Information Awareness for the Visually-impaired using Machine-Vision

Hariprasad KC, Vidhya K, Raghul Sundar S, Karan P, Perumalsamy M

Velammal Engineering College Chennai, Tamilnadu, India.

Received 28 January 2017; Accepted 22 March 2017; Available online 28 April 2017

ABSTRACT

There has been numerous developments for the visually-impaired in the past decade, be it obstacle detectors, GPS based navigators, etc. But there has been no prominent technology since Braille that aided the inference of Raw data. Thus we are proposing an Image Process based application that will process both Text and Faces. The next challenge lies in transforming this information in a way that a challenged individual can infer. For which we have chosen a Text-to-Speech API that can speak out to the individual. The image processing is done using OpenCV, embracing the concepts of Optical Character Recognition (OCR) and Facial Recognition. The Text-to-Speech module we suggest is the Java’s J2TTS or Python’s pyttsx. Once bundled with all these features, our application will serve as a really efficient and interactive aid.

KEYWORDS: OCR, Face Recognition, Text-to-Speech, OpenCV, Pyttsx.

INTRODUCTION

Given the constraints placed, on both inference of data from the outside world and the accurate depiction of data, in the forms of clear image-acquisition, accurate extraction of features and the scalability of the Text-to-Speech API (Application Program Interface), We have chosen the suitable technologies with lots of research such that, This particular concept can be extended as an application in all platforms and can also be associated into other open-source projects.

For Image Acquisition certain hardware features are essential like Onboard Memory, Fast Data Transfer to PC and Pre-processing functions. The camera’s auto-focusing sensors are the main engine behind achieving accurate focus, and are laid out in various array across an image’s field of view. ISO is the level of sensitivity of a camera to the available light and the ability to remove noise from the images. So optimally, base ISO can fetch us high quality images at ideal daylight. The Logitech C310 HD webcam does a fantastic job at shooting 720p images at 5 megapixels with features like automatic ISO, Focus and Aperture adjustments.

The Quick capture function appeals to the vanity in all of us and allows most of the preprocessing to be done at the hardware end itself.

In case of image acquisition from the screen itself, there are not as much concerns as for Real-time scenes. But the problem exists when we have to fetch the text omitting the design contents applied to the text like HTML and CSS. This greatly falters the OCR step and makes it misbehave. The HTML and CSS formatting are generally done at the user’s end, i.e. by the browser. Hence by bundling this project as a browser plugin, we can obtain the raw text embedded within the HTML and CSS thus increasing the quality of the recognized text.

Optical Character Recognition (OCR) has been one of the most widely used Image Processing techniques of the past decade. We experience it in Scanners, PDF-to-TEXT converters and even in commercial applications
like Google Docs. OCR simply converts images containing printed or handwritten text into raw text which can be freely manipulated. This field has been under a lot of research with close relation to pattern recognition algorithms to not just convert text but also to detect and separate text from wide scenarios.

Face Recognition is another popular sub-domain of Image Processing that has been under a lot of research. Considering all the popular face-detection algorithms, We have chosen the Haar-Cascading technique owing to its advantage that its relatively 20% more accurate than the rest. Though it may not scale really well in embedded environments as it works with float values, Modern embedded systems coupled with DSP processor can easily overcome this issue.

Text-to-Speech APIs (Application Programming Interface), as used in Artificially Intelligent assistants such as the one being developed by Facebook and the fictional JARVIS, is a speech synthesizer that constructs speech from language text. A lot of open-source APIs are available for the common user and having tested a few, We have gone with FTTS (Free Text-to-Speech) for JAVA Applets and Pyttsx for python-based web applications. These APIs offer a lot more than just speaking plain text. Their speech engines can be configured to analyze numerics, punctuations and expressions. Thus, The assistant we propose will relatively close to that of an average human companion.

**Methodology:**

A. **Tesseract-OCR:**

The Tesseract OCR is a Character Recognition Engine that is currently supported by Google. It has its support extended over all three major platforms of Windows, Linux and MacOs. The Tesseract module was primarily written in C++ but can be extended to other programming languages such as Python with the wrappers defined for it by open source developers. Pytesseract is the OCR wrapper for python. Currently Tesseract supports over 100 languages including Tamil, Hindi, Marathi, Hindi and more.

The main challenge underlying performing OCR on realistic scenes is when we have to extract text embedded in objects. This can be done Image Binarization and Enhancement technique.

1) **Pre-processing and Enhancement:**

In case of images with poor quality/resolution, we can resort to the Wiener filter to provide considerable amount of enhancement. The drawback is that the images have to be grayscale for enhancement because adjusting the luminance for colored images can prove to be tricky and hence inaccurate at times.

2) **Estimation of foreground regions:**

The next step involves the separation of the foreground containing text from the background. For this we suggest using Sauvola’s Adaptive Thresholding method. This technique extracts a binary version of the image from the original image where the 1’s or the whites map directly to the foreground regions. This when compared again with the original grayscale image or the inverted grayscale image gives us the more refined foreground estimation.

3) **Text-areas Detection:**

From the binary image, we decide on the area(s) containing text by connected component analysis procedure where bounding boxes are defined for each character recognized. If the neighbouring characters are of the same height or satisfy a certain threshold, they are included into the detected region. Thus by grouping connected components in the horizontal direction, the text area is well-defined. After this the actual OCR takes place.

B. **Haar-Cascaded Face Recognition:**

Haar Cascade is an object detection algorithm that utilizes a series of positive and negative images to train a classifier that can detect specific regions of an images. OpenCV’s Cascade Classifier contains a pre-trained face classifier which has high levels of accuracy as it has been trained with over 160000 faces.

Face detection is relatively easy when compared to face recognition owing to OpenCV libraries. In Order to recognize a face, we need to have facial database. A facial database can be anything - a CSV file, an XML file or even an EXCEL sheet. Since XML files allow attributes to be defined as trees, while fetching matching features, Recognition becomes way faster with XML-type-feature persistence.

So how do we decide what features to extract? how many features do we extract ? There are around1 6000 features that can be extracted from a 24x24 human face.

Calculating the Sum of black and white pixels for such an image would be costly in terms of both computation and time. Hence the concept of integral images is used which implements a Summed-Area-Table.
Text-area separation process:
As the name suggests, the value at any point \((x, y)\) in the summed area table is just the sum of all the pixels above and to the left of \((x, y)\), inclusive

\[
I_{\Sigma}(x, y) = \sum_{x' \leq x} \sum_{y' \leq y} i(x', y')
\]

C. Haar-Cascade feature extraction:
The next important step is feature extraction. We simply cannot decide on selecting from 16000 features and seldom do we know which features define a face better. This section is achieved by AdaBoost which stands for Adaptive Boosting. AdaBoost is a machine learning optimization algorithm that boosts a classifier's performance by training itself with decision trees.
Coming to the recognition section, Whenever a face is to be recognized, the same procedure for the training set is followed and instead of persisting the data in the XML file, the data is instead compared with the XML entries to find the nearest match. The degree of prediction is given by the term Confidence, which is a cumulation of the differences in predicted and the actual image. Very high confidences stand for a lesser match and a confidence of zero stands for the same image. Thus, a threshold condition can be set depending upon the value of confidence to decide on whether the prediction was correct or not.

We ran this face detection algorithm over the Yale face database which contained over 180 images of 15 individuals and came to the conclusion that the highest confidence for a correct match was exactly 99.6. Adding a 20% difference owing to illumination and noise, We have arrived at a conclusion that an ideal confidence value to set as threshold would be, 99.6 + (20% of 99.6) ~ 120.

With this value of confidence as threshold and having tested with a database of our own consisting of 25 images of 5 individuals taken at completely different levels of exposure and environment, 23 images were correctly identified to be predicted or wrong. Thus we have achieved an accuracy of 92%.

**D. Text-to-Speech API:**

A Text-to-Speech API is a speech synthesizer that can convert normal language text into speech. The process of synthesized speech involves construction using pieces of recorded speech stored in a database. The biggest challenge with speech synthesis is Text Normalization. Texts are full of Abbreviations, Numbers and Heteronyms. The ability of a speech engine to recognize these patterns define its efficiency.

We have tested a few engines and settled with Free TTS engine for Java based applications and Pyttsx for python based applications. These engines offer a wide range of features such as different voices, changing voice rates, understanding the numeric system and punctuations.

Thus, by bundling this TTS engine with the image processing module, we arrive at the conclusion of our proposal.

**Conclusions:**

The proposed work aims at combining the robust image processing technologies of OCR and Face Recognition in conjunction with Speech synthesizer to aid the visually impaired to comprehend both digital and real-time information around them.

Rather than just proposing this idea, we took the initiative to actually develop this application. The resultant software lived up to our expectations. However, if this concept has to be easily accessed by a layman, there’s still a lot of work to be done.

As the next step, we are trying to turn this little piece of software into a plugin or an add-on that will be compatible with internet browsers. This way, Our concept can be accessed without any hassle. Also, we would like to add-on that the inferences of our research, the datasets and the ground truth will be made accessible to all researchers.

**REFERENCES**

5. Haar Features for FACS AU Recognition, Jacob Whitehill, Christian W. Omlin (FGR’06).