Indoor localization System for Assisting Visually Impaired People

Md. Mazidul Alam[1], Sayed Erfan Arefin[2], Miraj Al Alim[1], Samiul Islam Adib[1], Md. Abdur Rahman[1]
Department of EEE, American International University-Bangladesh[1]
Department of CSE, BRAC University[2]
{Mazidul.alam14,erfanjordison,mirajalalim,samiul.adib}@gmail.com, arahman@aiub.edu

***Abstract*— In this paper we developed an image processing based indoor localization system with color detection technique of connected objects. By using color detection technique our system can pinpoint a user's location with very high accuracy for real-time applications. We used our system to filter out an image for a specific color and extracted pixel co-ordinates for that image. User's location is then determined by comparing the matrix for those values against a pre-created matrix of training images. We successfully conducted experiments in indoor environments as well and they yielded very good results. After analyzing these results, we propose to integrate our localization system with indoor navigation systems which can be used for blind people where accuracy is the most important element. To further facilitate the navigation process we also developed an android based application.**

Keywords—Image Processing, Indoor localization, Wireless communication, Color Segmentation, Connected object detection

I. INTRODUCTION

According to the WHO (world health organization), there are nearly 39 million visually impaired people worldwide [1]and 12.05 live in the south east Asian region [2].There are approximately 675,000 visually impaired adults in Bangladesh [3]. A visually impaired person traditionally uses a white cane, a trained dog or other people's help to navigate around. So they need a very reliable way to navigate around.

There are many devices and apps which are very efficient in guiding people outdoors with GPS. Nowadays they come with voice commands which can be used by blind people also, paired with a smart walking stick equipped with ultrasonic sensors and puddle detectors. But, these technologies are only available for outdoor situations.

Many promising research have been done on the field of indoor navigation but most of them focused on navigating a robot and others focused on creating guided tours of commercial spaces where a certain amount of error can be tolerated for the sake of commercial feasibility. They used technologies like RFID, Infrared light, Ultrasonic sound, Audible sound, cameras and most prominently Wi-Fi. But as we can see from the table 1 webcam based systems can provide most accuracy but aren't widely used for its relatively high cost.

Table 1: Comparison of existing technologies [4,5]

Technology	Positioning Accuracy	Equipment price
Using Wi-Fi	2-100m	High
Using Ultrasonic sound	1-10cm	High
Using Infrared light	5-10m	High
Using RFID	5cm-5m	High
Using sound	5-10m	Middle
Using camera	1cm-1m	High

Though there has been some camera based indoor navigation systems like Aoki's Real-time personal positioning system [6] or Tilch's CLIPS [7] etc. they have their limitations. For these examples, Aoki's system has a big user burden and large processing time whereas CLIPS has accuracy in cm range as well as laser used in the system can cause skin and eye problems.

So, we can see an accurate indoor localization system is very much needed which is specifically developed to aid blind people navigate inside an indoor environment. To do this we need to take care of some things like:

1. Very accurate positioning system
2. Low user burden
3. Fast processing time for real time operation
4. User friendly system

Hence, we chose a camera based system which caters to our needs. With many promises, our proposed systems have some limitations too. As, we are using cameras and there can't be any blind spots from the camera coverage we need a substantial number of cameras which makes our system a bit more expensive than other systems in the market. The proposed system also offer a high accuracy rate which should be worth the extra cost

II. Proposed System

A. System Overview

Our system consists of a server computer where necessary processing is done, cameras, a smartphone based app to connect the user to the system. Number of cameras depends on the area that needs to be covered and their quality.

The images captured from the cameras are fed to a server computer which processes the images for required information. For the image processing we took advantage of the vast library of functions available in MATLAB. Flow charts of our system are given in Fig. 1 and Fig. 2.

B. Reference grid

To set up a reference grid after setting up the cameras our system takes one picture from each of them called "Source Image". That source image is then divided into a grid of square cells. Size of the square is selected because the square should cover a size of an average person. Next, a matrix is created of the same size as the reference grid.

C. Color allocation

After turning on the app, our server allocates a color to the device. Sometimes light from sources like Sunlight, tube lights or energy bulbs can shine on our device screen and if the light is strong enough, it can fade our allocated color to the device which can make it unusable. In that case a new color is allocated to the device.

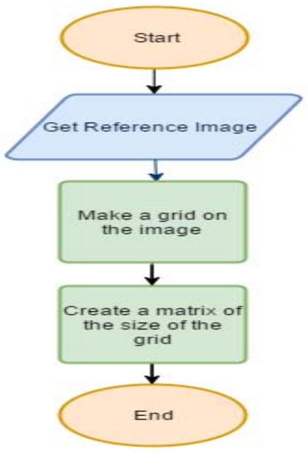


Fig. 1. Flowchart for making reference grid

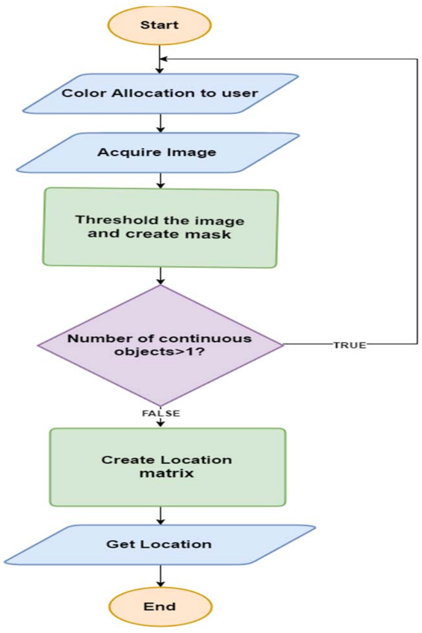


Fig. 2. Flowchart of the proposed system

Every device is assigned with a unique color, so that we do not miss-identify two different devices as one. The server process for color allocation is give below:

1. When a post request is received which requests for a new color, the database is checked for assigned colors and tries to determine a unique color which is not yet assigned.
2. Then the color is send to the device by generating a json which contains this color as "color" via the reply to the request.
3. The color is saved in the "assigned colors" table so that it can be later used to get unique colors for other devices.

D. Image Acquiring process

First, we capture images from the overhead camera which we preinstalled. Images are taken with an interval of 2 seconds. Then, Images are converted to desired resolution for ease of calculation as well as to avoid the output showing problem with MATLAB which happens when image size is bigger than screen's resolution. Working with lower resolution pictures also makes sure that cheap cameras are enough to run the system which ensures a low cost system.

E. Color Processing

"Color processing" is the core part of our system. The pixel values of the image are stored in a variable to start processing the image. The pixel values allow us to analyze the image for different colors. We look for pixel values which corresponds to our desired color. Then, those pixel values are used to create a thresholding[8] parameter which thresholds the image for that color only. Afterwards, this thresholded image is used to create a color mask. To avoid unwanted objects we filter the mask for pixels which have much lesser size than our user end device.

Since, the mask can contain other elements, MATLAB function “bwlabel” [9] is used which ensures that the created mask only contains connected objects like our device. “bwlabel” labels connected components and returns a matrix of the same size of the image.

Now that the mask is ready, we apply it on the image to keep only the user end device on the image with the help of MATLAB function “bsxfun” [10]. Bsxfun is used to apply element wise operation to two arrays. This masked image is our processed image.

If the created mask returns nothing that means our user is not under the coverage of the camera and our system automatically starts to work with latest captured image.

There are possibilities of other connected objects with the same color in the coverage area. This can create a havoc as it will be considered as another user. To avoid this problem, our system will count the number of connected objects of the same color first. If more than one connected object is detected, then this information will be relayed to the server and server will immediately change the color allocated to our user. Next, the same color processing procedures will be followed but this time after thresholding the image for the newly assigned color and creating the new mask, our system will apply the mask on the previously processed image.

That will make sure that other connected objects will be omitted as the only connected object with the color change will be our device. New mask will have connected objects of the newly assigned color only and the old mask had connected objects of the old color only. When the new mask is applied on the old masked image only common object that remains is our device. So, this insures the system against errors. This masked image will be the final processed image for more than one connected objects present in the area of coverage.

F. Location matrix

After getting the final processed image we make a pixel value matrix of the same size as the reference grid. Analyzing the matrix can give us exact location of our user. The way we have created our grid, one cell should be enough to house our user. So, the cell number with pixel values will be the position of our user as other cells won’t have any pixel value. The reason for that is after processing the image, we are only keeping the device and every other part of the image is blacked out and the pixel value for black is zero. This location matrix can further be used for creating a navigation system.

It is possible to get two or four cells with pixel values. That may happen because our user can stand between two cells or four cells, no other combination is possible. In that case we use “regionprops” [11] function of MATLAB with centroid property which gives us the middle point of our device.

G. Distance measurement

We can measure the distance of our user from a reference point by using the equation:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (1)$$

Here, d is the distance between the middle point of our device and the reference point. (x_1, y_1) are the coordinates of the reference point and (x_2, y_2) are the coordinates of the middle point of our device. This equation gives us the distance in pixels. To convert this value in meters:

$$distance = d \times \frac{pixel\ length\ of\ the\ cell}{actual\ length\ of\ the\ cell} \quad (2)$$

Here, Pixel length of the cell can be calculated by determining end to end points of a cell and using Equation(1). Actual length of cell should be calculated manually.

H. Smartphone application

Initially when the application is launched in the device, the onCreated() method is called in android. (On other platforms there is a method which is not a constructor but is called initially)

This method will run the following:

1. It will first request for a unique color. This request will be completed using the http protocol. In addition with the http post request we will send the unique user ID to the server.

2. The server will respond with a response code:200 and a json containing the hex value of the requested color.

3. The json will be parsed to get the hex value of the color.

4. Color of the main-layout will be set to the given color.

Our application will be registered with the platform's push notification listener service to receive push notifications. Push notifications are such a notifications or messages which are send to the device from the server without any request for those from the device. OnPushRecevied() is a common method which is triggered when a push notification is received.

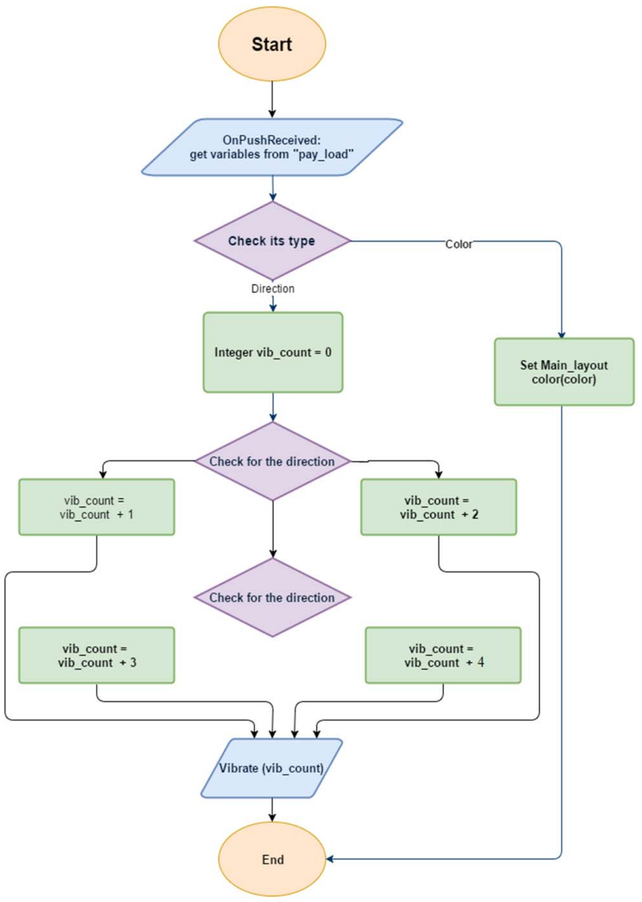

Start
OnPushReceived: get variables from "pay_load"
Check its type
Color
Direction
Integer vib_count = 0
Set Main_layout color(color)
Check for the direction
vib_count = vib_count + 1
vib_count = vib_count + 2
Check for the direction
vib_count = vib_count + 3
vib_count = vib_count + 4
Vibrate (vib_count)
End

Fig 3. Flowchart of the mobile application

After a push is received the following will take place:

1. Getting the payload of the push notification and checking for the "type" which is indicated in the payload. The "type" can be of two things: "Direction" & "Color"

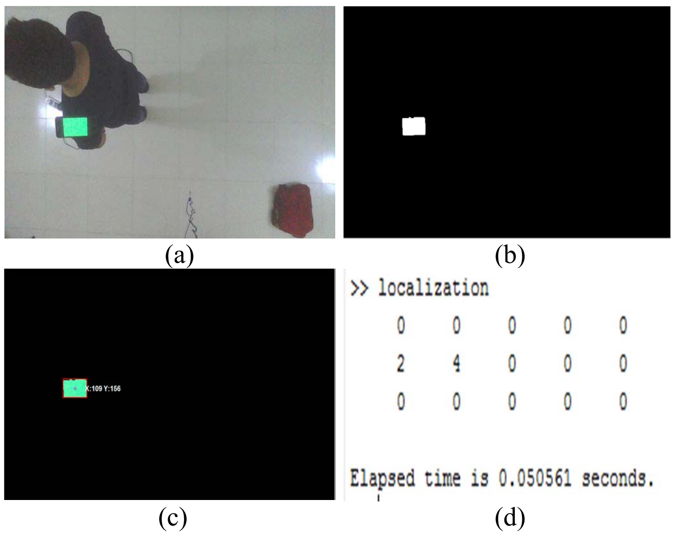
2. If it is "color" then, the message will also contain a hex value of the color. We will set the main-layout color of the application to the given color.

3. If it is of type "direction" then the app will check for which direction? To track this we are going to initialize an integer variable titled "vib_count" to 0. For up we will add 1 to it, for down it will be 2, for right and left we will add consequently 3 & 4. Then we will call the device's vibration method and set the vibration duration to 100ms and control the number of vibrations using a loop that is recorded in the "vib_count" variable.

Color change will happen when there are multiple connected objects or when the previously assigned color gets impossible to detect because of the effects of external lights.

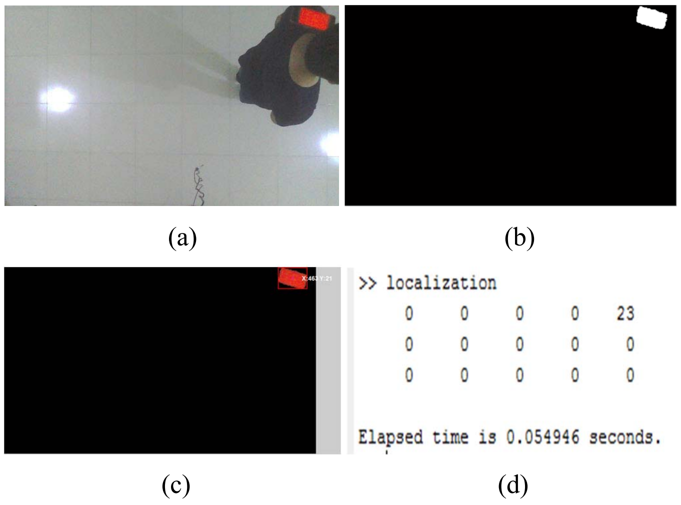
III. Experimental Result

We have implemented our system inside a single room with a single camera. Hercules HD twist webcam was used as the camera. As the server, we used a windows based computer with Intel core i3 processor, 4 GB RAM and Intel HD 4000 graphics card. The coverage area was $1.77 \times 2.87m$. We converted the HD images to the resolution of 500×300 pixels which made the image files very compact and easy to use. So, the reference grid was a 3×5 matrix and images were also divided into a 3×5 matrix with 15 location cells. Our code took 0.08 sec to run on average.



```
>> localization
     0     0     0     0     0
     2     4     0     0     0
     0     0     0     0     0

Elapsed time is 0.050561 seconds.
```

Fig. 4. Processed image for color green. Fig. 4(a) is the captured image and Fig. 4(b) shows the created mask. Fig. 4(c) is the final processed image while Fig. 4(d) shows the location matrix and elapsed time for the whole process.



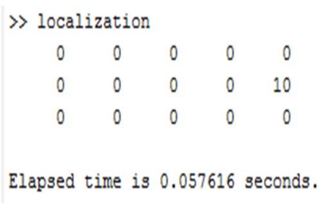
```
>> localization
     0     0     0     0    23
     0     0     0     0     0
     0     0     0     0     0

Elapsed time is 0.054946 seconds.
```

Fig. 5. Processed image for color red. Fig. 5(a) is the captured image and Fig. 5(b) shows the created mask. Fig. 5(c) is the final processed image while Fig. 5(d) shows the location matrix and elapsed time for the whole process

(a)

(b)

(c)



```
>> localization
      0     0     0     0     0
      0     0     0     0    10
      0     0     0     0     0

Elapsed time is 0.057616 seconds.
```

(d)

Fig. 6. Processed image for color blue. Fig. 6(a) is the captured image and Fig. 6(b) shows the created mask. Fig. 6(c) is the final processed image while Fig. 6(d) shows the location matrix and elapsed time for the whole process.

Fig. 2, Fig. 3 and Fig. 4 are for single connected object in RGB color space. We repeated the process 10 times for red, 10 times for green and 10 times for blue and each time we changed our user's location. Every single time we got an accurate reading.

We also calculated distance between a user and a reference point using equation(1) and equation(2). At first the distance between them was measured manually using a measuring tape and then they were measured using our system. Lastly, they were compared. Results are given in Table 2.

As we can see from table 2, the maximum error we got in distance measurement was 2cm. Out of ten distance measurement tests, eight distance accuracies were up to the decimeter mark.

Our experiment also shows that installing the camera on the ceiling of a room can solve the problem with line of sight which is a major issue in camera based localization systems.

Table 2: Distance accuracy

Case No.	Actual distance	Distance measured by our system	Error	Error Percentage	**Processing** Time
01	1.24m	1.22m	0.02m	1.61%	0.05s
02	1.12m	1.11m	0.01m	0.89%	0.09s
03	.72m	.716m	0.004 m	0.56%	0.08s
04	1.34m	1.342m	0.002 m	0.15%	0.04s
05	1.025 m	1.027m	0.002 m	0.19%	0.05s

(a)

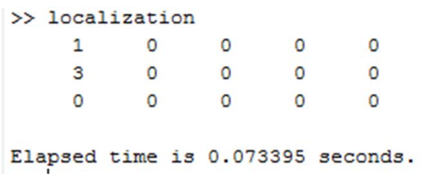
(b)

(c)

(d)

(e)

(f)



```
>> localization
     1     0     0     0     0
     3     0     0     0     0
     0     0     0     0     0

Elapsed time is 0.073395 seconds.
```

(g)

Fig. 7. Processed image for multiple connected objects. Fig. 7(a) is the captured image with multiple objects and Fig. 7(b) shows the first mask. Fig. 7(c) is the first masked image, Fig. 7(d) shows the new captured image with new allocated color, Fig. 7(e) is the new mask while Fig. 7(f) is the new masked image which us the final processed image. Fig. 7(g) shows the location matrix and elapsed time for the whole process.

IV. Conclusion

We developed an indoor localization system based on image processing with color detection. After implementing the system on a miniature scale, outputs were overwhelming as most of the times we got correct location with maximum error of 2cm and an average processing time of 0.08s. So, this paper provides solution to reliable and fast localization system for real time application.

Though our system is capable of handling one user at a time currently, adding support for more color can accommodate more users. Our system is ready to be used in navigation systems with our location matrix which can be used for short path calculation with algorithms like Astar [12] and Djikstra [13] and our developed android based application

As the system was developed considering a suitable system for visually impaired people where accuracy of the system was of utmost importance, we are very encouraged to see the results and future works will be done to develop a complete indoor navigation system equipped with turn by turn voice commands.

REFERENCES

[1] WHO, "Visual impairment and blindness," in *World Health Organization*, World Health Organization, 2016. [Online]. Available: http://www.who.int/mediacentre/factsheets/fs282/en/. Accessed: Oct. 27, 2016.

[2] WHO, "Global data on visual impairment," in *World Health Organization*, World Health Organization, 2012. [Online]. Available: http://www.who.int/blindness/publications/globaldata/en/. Accessed: Oct. 27, 2016.

[3] "National eye care,". [Online]. Available: http://nec.gov.bd/nio.php. Accessed: Oct. 27, 2016.

[4] K. Muthukrishnan, M. Lijding, and P. Havinga, "Towards smart surroundings: enabling techniques and technologies for localization", First International Workhop on Location- and Context- Awareness, LNCS3479, pp. 350-362 (2005)

[5] Kawaji, Hisato, Koki Hatada, Toshihiko Yamasaki, and Kiyoharu Aizawa. "Image-based indoor positioning system: fast image matching using omnidirectional panoramic images." In Proceedings of the 1st ACM international workshop on Multimodal pervasive video analysis, pp. 1-4. ACM, 2010.

[6] H. Aoki, B. Schiele and A. Pentland, "Realtime personal positioning system for a wearable computer," Wearable Computers, 1999. Digest of Papers. The Third International Symposium on, San Francisco, CA, USA, 1999, pp. 37-43. doi: 10.1109/ISWC.1999.806642

[7] S. Tilch and R. Mautz, "CLIPS proceedings," *2011 International Conference on Indoor Positioning and Indoor Navigation*, Sep. 2011.

[8] "Thresholding,".[Online].Available:https://www.wavemetrics.com/products/igorpro/imageprocessing/thresholding.htm. Accessed: Oct. 27, 2016.

[9] bwconncomp,"Bwlabel,"1994.[Online].Available:https://www.mathworks.com/help/images/ref/bwlabel.html. Accessed: Oct. 27, 2016.

[10] arrayfun,"Bsxfun,"1994.[Online].Available:https://www.mathworks.com/help/matlab/ref/bsxfun.html. Accessed: Oct. 27, 2016.

[11] bwconncomp,"Regionprops,"1994.[Online].Available:https://www.mathworks.com/help/images/ref/regionprops.html. Accessed: Oct. 27, 2016.

[12] R. Eranki, "Searching using A* (a-star)," 2002. [Online]. Available: http://web.mit.edu/eranki/www/tutorials/search/. Accessed: Oct. 28, 2016.

[13] "Data structures and Algorithms: Dijkstra's algorithm,". [Online]. Available: https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html. Accessed: Oct. 28, 2016.