Recognizing Object Manipulation Activities Using Depth and Visual Cues

Haowei Liu*, Matthai Philipose†, and Ming-Ting Sun*
*University of Washington, Seattle, US
E-mail: {hwliu, mts}@uw.edu
†Intel Labs Seattle, Seattle, US
E-mail: matthai.philipose@gmail.com

***Abstract*— We present the design of an approach to recognize human activities that involve manipulating objects. Our proposed approach identifies objects being manipulated and models high-level tasks being performed accordingly. Realistic settings for such tasks pose several problems for computer vision, including sporadic occlusion by subjects, non-frontal poses, and objects with few local features. We show how size and segmentation information derived from depth data can address these challenges using simple and fast techniques. In particular, we show how to robustly and without supervision find the manipulating hand, properly detect/recognize objects and properly use the temporal information to fill in the gaps between sporadically detected objects, all through careful inclusion of depth cues. We evaluate our approach on a challenging dataset of 12 kitchen tasks that involve 24 objects performed by 2 subjects. The entire system yields 82%/84% precision (74%/83%recall) for task/object recognition. Our techniques outperform the state-of-the-art significantly in activity/object recognition rates.**

I. INTRODUCTION

Many day-to-day human tasks involve the manipulation of objects. Medication regimens for elders, maintenance sequences for factory workers, experimental protocols for laboratory technicians and recipes for home cooks are useful applications that people have tried to develop. Details to be monitored include the identity of the task being performed (e.g., medicating), the object being manipulated (e.g., pill box, cup) and the manner in which it is handled (e.g., unscrewing the lid, picking and placing the medicine bottle). A restricted but broadly useful setting for such tasks, adopted in most preceding work, is a fixed indoor work area such as a counter, table or floor. In this paper, we propose an approach aimed at inferring details of manipulation-based tasks in such settings, with particular emphasis on addressing the challenges posed by realistic deployment scenarios.

Work over the past decade, e.g. [1] has shown that jointly reasoning about the task being performed, the identity of object being manipulated and the hand actions being performed on the object can significantly improve the accuracy of all three. Such reasoning requires recognition of the objects being manipulated and the (3-D) pose of the hand manipulating them. However, day-to-day indoor settings pose substantial challenges to solving these problems, including variability in pose and attributes of users and objects used, severe sporadic occlusion by users and the environment, a scarcity of perspective cues to judge depth, heavy clutter in the environment and working surfaces, fast motions and variable lighting.

One way in which recent approaches have handled these challenges is to identify scenarios that avoid some of them. For instance, users are required to face the camera such that the manipulated object is always visible, objects are required to have abundant SIFT features, or work surfaces are structured so that objects occlude each other minimally. However, many natural manipulation scenarios exhibit all the above challenges routinely. In fact, the interaction of objects with hands can have a mutually confusing effect in that hand occlusion and motion may make object recognition hard, and handled objects may make hand/arm detection difficult.

Kinect [2] has been a wide success since its launch yet most of its applications are on gesture control for gaming. In this work, we examine how to use the depth images generated by Kinect, to address the above problems. The use of depth derived from a non-visual channel has several advantages. First, a rapid depth-based segmentation of the scene can usually identify users holding objects regardless of the vagaries of lighting, clothing variability, and clutter. Second, the depth image yields object size and shape as cues that can be used for recognition even if motion blur and lack of key points make local features less informative. Third, distances between different entities such as hands and the environment are directly measurable, making it easier to detect and characterize hand-environment interactions.

In what follows, we present the design and evaluation of an algorithm that makes comprehensive use of depth cues to supplement visual cues in various aspects of understanding manipulation. The main contributions of the paper are:

- A novel algorithm based on local spatial statistics to parse a 3D-cloud representing the human body into torso and arms even under heavy occlusion.

- A technique to use the location of the arms to locate the object being held, and to incorporate size-based features of the object into a suitable object recognizer.

- A temporal super-segmentation scheme based on combined depth and visual cues that identifies long stretches of frames where the object being manipulated is highly unlikely to

change. Using these stretches as the units of object and activity recognition improves both significantly.

We evaluate our approach on a realistic dataset with 12 activities, 24 objects and 2 subjects with various lighting conditions and poses, to analyze the value and show the effectiveness of the above techniques.

II. Related Work

A number of researchers [1, 3-6] over the past decade have shown the utility and feasibility of understanding object manipulation. Their emphasis has been on demonstrating the value of inferring objects, actions, and activities together. We instead focus on improving the performance of lower-level detectors for these quantities by using depth cues.

Understanding manipulation typically requires tracking arms and detecting/recognizing objects being manipulated. Past systems have made various simplifying assumptions to reduce the demands on these capabilities. Moore et al. [6] and Gupta et al. [3] do not attempt to detect and recognize objects when they are in the hand, but rather rely on detecting the point of contact with plainly visible objects on the work surface. We believe that given natural occlusion and clutter, relying solely on on-surface detection of objects is insufficient. Like Kjellstrom et al. [7] and Wu et al. [5], we therefore detect and recognize objects as they are being used. Unlike Kjellstrom et al. [7], however, we do not assume that we have a frontal view of the task space, or that the manipulation happens so that the object being manipulated is mostly visible to the camera. We assume sporadic visibility is the norm and use a temporal super-segmentation technique derived from our depth data to cope. Unlike Wu et al. [5], who rely on SIFT features to avoid having to segment out the precise outlines of the manipulated object, we exploit the shape, texture, size, and other region-based cues for object recognition. Many common objects do not have useful SIFT-style features due to their coloring, material, and motion. We use our depth-based segmentation to determine the region representing the manipulated object, including the size of the region.

To find the pose, most systems typically perform a multi-scale, multi-orientation search for candidate body parts followed by imposing constraints on the part connectivity [7-9]. [2] is by far the most successful pose estimation system in terms of speed and performance. However, it does not model the scenarios when users are holding objects. Also, in our application, we only need arm locations instead of detailed pose information. Given a 3-D point cloud of the user, we show how to use local spatial statistics to determine whether each patch on the user belongs to torso or limb. Our local, bottom-up technique does not rely on having a view of the head and shoulder as a starting point for parsing the 3-D cloud as these body parts are often not visible in manipulation footage. On the other hand, we do not provide full upper-body kinematic modeling (we simply identify arms and torso) or enforce consistency globally across parts.

III. System Design

Fig. 1 shows the overall structure of our system. Given an incoming 640x480 frame of depth values, we detect the likely human(s) in the frame, parse them into torso and arms, identify the hands and extract pixels corresponding to the handled object, and classify the object based on the visual cues from these pixels combined with size cues. We use the position of the hand and the overall structure of the scene to determine when the hand is close to environmental surfaces. When combined with pixel information on whether the handled object has changed, we are able to conservatively infer whether an object has been picked up. We segment the incoming stream at these predicted pickup events, and classify entire segments based on which object we think is being used in that segment (we currently assume one object is manipulated at a time). Given a stream of segments annotated with object use, we use a Hidden Markov Model (HMM) to infer activity over time. The following sections provide more detail.

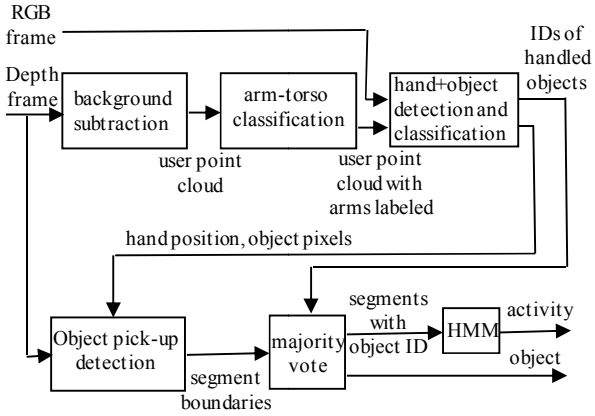


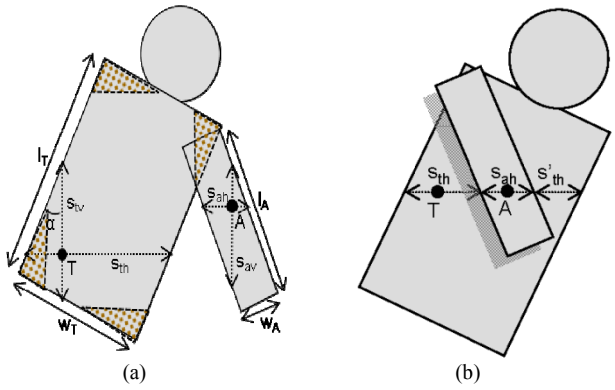
Fig. 1 High-level system design

A. Detecting People and Arms: Background Subtraction and Local Spatial Statistics

Given an incoming depth-map of the activity scene as in Fig. 2, we seek to identify the points representing the person performing the activity. We take the traditional point of view that the person is a foreground object. Given a frame representing the current scene and a frame representing the background, we subtract the latter from the former to get the foreground objects. Foreground objects that meet the size requirement then identify the humans.

(a) RGB (b) depth

Fig. 2. Synchronized depth and RGB images. Image (b) is a depth map of (a), with lighter shades closer to the camera.



(a) (b)

Fig. 3. Spatial support for points on the torso and arm without (a) and with (b) occlusion.

Given a point cloud representing the (upper) body of a human, we would like to identify points corresponding to arms and torso. Only partial views of the torso may be available and the head is often truncated or completely occluded. Fig. 3 (a) illustrates the central idea behind our algorithm. For an arbitrary point in the upper body (e.g., point T or A in Fig. 3 (a)), we compute the shortest distances from it to the depth boundary in different directions. The histogram formed by these distances will be different for points from different parts due to the size difference. For example, the histogram for a point from the torso would in general consist of larger distances than those from the arm since the torso occupies much more spaces than the arms do.

We take a learning based approach to implement the above idea. Denote the point cloud of n points as $\{< p_i, D(p_i) >\}_{i=1}^{n}$, where p_i is a 2-D vector, indicating the location of the ith point in the depth image D and $D(p_i)$ represents the depth reading at p_i. We first define the depth boundary as those points with p_i's lying on the foreground boundary or p_i's such that $|D(p_i) - D(p_j)| > t_d$ where p_j is a neighbor of p_i in the depth image and t_d is a user defined threshold. For each point with location p, we compute the distances from p to the closest points on the depth boundary in s uniformly spaced directions. For example, in Fig. 3 (a), for point T, we compute the distances from T to the depth boundary in eight directions, resulting in eight distance measures, i.e., L_{Td_r} r=1~8, where L_{pq} represents the length of the line segment $\overline{pq}$. Note that these distances are normalized by $D(p_i)$ to get the actual estimate.

In order to capture the surface variations of different parts, we incorporate the curvature features as follows. For a point with location p, and closest boundary points d_r's, we partition each $\overline{pd_r}$ into m equal-length line segments. Denote the set formed by the points partitioning $\overline{pd_r}$ along with d_r as $\{C_{rj}\}_{j=1}^{m}$, where $C_{rm}=d_r$, we compute $\{D(C_{rj}) - D(p)\}$ for r=1~s, and j=1~m as the curvature features. For example, for point T and $\overline{Td_3}$ in Fig. 3 (a), $D(C_{31})$-$D(T)$, $D(C_{32})$ -$D(T)$, $D(C_{33})$ -$D(T)$, and $D(d_3)$ -$D(T)$ are the curvature features.

Fig. 3 (b) illustrates the main complication: occlusion may make it hard to get a true estimate of the distance measures. In particular, an arm in front of the torso will present a surface that is over t_d from the torso surface and hence manifests itself as depth boundaries. In this case, the additional depth boundary makes the actually distance measure l shorter (becomes L_{AX}). This may make point A be erroneously labeled as an arm patch. For correctness, we want this distance measure to be L_{AX}+L_{XY}+L_{YB}. We obtain this by compensating the occlusions as follows. When computing the shortest distances to the depth boundary for a point with location p, we ignore those depth boundaries posed by the surface with depth readings smaller than $D(p)$. For example, for point A in Fig. 3 (b), when computing the shortest distance in the horizontal direction, instead of using L_{AX} as the distance measure, we use L_{AX}+L_{XY}+L_{YB}, since the arm-torso boundaries posed by the arm has smaller depth readings than $D(A)$, indicating possible occlusions.

Finally, a hold out set is used to train a random forest classifier [10] using these features. During the testing time, each point is independent evaluated and labeled with the classifier. We filter out isolated misclassified arm patches by requiring a connected set of arm patches to have area at least 60 cm^2 before considering it part of an arm. Our feature is compact (with dimension s*(m+1)) and training a random forest classifier is fast. Our proposed feature also generalizes to any trunk and limb system consisting of "squat" and long objects.

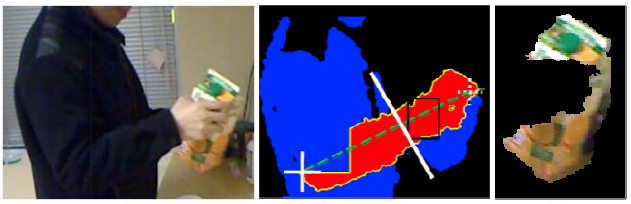
B. Detecting Hands and Recognizing Objects

Given a point cloud representing the (fore) arm, we seek to find the hand and extract the pixels of the object held by the hand. A standard way to find the hand is by looking for skin color. However, given that the forearm may be skin-colored, this in itself is not useful. Instead, we use the arm structure to hypothesize the likely location of the hand. Intuitively, the hand is the furthest point on the arm from the attach point of the arm to the torso.

We first compute the mean of the point cloud for the arm region. We define the attach point, shown as the solid cross in Fig. 4(b), as the limb-torso boundary point furthest from the mean. We designate the point on the arm boundary furthest from the attach point, shown as the dotted cross in Fig. 4(b), as the arm tip, and the attach-point-to-arm-tip line as the arm axis. If the user is holding an object (a common case), the arm tip may well be on the object, not the user's hand. We therefore also perform skin-color detection on the arm (i.e., on the RGB pixels derived from masking using the arm point cloud), project skin pixel coordinates onto the arm axis and designate the pixels with projected coordinates closest to the arm tip as the hand tip. Accurate location of the hand is important for object-pick-up/put-down detection.

We now turn to detecting the object. We expand the connected component of point clouds at the arm tip so as to include points that are not part of the arm, but rather part of

the object. We next remove all arm depth points corresponding to skin-colored pixels. This typically partitions the arm point cloud into multiple connected components. We exclude any points that project onto the arm axis such that they are more than 10cm away from the hand tip: these are shirt-sleeve points. The remaining components are used to mask out pixels for the object (see Fig. 4 (c)).



(a) original RGB frame (b) detected arm (c) object

Fig. 4. Hand localization result. (a) original RGB frame, and (b) hand localization result. The arm axis (dashed line) joins the arm tip indicated by the dotted cross and the attach point, the solid cross. The rectangle indicates the skin detection result. All the remaining pixels that are on the same side of the solid line as the detected hand region are treated as the object mask.

Our technique captures the shape and size of the object quite well, along with its basic color and texture. On the other hand, due to the nature of the objects, the distance of the camera and object motion, the objects typically do not have many local features. We therefore chose to use an object recognizer that makes good use of shape, size, and texture features [11]. The features used in [11] include the color histogram, segmented object mask, segmented object size, and textons. With depth, we could normalize the object mask, and size so that they are more representative. The original paper derived object region candidates by over-segmenting the image, whereas we use our depth map as discussed above. This recognizer also provides a useful confidence measure in its classification, a feature that is crucial to us given that certain views of objects are easily confused with others, and we provide machinery to pool information from confidently classified frames to the harder-to-classify ones.

C. *Countering Sporadic Visibility: Temporal Super-segmentation*

We anticipate that our system will only get sporadic good views of the object being used. For instance, if our object recognition system declines to classify half its input (under 50% recall), it has over 90% accuracy on the remaining examples. We therefore seek to infer entire segments where the user is likely to use a single object. Given these segments and a few (consistent) votes on the object used, we could label the whole segment.

In order to achieve this, given the hand and object localization from the previous section, we compute the distance to the tabletop of the lowest point of the hand and the object and the number of pixels in the object mask. We feed a window of these features derived from the last 10 frames into a boosted classifier [12] for recognizing whether a new object has been picked up. The intuition is that when the hand moves close to the surface, or when the number of pixels of the held objects changes, we have strong cues that a new object may have been picked up.

We do not expect to be able to predict pickup events with extremely high precision and recall. We therefore tune the classifier to have very high recall (so that pickup events are almost always detected), but modest precision, so that spurious pickup events are sometimes detected. In each of the resulting segments, we identify frames where our object recognizer has been classified with high confidence, and let these frames vote (by simple majority) as to which object is used in the segment.

Since each activity involves several objects, the sequence of objects used in segments are fed into a standard HMM framework to infer activities. The parameters of the HMM were not learned, but rather specified using "common sense values", mined from the how-to websites as in [4].

IV. EVALUATION

We now evaluate the effectiveness of our feature for arm-torso classification. To this end, we recorded a 10-minute footage using the 3-D camera, where a person is performing different actions. The person wears clothes of different colors to cover different body parts (torso and arms) so we can obtain the ground truth by applying a color detector. We uniformly sample the footage to create a dataset consisting of 1196 images (800 for training). The number of directions is set to eight when computing our feature. We follow the methodology in [2] where during training, a random set of 2000 points is selected per training image.

We train a random forest classifier of 100 trees trained on 200 images and evaluated on a subset of the testing images. We evaluate the classifier using 0, 2, 4, and 8 curvature features. Adding the curvature feature adds about 6\% improvement for torso-arm classification but the performance saturates at two samplings per direction. Next, we vary the percentage of the training images, ranging from 1 image/0.05%, 50 images/2.5%, and 100 images/5% ... up to 800 images/40%. The classifier achieves above 80% accuracy with 50 or more training images and 60% when trained using only one image. On average, with an OpenMPI implementation, our feature can be computed/evaluated within a few (3~5) milliseconds per frame.

Next, we seek to answer two main questions in the evaluation. First, how good is the end-to-end performance of the system? Second, how did the design choices contribute to the result? To answer these questions, we collected synchronized depth and RGB data at 30 fps in a simulated kitchen setting of the type pictured in Fig. 2. Two subjects performed 12 activities involving 24 objects. The activities were: a. make cereal, b. make soup, c. make coffee, d. make tea, e. make chocolate milk, f. wipe counter, g. tend plants, h. take vitamins, i. take antacid, j. drink juice, k. make buttered toast, l. make phone call.

Table I is the list of all the objects along with the associated activities in the dataset.

We made no special effort to pick objects with many SIFT features and subjects were instructed to perform activities naturally, and with no effort to face the camera, slow

manipulation of objects, keep the counter space tidy, place objects in particular locations, etc.

TABLE I OBJECT LIST WITH THEIR ASSOCIATED ACTIVITIES

1	cereal box (a)	13	bowl (a, b)
2	soup (b)	14	detergent (f)
3	coffee can (c)	15	kettle (b, d)
4	tea box (d)	16	sugar (c, d)
5	milk carton (a, c, d, e)	17	chocolate (e)
6	sponge (f)	18	coffee bottle (c)
7	miracle gro (g)	19	coffee cup (c)
8	vitamin (h)	20	water cup (h, i, j)
9	antacid (i)	21	tea cup (d, e)
10	juice carton (j)	22	watering-can (g)
11	jar (h, i)	23	bread (k)
12	butter (k)	24	phone (l)

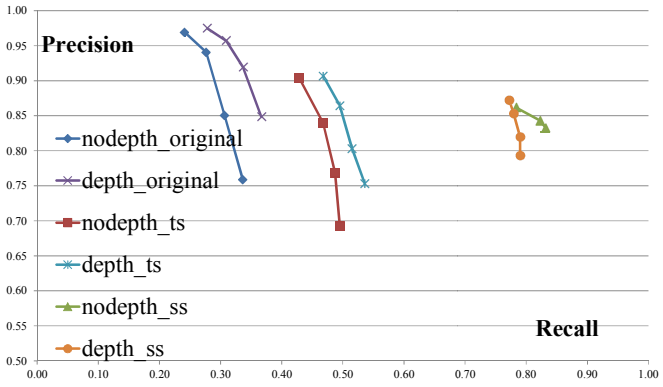


Fig. 5. Object recognition results.

We began by examining if we could get away without determining the boundaries of the objects and using SIFT features from a rough localization of the object. We used a standard Lowe ratio-test based recognizer [13]. The results were quite poor: object recognition rates averaged 23%. The mean number of SIFT features was 21 (median 17), whereas close-up still pictures of the coffee-can, milk carton, cereal box, and antacid yielded hundreds of features. Many objects such as the sponge, butter bowl, soup bowl, kettle, coffee cup, etc were specular or texture free.

We next investigated object recognition rates with our techniques enabled. Even with no temporal smoothing we get 85/37% precision/recall (P/R). We looked at the effect of turning on and off the object-size derived from the depth camera feature to the recognizer (depth-vs-nodepth), of using temporal smoothing (ts) to mitigate sporadicity of object recognition, and using super-segmentation (ss) for mitigation under different thresholds on the classifier confidence measure. As Fig. 5 shows, depth clearly makes over 10% difference in precision for a given recall, temporal smoothing (a state of the art technique) can improve rates substantially (as is well known, smoothing can resolve short glitches in inference results), but super-segmentation results in a much larger further improvement (per-frame object recognition rates improve up to 85/84% P/R). The reason is that many gaps in object recognition are long: over 70% of activities had runs of over 30 frames where our classifier declined to classify. Frame-to-frame smoothing fails with such long gaps.

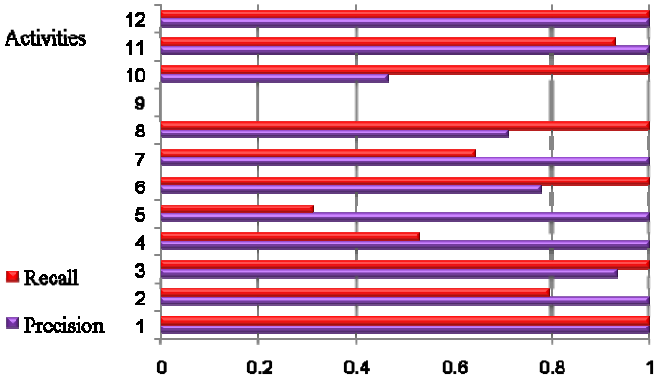


Fig. 6. Activity recognition results

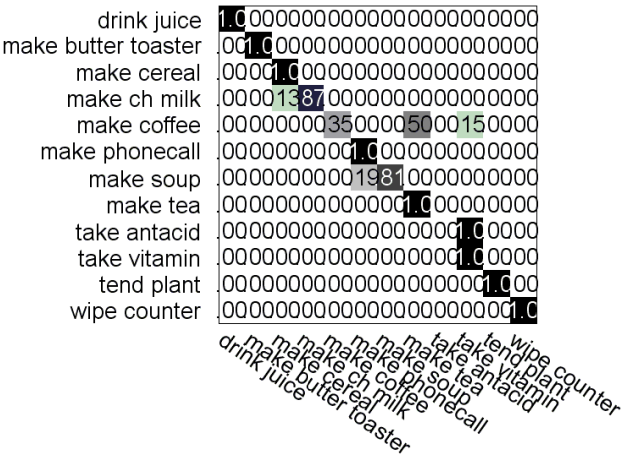


Fig. 7. Confusion matrix for activity recognition.

Fig. 6 shows activity recognition rates. Overall rates are good with an average of 82/74% recognition rates. In one case ("take antacid") we never classified a single super-segment correctly. This is because, due to heavy occlusion, antacid was very rarely correctly classified. When activity recognition was smoothed at a frame level (as is conventional) as opposed to super-segment level, P/R fall to 66/57%. Super-segmentation therefore provides clear benefits. Fig. 7 shows the confusion matrix for the activities. Other than the antacid sequence, the system also does poorly on “make coffee”, which involves several objects in common with “make tea”, such as milk and sugar. Also, due to the translucent nature of the coffee bottle, we are getting random and noisy patches from the sensor. The heavily occluded white coffee cup and white vitamin bottle also make the system classify some of the frames into “take vitamin”.

To examine how good our super-segmentation was, we measured for each actual object pick-up event, the minimum

frame distance to an inferred object pick-up event (d_t). We also measured the average distance between inferred pick-ups (d_b). In an ideal system, inferred pick-ups would coincide exactly with actual ones, and no other boundaries would be inserted, so that d_t would be small and d_b large. In our case, the average value of d_t is 9 (median 5) frames, and the average d_b distance is 47 frames when only pixel data are used for prediction, 79 with just depth, and 101 when depth and color are used (median 14, 31, 42). We determined our classifier has 90% precision when we classify 27% of the frames. At this setting, we expect 10 (= 0.27*0.9*42) frames to vote correctly and 1 to vote incorrectly in the median super-segment, a comfortable margin. Our super-segmentation technique is therefore effective, and the merging of depth and RGB cues provide significant benefits.

TABLE II PRECISION RECALL NUMBERS (PRECISION/RECALL) FOR DIFFERENT SMOOTHING WINDOW SIZE OF 30, 45, AND 60 FRAMES. INCREASING THE WINDOW SIZE IMPROVES THE RECALL BUT STILL NOT AS GOOD AS OUR SCHEME.

Threshold	30 frames	45 frames	60 frames
0.5	0.75/0.60	0.76/0.64	0.77/0.67
0.6	0.80/0.59	0.81/0.64	0.81/0.67
0.7	0.85/0.57	0.84/0.62	0.84/0.65
0.8	0.89/0.55	0.87/0.59	0.87/0.63

TABLE III TIMING BREAKDOWN FOR DIFFERENT COMPONENTS OF THE PROPOSED SYSTEM.

Stage	Time taken(sec/frame)
Motion-gated differencing + hand finding	0.05
Object and hand extraction	0.65
Compute object features	0.12
Object recognition	0.008
Pickup event detection	0.001
Activity recognition	0.007

Table II shows the precision recall numbers using different temporal smoothing window sizes (1-second/30 frames, 1.5-seconds/45 frames and 2-seconds/60 frames window) on the object recognizer outputs. Note that in general, increasing the window size helps bring the recall rate higher, but still not as good as the super-segmentation results.

Table III shows the detailed timing breakdown of our system based on a mixture of Matlab and C implementation, measured on an Intel Quad-core machine with 6 GB memory. The most expensive operation in our pipeline is computing the features, including texton, for object recognition but in general, our system can be further optimized in real-time.

V. CONCLUSIONS

We have presented a system for analyzing object manipulation based activity that provides good performance under challenging real-world conditions. The main novelty of our system is the extensive use of depth cues to augment illumination based cues. We have provided fast and effective techniques to parse the depth information, a comprehensive system design to infer activity and object-use from depth/vision data and an evaluation analyzing our approach on a realistic dataset. Our algorithms were simple but surprisingly effective. Our main conclusion is that use of depth information available from inexpensive modern depth cameras can improve the performance of video-based activity recognition, pose estimation and object recognition systems substantially.

REFERENCES

[1] Hedvig Kjellstrom, Javier Romero, David Martínez, and Danica Kragic, "Simultaneous Visual Recognition of Manipulation Actions and Manipulated Objects," in *IEEE European Conference on Computer Vision (ECCV)*, 2008.

[2] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake, "Real-Time Human Pose Recognition in Parts from Single Depth Images," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[3] Abhinav Gupta and Larry Davis, "Objects in Action: An Approach for Combining Action Understanding and Object Perception," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[4] Shiaokai Wang, William Pentney, Ana-Maria Popescu, Tanzeem Choudhury, and Matthai Philipose, "Common Sense Based Joint Training of Human Activity Recognizers," in *International Joint Conference on Artificial Intelligence*, pp. 2237-2242, 2007.

[5] Jianxin Wu, Adebola Osuntogun, Tanzeem Choudhury, Matthai Philipose, and James Rehg, "A Scalable Approach to Activity Recognition Based on Object Use," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1-8, 2007.

[6] Moore Darnell, Essa Irfan, and Hayes Monson, "Exploiting Human Actions and Object Context for Recognition Tasks," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 80-86, 1999.

[7] Abhinav Gupta, Trista Chen, Francine Chen, Don Kimber, and Larry Davis, "Context and Observation Driven Latent Variable Model for Human Pose Estimation," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[8] Antonio Micilotta, Eng-Jon Ong, and and Richard Bowden, "Real-Time Upper Body Detection and 3d Pose Estimation in Monoscopic Images," in *IEEE European Conference on Computer Vision (ECCV)*, pp. 139-150, 2006.

[9] Andriluka Mykhaylo, Roth Stefan, and Schiele Bernt, "Pictorial Structures Revisited: People Detection and Articulated Pose Estimation," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1014-1021, 2009.

[10] Leo Breiman, "Random Forests," in *Machine Learning,* vol. 45, pp. 5-32, 2001.

[11] Tomasz Malisiewicz and Alexei Efros, "Recognition by Association Via Learning Per-Exemplar Distances," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[12] Yoav Freund and Robert Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boostin," in *Journal of computer and system sciences,* vol. 55, pp. 119-139, 1997.

[13] David Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," in *International Journal of Computer Vision,* vol. 60, pp. 91-110, 2004.