6D Pose Estimation for Industrial Applications

Federico Cunico[1], Marco Carletti[1(✉)], Marco Cristani[1,2], Fabio Masci[3], and Davide Conigliaro[4]

[1] Department of Computer Science, University of Verona, Verona, Italy
marco.carletti@univr.it
[2] Istituto Italiano di Tecnologia (IIT), Genova, Italy
[3] The Edge Company, Srl, Rimini, Italy
[4] Humatics, Srl, Verona, Italy
https://www.theedgecompany.net
http://www.humatics.it

Abstract. Object pose estimation is important for systems and robots to interact with the environment where the main challenge of this task is the complexity of the scene caused by occlusions and clutters. A key challenge is performing pose estimation leveraging on both RGB and depth information: prior works either extract information from the RGB image and depth separately or use costly post-processing steps, limiting their performances in highly cluttered scenes and real-time applications. Traditionally, the pose estimation problem is tackled by matching feature points between 3D models and images. However, these methods require rich textured models. In recent years, the raising of deep learning has offered an increasing number of methods based on neural networks, such as DSAC++, PoseCNN, DenseFusion and SingleShotPose. In this work, we present a comparison between two recent algorithms, DSAC++ and DenseFusion, focusing on computational cost, performance and applicability in the industry.

Keywords: Pose estimation · Deep learning

1 Introduction

Estimating the 6D pose of an object is an important task for many applications, such as the interaction of a robot with the real world [35], automotive [9,22], augmented and virtual reality for both entertainment and remote maintenance and training [20,21]. Ideally, a solution should deal with objects of varying shape and texture, show robustness towards heavy occlusion, sensor noise, and changing lighting conditions, while achieving the speed requirement of real-time tasks. Due to the constraints of such applications, it becomes mandatory providing accurate and fast algorithms that are robust to acquisition noise and occlusions.

In recent years, the scientific community has proposed an increasing number of approaches to face the pose estimation problem, with a particular focus

M. Cristani et al. (Eds.): ICIAP 2019 Workshops, LNCS 11808, pp. 374–384, 2019.
https://doi.org/10.1007/978-3-030-30754-7_37

on neural network based methods [14,34]. Deep learning allows to infer the pose without a complete scene description, reducing the information to store and process. Strict space and computational requirements need to be solved on wearable devices [16] and augmented and virtual reality systems [15,32] where energy consumption is a crucial aspect to optimize.

The advent of cheap RGB-D sensors has enabled methods that infer poses of low-textured objects even in poorly-lighted environments more accurately than RGB-only methods: prior works either extract information from the RGB image and depth separately or use costly post-processing steps, limiting their performances in highly cluttered scenes and real-time applications [12]. Nonetheless, precise depth sensors are usually energy consuming while RGB cameras are cheap, lightweight and perfectly fit the demands of mobile and wearable technologies.

Fig. 1. 3D printed model of the electronic component used in the comparisons.

In this paper, we present a comparison among state of the art methods on 6D pose estimation. Our purpose is to find those techniques that are suitable for real and industrial applications. Hence, we selected and tested efficient algorithms which code is publicly available: DSAC++ [4] and DenseFusion [31]. The decision was led by their promising results on different datasets, their efficiency and reproducibility. We investigate the performances of those methods by generating a synthetic dataset in order to estimate the 6D pose of an electronic component of a control panel (see Fig. 1).

The structure of the paper is as follow. In Sect. 2 a brief review of the literature is presented, with an in depth and technical description of DSAC++ and DenseFusion frameworks. Then, we provide qualitative analysis of those methods in Sect. 3. In the last section, we discuss the results.

2 State of the Art

Object pose estimation and the complementary problem of camera localization are open problems in computer vision with many practical applications that

could benefit from their resolution, such as robot manipulation [35], autonomous vehicles [33], wearable devices [16] and virtual and augmented reality [15,32]. All these applications usually need lightweight, fast and robust localization systems that are reliable in cluttered scenes and under heavy object occlusions.

RGB Based. Classical methods rely on detecting and matching keypoints with known object models [1,19] while other methods address the problem by learning how to predict the 2D keypoints [23]. Such methods are not reliable when the system is fed with low resolution images or poor texture information. Despite many attempts provide interesting results, primarily based on neural architectures [27], the lack of depth information prevent reaching precise results.

Depth or Point Cloud Based. Working on 3D information is another way to tackle the pose estimation problem. Recent studies proposed a discretization of the space through voxelization [28] and 3D convolutions [24]. Despite the effective geometrical representation of the data, storing voxels is often prohibitively expensive [29]. Many alternatives have been proposed, working on point cloud data [34] representing urban driving environments. In such cases, depth information is usually enough to retrieve all the geometrical properties of the scene but in indoor tasks (i.e. small object pose estimation), appearance information should be taken into account. In [31] a 2D-3D sensor fusion architecture is proposed and described in the next sections.

RGB-D Based. Cheap depth cameras has spawned many RGB-D pose estimators [2,5,6]. The approaches that fuse both appearance and geometrical information (i.e. RGB and depth) often rely on 3D reconstruction of the scene [13], giving grouping hypothesis that are later validated [2]. Newer methods such as PoseCNN [34] directly estimate 6D poses from image frames where depth is later fused [18] as additional modules in the network architecture. Despite RGB-D based algorithms usually require expensive post-processing, they reach high accuracy and in general outperform RGB-only and depth-only methods.

In this work, we mainly focus on the following methods: DSAC++ [4] and DenseFusion [31], that we describe in details in the next subsections. For the sake of completeness, we also provide a brief description of other competitor, SingleShotPose [30]. We decided to compare those methods because of the following reasons: these are recent and good examples of state of the art approaches, results are promising for real applications, the source code is available and reasonably easy to adapt and run in near real-time on unseen scenarios.

2.1 DSAC++

DSAC++ is a new, fully differentiable camera localization pipeline which has only one learnable component, a fully convolutional neural network for scene coordinate regression. The authors proposed DSAC++ as an upgrade of their previous work DSAC [3]: they propose a probabilistic selection of the candidate

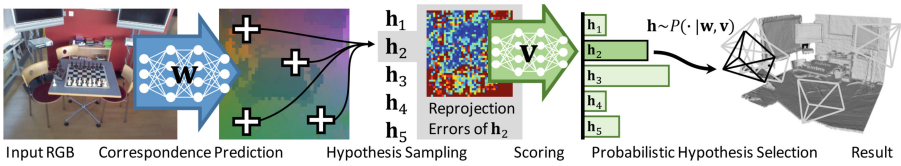

Fig. 2. DSAC pipeline overview. The difference in DSAC++ pipeline is that the scoring is no more performed by a learnable component

poses in order to obtain a differentiable loss function instead of using the deterministic selection based on RANSAC [7]. DSAC was developed with two learnable components: one for regression and one for scoring. However, the scoring CNN tends to overfit and does not generalize well on unseen scenes. DSAC++ uses only one learnable component related to regression (Fig. 2).

The pipeline is the following:

- **Hypothesis generation by regression**: the regression module is used to estimate the depth information of the scene. A certain number of 3D points ($n = 4$) is selected to solve the PnP problem [8] in order to get an hypothesis $\mathbf{h}$ of the camera pose. This operation is performed m-times to obtain m different pose hypothesis.
- **Hypothesis selection**: each hypothesis is ranked through a function $s(\mathbf{h})$ to build a distribution of scores. Instead of selecting the best hypothesis, the system propagates the inlier poses distribution.
- **Pose refinement**: the hypothesis generation is repeated in order to refine the inliner distribution and the consequent pose estimation.

2.2 DenseFusion [31]

In this work, the authors propose an end-to-end deep learning approach for 6D object pose estimation from RGB-D inputs. The main idea is to fuse the extracted features from RGB and depth channels at per-pixel level. This per-pixel fusion scheme enables DenseFusion to explicitly reason about the local appearance and geometry information, enabling the system to handle occlusion. The pipeline of the method is shown in Fig. 3 and could be divided as follow:

- **Semantic segmentation**: the objects in the scene are segmented. The output of this step is a crop of the RGB frame and the corresponding point cloud extracted from the depth frame;
- **Features extraction**: RGB crop and point cloud are processed through a custom fully-convolutional network and a PointNet-based [24] architecture;
- **DenseFusion module**: color and depth embeddings are fused together and processed to generate global features of the selection;
- **Per-pixel pose estimation**: local (per-pixel) and global features are combined and fed into a pose predictor network which returns a per-pixel pose estimation;
- **6D pose estimation**: final pose is the argmax of the output of previous step.

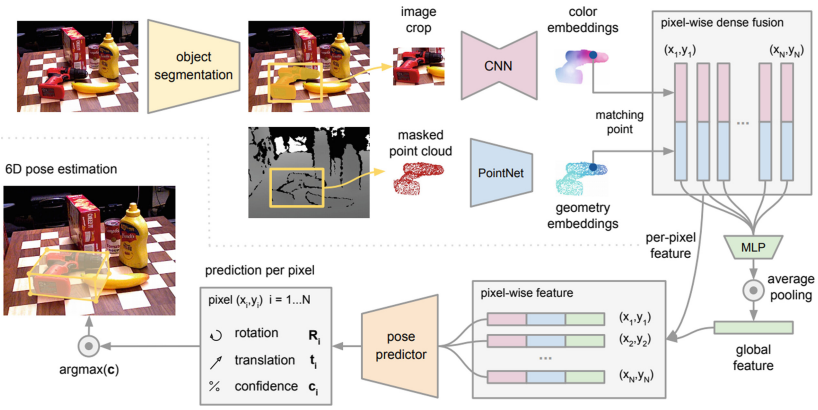

Fig. 3. Overview of DenseFusion. It generates object segmentation masks and bounding boxes from RGB images. The RGB colors and point cloud from the depth map are encoded into embeddings and fused at each corresponding pixel. The pose predictor produces a pose estimate for each pixel. Predictions are voted to generate the final 6D pose prediction of the object. Figure is taken from the original paper [31] (Color figure online)

2.3 SingleShotPose [30]

A state of the art method working on RGB frames is SingleShotPose [30]. The authors propose a single-shot deep CNN architecture based on YOLO [25,26] that takes a single RGB frame as input and directly detects the 2D projections of the 3D bounding box vertices of the objects. Given these 2D coordinates and the 3D ground control points of the bounding box corners, the 6D pose can be calculated algebraically with an efficient PnP algorithm [17]. Complete pipeline is shown in Fig. 4.
SingleShotPose pipeline could be divided in the following main steps:

- **Features extraction**: a single RGB image is processed with the fully-convolutional architecture in Fig. 4a;
- **2D prediction**: the features volume is subdivided into a 2D regular $S \times S$ grid (Fig. 4c) where each cell contains the predictions of 9 control points (the projection of the 3D bounding box and its centroid), the predicted class probabilities and a confidence value. During training, the model predicts the confidence of the bounding box projection according to Eq. 1:

$$c(x) = \begin{cases} e^{\alpha\left(1-\frac{D_T(x)}{d_{th}}\right)}, & \text{if } D_T(x) < d_{th} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $D_T(x)$ is the 2D Euclidean distance, in image space, between the predicted 2D point x and its ground truth; d_{th} is the cut-off threshold of the exponential decreasing function.

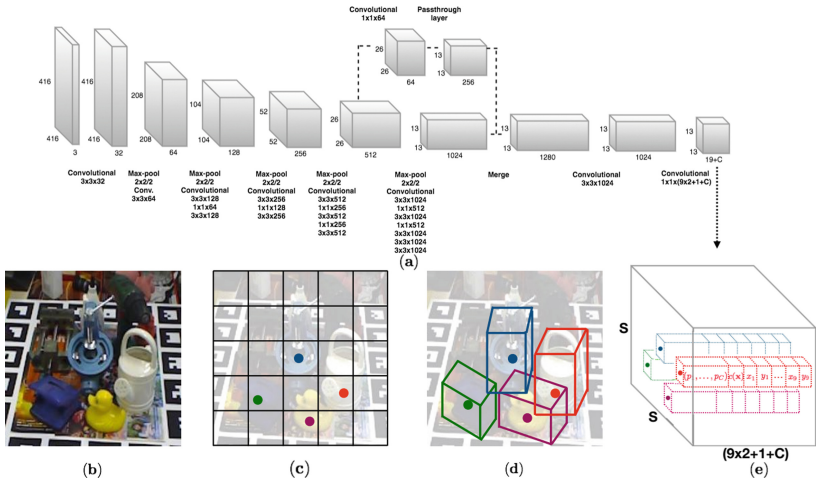

Fig. 4. Overview of SingleShotPose: (a) YOLO-based CNN architecture. (b) Example of input image. (c) The $S \times S$ grid showing cells responsible for detecting objects. (d) Each cell predicts 2D locations of the corners of the projected 3D bounding boxes in the image. (e) The 3D output tensor from our network. Figure is taken from the original paper [30].

- **Forward pass**: at run-time, the network returns the class prediction, its confidence and the 2D projections of the object 3D bounding box. For big objects, 2D coordinates are averaged on the occupied cells.
- **Pose estimation**: the PnP algorithm [17] is used to estimate the 6D pose from correspondences between the 2D and the 3D coordinates.

3 Experiments

We first describe how we create a dataset with ground-truth information followed by the experimental protocol. Then, we describe the setup and metrics we used to perform the evaluation. Qualitative and quantitative results are obtained on our synthetic dataset.

3.1 Dataset

In order to fairly compare the methods described in Sect. 2, we generated a new synthetic dataset in Unity[1] representing an electronic component of a control panel (see Fig. 1). We generated an office environment by modelling common furniture and objects (e.g. chairs, tables, monitor, keyboards): the target object is located on the central table in the world origin. Since in our data the world coordinate systems and the object pose are the same, the two problems of camera

[1] https://www.unity.com.

localization and object pose estimation coincide. The dataset consists of 7000 RGB images, with corresponding depth frames and binary object masks (see Fig. 5). Images are rendered in order to simulate a Microsoft Kinect v1 sensor pointing to the object centroid, with resolution 640×480 pixels and focal length of 26 mm, and the camera positions are uniformly sampled from the upper hemisphere of the scene. Each image is associated with the ground-truth pose of the object in camera space. In Fig. 5 is shown a subset of the rendered viewpoints.

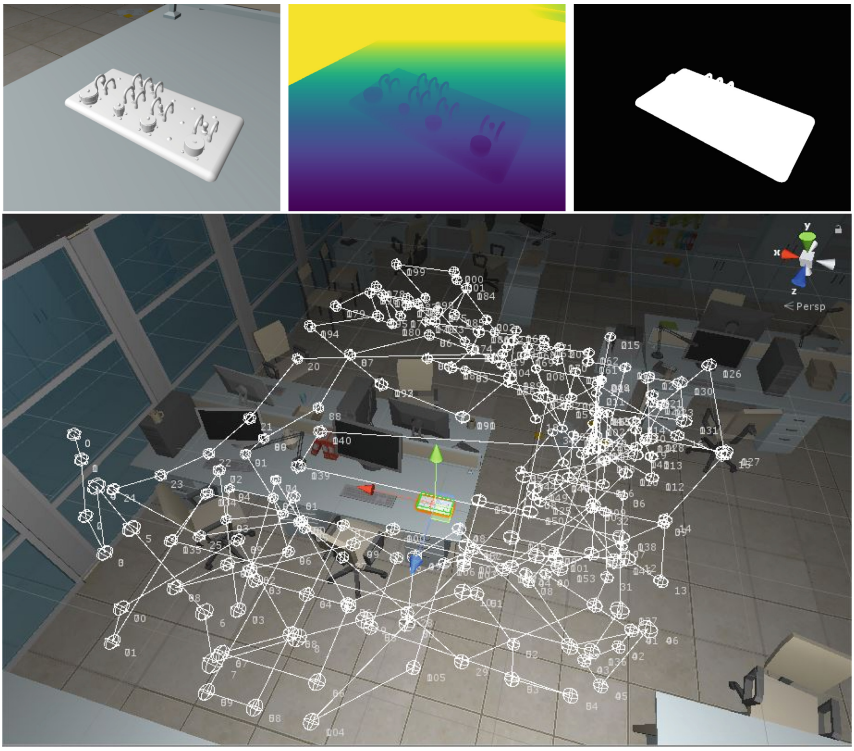

Fig. 5. Top: generated images for target object; from left to right: RGB, depth and binary mask frames. **Bottom**: overview of the scene with subset of visited viewpoints in white. (Color figure online)

3.2 Training

We run the experiments on a Linux machine with 16 GB of RAM, a NVidia RTX 2070 GPU (8 GB of vRAM) with NVidia CUDA 8.0 installed. We managed to run the original implementation and hyper-parameters of DSAC++ (C++, Lua 5.1) and DenseFusion (Python 3). Due to our hardware limitations, DSAC++ took roughly 5 days for training the complete model on our synthetic dataset. Testing runs at 200 ms per image. DenseFusion training converged in almost 3 days on the same machine. Testing is faster than DSAC++, requiring 50 ms per image.

3.3 Metrics

To compare the output poses of DSAC++ and DenseFusion, we compute the error of the rotation and translation components separately. We make use of the mean absolute error for both rotation (expressed as Euler angles, XYZ format) and the translation vector. Table 2 shows the averaged results on our synthetic dataset.

3.4 Evaluation

In our evaluation we have considered three state of the art methods: SingleShotPose, DSAC++ and DenseFusion. The decision was led by their promising results on different datasets and the availability of the source code. In Table 1 we give an overview of those methods. Before testing our dataset, we replicated the original results of all the methods: results have been successfully replicated on their datasets and same hyper-parameters. Tests on our data have been run by maintaining almost the same parameters of the original methods. We aim to train the models on a new domain and also to evaluate the robustness of the architectures by maintaining the same training settings.

Table 1. Main differences among the selected state of the art methods.

	Input data type	Output models	Trained on
SingleShotPose [30]	RGB	One per class	LINEMOD [11]
DSAC [3]	RGB, RGB-D	2	MS7S [10]
DSAC++ [4]	RGB, RGB-D	1	MS7S [10]
DenseFuion [31]	RGB-D	1	LINEMOD [11], YCB [34]

SingleShotPose. This method benefits of a RGB-based architecture which is capable to estimate the 6D pose of an object in the scene. Despite the promising results on the LINEMOD dataset [11], it quickly becomes clear that the training procedure is not simple to run: SingleShotPose needs a heavily annotated dataset, making very difficult to create one. Furthermore, it generates one model (CNN) for each class, occupying a significant amount of memory. This is a big limit in a scenario in which we want to minimize the energy and memory consumption, such as when using wearable devices. Since our goal is to simplify at most the training procedure and reduce the resource usage, we decided to focus on other methods and left SingleShotPose for future analysis.

DSAC++. It is a RGB-based camera localization method which exploits one single architecture to predict the camera pose. From a data generation point of view, the camera localization problem is simpler than the object pose estimation since it does not require any segmentation of the scene. Despite the training procedure

is simple to prepare and run, it takes a long time to converge. Unfortunately, our tests demonstrated that the generalization capabilities of DSAC++ are limited: after training the model on our synthetic dataset, numerical results are higher than expected, showing high prediction errors both in rotation and translation (1.7 radiants and 2.6 m, respectively). Further experiments need to be run, with exhaustive cross-validation of both the dataset quality and hyper-parameters settings. We think the insufficient results are due to the poor texture information in our data: RGB-based methods are known to work well where high-frequency information is available, such as in presence of rich textures and geometrical details (when depth frames are usable). For the sake of completeness, we initially replicated the results of the DSAC++ method, starting with the datasets they proposed. We focused more on scenes representing small-scale contexts, so especially on the Microsoft 7-Scenes [10] RGB-D datasets. The results have been validated correctly, with a precision of camera pose estimation between 1.5 and 3 cm, and around 1° of rotation error.

DenseFusion. The last method we analyzed requires RGB and depth frames and the segmentation of the target object to train a single model for pose estimation. DenseFusion could be considered as a good compromise between space usage, time complexity and ease of training. Convergence of the new model on our dataset took several days. Nonetheless, pose estimations are quite accurate, reducing the average error up to 0.05 radiants on rotation and less than 4 cm on translation. As additional feature, DenseFusion runs in almost real-time (50 ms per image) while the competitors run up to 200 ms.

Table 2. Averaged rotation and translation errors of state of the art methods on our synthetic dataset. Rotation error is in radiants, while translation error is in meters. Standard deviation is in parenthesis.

	Rotation error (rad)	Translation error (m)
DSAC++	1.728 (0.253)	2.594 (1.176)
DenseFuion	0.054 (0.207)	0.038 (0.021)

4 Conclusions

We presented an overview of 6D pose estimation techniques and a comparison between some recent algorithms, DSAC++ and DenseFusion, focusing on performance, ease of training and robustness. Despite RGB-only and depth-only approaches usually need less information and less complex datasets, RGB-D methods demonstrated higher accuracy on average. DSAC++ proposes an intriguing pipeline but our tests demonstrated that its generalization capabilities are compromised if data is poorly textured (i.e. missing high frequency information) and parameters are not precisely tuned. On the other hand, DenseFusion

showed stronger generalization, faster training and testing time. Results are also more encouraging without any specific parameter setting. In conclusion, after our evaluation, DenseFusion appears to be the best choice for 6D object pose estimation relying on RGB-D data. Its efficient implementation perfectly fits industrial and real application constraints, where space and time requirements are strict.

Acknowledge. We thank The Edge Company, Srl for the support to this research.

References

1. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006). https://doi.org/10.1007/11744023_32
2. Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6D object pose estimation using 3D object coordinates. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8690, pp. 536–551. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10605-2_35
3. Brachmann, E., et al.: DSAC - differentiable RANSAC for camera localization. In: CVPR (2017)
4. Brachmann, E., Rother, C.: 6D camera localization via 3D surface regression. In: CVPR (2018)
5. Choi, C., Christensenb, H.I.: 3D textureless object detection and tracking: an edge-based approach. In: IROS (2012)
6. Choi, C., Christensenb, H.I.: RGB-D object pose estimation in unstructured environments. RAS **75**, 595–613 (2016)
7. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**, 381–395 (1981)
8. Gao, X., Hou, X., Tang, J., Cheng, H.: Complete solution classification for the perspective-three-point problem. TPAMI **25**, 930–943 (2003)
9. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The Kitti vision benchmark suite. In: CVPR (2012)
10. Glocker, B., Izadi, S., Shotton, J., Criminisi, A.: Real-time RGB-D camera relocalization. In: ISMAR (2013)
11. Hinterstoisser, S., et al.: Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In: ICCV (2011)
12. Hinterstoisser, S., et al.: Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: Lee, K.M., Matsushita, Y., Rehg, J.M., Hu, Z. (eds.) ACCV 2012. LNCS, vol. 7724, pp. 548–562. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37331-2_42
13. Kehl, W., Milletari, F., Tombari, F., Ilic, S., Navab, N.: Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 205–220. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_13
14. Kendall, A., Grimes, M., Cipolla, R.: PoseNet: a convolutional network for real-time 6-DOF camera relocalization. In: ICCV (2015)
15. Lee, K.: Augmented reality in education and training. TechTrends **56**, 13–21 (2012)

16. Lee, Y.H., Medioni, G.: Wearable RGBD indoor navigation system for the blind. In: Agapito, L., Bronstein, M.M., Rother, C. (eds.) ECCV 2014. LNCS, vol. 8927, pp. 493–508. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16199-0_35
17. Lepetit, V., Moreno-Noguer, F., Fua, P.: EPnP: an accurate O(n) solution to the PnP problem. IJCV **81**, 155 (2009)
18. Li, C., Bai, J., Hager, G.D.: A unified framework for multi-view multi-class object pose estimation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11220, pp. 263–281. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01270-0_16
19. Lowe, D.G.: Object recognition from local scale-invariant features. In: ICCV (1999)
20. Marchand, E., Uchiyama, H., Spindler, F.: Pose estimation for augmented reality: a hands-on survey. TVCG **22**, 2633–2651 (2015)
21. Marder-Eppstein, E.: Project tango. In: SIGGRAPH (2016)
22. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: CVPR (2015)
23. Pavlakos, G., Zhou, X., Chan, A., Derpanis, K.G., Daniilidis, K.: 6-DOF object pose from semantic keypoints. In: ICRA (2017)
24. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: deep learning on point sets for 3D classification and segmentation. In: CVPR (2017)
25. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: CVPR (2016)
26. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: CVPR (2017)
27. Schwarz, M., Schulz, H., Behnke, S.: RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In: ICRA (2015)
28. Song, S., Xiao, J.: Sliding shapes for 3D object detection in depth images. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8694, pp. 634–651. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10599-4_41
29. Song, S., Xiao, J.: Deep sliding shapes for amodal 3D object detection in RGB-D images. In: CVPR (2016)
30. Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6D object pose prediction. In: CVPR (2018)
31. Wang, C., et al.: DenseFusion: 6D object pose estimation by iterative dense fusion. CoRR abs/1901.04780 (2019)
32. Webel, S., Bockholt, U., Engelke, T., Gavish, N., Olbrich, M., Preusche, C.: An augmented reality training platform for assembly and maintenance skills. RAS **61**, 398–403 (2013)
33. Weiss, S., Achtelik, M.W., Chli, M., Siegwart, R.: Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV. In: ICRA (2012)
34. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: a convolutional neural network for 6D object pose estimation in cluttered scenes. RSS (2018)
35. Zhu, M., et al.: Single image 3D object detection and pose estimation for grasping. In: ICRA (2014)