6 DoF Pose Regression via Differentiable Rendering

Andrea Simpsi[0000−0002−2700−4808], Marco Roggerini, Marco Cannici[0000−0002−9217−3552], and Matteo Matteucci[0000−0002−8306−6739]

Politecnico di Milano, Milano MI 20133, IT
{andrea.simpsi,marco.cannici,matteo.matteucci}@polimi.it
marco.roggerini@mail.polimi.it

Abstract. Six Degrees of Freedom (6DoF) pose estimation is a crucial task in computer vision. It consists in identifying the 3D translation and rotation of an object with respect to the observer system of coordinates. When this is obtained from a single image, we name it Monocular 6 DoF Regression and it is a very prominent task in several fields such as robot manipulation, autonomous driving, scene reconstruction, augmented reality as well as aerospace. The prevailing methods used to tackle this task, according to the literature, are the direct regression of the object's pose from the input image and the regression of the objects' keypoints followed by a Perspective-n-Point algorithm to obtain its pose.
While the former requires a lot of data to train the deep neural networks used to accomplish the task, the latter requires costly annotations of keypoints for the objects to be regressed. In this work, we propose a new method to address 6DoF pose estimation using differentiable rendering along the entire pipeline. First, we reconstruct the 3D model of an object with a differentiable rendering technique. Then, we use this information to enrich our dataset with new images and useful annotations and regress a first estimation of the 6DoF pose. Finally, we refine this coarse pose with a render-and-compare approach using differentiable rendering. We tested our method on ESA's Pose Estimation Challenge using the SPEED dataset. Our approach achieves competitive results on the benchmark challenge, and the render-and-compare step is shown to be able to enhance the performance of existing state-of-the-art algorithms.

Keywords: 6-DoF Pose Estimation · Pose Refinement · Differentiable Rendering.

1 Introduction

Six Degrees of Freedom (6DoF) pose estimation from monocular images is a crucial task in computer vision: it consists in obtaining the translation and rotation of an object with respect to a fixed system of coordinates from a single image. This task is very challenging since a single 2D image does not directly carry the

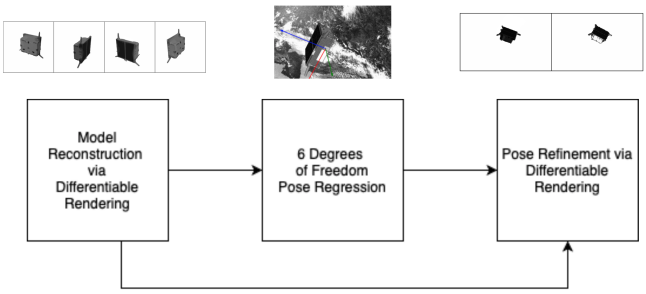

Fig. 1. A schematic version of the proposed pipeline.

depth information, and some form of a priori knowledge is required. In the last years, this topic has obtained a growing interest in different fields, as investigated by Sahin et al. in their review [30]. Among these, the two most known are surely robotics manipulation, where an autonomous device can pick up different objects and place them in the desired target location, and autonomous driving, where pose estimation of other vehicles and pedestrians is the base for collision avoidance in autonomous navigation. The space domain is another field interested in this task, as highlighted in the survey of Opromolla et al. [21]. Indeed, spacecraft pose estimation is a relevant problem in different scenarios, such as formation flying [5], comet and asteroid exploration [14, 7], on-orbit servicing [33, 9], and active debris removal [8, 1]. This interest is further shown by the recent ESA's Pose Estimation Challenges [2, 4], where machine learning enthusiasts from around the world joined together to tackle the 6DoF pose estimation problem of spacecrafts.

The solutions proposed by the participants to the ESA's Pose Estimation Challenge [2] had fallen into the two typical solutions in the literature for pose retrieval from monocular images. Either the pose of the object is directly regressed with a deep neural network from the available image (from now on *Direct Regression*), or some key points of the object are detected and then used by a Perspective-n-Point (PnP) algorithm for pose estimation (from now on *Keypoints Regression*). Both Direct Regression and Keypoints Regression achieved important results in different fields. The former has shown to be robust and easy to apply to different objects, but it is usually reported to be less accurate than Keypoints Regression. Moreover, Direct Regression requires a huge amount of data for training the deep neural network regressor. On the other hand, Keypoints Regression, while obtaining excellent results in the accuracy of the estimated pose, requires a costly annotation of the ground truth position of the key points and the 3D model of the object to be recognized.

In this work, we present a novel method to obtain the 6 DoF pose of an object entirely based on differentiable rendering, overcoming Direct Regression's huge data requirement and Keypoints Regression's need for time consuming 3D model annotations. As from the pipeline in Figure 1, at first an accurate 3D model of an object from a small set of RGBA images is obtained. This model, obtained by

minimizing the visual loss through a differentiable renderer, can then be used to generate new data and annotations. Then, we regress a coarse estimation of the object's pose, which is finally refined with a render-and-compare technique using again differentiable rendering. Experimental results on real data show that our method is competitive in ESA's challenge [2], and that the render-and-compare technique is effective in improving further the already good estimates made by state-of-the-art methods. Furthermore, we have reconstructed an extremely accurate model of the Tango satellite used in the challenge [2], that can be used to generate a virtually unlimited number of new images and thus improve the results even further.

The code of the pipeline is publicly available for experimenting with the proposed algorithm[1] on different domains as the method can work with virtually any kind of object. We show, indeed, that it is possible to reconstruct the 3D model of other objects, and from that reconstruction to apply the entire pipeline we presented, as it does not depend on any specific 3D model.

2 Related Works

6DoF pose estimation with neural networks. Deep neural networks for 6DoF pose regression can be categorized by the representation they use. *Classification methods* discretize the continuous space of rotations and positions [20, 34, 32] and frame the pose regression as a classification problem. Despite being easy to implement, this approach lead two different issues. Firstly, classes are independent one another, while they should be tied together by their proximity. Secondly, discretization in classes may lead to a loss of performance.

In *regression methods*, the neural network regresses directly the position and rotation of the object in a continuous way [13, 19, 35]. Generally, in these models, the rotation is represented as a quaternion, but in the study of Zhou et al. et al. [38] the strength of a 6D vector representation was proved. Su et al. et al. [32] have shown that rendered images can be used to train a neural network for pose regression effectively, thus solving the issue of data requirements. Xu et al. et al. [36] propose a regressor leveraging the use of a differentiable renderer for the estimation of the human 3D pose and obtain superior performance against state-of-the-art, despite being only applicable to humanoid shapes. Other works have used differentiable renderer for 6DoF pose estimation [22, 23, 37], but, unlike in our work, they do not obtain an accurate mesh of the target object.

Finally, *keypoints methods* do not regress directly the 6DoF pose of the object from the image. Instead, some keypoints of the object are regressed first, and then the position and rotation of the object are obtained from them with a PnP algorithm [24, 26]. Despite these methods are generally the most accurate, they often require costly annotations.

Differentiable Rendering. Differentiable rendering represents a class of techniques that enable the integration of a rendering procedure in an end-to-end op-

[1] Code can be found at https://github.com/Simpands/diff-6dof-regression.

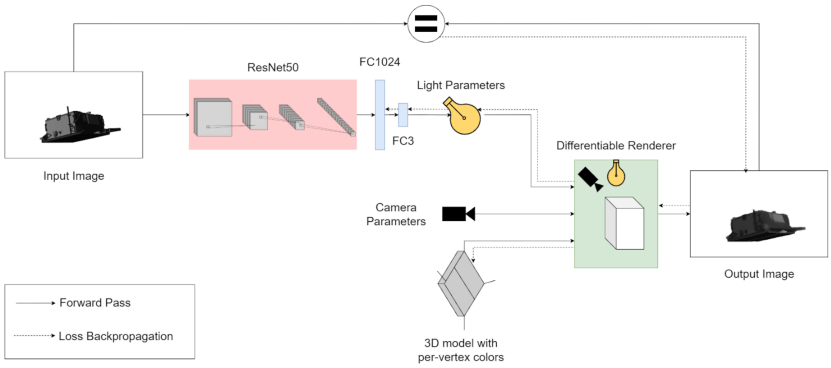

Fig. 2. A scheme of the network used in the model reconstruction (Section 3.1) step.

timization pipeline by obtaining useful gradients of the rendering process. One of the main discriminators among methods for differentiable rendering is the type of data representation they use. We focus here on reviewing the literature that uses a mesh-based representation, as this work relies on the same. Among mesh-based methods, there are two families which can be distinguished by the part of the rendering pipeline they approximate: the first family approximates the backward pass of the rendering process (approximated gradients), while the second family approximates the forward pass (approximated rendering). Regarding the *approximated gradients* family, Loper et al. [18] developed the first general-purpose differentiable renderer, called OpenDR. However, because of its general-purpose nature, OpenDR is often unsuitable for usage with neural networks. Kato et al. [12] proposed the Neural 3D Mesh Renderer (N3MR) and designed its gradients specifically for neural networks training. In particular, they propose to approximate the color's transition across pixels, which is typically discrete, with a smooth transition function with improved differentiability. Rhodin et al. [29] focused on a different solution, *approximated rendering*, that approximates the rasterization process by making objects' edges semi-transparent and thus ensuring differentiability. The Soft Rasterizer of Liu et al. [17] expanded this idea by modeling the contribution of each triangle to a pixel's color as a probability function. Finally, Chen et al. [6] proposed the DIB-R system, a novel approach that deals independently with foreground and background pixels to approximate the rasterization process. The foreground pixels are determined only by one face with a weighted interpolation of its local proprieties, while the background pixels are a distance-related function of global geometry.

3 Method

In this section, we describe in details our method to obtain the 6 degrees of freedom pose of an object using differentiable rendering. Our pipeline, shown in Figure 1, is composed by three different steps. In the the first step, *model recon-*

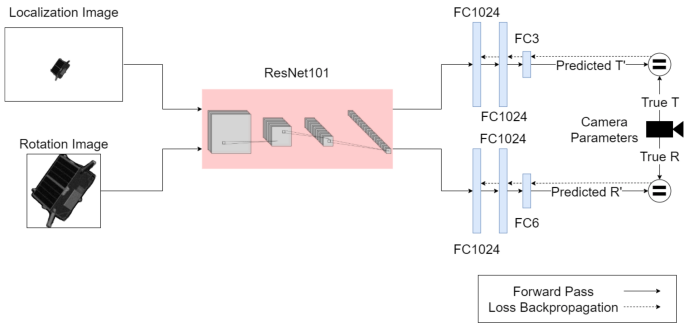

Fig. 3. A scheme of the network used in the pose regression (Section 3.2) step.

struction, we obtain an accurate 3D model of an object from a small set of RGBA images. This model, then, can be used to generate new data and annotations via rendering. In the second step, *pose regression*, we regress a coarse estimation of the object's pose which we then refine in the last step, *pose refinement*, that is a render-and-compare technique using again differentiable rendering.

3.1 Model Reconstruction

The network used for model reconstruction is shown in Figure 2. The goal of this neural network is to regress the position and color of the vertices of a 3D object. This allows us to obtain a good approximation of the true shape of the 3D object which can then be used for model refinement and data generation. Our network is composed by a convolutional neural network used to detect the light direction in the image and a 3D model (initialized as a icosphere with 10242 vertices and 20480 faces). This information (together with the pose of the object, provided by the dataset) is fed into a differentiable renderer to create an image which is then compared to the input image. This comparison is done with an Intersection over Union loss [27] on the alpha channel of the two images and a Mean Absolute Error on the color channels of the two images. From this comparison we optimize both the shape and the colors of the vertices of the 3D object and the parameters of the light detection network. In the Figure 4, images generated with the model reconstructed in this process are shown.

3.2 Pose Regression

The second step of the pipeline is addressed via direct regression both on the translation and on the rotation of the object, which we perform through the network shown in Figure 3. This network is composed by two separate branches: a *Rotation Branch*, used to regresses the rotation of the camera, and a *Localization Branch*, used to regress the position of the camera with respect to the object. Both branches use a feature extractor with the same architecture.

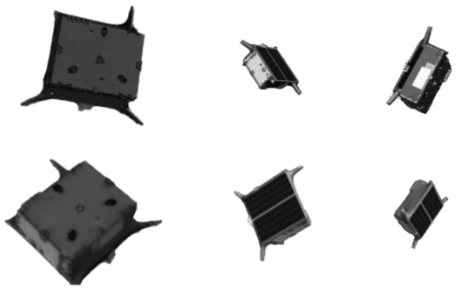

Fig. 4. The top row shows some of the input images used during the model reconstruction step. The bottom row shows some of the images generated from the model obtained during the model reconstruction.

The translation branch takes in input the full image (before any cropping) of size 384×240, which we call *Localization Image*. This image is processed in the feature extractor and then in two fully connected layers with 1024 neurons, followed by a fully connected layer with 3 neurons. This branch outputs the predicted position of the object, encoded as a 3D vector. The rotation branch takes in input only the region of interest representing the object, re-scaled to a fixed dimension of 240×240, and processes it with the feature extractor. We call this image *Rotation Image* as it is used by the network to regress the object rotation. Then, these features are passed trough 2 fully-connected layers with 1024 neurons and finally trough a fully-connected layer with 6 neurons. The feature extractor's output changes between the two branches as the dimension of the input image differs between the Translation Branch and the Rotation Branch (square 240×240 image for the Rotation Branch, w.r.t. a rectangular 384×240 image for the Translation Branch).

The loss used to train this network is given by the sum $L = L_T + \lambda L_R$, where L_T and L_R are the translation and rotation losses defined as it follows:

$$L_T\left(\widehat{T}, T\right) = \frac{1}{3}\left|\widehat{T} - T\right|_1, \quad L_R\left(\widehat{R}, R\right) = \sum_{e \in E}\left|\widehat{R}(e) - R(e)\right|, \tag{1}$$

where T is the ground-truth translation vector and $\widehat{T}$ the predicted one. E is the set of all elements of a 3×3 matrix, $\widehat{R}$ is the predicted rotation matrix and R is the ground-truth one. In this work we set $\lambda = 1$.

The 6D vector predicted as output of the network is converted to the rotation matrix R as follows:

$$\boldsymbol{c}_1 = \frac{\boldsymbol{v}_1}{|\boldsymbol{v}_1|}, \quad \boldsymbol{c}_2 = \frac{\boldsymbol{u}_2}{|\boldsymbol{u}_2|}, \quad \boldsymbol{u}_2 = \boldsymbol{v}_2 - \langle \boldsymbol{c}_1, \boldsymbol{v}_2 \rangle \boldsymbol{c}_1, \quad R = [\boldsymbol{c}_1 \quad \boldsymbol{c}_2 \quad \boldsymbol{c}_1 \times \boldsymbol{c}_2], \tag{2}$$

where $\boldsymbol{v}_1$ is the vector composed by the first 3 values of the 6D output vector, while $\boldsymbol{v}_2$ is the vector composed by the last 3 values. This matrix is orthonormal by construction, and thus a valid 3D rotation matrix. During training the 3D

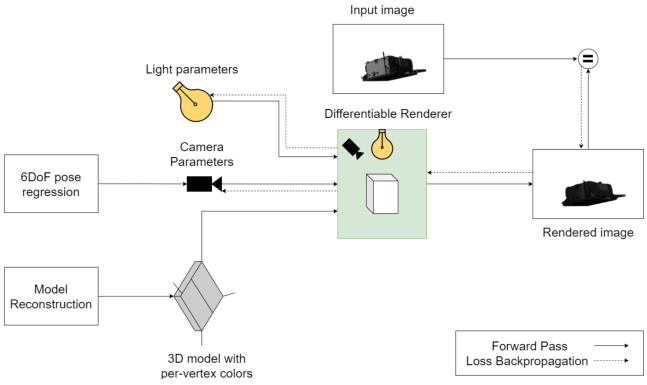

Fig. 5. A scheme of the network used in the pose refinement (Section 3.3) step.

model extracted by the model reconstruction process is used for data generation and augmentation.

3.3 Pose Refinement

In the final step of the pipeline, we use a render-and-compare approach exploiting the differentiable renderer for pose refinement. This technique consists in rendering with a differentiable renderer the image corresponding to a starting guess, e.g., the output of the 6DoF pose estimation process, compare this generated image to the source RGBA image, and then improve the predicted position based on this comparison. The comparison is given by the sum $L_{tot} = L_{IoU} + \lambda L_{col}$ of the Intersection Over Union loss L_{IoU} proposed in [27] and the color loss $\mathcal{L}_{col}$ defined as the Mean Absolute Error between each channel of the two images:

$$L_{col} = \frac{1}{|C|} \sum_{c \in C} \frac{1}{|P|} \sum_{p \in P} |I_1(c,p) - I_2(c,p)| \tag{3}$$

where C is the set of the color channels of the image, P is the set of all pixels of the image and I_1, I_2 are the two images to compare. The optimization procedure is applied with an iterative approach. In particular, as we can notice from Figure 5, the differentiable renderer takes in input:

- A 3D model of the object in the image;
- A rotation matrix R_m, which can be obtained either from 3 Euler angles in case of the Euler regression, 4 un-normalized values in case of the quaternion regression, or 6 un-normalized values in case of the 6D vector rotation regression;
- A translation vector $T = [X_t, Y_t, Z_t]$, representing the translation of the camera from the object;
- A directional light object, comprehensive of the direction of the light [X_l, Y_l, Z_l] and the light color (ambient color, diffuse color and specular color).

Table 1. The improvement obtained with the pose refining step on the score calculated on the Pose Estimation Challenge [2]. The value of UrsoNet in the *competition score* is the value obtained reported in the leaderboard. Instead, UrsoNet *before refinement* is the value obtained using the pre-trained network given by the authors.

Method	Competition score	Pose Error	
		Before Refinement	After Refinement
6D vector	—	0.38221	0.23191
Quaternion	—	0.47944	0.29602
Euler angles	—	0.83889	0.74775
Classification	—	1.20432	1.00247
UrsoNet	0.05546	0.07311	0.05279
UrsoNet Real	0.14763	—	0.14263

During the iterative refinement process, the differentiable renderer synthesizes the image given the current input parameters and compares it with the one observed, from which the object pose has to be estimated. The loss L_{tot} is then computed and its gradient backpropagated to update all renderer's inputs, except for the 3D model, in such a way to make the two images closer and thus improve the estimated pose. Regarding the rotation matrix, we decide not to optimize it directly, or to predict a normalized quaternion, as they both have to lay on a manifold and this cannot be enforced by direct gradient descent methods. Alternatively, manifold-aware gradient descend methods can be used and they can replace the method used in this work directly.

4 Experimental Evaluation

In this section we evaluate the proposed pipeline in the ESA's Pose Estimation Challenge [2]. We use a PyTorch3D's [28] renderer to implement all architectures that require a rendering step and ResNet [11] based backbones as feature extractors. Implementation details are provided in the supplemental materials, along with additional visualizations of the pipeline's outputs.

4.1 Dataset

The dataset used to test our proposed pipeline is the SPEED dataset [31] provided for the ESA's Pose Estimation Challenge [2]. This dataset is composed by in 4 different subsets. The *train* set consists of 12000 synthetic annotated images created by merging real images of the Earth with renderings of the Tango spacecraft at different positions and orientations. A small set of 5 *real* images is also provided with hand-made annotations of the pose, created by taking a photo of a replica of the satellite in a studio. The test set is divided in 2998 synthetic images generated similarly to the training ones, and a set of 300 real images. The dataset provides the annotations on the pose of the satellite: the satellite's

position is described by a 3D vector, while the orientation of the satellite is described as a quaternion. Moreover, the satellite provides also the characteristics of the camera used to obtain the images.

For each image of this dataset, we add an alpha channel representing the binary segmentation of the satellite. To obtain the segmentation on the train set, we use the model estimated in the reconstruction step (Section 3.1), render it at the position and orientation provided by the ground truth annotations, and use the rendered silhouette as a segmentation mask. Instead, to obtain the segmentation of the image of the test set, we use the Mask R-CNN [10] segmentation network with ResNet-101 [11] and Feature Pyramid Networks [15] as a backbone, pre-trained of COCO [16] 2017, and implemented in the Detectron2 framework [3]. We fine-tune this network using a dataset obtained by combining the train set augmented with the alpha channel annotations just discussed, a set of 5950 images created by segmenting the satellite from the previous set and performing additional data augmentation operations (addition of random images of the Earth as background, random rotation and resize of Tango, and darkening of the image), and a set of 100000 synthetic images rendered using the 3D model we estimated in Section 3.1.

4.2 Results

In this section, we report the result obtained by this pipeline in the ESA pose estimation challenge [2] just introduced. We do not provide any quantitative evaluation of the 3D model reconstruction as the true model of the Tango satellite is not available. A qualitative evaluation of the recontruction is given by the rendering reported in Figure 4, and the effectiveness of the pose refinement can be considered as an indirect assessment of its quality too.

In Table 1, we report the quantitative results for the whole pipeline, before and after the pose refinement step. In particular, we have evaluated our model with different methods for representing 3D rotations. In the *6D vector* case the model shown in Section 3.2 regresses the 6D rotation vector and the pose refinement optimization is performed on the elements of the 6D vector prior to transform them into a rotation matrix. In the *Quaternion* case the regression model output is a 4D unconstrained quaterion orientation, and the pose refinement optimization is performed on such a 4D vector, from which we obtain a real quaternion trough a normalization prior to convert it into the rotation matrix R. In the *Euler Angles* case we regress a 3D vector of Euler angles and the optimization is performed directly on these 3 Euler angles, which are applied in the order $Z \rightarrow X \rightarrow Y$ to obtain the final rotation matrix R. We have also tested a *Classification approach* for the angle regression by discretizing each Euler angle into 18 classes. Each class is converted to its corresponding Euler angle prior to optimize them directly in the pose refinement step. These are then applied in the $Z \rightarrow X \rightarrow Y$ order to obtain the final rotation matrix R. As we can see from the Table 1 the process of pose refinement using differentiable rendering improves the score in all situations. The 6D vector representations is the one performing the best, since, being continuous, it is easier to regress by a neural network,

as also reported in [38]. The Euler angles representation and its classification simplification are those the worse as they both suffer from gimbal lock.

We also notice that the quality of the final estimate strongly depends on the quality of the starting estimate. To verify to what extent this improvement is possible we evaluated the effectiveness of pose refinement using the pose estimate obtained by one of the best competitors in the ESA's challenge, i.e, UrsoNet [25]. *UrsoNet* is a neural network that competed for the first tranche of the Pose Estimation Challenge, achieving third position. The authors of UrsoNet made their neural network architecture and the weights used for the competition available for the public, so we can perform a direct comparison with their approach. Their neural network is composed by a pre-trained feature extractor followed by two branches: the localization branch that directly regresses the translation vector T, and a rotation branch that obtains the quaternion q via probabilistic fitting. We used this neural network with the provided pre-trained weights, scored it on the challenge website and use this result as a baseline. Then, we applied our proposed pose refinement step to UrsoNet predictions. The order of Euler angles follows the convention $Z \rightarrow X \rightarrow Y$ to obtain the final rotation matrix R. The results on UrsoNet are reported both for the test set of rendered images and for the test set of real images. We can do this only for this scenario: ESA's scoring page only shows the score of the rendered test set and not of the real test set. This score can be only obtained from the leaderboard (showing the best score both on the real images and on the rendered images). Once again, the proposed pose refinement approach is able to improve the results, proving that it can be used to enhance any pose regression estimation, even of high quality.

5 Conclusions

In this work we presented a new method to address the 6 degrees of freedom pose estimation using differentiable rendering. With this method, first we reconstruct the 3D model of an object with a differentiable rendering technique, then we use this information to enrich our dataset with new images and useful annotations, and regress a first estimation of the six degrees of freedom. Finally, we refine this coarse pose with a render-and-compare approach using differentiable rendering.

The use of differentiable rendering in the pose refinement scenario proved itself to be very solid and consistent, enhancing the prediction in each different scenario we tested it in, even enhancing the score of one of the best competitors of the first tranche of the challenge held by ESA [2]. As for the reduction in data requirement the method we propose has been able to reconstruct the 3D model of the object to localize from few images, nevertheless the pose regression part has required data augmentation via rendering.

To improve the perfomance of the pose regressor we are currently working on a end-to-end architecture for training which uses the differentiable renderer to compute an additional unsupervised loss beside the supervised one currently used.

References

1. Active debris removal: Recent progress and current trends. Acta Astronautica **85**, 51–60 (2013). https://doi.org/10.1016/j.actaastro.2012.11.009
2. Pose estimation challenge 2019 post mortem (2020), https://kelvins.esa.int/pose-estimation-challenge-post-mortem/
3. Detectron2: A pytorch-based modular object detection library (2021), https://github.com/facebookresearch/detectron2
4. Pose estimation challenge 2021 (2022), https://kelvins.esa.int/pose-estimation-2021/
5. Bauer, F., Hartman, K., How, J., Bristow, J., Weidow, D., Busse, F.: Enabling spacecraft formation flying through spaceborne gps and enhanced automation technologies (12 2002)
6. Chen, W., Gao, J., Ling, H., Smith, E.J., Lehtinen, J., Jacobson, A., Fidler, S.: Learning to predict 3d objects with an interpolation-based differentiable renderer (2019)
7. Cheng, A.F.: Near Earth Asteroid Rendezvous: Mission Summary, pp. 351–366 (2002)
8. Clerc, X., Retat, I.: Astrium vision on space debris removal. In: Proceeding of the 63rd International Astronautical Congress (IAC 2012), Napoli, Italy. vol. 15 (2012)
9. Flores-Abad, A., Ma, O., Pham, K., Ulrich, S.: A review of space robotics technologies for on-orbit servicing. Progress in Aerospace Sciences **68**, 1–26 (2014). https://doi.org/https://doi.org/10.1016/j.paerosci.2014.03.002, https://www.sciencedirect.com/science/article/pii/S0376042114000347
10. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015), http://arxiv.org/abs/1512.03385
12. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer (2017)
13. Kendall, A., Grimes, M., Cipolla, R.: Convolutional networks for real-time 6-dof camera relocalization. CoRR **abs/1505.07427** (2015), http://arxiv.org/abs/1505.07427
14. Kubota, T., Sawai, S., Hashimoto, T., Kawaguchi, J.: Robotics and autonomous technology for asteroid sample return mission. In: ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005. pp. 31–38 (2005). https://doi.org/10.1109/ICAR.2005.1507387
15. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
16. Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. CoRR **abs/1405.0312** (2014), http://arxiv.org/abs/1405.0312
17. Liu, S., Li, T., Chen, W., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning (2019)
18. Loper, M.M., Black, M.J.: Opendr: An approximate differentiable renderer pp. 154–169 (2014)
19. Mahendran, S., Ali, H., Vidal, R.: 3d pose regression using convolutional neural networks. CoRR **abs/1708.05628** (2017), http://arxiv.org/abs/1708.05628
20. Massa, F., Aubry, M., Marlet, R.: Convolutional neural networks for joint object detection and pose estimation: A comparative study. CoRR **abs/1412.7190** (2014), http://arxiv.org/abs/1412.7190

21. Opromolla, R., Fasano, G., Rufino, G., Grassi, M.: A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. Progress in Aerospace Sciences **93**, 53–72 (2017). https://doi.org/10.1016/j.paerosci.2017.07.001
22. Palazzi, A., Bergamini, L., Calderara, S., Cucchiara, R.: End-to-End 6-DoF Object Pose Estimation Through Differentiable Rasterization: Munich, Germany, September 8-14, 2018, Proceedings, Part III, pp. 702–715 (01 2019). https://doi.org/10.1007/978-3-030-11015-4_53
23. Park, K., Mousavian, A., Xiang, Y., Fox, D.: Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. CoRR **abs/1912.00416** (2019), http://arxiv.org/abs/1912.00416
24. Pavlakos, G., Zhou, X., Chan, A., Derpanis, K.G., Daniilidis, K.: 6-dof object pose from semantic keypoints. CoRR **abs/1703.04670** (2017), http://arxiv.org/abs/1703.04670
25. Proenca, P.F., Gao, Y.: Deep learning for spacecraft pose estimation from photo-realistic rendering (2019)
26. Rad, M., Lepetit, V.: BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. CoRR **abs/1703.10896** (2017), http://arxiv.org/abs/1703.10896
27. Rahman, M.A., Wang, Y.: Optimizing intersection-over-union in deep neural networks for image segmentation. In: ISVC (2016)
28. Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.Y., Johnson, J., Gkioxari, G.: Accelerating 3d deep learning with pytorch3d. arXiv:2007.08501 (2020)
29. Rhodin, H., Robertini, N., Richardt, C., Seidel, H.P., Theobalt, C.: A versatile scene model with differentiable visibility applied to generative pose estimation (2016)
30. Sahin, C., Garcia-Hernando, G., Sock, J., Kim, T.K.: A review on object pose recovery: from 3d bounding box detectors to full 6d pose estimators (2020)
31. Sharma, S., D'Amico, S.: Pose estimation for non-cooperative rendezvous using neural networks. arXiv preprint arXiv:1906.09868 (2019)
32. Su, H., Qi, C.R., Li, Y., Guibas, L.J.: Render for CNN: viewpoint estimation in images using cnns trained with rendered 3d model views. CoRR **abs/1505.05641** (2015), http://arxiv.org/abs/1505.05641
33. Tatsch, A., Fitz-Coy, N., Gladun, S.: On-orbit servicing: A brief survey. In: Proceedings of the IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR'06). pp. 276–281 (2006)
34. Tulsiani, S., Malik, J.: Viewpoints and keypoints. CoRR **abs/1411.6067** (2014), http://arxiv.org/abs/1411.6067
35. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. CoRR **abs/1711.00199** (2017), http://arxiv.org/abs/1711.00199
36. Xu, Y., Zhu, S.C., Tung, T.: Denserac: Joint 3d pose and shape estimation by dense render-and-compare (2019)
37. Yang, Z., Yu, X., Yang, Y.: Dsc-posenet: Learning 6dof object pose estimation via dual-scale consistency (04 2021)
38. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. CoRR **abs/1812.07035** (2018), http://arxiv.org/abs/1812.07035