ActiVis: Mobile Object Detection and Active Guidance for People with Visual Impairments

J.C. Lock[1], A.G. Tramontano[2], S. Ghidoni[2], and N. Bellotto[1]

[1] School of Computer Science, University of Lincoln, United Kingdom
[2] Department of Information Engineering, University of Padova, Italy

Abstract. The ActiVis project aims to deliver a mobile system that is able to guide a person with visual impairments towards a target object or area in an unknown indoor environment. For this, it uses new developments in object detection, mobile computing, action generation and human-computer interfacing to interpret the user's surroundings and present effective guidance directions. Our approach to direction generation uses a Partially Observable Markov Decision Process (POMDP) to track the system's state and output the optimal location to be investigated. This system includes an object detector and an audio-based guidance interface to provide a complete active search pipeline. The ActiVis system was evaluated in a set of experiments showing better performance than a simpler unguided case.

Keywords: Active vision, vision impairment, object detection

1 Introduction

There are an estimated half a billion people that live with mild to severe vision impairment or total blindness and this number is expected to significantly rise with an ageing population [7]. There has been a rise interest from industrial partners in utilising modern technology to make their products more accessible, and improvements in modern computing power and image processing capabilities have made this easier. This work is part of the ActiVis[3] project, which aims to enable people with vision impairments to independently navigate and find objects within an unknown indoor environment using only a mobile phone and its camera. Our solution is inspired by active vision research [3], but it replaces the electro-mechanical servo typically found in active vision systems with a user's arm and hand, as pictured in Fig. 1. This paper expands upon concepts proposed in [4, 14] and extends the concept presented in [15] with a fully working system.

ActiVis uses the camera's current and previous image data as input and leverages its understanding of inter-object spatial relationships to determine the best navigation action to find the target object. For this, we expanded upon our previous work and implemented a Partially Observable Markov Decision Process (POMDP) on a mobile phone that generates real-time navigation instructions for the user. The current paper includes the following main contributions:

[3] http://lcas.github.io/ActiVis

Fig. 1. The system in use during an experiment with a blindfolded participant.

- a new controller that enables object search and guidance on a mobile phone;
- a complete system pipeline that includes audio interface, object detection and human control;
- experiments that evaluate the efficacy of the proposed system.

Section 2 discusses relevant previous work, followed by a description of the active vision system and controller in Section 3. The experiments and their results are discussed in Sections 4 and 5. The paper is concluded in Section 6.

2 Previous Work

Early attempts to solve this guidance problem used markers encoded with object or environmental information and a smartphone that scans the environment for these simple patterns [9, 16]. The device uses audio feedback to read out the embedded information or guide the user towards the markers. While improvements to feature detectors have made it viable to replace markers with real objects [18], an alternative guidance approach is proposed in VizWiz [5] which uses a Mechanical Turk worker to manually locate and guide towards the desired object within a user-provided picture.

The issue with a marker-based approach is that it requires significant effort to place and maintain them in an environment, which is remedied by markerless systems. However, both of these methods use passive guidance approaches that rely on the user placing the desired marker/object within the camera's view by themselves before any guidance is provided. VizWiz [5] can leverage a human's understanding of the environment to guide a user to the correct location, but there is significant lag and a reliance on a good internet connection and remote worker being available. Previous work on the ActiVis system [15] addressed the passive guidance issue by implementing a Markov Decision Process (MDP) that gives the user pointing instructions to find an out-of-view object, showing that this is a viable method for object search. However, that work used QR-codes to simulate real objects and on-screen prompts to present the guidance instructions. In this paper we expand the control model, replacing the QR-codes with real

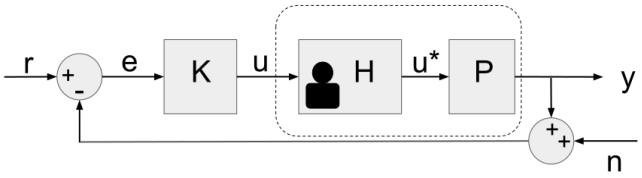

Fig. 2. The system control loop including a human and the controller.

objects and providing audio guidance, thereby implementing a complete object detection and guidance pipeline for people with vision impairments.

Object detectors can be broadly classified in two-stage models, which use an external algorithm to select regions of interest where to perform object inference [19], and single-stage models, which generate multiple windows of different sizes and checks each window for any objects. Between these two classes, there is a speed-accuracy trade-off, where the two-stage models typically produce more accurate results, but are slower and require more computing resources [12]. On the other hand, single-stage models, such as SSD and MobileNet [13, 2], produce less accurate results but require significantly less parameters and FLOPS to perform object inference than the two-stage models.

3 Active Vision System

A complete system diagram for ActiVis is given in Fig. 2. This diagram shows a typical feedback control loop that generates a control signal to minimise some error signal, e, and drive the output to some reference, r. In our case, the system incorporates a human within the loop; r is the target object and y is the actual observation of the mobile device's camera.

Adding a human into the loop requires an additional block, H, representing a human that receives a control signal, u, from the controller, K. However, the challenge is that a person may interpret u in some unpredictable way, resulting in the signal u^* that points the camera, P, to a new object observation, y. It is therefore important to design K to be robust enough to accommodate different user habits and limitations, to ensure that u^* tracks u as closely as possible. The object detector's classification error, n, is added to the feedback loop as a noise signal that affects K's output.

In our previous work [15], we assumed a perfect object classifier and solved the problem using an MDP. In this paper we remove that assumption and replace the MDP with a POMDP-based controller that can handle uncertainties in the object detection and classification output. Our new controller works by generating a trail of virtual waypoints for the user to point the camera towards, which eventually leads to the target object. The waypoint positions are based on the model's pre-trained internal knowledge of the inter-object spatial relationships and they are placed in a way that maximises the probability of the user finding the target object.

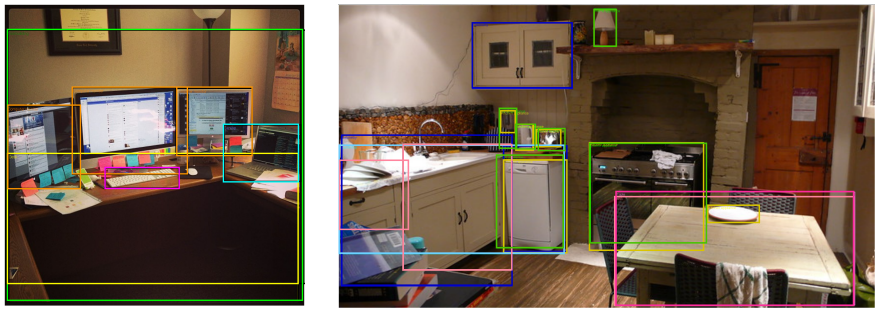

Fig. 3. Images from the OpenImage dataset [10] containing some typical objects.

3.1 Controller Design

A POMDP is an extension to a MDP that handles cases where the state is not directly observable, allowing it to be used in more realistic scenarios. The implication, however, is that a POMDP-based agent does not know its state at any point in time, but must infer it based on the known model parameters and sensor accuracy, which in our case is the object detector's accuracy. This state inference relies on a so-called *belief meta-state*, which is updated with additional observations to reflect the likelihood that the agent is in any given state. The belief state is fully observable by the agent and is used to infer the mobile device's current state and generate the next action.

A POMDP model is described by the 8-tuple $(\mathbf{S}, \mathbf{A}, \mathbf{T}, \mathbf{R}, \mathbf{\Omega}, \mathbf{O}, \mathbf{b}, \gamma)$, where $\mathbf{S}$ represents a finite set of discrete states, $\mathbf{A}$ is a set of discrete actions, $\mathbf{T}$ is a matrix containing the probabilities of transitioning from state s to state s' (where $s, s' \in \mathbf{S}$) after executing action a (where $a \in \mathbf{A}$), and $\mathbf{R}$ is the reward the agent receives for executing a and reaching s'. $\mathbf{\Omega}$ is the set of possible state observations, while the matrix $\mathbf{O}$ contains the probabilities of making observation $o \in \mathbf{\Omega}$ when in s after executing a. Finally, $\mathbf{b}$ is the belief vector containing the state probability distribution and γ is a discount factor that prioritises long-term over short-term rewards, which affects the model's convergence rate.

Parameters The state is given by $s = \langle u, n, v \rangle$, where u is the object within view, n is the number of search steps taken during the search and v is a binary variable indicating whether a waypoint has been visited before. The possible actions that dictate the location where the mobile device will generate the next waypoint are given by $\mathbf{A} = \{\text{UP, DOWN, LEFT, RIGHT}\}$. $\mathbf{T}$ was determined by extracting the inter-object spatial relationships for a limited number of objects, in terms of the actions $\mathbf{A}$, from the OpenImages [10] dataset. For example, by iterating over the images containing the objects of interest, we can see that the object 'monitor' is located above (i.e. UP) the object 'keyboard' in 16% of the images containing both objects (see Fig. 3). The transition function for this case is $t(s, a, s') = t(\text{keyboard}, \text{UP}, \text{monitor}) = 0.16$.

Table 1. The reward function for the POMDP.

Reward condition	POMDP	MDP [15]
$r(o = o_{target})$	10000	10000
$r(v = \text{true})$	-75	-10
$r(n > n_{\max})$	-75	-10
otherwise $r(\cdot)$	-100	-1

The reward function encourages the device controller to search for the target object by giving a substantial reward if the object is found, while penalising the controller for every action that does not result in a successful object detection. Furthermore, additional penalties are given if the controller generates a waypoint in an area it has explored before ($v = \text{true}$) or when it exceeds some search-length threshold denoted by $n_{\max}$. The reward values were empirically determined starting from the implementation of our previous MDP model [15], the rewards of which are summarised in Table 1 together with the new ones. The penalty given for every waypoint generated that does not lead to the target was significantly increased for this model to offset the delay before large penalties are introduced and increase the model's urgency.

The state observations are identical to the states that the mobile device can enter into. In this case, however, uncertainty is introduced into the observation by the object detector. Instead, the previous search locations and search time are fully observable, since they can be tracked by the mobile device. **O** therefore only contains the classification/misclassification probabilities of the object detector. These values were found by performing a set of classification tests with the object detector and generating a confusion matrix to populate **O**.

Training We encoded 15+1 objects into the current system, including a 'nothing' item in case the detector does not see anything of interest:

$$\mathbf{U} = \{nothing, monitor, keyboard, mouse, desk, laptop, mug, window, \\ lamp, backpack, chair, couch, plant, telephone, whiteboard, door\}.$$

We set $n_{\max}$ to 11 waypoints, after which the agent gets penalised for every additional waypoint it generates that does not lead to the target object. This results in a total of 352 reachable states ($n_{states} = 16 \times 11 \times 2$), with any state containing the target object acting as a terminal state.

The POMDP model is trained to generate a policy that contains the optimal belief-action mapping, which the controller can use to produce the optimal waypoint locations. This is done by the model exploring the entire state-action-observation space and optimising the policy to maximise its long-term cumulative reward. However, **b** is a vector of continuous probability distributions with infinite combinations, making POMDPs time-consuming or even impossible to solve exactly. This was the case for our moderately-sized state space, so we

opted for an approximate method instead, using the Point-Based Value Iteration (PBVI) algorithm [17] implemented in the AI-Toolbox library[4]. The PBVI algorithm speeds up the training process by selecting a smaller subset of representative belief points from **b** and tracking these points only. Using the PBVI algorithm, we generated a total of 15 policies, one for each target object.

3.2 Guidance System Implementation

We implemented the object detector and the POMDP controller in an Android app and combined it with an audio interface that provides non-visual guidance instructions. We use non-intrusive bone-conducting headphones to transmit the audio signals to the user without blocking other ambient sounds. The app runs in real-time on an Asus ZenPhone AR with Android 7.0 and ARCore[5] enabled.

Audio Interface To describe the waypoints' pan and tilt positions, we implemented an improved version of the interface described in [4] that uses a spatialised audio signal. However, since our headphones bypass the ear's pinnae that allow a person to localise the height [20], we spatialise the audio in the pan dimension only. To convey the tilt angle, we instead exploit a human's natural association of high and low sound sources with a high and low sound frequency, respectively [6], and adjust the sound source's pitch accordingly. A similar approach was used in [22].

Object Detector To recognise objects we used SSD-Lite, which is a single-stage object detection and classification network based on the SSD architecture that implements MobileNetV2 [21]. This is a lightweight model that requires relatively little memory to perform inference tasks, making it suitable for mobile platforms. This model achieves a mean average precision (mAP) of 0.22 with 4.3M parameters and 0.8B FLOPS on the COCO dataset [11]. The full SSD model achieves a slightly higher mAP (0.25), but with significantly more parameters (34.3M) and FLOPS (34.36B).

The network was trained with a maximum of 10,000 object samples for each class in **U**, taken from the OpenImages dataset [10], with a 60-20-20% split for training, validation and testing respectively. We set a relatively high confidence threshold of 0.7 to reduce the likelihood of false-positives. Training for 120 epochs, with 1000 iterations each, we achieved a mAP of 0.16 on this dataset and produced a TensorFlow Lite model that is Android-compatible. We used this model to finally implement our SSD-Lite object detector on the mobile device.

Waypoint Generation To search the target object, the system uses the policy file from the POMDP training process, which defines the best location of next waypoint based on the device's current state. This state is tracked by the device

[4] http://github.com/Svalorzen/AI-Toolbox
[5] http://developers.google.com/ar/

throughout the app's runtime by performing object detection and recording the previous search locations and waypoint positions. The latter are obtained by discretising the world into a 6×6 grid, where each cell represents a $35°$ rotation, and setting the relevant grid unit's v value when visited. This setup gives the state access to perfectly observable n and v parameters, while the observation u is generated by the object detector. When the device makes a new observation, either because the user rotates the device past $35°$ or sees a new object, the device triggers the controller to generate a new waypoint location. The controller uses o to update $\mathbf{b}$ and queries the policy for the best location to place the new waypoint. The policy output is an action from $\mathbf{A}$ that indicates the next adjacent grid square to place the waypoint, e.g. an 'UP' output would result in a waypoint being placed one grid square above the current view.

4 Experiment Design

To evaluate the guidance system's effectiveness, we designed a set of experiments to measure its performance in driving a user towards a target object within a static environment. We conducted an additional set of experiments with an alternative system that relies on a user's intuition and prior knowledge to generate actions instead, to act as a baseline measurement for our system's results.

Both experiment environments and the object placements within them were modelled on a typical office and care was taken to ensure that both environments were unique in layout and object placement, though some cross-experiment occurrences appear with the larger, more static objects (e.g. door, desk). Also, to minimise any cross-experiment learning effects, two different sets of objects were used for each experiment. If any (medium-small) objects occurred in both experiments, they were placed at different positions. In particular, the objects in the guided case experiment are $\mathbf{U}_g = \{door, desk, chair, whiteboard, mouse, laptop, backpack, mug\}$, while the objects in the unguided case are $\mathbf{U}_{ug} = \{door, desk, chair, whiteboard, mouse, monitor, telephone, keyboard\}$.

We recruited 10 participants (8 male, 2 female; average age 29.2 years) for the experiments, including 2 legally blind participants. The other 8 participants were blindfolded. A time limit of 45 seconds was set for each experiment run, which ended either by finding the target object or reaching the time limit. There was one experiment run per target in each respective environment, giving 8 experiment runs for each the of guided and unguided cases.

4.1 Unguided Case

For this experiment, the mobile camera and object detector acts as the participants' eyes and informs them about the objects within the camera's view. It is then up to the participant to exploit their prior knowledge and intuition to manipulate the camera and find the target object. In this case, the human acts as both the controller and the actuator. Similar to other commercially avail-

able applications, such as SeeingAI[6] and TapTapSee[7], the unguided version of our app only reads out the objects upon the user's request by tapping on the screen. When the target object comes within the camera's view and is correctly detected, the device vibrates to inform the participant.

4.2 Guided Case

In this experiment, we evaluate the performance of the guidance system in an object search task, where the perception and control tasks are performed by the guidance system and the participant acts as the actuator, interpreting control signals and outputting actuation forces on the camera sensor (see Fig. 2). To reduce the possibility of the participants ignoring the guidance instructions to find the target object by themselves, they were not told which objects they were actually looking for. This also helped to focus on the performance measurement of the guidance system only, isolating it from external factors such as user common-sense or other biases. A new run then started with the experiment staff selecting the (unknown) target object for the user.

5 Results

5.1 Target Acquisition

As in [15], we use the target acquisition rate (TAR) to compute the proportion of objects that the participants found during an experiment. For example, a TAR of 0.5 indicates that a participant found the target object in 50% of the searches. Taking each participant's TAR as a datum, the unguided case produced a higher average TAR (0.54 vs. 0.46), meaning that the participants found 8% more objects without guidance instructions. However, the Kruskal-Wallis (KW) test for non-normal data shows that statistically there is no significant difference between these results ($p_{kw} = 0.16$), meaning we cannot conclude which experiment produced the best TAR. All the experiment results are summarised in Table 2.

5.2 Time to Target

The time it takes to find a target object is an important indicator of system performance, where less time indicates a shorter search time and increased performance. The data for the search times in the guided and unguided experiments are shown in Fig. 4. It can be seen that the guidance system reduced the overall time to find each target object. This is confirmed by the data that show an average search time of 12.5s for the guided experiment and 17.2s for the unguided case ($p_{kw} = 0.045$), an improvement of around 27%. These results compare also favourably with those obtained in our previous version of the system [15], where

[6] http://www.microsoft.com/en-us/ai/seeing-ai
[7] http://taptapseeapp.com

Table 2. A summary of the experiment results.

	Guided	**Unguided**	**KW Statistic**
TAR [%]	46 ± 13	54 ± 15	0.16
Time to target [s]	12.5 ± 11.9	17.2 ± 11.5	0.045
Pan Angle Displacement [rad]	0.68 ± 1.1	0.99 ± 1.2	0.029
Tilt Angle Displacement [rad]	0.68 ± 1.1	1.04 ± 1.2	0.011
Linear Displacement [rad]	0.23 ± 0.22	0.36 ± 0.22	0.012

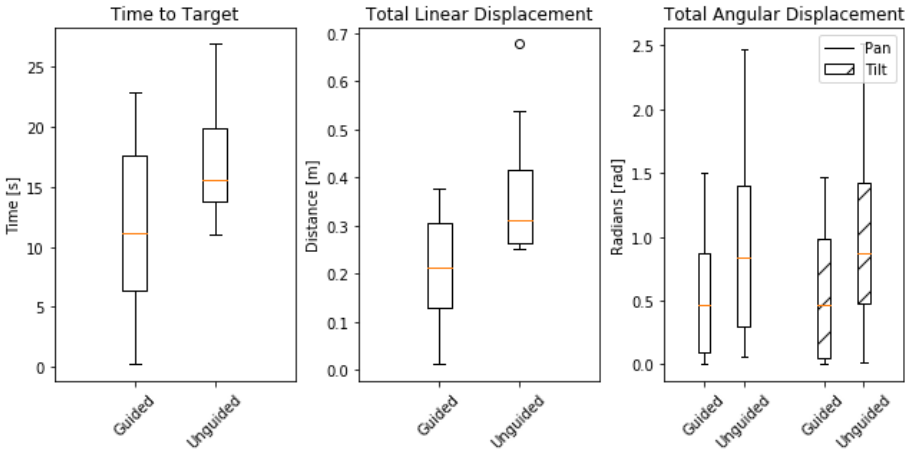

Fig. 4. A set of boxplots comparing the total linear and angular displacements, as well as the time to target for each experiment.

an average time-to-target of 34s was recorded.However, it should be noted that the system implementation and experimental design in the current paper are significantly different from our previous work, so this comparison is interesting but not indicative of any major improvement.

5.3 Movement

Finally, to give an indication of the effort required to find each target, we look at the mobile device's displacement data. In this case, less device displacement is desirable, since it implies less physical exertion was demanded from the user. The device displacement was measured in both linear (x, y, z) and angular (pan, tilt) dimensions. Integrating these data, we obtain the total absolute displacement in each dimension. These results are plotted in Fig. 4.

The boxplots for the total angular displacement show a consistent reduction in radians in the guided case for both the pan (0.68 rad vs. 0.99 rad, $p_{kw} = 0.029$) and tilt dimensions (0.68 rad vs. 1.05 rad, $p_{kw} = 0.011$). The guidance system

therefore reduces the total angular displacement to find the objects by 31% and 35% for the pan and tilt dimensions, respectively. The total linear displacement is also reduced when using the guidance interface by approximately 36% (0.23 m vs. 0.36 m, $p_{kw} = 0.012$). These results are summarised in Table 2. To conclude, the data show that the guidance interface reduced the total angular and linear displacement required to find the target objects in all dimensions by at least 31%, reducing the total effort required by the user.

6 Conclusion

In this work, we presented ActiVis, a mobile guidance system to scan the environment for finding objects, which uses a POMDP-based controller and a vision-based object detector, combined with an audio interface, to generate instructions for the user. We implemented this system on an Android app and tested it with a group of 10 participants to evaluate its effectiveness compared to an unguided object detector. The key results from these experiments are that the guidance system improved the participants' target-searching performance, reducing the total search time and overall camera manipulation effort required to find an object in an unknown environment. From these results, we can reasonably conclude that our new active search approach is potentially useful for similar mobile applications to help people with visual impairments.

The system can benefit from future work that focuses on improving the flexibility and usability of the system. New datasets that include the 3D positions of the objects and camera, similar to what has been done in robotics [1] but with a more diverse set of real scenes, would make it possible to train our model with more accurate transition and observation models and possibly extend it to be able to cope with depth. The usability of the system can potentially be improved with a more sophisticated audio interface and adding adaptive control algorithms that change the interface's behaviour based on the user performance over time [8].

Acknowledgements. This research is partly supported by a Google Faculty Research Award. We would like to thank the Voluntary Centre Services UK for their help in facilitating the experiments with people with limited vision.

References

1. Ammirato, P., Poirson, P., Park, E., Kosecka, J., Berg, A.C.: A dataset for developing and benchmarking active vision. In: Proc. of ICRA. IEEE (2017)
2. Andrew G. Howard, Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient convolutional neural networks for mobile vision applications. CoRR (2017)
3. Bajcsy, R., Aloimonos, Y., Tsotsos, J.K.: Revisiting active perception. Autonomous Robots pp. 1–20 (2017)

4. Bellotto, N.: A multimodal smartphone interface for active perception by visually impaired. In: SMC Int. Workshop on Human Machine Systems, Cyborgs and Enhancing Devices. IEEE (2013)
5. Bigham, J.P., Jayant, C., Miller, A., White, B., Yeh, T.: VizWiz::LocateIt – enabling blind people to locate objects in their environment. In: Proc. of CVPR Workshops. pp. 65–72. IEEE (2010)
6. Blauert, J.: Spatial hearing: the psychophysics of human sound localization. MIT press (1997)
7. 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.: 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 pp. 888–897 (2017)
8. Gallina, P., Bellotto, N., Di Luca, M.: Progressive Co-Adaptation in Human-Machine Interaction. In: Int. Conf. on Informatics in Control. pp. 2362 – 368 (2015)
9. Gude, R., Østerby, M., Soltveit, S.: Blind navigation and object recognition. Laboratory for Computational Stochastics, University of Aarhus, Denmark (2013)
10. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J.R.R., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Duerig, T., Ferrari, V.: The Open Images Dataset V4: unified image classification, object detection, and visual relationship detection at scale. CoRR (2018)
11. Li, Y., Li, J., Lin, W., Li, J.: Tiny-DSOD: Lightweight object detection for resource-restricted usages. In: Proc. of BMVC (2018)
12. Liu, L., Ouyang, W., Wang, X., Fieguth, P.W., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: A survey. CoRR (2018)
13. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: Proc. of ECCV (2016)
14. Lock, J.C., Cielniak, G., Bellotto, N.: Portable navigations system with adaptive multimodal interface for the blind. In: AAAI Spring Symposium – Designing the User Experience of Machine Learning Systems (2017)
15. Lock, J.C., Cielniak, G., Bellotto, N.: Active object search with a mobile device for people with visual impairments. In: Proc. of the 14th Int. Conf. on Computer Vision Theory and Applications. pp. 476–485 (2019)
16. Manduchi, R.: Mobile vision as assistive technology for the blind: An experimental study. In: Int. Conf. on Computers for Handicapped Persons. pp. 9–16. Springer (2012)
17. Pineau, J., Gordon, G., Thrun, S., et al.: Point-based value iteration: An anytime algorithm for POMDPs. In: Int. Joint Conf. on Artificial Intelligence Organization. pp. 1025–1032 (2003)
18. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proc. of CVPR. pp. 779–788. IEEE (2016)
19. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. Trans. on Pattern Analysis and Machine Intelligence pp. 1137–1149 (2015)
20. Roffler, S.K., Butler, R.A.: Factors that influence the localization of sound in the vertical plane. The Journal of the Acoustical Society of America (1968)
21. Sandler, M.B., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. Proc. of CVPR pp. 4510–4520 (2018)
22. Schauerte, B., Martinez, M., Constantinescu, A., Stiefelhagen, R.: An assistive vision system for the blind that helps find lost things. In: Int. Conf. on Computers for Handicapped Persons. pp. 566–572. Springer (2012)