KinSpace: Passive Obstacle Detection via Kinect

Chris Greenwood[1], Shahriar Nirjon[1], John Stankovic[1], Hee Jung Yoon[2],
Ho-Kyeong Ra[2], and Sang Son[2]

[1] University of Virginia, Computer Science Department
{cmg7t, smn8z, stankovic}@virginia.edu
[2] Daegu Gyeongbuk Institute of Science and Technology (DGIST), Department of Information and Communication Engineering
{heejung8, hk, son}@dgist.ac.kr

Abstract. Falls are a significant problem for the elderly living independently in the home. Many falls occur due to household objects left in open spaces. We present KinSpace, a passive obstacle detection system for the home. KinSpace employs the use of a Kinect sensor to learn the open space of an environment through observation of resident walking patterns. It then monitors the open space for obstacles that are potential tripping hazards and notifies the residents accordingly. KinSpace uses real-time depth data and human-in-the-loop feedback to adjust its understanding of the open space of an environment. We present a 5,000-frame deployment dataset spanning multiple homes and classes of objects. We present results showing the effectiveness of our underlying technical solutions in identifying open spaces and obstacles. The results for both lab testing and a deployment in an actual home show roughly 80% accuracy for both open space detection and obstacle detection even in the presence of many real-world issues. Consequently, this new technology shows great potential to reduce the risk of falls in the home due to environmental hazards.

Keywords: fall prevention, object detection, obstacles, safety, Kinect

1 Introduction

Falls account for a large number of the injuries sustained in the home. Various studies estimate that from 33-52% of adults aged 65 or greater have at least one fall per year [2, 13]. These falls are the leading cause of injury-related hospitalization for this population [13]. Behind motor vehicles, falls are the second largest contributor to the economic burden of injuries in the United States, and amount to almost $20 billion in estimated annual cost (about 1.5% of the national healthcare expenditure) [8].

It has also been shown that falls in the elderly population can largely be attributed to trips [3, 20]. Researchers at Colorado State University estimate that about one third of falls in the elderly occur due to environmental hazards in the home, the most common of which is tripping over objects on the floor [20].

This gives clear motivation for the development of a system to assist in keeping living spaces free of obstacles in an effort to prevent falls.

Most existing solutions deal with the detection of falls as they occur, not the detection of environmental factors that cause falls. This, of course, does not actually prevent falls. Some research has been done in systems that detect obstacles to prevent falls, however existing solutions to this approach require that the user wear a camera system on the body. A system that is statically installed in the home and requires no repeated user input represents a significantly more scalable and user-friendly approach.

This paper makes the following contributions:

1. KinSpace, a passive, automatic, Kinect-based open space detection, obstacle detection, and alert system
2. A set of system features that address real-world complicating factors involved with in-home deployments of Kinect systems.
3. A lab evaluation of KinSpace that fully analyzes the accuracy of both open space and obstacle detection.
4. An deployment of KinSpace in two homes that demonstrates the handling of real-world issues, showing 80% overall obstacle detection accuracy.
5. A large deployment data set spanning several rooms and multiple classes of objects, with roughly 5,000 detection frames.

2 Related Work

Our work lies at the intersection of several existing bodies of research.

One area is object segmentation. These systems perform processing on depth data to gain additional information about the scene. Silberman [19] presents work on scene segmentation, in which individual objects are segmented from the background of a scene. Greuter et. al. [6] use depth information to control a robot for the Eurobot Challenge, in which the robot must navigate obstacles while moving objects about a predefined playing space. Similar projects leverage depth information from the Kinect or similar sensors in the field of object tracking for home healthcare [12].

When considering the application area of fall detection and prevention, existing work has primarily focused on detecting a fall after it has occurred. This has been done in various ways. Many systems have been developed to perform fall detection using wearable sensors [5, 13, 15, 18]. Some systems employ the use of smart phones to reduce friction with the user [4, 7]. Other systems, such as Life Alert, employ the user as a sensor and provide a simple notification system to alert others about a fall [10].

Work has been done in the area of fall prevention using a depth camera, mainly in the assistance of visually impaired users. Bernabei et. al. [1] present a real-time system in which the depth sensor is attached to the user and the system notifies the user through earphones of obstacles in his immediate path. Zöllner et. al. [23] propose a similar system that uses vibrotactile feedback to communicate obstacle notifications to the user.

Our work is also influenced by the research area of general home monitoring. Well Aware Systems [21] has developed a comprehensive monitoring system aimed at the elderly and disabled living independently. Wood and Stankovic [22] propose AlarmNet, which uses a similar wireless sensor network to learn resident patterns to further inform healthcare providers. We hypothesize that KinSpace could be added as one element of such deployments.

3 Obstacle Detection: The Problem and Issues

We define obstacle detection as the process of monitoring a room and detecting objects that are likely to cause a fall. An obstacle is likely to cause falls because of its size and position in the room relative to where individuals routinely walk. An obstacle detection system detects any such objects and notifies the proper person in the event of detection so that appropriate action can be taken to minimize risk of falls.

This problem is difficult for several reasons. First, it is difficult to identify the "open space" where misplaced objects would be considered falling hazards. Second, once the open space is defined, we have an equally complex problem of determining which elements in the scene are non-obstacles (floor, background, furniture, large movable objects such as a chair, etc.) and which are true obstacles. This understanding of which objects are truly obstacles can also potentially change over time.

Intervention Strategy: When an obstacle is detected by the system as a risk to the resident, an intervention should take place to minimize the risk of falling. This intervention could be a notification to the resident (visual, auditory, technological, etc.) or a notification to another individual such as a healthcare provider. There are several factors that affect the success of these different modalities of intervention, such as the physical/mental condition of the resident, the reaction time of non-residents, and the user's reaction to false alarms.

False Alarms: False alarms are a problem that must be handled in any safety system. Particularly in this type of system, if the system warns about numerous obstacles that happen to not be obstacles, the user will lose confidence in the system and any caregivers notified of obstacles may not take the proper precautions.

Real-World Environment: There are several real-world factors that make the problem of obstacle detection more complex in deployment than in lab testing. For in-home deployment, the open space may not be as rigidly defined as in lab environments. It may also change over time due to rearranging furniture, addition of permanent objects, changing travel patterns, natural settling of objects, or sensor drift. A robust obstacle detection system must be flexible in its definition of open space so as to evolve over time.

Another complicating factor about real-world deployment is that different people and different scenarios lead to different objects being considered actual obstacles. For instance, if a pair of shoes is left in the middle of a hallway with no residents currently in the room, we might all agree that this is an obstacle and potential falling hazard. But what if a resident comes into the room and sets a shopping bag down temporarily? Some might consider this an obstacle because the user may turn around and forget the bag is there. But a user that remembers the bag is there may not want to be notified each time that obstacle is detected.

Real-world environments also present the system with a much more varied set of objects and detection patterns. Even if a system is extremely reliable in its detection of large, rigid obstacles, there is inherent complexity introduced when the obstacle is smaller or non-rigid (a towel, an empty shopping bag, or even a spill). Obstructions and occlusions are complicating factors that any real-world system must address.

There is also noise that occurs in real-world sensor data that makes obstacle detection much more difficult. For instance, in deployment, the Kinect sensor may not detect the presence of humans in the scene depending on their distance from the sensor, their angle with respect to the sensor, and the angle at which they enter the scene. This may lead the system to consider parts of a user's person as potential obstacles. Noisy depth data and irregular objects may also cause the system to incorrectly segment a single object into multiple, further complicating the problem of determining a true obstacle from background.

4 KinSpace

KinSpace uses a statically-installed Microsoft Kinect sensor to perform open space and obstacle detection in a room so as to minimize the risk of falls. This section discusses the process by which KinSpace learns the open space in the scene and then detects obstacles in that environment.

4.1 System Overview

Each KinSpace system (one or more per room to account for sensor range) is made up of a Kinect sensor, a processing unit (laptop or embedded processing device), and a GUI feedback unit on a laptop. At installation time the system is placed in training mode through indication on the feedback unit. During training mode, the processing unit receives skeleton data from the sensor. It uses this skeleton data to record where users walk in the environment as well as information about the orientation of the floor relative to the sensor.

The user then places the system in detection mode. In detection mode, the processing unit receives depth stream data from the sensor and user input from the feedback unit. It detects obstacles in the scene and adjusts its understanding of the open space based on detection results and user input. It then passes detection results to the feedback unit for notification to the user.

4.2 Algorithm Description

Training - Data Collection: The sensor is installed at its desired location and enabled in training mode. KinSpace then monitors the room using the skeleton data stream from the Kinect sensor [11]. Whenever a skeleton is detected, KinSpace checks the tracking state of each foot joint for each skeleton and records the position information for all foot joints that are marked as Tracked by the Kinect. The Kinect also provides an estimate of the floor plane equation (in a sensor-centric coordinate system), which KinSpace saves for normalization of these skeleton points.

Training - Data Processing: When the user places the system in detection mode, training data collection is complete, and post-processing occurs to produce the final training dataset. We use the latest floor plane equation to redefine the coordinate system for all depth points. The new coordinate system has its origin on the floor directly beneath the Kinect and a y-axis that is normal to the floor plane. KinSpace interprets the y-coordinate of a point as its height off the ground, and uses the Euclidean distance (considering only the x- and z-coordinates) between two points to represent the lateral distance (bird's eye distance) between those points. KinSpace calculates a transformation matrix (rotation and translation) that converts all Kinect coordinates into this new coordinate system. All foot points captured during data collection are transformed to generate a normalized training set of foot points. The system filters this set of projected points, removing any whose height is above a certain threshold. A large height value for a foot point indicates either that the user's feet were not on the ground in this frame or that a measurement error occurred. In either case, the foot point is probably not a good representation of the open space in a room. For example, if the user is lying on a bed, we do not want foot points to be captured and added to the open space. The transformation matrix for the scene and the resulting set of filtered projected foot points gives us the full training set to be used during detection.

Detection - Lateral Filter: In the detection phase, KinSpace captures and analyzes one frame per second. At each frame, all depth pixels are converted to the sensor-centric coordinate space by Kinect software, producing one 3D point per pixel. KinSpace then transforms these points into the floor-centered coordinate space. The system computes the lateral distance between each point and its nearest neighbor in the training set. That lateral distance is computed using a simple 2D Euclidean distance calculation, ignoring the y-coordinate. This results in a distance that does not take relative height of the objects into account. Any point with a lateral distance less than the *lateral distance threshold* is considered an obstacle pixel candidate.

Detection - Vertical Filter: After filtering by the *lateral distance threshold*, the set of obstacle pixel candidates will contain not only obstacle pixels, but also actual floor pixels in or near the open space. To account for this, the system filters the candidate set by the height of the pixel relative to the floor. Any pixel that has a height less than the *vertical distance threshold* is discarded. This gives us a candidate set of pixels that are within a specified distance of the open space and are above a specified height.

Detection - Cluster and Size Filter: This set of candidates is likely to contain some noise, or pixels that are not part of true obstacles. We make the assumption that actual obstacles are composed of mostly connected pixels, and that noise in the candidate set is mostly be made up of individual pixels or small patches of connected pixels. The system clusters candidate pixels into disjoint sets of pixels (segments) using binary connected component filtering [16]. To remove noise, KinSpace then filters the segment set to remove all segments whose pixel count is less than the *object size threshold.* All objects that remain are not just laterally close enough to the open space and high enough, but contain enough connected pixels to be considered a true obstacle.

Detection - Output: At every frame, the obstacle detection algorithm outputs this set of objects, each of which is made up of a set of pixels in the depth frame. This allows us to estimate the size of each object as well as its approximate location in the room.

4.3 Real-World Considerations

The algorithm described above performs very well in idealistic conditions. However, as discussed in Section 3, there are several real-world factors that make the problem of obstacle detection much more difficult in deployment, and KinSpace has several additional features implemented to address these factors.

The first real-world factor that must be addressed is false positives. KinSpace gives the user the ability to actively acknowledge an alert and provide feedback to the system. When an alert is raised, the user can either remove the true obstacle or push a button on the feedback unit that indicates to KinSpace that whatever is being detected at that time is not a true obstacle. KinSpace then adapts its detection process through what we call baseline filtering. When a false positive has been indicated by user feedback, KinSpace takes a snapshot of the pixels that were indicated as obstacle candidates during the frame in question. Then for future detection, candidate pixels in the baseline filter are discarded from the candidate pixel set. One possible use case for this would be if a resident places a new piece of furniture in the room. KinSpace would likely detect this obstacle and the user would be alerted to it. Once the user notifies KinSpace that the piece of furniture is meant to be there, all of the pixels detected as part of that chair are ruled out from future consideration, and the piece of furniture effectively becomes part of the background.

Sensor drift and natural settling of the environment lead to additional false positives. KinSpace provides more continuous false positive reduction through baseline evolution so as to minimize false positives caused by these phenomena. When KinSpace is first placed in detection mode it performs an initial baseline filtering to remove any false positives in the environment at startup time. This could occur if the user stepped particularly close to a background object, for instance. After this initial calculation, baseline evolution begins. At each detection frame, KinSpace has knowledge of its current baseline filter. It computes the obstacle candidate segments as described in Section 4.2. It then scans these candidate segments searching for segments that are (1) adjacent to existing baseline regions, and (2) small in segment size relative to the size of the adjacent baseline region. Pixel segments that satisfy these two conditions are considered not as obstacles but instead portions of the evolving baseline. These pixel segments are added to the existing baseline (and thus not detected as obstacle pixels for the current and all subsequent frames).

To account for objects being split into multiple discontiguous pixel segments, KinSpace uses isotropic dilation to fuse segments that are close to one another into a single detected obstacle. The more prevalent problem with signal noise is that users can be erroneously detected as obstacles themselves. To solve this problem, we implement a temporal consistency process. After obstacles are detected in the current frame, KinSpace checks each obstacle for temporal consistency with the previous N frames (N being defined by the *temporal consistency factor*). Temporal consistency is satisfied when a given obstacle in frame 1 and any obstacle in frame 0 have similar size (relative to the size of the object itself) and similar location (relative to the size of the frame). An obstacle is only considered a true obstacle if it maintains temporal consistency with all previous N frames (we use N=2 in our experiments). There is a tradeoff here in that when an object is initially placed in an open space, there is a slight delay as the object establishes temporal consistency with previous frames, during which time the obstacle is not detected. However, we found this to be a reasonable tradeoff since the intended use of KinSpace is detecting objects that have been misplaced, and are thus not in motion.

The final real-world factor we address is the position of detected obstacles relative to users in the scene. It is a trivial task to ensure that any pixels detected as obstacles do not lie on the users themselves. But we also do not want KinSpace to detect an obstacle when a user places an object down with the intention of picking it right back up. Because of this desired behavior, we implement the following protocol. When an obstacle is detected and there are valid skeletons detected in the scene, KinSpace delays an alert about this obstacle until one of two conditions are met: (a) the user moves a certain distance away from the detected obstacle, or (b) a certain period of time passes without the obstacle being moved. Both the user distance threshold and the time threshold are configurable parameters. This protocol aims to reduce frustration with the system by reducing false positives that occur when the user is fully aware of his or her placement of objects in the room.

5 Evaluation

We evaluate KinSpace using an extensive set of controlled lab experiments and several in-home deployments.

5.1 Lab Experimental Setup

Our lab experiments involve placing a Kinect in the lab and marking off a portion of the visible floor space as an open area. This allows us to test the effectiveness of the system at detecting objects within the open area while ignoring objects that are outside the open area. After designating the open area, we actively train the system by having an experimenter traverse the open area by foot.

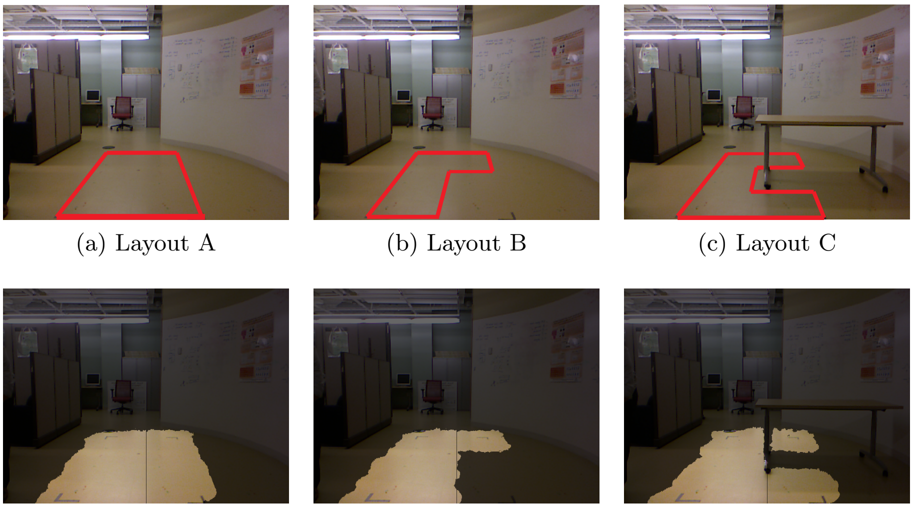
We test the system under three different floor layouts (see Figure 1). These layouts simulate those we would expect to see in a real deployment. Layout A is the simplest of the three, defined by a rectangle in the center of the frame of view with no obstructions. This is similar to what one would see in a hallway that is generally kept with little furniture. Layout B also has no obstructions, but the open area is defined as an L shape instead of a simple rectangle. For layout C, we started with the same simple rectangle as in layout A, but placed a table on one side. Users are forced to walk around the table when navigating the open space, and thus we expect to see the open space omitting space under the table. The sensor has a view of both under the table and the floor on the opposite side of the table to test detection accuracy under both cases.

After the training phase, we place obstacles inside and outside the open area to test the detection accuracy of our system. We capture 3 individual frames with each configuration of objects in the scene. Our first experiment tests the accuracy of the system at adapting its detection to the three different layouts. We do this by placing a series of objects about four inches in height at distances between seven and fourteen feet from the sensor throughout the open space. We visually inspected the detection results to ensure that the detected objects were not disjoint from the actual objects in the scene. We then conduct an experiment to quantify the maximum usable distance of the system by training on layout A as well as space directly behind layout A. We vary the distance from the sensor and at each distance, place multiple objects both inside and outside the open area. Finally, we perform an analysis of the accuracy of the system as the size of the object changes. Because of the measurement resolution of the Kinect and the lateral and vertical distance thresholding done by the system, we know that at a certain point an object is too small to detect. By using variably sized items, we aim to quantify the effect of these parameters.

5.2 Lab Results

Open Space Calculation: Figure 1 depicts the open space calculated by KinSpace when trained under each layout. In these images the bright areas represent portions of the scene that that KinSpace determines to be open space. We see that in each layout, KinSpace is able to learn nearly 100% of the open space through active training. We note that each calculated open space region also includes some overlapping onto non-open space areas - we define this as the false positive open space. This is due to the lateral distance threshold, which we set as 0.2 meters for all detection experiments. Decreasing this threshold would lead to less false positives caused by the border around the open space, but also leaves the potential for gaps in the true open space. We discuss the effect of this parameter in more depth in Section 5.3.

It is clear from inspection that the system adapts its understanding of the open space based on the foot patterns of users during training. Additionally, we note that in Layout C, the system effectively excludes portions of the scene that



(a) Layout A

(b) Layout B

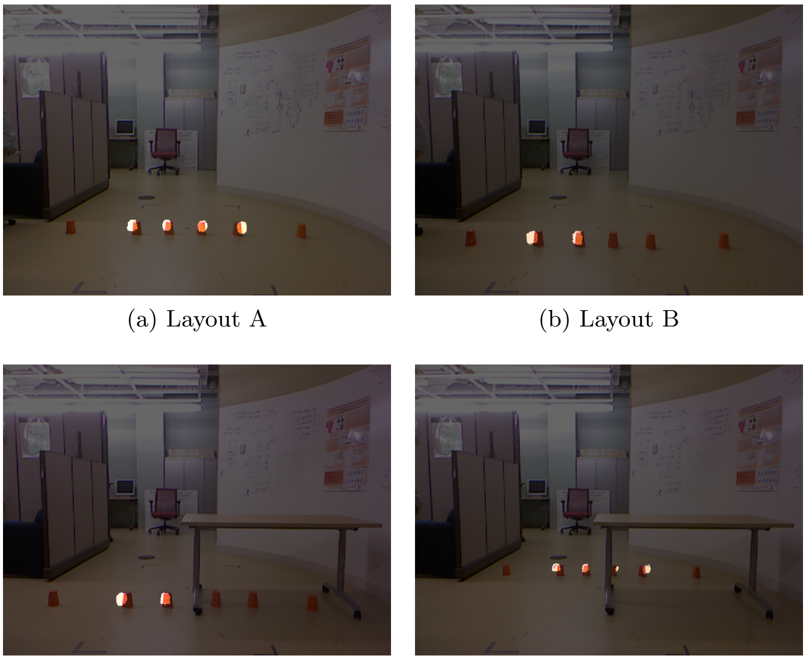
(c) Layout C

(d) Open space A

(e) Open space B

(f) Open space C

Fig. 1: Figures (a)-(c) show the three layouts defined for our lab experiments. Figures (d)-(f) depict the open space calculated by KinSpace when trained under each layout.



(a) Layout A

(b) Layout B

(c) Layout C with objects under table

(d) Layout C with objects on opposite side of table

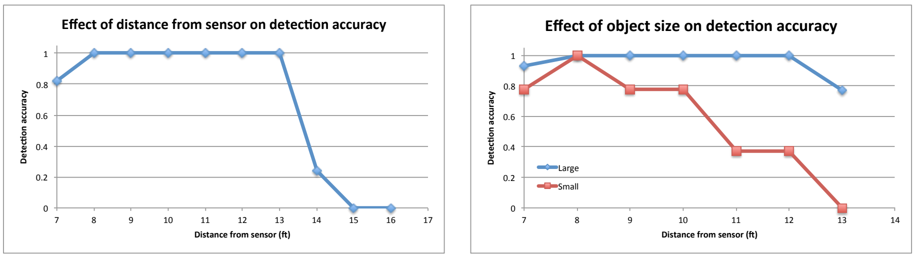
Fig. 2: The layout used to train the system dictates which objects are detected as obstacles by KinSpace

are not true floor space even if they are close to the walking path of the user (the legs of the table).

Effect of Layout: In Figure 2a, we see that under Layout A, the system successfully detects the four objects that lie within the defined open space of the environment while ignoring objects placed outside this open space. In layout B (Figure 2b), KinSpace again is able to detect objects inside the open space and discard objects that lie in areas not covered by the new training region. Under Layout C, we see that the system is effective both at discarding objects that lie underneath the table (thus outside of the open space) as well as correctly detecting obstacles on the opposite side of the table.

Effect of Distance from Kinect: This analysis is performed on layout A. We trained the system using a large area and attempted to detect objects as they were moved further from the Kinect. From Figure 3a we observe consistent detection accuracy in the range of 7-13 feet from the sensor, after which point the accuracy decreases sharply. This can be attributed to the maximum depth sensing distance.

Effect of Size of Object: The large objects used are roughly 4-5 inches in height, while the small objects are roughly 2-3 inches in height. From Figure 3b we observe that as the size of the object decreases, the effective detection distance decreases as well. This makes sense - the smaller an object is, the less distance it has to be from the sensor before the number of pixels it occupies falls below the threshold. As the object gets smaller it will also fall below the vertical distance threshold, which was set to roughly 1.5 inches for our experiments. A possible improvement to the current system would be to vary the object size threshold based on how far away the object is or use additional Kinect systems in the room.



(a) Distance

(b) Size

Fig. 3: Controlled lab experimentation that evaluates the effect of distance from the sensor and object size on detection accuracy

5.3 Deployment Experimental Setup

In the second phase of our evaluation, we deploy the system into two home environments. The first goal of these deployments is to evaluate how well the system can estimate the open space of a room by using passive training only. We install KinSpace in the room and allow it to train while residents of the home go about their normal activities. The system uses the activities of the residents to determine which portions of the room should be deemed open space.

We then test the system's calculation of the open space in the scene. A trial participant uses a color image of the scene to hand label the perceived open space, which provides us with ground truth. We compare the open space determined by KinSpace to this ground truth to determine how well KinSpace calculates the open space. An analysis is also done on this data to test the effects of the lateral distance threshold parameter. We do so by varying the parameter and testing the accuracy of KinSpace's open space determination with respect to ground truth.

The second goal of our deployments is to test the system's detection of obstacles in a real-world environment. We do so using a scripted set of actions performed by a test user over the course of several minutes (see Figure 4). The script is performed five times for various classes and sizes of objects. We test small objects that are likely to cause tripping, such as water bottles and shoes, as well as large objects that a resident may run into, such as chairs and light fixtures. We hand label each trial with the true number of obstacles in the scene at each frame and then compare the output of KinSpace to this ground truth.

1. Walk in, drop object 1, walk out
2. Walk in, pick up object 1, drop object 2, walk out
3. Walk in, drop object 1, walk out
4. Walk in, pick up object 2, drop object 3, walk out
5. Walk in, drop object 3, walk out
6. Walk in, pick up all objects, walk out

Fig. 4: Deployment experiment script used for all trials

5.4 Deployment Results

Open Space Calculation: We allow KinSpace to train on the natural movements of residents throughout the home. We then observe the accuracy of the calculated open space with respect to ground truth, as the number of frames considered by KinSpace increases. We also vary the *lateral distance threshold* to observe the effects of this parameter on the resulting open space calculation. One would expect that as the lateral distance threshold increases, passive training allows the system to learn more of the ground truth open space, but also increases the false positive rate.

The left graph in Figure 5 shows the portion of the true open space that is captured in the system's determination of open space as the number of frames considered increases. The right graph depicts the false positive rate as the system considers more frames. Note that the number of considered frames includes only those in which a valid skeleton was detected. We observe several distinct points in the training process where there is a sharp spike in the portion of the open

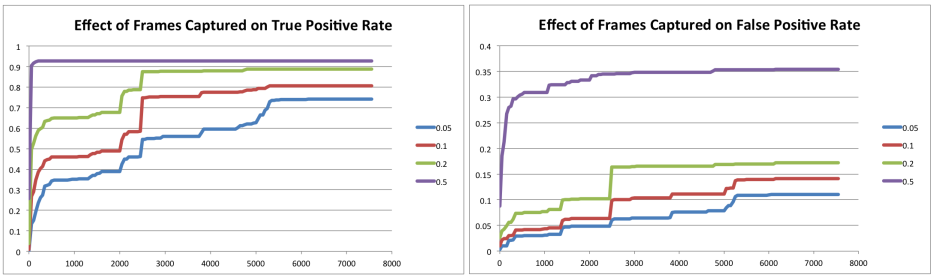


Fig. 5: Accuracy of passive open space detection, showing true positive and false positives rates as the number of training frames increases.

space that is recognized by the system - these spikes represent times when a resident moved into a portion of the open space for the first time. These spikes in the system's knowledge of the open space come with a smaller increase in false positives. As the resident moves about that space, additional knowledge of the open space slows.

After observing these results, we decided to use a lateral distance threshold of 0.2 meters for all experiments, as it offered the best tradeoff between true positive and false positive rates.

Obstacle Detection: We first present two examples of a scripted trial. Over each trial, KinSpace monitors the environment and indicates the number of obstacles it detects in the scene. We present this output over time, along with the hand-labeled ground truth.

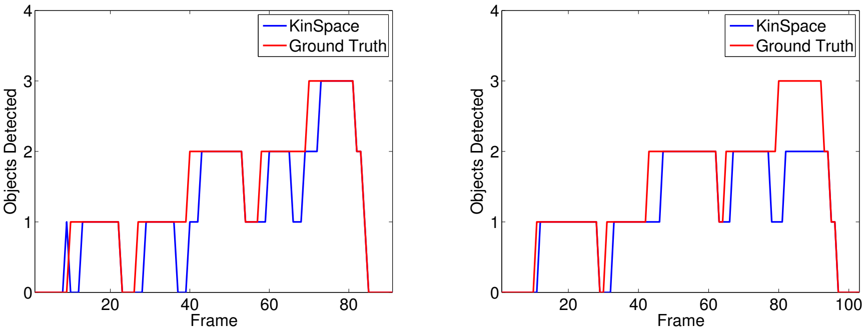


Fig. 6: Examples of scripted trial output. (a) Note very high detection accuracy except for transitions when user is actually manipulating objects; (b) An object is missed after Event 5, likely due to isotropic dilation.

In the first example (see Figure 6), for sections where the system is in a stable state and the user is not in the process of dropping or picking up an object, the system proves to be extremely reliable. The main source of detection error occurs when a user is near the obstacles in question, which causes objects to be in motion, occlusion of objects, and the user's body to be considered as part of an obstacle. Such factors cause temporal consistency to be broken and the system fails to recognize a stable obstacle.

We also present an example where even in a stable state, the system had problems detecting all objects in the scene. In this trial, the system is very accurate until Event 5, at which time it fails to detect a third object in the scene. Upon visual inspection of the output, we see that in a case like this, two of the objects are close enough to be considered the same object after applying isotropic dilation on the object segments. Though this is a potential source of inaccuracy of the system, since the main goal of the system is to detect the presence of obstacles at all, the confusion between one obstacle and several obstacles is not a critical flaw.

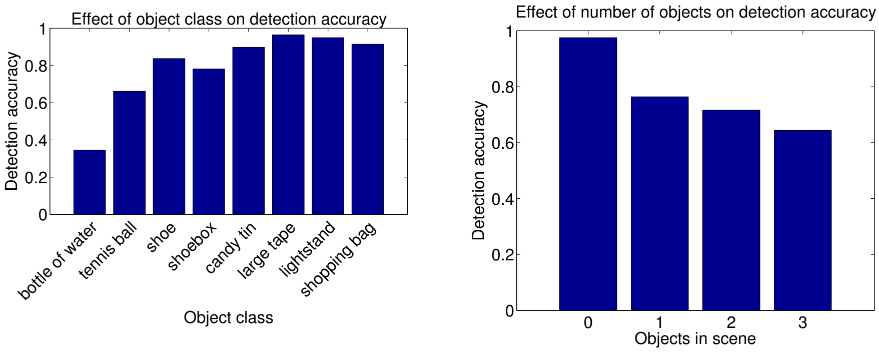
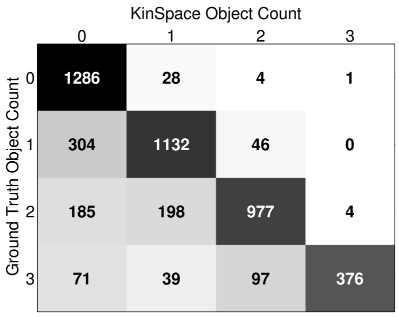


Fig. 7: (a) KinSpace detected most objects with high accuracy, but had trouble in particular detecting bottles of water; (b) Over all object classes, there was a consistent degradation of detection accuracy as number of objects increased.

We next present aggregate results of running the scripted deployment across multiple object classes, for 5 trials per object class. We first examine the effects of the class of object on the accuracy of detecting the correct object count, regardless of what the true object count is. We observe that KinSpace performs very well for a wide variety of object classes, but struggles on certain classes, particularly water bottles. This is likely due to the relatively small size of the water bottle combined with its translucence, which causes Kinect data errors. We next examine the effect of the number of objects in the scene on detection accuracy across all object classes. We saw consistently poorer detection accuracy in the frames in which more objects were present. This is likely due to a number of factors. First, with more objects in the same amount of area, it is more likely that occlusion (an external factor) or isotropic dilation (an internal factor) causes KinSpace to fuse two objects and consider them one, causing an erroneous detected object count. Second, the majority of frames in which a human was manipulating the scene were those in which the ground truth object count was greater than one. If we were to disregard these frames completely and focus on stable state we would likely see an increase in detection accuracy with multiple objects.

Figure 8 shows a confusion matrix between the ground truth obstacle count and the obstacle count detected by KinSpace for all object classes and trials (over 5000 frames of deployment data). We observe over 79% detection performance overall and note that when an error does occur, KinSpace is more likely to detect fewer objects than actually exist in the scene. This gives us confidence that if KinSpace alerts the user to one or more object in the scene, these are actually obstacles and not false positives.



	0	1	2	3
0	1286	28	4	1
1	304	1132	46	0
2	185	198	977	4
3	71	39	97	376

Fig. 8: Confusion matrix incorporating 5000 frames of deployment data. Error cells in the bottom left portion indicate false negatives (18.8%), while error cells in the top right indicate false positives (1.7%).

6 Discussion

KinSpace shows promising results both in lab and in deployment. However, there are several limitations of the system, as well as additional considerations that would have to be addressed in a production system.

One limitation is the size and profile of an object that can be detected by KinSpace. Because KinSpace applies a threshold to both an object's height and the number of contiguous pixels, objects that are small enough will not be detected. We adjusted the parameters of the system to minimize the effect and note the effect of this limitation in our results section. Furthermore, we note that if an object is small enough to be filtered out by the thresholding process of KinSpace, it is likely that its depth profile will be lost in the measurement accuracy of the Kinect sensor itself. This indicates that although object size is a limitation of KinSpace, it is a limitation that is (to some extent) imposed by the Kinect itself and applies to any similar system.

Another limitation of the system is the inherent inaccuracies that occur when a user is in the process of manipulating objects in the scene. As an object is being picked up or set down, there is often a 1-2 frame range on either end of the event during which time detection is inaccurate. Though this is a source of error, this is not a major limitation with respect to the primary motivation of KinSpace - to notify users about the presence of obstacles that may cause falls. Though the number of obstacles is helpful, the key information we want to capture is whether or not an object is there at all. As such, the confusion between two and three obstacles, for instance, is a minor issue. Also, KinSpace is primarily useful when a user forgets that an object has been placed in an open space. In fact, a production system would likely not notify the user of an obstacle for a certain period of time, during which time we can assume the user intends for that object to be there. Because of this, we felt it best for the system to default to not detecting obstacles when in doubt. This reduces the number of false positives and ensures that when KinSpace detects an object in the scene, it has a relatively high level of confidence that there is truly an obstacle there.

This work develops a technology whose motivation is fall prevention. We develop and analyze the properties of the technical solution. In the future, we would like to extend current technology to address additional object classes such as flat objects and spills. We would also like to perform additional analysis to evaluate the user feedback aspect of KinSpace as well as its potential to prevent falls.

7 Conclusion

We have presented KinSpace, a system that uses the Kinect sensor to observe an environment and learn the open space. It then monitors that open space for the presence of obstacles and notifies the resident if an obstacle is left in the open space. KinSpace employs a feedback loop with the user and with its own output to allow it to evolve its understanding of the objects in the scene. We have tested KinSpace extensively in lab to prove the potential of using the Kinect sensor for obstacle detection in open spaces. We have also presented two in-home deployments and produced a large data set that spans multiple rooms and numerous object classes (5,000 total frames of testing data). Through this deployment experimentation we have shown an obstacle detection accuracy of 80%. We have shown a very low false positive rate, proving the reliability of KinSpace as a tool for notifying residents about falling hazards. KinSpace is shown to be an easily-deployable, low cost system that helps keep open spaces clear in the home, potentially preventing falls and injuries for residents.

Acknowledgements This work was supported in part by NSF CNS-1319302, and by the DGIST R&D Program of the Ministry of Science, ICT, and Future Planning of Korea (CPS Global Center).

References

1. Bernabei, D., Ganovelli, F., Benedetto, M. D., Dellepiane, M., & Scopigno, R. (2011) A low-cost time-critical obstacle avoidance system for the visually impaired. International Conference on Indoor Positioning and Indoor Navigation.
2. Blake, A. J., et al. (1988). Falls by elderly people at home: Prevalence and associated factors. Age and Ageing, 17(6), 365-72.

3. Chen, H., Ashton-Miller , J. A., Alexander, N. B., & Schultz, A. B. (1994). Age effects on strategies used to avoid obstacles. Gait & Posture, 2, 139-146.
4. Dai, J., Bai, X., Yang, Z., Shen, Z., & Xuan, D. (2010). Mobile phone-based pervasive fall detection. Personal Ubiquitous Comput., 14(7), 633-643.
5. Diaz, A., Prado, M., Roa, L. M., Reina-Tosina, J., & Sanchez, G. (2004). Preliminary evaluation of a full-time falling monitor for the elderly. Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE, , 1. pp. 2180-2183.
6. Greuter, M., Rosenfelder, M., Blaich, M., & Bittel, O. (2011). Obstacle and game element detection with the 3D-sensor kinect. 161, 130-143.
7. Hansen, T. R., Eklund, J. M., Sprinkle, J., Bajcsy, R., & Sastry, S. (2005). Using smart sensors and a camera phone to detect and verify the fall of elderly persons. European Medicine, Biology and Engineering Conference.
8. Heinrich, S., Rapp, K., Rissmann, U., Becker, C., & Konig, H. (2010). Cost of falls in old age: A systematic review. Osteoporosis International, 21(6), 891-902.
9. Khan, A., Moideen, F., Lopez, J., Khoo, W., & Zhu, Z. (2012). KinDectect: Kinect detecting objects.7383, 588-595.
10. Life alert. http://www.lifealert.com/
11. Microsoft kinect coordinate spaces. http://msdn.microsoft.com/en-us/library/hh973078.aspx
12. Nirjon, S., and Stankovic, J. (2012). Kinsight: Localizing and Tracking Household Objects using Depth-Camera Sensors. In Proc. of Distributed Computing in Sensor Systems, Hangzhou, China.
13. Noury, N., et al. (2003). A smart sensor based on rules and its evaluation in daily routines. Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE, , 4. pp. 3286-3289 o.4.
14. Noury, N., et al. (2000). Monitoring behavior in home using a smart fall sensor and position sensors. Conference on Microtechnologies in Medicine and Biology, 1st Annual International. 2000, pp. 607-610.
15. Qiang Li, Stankovic, J. A., Hanson, M. A., Barth, A. T., Lach, J., & Gang Zhou. (2009). Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on, pp. 138-143.
16. Rosenfeld, A. (1970). Connectivity in digital pictures. J.Acm, 17(1), 146-160.
17. Rubenstein, L. Z., & Josephson, K. R. (May 2012). The epidemiology of falls and syncope. Clinics in Geriatric Medicine, 18(2), 141-158.
18. Shan, S., & Yuan, T. (2010). A wearable pre-impact fall detector using feature selection and support vector machine. Signal Processing (ICSP), 2010 IEEE 10th International Conference on, pp. 1686-1689.
19. Silberman, N., & Fergus, R. (2011). Indoor scene segmentation using a structured light sensor. Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pp. 601-608.
20. Tremblay Jr., K. R., & Barber, C. E. (2005). Preventing falls in the elderly., 2013, from http://www.ext.colostate.edu/pubs/consumer/10242.html
21. Well aware systems. http://wellawaresystems.com
22. Wood, A., Stankovic, J. A., Virone, G., Selavo, L., Zhimin He, Qiuhua Cao, et al. (2008). Context-aware wireless sensor networks for assisted living and residential monitoring. Network, IEEE, 22(4), 26-33.
23. Zöllner, M., Huber, S., Jetter, H., & Reiterer, H. (2011). NAVI - A proof-of-concept of a mobile navigational aid for visually impaired based on the microsoft kinect. 6949, 584-587.