O’Map: An Assistive Solution for Identifying and Localizing Objects in a Semi-Structured Environment

Shahinur Alam, ASM Iftekhar Anam, Mohammed Yeasin
The University of Memphis
salam@memphis.edu, aanam@memphis.edu, myeasin@memphis.edu

Abstract

A system capable of detection and localization of objects of interest (OI) in a semi-structured environment will enhance the quality of life of people who are blind or visually impaired. Towards building such a system, this paper presents a personalized real-time system called O’Map that runs on Android devices. Main feature of the system is to locate misplaced/moved personal items in a familiar setting. The system provides auditory feedback about the presence and relative position of the query item with respect to known landmarks. First, we adopted participatory design approach to identify users need, functionalities of the system and personalization (i.e., user profile and object map of the environment) in collaboration with representative users. Second, we used the concept from system thinking to develop a real-time object recognition engine that was optimized to run on low form factor devices. Finally, concepts from design thinking were adopted to implement the feedback and user interface to interact with the system. Quantitative evaluation demonstrates that O’Map identifies object of interest with an average F-measure of 0.9650.

Keywords

Introduction

The World Health Organization estimated in 2013 that 285 million people are visually impaired. Among those, 39 million are blind and 246 million have low vision. According to project “Cost of vision” conducted by Prevent Blindness America (PBA), the cost of ocular disorders was $139 billion for the year 2013 in the United States. While the economic cost is well documented yet the personal cost and quality of life remains poor despite the progresses in assistive solutions. People with impairment/loss of sight face difficulties interacting with the visual environment, especially in finding misplaced items, navigation, and understanding non-verbal communication, just to name a few. A reconfigurable and portable assistive solution can make semi-structured environments more accessible. In recent years, a plethora of systems were reported in the literature to improve the mobility, readability and interaction with environment. Despite the progresses made most of the systems are not fully accessible and portable. In addition, these systems lack the robustness to be used in natural environment.

In this paper, we present a novel system, O’Map, to find misplaced/query items (Fig.1) and its relative position with respect to reference items (Fig.2) in a semi structured environment.

Fig. 1. Sample Query Item.
Users can use either smartphone or Google glass to capture video of the environment and to receive feedback. Though the Google Glass is expensive and not widely available yet, it is more ergonomic compared to head or neck mounted cameras and, hence, can be put into practical use. Therefore, we designed the system that can be used on either smart phone or Glass. At first, the users create a personal profile with items that they use frequently (see “Profile Creation” section). The O'Map has an android app, which communicates with a server. The android application supports creating and maintaining user profile through a set of utility features. The user can interact with the application via speech command that is processed using Google Speech Recognizer. When the user selects first feature- “find item”, the O'Map starts with checking the lighting condition of the environment. Then, it prompts the user to record the name of the item of interest. The user is then asked to create 360° panorama. The system assists the users to create a panorama using information from the compass and inertial sensor. The data are then sent to the server where matching is performed with “template images” of query item
and reference items (described in “Item Matching” section). The system builds an object map of reference items and read out the positions in clock-orientation which is closely located to the identified item. Once the item is found, it keeps the record in log. However, if the query item is not found in the panorama, the user is instructed to move towards a reference item based on the past recognition history or user discretion. The client application is notified to send individual video frames which are processed until the item is found. A demo is available at http://youtu.be/GBgO5o8dptM

Related Work

A number of systems were reported in the past literature for finding item of interest using a wide spectrum of technology such as Audio Signal/Sonar, Human assistance, Computer Vision, and RFID/Tag. Table 1, 2, 3, and 4 shows related works from aforementioned four categories highlighting approach used, functionalities considered, key contribution and shortcomings. Works most relevant to the O’Map are presented in Table 3.

Table 1. Audio Signal/ Sonar based System

<table>
<thead>
<tr>
<th>Reference</th>
<th>Method</th>
<th>Key Findings</th>
<th>Shortcomings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kao et al.</td>
<td>FM sonar is used to detect object and the distance is calculated from reflecting signals</td>
<td>Detect objects and obstacles with depth information of open space</td>
<td>• It requires external FM sonar signal generator and not robust in complex and noisy environment. • This system cannot be used to recognize item of interest</td>
</tr>
</tbody>
</table>
### Table 2. Human Assistance based system

<table>
<thead>
<tr>
<th>Reference</th>
<th>Method</th>
<th>Key Findings</th>
<th>Shortcomings</th>
</tr>
</thead>
<tbody>
<tr>
<td>VizWiz TapTapSee</td>
<td>Captured picture with query is sent to crowd to receive feedbacks.</td>
<td>• It enables user to ask free form query.</td>
<td>• Depends on human assistances</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• System does not require any training</td>
<td>• Sometimes it takes longer time to receive response, especially at night.</td>
</tr>
</tbody>
</table>

### Table 3. Computer Vision and Image Analysis Based System

<table>
<thead>
<tr>
<th>Reference</th>
<th>Method</th>
<th>Key Findings</th>
<th>Shortcomings</th>
</tr>
</thead>
<tbody>
<tr>
<td>LookTel Recognizer</td>
<td>Instantly recognize objects from recorded list.</td>
<td>• Real time system</td>
<td>• Less flexibility in multi-view detection</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fast recognition</td>
<td>• No option to provide user query for a specific item</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Requires no internet connection</td>
<td></td>
</tr>
<tr>
<td>ORB Object Recognition</td>
<td>Matches selected images from gallery or video stream</td>
<td>Invariant to the photometric and geometric changes</td>
<td>Considers single view only</td>
</tr>
<tr>
<td>Talking Goggles</td>
<td>Provides descriptions of familiar objects from video stream</td>
<td>Tells where (shops) identified item can be found and price comparison</td>
<td>• No option to provide user query for a specific item</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Lack of accessibility</td>
</tr>
<tr>
<td>Andreas Hub et al.</td>
<td>Sensor module with stereo camera is used to detect objects based on image segmentation and color</td>
<td>Tells about object characteristics, position, orientation and way to navigate</td>
<td>• Requires extra devices such as WLAN card to integrate multiple sensors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Not invariant to photometric changes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Stereo view processing takes time</td>
</tr>
<tr>
<td>Reference</td>
<td>Method</td>
<td>Key Findings</td>
<td>Shortcomings</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Caperna et al.    | GPS and internal navigation unit (INU) are used to provide walking direction in open space | Helps to identify signs, landmarks and arbitrary objects          | • Provides only way-finding utility  
• No option to provide user query for a specific item. |
| Boris et al.      | Uses color attribute and SIFT feature to find item and sonification for feedback | Builds color model from large image collection                     | Color attribute is not invariant to the photometric changes                    |
| Yi et al.         | The places where items get displaced very frequently are equipped with fixed camera. Images from all cameras are matched with query Item. | Integrated hand free device                                        | • Requires multiple camera and processing images from all sources requires time  
• Can find items from arbitrary places |
| Ricardo et al.    | Use template matching technique to recognize item                      | Real-time object recognition                                      | Do not help to localize items                                                  |
| Andreas Hub et al.| Sensor data, 3D model, shape and color information are used to detect objects | It can detect free and movable objects                              | No option to search and localize item of interest                             |
| Tanveer et al.    | Google glass is used to recognize affective cues.                      | Tells about number of people, gender and age in small talk        | No option to search and localize item of interest                             |
| Alexander et al.  | Hand free Optical Head Mounted Display used in traversing large open space | Used to identify salient landmarks such as doors, exit signs etc.  | • Helps only way finding  
• No option to search and localize item of interest |

Journal on Technology and Persons with Disabilities  
Santiago, J. (Eds): Annual International Technology and Persons with Disabilities Conference  
© 2015 California State University, Northridge
Table 4. RFID/Bluetooth Tag Based System

<table>
<thead>
<tr>
<th>Reference</th>
<th>Method</th>
<th>Key Findings</th>
<th>Shortcomings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kientz et al.</td>
<td>Bluetooth enabled tags and cell phones or laptop are used to identify items which are attached with RFID tag.</td>
<td>• Can track and locate objects</td>
<td>Requires extra tags for each items</td>
</tr>
<tr>
<td>FindIt, KeyRinger</td>
<td>Misplaced items are tracked using State-of-the-art circuitry where one part of paired devices is attached with query item and other part is placed in safer place</td>
<td>• Can track and locate objects</td>
<td>Requires extra tags for each items</td>
</tr>
<tr>
<td>SonicKeyFinder</td>
<td></td>
<td>• Has wide coverage</td>
<td>Extra maintenance cost for battery</td>
</tr>
<tr>
<td>FindOne</td>
<td></td>
<td></td>
<td>Not free</td>
</tr>
</tbody>
</table>

A survey with visually impaired people revealed that an effective and usable object recognition system must address accessibility issue, focalization and cropping problem, and will be invariant to the photometric and geometric changes. Past research fell short addressing all these requirements. This observation led us to design the O’Map described below.

Assumption, Design, and Development of O’Map

We adopted principles from participatory design to understand the user’s needs and applied ideas from System, Design, and Assistive Thinking to optimize system performance, increase usability and reduce complexity. The distribution of targeted users and their ability to receive O’Map’s service was accounted in system design. Although smartphone is widely available and preferred device, some hands free device like Google Glass is a rare commodity, especially in the rural area. To reach a largest possible number of users, we decided to develop O’Map on multiple platforms- either on standalone smartphone or a combination of smartphone and Google Glass with a remote server (or cloud). In the alpha version, O’Map assumes that sighted people will help to create personal profile. The beta version enables visually impaired users to enlist items by themselves (cf. “Profile Creation” section). We also assume that, the dimensions
of each item are known, which are necessary to estimate the approximate distance between identified item and user (cf. "Distance Estimation" section).

**Personal Profile creation:**

To personalize the application we account for user’s preferences through profile. Initially, with or without the assistance of sighted person, the user creates her profile that contains a list of items with their name, approximate size (height and width), and images. In the alpha version, four utility options such as “View”, “Add”, “Edit”, “Delete item”, were used to create personal profile which are shown in Fig.3.

![Personal Profile Creation](image)

Fig.3. Personal Profile Creation

In the beta version, O’Map enables visually impaired users to create profile using voiceover command. The “Add” option allows user to enlist a new item with associated parameters with the help of a beep sound and speech feedback. For example, the system prompts
the user to “speak the name of the item after beep,” then the name is recorded. O’Map suggests to add multiple images covering all views (see Fig.4) of an item to get robust recognition irrespective of the user’s searching direction. However, the symmetric view does not contribute to improve matching accuracy and waste storage. So, the redundant images of symmetric views are eliminated by comparing color histograms. Fig.5 shows two symmetric views of a water bottle and their color histograms. When the color histogram of a new view matches with any of the existing ones by 75%, it is discarded.

Fig. 4. Views of a sample item.

Fig. 5. Symmetric view detection. Bins are in x-axis, number of pixels in y-axis.
When the server receives a request for adding a new item, it creates a new group to reduce search space. In order to build robust and invariant recognition system, SURF [1] (Speeded Up Robust Features) features and descriptors are extracted from item images. SURF descriptor is a 128-dimensional vector consisting of sixteen eight-bin weighted histograms of gradient orientations. A K-D tree [2] is built from the feature descriptor to optimize matching time (O (log n)). A hash table with item name as key and K-D tree index as value is built and saved in persistent storage to eliminate pre-calculation time during search period.

**Implementation Details**

The O'Map consists of two main modules: 1) Client-Interface that handles data acquisition, communication, and feedback and 2) Server-Object recognition engine (see Fig. 6).

![Architecture of Search Module](image_url)

Fig. 6. Architecture of Search Module.
In data acquisition module, to collect input parameter efficiently some utility features such as automated detection of lighting condition, guidance to create panorama exploiting compass and rotational information, has been integrated. Google Glass requires customization to implement some of these features. The rationality of including these features is discussed in detail below.

**Checking Light**

The lighting condition of surroundings needs to be sufficient to analyze captured images. Since it is very difficult for visually impaired people to infer the lighting condition, the system automatically examines it by calculating color histogram. Fig. 7 shows the images of a dark and well-lit room and their corresponding intensity histogram. In the dark image histogram, first few bin contains most of the pixels, while for the lighted image, pixels are distributed all over the histogram. So we applied this simple logic, if the first five bins of histogram contain 80% of pixels then the user is informed about inadequate lighting condition.
Voice Enabled Interface

The accessibility issue was addressed using voiceover interaction utility from Google Speech Recognizer (see Fig.8). Only single-word input parameter is allowed to record which eliminates parsing task for raw text. Google speech recognizer sometimes generates erroneous text, especially for the non-native English speakers.

Fig. 7. Checking lighting condition.
Object Map

The object map, which contains reference item’s name and corresponding clock position, of a semi-structured environment is created from panorama (see Fig.9).
The benefits of using panorama are: 1) it helps users to construct mental map of the environment; 2) solves focalization and cropping problem; 3) reduce processing time of continuous frame, if item is found in the panorama. Focalization is aiming camera in the right direction. The 360° panorama is created by stitching 18 frames which are captured in 20° apart with wide Vertical (42.6°) and Horizontal (54.8°) field of view angle of camera. The consecutive frames have large overlapping area, which generates adequate correspondence that result seamless blending of frames. However, creating a panorama becomes difficult and picture might get blur, if camera shakes abruptly or the user rotates too fast. We provide a speech feedback “too fast” when user crosses an angular velocity threshold (empirically found 20°/sec) to prevent blur in image. The rotational angle is read out 20° apart with major four directions (North, East, West and South) during panorama creation because they usually have difficulty in inferring area covered.

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wall Clock</td>
<td>12 O’Clock</td>
</tr>
<tr>
<td>Sofa</td>
<td>5 O’Clock</td>
</tr>
<tr>
<td>Dining Table</td>
<td>10 O’Clock</td>
</tr>
<tr>
<td>Piano Desk</td>
<td>3 O’Clock</td>
</tr>
<tr>
<td>Computer desk</td>
<td>2 O’Clock</td>
</tr>
</tbody>
</table>

Fig. 9. Sample panorama and Object Map.
Handling Query

When a user-request arrives with data, a multithreaded server extracts SURF descriptors from panorama and finds correspondences with reference items using FLANN [8] (Fast Library for Approximate Nearest Neighbor search) matcher. The FLANN matcher is initialized with pre-calculated K-D tree indices at the beginning and kept updated based on changes in views. The sparse object map is built based on index of matched keypoints and centroid of bounding box of reference item. If the width of panorama is \( w \) and pixel index of a keypoint or centroid is \( x \), and start position of rotation is \( S_i \), then position of that reference item, \( P_{item} \) can be calculated using following formula:

\[
P_{item}(degree) = S_i \pm \frac{x \times 360^\circ}{w}
\]

The \( \pm \) symbol is for clockwise and anticlockwise rotational direction respectively. The measured value of \( P_{item} \) is then converted in clock-orientation because visually impaired users are more comfortable with this mode of feedback. Then, it performs matching for query item and if it is found within the reference item bounding box, it generates feedback like “item on top right”, or “item on top left” etc. depending on position. The detailed steps for client-server communication and item matching are presented in Fig.10, Fig.11 and Fig.12.
Algorithm 1 (Client) Finding missing Item in semi-structure environment

**Notation:** F is a image matrix of gray level pixel value, total N number of pixels
**Input:** Item name
**Output:** Items relative position

1: Capture frame(F) on camera preview mode
2: Calculate intensity histogram (H) from captured frame
3: N ← width(F)*height(F)
4: pixelCounter ← 0
5: for i = 1 to 5 do
6: pixelCounter ← pixelCounter + H(i)
7: end for
8: if pixelCounter/N ≥ 0.8 then
9: Turn ON Torch
10: Notify user using text to speech service
11: end if
12: create panorama from frames at 20° apart
13: provide feedback about 360° coverage
14: instruct user to provide speck item name
15: start Google speech recognizer Intent
16: parse item name from voice over input
17: encode and compress input data and transmit
18: listen to port for the feedback
19: if item not found in panorama start sending individual frame.

Fig. 10. Algorithm for data acquisition module.
Algorithm 2 (Server) Finding missing item in semi-structure environment

Notation: P is a matrix with gray level pixel value of panorama, queryItem is the missing item, startPosition is angle from of rotation(clock wise=0, anti-clockwise=1), ObjectMap holds reference item position in clock-orientation. from which user start rotating to create panorama, panoRotation is direction of rotation(clock wise = 0, anti-clockwise = 1), ObjectMap holds reference item position in clock-orientation.

Input: image frame/panorama, item name, start position
Output: Items relative position in object map

1. Initialize FLANN matcher with precalculated SURF descriptor of reference items
2. listen to port for client data
3. panoramaChecked ← false; ObjectMap ← nil P ← nil; queryItem ← nil
4. startPosition ← 0; panoRotation ← 0
5. if panoramaChecked=false then
6.   P ← create a image matrix from decoded data
7.   set queryItem, startPosition, panoRotation from frame's header
8.   extract SURF features descriptors from panorama
9.   for each reference item descriptor do
10.    find correspondences with panorama
11.    find location (x) of correspondences keypoint
12.    refItemPosition ← GetPosition(width(P), startPosition, panoRotation)
13.    ObjectMap(refItem) ← refItemPosition
14.    Send speech feedback about item and position
15. end for
16. Load K-D tree index for query item
17. panoramaChecked=true
18. find correspondences between panorama descriptor and K-D tree descriptor of query item
19. if item found in panorama then
20.   provide feedback about relative position to reference item
21.   keep record in log history
22. else
23.   notify user item is not identified in panorama
24.   if history log is not empty then
25.     suggest probable area from log history
26.     provide direction from object map
27.     Notify client app to transmit individual frame
28.   else
29.     Take user preference
30. end if
31. end if
32. end if
33. else
34. if item not found & panoramaChecked=true then
35.   calculate SURF descriptor from individual frame
36.   find correspondences between frame & query item
37.   if item found in individual frame
38.     provide feedback about relative position
39.     keep record in log history
40. end if
41. end if
42. end if

Fig. 11. Algorithm for Item recognition module.
Algorithm 3 (GetPosition) Calculate position of an item

**Input:** frame width, start position, rotational direction  
**Output:** Items relative position in clock orientation

1: tmpAngle ← 0  
2: if panoRotation is clock wise then  
3:    tmpAngle ← startPosition + x*360/width(P)  
4:    tmpAngle ← tmpAngle%360  
5: else  
6:    tmpAngle ← startPosition – x*360/width(P)  
7: if tmpAngle < 0 then  
8:        tmpAngle ← tmpAngle+360  
9: end if  
10: end if  
11: convert angle to clock- orientation and return

Fig. 12. Algorithm for position calculation.

**Item Matching**

The correspondences between pre-calculated template and panorama or individual frame descriptors were established using Brute-Force search at the beginning. Later, it was optimized using K-D tree with k-nearest neighbor (KNN) search. The false and weak correspondences are eliminated by two types of ratio test: 1) distance ratio test- ratio of distance between first and second closest neighbors of template patches; 2) scale [7] ratio test- ratio of scale of two matched keypoint (shown in Fig.13 & Fig.14). From a pilot study with RGB-D Object Dataset and some custom samples, we found that 99% of false matching is removed if the threshold for distance ratio is set to 0.54. Fig.15 shows that 42 out of 4267 dominant pairs survived after the distance ratio test. We also found that, if the matching is correct, the ratio of scale for every pair of two matched key points remains almost identical. So we removed all the pairs which have a large divergence in scale-ratio from baseline, (shown in Fig.15). Further, matching performance is improved by filtering outliers using random sample consensus (RANSAC) method. The correspondences are verified using homography transformation.
Fig. 13. Sample Item Matching.

Fig. 14. Matching after ratio test.
Distance Estimation

In addition to relative position, we measured the approximate distance of the identified item from the user’s position. We have adopted two simple and computationally efficient ideas from [6, 9]. Neither approach estimates distance with high accuracy but are reliable within 4 meter. The equations for both approaches are explained in Fig.16 for the sake of clarity.
**Distance estimation using scale of keypoints:**

\[ D = \frac{W_{real} \times W_{im} \times S_{tr}}{W_{tr} \times S_{real} \times \tan(\alpha/2)} \]  

(2)

Equation 2: \( S_{tr} \) and \( S_{real} \) denotes the scale of matched keypoints of templates and recognition image; \( W_{tr} \) and \( W_{im} \) denotes the width of the object in the templates and recognition image measured in pixels, \( \alpha \) is view angle, \( D \) = Distance between object and user.

**Distance estimation using camera properties:**

\[ H_{objCMOS} = \frac{H_{CMOS} \times H_{MBR}}{H_{Frame}} \]  

(3)

\[ D_{obj} = \frac{H_{obj} \times D_{f}}{H_{objCMOS}} \]  

(4)

Equation 3 & 4: \( H_{objCMOS} \) is height of object on CMOS sensor, \( H_{CMOS} \) denotes height of CMOS sensor, \( H_{MBR} \) represents height of bounding box on query frame, \( H_{Frame} \) is height of frame, \( D_{obj} \) is distance between object and user, \( H_{obj} \) is real height of object and \( D_{f} \) is focus distance of camera.

**Quantitative Evaluation**

In order to evaluate the system objectively, we selected representative data set (RGB-D Object Data and custom samples) that account for variabilities that may occur in indoor settings. The dataset includes items with different shapes, texture, shade, surface reflectance, background clutter and occlusion. We formulated some case studies and each of the cases has been
thoroughly examined with 30 indoor items. Those cases are: 1) Reduction of matching time-
Matching time is reduced by 24% using k-d tree over Brute-Force (Table 5). 2) Finding Robust
Correspondences- The system performance is improved by the distance and scale ratio test,
which is shown in two ROC curve (Fig.17).

Table 5. Matching time optimization using K-D tree over Brute-Force

<table>
<thead>
<tr>
<th>Item Name</th>
<th>#Keypoint in Template</th>
<th>#Keypoint in Panorama</th>
<th>#Matches using K-D</th>
<th>#Matches using B-F</th>
<th>Time Gained (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jar</td>
<td>10899</td>
<td>2819</td>
<td>14</td>
<td>37</td>
<td>515.58</td>
</tr>
<tr>
<td>Wallet</td>
<td>9351</td>
<td>3568</td>
<td>55</td>
<td>86</td>
<td>438.47</td>
</tr>
<tr>
<td>Bookshelf</td>
<td>10991</td>
<td>3711</td>
<td>19</td>
<td>30</td>
<td>84.67</td>
</tr>
<tr>
<td>Key</td>
<td>3672</td>
<td>3265</td>
<td>14</td>
<td>30</td>
<td>56.05</td>
</tr>
<tr>
<td>Coffee mug</td>
<td>10241</td>
<td>3745</td>
<td>40</td>
<td>124</td>
<td>1090.4</td>
</tr>
</tbody>
</table>

Fig. 17. ROC of system performance. a) before scale ratio test b) after scale ratio test.
The average F-measure before and after scale ratio test is 0.9362 and 0.9650, respectively. The overall recognition accuracy is 95%, which outperforms the system proposed by [3, 10]. Two ROC curves views that area under curve is 0.987 and 0.989, which are near perfect. 3) Correctness of object map- The object map was created and evaluated from 50 panoramas, which contained three reference items (table, bookshelf, computer desk). The system measured the items location with 3.01° Root Mean Square Error (RMSE). In Fig.18 correctness of the object map is shown for two sample items. Panoramas were captured from arbitrary position and direction. The actual angle and distance between items and the user were measured using smartphone compass and meter scale. We can see from the graph that the calculated values using two approaches are very close to actual one. 4) Feedback Time- The round trip feedback time after panorama creation is 4-6 second when item is found in the panorama. The used server configuration is: CPU 2.53 GHz, Memory 6 GB, 64 bit windows 7 and during matching it used 60-84% CPU.
Fig. 18. Correctness of Object Map (top: computer desk; bottom: table).

**Discussion**

We started designing O'Map by collaborating with a group of visually impaired people at Mid-South Access Center for Technology (Mid-South ACT). Since visually impaired people are included in design cycle to assess usability, accessibility and usefulness of O'Map and the system performs robustly, we believe that the indoor environment will be more accessible using O'Map. During the development cycle, we discovered and handled some challenges such as designing interactive interface to create personal profile, providing proper guidance and feedback. The participatory design revealed that speech feedback is more preferable than sonification or vibration because it requires no training to understand.
Conclusion

We present a new approach to find and locate missing items. The key contributions of this work are: 1) integration of object map to help visually impaired people to create a mental map of the environment; 2) solving focalization problem by panorama; 3) improvement in template matching using scale ratio test to build recognition engine on low form factor devices. Although the prototype system is successful, yet there are many improvements necessary to be meaningful for a wide range of users. One of the participants suggested adding navigation and way finding utility to this system. We also have plans to conduct large-scale usability study using subjects with various degrees of disability. However, our work is not beyond limitations. One of the participants said that, sometimes the visually impaired people don’t receive assistance from sighted individuals. So, in those scenarios, personal profile creation becomes difficult.
Works Cited


