig 4.6: Result of Google Tesseract OCR Engine without Image Enhancement

As we can see the difference between the enhanced and non-enhanced image in the picture. This example of image does not give us a bright extracted output because the image has not been enhanced. The default image has been provided into it for processing and after extraction process the system is giving us extracted text from extremely low quality image. Often, the system failed to apply OCR Engine in order extract text from unenhanced image or frames. Thus, the emguCV provide the enhanced tool to process the image and that solves this technical hazard as we have seen in the previous example.

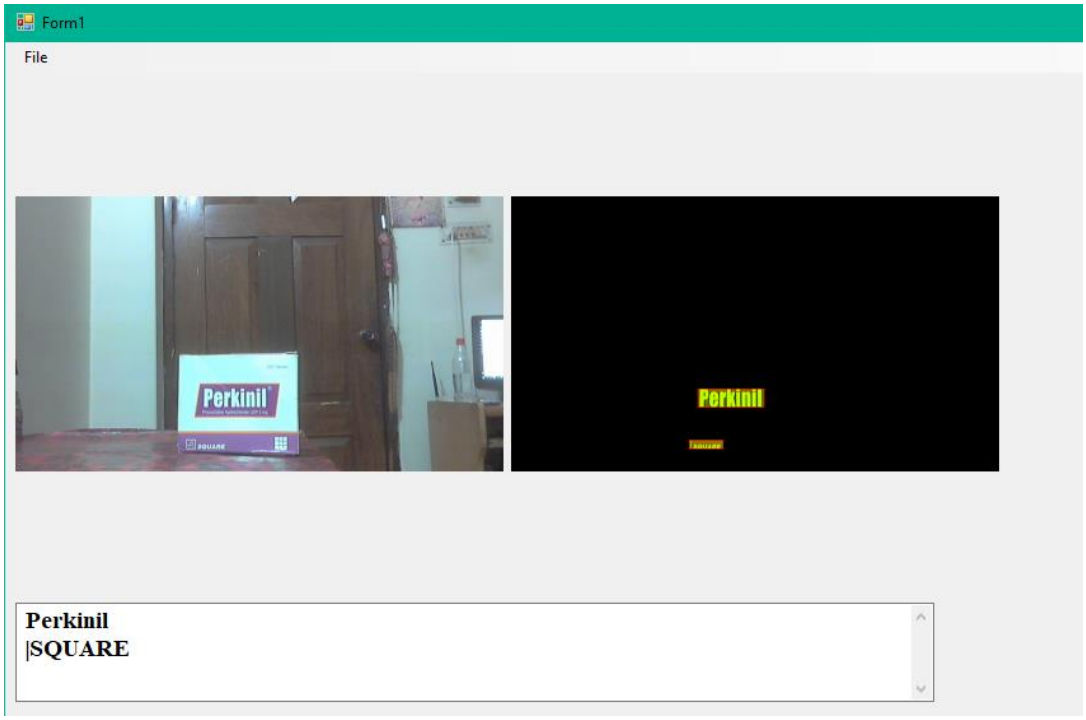

Fig 4.7: Result of Google Tesseract OCR Engine with Image Enhancement

From this result image example, the system shows that how it triggered RGB color and depth sensor in purpose of locate coordinate location from extracted text area of an object. In this example, we took a medicine box as an object in order to get its static location from the environment of Kinect Video Camera. It clearly shows that how the system has successfully locate 3D coordinate (X, Y, Z) location from this particular medicine box and storing result into output box that can provide to user along with voice output in order help user to reach to medicine box.

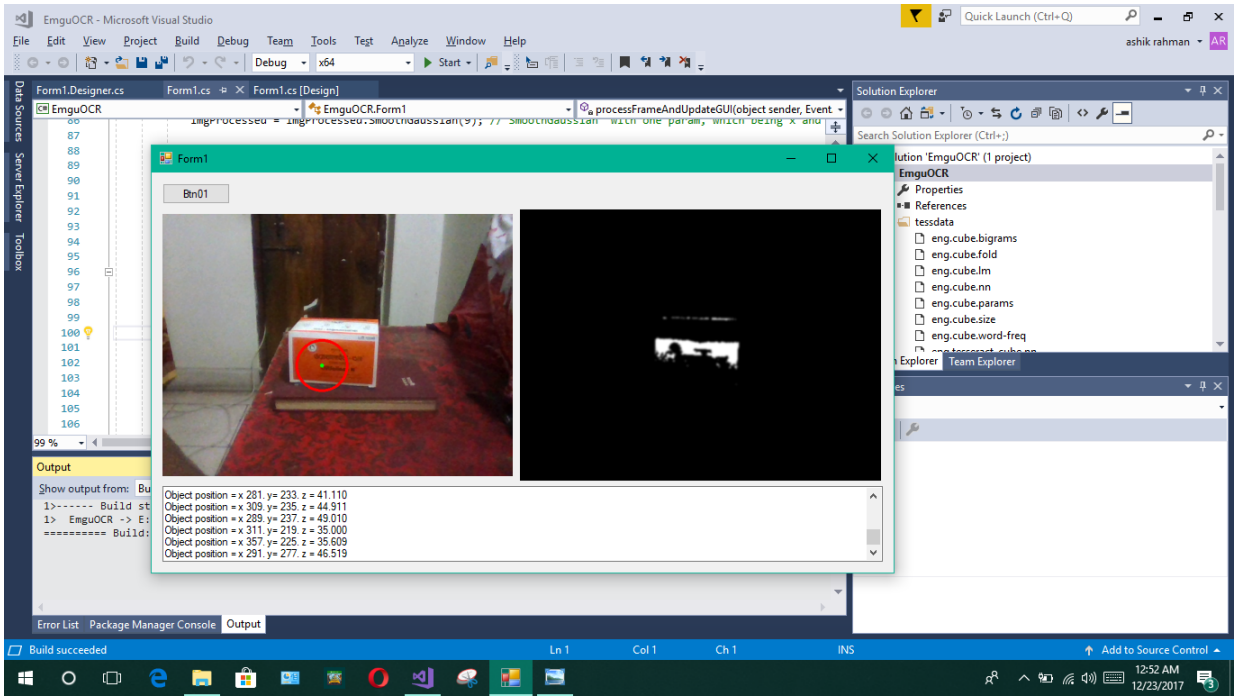

Fig 4.8: Result of Medicine location using Kinect RGB color & Depth sensor

To summarize the result example, the system enables the Kinect sensor of RGB color and depth to locate the current 3D coordinate location of the drug (medicine) based on extracted text area and stores to give output to the user.

4.4 Future Plan

For any research, there is always room for improvement. Ours is not an exception of that. While we have marked the end of our research for the time being, we have also pointed out some areas where this research can be stretched:

1. **Other Optical Character Recognition tool:** Other optical character recognition model like AForge OCR, ABBYY OCR, Iron-OCR are also popular tools which can be used for better text extraction result which improve the accuracy of the extracted text from images.

2. **Better Training Improves Extraction Accuracy**: Whenever we built manually our own trained data into the OCR engine it shows better result than the default trained data of

English language. So it is a matter of fact that in order to extract text from images is better to use train characters as train-data.

3. **Add features on Kinect:** So far we have tried to detect an object (Medicine box) by using RGB and depth sensor of Kinect. In future we would like to work on hand recognition of the user that ensures the finding of medicine properly.

Extend the System into a Mobile Application: In far future, when all other smartphone will have OCR library, RGB and depth sensor included as mobile hardware then we will develop the same application that can be operated and get the full functionality of our built system.

4.5 Difficulties

Real-time video frames to text extraction and recognition is our core part of the work. In order to apply this idea, we used tesseract and Aspose OCR to process acquisition and segment the image into text is a complex process. We have used the provided engine of OCR which is trained by the library itself. For this approach we found that OCR engine often gives partial or inaccurate result. To solve this, we increase the adaptive threshold value of the frames and got satisfied output. Another barrier we faced regarding implement Kinect sensors which will give us RGB and depth value to detect objects current location. It is difficult to guide visually impaired person to give proper information by providing only medicine box's 3d coordinate value in voice output. The proposed system that we talked about need to apply artificial intelligence to make the system more intelligent and concern about users location as well as medicine's location, in order to verify that user has successfully reached to medicine box and found it with the guided information provided by the system. Another difficulties we faced regarding implement our work that Kinect SDK version often mismatches with the latest windows 10 operating system. Thus we had to work on downgraded OS (Operating System) in order to implement this method.

Chapter 5

Conclusion

We have been working in this research about a year ago. We believe we proposed a unique solution of a complex model that still have many areas to improve. We faced many obstacles in this research, and most of them successfully overcome. A few limitations forced us to adapt to them and reconstruct our models. This proposed system has offered a promising solution for the visual impaired people to find their medicine on by themselves. It does not even need any manual instructions or a third party software to operate the system. We believe that if we are able to make this proposed model into the form of hardware implementation then it will more compatible to the visually impaired communities. We will try to make this production reasonable so that a lot of people can able to afford it from online stores and get aided by this product. If our proposed system get sponsored by an advanced technological companies like Microsoft, Google, Mozilla etc. then the research will amplify and the level of implementation will improve. Our main goal was to help the visually impaired people and this project give us an opportunity to make something good for them. To make an improvement for the system, we will use convolutional neural network to train our language data of OCR engine so that the improvement of medicine recognition from real time video will be efficient. Before the implementation of our system, we thought about this small dream that someday this system going to be used by a lot of people and that will make us more enthusiastic to work on many research based work. In future, we will try to make it more accurate and compatible within embedded mobile devices so that the portability can be ensured.

References

[1] S.V. Rice, F.R. Jenkins, T.A. Nartker, *The Fourth Annual Test of OCR Accuracy, Technical Report 95-03*, Information Science Research Institute, University of Nevada, Las Vegas, July 1995.

[2] R.W. Smith, *The Extraction and Recognition of Text from Multimedia Document Images*, PhD Thesis, University of Bristol, November 1987.

[3] M. Bokser, "Omnidocument Technologies", *Proc. IEEE* 80(7), IEEE, USA, Jul 1992, pp. 1066-1078.

[4] H.S. Baird, R. Fossey, "A 100-Font Classifier", *Proc. of the 1st Int. Conf. on Document Analysis and Recognition*, IEEE, 1991, pp 332-340.

[5] G. Nagy, "At the frontiers of OCR", *Proc. IEEE* 80(7), IEEE, USA, Jul 1992, pp 1093-1100.

[6] G. Nagy, Y. Xu, "Automatic Prototype Extraction for Adaptive OCR", *Proc. of the 4th Int. Conf. on Document Analysis and Recognition,* IEEE, Aug 1997, pp 278-282.

[7] I. Marosi, "Industrial OCR approaches: architecture, algorithms and adaptation techniques", *Document Recognition and Retrieval XIV*, SPIE Jan 2007, 6500-01.

[8] R. Jiang, Q. Lin, and S. Qu, “Let Blind People See: Real-Time Visual Recognition with Results Converted to 3D Audio”, IEEE, 2016

[9] S. K. Singla, R.K.Yadav, “Optical character recognition based speech synthesis system using lab view”. *Journal of Applied Research and Technology*, vol. 12(5), 919–926, 2014.

[10] V. Filipe, F. Fernandesb, H. Fernandesc, A. Sousad, H. Paredese, And J. Barroso, “Blind navigation support system based on Microsoft Kinect”, *Procedia Computer Science*, vol. 14, pp. 94-101, 2012.

[11] J. Opeza, A. Olveraa, J. ireza, F. Butandaa, M. Manzanoa, D. Ojedab, “Detecting objects using color and depth segmentation with Kinect sensor” *Procedia Technology*, vol. 3, pp. 196-204, 2012.

[12] Efford. Nick, “Chapter 10 "Segmentation".” Digital Image Processing: A Practical Introduction Using Java, Pearson Education, 2000.

[13] B.Sindhu, J.B.Jeeva, “Automated Retinal Vessel Segmentation Using Morphological Operation and Threshold” *International Journal of Scientific & Engineering Research*, vol. 4, Issue 5, May 2013, ISSN 2229-5518.

[14] Shirly, Adin and C. N. Savithri. “Detection of Ulcer in the Gastrointestinal Tract Using the Wireless Capsule Endoscopy Images.” *International Conference on Engineering Innovations and Solutions*, (2016).

[15] S. Jyothi &, K.Bhargavi. (2014). A Survey on Threshold Based Segmentation Technique in Image Processing. *International Journal of Innovative Research & Development*, vol. 3, Issue 12, November 2014, ISSN 2278-0211.

[16] Gonzalez, Rafael C., and Richard E. Woods. “Chapter 9.” Digital image processing, Pearson, 2002.

[17] J. barlow, “Tesseract-Ocr/Tesseract.” GitHub, 8 Jan. 2017, github.com/tesseract-ocr/tesseract/wiki/4.0-Accuracy-and-Performance.

[18] Rouse, Margaret. “What is image? - Definition from WhatIs.Com.” *WhatIs.com*, Mar. 2016, whatis.techtarget.com/definition/image.

List of Abbreviations

OCR – Optical Character Recognition

SDK – Software Development Kit

RGB – Red, Green and Blue

OS – Operating System

ROI – Region of interest

FPS – Frame per Second

API – Application Program Interface