PreSight: Enabling Real-time Detection of Accessibility Problems on Sidewalks

Zheng Li*, Mahbubur Rahman*, Ryan Robucci, Nilanjan Banerjee
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
{zhengli1, mahbub1, robucci, nilanb}@umbc.edu

***Abstract*—Accessibility problems such as obstacles on sidewalks can make navigation dangerous for the visually impaired. Detecting these accessibility problems using embedded cameras is a plausible remedy. However, current computer vision algorithms for object detection rely on exhaustive search with high-dimensional features that present a heavy computational burden and incur a long latency, making them non-ideal for real-time object detection on embedded platforms. To address this problem, inspired by prior-based searching schemes from human vision, we accelerate the machine vision process by using scene-specific features to select candidate regions in the view for further processing. Our system, `PreSight` achieves speedup by trading off the workload from on-line detection to off-line prior data collection and extraction. We demonstrate a complete, scalable `PreSight` prototype to accelerate general computer vision object detection algorithms with focus on detecting of sidewalk accessibility problems. Our prototype system automates the process of creating a geo-tagged database of object-specific priors using crowdsourcing and utilizes this prior knowledge to speedup object detection on embedded platforms. Evaluating under two benchmark object detection algorithms, we demonstrate that the detection latency can be reduced by around 8 times with the aid of `PreSight`.**

1. Introduction

A key element to improving the standard of living for people who have a visual impairment is independence [1]. Navigation on sidewalks and other walkways comprise a major ingredient of independent living. Unfortunately, sidewalks today are ridden with accessibility problems such as obstacles in path and unexpected flights of stairs. Existing systems such as smart canes [2], [3] or smart wheelchairs [4] suffer from the limitations of sensors such as ultrasound or infra-red used in these systems [5]. These sensors are not suitable for detecting multiple obstacles, or for applications that require object recognition, such as landmark detection for navigation.

A camera-based system is a plausible solution to the problem. Embedded cameras worn by users or built into canes combined with computer vision algorithms can be used for object detection and recognition. Unfortunately, however, current discriminative object detectors that mostly based on [6], [7] require extensive training in a high-dimensional feature space. During sensing and classification, the high-dimensional feature extraction and matching can be prohibitively expensive to compute in real time on embedded platforms.

To address this problem, we derive insight from the following human vision attribute: a human at a given scene can quickly select candidate regions of an object based on approximate descriptions of location, color, and size. These coarse-grained selections are considered top-down pre-attentive computation with prior knowledge [8], that can guide a human's attention [9], [10] into narrow regions for further fine-grained processing. This suggests that a faster searching scheme on limited hardware can be performed for a given object in a given scene if the object guidance attributes are known on a per-scene basis.

Inspired by this observation, in this paper, we propose a system `PreSight` that accelerates general object searching using per-case *a priori* characteristics. `PreSight` detects obstacles on sidewalks accurately in real-time with limited computational resources. The system guides its "attention" into narrow areas by incorporating prior knowledge of individual objects. For example, if we know *a priori* the color of the object, the regions in the view that have similar color can be pre-selected for further processing. Note that while the detection process is accelerated with per-case priors, additional complexity is introduced in the off-line data collection and extraction. Fortunately, the growth of crowdsourcing and smartphones provide a low-cost scalable solution for this task.

The overall operation of `PreSight` is illustrated in Fig. 1. In `PreSight` we build *a priori* scene specific information using data provided by the system users. When a user pass by an obstacle and do not get avoidance notification, s/he can provide information(image) of obstacle using preSight mobile application. These geo-tagged images are then annotated using Amazon Mechanical Turk (AMT). Scene specific object data such as the type, color, and physical size of the obstacle is collected using Mechanical Turk. The scene-specific attributes lazily populate a backend database. In the real-time detection system, an embedded camera retrieves data of the scene from a pre-cached local database. It is noteworthy that the data cached per scene is as minimal as 415 Bytes (only specific extracted features will

** The first two authors are listed in alphabetical order*

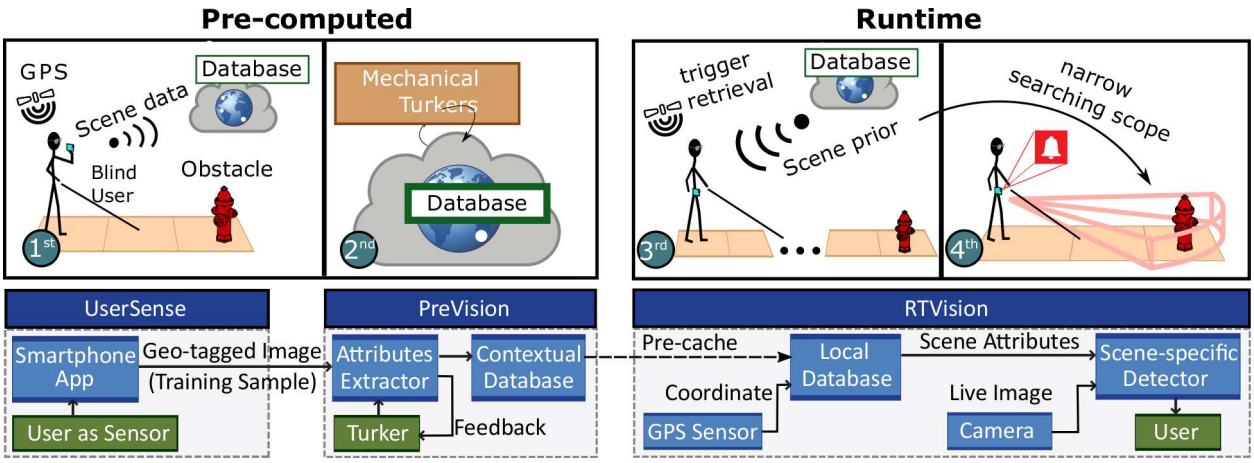

Figure 1: A cartoon figure illustrating the operation of `PreSight`. In the pre-computing process, the blind user-A reports the scene with the accessibility problem to our system by taking an image of the scene in the 1st step. Then our system publishes a HIT of the scene image and then the Turkers extract the parameters for the scene-specific detector in the 2nd step. In the real time subsystem, the smartphone-based system worn in another user-B retrieves the parameters from a pre-cached local database when approaching the scene, as shown in the 3rd step. Based on the scene attributes, the system focuses on sparse areas in the view to detect the object in real time, as shown in the 4th step.

be cached) and it can also be pre-cached when there is Wi-Fi available. The embedded system (a generic smartphone) utilizes the prior data to accelerate the detection process.

Research Contributions: The design, implementation, and evaluation of `PreSight` presents two novel research contributions.

A heuristic strategy to accelerate object detection by utilizing case-specific *a priori* characteristics to pre-filter live images: For individual scenes, this heuristic strategy enables attentive machine vision for which machines are able to focus the computation power on narrowed areas in the view with the help of *a priori* characteristics of the target. In another word, we offload the on-line machine computation by performing off-line prior extraction using human computation. This enables real-time computer vision object detection in embedded platform.

An end-to-end implementation and evaluation of the framework with application to accessibility problem detection: We demonstrate a functional prototype of `PreSight` that can accelerate general object-detection algorithms and achieve real-time obstacles detection in a commercially off-the-shelf smartphone. Our prototype `PreSight` system includes an interactive crowdsourcing web interface for prior data extraction, and a fully functional smartphone-based vision system. The evaluation shows that `PreSight` gains an 8x speedup on two benchmark object detection algorithms (HOG [6], DPM [7]) without degrading detection accuracy.

2. Related Work

Our work builds on existing literature on accelerating vision-based object detection, crowd-assisted vision, and assistive systems for the visually impaired. Here we compare and contrast our work with the most relevant literature.

Object Detection in Computer Vision: The benchmark computer vision object detection algorithms [6], [7], which also form the basis of a large portion of current state-of-the-art algorithms, rely on high-dimensional feature extraction and exhaustive matching. Such computation is prohibitively expensive on embedded platforms. `PreSight` simplifies the detection problem for a general object to scene-specific object detection. [11] tries to simplify DPM and accelerate with GPU, which is orthogonal to our approach while also requires extra hardware acceleration.

Reducing Computation Burden of Computer Vision in Mobile Systems: [12], [13] demonstrate techniques to reduce the computational burden of computer vision algorithms in mobile platforms. These include using location information to reduce search range and off-loading computation onto a backend server. While our system also uses a GPS unit to partition the search space, it does all the image computation locally in real time that eliminates the need to off-load vision computation onto a server.

Crowd-assisted Computer Vision: There is a body of work [13], [14], [15] that utilizes crowd sourcing to assist computer vision tasks, such as CrowdSearch or VizWiz. But these systems require the crowd to be constantly connected to the front-end system that largely affects the system latency. ALPS [16] can detect landmark using Google Street View images. But it propagates the GPS error, which makes it insufficient for blind users. Our system breaks the runtime dependency between human computation and machine computation. While the computation using humans is performed lazily in the background, the real-time object detection is performed locally in the embedded system.

Assistive Devices for the Visually-Impaired: Current non-camera-based assistive devices for the visually impaired [2], [3] utilize distance measurement sensors like laser, infrared and ultrasound to alert users of obstacles. These can be considered as extensions of traditional probing tools such as a white cane. Our system uses an image sensor to detect different objects, and can be applied to more general cases in obstacle avoidance and applications such as landmark localization. Current camera-based assistive systems [17], [18], [19] either require more computational resources to perform object detection which is impracti-

cal on embedded platforms, or only focus on very specific pattern detection. Our approach utilizes scene-specific priors to accelerate object detection that requires reduced computational resources and is also generalizable to other applications that require object searching and detection.

3. System Overview

`PreSight` comprises of three key components illustrated in Fig. 1: (1) UserSense; (2) PreVision; and (3) RTVision (Real-time Vision). Our system senses scenes with accessibility issues and extracts scene-specific prior in UserSense and PreVision, and performs realtime obstacle detection using RTVision. In UserSense, blind users capture images of accessibility problems on sidewalks using the mobile application when they find a problem on a sidewalk. The geotagged image is transferred to a backend server. The image is then encoded as a HIT (Human Intelligent Task) and disseminated to AMT in PreVision. Turkers annotate accessibility problem in the image, classify them, and provide data on the color and physical dimension of the accessibility problem. The collected data is then aggregated to build a geotagged database of scene-specific information. When another user visits the scene, the front-end smartphone-based application will utilize the prior data in the contextual database to accelerate general object detection algorithms to detect accessibility problems in real-time. This front-end, called RTVision, triggers the preprocessing and object detector only when the user is within a radius r of an accessibility problem registered in the PreVision database. Next, we discuss the three key components of our system in detail.

4. UserSense: Using Mobile Users to Sense Scenes with Accessibility Problems

In order to accelerate detection of accessibility problems, our system requires scene specific prior information which includes the following: an approximate location, the color and physical dimensions of an accessibility problem on a sidewalk. In our system, we ask users to report a scene with accessibility problem with as simple as a single geo-tagged image. And the rest required prior information will be automatically generated with AMT. In UserSense, we leverage blind users who are willing to contribute to help the community to "sense" scenes. When a user encounters an obstacle on a sidewalk, s/he could capture an image of that obstacle using our mobile application. Then the system will automatically collect data on the GPS coordinates, which are used for geo-tagging the image. Over time, scenes with accessibility problems will be collected and updated lazily by the users.

5. PreVision: Extracting Scene-Specific Features

In PreVision, the geo-tagged images collected by users are used to automatically generate HITs (Human Intelligence Tasks). The HITs are then disseminated to Amazon Mechanical Turkers to extract scene- and object-specific features. While data collected using Turkers can be subjective and biased, it is possible to design HITs to control quality, and de-bias by aggregating data from multiple Turkers. In this section, we first describe the design of HITs. We then describe how we aggregate data from multiple Turkers.

5.1. HIT Design

Our system (described in Section 6) uses two characteristics that humans use in their pre-attentive vision [10] (color and size) to accelerate object detection. Specifically, two key attributes of the scene are used to select candidate regions by a color segmentation and multi-scale size matching: (i) color of the obstacle in the image; and (ii) physical dimensions of the object. However, it is critical that the HITs should be presented using an intuitive and easy to understand interface [20], [21], [22]. In PreVision we provide explicit real-time feedback to the Turker on how well s/he has answered the HIT. Through this feedback the Turker has the opportunity to correct his/her answer and converge to a more precise answer. We explain this feedback mechanism in the context of collecting the two scene-specific attributes.

HIT 1: Extracting Object Color

The first HIT focuses on extracting the color of the obstacle in the image. Finding color of a specific object from an image using vision algorithm is a complex task as lots of other objects of same color may coexist in the image. One way to get the color of an object in the image is to tag that specific object, segment object by a color segmentation algorithm, create color histogram that represents the object. The accuracy of color information thus depends on the segmenting accuracy. But, this process is useful only for color retrieval process. For example, if we require the ground color instead of background color, this process may definitely fail. The background of an object may consist of different type of objects, i.e., trees, sky, grass etc. We can determine background color using the process, but it does not guarantee us to give the ground color. We use here the process of finding a specific feature of an object from an image asking human to provide information about that feature. In the first HIT, we ask human to provide best representing color of an obstacle on the sidewalk. Instead of directly asking the color of the obstacle (fire hydrant in Fig. 2(a)) we have designed a HIT where the user annotates a quadrilateral inside the object (shown in Fig. 2 (a)). Drawing a quadrilateral has two distinct advantages. First a single object may have more than one color, hence, a quadrilateral inside the object helps us infer the majority color. Second, it is easier for a Turker to annotate a shape inside the object rather than accurately point out the color of the object.

A key component of PreVision is that it provides real-time feedback to Turkers while they perform the annotation. This serves two purposes. First, it allows the worker to correct for mistakes by allowing them to see the impact of their input. In the most obvious case where they misunderstood the directions, the results of the feedback should

Figure 2: Amazon Mechanical Turk task for collecting data on color. (left) selecting object area and (right) corresponding feedback on the selection to the Turker.

make the error obvious. For example, in the color extraction HIT, the system computes a segmentation immediately using the input and returns a visual result to the worker. If the Turker accidentally selected a region misrepresenting the color of the object then the feedback results make the mistake obvious – the Turker is simply allowed to make the correction before completing the HIT. This feature minimizes the bias when aggregating data from multiple Turkers. Thus, a human that does not necessarily understand the image processing operations (a non-expert) may still work to produce a result meaningful for the machine-vision system. The Turker can modify the annotations as many times as he wants. Fig. 3 shows three iterations where the Turker annotates the image, observes the segmented image, and then modifies the annotations to provide more accurate results.

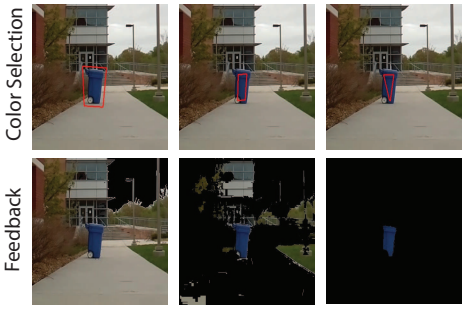

Figure 3: Interactive color selection of an object by a Turker. (1^{st} column) is the first attempt the Turker made and s/he can see the segmented image on the bottom. In (2^{nd} column), s/he tried to correct her/his annotation based on the feedback shown on the bottom. When s/he is satisfied with her/his annotation (3^{rd} column), then s/he submits the response.

HIT 2: Annotating Physical Dimension.

Another attribute that the real-time detection algorithm used is the physical dimensions (height and width) of the vertical object. To elicit this data, we have designed another HIT. In this task, the Turker draws a bounding box around the object and provides the physical height and width of that bounding box. However, it is difficult for common Turkers to accurately provide the absolute dimensions of the object. Humans are better at relative rather than absolute measurements [23]. So, to this end, we provide a reference object, a United States 100-dollar bill, which measures $6.14\,\text{in} \times 2.61\,\text{in}$. When the Turker selects an object and provides the dimension, the selected part or object is cropped from the image and is displayed side by side with the dollar bill. The object is transformed to a new dimension using a perspective transform of the object using the height and width provided by the user. For example, if a Turker provides the dimension of an object as $22\,\text{in} \times 9\,\text{in}$, then in the preview mode the Turker will see an object image, juxtaposed to a dollar bill and scaled to about 3 times the size of the dollar bill. In Fig. 4, we can see that one of the Turker annotation together with the preview image wherein the Turker provided $34\,\text{in}$ as the height of the vertical pole. So, the vertical pole looks almost 5 times longer than the one dollar-bill. Based on this feedback, the Turker can change the dimensions he provided for the object dimensions.

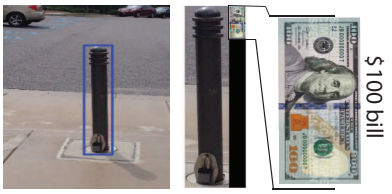

Figure 4: Interactive physical dimension estimation. Turker selects an object using a bounding box (left) and provides height and width, gets the feedback (right side). The feedback is the comparison of the dimensions between a US 100-dollar bill and the selected object which is rescaled based on the Turker's provided dimension.

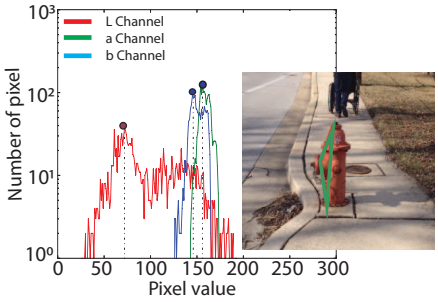

Figure 5: Color extraction from user annotation. Histograms of annotation for all three channels are generated by PreVision. From the histograms, three peak values are extracted as the representing color of that annotated area.

5.2. Data Aggregation

The PreVision subsystem of `PreSight` uses Turkers to extract *a priori* scene-specific data in the form of approximate geolocation of the accessibility problem, the color and physical dimensions of the object. For a single image this annotation is provided by multiple Turkers, and for a single scene, multiple volunteers may provide an image of the scene from different perspectives. This data, therefore, must be aggregated for the front-end RTVision system described in the next section. On average, we received 15 responses from Turkers for each HIT.

PreVision aggregates data from Turkers responding to HIT 1 (object color). For this HIT the Turkers select an area within the object of interest (accessibility problem). The RTVision algorithm described in the next section uses a color value in the CIELAB color space. To extract the representative color of the object, we first crop the image based on the Turker selection and then generate a distribution of the pixel color values in the CIELAB color space for the cropped image. Fig. 5 illustrates a histogram for a single Turker selection. From the histogram, we pick the `mode` color value in each of the color channels. This color represents the most frequently occurring color value for pixels in the selection made by the Turker. This algorithm is robust to poor selections made by the Turkers. Because

of the color channel histogram mechanism, the algorithm will always return the object color unless the selected area is dominated by objects other than its background. We aggregate the results across Turkers by taking an average of the pixel color values selected by each Turker.

The goal of the second HIT is to extract the physical dimensions of the object. For aggregating data from multiple Turkers, we can calculate the arithmetic mean of the dimension values, however, arithmetic mean is not robust to outliers. The Harmonic mean can handle outliers but it requires a large set of samples. We can also use a pruning-based outlier removal technique if we have a large training data. But, in our case the cost of recruiting Turkers prohibits the collection of a large training sample. In PreVision, therefore, we use the geometric mean of the dimension values to aggregate data from multiple Turkers. The geometric mean can handle a small training sample size and produces favorable results for our vision algorithm. The aggregate value of the dimensions populates a backend contextual database in PreVision and used in the RTVision subsystem. The RTVision system is described below.

6. RTVision: Real-time Accessibility Problem Detection and Localization

Figure 6: Prototype of the front-end system, RTVision. It is implemented in a generic smartphone and utilizes the camera, the GPS sensor which is used to trigger the camera, and the Wi-Fi module used to download the contextual database.

Our front-end system, RTVision, is implemented on a smartphone as shown in Fig. 6. RTVision takes GPS readings and queries a locally cached version of the PreVision database for accessibility problems in the vicinity of the GPS coordinates. It then uses the embedded camera to take images and uses the scene-specific features to preprocess the images, which will select a set of regions in the images for further processing. Then, general object detection algorithms can be run on only specific regions instead of the entire image to detect the accessibility problems. The underlying RTVision preprocessing algorithm comprises two parts: (1) segmenting the image using the color information; and (2) searching for a target in possible scales. We describe each part in detail below.

6.1. Color Image Segmentation

Based on the color information provided by the PreVision system, a typical way to segment the colored image is transforming the image from RGB into HSV and segmenting the image in a certain range in the Hue-Saturation plane. However, the reference image might be taken with another user's mobile device. The HSV color value might drift or scatter across different devices. Thus, instead of using HSV

Figure 7: Example of image segmentation across 4 different scenes with a consistent threshold in each color space. (a) images taken in the four scene with different types of accessibility problems. (b) the segmentation result in HSV space with a consistent dissimilarity threshold. (c) the segmentation result in LAB space with a consistent dissimilarity threshold. The figure shows segmenting in LAB space based on the color similarity is invariant cross different scenes. In the first column, the object is included in the segmentation in LAB, though the background trees with similar color are also included. For the object with salient color (2nd-4th rows), less background pixels are included in the segmentation in LAB while most of the objects pixels are included.

value, our system segments the color image in CIELAB space which is device independent. Fig. 7 shows an example of image segmentation across four different scenes with a consistent threshold in each color space. It is difficult to obtain acceptable segmentation in HSV with a fixed threshold across different devices. Also, as CIELAB is designed to mimic the color response of human eyes that uses three color opponents, the perceptual color dissimilarity can be measured by the Euclidean distance in the CIELAB space. The final segmented image is generated as:

$$I_{seg} = \begin{cases} 1, & \|(L,a,b) - (L',a',b')\|_2^2 < \epsilon_{Lab} \\ 0, & \text{otherwise} \end{cases}$$

where (L,a,b) is the color channels of the image in CIELAB space and (L',a',b') is the target color from the reference image. The output of this algorithm is a binary segmented image.

6.2. Distance-based Multi-Scale Searching

By knowing the aspect ratio of the target, we can also select regions with reasonable scales by multi-scale searching. Traditional scale-space [24] searching approaches downsample the image by a scaling factor σ into sufficient levels to construct an image pyramid. Then a fixed size feature template is used to match across the entire image in each layer of the pyramid. This method searches several scales of the object exhaustively. While it might get all possible matching, it also generates more false positives areas at different scales. This will produce more candidate areas for a following expensive detection operation and so degrade the acceleration.

As the scale is varied with the distance to the target, the search time can be reduced significantly by only searching one scale at a specific distance. In order to retrieve the distance (depth) information in the image pixels, the smartphone is mounted onto the waist of the user with a fixed height. With pre-calibration for each user, the height can

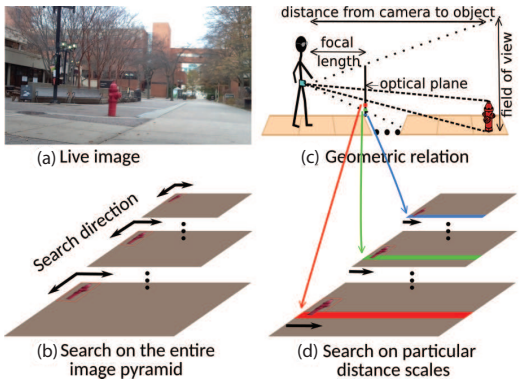

Figure 8: Multi-scale searching in the scale-space pyramid and our distance-based pyramid. (a) the image to be searched in. (b) the scale-space pyramid in which the target template is searched across both directions. (c) the geometry relation of the mounted camera, target and the projections of that in the image. (d) the narrow band needed to be searched in a specific layer in the pyramid. Compared to using a scale-space pyramid, our method only requires searching in much narrower area, hence it incurs less false positives and is faster.

be used for computing the distance based on the geometry relation. As illustrated in Fig. 8, the distance value can be warped onto the image pixels with the assumption that the ground between the camera and the target is parallel to the look-at vector of the camera. Using this approach, the false positives at unreasonable distance-scales can be eliminated. Also, the computation time for this preprocess itself will also decrease since less convolution is needed compared to scale-space method.

Specifically, RTVision generates the matching template T for the obstacle in its lowest scale based on the physical height and width of the obstacle provided by the PreVision system. The template is a center-surround box filter that emphasizes the central part and penalizes the surrounding. The size of the central part $k * [W_{phy}, H_{phy}]$ is calculated from the physical size of the trash can $[W_{phy}, H_{phy}]$, where k is the warping factor related to camera parameters (focal length, height of the camera to the ground, pixel resolution and the distance-map). The camera parameters are obtained through calibration. Then this template is convolved with the segmented image in our distance-based multi-scale method and the output is shown in Fig. 9.

Figure 9: An example of distance-based multi-scale matching. The first one is the image to be searched in. The second image is the segmented image generated by the approach described in the previous section. The last one is the final response of convolving the custom template in the segmented image in distance-based scales. The convolved output provides a likelihood map in which further detection can be operated on high-probability areas only.

After convolution, there might be a large concatenated region in the map with high likelihood value caused by the scaling effect, as shown in Fig. 9(c). However, the possible target will only be in the area around the local maxima. Here, we conduct non-maximum suppression to select the area only related to the local maxima.

7. System Evaluation

In this section, we first evaluate the acceleration of our system over two benchmark object detection algorithms, HOG and DPM. Then we evaluate the RTVision and PreVision subsystems respectively.

Experimental Setup

We perform `PreSight`'s evaluation using data collected with our prototype system. We identified 10 scenes with "obstacles-in-the-path" type of accessibility problems [25] around our university campus for our experiments. We collect videos of these scenes by walking on sidewalks with our system. To take account for the variance in real-world luminance and viewing angle, we also collect video clips at different times of the day and different approaching directions. Totally, 40 video clips from the 10 scenes are collected for the evaluation. All the videos are collected with a frame rate of 25 fps and a resolution of 1024×768.

To evaluate the latency while also demonstrating that `PreSight` can enable real-time object detection in embedded platform, we implement and evaluate the prototype on a generic smartphone with a 1.3-GHz quad-core ARM Cortex-A7 processor which was released on 2014.

Latency and Accuracy of Detection with `PreSight`

In our first set of experiments, we evaluate `PreSight` over two benchmark object detectors, HOG and DPM. We trained detectors for HOG and DPM respectively using the same methods authors described in the original paper. The detectors are trained with a set of images that consists of 270 images obtained using Google's image search utility. And we use the OpenCV implementation of HOG and DPM.

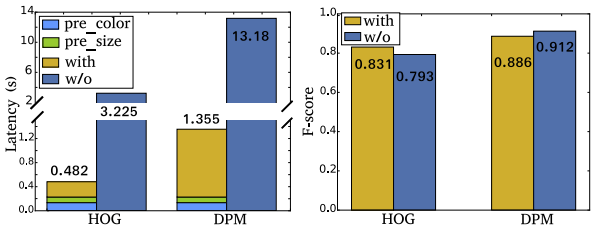

Figure 10: The latency and accuracy of HOG and DPM with or w/o using `PreSight`. The figure shows that `PreSight` can achieve an average 8x speedup to HOG and DPM without degrading the accuracy by more than 5%.

Fig. 10(a) illustrates the speedup of `PreSight`. With the preprocessing of `PreSight`, the average operating time of HOG and DPM on the dataset is decreased in about 1/12. The overhead of preprocessing (as pre_color and pre_size in the figure) is small compared to the time of the standalone detection process without the aid of `PreSight`, which is 1/14 of the time required in HOG and 1/58 of that in DPM respectively. The overhead is the same in both cases since it is independent to the process of detectors. Overall, with `PreSight`, the total latencies of detection using HOG and DPM are reduced by about 1/7 and 1/10 respectively. It's noteworthy that although there are several ways to enhance the real-time performance of HOG or DPM by exploring hardware capability [26], [27], they require

expensive or dedicated devices and PreSight would show similar performance improvements on these platforms.

We also evaluate how PreSight's preprocessing affects detection accuracy. Our primary evaluation metric for accuracy is F-score ($\frac{2TP}{2TP+FP+FN}$), which considers the true positives (TP), false positives (FP), and false negatives (FN). Fig. 10(b) compares the F-scores in cases of with or without PreSight. They are less than 5% difference in both detectors. In HOG, the F-score of the case of "with PreSight" is slightly higher than that of "without PreSight". This is because the preprocessing filters out areas which will generate false positives in HOG detection. In DPM, the F-score of the case of "with PreSight" is slightly lower. This is because that DPM is more accurate than that in the coarse-grained preprocessing. The miss cases of the preprocessing decrease the detector's performance slightly. Details of the performance of PreSight's preprocessing will be demonstrated below.

Distance-based Multi-Scale Preprocessing

Here we evaluate the performance of PreSight's preprocessing, while also comparing our distance-based multi-scale searching to general scale-space searching. Our primary evaluation metrics are precision and recall. Precision is defined as the portion of detections representing true positives ($\frac{TP}{TP+FP}$), which relates to the latency of follow-up detection, as more areas are filtered out less areas need to be searched. Recall is defined as the portion of the true positives actually found ($\frac{TP}{TP+FN}$), which relates to how PreSight affects detection accuracy of follow-up detectors. A lenient threshold is needed for coarse-grained preprocessing in order to preserve target area in the intermediate output. In Fig. 11 we show that our distance-based searching scheme (denoted as DB in the figure) can provide a similar recall rate (detection accuracy) with lower false positive areas (lower latency) when compared to a typical scale-space pyramid scheme (denoted by SS in the figure). Besides, our algorithm performs computation 2 times faster than the multi-scale pyramid as it only searches different scales at specific distances.

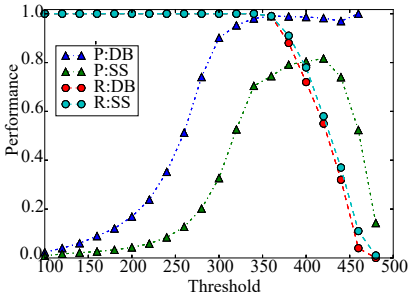

Figure 11: The Precision-threshold and Recall-threshold curves of matching in scale space pyramid and our distance-based scheme. The figure shows similar recall rate of two preprocessing approaches. But at the same recall point, there will be less false positives in distance-based (DB) method so that the precision of DB is much higher than that of the scale space (SS) method.

Analysis of Turkers Data

We next evaluate data provided by Turkers that has been used to generate contextual database off-line. The obstacle detection algorithm depends on three different features of an object. The PreVision thus has two different HITs to collect information about those two features from human crowd of Amazon Mechanical Turk. To evaluate PreVision data, we take images from 10 different sites around the campus and publish those to Amazon Mechanical Turk for human annotation. The color that largely represents the object is obtained from those responses using the process described in Section 5. The graph (see Fig. 12) shows that the differences between human responses and actual color is very small.

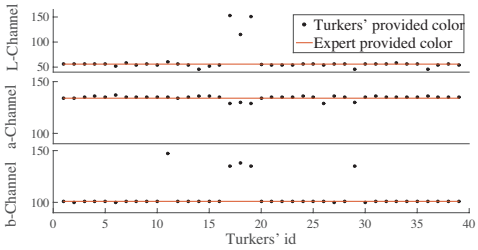

Figure 12: Turkers' responses of object color in L*a*b color space. The distance from solid line to circular points represent the differences between Turkers' and experts' responses.

The aggregated colors of obstacles (see Fig. 13) show that for channels a and b in L*a*b color space, the human provided and expert provided colors are almost equal all the time. The only difference is the L-channel value. L-channel represents the luminosity of that object. In our designed HITs, the turkers select an area of their choice from the object in order to find its color. The brightness of the object is not equal all over its surface though the color of the object looks the same. And, there is no guarantee that both turkers and experts select the same area of that object. So, the L-channel value in turkers' annotation differs from that in experts'.

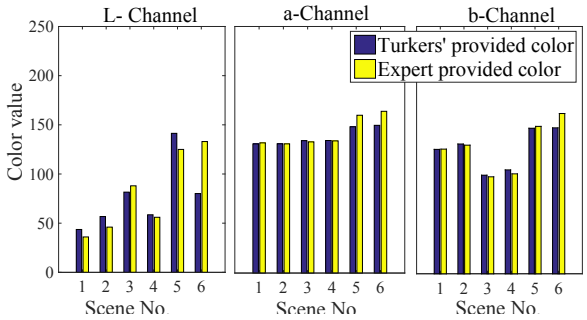

Figure 13: Aggregated color value from Turkers' and experts' responses in three different channels (L, a and b).

Our HIT of finding object dimension takes responses of human crowds for the object's height and width from an image. Human being has the intelligence of understanding object's dimensions if they ever have seen those objects in real world by their own; or at least an object is present in the image that they are familiar with. If so, they will be able to compare the unknown objects to the known one to guess the dimensions of unknown object. But they cannot guess dimensions of an object if they never see that object which is present in the image even with the feedback. But, most

of the objects are known to the human crowds which are generally presents on sidewalks. To evaluate the correctness of Turkers' responses, we measured the objects height and width using a measuring tape. We also use geometric mean to aggregate human provided height and width of that object. The graph of the object's height and width (see Fig. 14) shows that most of the Turkers can correctly provide the object's height and width if the objects are familiar to them or if any other familiar object is present in that image.

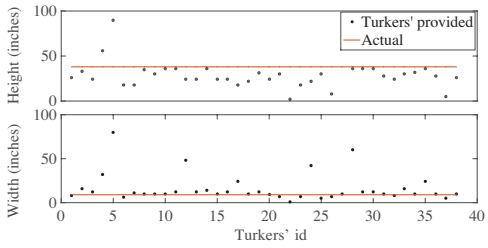

Figure 14: Turkers' responses of object height and width. The distance from solid line to circular points represent the differences between Turkers' responses and the ground truth of that object.

This is obvious when we see the aggregated responses for all the objects. The results (see Fig. 15) from human crowd comes close to actual values for each object if that object is known to turkers, i.e, trash cans and fire hydrant. But if the object is an unknown one or a known one with variable dimensions such as poles, there is no definite dimension where turkers would fail to guess correct dimensions most of the time.

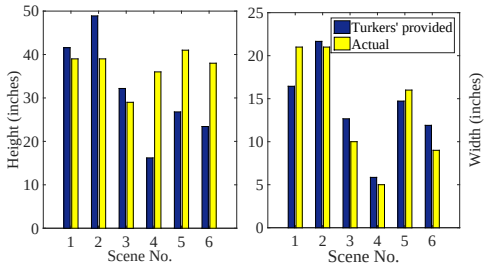

Figure 15: Aggregation of height and width from Turkers' responses.

`PreSight` uses human crowds to collect information for building contextual database. But data from human crowds have a cost in the form of money and time. In Amazon Mechanical Turk, we pay at least 1 US cent to get a single response. If we want at least 15 responses for unbiased data, it costs approximately 15 cents for each obstacle. We take into account the time to obtain 15 responses of each request to calculate the annotation time and can see that it takes almost a day (see Fig. 16) to get 15 responses for a request.

Detection Trajectory

Since our algorithm has low latency, it is possible to perform real-time detection on a per-frame basis with a usable update rate. This method can be used to improve the precision and recall by analyzing the detection results across a series of frames. In Fig. 17, we show the detection result

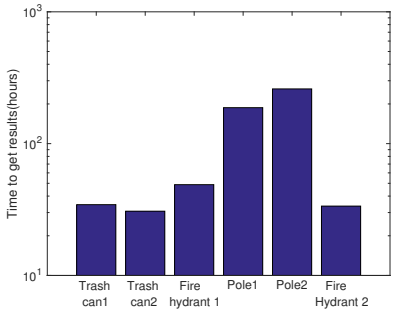

Figure 16: The time of collecting 15 responses from Amazon Mechanical Turk.

in each frame in a 25-FPS video. The video was taken as the camera approached a fire hydrant. The three dimensions in the graph represent ***time***, ***space***, and ***scale***. For clarity, we annotate the data by performing a 3D-point-linking rule to create color-coded clusters, based on time-space-scale locality and length of chains. While some false detections occur on individual frames, most of these false detections cannot be linked over a few frames. On the other hand, the detection points clustered and annotated in red, representing the actual fire hydrant detection, present a plausible path for an object relative to the approaching camera system.

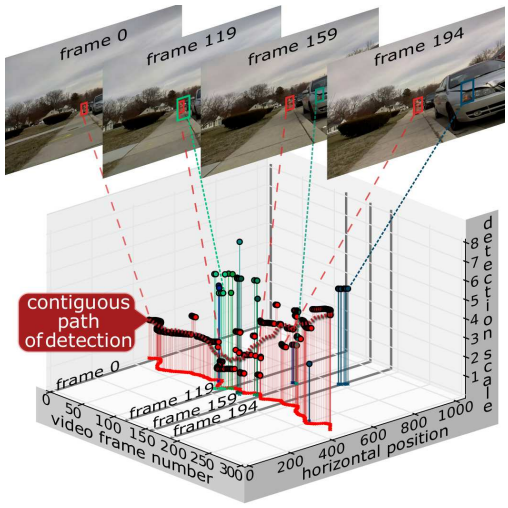

Figure 17: Detection points in time-space-scale, showing both false positives and true positives. The true detection of an object in a forward collision path obeys an expected trajectory model, increasing in scale with time with a high continuity in position near the middle of the frame.

The detection path of the fire hydrant shows the expected trend in the trajectory, moving from a small scale (long distance) to a large scale (short distance) over time. The vibration of camera system rolling along a rough side-walk path causes expected noise in the detection scale between adjacent scales. A contiguous path of detection is seen from a smoothed version of the data. False detections might be eliminated by evaluating the continuity in detection across frames, at least at a rate of one frame per second for our presented system. As shown in Fig. 17, the reddish portion of the image is a car twice briefly captured near Frames 159 and 194; while in Frame 119 the incorrect scale is detected in a few frames. More advanced application of Kalman

filters to track the object position, optionally combined with inertial sensor data, could be used to improve estimation of distance, and eliminate implausible detection sequences and trajectories irrelevant to obstacles.

8. Conclusions

TABLE 1: Summary of cost and automation of processes in PreSight

Component	Description	Automated	Cost per Scene
UserSense	Collect scenes	x	One image
PreVision	Generate HITs	☑	< 1 sec
	Turkers annotate	☑	\$0.15, < 24 hr
	Aggregate data	☑	< 1 sec
RTvision	Cache database	☑	415 Byte
	Preprocess	☑	0.23 sec

In this paper, we present PreSight, a prior-based vision system for speeding up general object detection and enabling real-time obstacle detection in a smartphone platform. The key idea behind PreSight is to preprocess the image with scene specific *a priori* information to select smaller areas for detection. The *a priori* used in PreSight includes color of the object and dimensions of the object. This scene-specific *a priori* information is collected in our system using mobile user "sensors" who take images of accessibility problems on sidewalks and Turkers who annotate information on the images. We have implemented a full functional prototype and automate most processes in low cost, as shown in TABLE 1. In the evaluation, we show that PreSight can accelerate general object detection algorithms, like HOG or DPM, in an average of 8 times faster without degrading the detection accuracy by much. While we demonstrate the feasibility of our approach in the context of sidewalk accessibility problem detection, we believe that such a strategy could be applied to navigation, indoor location, and augmented reality.

References

[1] WHO, "Action plan for the prevention of avoidable blindness and visual impairment," http://www.who.int/blindness/ACTION_PLAN_WHA62-1-English.pdf.

[2] S. Shoval, I. Ulrich, and J. Borenstein, "Navbelt and the guidecane: obstacle-avoidance systems for the blind and visually impaired," *Robotics Automation Magazine, IEEE*, vol. 10, no. 1, pp. 9–20, Mar 2003.

[3] J. Borenstein, "The guidecane-applying mobile robot technologies to assist the visually impaired," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 31, no. 2, pp. 131–136, Mar 2001.

[4] V. Ivanchenko, J. Coughlan, W. Gerrey, and H. Shen, "Computer vision-based clear path guidance for blind wheelchair users," in *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, 2008, pp. 291–292.

[5] M. A. Hersh and M. A. Johnson, *Assistive Technology for Visually Impaired and Blind People*. Springer London, 2008.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*. IEEE, June 2005, pp. 886–893.

[7] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.

[8] L. Itti and C. Koch, "Computational modelling of visual attention," *Nature reviews neuroscience*, vol. 2, no. 3, pp. 194–203, 2001.

[9] J. Moran and R. Desimone, "Selective attention gates visual processing in the extrastriate cortex," *Science*, vol. 229, no. 4715, pp. 782–784, 1985.

[10] J. M. Wolfe and T. S. Horowitz, "What attributes guide the deployment of visual attention and how do they do it?" *Nature reviews neuroscience*, vol. 5, no. 6, pp. 495–501, 2004.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *arXiv preprint arXiv:1506.02640*, 2015.

[12] P. Jain, J. Manweiler, and R. Roy Choudhury, "Overlay: Practical mobile augmented reality," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2015, pp. 331–344.

[13] T. Yan, V. Kumar, and D. Ganesan, "CrowdSearch : Exploiting Crowds for Accurate Real-time Image Search on Mobile Phones," *Image Rochester NY*, pp. 77–90, 2010.

[14] J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, B. White, S. White *et al.*, "Vizwiz: nearly real-time answers to visual questions," in *UIST '10*. ACM, 2010, pp. 333–342.

[15] M. Rahman, B. Blackwell, N. Banerjee, and D. Saraswat, "Smartphone-based hierarchical crowdsourcing for weed identification," *Comput. Electron. Agric.*, vol. 113, no. C, pp. 14–23, 2015.

[16] Y. Hu, X. Liu, S. Nath, and R. Govindan, "Alps: Accurate landmark positioning at city scales," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016, pp. 1147–1158.

[17] A. Rodríguez, J. J. Yebes, P. F. Alcantarilla, L. M. Bergasa, J. Almazán, and A. Cela, "Assisting the visually impaired: obstacle detection and warning system by acoustic feedback," *Sensors*, vol. 12, no. 12, pp. 17 476–17 496, 2012.

[18] L. Shangguan, Z. Yang, Z. Zhou, X. Zheng, C. Wu, and Y. Liu, "Crossnavi: enabling real-time crossroad navigation for the blind with commodity phones," in *UbiComp '14*. ACM, 2014, pp. 787–798.

[19] Y. Lu and H. A. Karimi, "Real-time sidewalk slope calculation through integration of gps trajectory and image data to assist people with disabilities in navigation," *ISPRS International Journal of Geo-Information*, vol. 4, no. 2, pp. 741–753, 2015.

[20] a. Kittur, B. Smus, S. Khamkar, and R. Kraut, "Crowdforge: Crowdsourcing complex work," *UIST*, pp. 43–52, 2011.

[21] S. Dow, A. Kulkarni, S. Klemmer, and B. Hartmann, "Shepherding the crowd yields better work," in *CSCW*, 2012, pp. 1013–1022.

[22] A. Finnerty, P. Kucherbaev, S. Tranquillini, and G. Convertino, "Keep it simple: Reward and task design in crowdsourcing," in *Proceedings of the Biannual Conference of the Italian Chapter of SIGCHI*. ACM, 2013, p. 14.

[23] B. Wu, T. L. Ooi, and Z. J. He, "Perceiving distance accurately by a directional process of integrating ground information," *Nature*, vol. 428, no. 6978, pp. 73–77, 2004.

[24] T. Lindeberg, "Scale-space theory: a basic tool for analyzing structures at different scales," *Journal of Applied Statistics*, vol. 21, no. 1-2, pp. 225–270, 1994.

[25] K. Hara, V. Le, and J. Froehlich, "Combining crowdsourcing and google street view to identify street-level accessibility problems," in *CHI '13*, 2013, pp. 631–640.

[26] M. Hahnle, F. Saxen, M. Hisung, U. Brunsmann, and K. Doll, "FPGA-Based real-time pedestrian detection on high-resolution images," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 629–635, 2013.

[27] P.-Y. Chen, C.-C. Huang, C.-Y. Lien, and Y.-H. Tsai, "An Efficient Hardware Implementation of HOG Feature Extraction for Human Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 656–662, 2014.