MAVI: An Embedded Device to Assist Mobility of Visually Impaired

Rajesh Kedia*, Yoosuf K K*, Pappireddy Dedeepya*, Munib Fazal*, Chetan Arora†, and M. Balakrishnan*
* Indian Institute of Technology Delhi
† Indraprastha Institute of Information Technology Delhi
Contact Email: kedia@cse.iitd.ac.in

***Abstract*—Mobility for visually impaired people in an outdoor environment has always been a challenge. Traditionally, white cane or guide dogs have been used to help in mobility, but they suffer from many limitations. Modern solutions like Smartcane or Ultracane can help detect obstacles in the path, but cannot differentiate among these obstacles. Moreover, they do not support navigation. Considering the limitations with existing aids for mobility of visually impaired, we propose a new device to help visually impaired in their outdoor mobility. This device uses an RGB camera and image processing techniques to recognize the surrounding objects and inform/alarm the user accordingly. Our prototype incorporates signboard detection, face detection and surface texture detection to address issues related to mobility of a visually impaired user. Localization module implemented using GPS and IMU helps to support navigation. In this paper, we explain our initial implementation of various image processing tasks. We focus on the exploration for the choice of algorithms, design decisions, and various implementation decisions. We explain the challenges and results of implementing these applications on an embedded platform (Zedboard) for prototyping.**

I. INTRODUCTION

As per World Health Organization (WHO) fact sheet released in August 2014 [1], globally 285 million people are estimated to be visually impaired, out of which 39 million are blind and 246 million have low vision. The major problems faced by the visually impaired during outdoor pedestrian movement are obstacle detection and hindered navigation. These problems also lead to safety issues, social isolation, etc. Thus, there is a dire necessity to develop aids to assist the visually impaired people in overcoming these problems. Since 90% of the world's visually impaired people live in developing countries in low-income settings, the solutions targeting to address their mobility challenges should be low cost and affordable.

Traditionally white cane and guide dogs have been used by blind people to avoid obstacles in their walking path. White cane can detect obstacles only from a close distance and guide dogs are very expensive to breed and train as well as maintain. Solutions like Smartcane [2] and Ultracane [3] can improve the detection distance for cane and can also warn the user of a potential obstacle 3-4 meters ahead. However, they are not able to classify obstacles based on potential danger and also do not support navigation. Recent smart solutions built using Google Glass [4] or Microsoft's seeing AI [5] use a cloud server to perform all the processing, limiting their usability in areas without connectivity. Moreover, in order to save energy, such solutions tend to be user triggered and not spontaneous, restricting their usage in safe navigation applications.

Ye et al. [6], Apostolopoulos et al. [7], Jain et al. [8] propose aids for navigation and obstacle detection for visually impaired, but their research is limited to an indoor environment only. We are trying to address mobility in an outdoor environment which pose different challenges compared to indoors. Yasser et al. [9] proposed a solution for outdoor navigation but it assumes a controlled environment with various locations tagged with barcode. Moreover, the existing aids (commercial or academic) do not address problems specific to an unstructured environment, typical of a developing country like India. Our proposal identifies some of these specific challenges for visually impaired pedestrians walking outdoors and attempts to address them through the design of a dedicated device, which we name as Mobility Assistant for Visually Impaired (MAVI).

MAVI uses an optical RGB camera of VGA resolution to capture the surrounding scene and process the captured frames to extract useful information for the user. The specifications of MAVI has been defined based on a survey conducted by our group to assess the mobility needs for visually impaired and limitations with existing solutions. We aim to address navigation, safety, and social inclusion problems related to the outdoor mobility of visually impaired people, specifications drawn from the typical street infrastructure and environment of developing countries like India. We explain the features of MAVI in detail in section II.

Rather than developing new algorithms, MAVI development focusses on using existing image processing algorithms and adapting them to meet the requirements of the device. We have explored different algorithms to implement each of the requirements. For testing the developed system, we have conducted experiments on real pictures taken from our university campus. We had overcome various challenges associated with porting the developed algorithms to an embedded platform and getting the live system to work. We will discuss these in the later part of this paper.

The specific contributions of this paper are as follows:

- We have developed a concept demonstrator of a device targeting independent mobility for visually impaired people.
- The effectiveness of our algorithms has been validated offline using dataset captured in real settings.

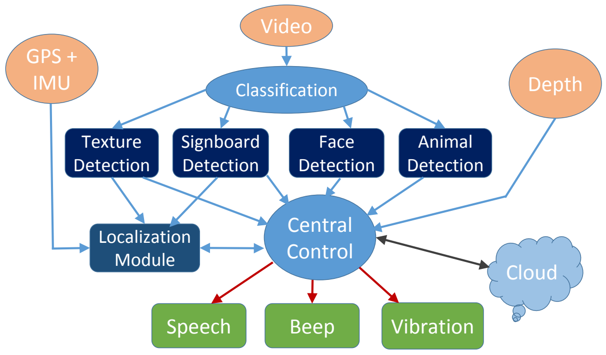

Fig. 1. Block diagram of MAVI system. The system aims to help visually impaired people in their following needs (a) Safety: Identify digging, potholes (surface texture module) and stray animals (animal detection module), (b) Navigation: Localization with GPS and IMU and identifying multi-lingual signboards, and (c) Social Inclusion: Detecting people and recognizing known faces (Face Detection module). The input to the current system is a VGA image from an RGB camera - depth sensor is planned to be added in future. Feedback to a user is given by voice, beep and vibration modes.

- We have validated our implementation and generated the results from a real embedded platform.

The rest of the paper is organized as follows. Section II describes the features and block diagram of MAVI. Section III explains the implementation of various sub-blocks in MAVI. Details of prototyping and porting on embedded platform are described in Section IV. We present the results from initial prototyping of MAVI in Section V. Section VI concludes the paper and describes the future work.

II. MAVI Specifications

Mobility Assistant for Visually Impaired (MAVI) is a device to help in independent mobility for visually impaired users. Fig. 1 explains the block diagram and features of the MAVI system. The input sensors consist of an RGB optical camera, GPS and IMU sensors. There are four image processing modules integrated with the system: Texture Detection (TD), Signboard Detection (SBD), Face Detection (FD) and Animal Detection (AD). The VGA image captured through RGB camera is passed onto these blocks and depending on their outputs, further actions are invoked. For example, if SBD detects a signboard, then an Optical Character Recognition (OCR) subsystem is invoked to read the contents of the signboard. A central controller coordinates enabling each of the modules as per the need. A cloud based server is integrated for storing landmarks and other relevant information. The server may also be used to offload intensive computations in future releases. The localization module uses GPS information augmented with IMU mechanization to detect the location of the user. The user interface is implemented over mobile which also acts as a gateway for the device to connect to the cloud.

III. Implementation Details

In this section, we explain the choice of image processing algorithms for SBD, TD and FD modules and corresponding implementation details. The implementation is based on OpenCV 3.1 libraries [10]. Our current implementation is limited to the infrastructure - footpath, signboards, etc. available within our university campus.

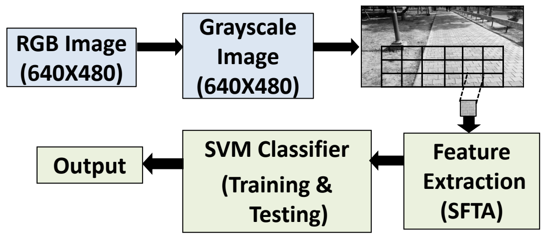
RGB Image (640X480)
Grayscale Image (640X480)
Output
SVM Classifier (Training & Testing)
Feature Extraction (SFTA)

Fig. 2. Block diagram of Texture Detection module. We use features based upon Segmentation-based Fractal Texture Analysis (SFTA) [15] followed by one vs all SVM classifier for different classes. SFTA features gave similar accuracy as Gabor but computed in a fraction of time. At test time we used sliding window based approach and classified each window into one of the four categories: pavement, road, grass, and mud.

A. *Texture Detection*

We chose four different types of surface textures: pavement, road, grass and mud. Texture Detection (TD) implementation to classify the surface into one of these textures is based on Support Vector Machine (SVM) classifier owing to its effectiveness and the portability constraints to an embedded platform. Though Deep Neural Networks (DNN) have shown state of the art results in many areas in last few years, we have not used them in our setup due to limited dataset available and limited memory available on the target embedded platform. In the future, we plan to explore DNN with pre-training/fine-tuning techniques to work with the limited dataset and pruning techniques to fit the network into limited memory. Fig. 2 shows an overview of our implementation of TD, which is explained in following section.

1) Texture Feature Extraction: Texture feature extraction involves creating a database of various classes of textures to be identified. For the training and test dataset, we have captured VGA images (640$\times$480) with various camera orientation, scale and illumination conditions. We have captured more than 200 samples per texture class. After experimenting with various descriptors like SIFT [11], SURF [12], LBP [13], Gabor [14] etc., we found Segmentation-based Fractal Texture Analysis (SFTA) [15] to be a good compromise for speed and accuracy. The samples for training are cropped from original images of size 640$\times$480 to sizes of: 80$\times$60, 80$\times$80, 160$\times$120, and 160$\times$160. We chose SVM classifier with linear kernel, since in our experiments linear kernel showed reasonable accuracy and was faster compared to polynomial and RBF kernels. We train separate one vs. all SVM classifiers for each class.

2) Texture Classification: The extracted features from previous step is used for training the SVM classifier. Each classifier is trained with more than 3000 samples.

At test time we used sliding window based approach. We tested with both overlapping and non-overlapping sliding windows. To reduce the search space, we choose the region of interest (ROI) by removing image portions containing sky

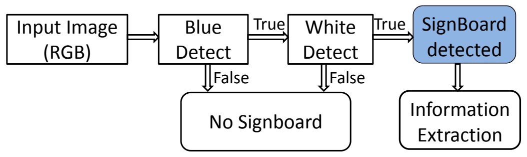

Fig. 3. Overview of the signboard detection: The signboard in our context are standardized as white text written on a blue background. The detection module is designed accordingly detecting the blue clusters first. We select two candidate regions with maximum blue and validate the same by ensuring a pre-selected amount of white pixels in it. Each such verified candidate regions is then sent to the next stage for OCR.

or trees. This improves accuracy as well as performance of the task. In the following text, we explain our methodology to choose the ROI.

3) Extracting Region Of Interest (ROI): We analyzed the distribution of features of interest in various images by analyzing how many rows of pixels contain desired pavement information. Our analysis indicates that in 90% of the test samples, relevant texture is present only in lower 240 pixel rows. Hence, we removed upper 240 rows of pixels before attempting to classify the image regions. Based on similar analysis, we also removed 80 vertical rows of pixels from both sides of the image. Thus the region of interest within each image gets restricted to 480×240 and reduces computation time substantially.

B. Signboard Detection and Information Extraction

Fig. 3 depicts various steps involved in the signboard detection algorithm. The detection process is divided into two stages: The blue detection stage followed by the white identification stage. The input image for all modules of MAVI system is of VGA resolution (640×480 pixels).

1) Blue Detection Stage (BD): In the blue detection stage, the image is transformed from the RGB space into a Hue, Saturation, and Value (HSV) space and then a mask is obtained for the blue coloured pixels in the image using simple thresholding. We then obtain the contours for each connected region of blue pixels using implementation of [16] in OpenCV [10]. We sort the contours in descending order in terms of their area and then approximate and bound the contours using rectangles. The two largest rectangles obtained are assumed to be possible candidates for signboards and are passed as inputs to the next stage.

2) White Detection Stage (WD): Within each candidate region detected as blue box, we find pixels coloured white and cluster the pixels into different clusters based upon flood fill algorithm. Then, we calculate the amount of white colour by counting the number of white pixels in all these regions. Further, the ratio of this to the total number of pixels in the image is compared against a predefined threshold. If it exceeds the threshold, then the bounding box is considered to be a potential signboard and passed onto next stage for text extraction.

3) Information Extraction: This stage is organized into various steps like noise removal, word extraction, rectification of words, character extraction and passing it to the OCR engine. The signboards are assumed to be bilingual and depending on the language (Hindi or English), we choose the OCR engine. Currently, the information extraction is implemented in MATLAB only. The Tesseract engine [17] is used for the English text identification where as a preliminary Hindi text identification module developed in-house is used for Hindi text. Currently, Hindi text identification is in early stage and needs further improvement. The implementation of information extraction block is shown in Fig. 4. Noise removal is done using the area of connected components as a parameter. Word extraction is done based on the aspect ratio of connected components. Finally, each text line is passed to the appropriate OCR engine depending on the language.

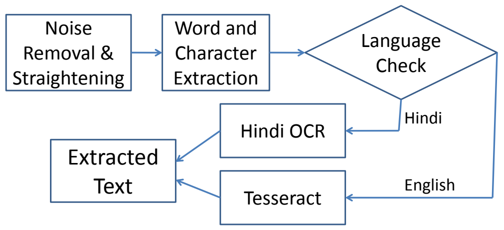

Fig. 4. Block Diagram of Information Extraction: For each candidate region for signboard detection, we attempt to recognize the text by an external OCR engine. We first perform perspective correction to compensate for tilted text. The signboards are bilingual and we detect each text line to be Hindi or English based upon the presence of header line (in Hindi). Each text line is passed separately to Tesseract (for English) or an in-house engine (for Hindi).

Modes of Operation: The algorithm for signboard detection supports two modes: Default mode and low energy mode. We observe that we need VGA sized images for reasonable OCR, however, just the signboard detection can be done reliably even at lower resolutions as well. Thus, while the default mode is same as described above, in low energy mode, we reduce the computation overhead by either resizing the image to 320×240 for blue detection stage or sending only the largest bounding box from BD stage to WD stage. In both the cases, the search space is reduced resulting in a lower number of computations. Based on the above description, we define four operating modes in SBD task: NRS_BOTH, NRS_SINGLE, RS_BOTH and RS_SINGLE. The SINGLE suffix denotes that a single bounding box is passed from BD to WD stage while NRS and RS refer to No-ReSize or ReSize of the input image before BD stage.

C. Face Detection and Recognition

The motivation for including face recognition (FR) in a mobility aid is to help visually impaired users identify their known persons, if encountered on the path. Face detection (FD) helps to inform the users about any human being present in the vicinity, so that the user can seek help if needed. The block diagram of the implementation is shown in Fig. 5. The following sections explain the details of FD and FR modules.

1) Face Detection: The seminal work by Viola and Jones [18] lead to the first real time face detection framework in computer vision. We follow the same classical algorithm and use pre-trained OpenCV Haar cascade classifier consisting of

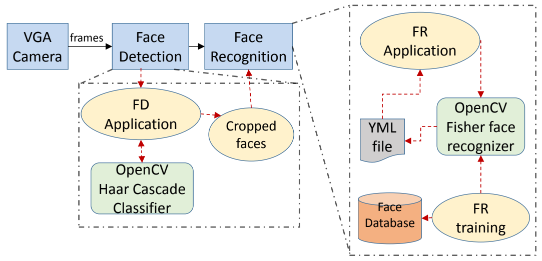

Fig. 5. Block Diagram of Face Detection (FD) and Recognition (FR) Module: The algorithm operates in two steps: FD and FR. FR is invoked only if a face is detected in FD stage. Instead of sending the entire frame to FR, the cropped portion corresponding to the detected face is sent.

20 stages and 1047 features. The OpenCV functions require multiple parameters to be specified based on the application needs. These parameters impact the accuracy of detection as well as execution time. We explored *scaleFactor* and *minNeighbor* parameters during our implementation of FD.

Parameter tuning for FD: scaleFactor is one of the parameters for FD which controls the scaling of frames from one pass of the algorithm to the next. We varied scaleFactor from 1.05 to 1.4 in intervals of 0.05 over the test data of 65 images containing 0, 1 or 2 faces. We measured the accuracy and computation time from the embedded implementation for each of these settings to come up with the optimal value of scaleFactor. Another parameter named minNeighbor allows for removal of false positives from the detection results but leads to missing out some of the true results. Based on experiments with different database, we chose minNeighbor=2 for our implementation.

Minimum face size: Ability to detect faces of smaller size enables the device to support face detection from larger distance. The trained cascade classifier provided along with OpenCV has minimum face window of 20×20 pixels. From our experiments, we determined that this corresponds to a maximum detection distance of 6 meters. For detection of further small faces, we opted for scaling the input image and then performing FD. This has adverse effect on computation time and false positives though it could detect smaller faces of size up to 7×7 pixels (by scaling the image to four times). This corresponds to a distance of about 12 meters. Note that smaller faces also impact the accuracy in FR and our experiments conclude that faces closer than 3 meters gave a recognition rate of 65%, while beyond 3 meters, an accuracy of less than 50% was achieved.

2) Face Recognition: Once a face is detected by the FD stage, FR system attempts to match the cropped portion of image with its database of faces. Simonyan et al. [19] have shown that Fisher vectors [20] on densely sampled SIFT features [11], i.e. an off-the-shelf object recognition representation, are capable of achieving state-of-the-art face verification performance on the challenging “Labeled Faces in the Wild” benchmark [21]. We use the fisher vector representation as given by the Simonyan et al. and trained the face verification module with subjects from our research group.

FR training database size: FR involves training the classifier using sample faces of interest. The number of faces used in the training is one of the factors impacting the accuracy of recognition. We experimented with sizes of 5, 8 and 10 images/face for each of 10 different persons. The test dataset consisted of 25 images from 2 seen and 1 unseen person. Highest accuracy of 65% was achieved with 8 images/face and we chose this size for our final implementation.

FR database creation and training: Creation of training database for face recognition is one of the most time consuming activity during the implementation of FD+FR module. Multiple pictures were taken for 10 subjects in indoor and outdoor illumination conditions with varied facial expressions like smiling, sad, laughing, straight, serious. Few images with some shadow and slight tilting were also included to account for varying sunlight and face orientations. Next, the faces were cropped manually from these images. The face images were registered with nose as the center and the area under hair, ears and tip of the chin removed from the faces.

IV. Embedded Implementation

Our embedded implementation uses ZedBoard [22] as the platform which consists of Zynq device from Xilinx. It consists of a dual core ARM Cortex-A9 processor forming the processing system (PS) and a programmable logic (PL) to implement custom hardware. The PL is tightly integrated with the PS. We explain key aspects related to our embedded implementation and how we managed associated challenges.

A. Linaro Operating System and Booting

For the software-only implementation, Linaro Ubuntu distribution was used [23]. Linaro is an open-source Linux distribution based on Ubuntu. It supports graphical desktop through on-board HDMI port. It is a persistent OS, i.e. all changes are written to memory and it saves files after a reboot or shut down. OpenCV 3.1 was built on top of it.

B. Compiling OpenCV and Application for ZedBoard

ARM toolchain provided for Linux *arm-linux-gnueabihf* was installed and used to compile OpenCV 3.1 source for Linaro. OpenCV compilation was explored by using shared library, static library and stripped down version of OpenCV. The same tool chain was used to compile the application for ZedBoard.

One of the biggest challenge in porting the implementation to ZedBoard was the large size (≈600MB) of OpenCV library. It could not fit in limited memory available on ZedBoard. To enable porting to ZedBoard, we strip the library to consist of only the functions required as per our algorithm implementation. With this, we were able to reduce the library size to ≈50MB which could easily fit on the embedded platform.

C. Profiling of the Application

Profiling is a key methodology used to analyze the relative software runtime for different blocks. We performed profiling to identify candidates for hardware acceleration within each task. We performed profiling using gprof and perf tool.

TABLE I
ACCURACY, EXECUTION TIME AND ENERGY FOR TEXTURE DETECTION

Classifier	Accuracy (in %) for various window sizes			
	80×60	80×80	160×120	160×160
Pavement	87.8	91.26	86.91	82
Road	65.8	70.3	75.6	71
Grass	83.28	96.74	88.3	92.57
Avg. Exec. Time (s)	2.55	1.83	2.72	2.69
Energy (mJ)	306	219.6	326.4	322.8

D. Application Power Measurement on ZedBoard

A 10 milli Ohm, 1 watt current sense resistor is present in series with the 12V input power supply. A multimeter (Agilent 34410A) was used to measure the voltage. It can be interfaced to PC and allows to start/stop the measurement, view the waveform and export the results to a text file. We obtained the current drawn by the board by dividing the voltage value with resistance (Ohm's law).

The base power drawn by the ZedBoard was very high due to the presence of many other components on the board. To explore various implementation options, we needed to measure the contribution from our application alone which posed another challenge during our implementation. Our approach to this involved running a dummy code pinned to core-0 to keep it busy. This was measured as the reference current. Now the application was pinned to core-1, and the incremental change in current was considered as the contribution from our application.

V. RESULTS

A. Texture Detection

The accuracy in percentage for various classifiers, average execution time and energy consumption for various window sizes is shown in Table I.

The time taken for classification depends on the following factors:

- Number of sample windows per frame, which reduces as the window size increases.
- Computational overhead of feature extraction, which increases with increasing window size.

The two factors affect the computation time in opposite ways and explains the results presented in Table I. We observed best compromise between accuracy and computation time with a window size of 80×80 and have chosen this size for all our experiments.

B. Signboard Detection

The dataset consisted of 2957 images of various signboards in our campus. The images were captured for various directions, time of the day and distances and then manually segregated as True (with signboard) and False (without signboard). 1724 images were tagged as True while 1233 were tagged as False. Detection accuracy for these images is shown in Table II. In our experiments, the proposed system works with more than 90% accuracy up to a distance of 4 meters.

TABLE II
ACCURACY FOR SIGNBOARD DETECTION

Type	Number	Percentage
True Positives	1658	96.1721
True Negatives	1168	94.7283
False Positives	65	5.27169
False Negatives	66	3.82830

TABLE III
PERFORMANCE COMPARISON FOR DESKTOP AND ZEDBOARD IMPLEMENTATION FOR DIFFERENT MODES IN SBD

Mode and Outcome of SBD		Time Elapsed (ms)		Energy (mJ)
		Desktop	ZedBoard	on ZedBoard
NRS_BOTH	True	34.25442	457.1337	71.31
	False	28.6954	209.850	32.73
NRS_SINGLE	True	32.42210	444.5515	69.35
	False	25.33441	206.9383	32.28
RS_BOTH	True	17.9652	261.2473	40.75
	False	12.1194	54.51582	8.5
RS_SINGLE	True	17.9232	234.5172	36.58
	False	10.94612	53.72602	8.38

We also measured the average time taken for executing the task for each image using different modes of the algorithm. Table III shows the performance and energy results from Desktop and ZedBoard implementations. As expected, the energy consumed is least for RS_SINGLE mode while it is highest for NRS_BOTH. Currently, due to software only implementation, energy consumption results follow the execution time results. However, in future, when some of the kernels are implemented in hardware, we expect the results to provide interesting design trade-offs.

C. Face Detection and Recognition

1) Variations with scaleFactor: Face detection task was run on ZedBoard for 65 outdoor images consisting of no faces, one face or more than one faces in different images. We varied *scaleFactor* parameter of the OpenCV API and measured various metrics for each *scaleFactor* setting. Fig. 6 shows the variation in accuracy of face detection with the increase in *scaleFactor*. The results are normalized w.r.t. the accuracy obtained for *scaleFactor*=1.05 (it corresponds to an accuracy of 64.55%). Since a higher *scaleFactor* reduces the search space, we observe a decrease in execution time as *scaleFactor* is increased. Fig. 7 shows the trend. We selected *scaleFactor* of 1.2 for our prototype implementation since it provides a reasonable trade-off between execution time and accuracy.

Fig. 8 shows the variation in energy consumption for face detection with *scaleFactor*. Increasing the value of *scaleFactor* provides a considerable savings in the energy consumed for processing. We plan to switch to a higher value of *scaleFactor* in case the battery level is low and achieve energy savings at the cost of accuracy.

2) Distance supported vs. accuracy: We performed experiments to measure the size of face at various distances which

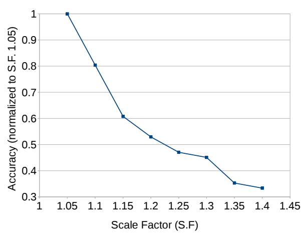

Fig. 6. Face Detection: Accuracy

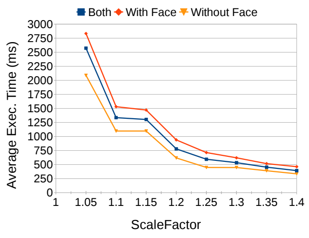

Fig. 7. Time taken for FD

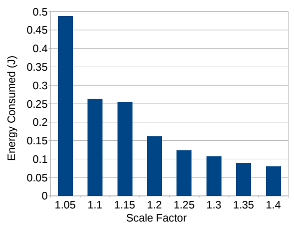

Fig. 8. Energy for Face Detection

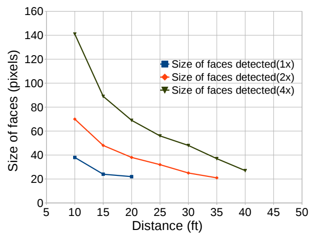

Fig. 9. Size of detected face at various distances

are successfully detected by our algorithm. The results are presented in Fig. 9. We only plot the points which are greater than 20 pixels (approximately 6 meters in our camera settings). A digital scaling of the image by $2\times$ was observed to increase the detection accuracy at larger distances. The number of false positives increases substantially when the image was scaled by $4\times$. However, we did not implement scaling in the current version due to a substantial increase in computation time.

VI. Conclusion

In this paper, we present the design of a dedicated device (MAVI) being developed by us to assist outdoor mobility for visually impaired users. The specifications of MAVI and details related to the implementation of various modules have been described. We have presented the results related to accuracy, execution time and energy consumption and corresponding design decisions based on these results. One key learning from this design exercise was that for such complex embedded systems, application performance metrics (accuracy, response time, battery life) and associated trade-offs with power, hardware resource requirements, etc. are key to getting an efficient implementation.

The current status of the device is an initial concept demonstrator with SBD, TD and FD tasks implemented and integrated on ZedBoard as software modules. The localization module is implemented as hardware software co-design. A simple controller is implemented in software to schedule different tasks. A mobile application provides feedback from the device to the user.

As future work, we plan to improve the prototype by implementing key kernels from FD and TD tasks as hardware in PL portion of ZedBoard and improve the achieved frame rate. We also plan to make the central controller more intelligent so that appropriate runtime decisions could be taken for efficient utilization of resources. Animal detection is also being worked on and integrated on the system. Our another ongoing work on systematic design space exploration for such complex systems would help in better analysis leading to optimal design decisions.

Acknowledgement

We would like to thank Piyush Chanana from IIT Delhi for the detailed user survey which helped define MAVI specifications. We would also like to thank Ministry of Electronics and Information Technology (MeitY) for funding MAVI under the project “Special Manpower Development Programme for Chips to System Design (SMDP-C2SD)”.

References

[1] “World health organization. visual impairment and blindness.” www.who.int/mediacentre/factsheets/fs282/en.

[2] “Smartcane,” http://assistech.iitd.ernet.in/smartcane.php.

[3] “Ultracane,” https://www.ultracane.com/.

[4] “Google glass applications for blind and visually impaired users.” http://www.visionaware.org/blog/visionaware-blog/google-glass-applications-for-blind-and-visually-impaired-users/12.

[5] “Seeing AI project,” http://www.pivothead.com/seeingai/.

[6] C. Ye, S. Hong, X. Qian, and W. Wu, “Co-robotic cane: A new robotic navigation aid for the visually impaired,” *IEEE Systems, Man, and Cybernetics Magazine*, vol. 2, no. 2, pp. 33–42, April 2016.

[7] I. Apostolopoulos, N. Fallah, E. Folmer, and K. E. Bekris, “Integrated online localization and navigation for people with visual impairments using smart phones,” *ACM Trans. Interact. Intell. Syst.*, vol. 3, no. 4, pp. 21:1–21:28, Jan. 2014.

[8] D. Jain, A. Jain, R. Paul, A. Komarika, and M. Balakrishnan, “A cellphone based path-directed indoor navigation system for persons with visual impairment,” in *15th ACM SIGACCESS International Conference on Computers and Accessibility (ASSETS)*, 2013.

[9] Y. Ebrahim, W. Abdelsalam, M. Ahmed, and S.-C. Chau, “Proposing a hybrid tag-camera-based identification and navigation aid for the visually impaired,” in *Second IEEE Consumer Communications and Networking Conference, 2005. CCNC. 2005*, Jan 2005, pp. 172–177.

[10] “OpenCV. Open source computer vision.” http://opencv.org/.

[11] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[12] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[13] T. Ojala, M. Pietikäinen, and T. Mäenpää, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[14] M. R. Turner, “Texture discrimination by gabor functions,” *Biol. Cybern.*, vol. 55, no. 2-3, pp. 71–82, Nov. 1986.

[15] A. F. Costa, G. Humpire-Mamani, and A. J. M. Traina, “An efficient algorithm for fractal analysis of textures,” in *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, Aug 2012, pp. 39–46.

[16] S. Suzuki and K. Be, “Topological structural analysis of digitized binary images by border following,” *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, Apr. 1985.

[17] “Tesseract,” https://github.com/tesseract-ocr.

[18] P. Viola and M. J. Jones, “Robust real-time face detection,” *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[19] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Fisher Vector Faces in the Wild,” in *British Machine Vision Conference*, 2013.

[20] F. Perronnin, J. Sánchez, and T. Mensink, “Improving the fisher kernel for large-scale image classification,” in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ser. ECCV’10, 2010, pp. 143–156.

[21] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.

[22] “Zedboard,” http://zedboard.org/product/zedboard.

[23] “Linaro,” http://www.linaro.org/.