The 11th World Congress on Intelligent Control and Automation, 2014

Robust Object Tracking based on RGB-D Camera[*]

Wenjing Qi[a], Yinfei Yang[b], Meng Yi[b], Yunfeng Li[c], Zygmunt Pizlo[c], Longin Jan Latecki[b]

a. Dept. of Computer Science and Technology, Shandong Jianzhu University, China
b. Dept. of Computer and Information Sciences, Temple University, USA
c. Dept. of Psychological Sciences, Purdue University, USA

qiwj@sdjzu.edu.cn, latecki@temple.edu

***Abstract* - A novel object tracking method based on RGB-D camera is proposed to handle fast appearance change, occlusion, background clutter which may arise for vision-based robot navigation. It makes use of appearance and depth information that are complementary to each other in visual perception to get robust tracking. First, RGB image and depth information are captured by the RGB-D camera. Then, an online updating appearance model is created with features extracted from RGB image. A motion model is created on plan-view map that is drawn from depth information and camera parameters. The estimation of object position and scale is performed on the motion model. Finally, appearance features are combined with position and scale information to track the target. The performance of our method is compared with a state-of-art video tracking method. It shows that our tracking method is more stable and accurate, and has overwhelming superiority when there is a great appearance change. A vision-based robot using our tracking method can navigate in cluttered environment successfully.**

Index Terms - Object Tracking, RGB-D camera, Plan-view, Detection, Robot Navigation.

I. Introduction

Vision ability of robot will broaden its applications in surveillance, search and rescue, outdoor and indoor building inspection, especially in high risk missions. Vision-based navigation can be roughly divided into map-based navigation that need previous knowledge of the whole environment and mapless navigation that perceive the environment as they navigate through it [1]. We intend our robot can take a mapless navigation like human, and handle the dynamically changed scene, a "look and move" strategy is employed by our robot, which means robot move a step then stop to detect and localize the target, renew the path and walk a step on the new path, repeat this process till it reaches the target.

One of the most challenging problems during this process is how to recognize and localize the target correctly in each separate step. Traditional tracking methods [2][3][4] mainly depend on appearance features of objects, they perform well in video tracking tasks, but are weak in handling abrupt appearance change. In our application, robot takes one picture per step, the appearances of objects in pictures vary a lot due to the views and scales changes between two steps. Furthermore, the movement of target objects and robot camera may also add the instability of appearance features. To tackle these problems, we combine the depth information and RGB together to get a robust tracking. An RGB-D camera will be employed as vision system of robot, which will provide robot with appearance features and depth information.

Fig. 1 shows how the robot responded to intentionally blocked way during walking towards target. The first row shows the scene that robot saw, the second row shows the planned path in each step. In third step, it found there was a change happened in the scene, a chair was put in the way of heading for target, it recalculated a new path to bypass the chair.

In this paper, we will first introduce our application briefly, and then focus on discussing how to make tracking more robustly by using RGB-D camera. The main contributions of our work include: (1) Plan-view maps generated from depth information are used to predict the scale and location of target. (2) The combination of Appearance and location information makes tracking more robust, meanwhile, the predicted scale and location makes detection more efficient. (3) Adaptable position probability map is used to help improve the accuracy in tracking occluded target, moving target and distinguishing the correct target from other objects with same appearance.

The rest of the paper is organized as follows: section 2 reviews recent object detection and tracking methods, section 3 outlines how our robot navigates among obstacles, section 4 describes our object tracking method in detail. Then experimental results and performance evaluation are presented in section 5, and a conclusion will close this paper.

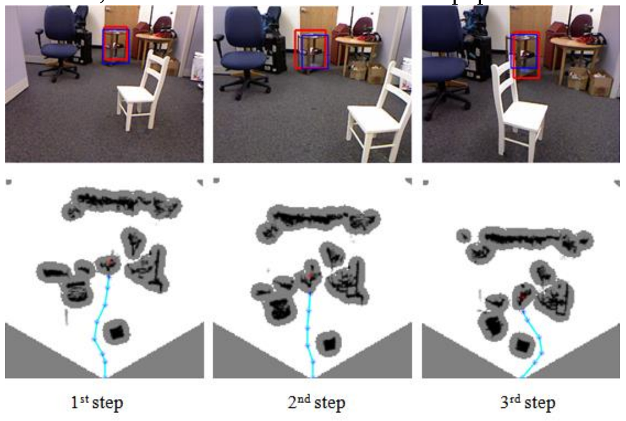


Fig. 1 Plan the path dynamically. Top row: RGB images show what robot had seen, object in bounding box is the target. Bottom row: planned path in plan-view map, path is recalculated according to the changes of scene.

[*] This work is partially supported by Grant OIA-1027897 from US National Science Foundation, Shandong NSF Grant # ZR2013FL024 and Doctor Foundation # XNBS1261 from Shandong Jianzhu University

II. Related Works

A. Traditional Object Tracking

Object tracking is widely used in many applications, such as surveillance, robot navigation, and human-computer interaction. The main challenges include changes of object appearance and view, occlusion between objects, abrupt motion of object and camera motion[6]. Some recent literatures make use of tracking-by-detection method, which usually perform one-shot learning at the first frame, then perform an online updating by using the tracking result of the successive frames to adapt to the changes of appearance of objects[7]. A critical problem in tracking-by-detection methods is the stability-plasticity dilemma, which means how can a system retain old memories but learn new ones. In [8], a method of one-shot semi-supervised learning problem using online boosting was proposed. Supervised updates are only performed at the 1st frame and all subsequent patches are exploited as unlabeled data with the help of a non-adaptive prior classifier. Although this method is less susceptible to drifting and more adaptive than an off-line learner, it is still not adaptive enough to fast appearance changes. Stenger et al.[9] investigated in different combinations of tracking methods. Given a particular tracking scenario, they tried to learn which methods are useful and how they can be combined to yield good results. Santner et al.[3] also employed a combination of several trackers: template matching based on normalized cross correlation, mean shift optical flow and online random forests to predict the target location, the main difference between[9] and[3] is that [3] did not require off-line pre-training of possible combinations. Babenko et al.[10] adopt Multiple Instance Learning [2] to alleviate the possible drift and degraded model led by the inaccuracies in the tracker. In [4], a tracking-by-detection framework combines nearest-neighbor classification of bags of features, efficient sub-window search and a novel feature selection and pruning method to achieve stability and plasticity in tracking targets of changing appearance. In most of these object detection, sliding widows scheme is employed to search for the potential matched target in images, it is a time consuming process, efficient sub-window search[5] improved the searching efficiency by employing a divide and conquer strategy.

B. Stereo or RGB-D camera based tracking or detection

Recently, stereo or RGB-D camera have been widely used in object detection and tracking. Many literatures show dramatically increases in the robustness. Ess et al.[11][12] describe a stereo based system for the creation of dynamic obstacle maps for automotive or mobile robotics platform. The authors showed that the use of sparse three dimensional structures significantly improved the performance of pedestrian detection and tracking. [13] describe a fast and stable human detection based on subtraction stereo with HOG features[14] which can measure distance information of foreground regions.[15][16] present a new general detector HOD(Histogram of Oriented Depths),both of them show that their proposal is faster and less false detection than that in [14]. [17] proposed a new method to quickly and accurately predict 3D positions of body joints from a single depth image. Bo et al.[18] use their hierarchical kernel descriptors over RGB-D images for real-life recognition. We don't use 3D features or 3D structure analysis directly as the above literatures did. Instead, our work makes use of plan-view maps constructed from depth information.

C. Tracking with Plan-view Map

Plan-view maps were used in human detection and trajectory estimation [19][20][21]. The differences between our method and theirs are as followings: *First*, their tracking method is only based on the plan-view maps, which will likely lead to error detection when two objects are too close or trajectories cross each other. While in our method, appearance features in RGB image helps to correct the error. We predict possible position of object in plan-view, and combine it with appearance to determine the actual target and position. Second, the detection of the foreground objects must be performed first in the plan-view map by background subtraction, so it requires cameras keep static and learn the background model. In contrast, our approach works with moving cameras and dynamic changing background. Third, many assumptions were made in their papers, e.g. known environment, fix camera [19], and constant velocity, no great pose change and no change in occlusion [21] in consecutive frames. All of these assumptions are not valid happen in our application, so appearance feature must be employed to help padding the gap of great changes happened in each step of robot.

III. Overview of Robot Navigation

Robot performs three main operations during his navigation, tracking the target, planning a path, then moving forward, which repeated until it reach the target. Our robot equipped with a Kinect sensor as his vision system, which provide depth information aligned with image pixels from a standard camera. From the depth map, plan-view maps (PVM) is extracted, which provide us with a spatial distribution of objects on ground. Appearances from RGB image and position information from plan-view map are both used as cues for target object detection. Then, A* algorithm[22] is used to find a reasonable path, and a motion command is issued to steer robot walking. The working process of robot can be summarized by pseudo-code as follows:

```
Main(){
    Initialize();
    while robot does't reach target{
        Track();
        PlanPath();
        Move(); }
}
sub-routine Initialize() {
    Startup robot, robot takes the first look at the scene;
    Designate a target object for robot in the 1st RGB image;}
sub-routine Track() {
        Reconstruct plan-view map aligned with RGB image;
        Online updated appearance model;
        Construct motion model based on plan-view map;
        Detect target with appearance and motion model;  }
sub-routine PlanPath(){
        Find the target object in plan-view map;
        Plan a path to the target.  }
sub-routine Move() {
        Robot turns to target and moves a step on planned path;
        Turns to the target again, takes another RGB-D image;
        Get the motion parameters of robot;  }
```

Obviously, tracking the object correctly is the critical step for our "look and move" robot, if it fails, all subsequent operations can't proceed. The key points in our tracking method include: (a) Finding the objects correspondence between RGB image and plan-view image. (b) Predicting of position and scale of object in plan-view image. (c) Integrating position and appearance together to detect the object. We will describe these key points in detail in next section.

IV TRACKING THE TARGET OBJECT

A. Plan-view Map Reconstruction

Data from RGB-D camera include both an RGB image and depth information, which allows us to recover a 3D point cloud of objects in the view of robot. Then 3 steps are performed to get a PVM [23]: (1) finding the ground plane (e.g., floor), (2) removing the ground plane points from the scene, (3) projecting the remaining 3D points to ground plane. An RGB image and its corresponding PVM are shown in Fig. 2. As we can see, some furniture are occluding each other in RGB image, but are clearly separated in PVM. Since PVM provide us with additional information about the layout and position of objects, object tracking can take advantages of these facts to get more accurate results. The color of of object footprint in Fig.2(right) represents the maximum height of 3D points that project to this location.

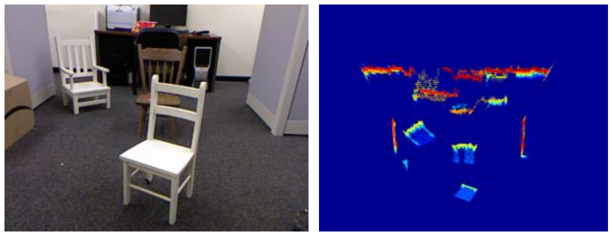


Fig. 2: An RGB image and its corresponding PVM.

Each detected object footprint in PVM can be back transformed to the RGB image. This fact is illustrated in Fig. 3, where each footprint marked with a red cross in PVM represents a detected object. The convex hulls of detected objects are overlaid on the RGB image. We illustrate the correspondence relation with arrows from PVM to RGB image. With the correspondence between RGB image and PVM, we can just evaluate windows with predicted scale around the detected convex hulls to find the target, which definitely improves the efficiency of object detection.

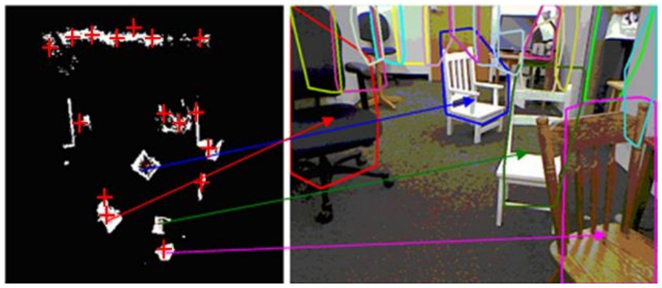


Fig. 3: Object correspondence between PVM and RGB image

B. On-line Updated Appearance Model on RGB Images

Our target tracking method is a tracking by detection. We use the idea of one-shot learning [24] and online updating scheme [4] to generate an accumulated appearance model. The object features in first frame form the initial appearance model. Then in the subsequent frames, matched features from detected object are used to update the appearance model. In this way, we keep both the initial features and the new features due to the changes in object view, scale and other properties.

The object appearance is described by dense-sift[25] features and bag-of-features. Dense-SIFT calculate features on evenly sampled points with predefined grid size while key-point SIFT only make use of key-points which produced by using scale-space extrema in the difference-of-Gaussian function. Compared to key-point SIFT, dense-SIFT can unbiasedly capture features on object, while key point SIFT probably lose some features due to key-points detection failure which may arise when the resolution or illumination change. Although dense-SIFT feature is not scale-invariant, we can overcome this shortcoming by that we estimate the scale of the object in PVM.

In the first image taken, the target is manually labeled with a rectangle around it. Let Λ_{rect} denote the set of sampled points of dese-SIFT in the rectangle. All features in the rectangle are considered as object features and form the initial object appearance model O_1. All features outside the rectangle form background model B_1. Let $f(x,y)$ denote feature at point (x,y), then:

$$O_1=\{f(x,y)|(x,y)\in\Lambda_{rect}\} \tag{1}$$

$$B_1=\{f(x,y)|(x,y)\in\overline{\Lambda_{rect}}\} \tag{2}$$

In subsequent frames, a sub-window with highest score is taken as the detected object. The sampled points matched with the object model, $\Lambda_{matched}$, will be add to the object model, and at the same time, the sampled points outside the bounding box Λ_{bdbox} is used to update the background model.

$$O_k=O_{k-1}\bigcup\{f(x,y)|(x,y)\in\Lambda_{matched}\} \tag{3}$$

$$B_k=\{f(x,y)|(x,y)\in\overline{\Lambda_{bdbox}}\} \tag{4}$$

We always keep the first object model and the matched features in M nearest frames, i.e. the object features in first frame and matched features in k-M+1,…,k frame forms the *kth* object model O_k. In our experiments, M = 5

C. Motion model based on Plan-view Map

We use the current position of target and camera motion parameters to estimate the possible position and scale of target in next PVM. Assume that (xp_i, yp_i) is the coordinates of target position in the *i*th PVM, the position of camera is origin O, axis-x is from left to right, axis-y is from bottom to top, assume robot rotates by an angle α and headed to target by distance d, then the position of target in *i+1*th frame is:

$$\begin{pmatrix} xp_{i+1} \\ yp_{i+1} \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & -d \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} xp_i \\ yp_i \\ 1 \end{pmatrix} \qquad (5)$$

The *predicted position* as shown in Fig. 4(a) is marked with blue star, a red cross represent the existence of an object, the green point means the real position of target. There sometimes exists deviation between predicted and real target position due to the small inaccuracy of robot motion, which probably lead to confusion when two objects are very close. In addition, if we want to track a target object among many objects with same looks as the target, position prediction will be more critical to avoid appearance confusion. On the other hand, we hope robot can track a moving object, which means an active deviation will generate from predicted position. To handle all these conditions, we calculate an *adaptable position probability map* on PVM. Assume the probability of a position to be the target subject to Gaussian distribution centers on predicted position:

$$p(xp, yp; xp_e, yp_e, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(xp^2+yp^2)-(xp_e^2+yp_e^2)}{2\sigma^2}} \qquad (6)$$

Where (xp_e, yp_e) is the coordinates of predicted target, σ is an adjustable parameter that can be set to adapt to certain conditions. E.g. if we want to move the target in a relatively large range, σ is set to a larger value, while if the target keeps still or we need track one among some same looking objects, σ should be small. We can also use anisotropic Gaussian distribution to make robot adaptable to target movement in different direction. Fig. 4(b) shows the position probability map.

We can also estimate the scale of the object after we get the predicted position. Let H_k and H_{k+1} denote the object scale in k and $k+1$ frame, D_k and D_{k+1} is the distance between camera and target in k and $k+1$ frame, we simplify the camera imaging as a pinhole imaging, then we get:

$$H_{k+1} = H_k \times \frac{D_k}{D_{k+1}} \qquad (7)$$

H_1 is the scale of the target in first frame. Then we just need to evaluate sub-window of estimated scale in each step to find the target in image, this will definitely decrease computation cost.

D. Target object detection

In i+1th frame, each sampling point is evaluated to determine if it is an object feature or a background feature.

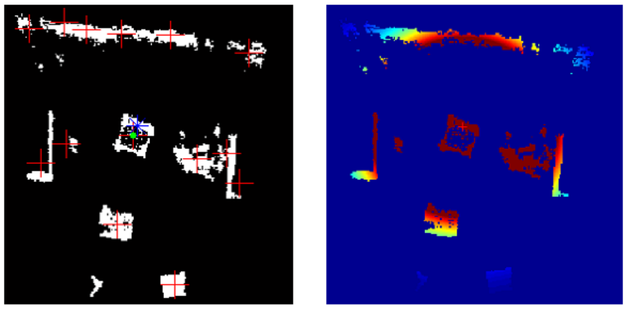

(a) (b)

Figure 4: (a) detected position (green point) and predicted position(bluepoint) of target,(b) Position probability map

First, we find its N nearest neighbors in both models. Let L_o and L_b represent the average L2 distance to its N nearest neighbors in object model O_{i+1} and background model B_{i+1}. A score was given to each point:

$$s(x_i, y_i) = \begin{cases} a & \text{if } L_o/L_b \le threshold \\ -a/c & otherwise \end{cases}, \qquad (8)$$

where threshold is a statistical value that measures the difference between features of background and object, $L_o/L_b \le threshold$ means sampling points match with object model, and assigned positive score a . For unmatched sampling points are assigned a penalty value $-a/c, a>0, c>0$. For all possible sub-windows $w_j \in W$ that an object may exist, the score of the window is defined as:

$$S_{w_j} = \sum_i \{ s(x_i, y_i) | (x_i, y_i) \in w_j \} \qquad (9)$$

We need guarantee that S_{w_j} is always non-negative. Suppose that maximum number of matched sampling points in sub-window is N_o, and N_b is background feature, $c > N_b/N_o$, if we assume that at least 10 matched object points means a target, then $c > N_b/10$. The value of *a*, *c* and *threshold* used in our application is 2, 100 and 1.5.

For each possible sub-window, we got an appearance score S_{w_j}, then a position factor p_{w_j}, which got from (6) with center coordinates of sub-window w_j, is added to the score, the target window is defined as follows:

$$w_t = w_{\arg\max_j (S_{w_j} \times p_{w_j})} \qquad (10)$$

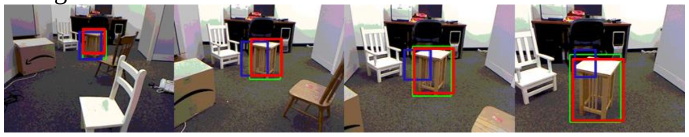
V. Experimental Results

By integrating appearance feature extracted in RGB image with position and scale information got from PVM, robot can track the target robustly. In our experiments, robot walked 27 rounds on tracking 9 kinds of different objects, including 4 different chairs, 1 garbage can, 2 tables and 2 boxes, there are totally 173 steps and detections. Here we show some experimental results in different conditions and we compared its performance with *nntracker*[4] in same features. In Fig. 5 to

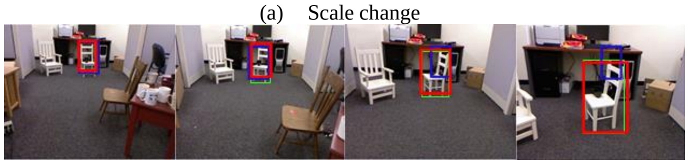
Fig. 12, bounding boxes or curves in red and blue are corresponding to results of our method and *nntracker* respectively. Green bounding box in Fig.5 is ground truth, since it is easy to tell the right target, we did not show ground truth in other figures.

A. Robustness to Scale and View Change

In Fig. 5(a), the scale of target was changing greatly among frames as robot moving towards it. In Fig5(b), we intentionally changed the view of the chair in each step. The results show that our method is robust to great appearance changes.



(a) Scale change



(b) view change

Fig.5 Robustness to appearance change

B. Robustness to Occlusion

In Fig. 6, there were occlusions in second and third frames. When occlusion occurs, some “noise features” will be added to appearance model during model updating process, this will lead to the tracking results of method merely based on appearance drift greatly from target. While in our method, the position factor complements the negative effect of noise in appearance by (10), and the tracking results are stable.

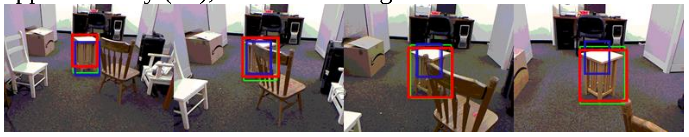


Figure 6. Robustness to Occlusion

C. Self-Recovery from Error

Another advantage of our method is that it has self-recovery ability when a wrong detection occurs in a frame. The error will not affect the successive frame because we do not only rely on appearance, the location information will help to recover from fault. As shown in Fig. 7, we observe that a large detection error occurred in 1st and 2nd images, but this error did not accumulate in next frames, it eventually recovered from the error and got a good result.

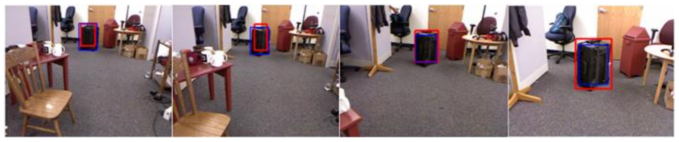


Fig. 7 Robustness in recovery from error

D. Distinguish similar objects

In Fig.8, there are two chairs with similar appearance, the left one was designated as target. There is only a little drift from target with our tracking method, while *nntracker* get a totally wrong detection.

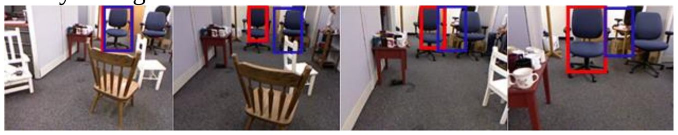


Fig. 8 Robustness in distinguish similar objects

E. Tracking Performance Evaluation

We evaluate tracking results with 3 methods. First we test our method with a common evaluation criterion as being used in [4], which computes the mean distance error(MDE) e:

$$e=\frac{1}{n}\sum_{i=1}^{n}\|O_i-O_i^g\| \qquad (11)$$

where n is the number of frames in each round, and $\|O_i-O_i^g\|$ is the Euclidean distance between the tracked window centroid O_i and the ground truth window centroid O_i^g. The results of $\|O_i-O_i^g\|$ for 173 frames are shown in Fig. 9, it shows that the distance errors of our method are lower than that of *nntraker*. The average of e in 173 frames is shown in Table I.

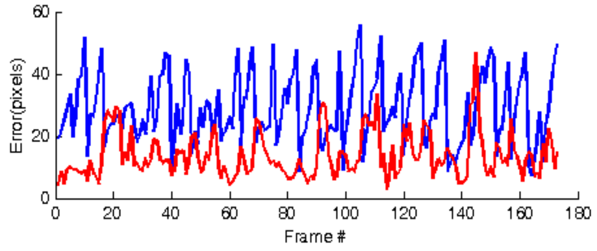


Fig. 9 Mean distance error

Although (11) is mostly used in video tracking scenarios, it can only tell the accuracy of the position (centroid) of tracked window. But in our application it is not enough to represent tracking accuracy, because there are great scale changes of target, we need also to guarantee the right scale of the tracking target. Since our method is a tracking-by-detection method, we can naturally evaluate the accuracy of tracking by commonly used method[3] in object detection:

$$Accuracy=\frac{A(R_{\det}\cap R_{gt})}{A(R_{\det}\cup R_{gt})}\quad , \qquad (12)$$

where A($\cdot$) denotes the area of a region, R_{gt} is the region of ground truth and $R_{\det}$ is the region of detected object, a higher value means a more accurate detection. Fig. 10 shows that our tracking accuracy is higher and more stable. In Fig.11, we illustrate the accuracy for each round, It shows that accuracy of our method is relatively stable from the first step to the last step in each round, while accuracy of *nntracker* drops a lot as robot approaching the target. It means robot using our tracking method will eventually reach the target, despite the great view change and occlusion occurred during this process.

If $Accuracy \geq 50\%$ means a right detection, there are only 8 out of all 173 frames in which the target was missed. The detection rate, average accuracy and tracking error are shown in TABLE I

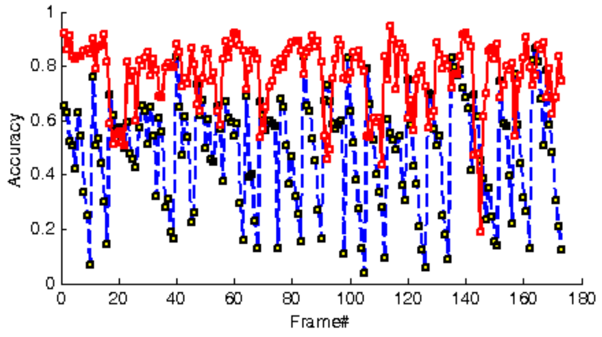


Fig. 10 Comparison of tracking accuracy

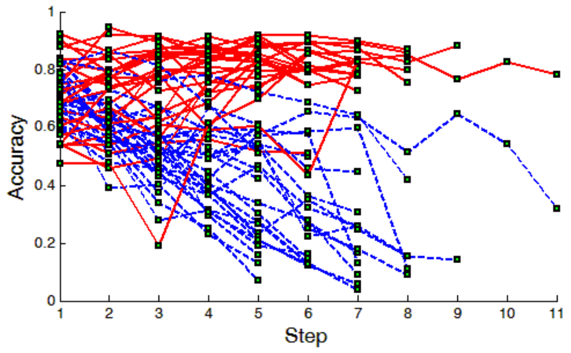


Figure 11: Accuracy for each navigation round

TABLE I
EVALUATION OF PERFORMANCE

Method	feature	Performance Evaluation		
		Detection rate	Average accuracy	MDE *e* (pixels)
Ours	dense-SIFT	**95.38%**	**76.66%**	**13.54**
nntracker		57.14%	48.48%	28.67

VI. CONCLUSIONS

In video tracking, the difference between two frames are not so salient, the appearance changes gradually or relatively stable. It is not the case when robot navigate with a mapless and "look and move" pattern, the images taken are not continuous, together with robot walking or target moving, the changes of view, scale, and occlusion are unpredictable, which makes target tracking difficult. RGB-D cameras that provide depth information aligned with RGB image present a new way for object detection and tracking. Plan-view map constructed from depth information provides the position and scale information of objects. By integrating this information with appearance features from RGB image, we get robust tracking results. Moreover, another two kinds of useful information can be obtained from depth information, one is convex hulls around objects in RGB image, and the second is estimated scale of object. These two kinds of information help to improve detection speed, which is critical for real-time robot navigation. Presented experiments show our tracking method is very robust and efficient in tracking object with appearance changes and occlusions; also, it works well at tracking a moving object. Since our method is closely related to depth information and there may exist small errors occasionally in convex hull inference, we will refine this part in the future work in order to further improve the performance of our method.

REFERENCES

[1] F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: A survey. Journal of Intelligent and Robotic Systems, 53(3):263–296, 2008.
[2] P. Viola, J. C. Platt, and C. Zhang. Multiple instance boosting for object detection. In: NIPS, 18:1419–1426, 2006.
[3] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost parallel robust online simple tracking. In: CVPR'10, pages 723–730,2010.
[4] S. Gu, Y. Zheng, and C. Tomasi. Efficient visual object tracking with online nearest neighbor classifier. In: ACCV2010, 2010.
[5] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. IEEE Trans. Pattern Analysis and Machine Intelligence, 31:2129–2142, 2009.
[6] Y. Alper, J. Omar, and S. Mubarak. Object tracking: A survey. ACM Computing Surveys, 38(4), 2006.
[7] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In Proceedings British Machine Vision Conference, 1:47–56,2006.
[8] H. Grabner, C. Leistner, and H. Bischoff. Semi-supervised on-line boosting for robust tracking. In ECCV'08, pages –, 2008.
[9] T. Stenger, B. Woodley, and R. Cipolloa. Learning to track with with multiple observers. In: CVPR'09, 2009.
[10] B.Babenko, M.Yang, and S.Belongie. Visual tracking with online multiple instance learning. In : CVPR'09, pages 983–990, 2009.
[11] A. Ess, B. Leibe, and L. V. Gool. Depth and appearance for mobile scene analysis. In: ICCV'07, 2007.
[12] A. Ess, B. Leibe, K. Schindler, and L. van Gool. A mobile vision system for robust multi-person trackin. In CVPR'08, 2008.
[13] M. Arie, A. Moro, Y. Hoshikawa, T. Ubukata, K. Terabayashi, and K. Umeda. Fast and stable human detection using multiple classifiers based on subtraction stereo with hog features.
[14] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In CVPR, pages 886–893, 2005.
[15] L. Spinello, M. Luber, and K. O. Arras. Tracking people in 3d using a bottom-up top-down detector. In: ICRA'11), pages 1304–1310, 2011.
[16] M. Luber, L. Spinello, and K. O. Arras. Learning to detect and track people in rgbd data. In:Workshop on RGB-D Cameras RSS), 2011.
[17] J. Shotton, A. Fitzgibbon, and M. C. et al. Real-time human pose recognition in parts from single depth images. In: CVPR2011, 2011.
[18] L. Bo, X. Ren, and D. Fox Depth Kernel Descriptors for Object Recognition. In: IROS2011, 2011
[19] T. Darrell, D. Demirdjian, N. Checka, P. Felzenszwalb. Plan-view trajectory estimation with dense stereo background models. In: ICCV'01,2001.
[20] Michael Harville. Stereo person tracking with adaptive plan-view templates of height and occupancy statistics. Image and Vision Computing,22(2)127-142,2004
[21] F. Bonin-Font, A. Ortiz, and G. Oliver. People detection and tracking using stereo vision and color. Image and Vision Computing, 25(6): 995-1007, 2007.
[22] Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. on Systems Science and Cybernetics 4 (1968) 100 – 107
[23] D. Burschka and G. Hager. Stereo-Based Obstacle Avoidance in Indoor Environments with Active Sensor Re-Calibration. In: ICRA 2002, 2066-2072,2002
[24] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. IEEE Trans. Pattern Analysis and Machine Intelligence,24(4):594 – 611, 2006.
[25] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. SIFT flow: dense correspondence across different scenes. In: European Conference on Computer Vision (ECCV), 2008