Texture-Less Object Tracking with Online Training using An RGB-D Camera

Youngmin Park* Vincent Lepetit† Woontack Woo‡
GIST U-VR Lab EPFL - CVLab GIST U-VR Lab

ABSTRACT

We propose a texture-less object detection and 3D tracking method which automatically extracts on the fly the information it needs from color images and the corresponding depth maps. While texture-less 3D tracking is not new, it requires a prior CAD model, and real-time methods for detection still have to be developed for robust tracking. To detect the target, we propose to rely on a fast template-based method, which provides an initial estimate of its 3D pose, and we refine this estimate using the depth and image contours information. We automatically extract a 3D model for the target from the depth information. To this end, we developed methods to enhance the depth map and to stabilize the 3D pose estimation. We demonstrate our method on challenging sequences exhibiting partial occlusions and fast motions.

Index Terms: I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Motion; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Range Data

1 INTRODUCTION

While many vision-based 3D object detection and tracking algorithms have been proposed, most of them exploit keypoint-based approaches which require enough texture on the target object. By contrast, while tracking solutions based on edges for texture-less objects have been proposed [5, 3], the complexity of edge-based detection methods [10, 7] are typically too high for interactive applications. Moreover the pose estimation requires the 3D CAD model and acquiring such model is clearly cumbersome.

Our work is motivated by DOT, a fast 2D template detection method [6] and the recent release of the Kinect which provides a depth map together with a color image in real-time: If the pose corresponding to the template is known, an initial estimate of the object pose can be computed from the 2D template detection. The Kinect is also attractive as it provides depth information even for texture-less surfaces, by contrast with approaches based on passive stereo cameras.

The method we propose therefore detects the target object in the input color image using DOT templates and computes a first estimate of its 3D pose. This estimate is then refined using both the depth map and the color image. For a given target object, we automatically collect a set of appropriate templates labelled with the object 3D pose and other useful information for tracking during a training phase, in a way similar to what was done in [6] to capture new templates: The user first selects the target object in the first frame of a training sequence, then the camera can be moved around the object, and the system captures new templates from the new viewpoints. By simultaneously tracking a known pattern, the system can label these templates with the camera 3D pose, used to predict the object pose when a template is detected again.

In addition, we process the depth map to reduce the noise. The raw depth map from the Kinect can be affected by noise which may cause unstable pose estimation. Our method reduces noise at both plain surface areas and object boundaries, and we show this improves the stability of the estimated pose.

In the next section, we discuss related works and background information. Section 3 presents the proposed method. The experimental results are described in Section 4. We conclude with Section 5.

2 RELATED WORK

3D detection of texture-less objects was a popular Computer Vision topic more than a decade ago [10, 7], however the complexity of the developed algorithms tend to be high, and the focus turned to textured objects for which considerable progress has been accomplished [11]. More recently, [4] proposed an edge-based detection technique but its computation time is still prohibitive for interactive applications. [9] exploited an inclination sensor mounted on the camera together with geometric constraints. [6] proposed a fast template detection method called DOT that works on texture-less objects, and we use it to detect our target object [1]. DOT provides a 2D detection and we show how to use it to obtain an estimate of the 3D pose.

Efficient 3D tracking methods for texture-less objects based on image contours already exist [5, 3], and compute the object pose by minimizing the 2D displacement between the projected edge of a known 3D model and the contours in the image. Edge information can also be combined with keypoints [17, 15]. All these works assumed that the 3D model of the object is somehow pre-built and available, however building such model is cumbersome if the object is complex or texture-less. An automated runtime 3D CAD model generation method was proposed in [13], however it requires enough texture on the target surface. The proposed method does not require any prior CAD model, and extracts the needed 3D information from the Kinect output.

Depth map enhancement with the help of the video image was done by [2] with a Markov Random Field applied to the high-resolution range image generation. [8] makes use of the Bilateral Filter [16] approach for preserving the discontinuities of the depth map while smoothing it. We propose an enhancement method that explicitly distinguishes plain surfaces from object boundaries in the depth map.

3 PROPOSED METHOD

This section describes the training and tracking methods we propose. Assuming the RGB image and depth map are synchronized, we geometrically calibrate both images and enhance the quality of the depth map. This is followed by object detection and frame-to-frame tracking. Note that no object model or keyframes are required in advance, that is, all the detection and tracking data are

*e-mail:ypark@gist.ac.kr
†e-mail:vincent.lepetit@epfl.ch
‡e-mail:wwoo@gist.ac.kr

IEEE International Symposium on Mixed and Augmented Reality 2011
Science and Technolgy Proceedings
26 -29 October, Basel, Switzerland
978-1-4577-2185-4/10/$26.00 ©2011 IEEE

[1] We adapted the DOT open source code available at `http://campar.in.tum.de/personal/hinterst/index`.

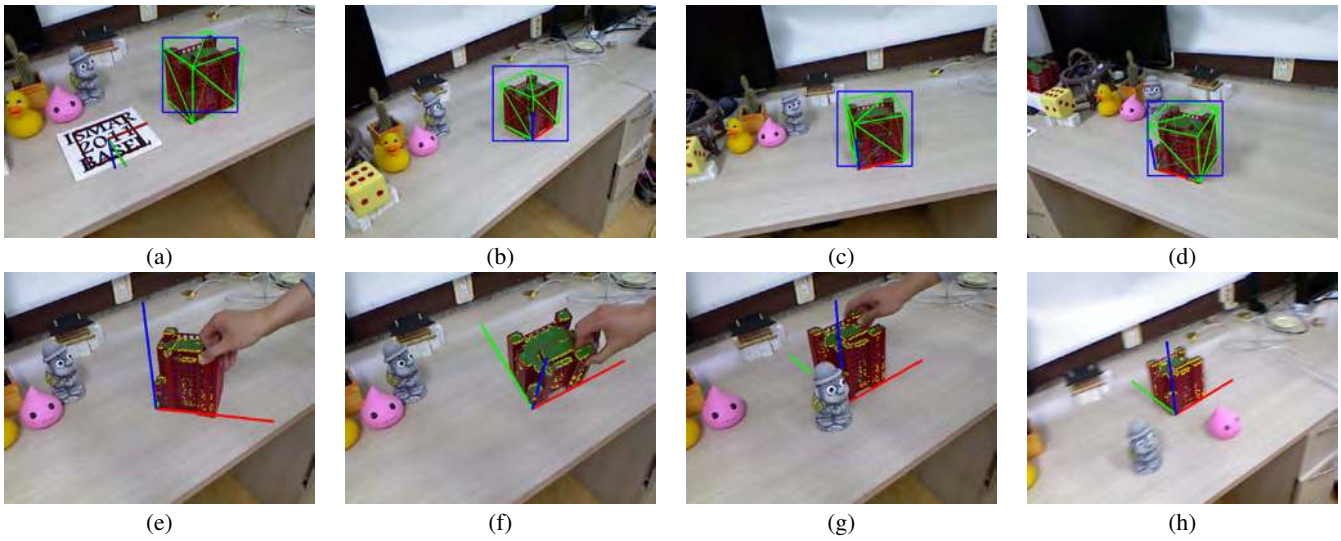

Figure 1: Overview of our method. (a) The user selects the object in the first frame of a training sequence by defining its position with respect to a known pattern and a bounding box. The pattern defines the object coordinate system. (b-d) The camera can then be moved, and the system captures DOT templates of the object and the corresponding depth maps from the new viewpoints. (e,f) The object can then be tracked under different viewpoints even under partial occlusion (g) and fast motion (h). We do not need any prior CAD model and the object does not have to be textured. *The complete sequences is available in the supplementary video.*

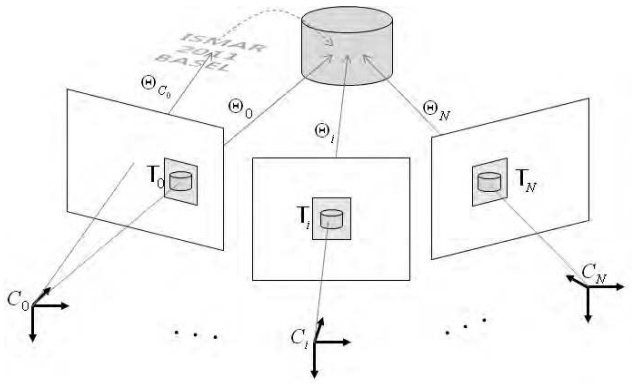

Figure 2: The configuration for training. At the beginning, the object coordinate frame is set with the help of a known pattern. We then capture new templates of the object appearance together with the object pose and the depth map while the user moves the camera around the object.

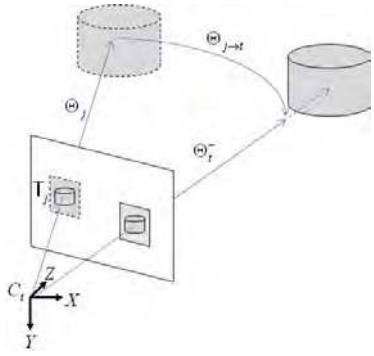

Figure 3: Because the object is generally not at the same image location as the one when the template was created, we compensate for the translations along the X- and Y- axes based on the difference of 2D locations and the camera internal parameters.

acquired on the fly. The following sections describe each step in more details.

3.1 Training an Object Model

To detect and track an object, we obtain the templates and other useful data on the fly without any prior object model. As shown in Figure 2, the training requires only minimal interaction, which consists in moving the camera around the object.

The training configuration is depicted in Figure 1. At the beginning, we align the object with a known pattern as Figure 1(a) which is detected by Ferns [12], and the pose is refined by ESM-Blur [14]. The world coordinate frame of the object is manually defined with respect to the pattern. Once the first template and its pose are obtained, the pattern is no more required as shown in Figure 2(b).

The first template is obtained at the center of the projected bounding box. It is then used to track the object while the user moves the camera around the object. The system automatically decides when to create a new template based on the following criteria:

1. Minimum rotational step: We add a new template if the estimated pose (rotation) is too different to any of the rotations of the previously acquired templates.

2. Detection score: We add a new template if the score returned by DOT is below than a fixed threshold.

If one of these criteria is satisfied, a new template from the current viewpoint is added. Together with the DOT template, we store the object pose, the depth map for the template region, and contour points extracted from the image after non-maximal suppression. The additional data is used for pose estimation and tracking as described below.

3.2 Detection and Initial Pose Computation

DOT detects 2D templates in an image even in presence of small viewpoint differences, and we use it to detect the target object and predict its 3D pose. Each template is labeled with the corresponding 3D pose, as determined during training. However, as shown in Figure 3, even if the object has an appearance similar to a template, the object 3D pose can still be different from the pose label. The difference in rotation is typically negligible, but the difference in translation can be large.

More exactly, the translations along the camera z-axis of the pose label and the detected object are similar as the object scale is similar (DOT is not invariant to scale), but the translations along the two other axes can be very different. Fortunately, they can be approximated from the object 2D location in the image. We estimate the differences in translation as:

$$\delta t_x = (u - c_x)\frac{t_z}{f_x},\ \delta t_y = (v - c_y)\frac{t_z}{f_y} \quad ,$$

where u and v are the 2D coordinates of the object center as detected by DOT, and c_x, c_y, f_x, and f_y the camera internal parameters. We add these values to the template pose label to obtain a first estimate of the object pose. This estimate is refined using the method described in the next section.

3.3 Pose Estimation and Frame-to-Frame Tracking

We refine the pose estimated from the template detection by aligning the depth and image contours data. We also use this method to track the object frame-by-frame once it has been detected in one frame, as this is typically faster and more stable than a tracking-by-detection. In case of failure, we re-run the template detection.

We first consider the depth information only, and use the Iterative Closest Point algorithm (ICP) [1] to align the depth map of the object as stored with the detected template and the depth map captured from the current image. However, even with the depth enhancement procedure described in Section 3.4, the pose estimated with ICP only still jitters because of the depth noise and we perform further refinement by considering the image contours.

ICP efficiently estimates the transformation that aligns two 3D point clouds. This is formulated as:

$$\widehat{\Delta\Theta_{\mathcal{D}}} = \underset{\Delta\Theta}{\operatorname{argmin}} \sum_i \|\mathbf{M}'_i - \varphi(\mathbf{M}_i, \Delta\Theta)\|^2 , \tag{1}$$

where $\mathbf{M}_i$ is a 3D point of the template, $\mathbf{M}'_i$ is the corresponding point in the input depth map, and $\varphi(\mathbf{M}_i, \Delta\Theta)$ is a function that transforms point $\mathbf{M}_i$ by motion $\Delta\Theta$.

This pose is further refined by fitting the 3D contours stored with the template to the current image, as done in [3]. This is formulated as:

$$\widehat{\Delta\Theta_{\mathcal{I}}} = \underset{\Delta\Theta}{\operatorname{argmin}} \sum_{i,j} \operatorname{dist}\left(\mathbf{e}_{i,j}, \operatorname{Proj}(\mathbf{E}_i, \Theta_t^- + \widehat{\Delta\Theta_{\mathcal{D}}} + \Delta\Theta)\right)^2 , \tag{2}$$

where $\mathbf{E}_i$ is a 3D contour, $\mathbf{e}_{ij}$ is a corresponding contour point extracted from the current image, and $\operatorname{Proj}(\mathbf{E}_i, \Theta)$ is the 2D projection of $\mathbf{E}_i$ under pose Θ, and dist denotes the distance between a 2D point and a 2D contour. Θ_t^- denotes the pose estimated from the template detection as explained in the previous section, or the pose estimated for the previous frame if the object was already detected.

The $\mathbf{E}_i$ edges are obtained during training by extracting contours from the image, and back-projecting them to the depth map to get their 3D locations. As in [3], the 2D points e_{ij} are found by first sampling the 3D contours into 3D points, projecting them giving the current pose estimate, and looking for large image gradients in a direction normal to the projected contour. We optimize Eq. (2) using Iterative Re-weighted Least Squares [5, 3]. The final pose is updated by the refinements from the depth and contour data as:

$$\Theta_t = \Theta_t^- + \widehat{\Delta\Theta_{\mathcal{D}}} + \widehat{\Delta\Theta_{\mathcal{I}}} . \tag{3}$$

We detect tracking failure based on the robust average distance between matched 3D points after ICP. If this distance is too large, we re-initialize the tracking by detecting the object as described in the previous section.

3.4 Depth Enhancement

The depth values returned by the Kinect suffer from noise, and are sometimes even absent for some points depending on the reflectance properties of the scene. We therefore process the depth map before using it. Simple Gaussian smoothing is not suitable as it would not preserve the depth discontinuities. Instead the bilateral filter [16] is more adapted.

The standard bilateral filter $F[\mathcal{I}]$ can be seen as a Gaussian filter weighted by a Gaussian weight depending on the intensity variation as:

$$F[\mathcal{I}]_{\mathbf{x}} = \frac{1}{W_{\mathbf{x}}} \sum_{\mathbf{y}\in S} G_{\sigma_s}(\|\mathbf{x}-\mathbf{y}\|) G_{\sigma_r}(\|\mathcal{I}_{\mathbf{x}} - \mathcal{I}_{\mathbf{y}}\|)\mathcal{I}_{\mathbf{y}} , \tag{4}$$

where $\mathcal{I}$ is the processed image, $G_\sigma(\cdot)$ is the Gaussian kernel, σ_s and σ_r denote the standard deviation for spatial distance and intensity for filtering, S the image window, and $\mathbf{x}$ and $\mathbf{y}$ pixel coordinates. $W_{\mathbf{x}}$ is a normalizing factor. Throughout this section, the subscript $_{\mathbf{x}}$ denotes the value at pixel coordinates $\mathbf{x}$.

The main limitation when applying directly the bilateral filter or joint bilateral upsampling to a depth map captured by the Kinect is that the noisy boundary is also preserved and the holes in the depth map are not properly handled. Our approach is therefore a variant of the bilateral filter applied to the Kinect depth map that explicitly considers missing data. Filtering is done in different ways depending on whether the depth pixel is close to a depth discontinuity or not. Far from a discontinuity, our filter strictly behaves like the bilateral filter; when close to a contour, it starts using predicted depth values where it is missing.

More exactly, our depth enhancement filter is defined as:

$$DEF[\mathcal{D},\mathcal{I}]_{\mathbf{x}} = \begin{cases} SF[\mathcal{D}]_{\mathbf{x}} & \text{if } \mathbf{x} \text{ is not close to a discontinuity} \\ EF[\mathcal{D},\mathcal{I}]_{\mathbf{x}} & \text{otherwise} \end{cases} , \tag{5}$$

where $SF[\mathcal{D}]_{\mathbf{x}}$ is the standard bilateral filter applied to the depth map $\mathcal{D}$, and $EF[\mathcal{D}]_{\mathbf{x}}$ is similar to the bilateral filter but relies on the predicted depth $d_{\mathbf{x}}(\mathcal{D},\mathcal{I})$ for $\mathbf{x}$. To find the pixels close to depth discontinuities, we first apply the Sobel operator on the depth map, and threshold the results. $d_{\mathbf{x}}(\mathcal{D},\mathcal{I})$ is estimated as a weighted average of the depths of the pixels around $\mathbf{x}$ with a similar color:

$$d_{\mathbf{x}}(\mathcal{D},\mathcal{I}) = \frac{1}{W_{\mathbf{x}}^{(d)}} \sum_{\mathbf{y}\in S, \mathcal{D}_{\mathbf{y}}\neq\perp} G_{\sigma_s}(\|\mathbf{x}-\mathbf{y}\|) G_{\sigma_r}(\|\mathcal{I}_{\mathbf{x}} - \mathcal{I}_{\mathbf{y}}\|)\mathcal{D}_{\mathbf{y}} , \tag{6}$$

with the convention $\mathcal{D}_{\mathbf{y}} = \perp$ if depth data is missing at location $\mathbf{y}$. The normalisation factor is computed as:

$$W_{\mathbf{x}}^{(d)} = \sum_{\mathbf{y}\in S, \mathcal{D}_{\mathbf{y}}\neq\perp} G_{\sigma_s}(\|\mathbf{x}-\mathbf{y}\|) G_{\sigma_r}(\|\mathcal{I}_{\mathbf{x}} - \mathcal{I}_{\mathbf{y}}\|) . \tag{7}$$

Then $EF[\mathcal{D},\mathcal{I}]_{\mathbf{x}}$ is defined as:

$$EF[\mathcal{D},\mathcal{I}]_{\mathbf{x}} = \frac{1}{W_{\mathbf{x}}^{(EF)}} \sum_{\mathbf{y}\in S} G_{\sigma_s}(\|\mathbf{x}-\mathbf{y}\|) G_{\sigma_d}(d_{\mathbf{x}}(\mathcal{D},\mathcal{I}) - \mathcal{D}_{\mathbf{y}}\|)\mathcal{D}_{\mathbf{y}} . \tag{8}$$

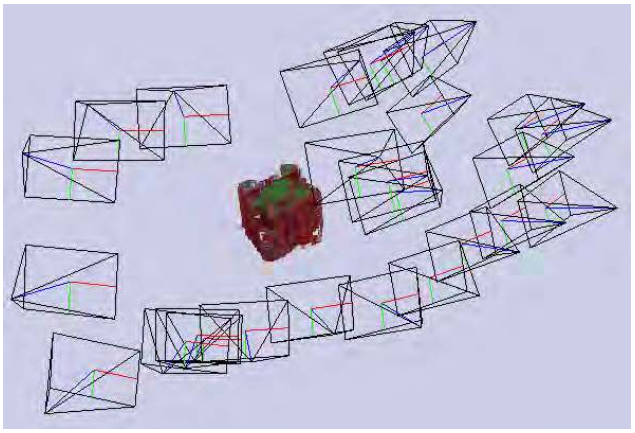

Figure 4: The camera positions and generated model for the sequence shown in Figure 1.

Before using $d_{\mathbf{x}}(\mathcal{D},\mathcal{I})$, we evaluate if it is reliable by considering the ratio:

$$r = \frac{W_{\mathbf{x}}^{(d)}}{\sum_{\mathbf{y}\in S} G_{\sigma_s}(0) G_{\sigma_r}(0)} . \quad (9)$$

If r is too small, only a few pixels, or only far away pixels, contributed to $d_{\mathbf{x}}(\mathcal{D},\mathcal{I})$, and we then leave $\mathbf{x}$ without any depth data.

4 Experimental Results

We experimented the proposed methods on a desktop PC. The PC used for the experiment has a 3.2 GHz CPU and an NVIDIA GeForce GTX 580 graphics card. The Kinect captures a 640×480 color image and a depth map up to 30 Hz. The official camera driver for Kinect is not currently available, and we used a third-party implementation [2] for image and depth map capture.

4.1 Training

Figure 1 shows some pictures captured during training for an object. The object is first aligned with a known image pattern, and the world coordinate frame is defined by the user (Figure 1(a)). During the motion of the camera, templates of another views are captured (Figure 1(b)-(d)). Figure 4 shows the camera poses for the templates and the 3D points after training on a miniature building. The clouds of 3D points extracted at different poses are correctly overlapping showing the tracking performance.

4.2 Tracking

Tracking results for complex and texture-less objects are shown in Figures 1 and 5. We tried keypoint-based detection methods for detecting and tracking these objects, but because too few keypoints can be extracted or matched, they did not work properly. By contrast, the proposed methods detect and track these objects successfully under various viewpoints, partial occlusions, and fast motion even without prior object data.

Figure 6 shows the stability of the pose estimates. The graph plots a given 3D point projected with the estimated object pose through a sequence where the camera and the object are both stationary. We compared the results after ICP applied on the raw depth map, ICP on the depth map enhanced with our method, and ICP on the enhanced depth map followed by the optimization on image contour alignment. The standard deviations of the projection coordinates are respectively $(0.43, 0.33)$, $(0.40, 0.25)$, and $(0.23, 0.23)$, showing our methods reduce jitter.

[2] CL NUI Platform is available at `http://codelaboratories.com/nui`.

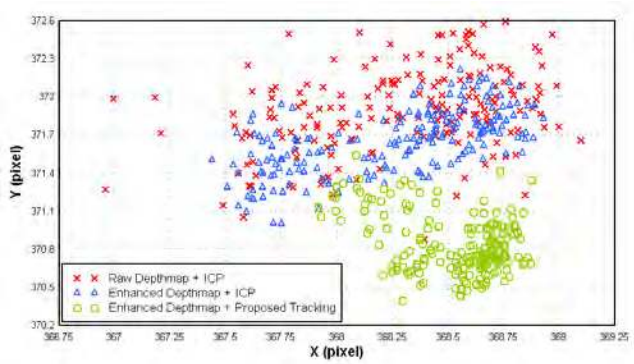

Figure 6: Estimated poses for a static viewpoint and different methods. Each point is the projection of the $(0, 0, 300\ mm)$ 3D point (in the object coordinate frame) for a different input image. The red points are obtained with ICP applied on the raw depth map, the blue points with ICP on the depth map enhanced with our method, and the green points with ICP on the enhanced depth map followed by the optimization on image contour alignment. The point clouds are less and less spread, showing our methods reduce jittering.

Table 1: Registration error of the objects shown in Figure 5. The registration error was measured while the object and the camera are stationary.

Object	**Registration Error** (Raw depth map + ICP)	**Registration Error** (Enhanced depth map + Tracking)
Hon-chun-eui	0.9948	0.6633
White building	0.8617	0.5280
Cactus	1.0605	0.7869
Second building	0.9020	0.7952
Toy ship	0.8703	0.6742

4.3 Depth Enhancement

The original depth maps acquired by the Kinect are typically noisy as shown in Figure 7(a, b). The overall shape is captured, however the object boundaries are noisy and not completely acquired. Figure 7 compares the results of joint bilateral upsampling and of the enhancement method we proposed in Section 3.4 to these maps.

The depth map from the Kinect on a planar surface is also noisy as shown in Figure 8(b). In frame-to-frame tracking, the noise decreases the pose accuracy and causes jitter since this noise is time-variant. The depth map enhanced by our method is shown in Figure 8(c) showing reduced noise.

We conducted an experiment to evaluate the stability of the estimated 3D pose for various objects and the results are presented in Table 1. Here we measured the registration error between a trained points and the input depth map. In every case, the proposed methods yield better registration.

4.4 Computation Time

Table 2 details the computation time for the proposed method. We implemented the depth map enhancement using GPU programming with CUDA. The given time also includes the time required for memory transfer between the main memory and the video memory. The other steps are running on only the CPU. The training does not take more than a few milliseconds for DOT, and the processing time for training is negligible. Note that during frame-to-frame tracking, the detection step is omitted, and tracking becomes faster than the camera frame rate. Since the depth enhancement performs only

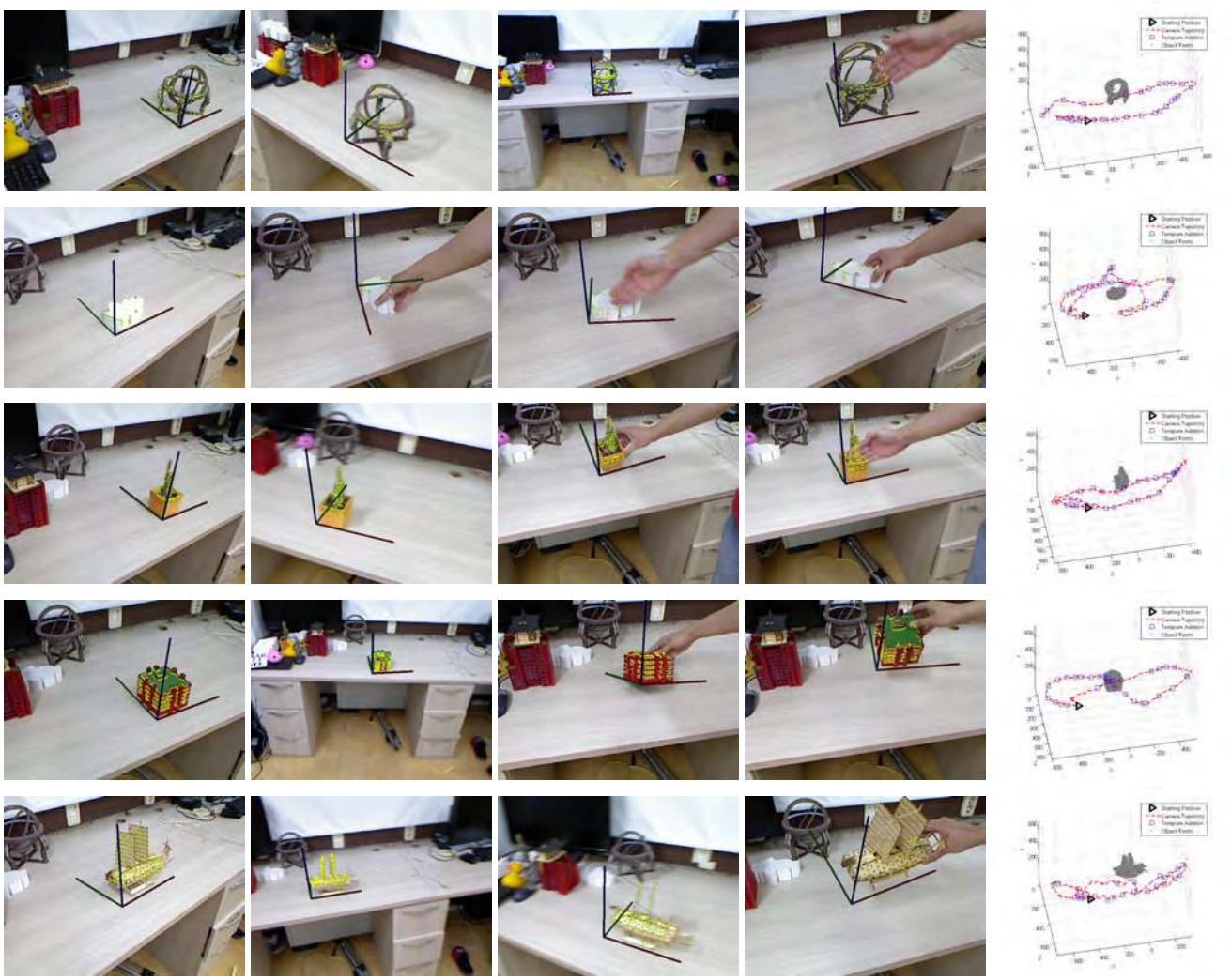

Figure 5: Tracking of various objects. First row: A hon-chun-eui (an old armillary sphere in Korea); Second row: A miniature building; Third row: A cactus; Fourth row: Another miniature Building; Fifth row: A toy ship. The poses are correctly estimated under various viewpoints, partial occlusions, and fast motion. The rightmost column shows the estimated trajectory of the camera with respect to the object; For visibility, we show only the training part. *The complete sequences are available in the supplementary video.*

Table 2: Average processing time in milliseconds. The number of templates was 26 for this experiment.

Step	Mean	Standard Deviation
Calibration & Enhancement (GPU)	5.57	0.19
Detection by DOT	13.4	1.18
Model Generation	1.15	0.06
Pose Estimation	6.58	0.56
Total (Detection+Pose Estimation)	26.71	1.31
Total (Frame-to-Frame Tracking)	13.31	0.53

once for a frame, our current implementation of the frame-to-frame may handle up to 3 objects at 30 Hz, which is the frame rate of the Kinect.

5 Conclusion and Discussion

We presented a texture-less object detection and 3D tracking with online training using a depth camera. The proposed method eliminates the requirement of a prior object model, since any data for detection and tracking are obtained on the fly. We adopted DOT to compute an initial estimate of the 3D object pose. Depth information together with object contours in the image are used to refine the pose. We also developed a method to enhance the depth map. The tracking performance was demonstrated on several challenging objects.

It is an issue that the RGB-D may not acquire the depth map under strong infrared light. Therefore, using an RGB-D cameras is mostly limited to indoor environment currently.

Acknowledgements

This work was supported by the Global Frontier R&D Program on <Human-centered Interaction for Coexistence> funded by the National Research Foundation of Korea grant funded by the Korean Government (MEST)

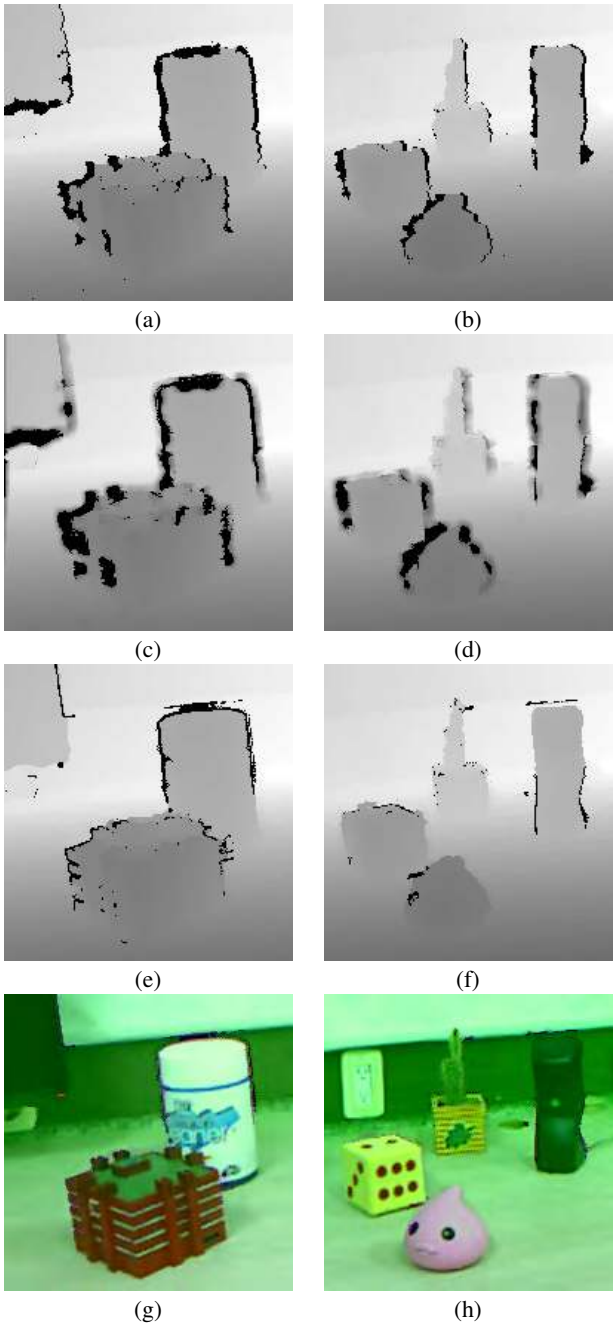

Figure 7: Two depth maps before and after enhancement. (a, b) show the raw depth maps produced by the Kinect. (c, d) are the results of Joint Bilateral Upsampling. The holes are not correctly handled. (e, f) show the enhanced results by the method we propose, which cleans the boundary and shape of the objects. Note the dramatic improvement on the left part of the miniature building. (g, h) show the overlay of the enhanced depth map on the color image.

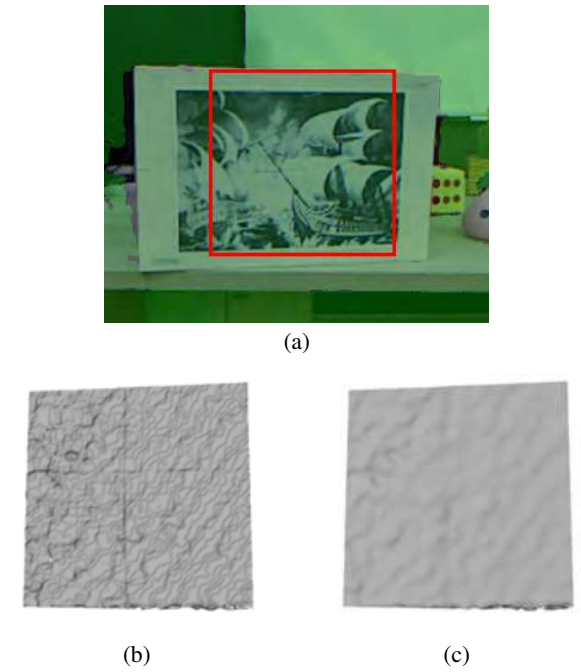

Figure 8: The depth of a planar region. (a) is the object and considered region. The rectangular region is modelled. (b) is the reconstructed surface with the raw depth map. (c) is reconstructed with the enhanced depth map.

REFERENCES

[1] P. Besl and H. McKay. A Method for Registration of 3-D Shapes. *PAMI*, 14(2):239–256, Feb. 1992.

[2] J. Diebel and S. Thrun. An Application of Markov Random Fields to Range Sensing. In *NIPS*, 2005.

[3] T. Drummond and R. Cipolla. Real-Time Visual Tracking of Complex Structures. *PAMI*, 24(7):932–946, 2002.

[4] V. Ferrari, T. Tuytelaars, and L. Van Gool. Object Detection by Contour Segment Networks. In *ECCV*, pages 14–28, June 2006.

[5] C. Harris. Tracking with Rigid Models. In *Active Vision*, pages 263–284. MIT Press, 1992.

[6] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab. Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects. In *CVPR*, 2010.

[7] F. Jurie. Solution of the simultaneous pose and correspondence problem using gaussian error model. *CVIU*, 73:357–373, 1999.

[8] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint Bilateral Upsampling. *ACM Transactions on Graphics*, 26(3), 2007.

[9] D. Kotake, K. Satoh, S. Uchiyama, and H. Yamamoto. A Fast Initialization Method for Edge-based Registration Using an Inclination Constraint. In *ISMAR*, Nov 2007.

[10] D. G. Lowe. Robust Model-based Motion Tracking through The Integration of Search and Estimation. *IJCV*, 8(2):113–122, 1992.

[11] D. G. Lowe. Distinctive Image Features from Scale-invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[12] M. Ozuysal, P. Fua, and V. Lepetit. Fast Keypoint Recognition in Ten Lines of Code. In *CVPR*, 2007.

[13] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In *BMVC*, London, September 2009.

[14] Y. Park, V. Lepetit, and W. Woo. ESM-Blur: Handling & Rendering Blur in 3D Tracking and Augmentation. In *ISMAR*, 2009.

[15] E. Rosten and T. Drummond. Fusing Points and Lines for High Performance Tracking. In *ICCV*, Oct 2005.

[16] C. Tomasi and R. Manduchi. Bilateral Filtering for Gray and Color Images. In *ICCV*, Jan 1998.

[17] L. Vacchetti, V. Lepetit, and P. Fua. Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. In *ISMAR*, Nov 2004.