RGB-D TRACKING OF COMPLEX SHAPES USING COARSE OBJECT MODELS

Agniva Sengupta, Alexandre Krupa, Eric Marchand
Univ Rennes, Inria, CNRS, IRISA, France

ABSTRACT

This paper presents a framework for accurately tracking objects of complex shapes with joint minimization of geometric and photometric parameters using a coarse 3D object model with the RGB-D cameras. Tracking with coarse 3D model is remarkably useful for industrial applications. A technique is proposed that uses a combination of point-to-plane distance minimization and photometric error minimization to track objects accurately. The concept of 'keyframes' are used in this system of object tracking for minimizing drift. The proposed approach is validated on both simulated and real data. Experimental results show that our approach is more accurate than existing state-of-the-art approaches, especially when dealing with low-textured objects with multiple co-planar faces.

Index Terms— Visual tracking, object tracking

I. INTRODUCTION

The research related to localizing and tracking simple geometric shapes is extensive [1] and is being studied since a long time [2]. Existing approaches for real-time, 6 degrees of freedom (DoF) object tracking can be broadly divided into three categories. The *first* group consists of algorithms that uses probabilistic models for tracking shapes. For example, [3] aims to maximize the discrimination between background and foreground of a scene while simultaneously tracking the 6DoF pose of an object with known 3D model. However, this method used only RGB images as input. [4] extended this method to depth cameras, adding more cues like local image appearance, depth discontinuities, optical flow, and surface normal to inform the segmentation decision in a conditional random field model. Similar approaches have been used to track articulated objects [5] with remarkable precision, but the tracking of articulated joints remain beyond the scope of our discussion. Particle filtering based approaches have been parallelized using GPU and implemented towards tracking rigid objects [6]. A probabilistic generative model based tracker using 3D signed distance field (SDF) was proposed in [7], but it needs accurate object model. The *second* group of algorithms use learning based techniques to track objects. [8] proposes a pose estimation technique of complex objects under heavy clutter, but needs color gradient information of the model for training. However, in our approach, we assume that the accurate model of the object, as well as any color/texture based model are not available. [9], [10] and [11] propose increasingly accurate techniques for pose estimation, but they cannot be considered real time. [12] and [13] are both a dynamic SLAM algorithm utilizing Mask R-CNN for segmentation and performs object tracking as a byproduct. The per-object tracking accuracy is never evaluated explicitly. The *third* group of algorithms use well known minimization techniques to track objects using geometric and photometric constraints. Among these approaches, Iterative Closest Point (ICP) is a popular algorithm [14] that has been used extensively. Many variants of ICP have been proposed for probabilistic [15] implementation of point set registration. KinectFusion [16] is a popular research work for tracking and reconstruction of static scene using RGB-D sensor. In this context, SLAM and object tracking have been tackled interchangeably by the same authors [17], while some others have used KinectFusion as a tool for benchmarking their own object tracking algorithm [7]. CoFusion [18] is a dynamic SLAM system that uses a combination of dense ICP and photometric error minimization between the model and the current frame to track objects. The approach we propose, in terms of the framework for object tracking, is somewhat closer to CoFusion [18]. However, we do not undertake explicit motion segmentation, we are not interested in reconstruction and we use the concept of keyframes. Moreover, the presence of coarse object model makes it a different problem statement altogether.

In this paper we focus on the high-accuracy tracking of rigid, complex shapes with approximately known geometry using depth cameras. We consider the *coarse model* to be a highly decimated, minimal representation of the object model (instead of a high-polygonal, detailed CAD model), containing very few triangular faces at best. The model does not contain any color or textural information. This is easy to generate and can also be rendered without a depth scanner (e.g: using manual measurements), making it suitable for various industrial applications. Our approach is significantly robust to measurement errors in the *coarse model*. We are interested in tracking all the 6 degrees of freedom [19] of the object. The proposed approach is validated both quantitatively and qualitatively in the subsequent sections. The key contributions of this paper are:

- Combining point-to-plane distance minimization and photometry for tracking of complex objects using coarse model
- Using the concept of 'keyframe' in object tracking for

increased robustness
- Accurate tracking of objects with inaccurate and coarse object models

II. METHOD

The proposed approach to track rigid objects is a combination of point-to-plane distance minimization and photometric error minimization between frames.

II-A. Notation

We work with two types of data: 1) calibrated and registered depth and grayscale images provided by a depth camera, and 2) simulated data containing 3D points along with their corresponding grayscale intensity. Using conventional notations, we denote the depth data as $\mathbf{\Omega} = \Big((\mathbf{P}_1, c_1), ...(\mathbf{P}_N, c_N)\Big)$, where $\mathbf{P}_i = (X_i, Y_i, Z_i)$ and c_i is the image intensity value of the point $\mathbf{P}_i$, expressed in the camera centered coordinate frame. We define a function $\pi(\cdot)$ that acts upon any 3D point and projects it to an image plane using the pinhole camera model such that $\mathbf{p}_i = \pi(\mathbf{P}_i)$. The function $c_i = \mathbf{I}(\mathbf{p}_i)$ provides the image intensity of the point $\mathbf{p}_i$. $\nabla\mathbf{I}_{i,x}$ and $\nabla\mathbf{I}_{i,y}$ gives the gradient of the image along X and Y axis.

We also consider the model of the object for tracking. We represent it as a surface 3D mesh composed of a set of planes given by $\mathcal{M} = \Big\{\{\mathbf{n}_1, d_1, \mathbf{Q}_1\}, \{\mathbf{n}_2, d_2\mathbf{Q}_2\}.....\{\mathbf{n}_M, d_M, \mathbf{Q}_M\}\Big\}$ where $\mathbf{n}_j = (n_j^X, n_j^Y, n_j^Z)$ is the normal to the j-th plane of the mesh which lies at a perpendicular distance of d_j from the origin of the camera-centered coordinate frame. $\mathbf{Q}_j$ denotes the set of vertices of the bounding triangle for every plane. The n-th frame from the depth sensor (or from a simulated data) is denoted by C_n, while certain frames get tagged as keyframes $C_n \Rightarrow C_n^K$. O denotes the object centered coordinate frame. These keyframes serve as the reference for photometric tracking for all subsequent frames. We use the notation ${}^A\mathbf{T}_B = \left[\begin{array}{c|c} {}^A\mathbf{R}_B & {}^A\mathbf{t}_B \\ \hline 0 & 1 \end{array}\right]$ to denote the rigid transformation from any arbitrary Cartesian frame A to B, where $\mathbf{T} \in \mathbb{SE}(3)$. In between frames, the motion of the object with respect to its previous pose is denoted by $\mathbf{q} = ({}^A\mathbf{t}_B, \theta\mathbf{u})$, where θ and $\mathbf{u}$ are the angle and axis of the rotation ${}^A\mathbf{R}_B$. The time derivative of $\mathbf{q}$ is given as ${}^{n-1}\mathbf{v}_n = \delta\mathbf{q}$, where $\mathbf{v} \in \mathfrak{se}(3)$ is the velocity screw.

II-B. Tracking

For each frame, we minimize two cost functions that depend on a geometric term based on point-to-plane alignment and a photometric term that minimizes the intensity difference between the predicted image (based on an estimate of inter-frame transformation) and the projected image (from the point cloud).

Point-to-plane Distance Minimization: For the geometric term, we minimize the point-to-plane distance between the 3D points in the n-th frame $\mathbf{\Omega}_n$, with respect to the set of planes registered with the point cloud in the previous frame $\mathbf{\Omega}_{n-1}$. This is given by minimizing the distance error:

$$\mathbf{e}_i^{dist}({}^n\mathbf{q}_{n-1}) = \Big(\big({}^n\mathbf{R}_{n-1}\mathbf{P}_i + {}^n\mathbf{t}_{n-1}\big)\cdot\mathbf{n}_k\Big) - d_k \quad (1)$$

where i denotes a specific point in the pointcloud and k is the index of the plane in the object model, to which it corresponds. We address the topic of this correspondence in the next subsection.

The cost function is optimized using Gauss-Newton optimization and the Jacobian is obtained by partial differentiation of eq. (1) with respect to ${}^n\mathbf{q}_{n-1}$, given by:

$$\mathbf{J}_i^{dist} = \begin{bmatrix}\mathbf{n}_k^\top & \big[\mathbf{n}_k\big]_\times \mathbf{P}_i{}^\top\end{bmatrix} \quad (2)$$

Photometric Minimization: The initial estimate for the transformation between the last keyframe C_p^K and the current data frame C_n used in photometry is given by: ${}^p\mathbf{T}_n = \Big({}^n\mathbf{T}_O \cdot \big({}^p\mathbf{T}_O\big)^{-1}\Big)^{-1}$ where p denotes the frame number for the last keyframe. Starting with this initial estimate of the transformation between the last keyframe and current frame, the image intensity error we seek to minimize for each image point is given as:

$$\mathbf{e}_i^{img}\Big({}^p\mathbf{q}_n\Big) = \mathbf{I_k}\Big(\pi\big(\mathbf{P}_i\big)\Big) - \mathbf{I_n}\Big(\pi\big({}^p\mathbf{R}_n\mathbf{P}_i + {}^p\mathbf{t}_n\big)\Big) \quad (3)$$

The Jacobian used for this minimization is:

$$\mathbf{J}_i^{img} = \nabla\mathbf{I}_i \cdot \mathbf{A}_i \quad (4)$$

where $\nabla\mathbf{I}_i = \begin{bmatrix}\nabla\mathbf{I}_{i,x} & \nabla\mathbf{I}_{i,y}\end{bmatrix}$ and $\mathbf{A}_i = \mathbf{f}\cdot\mathbf{B}_i$, given $\mathbf{f} = \begin{bmatrix} f_x & 0 \\ 0 & f_y\end{bmatrix}$, f_x and f_y being the focal lengths and:

$$\mathbf{B}_i = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{X}{Z^2} & \frac{XY}{Z^2} & -(1+\frac{X^2}{Z^2}) & \frac{Y}{Z} \\ 0 & -\frac{1}{Z} & \frac{Y}{Z^2} & -(1+\frac{Y^2}{Z^2}) & -\frac{XY}{Z^2} & -\frac{X}{Z}\end{bmatrix} \quad (5)$$

Optimization: The pose update is given by:

$$\mathbf{v} = -\lambda\big(\mathbf{WJ}\big)^+\mathbf{We} \quad (6)$$

where $\mathbf{J} = \big(\mathbf{J}_1^{dist}, \mathbf{J}_2^{dist}, \cdot\cdot\cdot, \mathbf{J}_1^{img}, \mathbf{J}_2^{img}, \cdot\cdot\cdot\big)$ and $\mathbf{e} = \big(\mathbf{e}_1^{dist}, \mathbf{e}_2^{dist}, \cdots, \gamma\mathbf{e}_1^{img}, \gamma\mathbf{e}_2^{img}, \cdots\big)$ are the stacked Jacobian matrices and error vectors respectively. $\mathbf{v}$ can be interpreted as the velocity screw acting on the camera that transforms it from data frame n-1 to n. The pose update is given as ${}^{n-1}\mathbf{T}_n = {}^{n-1}\hat{\mathbf{T}}_n\mathbf{\Delta T}$, where $\mathbf{\Delta T} = \exp(\mathbf{v})$ and ${}^{n-1}\hat{\mathbf{T}}_n$ represents the previous estimate of the transformation. Here, $\gamma = \frac{\overline{\mathbf{e}^{dist}}}{\overline{\mathbf{e}^{img}}}$ is computed only once per frame, at the first iteration. γ serves as a scaling factor to ensure that the point-to-plane distance error and the photometric error are at a similar order of magnitude. $\mathbf{W}$ is a diagonal matrix of weights obtained from the m-estimator for being robust to outliers. We use Tukey biweight operator as the m-estimator

[20]. For any given residual $\mathbf{e}_i$, we define the Tukey operator as:

$$\rho(\mathbf{e}_i, K) = \begin{cases} \left(\frac{\mathbf{e}_i^6}{6} + \frac{K^2\mathbf{e}_i^4}{2} + \frac{K^4\mathbf{e}_i^2}{2}\right) & \text{if } |\mathbf{e}_i| < K \\ \frac{1}{6}K^6 & \text{else} \end{cases} \quad (7)$$

where $K = 4.7\hat{\sigma}$ and $\hat{\sigma} = \left\{1.48 \times \mathbf{Median}(|\mathbf{e}_i - \tilde{\mathbf{e}_j}|)\right\}$

II-C. Point Correspondence

As indicated in eq. (1), there is a need to associate every 3D point $\mathbf{P}_i$ with one of the planes of the model, which can be represented by the tuple $\{\mathbf{n}_k, d_k, \mathbf{Q}_k\}$, the k-th plane of $\mathcal{M}$. At every frame, we know the value of ${}^{C_{n-1}}\mathbf{T}_O$. We project the 3D points $\mathbf{Q}_k$ for all visible planes into the image obtained in the current frame. This gives us a set of 2D polygons in the image, each representing one of the visible faces. All the points in the image (x_i, y_i) that intersects with a particular plane obtained from $\pi\left(\mathbf{Q}_k\right)$ are considered to be associated with this plane.

II-D. Selection of Keyframe

We use keyframes to reduce drift in frame-to-frame photometric tracking. The estimate of the transformation between the last keyframe and the current frame is given by ${}^{C_k}\mathbf{T}_{C_n}$. We decompose this transformation matrix into the translational component (t_x, t_y, t_z) and the $\theta\mathbf{u}$ rotational component. We define a variable p_k, such that:

$$p_k = \begin{cases} 1 & \text{if } ||(t_x, t_y, t_z)|| > 0.05 \text{ or } \theta > 0.15 \\ 0 & \text{else} \end{cases} \quad (8)$$

Here, the distance threshold is in millimetre while the angular threshold is expressed in radian.

III. RESULTS

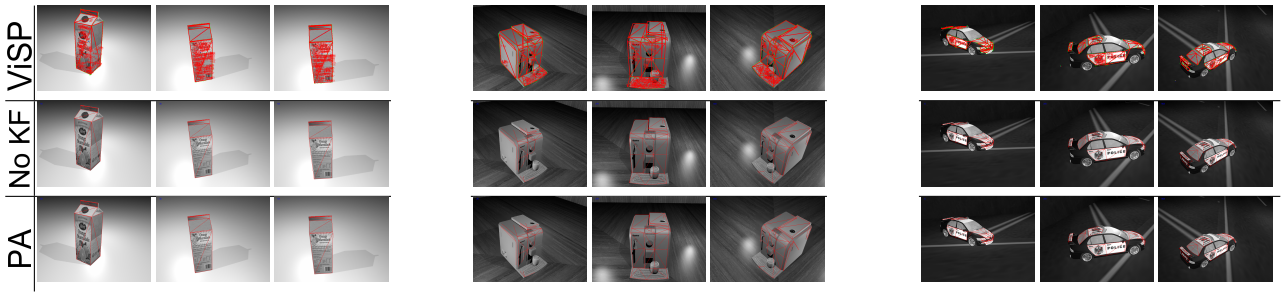
For quantitative analysis of the tracker, we generate three short sequences of simulated RGB-D dataset, comprising of three objects: a *marmalade container*, a *coffee machine* and a *simulated car*. A very minimalist model is used to track these objects. We do not require the model to be continuous or topologically closed. We end up with a very rough approximation of the shape of the object, comprising of 24, 48 and 62 faces for the three objects respectively. Initialization of the tracker is done using the ViSP library [21], by matching keypoints detected in the very first image with those extracted in the training images using an approach similar to [22]. Interested readers may look into more recent techniques, such as [8], [23] and [24] for handling the initialization problem. Only a few model-free tracking and reconstruction algorithm are publicly available along with their open source code (e.g: [16], [18]), but none of them are directly applicable to model-based object tracking with RGB-D data. We compare the proposed method with two recent approaches: **a)** 'edge + keypoint + depth tracker'from [25] (denoted in the figures as *ViSP*), and **b)** stacked error minimization of point-to-plane distance with photometric error, without using keyframes (denoted as *No KF*). *No KF* is close to the tracking module of [18]. However, we use keyframes instead of coarse-to-fine spatial pyramid. The proposed approach is denoted as *PA* in the figures.

The comparison of the output of the various tracking methods[1] with the ground-truth is summarized in Table I. The *marmalade container* sequence (Fig. 1a) shows accurate tracking with all the approaches. The proposed approach manages to outperform the other two methods, although by a small margin. In the *coffee machine* sequence, the proposed approach outperforms *ViSP* significantly. As shown in frame 275 of Fig. 1b, *ViSP* shows a noticeable drift while tracking a set of co-planar faces with not enough image features in it. This is the only instance where the tracking resulted in a visually noticeable drift. *No KF* alone, does not solve the issue of the drift completely, but the proposed approach eliminates the visible drift and the positional tracking is 83.09% better than that of *ViSP*. *Simulated car* is a bit more challenging sequence due to larger inter-frame motion towards the end. *No KF* shows very low accuracy in this sequence. In the rotation, *ViSP* outperforms the proposed approach by average of $0.51°$ over the entire sequence. This drift is not noticeable visually, and happens because a combination of a large number of robust feature points and edges makes it easier for *ViSP* to track the overall orientation of the car, while larger inter-frame motion disadvantages the photometric minimization. However, the proposed approach is more accurate than *ViSP* in terms of translation.

Across the three sequences, it can be concluded that the proposed approach performs better than both *No KF* and *ViSP*. For brevity, we show the tracking error plots of only *ViSP* and the proposed approach in Fig. 2a and Fig. 2b. For the real dataset, all the object models were constructed using manual measurements of the object. The *banana* model is only a rough approximation of the shape of the real banana. Both the *box* and the *car* got accurately tracked (for visual validation, refer to Fig. 3), despite the obvious inaccuracies in the model. The proposed approach was not affected by the moderate occlusion of the objects by the hand. There were some slippage of the model from the actual object while tracking the *banana*. However, it never completely looses tracking.

We tested our algorithm on an Intel Core i7-6600U CPU with 16 GB RAM. Running on a single core of the system, without SSE optimization and without using any GPU, the basic C++ code written for implementing the proposed approach achieves a runtime of 100 - 160 ms per frame, including data capture, tracking and display on a simple GUI. It can be envisaged that with either SSE optimization or with the use of GPU, the overall algorithm can run much faster, if required.

[1]All results can be viewed at: vimeo.com/316228510



(a) *Marmalade Container*: Frame number: 50, 627, 650 (b) *Coffee Machine*: Frame number: 25, 275, 600 (c) *Simulated Car*: Frame number: 6, 200, 553

Fig. 1: Comparison of tracking results from 'ViSP Generic Tracker' [25] (**ViSP** - first row), 'Point-to-plane + Photometry without Keyframes' (**No KF** - second row) and the 'Proposed Approach' (**PA** - third row)

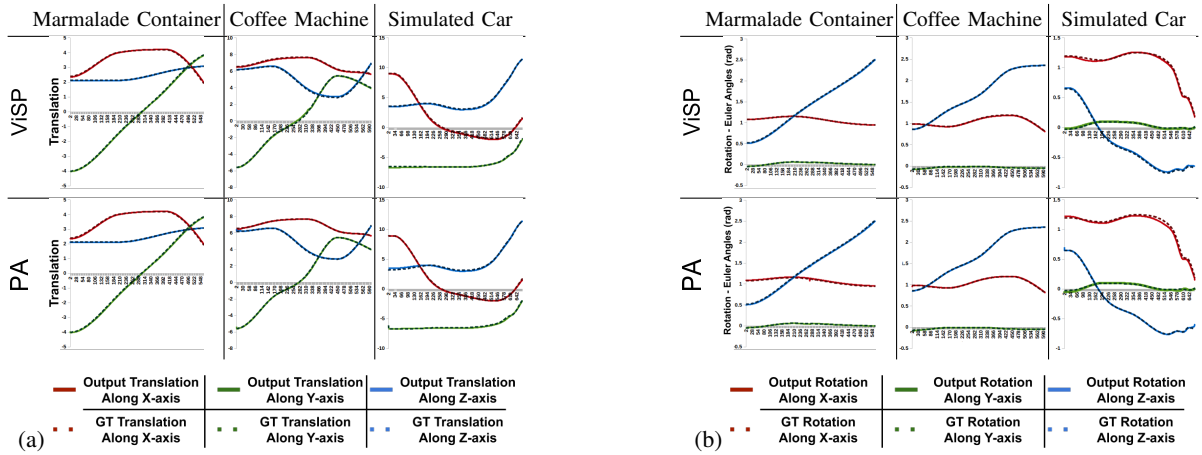


Fig. 2: Comparison of **a)** Translation along X, Y, Z axis, plotted against groundtruth (*GT*), and **b)** Rotation (in radian) along X, Y, Z axis, plotted against groundtruth (*GT*). X-axis shows the frame number, Y-axis represents the translation and rotation respectively

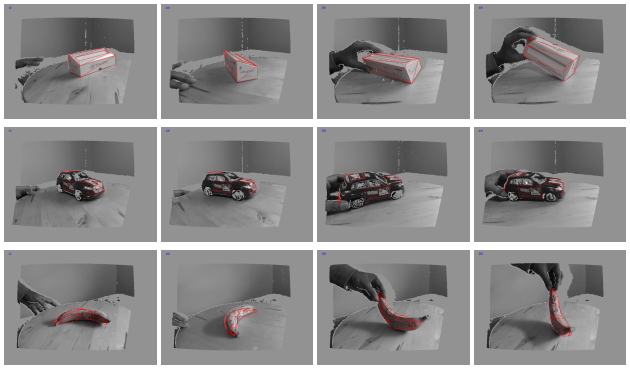


Fig. 3: Tracking results from real data captured using Intel RealSense for a) a box, b) a toy car, c) a banana

Table I: Summary of RMSE values

	Marmalade Container		**Coffee Machine**		**Simulated Car**	
	Translation	Rotation	Translation	Rotation	Translation	Rotation
No KF	0.045	0.139	0.284	0.032	0.265	0.072
ViSP	0.053	0.004	0.118	0.008	0.183	**0.027**
PA	**0.043**	**0.003**	**0.048**	**0.006**	**0.179**	0.036

The RMSE values for translation and rotation for the three synthetic sequences, obtained by comparison with the groundtruth

IV. CONCLUSION

We present an algorithm that accurately tracks the pose of a complex object. The tracking is robust to occlusion and partial specularity of the scene. We provide validation on both simulated and real data. The proposed approach outperforms one of the best among the open-sourced, model-based, 6DoF object tracking methods. It also outperforms a partial re-implementation of a state-of-the-art tracking method from recent advances in the field of tracking and reconstruction. The proposed approach is an efficient method to track complex objects that a) does not require detailed object model (reducing the setup time in practical applications), b) tracks objects with better accuracy than comparable state-of-the-art approaches.

V. REFERENCES

[1] V. Lepetit, P. Fua, et al. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005.

[2] M. Armstrong and A. Zisserman. Robust object tracking. In *Asian Conference on Computer Vision*, volume 1, pages 58–61, 1995.

[3] V. A Prisacariu and I. D Reid. Pwp3d: Real-time segmentation and tracking of 3d objects. *International journal of computer vision*, 98(3):335–354, 2012.

[4] A. Teichman, J. T Lussier, and S. Thrun. Learning to segment and track in rgbd. *IEEE Transactions on Automation Science and Engineering*, 10(4):841–852, 2013.

[5] C. G. Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg. Probabilistic articulated real-time tracking for robot manipulation. *IEEE Robotics and Automation Letters*, 2(2):577–584, 2017.

[6] C. Choi and H. I Christensen. Rgb-d object tracking: A particle filter approach on gpu. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1084–1091. IEEE, 2013.

[7] CY Ren, VA Prisacariu, O Kähler, ID Reid, and DW Murray. Real-time tracking of single and multiple objects from depth-colour imagery using 3d signed distance functions. *International Journal of Computer Vision*, pages 1–16, 2017.

[8] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[9] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T-K Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3583–3592, 2016.

[10] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchynskyy, and C. Rother. Global hypothesis generation for 6d object pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] Yu Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[12] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. Fusion++: Volumetric object-level slam. In *2018 International Conference on 3D Vision (3DV)*, pages 32–41. IEEE, 2018.

[13] M. Runz, M. Buffier, and L. Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20. IEEE, 2018.

[14] F. Pomerleau, F. Colas, R. Siegwart, et al. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104, 2015.

[15] A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435, 2009.

[16] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[17] R. F Salas-Moreno, R. A Newcombe, H. Strasdat, P. HJ Kelly, and A. J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1352–1359, 2013.

[18] M. Rünz and L. Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4471–4478. IEEE, 2017.

[19] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

[20] A. Ruckstuhl. Robust fitting of parametric models based on m-estimation. *Lecture notes.[Available at https://stat. ethz. ch/wbl/wbl4/WBL4_robstat14E. pdf.]*, 2014.

[21] E. Marchand, F. Spindler, and F. Chaumette. Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4):40–52, December 2005.

[22] C. Choi and H. I Christensen. Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4048–4055. IEEE, 2010.

[23] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014.

[24] B. Tekin, S. N Sinha, and P. Fua. Real-time seamless single shot 6d object pose prediction. *arXiv preprint arXiv:1711.08848*, 2, 2017.

[25] S. Trinh, F. Spindler, E. Marchand, and F. Chaumette. A modular framework for model-based visual tracking using edge, texture and depth features. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 89–96. IEEE, 2018.