Active Object Search with a Mobile Device for People with Visual Impairments

Jacobus C. Lock[1], Grzegorz Cielniak[1], Nicola Bellotto[1]
[1]*Lincoln Centre for Autonomous Systems (L-CAS), University of Lincoln, Lincoln, UK*
{*jlock, gcielniak, nbellotto*}*@lincoln.ac.uk*

Keywords: Active vision, object search, visual impairment, Markov Decision Process

Abstract: Modern smartphones can provide a multitude of services to assist people with visual impairments, and their cameras in particular can be useful for assisting with tasks, such as reading signs or searching for objects in unknown environments. Previous research has looked at ways to solve these problems by processing the camera's video feed, but very little work has been done in *actively* guiding the user towards specific points of interest, maximising the effectiveness of the underlying visual algorithms. In this paper, we propose a control algorithm based on a Markov Decision Process that uses a smartphone's camera to generate real-time instructions to guide a user towards a target object. The solution is part of a more general active vision application for people with visual impairments. An initial implementation of the system on a smartphone was experimentally evaluated with participants with healthy eyesight to determine the performance of the control algorithm. The results show the effectiveness of our solution and its potential application to help people with visual impairments find objects in unknown environments.

1 INTRODUCTION

It is estimated that almost half a billion people worldwide live with mild to severe visual impairments or total blindness (Bourne et al., 2017) and significant effort is being made to enable these people to lead more independent lives. Modern improvements in mobile computing power and image processing techniques have provided researchers with new and powerful tools to solve this problem. The work presented here is part of a project to assist people with visual impairments to navigate and find objects in unknown environments with the aid of a smartphone. The proposed system implements ideas from the field of active vision (Bajcsy et al., 2017), but replaces the typical electro-mechanical actuators of a moving camera with the body (i.e. arm, hand) of the user holding the smartphone, as pictured in Figure 1, expanding upon concepts originally proposed in (Bellotto, 2013) and (Lock et al., 2017).

The goal of our active search system is to understand the user's surroundings and determine what the next best course of action is to reach the target object based on what is currently within view and what has been observed in the past. To this end, we implemented a smartphone guidance system based on a Markov Decision Process (MDP) (Bellman, 1957)

Figure 1: The system in use during an experiment.

that generates, in real-time, a series of instructions for the user to point to the target, depending on a previously-learned spatial distribution of known objects and on the camera's current view. This work includes three main contributions:

- an MDP-based human controller that can guide a user in a visual search task;
- a data-based transition model for the MDP which includes spatial relations between known objects;
- a set of user experiments that prove the effectiveness of our active search implementation.

Section 2 discusses other relevant work done in this field, followed by a general explanation of the active vision system for human guidance in Section 3, and a detailed explanation of the human-control module in Section 4. The experimental results are presented in Section 5, after which we conclude the paper and discuss future work in Section 6.

2 PREVIOUS WORK

Assistive technology for people living with visual impairments is a growing research area (Manduchi and Coughlan, 2012; Khoo and Zhu, 2016). In recent years, the increase in mobile processing power and computer vision improvements have led to research in the use of smartphone cameras to augment or enhance a user's vision and help them find objects or other points of interest. Earlier attempts at the problem involved placing special markers or barcodes around an environment, which the user then scans with a smartphone or similar mobile device (Gude et al., 2013; Iannizzotto et al., 2005; Manduchi, 2012). This device then uses some feedback mode, e.g. Braille or sound, to guide the user towards the target.

Another approach is to discard tags completely and rely on computer vision to perform the object detection, something that has become more practical with recent improvements to feature detectors and deep networks (Huang et al., 2017; Redmon et al., 2016). SIFT and SURF-based object detectors have also been used to detect known objects, when they are in the camera's view, and to guide the user to them using sonified instructions (Schauerte et al., 2012). These type of systems is more flexible than the tag-based ones, but it has the same drawback of being *passive*, in the sense that it relies on having the object within the camera's view in the first place. Also, no clear performance metrics are reported in the previous paper. The VizWiz system (Bigham et al., 2010) offloads the object recognition tasks to an Amazon Mechanical Turk worker who then provides feedback on where the object of interest is located relative to the user. The VizWiz has the advantage of being fairly robust and is able to classify a great deal of objects with little effort from the user and can provide natural, human-generated and curated directions. However, this approach does not enhance user independence, since a person with visual impairments is now beholden to an online worker instead of a relative, friend or bystander. Furthermore, a good internet connection is required on the device, possibly limiting its use in some poor-reception areas.

Previous researchers have implemented active search and perception strategies in robots and image classifiers (Bajcsy et al., 2017) in an attempt to optimise their classification and planning tasks, for example by exploiting the structured nature of human environments and object placements. Two research teams have recently implemented an active object search strategy into their image classifiers (Caicedo and Lazebnik, 2015; Gonzalez-Garcia et al., 2015). Their approaches use different methods but conceptually similar models to generate windows of interest for visual classification. The size and locations of the windows within the image are generated using the spatial relationship between objects, taken from the SUNCG and PASCAL datasets (Song et al., 2017; Everingham et al., 2010), and are iteratively changed based on the output from the respective models. The advantage of their approaches is that fewer windows are generated and submitted to the classifier, resulting in lower object classification times while still keeping state-of-the-art results for accuracy.

Similar strategies have been incorporated on robotic platforms to improve autonomous object search, manipulation and localisation tasks. For example, some researchers have developed a planning algorithm for a robotic manipulator that performs an optimal object search in a cluttered environment (Dogar et al., 2014). Another team implemented an MDP generating an optimal object search strategy in a room over a belief state of object positions and configurations (Aydemir et al., 2011). However, the authors trained their MDP using a custom object-placement and configuration scenario, so their results are sensitive to changes within this distribution.

In summary, much research has been conducted on recognition of and guidance towards target objects, including active vision solutions for image classifiers and robotic systems. However, to our knowledge, no previous work has been done on *active* object search and guidance for humans, which would especially benefit people with visual impairments. In this paper, we implement such an active vision system with a human in the loop that guides the user towards an out-of-view target object. Our system exploits prior knowledge of the objects spatial distribution within an indoor environment, learned from a dataset of real-world images, and the history of past object observations made during the search.

3 ACTIVE VISION SYSTEM

The work presented in this paper is a fundamental step towards a more general project's goal to develop a stand-alone system that can guide a person with vi-

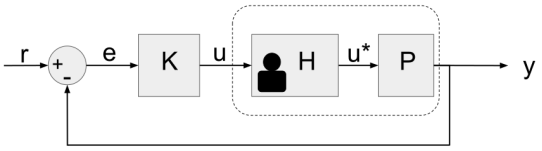

Figure 2: System control loop: r is the reference object, e the error signal, u and u^* the original and interpreted control signals and y is the current object observation. K, H and P are the control, human and sensor blocks respectively.

sual impairments to his/her destination with minimal user input or intervention. A complete system diagram is given is Figure 2. This closed-loop system is conceptually similar to other classical control problems, where the difference between desired and actual state of a process is used to generate a control signal that changes the process itself.

In this case, the reference signal, r, is the object the user wishes to capture with the smartphone's camera. The goal of the control block, K, is to generate human interpretable instructions, u, to guide the user towards the target object. The process to be controlled involves a human, H, who interprets the instruction and executes a physical action, u^*, to actually manipulate the smartphone's camera, P. A new observation, y, with the camera is then fed back to the loop and the error signal, e, is updated accordingly.

Here we focus in particular on the implementation of the control module K. Two important points are considered in the design of the controller. Firstly, K must be scenario-agnostic, meaning that objects could be placed in different places with unknown a priori information. Secondly, since each person could interpret the instruction u differently (i.e. different transformation block H), the controller must be robust enough to handle such incorrect interpretations. For example, one person might interpret and execute an 'UP' instruction correctly (i.e. $u \backsimeq u^*$), while another might interpret it correctly, but execute the wrong action. This risk can be mitigated by the use of clear and simple instructions that helps u^* be as close as possible to u. The implementation of this controller is discussed in detail next.

4 HUMAN-CONTROL MODULE

Our active search system guides the user by generating a set of waypoints that need to be observed by the camera, tracing a path that will eventually lead to the target object. Note that the actual location of the latter is *unknown*, meaning that the system will guide the user towards the most likely location where the object might be found, based on its internal knowledge of spatial relations between objects (e.g. a computer monitor is more likely to be above than below a desk). This path is generated one waypoint at a time and is updated with every new object observation captured by the camera, or after a re-orientation of the latter beyond a certain angle. We tackle the problem using an MDP, the design and implementation of which are discussed in the following sub-sections.

4.1 MDP for Human Control

An MDP produces a policy of optimal actions for an agent to take in any given pre-defined state. In this case, the agent is defined as the guidance system and the policy is used to generate the next waypoint on the search path towards the target object. We assume fully observable states and known state transitions probabilities. The MDP is represented by the 5-tuple

$$(\mathbf{S}, \mathbf{A}, \mathbf{T}, \mathbf{R}, \gamma), \tag{1}$$

where $\mathbf{S}$ is a set of possible agent' states, $\mathbf{A}$ is a set of possible actions the agent can take in any given state, $\mathbf{T}$ is a set of state transition probabilities from state s to state s', with $s, s' \in \mathbf{S}$, and $\mathbf{R}$ is the reward the agent receives for reaching state s' after executing action $a \in \mathbf{A}$ in state s. The scalar γ is a discount factor that prioritises immediate over long-term rewards and which affects the model's convergence rate (Russell and Norvig, 2009). Each of these elements are defined and discussed next.

4.1.1 States

The state is a combination of parameters that defines the agent's world and decision process. Our state vector is defined as

$$s = \langle o, n, v \rangle, \tag{2}$$

where o is the current object in viewed by the camera, n is the number of steps taken since the search started, and v is a binary variable that keeps track of whether a waypoint for the current state was already generated during the current object search.

4.1.2 Actions

The policy produced by an MDP defines the action the agent will take when it finds itself in any given state. In this case, the action is the direction of the next waypoint relatively to the current device's pose. The possible actions are given by

$$\mathbf{A} = \{UP, DOWN, LEFT, RIGHT\}. \tag{3}$$

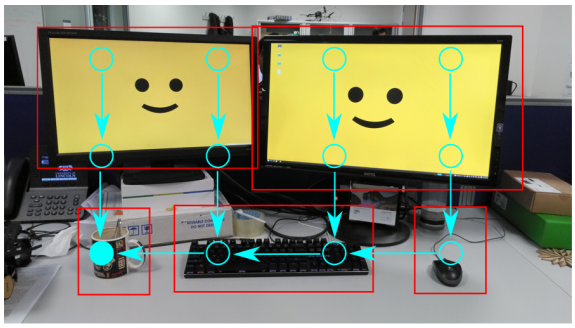

Figure 3: Example of action policies generated by the MDP to guide the user in pointing the camera from a random starting object (e.g. monitor) to a target object (e.g. mug).

To illustrate an example of actions sequence, let us consider the scene in Figure 3, which contains a number of simple, distinct objects (red boxes). The MDP guides the user in pointing the camera to the target object (the mug at the bottom-left of the figure). It does this by inferring the current state, which depends on the object currently observed by the camera, and generating an action, or instruction, that leads the user to the target object. An action is considered completed, and therefore a new state reached, when the camera has rotated more than a predefined angle or a new object is detected.

4.1.3 State Transition Probabilities

The state transition **T** defines the probability of the agent switching from state s to state s' due to action a, i.e. the probability of observing object o' after object o due to a pan/tilt rotation of the camera. Therefore, **T** represents the spatial relationships between the different objects in our environment model. These spatial relationships are learned from a dataset during an initial training process, which is discussed more in detail in Section 4.2.1.

4.1.4 Reward Function

The reward **R** is the immediate reward that the agent receives after transitioning from state s to state s'. The goal of the agent is to maximise its cumulative reward and it is very important to fine-tune **R** correctly for producing an effective action policy. In order to encourage the agent to find the target object as fast as possible, a relatively large positive reward should be assigned for successfully reaching the goal state, and a negative one in any other case. These parameters must be finely balanced to ensure effective object search behaviour.

4.2 System Implementation

This section describe the actual implementation details of the MDP for active object search, including initial training and software deployment on our smartphone device.

4.2.1 MDP Training

A policy that defines the optimal action for an agent to take for any given state is generated through a training process that involves letting the agent explore the entire state-space and iteratively improve its decision function, i.e. policy, in order to reach the target state in a way that maximises its cumulative reward. This method, called Q-learning (Watkins and Dayan, 1992), does not require a model of the agent's environment during training, allowing the policy to be used in many different scenarios.

Currently there are 7 objects encoded into the system, plus a 'nothing' instance where nothing of note is observed. Our initial implementation considers a simple office desk scenario containing the following objects:

$$o \in \mathbf{O} = \{monitor, mouse, keyboard, window, mug, stationery, desk, nothing\}. \quad (4)$$

The spatial relationships between the objects in **O** are extracted from the OpenImage dataset (Krasin et al., 2017), which consists of 1.74M images containing 14.6M manually drawn and labelled bounding boxes around objects (see Figure 4 for some examples). The dataset is primarily aimed toward object recognition researchers to benchmark their models. In our case though, the bounding boxes and object labels are used to extract the spatial relationships between the different objects in **O**. Since the camera perspectives and absolute distances between the objects in the images are not given, we can only extract the relationships in the basic action terms specified in **A**, e.g. we can only say that object 1 is above object 2, but not how far above. Our relatively simple action-space is therefore suitable for the limited dataset information.

Figure 5 shows the spatial relationship between a subset of **O** (desk, keyboard and mouse). For example, when the agent is in state $s = \langle o = mouse, n, v \rangle$ and is searching for the object $o_{target} = keyboard$, there is a strong probability that the target object is on the mouse's $LEFT$. The MDP of course will consider all of the objects' spatial relationships when generating the optimal policy.

The agent's target state is then any state where $s = \langle o = o_{target}, n, v \rangle$. This gives a total of 14 terminal states (7×2) per policy, since the target object can

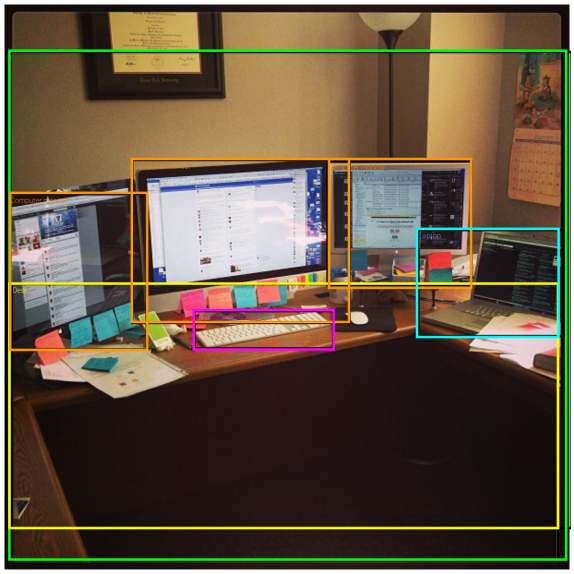

Figure 4: Examples of images from the OpenImage dataset containing objects from our set **O** (Krasin et al., 2017).

be found at any point in the search or in a position that was previously explored by the user. Each target object has its own unique policy file.

The reward function was hand-crafted and the parameters were empirically selected. The function values can be found in Table 1. The reward punishes the agent for every step it takes without finding the target object. The reward becomes increasingly negatives as the agent progresses without finding the target ($n > n_{\max}$) or when it generates the same waypoint more than once ($v = true$) during the same search.

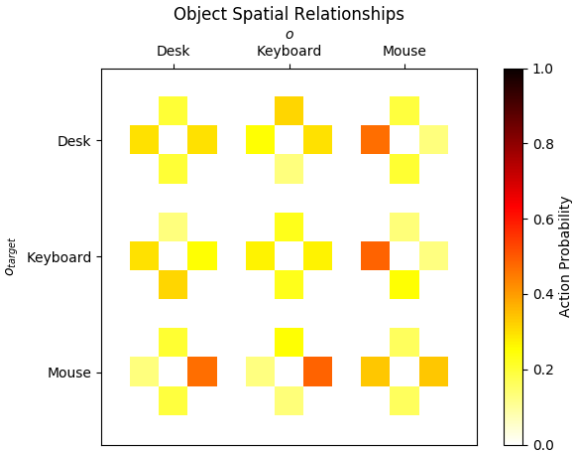

Figure 5: Examples of the spatial relationships between the desk, keyboard and mouse objects. Each square corresponds to the probability of executing an action (top square for UP, left square for $LEFT$, etc.)

Table 1: The reward functions for the MDP.

$r(o = o_{target})$	10000
$r(v = true)$	-10
$r(n > n_{\max})$	-10
otherwise $r(\cdot)$	-1

Conversely, it gives a significant positive reward when the target object is found.

We force the MDP to generate a maximum of 11 (inclusive) steps to the target, with 11 being the longest possible route on the action grid (more details about the grid are in Section 4.2.2). A search could take longer than 11 steps, but the MDP considers that the maximum, which is convenient for keeping a manageable state-space and a simple reward function. The MDP therefore has a total of 154 reachable states ($s_{tot} = 11 \times 7 \times 2$).

The lack of absolute spatial information in the OpenImage dataset generates ambiguities, which makes it hard for the model to converge to a single, optimal solution. We therefore opted to use the more conservative state-action-reward-state-action (SARSA) algorithm (Rummery and Niranjan, 1994). SARSA is an on-policy algorithm that allows us to control the level of exploration vs. exploitation that makes it easier to find a solution, although this is not guaranteed to be optimal.

The MDP is trained until it converges to the optimal policy, or for a total of approximately 17 million episodes. The parameter α, which controls the exploration vs. exploitation behaviour during training, maximises the exploration when set to 1 and the exploitation when 0. We therefore set α to be a function

of the training episodes, starting with a high exploration value and exponentially changing to exploitation as the training progresses:

$$\alpha = \exp\left(\frac{-i}{10\ s_{tot}}\right) - 0.001, \qquad (5)$$

where i is the episode index. The discount factor γ is set to 0.95 to prioritise long-term rewards and guarantee convergence.

Our MDP has a relatively small state-action space. Therefore, a solution can be found within a reasonable amount of time. However, it should be noted that adjusting the angle interval, or adding more actions or objects, can easily lead to an intractable space size.

4.2.2 Waypoints Generation

The system uses a 6×6 discretised radial grid to simplify the tracking and waypoint generation processes. The grid spans 120° in both the pan and tilt dimensions, giving a resolution of 20° per grid cell. A policy action is converted by the system into a new search waypoint centred on a cell of the radial grid, e.g. an '*UP*' action will generate a waypoint one grid cell above the camera's current orientation. Note that this cell is not part of the MDP' state and the radial grid is only used to discretise the pan-tilt movements of the camera and to guarantee a minimum angular variation between subsequent actions.

The system uses the waypoint's location to provide the user with guidance instructions (i.e. u in Figure 2). The policy actions, and waypoints by extension, are relative to the current camera's pan-tilt orientation. The grid is also wrapped so, if the location of a waypoint exceeds the 120° limit, the same waypoint is moved to the opposite side of the grid, effectively limiting the search space to a 120°×120° area.

4.2.3 Smartphone Application

We incorporated the trained system into an app (see Figure 6) for an Asus ZenPhone AR smartphone, running Android 7.0, with Google's augmented reality toolkit (ARCore), which provides the necessary 3D pose of the device. No further software or hardware modifications were required. This app is responsible for generating the guidance instructions and tracking the camera sensor (K and P blocks in Figure 2) throughout a search session. Tracking the camera's pose allows the app to infer the current state and choose the optimal action to take next.

The system determines the state values for n (number of search steps so far) and v (waypoint already visited or not) described in Section 4.1.1, by recording the previous positions and waypoint locations. The camera provides the ID of the object currently within view, which is assigned to the state variable o. In the current implementation, we did not use a real object detector, but we simulated it with 7 different QR codes, one for each unique object, and a camera-based QR code scanner from Android's machine learning API (MLKit). This simplification guarantees full observability of the state and let us focus on the performance of the MDP-based controller in the following experiments. Moreover, to speed up the image processing and avoid scanning multiple QR codes, we only used the central part of the camera's frame, which is 300×300 pixels. This choice also defines the precision required in pointing the camera towards the object (see Figure 6).

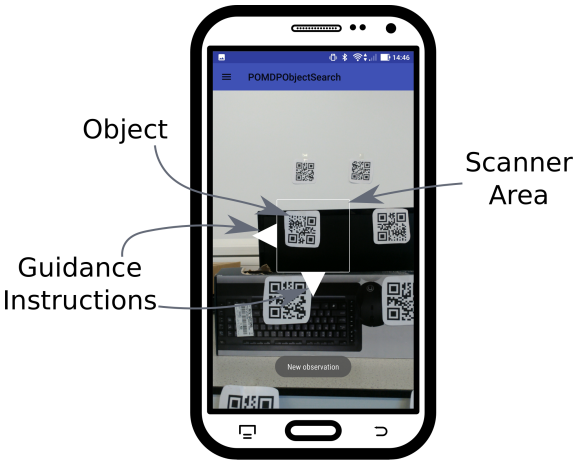

Figure 6: A screenshot of the smartphone interface showing an example of guidance instruction (down-left in this case) towards a waypoint and the QR-object scanner area.

In a real application for people with visual impairments, the position of the waypoint would be given to the user by a set of audio or vibrotactile instructions. However, since we are mainly interested in evaluating the control algorithm of our system and not the interface (K and not u), our current prototype generates guidance instructions with four on-screen arrows (see Figure 6). Obviously, this visual interface is only used for debugging and experimental evaluation of the controller, and it will be replaced by an opportune audio interface, e.g. (Bellotto, 2013), at a second stage.

5 EXPERIMENTS

To evaluate our system, we designed a set of experiments that determined how effective the MDP and its policies are at guiding the user in an object search task with the smartphone's camera. Since the focus of

Figure 7: A snapshot of the environment used for the experiments. Each QR code represents an object.

this work is on the algorithm for active object search, and not on the actual human performance, in the following experiments the system was tested by participants without any significant visual impairment. As explained in the previous section, this simplified the experimental design, allowing us to use the smartphone's display for the guidance instructions.

5.1 Experimental Design

For the experiment, the MDP policies were integrated into an Android application that uses the camera to provide observation data and track the pose and viewing direction. Guidance instructions towards the waypoints were visualised on the screen, which the participants were allowed to use. The experimental environment mimicked a typical office desk layout and contained 7 different objects (i.e. encoded QR codes), one of which could be selected for each experiment run. See Figure 7 for a snapshot of the environment.

For each experiment, the participant was placed approximately at 1m from the closest barcodes and was asked to remain on that spot during the experiment. The participant started by pressing a button on the app, which randomly selected a target object and then guided the user towards it. Since the participants were allowed to use the smartphone's display, the target was randomly selected by the app without informing them, at least until it was found. This prevented the participants from learning the target objects' locations between subsequent runs of the experiment.

To avoid pointing at uncluttered edges of the search space, where the system had difficulty guiding the user back to the centre, we set a search step-limit of 15, which means the search was terminated when the number of waypoints generated by the system exceeded 15. A search run therefore ended when the participant either successfully found the target object by pointing the phone camera to it and scanning the barcode, or exceeded the waypoint limit. After this, the participant then restarted from the central position, generating a new random target object and repeating the experiment.

Table 2: Results for the TAR, number of steps and time to target means and standard deviations for each participant.

Participant	TAR [%]	Num. Steps	Time [s]
s1	94	7.2 ± 5.4	29 ± 22
s2	79	6.7 ± 4.8	34 ± 5.1
s3	91	6.3 ± 4.9	31 ± 21
s4	79	6.7 ± 4.3	37 ± 5.6
s5	76	7.2 ± 4.9	33 ± 14
s6	60	8.2 ± 5.4	24 ± 10
s7	86	8.5 ± 5.8	31 ± 16
s8	88	5.1 ± 4.0	39 ± 21
s9	98	7.2 ± 5.4	39 ± 18
s10	67	6.6 ± 5.4	26 ± 12

We recorded 10 searches per object, giving us a total number of 70 search samples per participant. The system was tested by 10 participants, mostly from our research group. Our final dataset consisted therefore of 700 search samples, which are analysed next.

5.2 Results

We identified 3 different measures to evaluate the system's performance: target acquisition rate (TAR), number of steps to the target and the total time it took to find a target object. We present and discuss the results for each of these parameters in the next sections. The results for each individual participant are presented in Table 2, while those across all the participants are shown in Table 3. To provide a baseline measure for the ideal case, we ran a number of simulations in an environment mimicking the experiment setup with a "virtual" user who perfectly executes the policy, i.e. $u = u^*$. The TAR and steps to target results for the simulation are included in Table 3.

Table 3: The means and standard deviations for the entire participant population for the experiments and simulations.

	Experiments	Simulations
TAR [%]	82 ± 11	99.7
Num. Steps	6.8 ± 5.1	5.4
Time [s]	34 ± 23	–

5.2.1 Target Acquisition Rate

The TAR is a measure of how successful the system was at directing a participant to the target object within our 15-step limit. It is simply calculated as a ratio between the number of completed searches vs. the total number of searches. Please note, however, that this ratio depends also on the step-limit and should not be taken as an absolute measure of performance (i.e. if the step limit was much bigger, the TAR would tend to 100%). Figure 8 shows the TAR as a function of the step-limit and it shows that there is a gradual increase in the TAR as the step limit increases, but tapers off as the step-limit increases.

Table 2 shows a fairly consistent spread for the TAR across the participants. The inter-participant spread in Table 3 ($\sigma = 11\%$) is fairly significant, perhaps indicating that the user's search behaviour and strategy affects the target acquisition performance, but with an average TAR of 82%, it is clear that the system successfully finds the target object during the vast majority of searches.

Figure 9 shows the TAR for each object in our set **O**. There are TAR variations for the different objects, with the smaller objects typically being the hardest to find. However, the differences are not extreme and indicate that all the objects in **O** are roughly equally hard to locate. This is also displayed in the simulation's TAR in Table 3, which could not achieve 100% because of the difficulty the agent had in finding the objects on the fringes of the environment.

Failure cases were typically caused by the system entering a no-recovery state where the user was directed into dead-space with no spatial information (e.g. ceiling or wall section). In this case the system could not observe useful clues to intelligently guide the user. Possible improvements for future versions of the algorithm would be to implement some fall-back method that can detect a no-recovery state (e.g. exceeding a set number of steps/time without any new object observation) and guide the user back to a position where to restart the search.

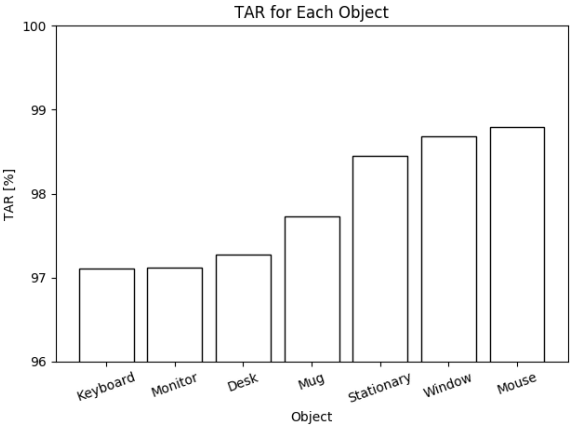

Figure 9: The TAR for each of the objects within **O**.

5.2.2 Number of Steps to Target

The number of steps to the target indicates the number of waypoints the system generated for the participant during the guidance process. This is a good indication of system performance, where less waypoints means faster target acquisition and therefore better object search strategy. Figure 10 shows the cumulative distribution of the number of steps to the target for all the participants.

The number of waypoints each participant required is fairly evenly spread across all of the participants, with the majority of searches ending within

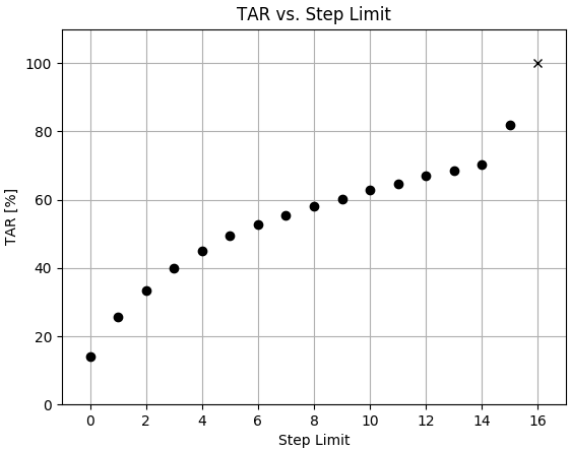

Figure 8: TAR as a function of the step limit for a search. The 'x' indicates the cases that exceeded the 15-step limit.

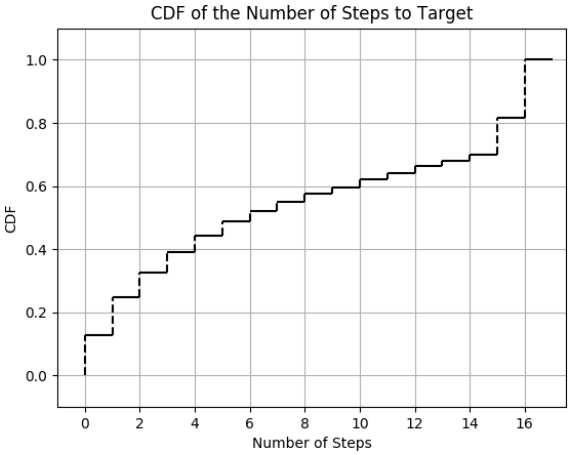

Figure 10: The cumulative distribution of the participants' number of steps taken to find a target object.

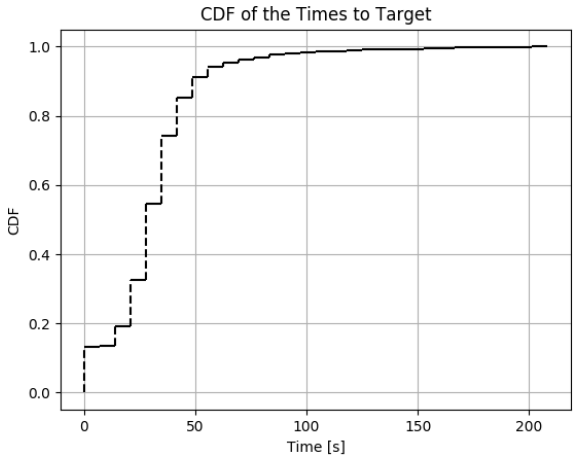

Figure 11: The cumulative distribution of the participants' time taken to find a target object.

a few search steps. The population mean and standard deviation is 6.8 and 5.1 waypoints, respectively. This is a reasonable result, since most target objects were placed within approximately 4 grid squares away from the participants' initial looking directions. The relatively high standard deviation is due to the aforementioned no-recovery states and could be reduced by opportune mitigation strategies to avoid uninformative areas.

5.2.3 Time to Target

The cumulative distribution of the time it took the participants to reach the target object is given in Figure 11. We see that the distribution is heavily skewed to the bottom with a long tail. The mean and standard deviation of the data is 34s and 23s respectively.

In comparison to the remotely-assisted VizWiz system (Bigham et al., 2010) covered in the related work (mean 92s, standard deviation 37.7s), our results look very encouraging, although there might be variations in case of participants with visual impairments.

6 CONCLUSIONS

In this work we presented and tested an MDP-based system to guide a person with visual impairments towards a target object with no prior knowledge of the environment. We implemented the system in an Android app and tested it with sighted users to determine the effectiveness of the active object search algorithm. We found that it works generally well, even when compared to alternative human-guided systems. However, the solution can be improved by refining the search strategy and implementing an automatic fail-state recovery when the user points to an empty scenario, like a blank wall. Furthermore, a purpose-built dataset with clear object spatial relations would enable the creation of a more accurate transition model for the MDP controller.

Future work will include the replacement of the QR codes with a real vision-based object detector, possibly extending the number of items and actions. However, in order to consider the uncertainty introduced by such detector, the MDP will have to be replaced by a Partially Observable MDP (POMDP), taking into account that the objects are not perfectly observable and humans might not follow the guidance instructions accurately. Further directions of research include on-line learning techniques for model adaptation that better follow the user profile of each individual and the possible performance change over time.

REFERENCES

Aydemir, A., Sjöö, K., Folkesson, J., Pronobis, A., and Jensfelt, P. (2011). Search in the real world: Active visual object search based on spatial relations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2818–2824. IEEE.

Bajcsy, R., Aloimonos, Y., and Tsotsos, J. K. (2017). Revisiting active perception. *Autonomous Robots*, pages 1–20.

Bellman, R. (1957). A markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684.

Bellotto, N. (2013). A Multimodal Smartphone Interface for Active Perception by Visually Impaired. In *IEEE SMC International Workshop on Human Machine Systems, Cyborgs and Enhancing Devices (HUMASCEND)*.

Bigham, J. P., Jayant, C., Miller, A., White, B., and Yeh, T. (2010). Vizwiz:: Locateit-enabling blind people to locate objects in their environment. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 65–72. IEEE.

Bourne, R. R., Flaxman, S. R., Braithwaite, T., Cicinelli, M. V., Das, A., Jonas, J. B., Keeffe, J., Kempen, J. H., Leasher, J., Limburg, H., et al. (2017). Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis. *The Lancet Global Health*, 5(9):e888–e897.

Caicedo, J. C. and Lazebnik, S. (2015). Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2488–2496.

Dogar, M. R., Koval, M. C., Tallavajhula, A., and Srinivasa, S. S. (2014). Object search by manipulation. *Autonomous Robots*, 36(1-2):153–167.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338.

Gonzalez-Garcia, A., Vezhnevets, A., and Ferrari, V. (2015). An active search strategy for efficient object class detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3022–3031.

Gude, R., Østerby, M., and Soltveit, S. (2013). Blind navigation and object recognition. *Laboratory for Computational Stochastics, University of Aarhus, Denmark*.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*, volume 4.

Iannizzotto, G., Costanzo, C., Lanzafame, P., and La Rosa, F. (2005). Badge3d for visually impaired. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pages 29–29. IEEE.

Khoo, W. L. and Zhu, Z. (2016). Multimodal and alternative perception for the visually impaired: a survey. *Journal of Assistive Technologies*, 10(1):11–26.

Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Kamali, S., Malloci, M., Pont-Tuset, J., Veit, A., Belongie, S., Gomes, V., Gupta, A., Sun, C., Chechik, G., Cai, D., Feng, Z., Narayanan, D., and Murphy, K. (2017). Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*.

Lock, J., Cielniak, G., and Bellotto, N. (2017). Portable navigations system with adaptive multimodal interface for the blind. AAAI Spring Symposium.

Manduchi, R. (2012). Mobile vision as assistive technology for the blind: An experimental study. In *International Conference on Computers for Handicapped Persons*, pages 9–16. Springer.

Manduchi, R. and Coughlan, J. (2012). (Computer) Vision Without Sight. *Communications of the ACM*, 55(1):96–104.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, England.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Schauerte, B., Martinez, M., Constantinescu, A., and Stiefelhagen, R. (2012). An assistive vision system for the blind that helps find lost things. In *International Conference on Computers for Handicapped Persons*, pages 566–572. Springer.

Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., and Funkhouser, T. (2017). Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition*.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.