Texas Instruments Innovation Challenge
India Design Contest 2015

PAGE 1 - SELF ASSESSMENT

1. "We, authors of the report entitled **Reading Assistance For Blind/Low-Vision People** confirm that this report has not been submitted to any other forum such as another contest or conference for publication. We hereby state that we will not submit the same work for any other contest in the future. We understand that Texas Instruments has the right to use this report in its conferences/publications. We will seek TI's permission before we submit the report for publication in an external forum."

2. Our idea is not completely original. Previous attemps at making reading assists for blind people have been made. We mainly derived inspiration from the products that use character recognition to read aloud the pages of a book and also the fact that the Braille system of reading is very cumbersome and expensive. A reference is MIT Media Lab's "FingerReader"

3. Professor Goutam Das, our mentor helped us during any obstacle that we faced in this whole process. Mr. Rajesh Kedia, a former TI employee, helped us get familiar with the dev boards. The seniors who have participated in the contest in the previous years also aided a lot and guided us. Lastly, the online blogs on TI parts electronics, software, open source etc. also proved to be of great help.

4. We faced a problem with the webcam used for this project to take the snapshots of the pages in the book as the beaglebone black didn't have the required amount of video memory available. After taking one or two photos using the webcam the usb controller being out of memory was failing to initialize the device to take another snap. So, we used our laptop instead to handle the webcam and the beaglebone copies the image from our laptop using an ssh connection.

   We faced another problem with the image taken by the webcam. The OCR program (Tesseract) was facing problem reading the image file as in most of the cases it was tilted and and text near the edges of the pages were blurred. 2D prespective correction using OpenCV was employed to make the camera image fit for OCR.

5. There is no proper place to work during late hours as all the labs are closed after 5'o clock and the electronics hobby hub which is the only place open during all the hours didn't had many facilities like oscilloscope with the functionality of FFT. Also this laboratory only had 4 keys available which made it even more difficult for all the teams to work whenever the time was appropriate.

6. No, we didn't use WEBENCH to design power supply or filter our project.

7. TI India Analog Design Contest helped us learn about using the beaglebone environment for integrating various software aspects of project such as OCR,text-to-speech conversion, communication between PC, beaglebone and launch pad.It gave us opportunity to make a real-time embedded-cum-software project.

8. **Movable Camera Stand**: The current camera stand is fixed, thus limiting the usage of camera to a single side of the book only. A movable stand will make the product more usable and efficient.
   Dictionary Support of Tesseract: The library 'Tesseract' used for OCR just identifies the characters. Better prediction would be possible if the predicted word is matched for a possible word in the dictionary. This will further increase the efficiency of the OCR
   Output of Sound Card: The output of the speakers are low beause of the quality of the sound generated is poor. A better sound quality output will make the hearing process easier. Even a volume control module can be used for great user interface

9. Please tick all aspects of your project that you believe are now complete.

Paper design of hardware ✓	Algorithm/software design ✓
Hardware implementation on breadboard ✓	System-levell testing with examples □
Hardware implementation on PCB □	Benchmarking/Performance Analysis □
Hardware Testing ✓	Short Video on Project ✓ Youtube video link : http://youtu.be/QZlMIZKv6Ws

Ajay Agarwal

Aranya Goswami

Aranya Goswami

Bishal Santra

Bishal Santra

Soumya Sanyal

Soumya Sanyal

Vardaan Pahuja

Prof. Goutam Das

Reading Assistance for Blind/Low-Vision People

Ajay Agarwal, Aranya Goswami, Bishal Santra, Soumya Sanyal, Vardaan Pahuja
Prof. Goutam Das
Indian Institute of Technology, Kharagpur
West Midnapore, West Bengal
ajayagarwal1995@gmail.com

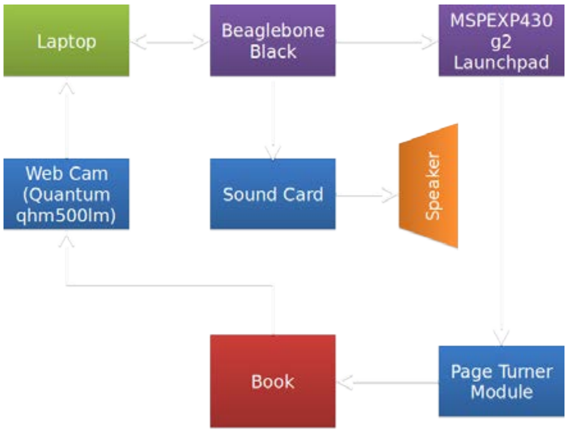
Abstract—

The project addresses the basic problem of reading faced by blind and low-vision people. Our prototype is basically a text-to-speech converter. It takes the image of a page using a camera and this image is processed and read out aloud. The prototype also includes a mechanism for the turning of the pages of the book. People with retarded vision either have to resort to hearing or using Braille characters in order to read anything; listening being the obvious convenient option. The product thus caters to the basic recreational and developmental need of reading of visually impaired people who constitute a substantial section of the society.

I. INTRODUCTION

A. Technical Background

The aim of project is to do Optical Character Recognition (OCR), feed the text to a text to speech converter and play the audio through a DAC (Digital-to-Analog) converter.
The salient features of a similar product are available at :
https://www.enhancedvision.com/downloads/brochure/DaVinci-Flyer.pdf
Our solution is not as a full-fledged commercial product due to limited resource availability but accomplishes the basic task as efficiently as the state-of-art with added advantage of being inexpensive.



II. PROPOSED SOLUTION

The proposed solution is to integrate a webcam with a microcontroller which can do Optical Character Recognition of the input image from the webcam. For this the webcam is used to capture an image of the page to be read out. This image is then fed to a laptop using USB port for image processing using OpenCV library. In image processing the edges are properly detected, grayscale version of the image is formed and 3-D perspective mapping of the image is done to make the image appear perfectly flat. This image is then give to Beaglebone Black for Optical Character Recognition using Tesseract. The output of the OCR is a text file containing the words of the decoded image. This is fed to a text to speech converter (festival) which generates a .wav file corresponding to the given text input. This .wav file is played using a sound card through a speaker. Then once the sound is completely played, the the beagle bone sends an interrupt call to the MSP430 launchpad which is interfaced to control the motors of the page turning mechanism. First a motor rotates a wheel for very short duration which raises the page a bit. Then another motor rotates a stick in circular motion to turn the page. In this way the next page is available for decoding. this process is continued.

III IMPLEMENTATION

A. Hardware Implementation

The hardware implementation includes page turner mechanism which includes a small wheel whose rim lifts the page and a rotating shaft which turns the page over.The motors used are simple DC motors controlled with L293 driver.When OCR and audio output of text generated is complete, an interrupt is sent by beaglebone to Tiva MSP430 launch pad which signals the motor driver to rotate the motors for a specified duration to accomplish the specified task.

B. Software Implementation

Pseudo code for OCR pre-processing:

1. Convert image to grayscale.
2. Binarise a copy of image and perform contour detection to find contour of largest area. This corresponds to the page of book.
3. Perform OR operation of inverted binary image with grayscale image to eliminate background of page.
4. Perform Harris corner detection to find corners in each quadrant. These four points are corners of page.
5. Do perspective correction using the 4 points obtained earlier and map them to a rectangle of size of original image.
6. The output of perspective correction is fed to Tesseract for OCR.
7. The text file generated by the tesseract is fed to the Festival for text-to-wav file conversion.

IV. RESULTS

The image was properly taken and processed using OpenCV. Tesseract OCR gave fairly satisfactory results. The festival software also gave proper sound outputs. The integration of laptop and sound card with Beaglebone Black was also successfully achieved. The output sound was fairly good and audible. The page turner worked as expected. Overall, the outcomes were desirable with scope of further changes at some places.

V. CONCLUSIONS

Our aim was to make a fully automated prototype for reading books using character recognition. Due to physical memory limitations of the beaglebone, we had to use the laptop for our assistance which limited the automation of our prototype. We were successful in reading aloud the pages and implementing the page turning mechanism. The OCR engine gets better with time as it self-trains itself. The future scope of this project is very widespread. The whole project can be developed into an ASIC which would be of very low cost when manufactured on a large scale.

ACKNOWLEDGMENTS

We sincerely thank the following people for their support without which the project would not have been compelte:
Professor Goutam Das, Department of Electronics and Electrical Communication Engineering, IIT Kharagpur
Mr. Rajesh Kedia, former TI employee
The seniors of our institute who previously participated in TI ADC

REFERENCES

[1] A.N. Author. Title of the paper. Journal of Innovation. Vol 3, No. 2. 2011.

[2] M.Y. Favorite Author. Title of the paper. Available from Internet. URL. 2013.

[3] A.N. Other Author et al. Title of their paper. Conference Title. 2012

APPENDIX B : CODES

```
/*Code for Motot Control

int pwmpin = 4;
int pwmpin1 = 9;
volatile int state = LOW;
void setup()
{
  pinMode(GREEN_LED, OUTPUT);
  pinMode(pwmpin, OUTPUT);
  pinMode(pwmpin1, OUTPUT);
  digitalWrite(GREEN_LED, HIGH);
  /* Enable internal pullup.
   * Without the pin will float and the example will not work */
  pinMode(PUSH2, INPUT_PULLUP);
  attachInterrupt(PUSH2, blink, FALLING); // Interrupt is fired whenever beagle gives a pulse

void loop()
{
  digitalWrite(GREEN_LED, HIGH); //LED starts ON
  if(state == HIGH)
  {
    analogWrite(pwmpin, 200);     //start the top motor
    analogWrite(pwmpin1, 0);
    delay(400);
    analogWrite(pwmpin1, 255);   //start the bottom motor
    analogWrite(pwmpin, 0);
    delay(4000);                          //calibrated delays
    state = LOW;
  }
  if(state == LOW)
  {
    analogWrite(pwmpin, 0);        // stop both motors
    analogWrite(pwmpin1, 0);
  }
}
void blink()
{
    state = HIGH;
}
/* CODE FOR AUTOMATION OF THE WHOLE PROCESS*/

#include <stdio.h>
#include <stdlib.h>

#define NUM_COMMANDS 8
typedef char* string;
int main()
{
//string fileName[2] = {"send1.jpg","send2.jpg"};
string paths[3] = {"/sys/class/gpio/","gpio60","/root/"};
string commands[] = {
"sshpass -p '123456' scp bishal@192.168.7.1:~/Desktop/send1.jpg /root/",
"tesseract send1.jpg out",
"text2wave out.txt -o play.wav",
  //"display send1.jpg",
"ffmpeg -i play.wav -f alsa 'default:CARD=Device' -re -vol 200",
"echo out > direction",
“echo 0 > value",
"echo 1 > value",
"echo 0 > value"};
int i,j;
for(i=0;i<4;i++)
{
       system(commands[i]);
}
chdir(paths[0]);
system("echo 60 > export");
chdir(paths[1]);

for(i=4;i<NUM_COMMANDS;i++)
{
system(commands[i]);
}
chdir("/root");
```

```
 commands[0] = "sshpass -p '123456' scp bishal@192.168.7.1:~/Desktop/send2.jpg /root/";
 commands[1] = "tesseract send2.jpg out";

for(i=0;i<4;i++)
  {
        system(commands[i]);
  }
chdir(paths[0]);
  system("echo 60 > export");
  chdir(paths[1]);

  //for(j=0;j<10;j++)
  {
        for(i=4;i<NUM_COMMANDS;i++)
        {
              system(commands[i]);
        }
        //sleep(1);
  }
  chdir("/root");
return 0;
}
```

OpenCV OCR Preprocessing code:

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/opencv.hpp"
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <vector>
#include <stdio.h>
#include<math.h>

using namespace cv;
using namespace std;

int main()
{
                                        Mat img,gray_img,img_bin,kernel;
                                        img=imread("p1.jpg");
                                        resize(img,img,Size(img.cols/4,img.rows/4));
                                        //resize(img,img,Size(0,0),0.85,0.85);
                                        cvtColor(img,gray_img,CV_BGR2GRAY);
                                        threshold(gray_img, img_bin,150.0, 255.0, THRESH_BINARY_INV|THRESH_OTSU);
                                        kernel = getStructuringElement(MORPH_ELLIPSE, Size(1, 1));
                                          bitwise_not(img_bin,img_bin);

                                        vector<vector<Point> > contours,contours2;
```

```
vector<Vec4i> hierarchy,hierarchy2;
vector<int> small_blobs,small_blob2;
double contour_area;
Mat temp_image;
double threshold=img.rows*img.cols/2;

// find all contours in the binary image
img_bin.copyTo(temp_image);
findContours(temp_image, contours, hierarchy,
CV_RETR_CCOMP,CV_CHAIN_APPROX_SIMPLE);

// Find indices of contours whose area is less than `threshold`
if ( !contours.empty()) {
  for (size_t i=0; i<contours.size(); ++i) {
      contour_area = contourArea(contours[i]) ;
      if ( contour_area > threshold)
                  small_blobs.push_back(i);
  }
}

// fill-in all small contours with zeros
for (size_t i=0; i < small_blobs.size(); ++i) {
  drawContours(temp_image, contours, small_blobs[i], cv::Scalar(255),CV_FILLED, 8);
}
Mat new_img=Mat(img.rows,img.cols,CV_8U,Scalar(0,0,0));
Mat out_img=Mat(img.rows,img.cols,CV_8U,Scalar(0,0,0));

bitwise_not(temp_image,temp_image);

for(int i=0;i<img.rows;i++)
{
  for(int j=0;j<img.cols;j++)
  {
      new_img.at<uchar>(i,j)=temp_image.at<uchar>(i,j)+gray_img.at<uchar>(i,j);
  }
}
//bitwise_not(new_img,new_img);


Mat dst, dst_norm, dst_norm_scaled;
dst = Mat::zeros(img_bin.size(), CV_32FC1 );
int blockSize = 7;
int apertureSize = 5;
double k = 0.04;
int thresh=100;
/// Detecting corners
medianBlur (temp_image,temp_image,3);
```

```
  cornerHarris(temp_image, dst, blockSize, apertureSize, k, BORDER_DEFAULT );

  /// Normalizing
  normalize( dst, dst_norm, 0, 255, NORM_MINMAX, CV_32FC1, Mat() );
  convertScaleAbs( dst_norm, dst_norm_scaled );

  /// Drawing a circle around corners

Point p1,p2,p3,p4;


//if(dst_norm.at<float>(0,0)>dst_norm.at<float>(0,1) && )
    p1=Point(0,0);
 for( int j = 0; j < dst_norm.rows/2 ; j++ )
    {
                                        for( int i = 0; i < dst_norm.cols/2; i++ )
        {
                                        if(dst_norm.at<float>(j,i) > dst_norm.at<float>(p1.y,p1.x))
                                          p1=Point(i,j);
        }
     }

p2=Point(dst_norm.cols/2,0);
for( int j = 0; j < dst_norm.rows/2 ; j++ )
    {
                                        for( int i = dst_norm.cols/2 ; i < dst_norm.cols; i++ )
        {
                                        if(dst_norm.at<float>(j,i) > dst_norm.at<float>(p2.y,p2.x))
                                          p2=Point(i,j);
        }
     }

p3=Point(0,dst_norm.rows/2);
 for( int j = dst_norm.rows/2; j < dst_norm.rows ; j++ )
    {
                                        for( int i = 0; i < dst_norm.cols/2; i++ )
        {
                                        if(dst_norm.at<float>(j,i) > dst_norm.at<float>(p3.y,p3.x))
                                          p3=Point(i,j);
        }
     }

p4=Point(dst_norm.cols/2,dst_norm.rows/2);
 for( int j = dst_norm.rows/2; j < dst_norm.rows ; j++ )
    {
                                        for( int i =  dst_norm.cols/2; i < dst_norm.cols; i++ )
        {
                                        if(dst_norm.at<float>(j,i) > dst_norm.at<float>(p4.y,p4.x))
```

```
                                    p4=Point(i,j);
        }
      }
cout<<"p1= "<<p1.x<<" "<<p1.y<<endl;
cout<<"p2= "<<p2.x<<" "<<p2.y<<endl;
cout<<"p3= "<<p3.x<<" "<<p3.y<<endl;
cout<<"p4= "<<p4.x<<" "<<p4.y<<endl;

circle( dst_norm_scaled, p1, 15,  Scalar(255), 2, 8, 0 );
circle( dst_norm_scaled, p2, 15,  Scalar(255), 2, 8, 0 );
circle( dst_norm_scaled, p3, 15,  Scalar(255), 2, 8, 0 );
circle( dst_norm_scaled, p4, 15,  Scalar(255), 2, 8, 0 );
                                   //medianBlur (new_img,new_img,1);
                                   /*
                                   Mat p3=imread("p3.jpg");
                                   Mat p4=Mat(p3.rows,p3.cols,CV_8UC3);

                                   Mat mmat = Mat(3,3,CV_32FC1);
  //Point2f c1[4] = (Point2f(80,200),Point2f(552,80),Point2f(200,756),Point2f(552,756));
  //Point2f c2[4] = (Point2f(0,0),Point2f(img.cols,0),Point2f(0,img.rows),Point2f(img.cols,img.rows));

  Point2f c1[4] = (Point2f(43,18),Point2f(280,40),Point2f(19,223),Point2f(304,200));
  Point2f c2[4] = (Point2f(0,0),Point2f(320,0),Point2f(0,240),Point2f(320,240));

  mmat = getPerspectiveTransform(c1, c2);
  warpPerspective(p3, p4, mmat,Size(p3.cols,p3.rows));
*/     // Input Quadilateral or Image plane coordinates
  Point2f inputQuad[4];
  // Output Quadilateral or World plane coordinates
  Point2f outputQuad[4];

  Mat lambda = Mat::zeros(new_img.rows, new_img.cols, new_img.type() );
/*
  inputQuad[0] = Point2f( 200,80 );
  inputQuad[1] = Point2f( new_img.cols-60,80);
  inputQuad[2] = Point2f( new_img.cols-60,new_img.rows-60);
  inputQuad[3] = Point2f( 200,new_img.rows-60  );
  // The 4 points where the mapping is to be done , from top-left in clockwise order
  outputQuad[0] = Point2f( 0,0 );
  outputQuad[1] = Point2f( new_img.cols-1,0);
  outputQuad[2] = Point2f( new_img.cols-1,new_img.rows-1);
  outputQuad[3] = Point2f( 0,new_img.rows-1  );
*/
  cout<<"a= "<<a<<"b= "<<b<<"c= "<<c<<"d= "<<d<<endl;
/*
                                   inputQuad[0] = Point2f(b,a);
  inputQuad[1] = Point2f(c,a);
  inputQuad[2] = Point2f(c,d);
```

```
    inputQuad[3] = Point2f(b,d);
*/
    inputQuad[0] = Point2f(p1.x,p1.y);
    inputQuad[1] = Point2f(p2.x,p2.y);
    inputQuad[2] = Point2f(p4.x,p4.y);
    inputQuad[3] = Point2f(p3.x,p3.y);
    // The 4 points where the mapping is to be done , from top-left in clockwise order
    outputQuad[0] = Point2f( 0,0 );
    outputQuad[1] = Point2f( new_img.cols-1,0);
    outputQuad[2] = Point2f( new_img.cols-1,new_img.rows-1);
    outputQuad[3] = Point2f( 0,new_img.rows-1  );
    // Get the Perspective Transform Matrix i.e. lambda
    lambda = getPerspectiveTransform( inputQuad, outputQuad );
    // Apply the Perspective Transform just found to the src image
    warpPerspective(new_img,out_img,lambda,out_img.size() );

    cout<<"rows= "<<img.rows<<" cols= "<<img.cols<<endl;

    cvNamedWindow("win3");
                                                imshow("win3",dst_norm_scaled);

                                                cvNamedWindow("win4");
                                                imshow("win4",temp_image);

                                                cvNamedWindow("win1");
                                                imshow("win1",out_img);
                                                imwrite("abc.jpg",out_img);

                                                cvNamedWindow("win2");
                                                imshow("win2",new_img);

                                                cvWaitKey(0);

                                                return 0;
}
```

APPENDIX C – BILL OF MATERIALS

	Component	Manufacturer	Cost per component	Quantity	Total cost of component	TI Supplied/ Purchased
1	**Beaglebone Black**	**Element 14**	**45$**	**1**	**45$**	**TI Supplied**
2	**MSP-EXP430G2**	**TI**	**9.99$**	**1**	**9.99$**	**TI Supplied**
3	**Quantum QHM-500LM Webcam**	**Quantum**	**8$**	**1**	**8$**	**Purchased**
4	**USB Audio Adapter**		**2$**	**1**	**2$**	**Purchased**
5	**Motors**		**2$**	**2**	**4$**	**Purchased**
6	**Speakers**	**Zebronics**	**6$**	**1**	**6$**	**Purchased**
7	**Miscellaneous (Wires, battery, breadboard etc.)**		**5$**		**5$**	**Purchased**
	Total Cost of the Project (approx.)				**80$**	