Kinsight: Localizing and Tracking Household Objects using Depth-Camera Sensors

Shahriar Nirjon
Department of Computer Science
University of Virginia
smn8z@virginia.edu

John A. Stankovic
Department of Computer Science
University of Virginia
stankovic@virginia.edu

***Abstract*—We solve the problem of localizing and tracking household objects using a depth-camera sensor network. We design and implement *Kinsight* that tracks household objects indirectly – by tracking human figures, and detecting and recognizing objects from human-object interactions. We devise two novel algorithms: (1) Depth Sweep – that uses depth information to efficiently extract objects from an image, and (2) Context Oriented Object Recognition – that uses location history and activity context along with an RGB image to recognize objects at home. We thoroughly evaluate Kinsight's performance with a rich set of controlled experiments. We also deploy Kinsight in real-world scenarios and show that it achieves an average localization error of about 13 cm.**

I. Introduction

We interact with varieties of objects at home everyday. We grab objects, interact with them, and place them somewhere once we are done with them. Imagine the possibilities that open if we had a system that could keep account of all the objects that we interact with in our daily lives. By knowing what objects one is dealing with, we could infer what activity that person is doing [13]. By keeping track of the locations of the objects, we could build a smart search engine for our home that could answers queries like – where are my eye glasses, or my tv-remote controller, or my wallet? To materialize such possibilities, as a first step, we build Kinsight, which detects human-object interactions, recognizes objects, and keeps track of the locations of the objects – using its keen sight.

Tracking systems – be it for household objects, or cars and pedestrians on the road, or the zebras in the wilderness – can be divided into two broad categories: *intrusive* and *non-intrusive* systems. Intrusive systems [10, 11, 17, 20] attach small sensors or tags to the object and monitor the tags remotely. Typically, these are proximity sensing systems. The downside of using such a system for household object tracking, e.g., RFID based localization systems [28], is that – first, we need to attach tags to every object we wish to monitor which is a burden, and second, to localize the objects, we require a large number of readers, each costing around $\$500 - \$2,000$ [3]. Non-intrusive systems [8, 24, 33], on the other hand, are primarily computer vision based. These systems use cameras and apply high-end image processing algorithms to detect and track objects in the scene.

Kinsight falls into the category of non-intrusive, vision based object localization system. It uses a network of Kinects[1] to track the human figures in 3D. It detects human-object interactions, samples images of objects in real-time, and applies image processing and machine learning algorithms to recognize and localize objects.

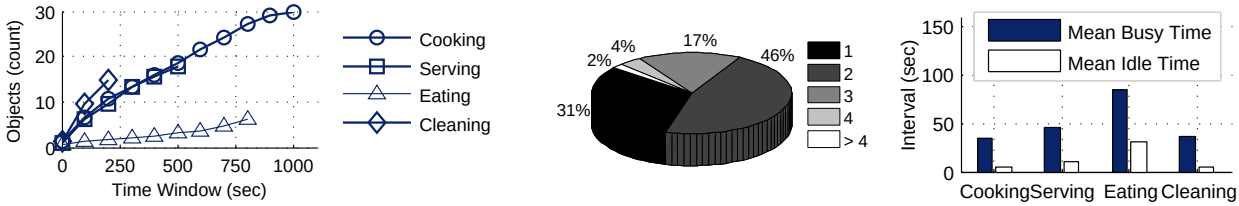
Kinsight is designed and optimized for households – considering various unique features of household objects and the environment. The task distribution within a Kinsight node is designed to handle the dynamics of the household environment at different levels – allowing us to inter-mingle online and offline algorithms to solve the problem. Unlike most vision based algorithms that depend solely on images, Kinsight applies unsupervised learning algorithms to learn the appearance, likelihood locations, and the activity context of individual object *instances* to improve its performance over time. Installation of Kinsight is easy as the hardware is an off-the-shelf, commercially available, USB connectible device, and the software runs as an application. Kinsight offers a cost effective solution – costing about $\$120$ per sensor which is $4-16$ times cheaper than RFID based systems [3] to monitor the same sized area.

The design of Kinsight is based on a set of assumptions that we believe reasonable for most multi-person households. Violation of any of these assumptions may result in a sub-optimal performance, but the system won't fail catastrophically. We throughly evaluate the performance of Kinsight with a rich set of controlled experiments to analyze its sensitivity to distance of objects, human motion, sampling rate, colors and sizes of objects, occlusion and multiple views. To demonstrate the practicality of Kinsight, we also deploy the system in two multi-person households and discuss the findings.

The contributions of this paper are the following:

- We design and implement an inexpensive, easy to deploy, highly accurate, and novel depth-camera sensor system which is optimized to detect and track household objects' locations.
- We describe two novel algorithms: (1) *Depth Sweep* - that exploits human-object interaction and depth information to efficiently extract objects from images, and (2) *Context Oriented Object Recognition* - that uses location history and activity context along with RGB images to recognize objects.
- We evaluate the system under different conditions to quantify its sensitivity to a wide variety of parameters. We deploy Kinsight in real-world scenarios and demonstrate that its average localization error in practical situations is about 13 cm.

[1] A depth-camera sensor based natural user interface developed at Microsoft.



(a) Number of objects used within a time window. (b) Modalities of objects' locations. (c) Length of idle and busy times.

Fig. 1. Study results from two households of four daily activities that use varieties of objects.

II. Household Object Localization

Localization of household objects by tracking is a special case of the tracking problem, in which, the objective is to find a mapping between a set of objects and their corresponding locations. It involves discovery of objects, obtaining their 3D locations, and updating their locations whenever they are changed. Several distinguishing characteristics make this problem different than the general tracking problem. These observations give us the opportunity to apply various optimization techniques to design and implement an efficient household object tracking system. In this section, we describe these observations, which form the basis of the system assumptions in Kinsight. While most of these assumptions are readily understandable from our everyday experiences, in an attempt to quantify these, we conduct some experiments in two multi-person households using video cameras and RFID tags. We use these data in our discussion whenever they are relevant.

A. Human-object interaction is the prime cause of location changes of household objects.

Tracking algorithms are computation intensive. As the number of targets grow, running time of these algorithms become comparable to the frame rate and thus makes it infeasible to track all of them an once. However, for household objects, we make a simplified assumption that, objects change locations due to human interactions only. In our study, we observe this assumption to be always true. This gives us the opportunity to design a novel object tracking system, in which, instead of tracking the objects, we track the human figures in a scene and keep looking for location changes of nearby objects due to human-object interactions.

B. The number of objects a person interacts with during an activity is limited within a limited time window.

A person may interact with many objects during an activity, but if we count the number of objects that one deals with within a fixed length time window, the number cannot be unbounded. The limit comes from the physical limitation in human movements and working styles. To have an idea about the relationship between the time window and the number of objects, we deploy video cameras in the living room, dining room, and the kitchen of two households. We analyze the recorded video frames and identify the objects that are used at any instant during 4 different activities: cooking, serving food, eating and cleaning – which are typical daily activities that use varieties of objects. In Figure 1(a), we plot the average number of different objects that are used in each of these activities within varying sizes of the time-window. This curve gives us an important insight i.e. how much a system can delay the image processing task, given the capability of an object recognizer. For example, if we plan to accumulate video frames for 500 seconds before starting to process them, we need a classifier that may have to deal with 20 different classes of objects on an average.

C. Objects are mostly stationary and their location-likelihood function is multi modal.

An object can be at different locations during its lifetime. But, if we analyze the locations of a specific object over a period, we see that it tends to stay at some limited number of specific areas. For example, a coffee mug is most likely to be at the study desk, or may be in the kitchen table, or in the kitchen sink. Knowing the likelihood location of an object adds value to the object recognition task. In our study, we divide each of the rooms in two households into a number of approximately 2 square meters zones and use RFID readers to monitor each zone separately. We record any location changes of a total of 60 objects for 3 days in these places. We compute the total time an object stays in each of these zones and analyze the modality of the distributions. Figure 1(b) shows the pie chart of the modalities. We see that more than 90% of the objects have modality of at most 3. This means that, when in doubt, an object recognition algorithm can use this knowledge to identify an object by analyzing the likelihood of its being at some location, or looking for the candidate objects in their other possible locations.

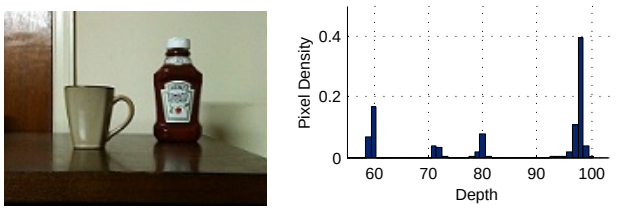
D. Idle time creates an opportunity for offline processing and learning.

There are many tasks involved in a system like Kinsight. Some of these tasks must be performed in real-time e.g., tracking human figures, detecting human-object interactions, and sampling. On the other hand, there are some tasks which are better done offline. These are mainly the image processing, classification and data mining tasks that operate on the data that are collected in real-time and require more computational time and resources. In Kinsight, we perform the real-time operations when the system is *busy*, and the offline analysis is done when the system is *idle*. We call a time interval 'busy' only if there is a change of location of any object during that interval. Figure 1(c) shows the average length of a busy time and idle time intervals for the same activities in Figure 1(a). We see that, for activities such eating, the average busy time is

as high as 80 seconds, and for activities such as cleaning, the average idle time is as small as 5 seconds. For a sampling rate of 200 samples per minute (enough for human activities), in this example, we have to process about 54 images per second, which is quite easily achievable with a modern processor.

E. Using only an RGB camera is not sufficient to detect household objects. Depth information improves detection accuracy.

Most image processing algorithms detect objects by sliding a window over the image to identify the regions that have interesting points, corners, edges, or changes in colors within it. These algorithms are sensitive to the image resolution, background, and clutter as they depend on the difference of intensities around each pixel. Having the depth information for each pixel greatly simplifies and improves these algorithms [14]. By looking at the depth values, it is easier to determine which pixel belongs to the object, which pixel is possibly occluding part of the object, and which pixel is part of the background. In Kinsight, we design and implement a novel image segmentation algorithm that uses the depth information to extract objects from an image. Figure 2(a) shows a simplified example where 2 objects are placed on a table at different depths. Figure 2(b) shows the distribution of the number of pixels at different depths. We see 4 different peaks within the plot corresponding to the edge of the table, 2 objects and the wall at the back, respectively. We identify these peaks and run image segmentation[2] algorithm on the pixels corresponding to each peak separately to extract one or more objects at the same depth.



(a) Original image (b) Pixel density

Fig. 2. Pixel density at different depths of (a) is shown in (b). The four peaks correspond to the table edge, cup, ketchup and the wall, respectively.

F. Context plays a vital role in detecting and recognizing household objects.

Context plays a vital role in recognizing household objects. This is specially true when the quality of an image is poor and the color values do not contain enough information to distinguish it from others. Although this assumption is true in general for many vision based systems [21, 23, 29], for household objects, this is more applicable due to the richness of available contexts. Sources of context that are used in Kinsight are: the time of day and light intensity for different appearances of the same object, depth information for determining the size and aspect ratio of an object, location history for determining the probability of an object being at a certain location, and the correlation among objects for determining the probability of their being used in the same activity.

[2]Partitioning a digital image into multiple segments based on pixel disparity.

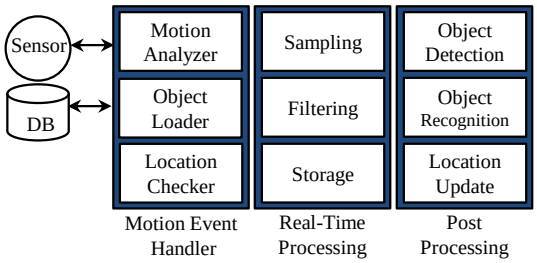


Fig. 3. Task distribution within a Kinsight node.

III. System Design

Kinsight consists of a network of depth-camera sensors. Each node within the network has its own sensor, processing unit and a local database. We assume that all nodes are stationary. A master node acts as the coordinator, which manages the central database and communicates to zero or more slave nodes. Figure 3 shows the task architecture within a node. Tasks performed at each node are divided into four operational stages: sensing, motion event handing, real-time processing and post processing. This section describes these stages in brief.

A. Sensing

Kinsight uses image, depth, and light intensity sensors. The image and depth data are obtained using Kinect [30]. The range limit of a Kinect sensor is approximately 11 feet. This range can be increased or decreased using special lenses such as [5]. Kinect connects to a PC via USB and multiple Kinects are connected to a single PC. The image and depth data are read as two separate streams. The image stream has a resolution of 640×480 and provides 32-bit colored images at 30 fps. The depth stream has a resolution of 320×240 and provides 16-bit depth value for each pixel. Kinect also annotates the pixels that are part of a human body and tracks up to 20 body-joints. The light intensity sensor on a MICA2 sensor board (MTS310) gives us the light intensity.

B. Motion Event Handling

The *Motion Analyzer* detects the presence and movement of any human figure within the camera's view using Kinect's skeleton tracking engine. It creates a new session for each detected skeleton and tracks the 3D coordinates of the body-joints in real-time until the end of the session. Based on the relative positions of the joints, it determines the 3D volume of human-object interaction. The *Object Loader* loads all the objects within this volume into the memory from the local database. These are the objects that might change locations during this session along with other objects that are not yet discovered. As the human figure moves, new objects are loaded into the memory. The *Location Checker* keeps checking for any location changes of these in-memory objects in a round-robin fashion and updates the timestamps of the in-place objects. If an object is not present at its last recorded location, or it detects that the human subject is interacting with an object at this moment, the current session goes into the *real-time processing* stage. Otherwise, the session is terminated.

C. Real-Time Processing

When the current session is in real-time processing mode, it periodically samples the objects that the person interacts with and stores them in a list as a time series. Each sample is comprised of the timestamp, color and depth information, light intensity, and the 3D coordinate of the body joint of human subject. The color and depth information is taken from a rectangular window within the current image frame. This window (we call it- the *sampling window*) is identified based on the relative positions of the joints of the human skeleton. Consecutive windows are compared to filter out images that are too similar to the previous ones or the quality of the image is very poor. The filtered samples are then added to a list for analyzing during the *post processing* stage. A transition to the post processing stage happens when the motion analyzer detects a long absence of motion within the view (e.g. the human subject moved out of the scene).

D. Post Processing

The post processing stage performs offline analysis to detect, classify and localize objects. At first, it extracts the objects from the samples by removing the pixels that are not part of an object (e.g. the pixels from the background, or any part of the human body). This is done using the *Depth Sweep* algorithm (Section IV-B). It then applies *k-means* clustering algorithm to cluster the samples. Ideally, k should be equal to the number of objects that changed locations during the session. But since we do not know this ahead of time, we set k to a larger value, determined by the sequence of presences and absences of objects within the time series.

After the clustering, we get a fair amount of samples in each cluster. The samples within each cluster are then matched against the existing objects in the database. A context oriented object recognition algorithm (Section IV-C) is used to find a match. For each cluster, if a match is found, the new samples are added to the existing object; otherwise, a new entry is created in the database and the samples are added to it. The location of an object is computed from the relative position of the object within the sampling window, depth values of the object, and the 3D coordinates of the body-joint (stored during sampling) of the interacting person.

The post processing stage also adds *activity context* to each object. An activity context is a list of $< object, score >$ pairs, where the score corresponds to the portion of time an object is in use during a session. The score is computed from the number of objects that are in each cluster. For example, if the human subject interacts with his coffee mug, milk jar, and sugar bottle during the session, and we have 600, 300, and 100 samples of these 3 objects respectively, we define the context as: $c = \{< mug, 0.6 >, < milk, 0.3 >, < sugar, 0.1 >\}$. We look up the context definition library, and either we find a similar context that has the same set of objects with very close scores, or we add this new context definition to the library. The context is then associated to the newly added objects into the database for later use.

IV. Algorithms

A. Sampling Window Estimation

The most common technique for detecting objects within an image is to slide one or more windows of different sizes and scales, and to determine which of these windows contain an object [32, 34]. This method is not suitable in a real-time setup due to its computational cost and latency. Instead, we use a trained linear regression learner to determine the size and position of the sampling window y in $O(1)$, in real-time. We use the 3D coordinates of the spine, hip, shoulders, knees, hands and wrists, and the sizes and depths of the in-memory objects to compute a 32×1 sized feature vector x. Assuming y a 4×1 vector, where the 4 elements correspond to the width, height, and the left-top coordinates of the sampling window, we estimate y by $\hat{y}(\theta) = \theta \times x$. Here, θ is a 4×32 matrix, which we learn using linear regression with a least-squared-error cost function.

B. Depth Sweep and Object Extraction

The Depth sweep algorithm takes image and depth data as inputs and performs image segmentation to identify possible objects within a scene. This algorithm is applied during the object detection and extraction in the time series analysis. Unlike most segmentation algorithms [9, 16] where the entire image is treated as a whole, we use depth information to process a subset of the pixels. Often the reduction in the number of candidate pixels is $50 - 100$ times (e.g. 640×480 vs. 64×64), which substantially reduces the running time of the segmentation algorithm and improves efficiency.

The steps of the algorithm are the following:

Step 1: Compute a B-bin histogram of the number of pixels for the entire depth range, and normalize it to obtain the pixel distribution $h(d_i)$.

Step 2: Apply EM algorithm [26] to fit $h(d_i)$ to a mixture of Gaussian distributions.

Step 3: For each Gaussian (μ_i, σ_i), take the set of N pixels within the depth range $[\mu_i \pm 1.96\ \sigma_i/\sqrt{N}]$, to perform image segmentation using [9]. Note that, multiple objects are extracted from the same depth range in this step.

In our implementation, we use $B = 100$, and set the parameter of EM algorithm to 10. An optimization in applied in Step 3 when extracting a single object from a sampled image during the time series analysis. Since we know the depth of the interaction point (e.g. depth of wrist), we pick the Gaussian whose mean is the closest to the depth of the interaction point, and apply image segmentation only once to extract the object.

C. Context Oriented Object Recognition

The object recognition algorithm takes an image sample s, represented by the RGB pixel values p, current 3D location l and activity context c as inputs, and matches the sample with the objects $\{v_j\}$ in the database V. It determines the most likely class v_{NB} of s using a Naive Bayes classifier using the

following equation:

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j|p,l,c) = \underset{v_j \in V}{\operatorname{argmax}} P(p|v_j)P(l|v_j)P(c|v_j)P(v_j) \quad (1)$$

Here, we assume all objects $v_j \in V$ are equally probable and the pixel values, location, and activity context are independent. This might not be always true, e.g., some activities are done at a particular location in some households. But having those independent makes the system more flexible.

We obtain $P(p|v_j)$ by computing the similarity of the feature vectors of the two images using Bhattacharyya Coefficient [6]. The 1×40 feature vector contains three 8 bin color histograms corresponding to 3 color channels, and one 16 bin histogram corresponding to key-points generated using the technique in [22]. We use the depth and light intensity values to select the appropriate object from the database.

To compute $P(l|v_j)$, we divide the view volume of the camera into unit sized equiprobable cubes $\{l_k\}$ and determine which cube l falls into. From the location history of each object $v_j \in V$, we compute $P(l = l_k|v_j)$ by counting the number of occurrences v_j ever visited location l_k.

We compute $p(c|v_j)$ by taking the normalized contextual score associated with c for v_j in the activity context definition library C. But finding c in the first place is a bit tricky since according to the definition of activity context (Section III-D), we require to know the classes of all the clusters in order to compute the context. Hence we follow the following steps to compute c:

Step 1: For all samples s_i, we perform an initial class assignment using $P(c|v_j) = 1$ in Eq. (1) and store the probability $P(v_j|p,l,c)$ along with the class labels.

Step 2: Using the class labels found in Step 1, we compute a new activity context c'. Let, $c_m \in C$ be the context that minimizes $||c_m - c'||$.

Step 3: For each sample, we again compute its class using context c_m in Eq. (1), and compute the degradation of probability obtained in Step 1.

Step 4: If the mean degradation over all samples is less than a threshold α, we choose c_m as the context; otherwise we insert c' in C as a new context and use c'.

The idea here is to find an initial context c' using only the location and pixel values, and use it to find the closest context c_m. Note that, c_m will always be a looser fit than c' since $p(c_m|v_j) \leq 1$. The value of α controls how much we would rely on an existing context, as compared to having a new context. We pick an α of 0.1, a small value, so that Kinsight is not confused by two similar contexts that are actually from different activities.

D. Occlusion and Multiple Views

A single depth-camera provides only one view which may not be enough to continuously track the objects due to occlusions. More than one sensor provides more views, more samples and more accuracy in object recognition and localization. In Kinsight, each node independently detects, recognizes and localizes the objects within its view. Once the session is over, the master node queries other nodes to send their local database information. This includes the updated locations, samples and contexts of all the objects that are stored in the local database. Objects from the slave nodes are combined with the master's by matching the location and timestamps of each object. The locations obtained in a slave node's coordinate system is converted to the master node's coordinate system using an affine transformation. The transformation matrix is automatically computed by Kinsight during the first few seconds of the very first session when the system starts. Both the Kinects during this period record the 3D coordinates of a body joint (e.g. head) and the recorded coordinates are then collected at the master node to compute the transformation matrix.

E. Objects with Multiple Copies

Multiple copies of the same object are treated different by Kinsight. This is inherently handled by design, as to Kinsight, an object's identity includes its current location along with its image and contextual information. Despite this, a possible way to trick Kinsight is to mask its view and then swap the locations of two copies of the same object. To handle such a situation, when an in-memory object, loaded into the memory by the object loader, is out of sight of the location checker, it is removed from the database prior to entering into the real-time stage. Later at some point, when the user again interacts with the object, it is added to the database as a new object.

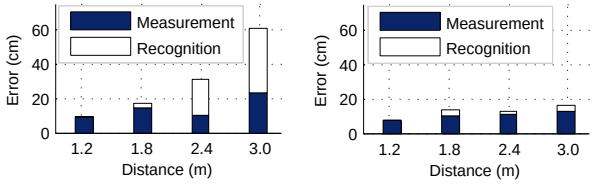
V. Experiments

We conduct a number of experiments to evaluate the performance of Kinsight. These experiments are performed using Kinect sensors connected to a laptop having a 2.3 GHz Intel Core $i5$ processor and 4 GB RAM. We label 48 household objects and 80 locations with numeric tags, and ask the human subjects to move objects according to randomly generated scripts. The list of items includes personal items (e.g., phone, wallet, keys), stationary (e.g., pen, boxes, stapler), utensils (e.g., cups, bottles, pots), toys (e.g., dolls, cars), and entertainment (e.g. xbox, remote controller). A subset of these items are shown in Figure 4. Location contexts of these items are generated following the distribution as in Figure 1(b), and activity contexts are generated by restricting the number of objects for each session. This is done to control the variations of different variables to see how they influence the system. For each movement of an object, Kinsight classifies the object and stores its identity and the location. Localization error is measured by taking the Euclidean distance between the actual location of an object and the location where Kinsight thinks the object currently is. Localization error is thus generated from two sources: (1) *measurement* error (when the object is classified correctly), and (2) *recognition* error (when the localization error is due to misclassification).

Fig. 4. A subset of the objects that are used in the experiments are shown.

A. Distance

The farther an object is from the sensor, the less number of pixels we get from it. This affects the quality of the sampled image and produces localization error due to poor recognition in absence of contextual information. This is evident from Figure 5(a) where we see that, the localization error increases with distance, but the increase is due to recognition error. Figure 5(b) shows that, when we apply context, the recognition error becomes $8-10$ times smaller, the measurement error becomes mostly constant with distance, and overall we have an average localization error of 12.54 cm. Note that, context has an indirect effect on the measurement error. For example, at 3.0 m distance, we see about 65% reduction in misclassification in 5(b) than in 5(a). This results in more data points for the measurement error case in 5(b) than in 5(a) and hence we observe a different and more confident mean for the measurement error in 5(b).



(a) Without Context (b) With Context (Kinsight)

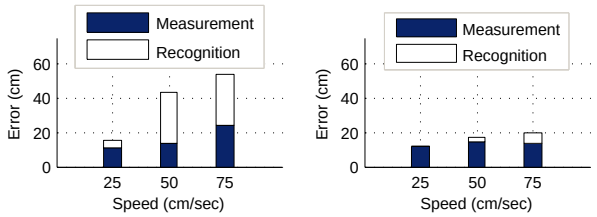
Fig. 5. Effect of distance from camera.

B. Speed

The speed of limb movement has adverse effect on the accuracy of human figure tracking. Kinsight alleviates this problem to some extent by tuning the smoothness parameters of Kinect's skeleton tracking engine[4]. But, for a given sampling rate, we get a lower number of samples with increasing speed. This affects the image recognition task as we have a smaller number of samples for each object. Figure 6(a) quantifies this error by plotting the localization error at various speeds when we do not use context. We see that, as the speed grows beyond 50 cm/sec, Kinsight makes more classification errors, which results in higher localization error. Figure 6(b) shows that the recognition errors are $5-11$ times reduced when we use context.

C. Sampling Rate

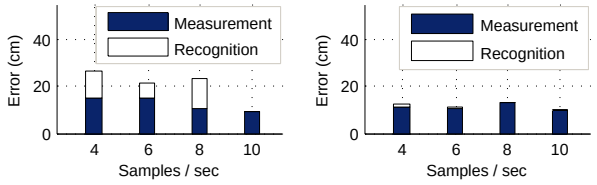
Kinect provides skeleton data at 15 frames/sec which is the maximum sampling rate Kinsight can achieve. This gives us enough data to reduce localization error due to misclassification. But with this highest rate, we accumulate on average



(a) Without Context (b) With Context (Kinsight)

Fig. 6. Effect of speed of movement.

120 MB of data per second during the real-time sampling stage, which may take several seconds to process afterwords. We therefore analyze the trade off between the sampling rate and localization error. We control the sampling rate by dropping frames. Figure 7(a) shows that both portions of the localization error are reduced when the sampling rate is as high as 10 samples/sec. But the same error margin is achieved in Figure 7(b) at a lower frequency of 4 samples/sec when we use context.



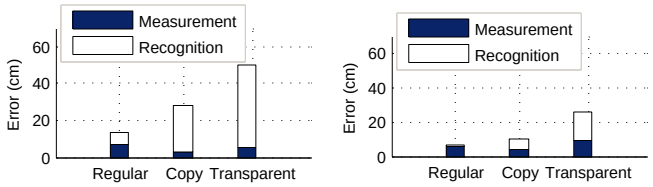
(a) Without Context (b) With Context (Kinsight)

Fig. 7. Effect of sampling rate.

D. Appearance

To see the effect of color and size variations, we test Kinsight with multiple sets of objects, each having a different degree of color and size variation. In Figure 8, we take 3 sets of objects: (1) regular – household objects except for the duplicates, non-rigid, and transparent ones, (2) copy – objects that have multiple copies, and (3) transparent – objects that are transparent or non-rigid. In absence of contextual information, (2) and (3) makes classification error of 28% and 60%, respectively. By using contexts, we are able to eliminate 16% errors from (2), and 30% from (3). The remaining errors in (2) are due to the fact that some of the copies were too close (within 13 cm) to each other to be distinguishable by Kinsight. And in (3), the remaining errors are caused by the transparent objects that were not properly visible by the camera.

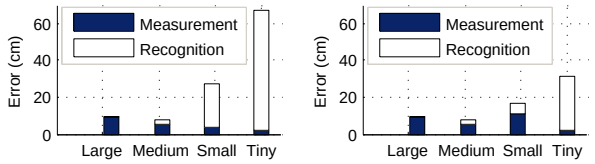
In Figure 9, we take 4 sets of objects: large, medium, small, and tiny, having average dimensions of $30\times24\times19$, $21\times12\times7$, $11\times9\times8$, and $8\times4\times3$ cm^3, respectively. For the first two



(a) Without Context (b) With Context (Kinsight)

Fig. 8. Effect of color variation.

sets, the mean localization error of Kinsight is 8.84 cm. For small objects, Kinsight reduces the misclassification from 47% to 13%, and keeps the localization error within 16 cm. But for tiny objects, although the misclassification is reduced from 74% to 46% when we use context, the remaining errors are still significant. These errors are caused by those tiny objects that were more than 2.5 m away from the sensor, and therefore, we do not get enough pixel information from them.

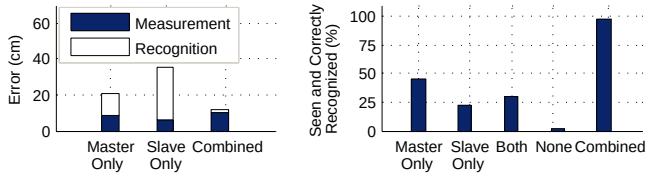


(a) Without Context (b) With Context (Kinsight)

Fig. 9. Effect of size variation.

E. Multiple Views

So far in the experiments, we have used only one Kinect sensor. But multiple Kinects might be beneficial in some environments where there are high chances of occlusions. For example, to monitor an L-shaped region, we place two Kinsight nodes (a master and a slave) at the two ends of the region. The master node has a greater view area, and the slave node helps the master to monitor the portion that is occluded to the master. Figure 10(a) shows the localization error by only the master only, only the slave, and the combined system. We see that the combined network has a very high accuracy in localizing the objects. The reason is explained by Figure 10(b). The master alone sees and correctly classifies 46% of all the objects, 22% are seen and correctly classified by the slave, 30% are correctly classified by both, and only 2% are missed by both. Using this two node system, 98% of the objects are correctly recognized – yielding to a localization error of 11.7 cm.

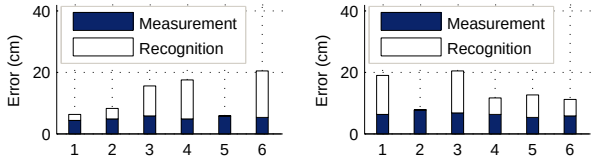


(a) Localization Error (b) Coverage

Fig. 10. Using multiple views.

VI. Real Deployment

We deploy Kinsight in two multi-person households to evaluate its performance in real-world setups. We conduct experiments in the living room, study room, and the kitchen of the households. The largest room has the dimension of 11×10 square feet. We attach about 70 paper tags on the surfaces that may hold an object (e.g. tables, sofas, floor, walls), measure and note the coordinates of the tags, and video-tape the daily activities of the human subjects to obtain the ground truth. The activities are – (1) cleaning, (2) coffee making, (3) cooking, (4) eating, (5) entertainment (watching tv, playing indoor games, talking on phone), (6) studying and working on computers. These activities are chosen since they are very common in every household, and such activities are often studied for various reasons e.g. activity recognition [13] and wellness [36]. The duration of these activities are about $5-20$ minutes. Each activity is performed at least twice. Later, we analyze the video-tapes manually to note the objects and their locations (the location of the nearest tag) every 5 seconds. We use one Kinect per room, and program Kinsight to log the objects (a sample image per object) and their locations every 5 seconds.



(a) Household 1 (b) Household 2

Fig. 11. Localization errors for six activities: (1) cleaning, (2) coffee making, (3) cooking, (4) eating, (5) entertainment, and (6) study.

Figure 11(a) and Figure 11(b) show both sources of the localization errors for the six daily activities in the two households, respectively. In household 1, the best results are obtained during entertainment, cleaning, and coffee making activities. These activities in this household use a small number of $(3-5)$ easy to distinguish items, e.g., game controller, Rubic's cube, coffee cups and maker, dish wand, plates, etc. Larger errors are observed in cooking, eating and studying, as these activities involve a large number of small items, such as, pens, sharpener, spoons, forks, spice bottles, etc. In household 2, we see similar trends as in household 1 for coffee making, entertainment, and cooking. But the other activities e.g. cleaning and study show different trends. The differences occur as inhabitants in household 2 use different types of objects such as transparent bottles (washing liquid) for cleaning, books and laptop for studying, and have a different way of handling objects. Overall, the average localization errors in these two households are 12.13 cm and 13.55 cm, respectively.

VII. Discussion

Kinsight achieves its best performance when it has a good view of the object. We observe a performance degradation when the objects are either transparent or very small and far. This is a common limitation for any vision based system; Yet, Kinsight is useful in a household setup where a vast majority of the objects are not from any of these categories. Losing track of the objects due to occlusion or hiding might happen

if the camera position is odd. Suitable placement of the camera or adding more sensors alleviates the situation, but may not solve the problem completely. However, even in those cases, Kinsight is able to tell where the object was last seen, which we think is also valuable. Although Kinsight is built on Kinect sensors, the design and principles of Kinsight are applicable to any depth-camera sensor system. The present Kinect sensor has a range limit of 11 feet. But the amount of attention it has received from the research community, we believe, it will encourage several manufacturers to build more powerful hardwares in coming days, and more and more systems like Kinsight will appear.

VIII. Related Works

There are numerous works in computer vision research pertaining to object detection, recognition and tracking. We mention a few that are recent, and refer readers to [37] for a survey. [12, 15, 16] detect objects by performing image segmentation and contour detection. These are detectors for general objects and do not use any contextual information. [21, 23] use contexts to improve object recognition accuracy, but it is either supplied during training or extracted from the image itself. [24, 33] are model based approaches for tracking single objects in real-time; [8] tracks multiple objects simultaneously, but is not real-time. The differences between these works and ours are that, ours is a more specialized system dealing with only household objects, we learn object instances (not class), and we go beyond images to use location and activity contexts.

Some recent works use depth data along with RGB image for human pose estimation [30], illumination invariant tracking [27], 3D mapping for mobile robots [18], and human activity detection [31]. [19] describes a template matching algorithm that uses depth to detect objects in images. But they require to search the whole image for a match, while Kinsight selects only a subset of pixels that are close to the depth of human-object interaction point. [7] uses depth to enhance object tracking. But unlike Kinsight, they track each object separately.

Localization with RFID [20, 25, 28, 35] and WSN [10, 11, 17] technologies have been studied by many. There are also a number of commercial products that uses RFID or similar technologies [1, 2]. But these systems are intrusive, i.e., require us to attaching tags to every object we want to keep track of, and the readers are expensive.

IX. Conclusion

In this paper, we describe Kinsight which uses a depth-camera sensor network to detect and track household objects' locations. Kinsight discovers objects from human-object interactions, uses unsupervised learning techniques to recognize objects from their appearance, location history and activity contexts, and updates the locations of the objects. We evaluate Kinsight in both controlled and uncontrolled environments to quantify its sensitivity to a wide range of parameters and to demonstrate its practicality. In real-world scenarios, Kinsight's average localization error is about 13 cm.

References

[1] Finder. yankodesign.com/2011/11/03/keycontrol/.
[2] KeyRinger. keyringer.com.
[3] RFID FAQ. rfidjournal.com/faq/20/86.
[4] Skeletal Tracking Fundamentals. channel9.msdn.com/Series.
[5] Nyko zoom lets you play kinect in a closet. *PCWorld*, Sept. 2011.
[6] Aherne, F. et al. The Bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika*, 1997.
[7] C. Bibby and I. Reid. Real-time tracking of multiple occluding objects using level sets. In *CVPR*, June 2010.
[8] W. Brendel, M. Amer, and S. Todorovic. Multiobject tracking as maximum weight independent set. In *CVPR*, June 2011.
[9] T. Chan and L. Vese. Active contours without edges. *Image Processing, IEEE Transactions on*, Feb 2001.
[10] H. Chenji and R. Stoleru. Mobile sensor network localization in harsh environments. In *DCOSS*, June 2010.
[11] J. Eckert, R. German, and F. Dressler. Alf: An autonomous localization framework for self-localization in indoor environments. In *DCOSS*, June 2011.
[12] I. Endres and D. Hoiem. Category independent object proposals. In *ECCV*, 2010.
[13] J. W. et al. A scalable approach to activity recognition based on object use. In *ICCV*, Oct. 2007.
[14] Q. W. et al. Reduced-complexity search for video coding geometry partitions using texture and depth data. In *VCIP*, Nov. 2011.
[15] S. Gould, T. Gao, and D. Koller. Region-based segmentation and object detection. In *NIPS*, 2009.
[16] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *Proc. of ICCV*, 2011.
[17] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proc. of MobiCom*, 2003.
[18] E. Herbst, P. Henry, X. Ren, and D. Fox. Toward object discovery and modeling via 3-d scene comparison. In *ICRA*, may 2011.
[19] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *ICCV*, Nov. 2011.
[20] L. Ni and D. Zhang and M. Souryal. RFID-based localization and tracking technologies. *Wireless Communications, IEEE*, April 2011.
[21] Y. J. Lee and K. Grauman. Object-graphs for context-aware category discovery. In *In Proc. of CVPR*, June 2010.
[22] V. Lepetit and P. Fua. Towards recognizing feature points using classification trees. Technical report, 2004.
[23] C. Li, D. Parikh, and T. Chen. Extracting adaptive contextual cues from unlabeled regions. In *Proc. of ICCV*, 2011.
[24] B. Liu, J. Huang, L. Yang, and C. Kulikowsk. Robust tracking using local sparse appearance model and k-selection. In *CVPR*, June 2011.
[25] X. Liu, M. D. Corner, and P. Shenoy. Ferret: Rfid localization for pervasive multimedia. In *Ubicomp*, 2006.
[26] G. McLachlan and T. Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley and Sons, 2008.
[27] H. Nanda and K. Fujimura. Visual tracking using depth data. In *CVPRW*, june 2004.
[28] B. Nemmaluri, M. Corner, and P. Shenoy. Sherlock: automatically locating objects for humans. In *Proc. of MobiSys*, New York, NY, 2008.
[29] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *ICCV*, oct. 2007.
[30] J. Shotton, A. FItzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *proc. of CVPR*, June 2011.
[31] J. Sung, C. Ponce, B. Selman, and A. Saxena. Human activity detection from rgbd images. *CoRR*, 2011.
[32] A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. *PAMI*, may 2007.
[33] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *PAMI*, Oct. 2004.
[34] P. Viola and M. Jones. Robust real-time object detection. In *IJCV*, 2001.
[35] C. Wang, H. Wu, and N.-F. Tzeng. Rfid-based 3-d positioning schemes. In *INFOCOM*, May 2007.
[36] A. Wood, J. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru. Context-aware wsn for assisted-living and residential monitoring. *IEEE Networks*, July/Aug 2008.
[37] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38, December 2006.