Robust Probabilistic Robot Arm Keypoint Detection Exploiting Kinematic Knowledge

Lukas Meyer[1*], Leonard Klüpfel[1*], Maximilian Durner[1], and Rudolph Triebel[1,2]

Abstract— **We propose PK-ROKED, a novel probabilistic deep-learning algorithm to detect keypoints of a robotic manipulator in camera images and to robustly estimate the positioning inaccuracies w.r.t the camera frame. Our algorithm uses monocular images as a primary input source and augments these with prior knowledge about the keypoint locations based on the robot's forward kinematics. As output, the network provides 2D image coordinates of the keypoints and an associated uncertainty measure, where the latter is obtained using Monte Carlo dropout. In experiments on two different robotic systems, we show that our network provides superior detection results compared to the state-of-the-art. We furthermore analyze the precision of different estimation approaches to obtain an uncertainty measure.**

I. Introduction

Robotic manipulation systems can suffer from an imprecise estimate of the robot's forward kinematics. For robots with a fixed camera observing the moving arm (so-called eye-to-hand systems), vision algorithms can be used for correction. In this work, we propose an algorithm that detects the position of the arm from the images and corrects the position estimated from the forward kinematics. More precisely, our algorithm provides distributions of 2D image locations for a set of predefined keypoints of the robotic arm. These are then fused with the forward kinematics using Bayesian filtering. The fusion step of uncertain robot kinematics with 2D image information is already discussed in [2].

In our present contribution, we focus on how to reliably detect probabilistic 2D information of robotic arms in images. For this task, we define fixed *keypoints* on a robot arm and train a deep learning (DL) network to detect these keypoints in monocular RGB images. In addition, using Monte Carlo dropout, our network is able to provide uncertainty estimates on the detected keypoints to enable the probabilistic downstream fusion task. The network output is visualized in Fig. 1.

The forward kinematics can be assumed as an erroneous sensor, however with bounded error. We use this fact to aid our network by providing it with prior kinematic knowledge (PK) as additional input channels, i. e., heatmaps on areas in the image that potentially contain the respective keypoints, resulting in our *Prior Knowledge - RObot KEypoint Detection (PK-ROKED)* network.

We train our network using synthetic data and show the performance on real world data sets of two different robots,

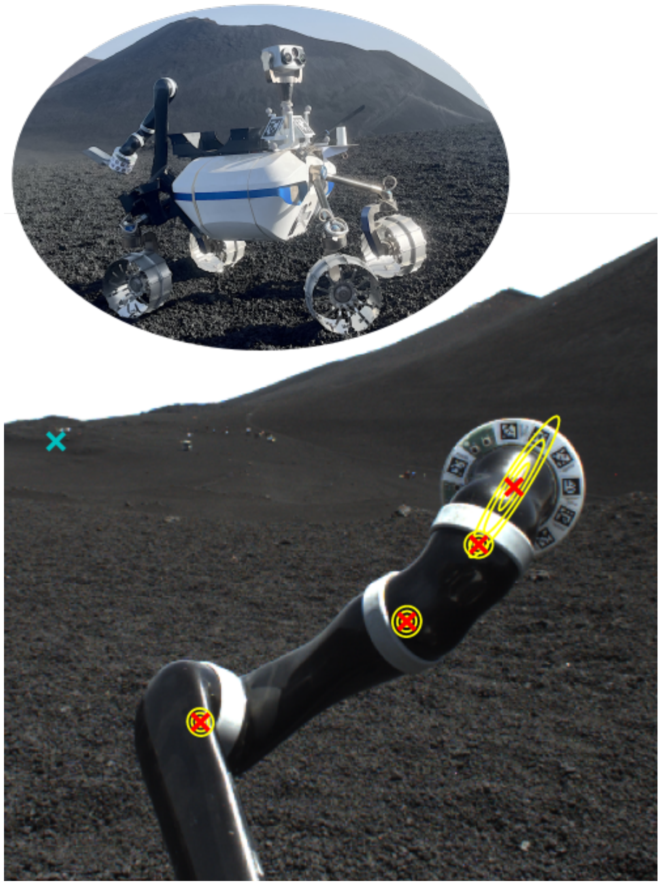
Fig. 1: top – The Lightweight Rover Unit (LRU) on Mt. Etna during the ARCHES field test [1]. bottom – The middle camera on the robot's head records the movement of the Jaco2 arm. Detected keypoints (red), the corresponding uncertainty ellipses (yellow), and detected false positives (petrol) in the background are shown.

once the *Panda* arm using the data sets of [3] and once the *Jaco2* arm of our own Lightweight Rover Unit (LRU) – see Fig. 1. For comparison, we use the DREAM network from [3] as baseline and show that PK-ROKED outperforms DREAM in all scenarios.

II. Related Work

Several works use DL methods on monocular images to detect robotic arms. The previously mentioned DREAM network [3] considers the joints of a robot arm as keypoints and extracts their 2D image coordinates from heatmaps. The work of [4] detects 2D robot keypoints as well, but additionally considers optimization steps to select an optimal subset of a group of potential keypoints. Lambrecht *et*

*Equal contribution to this work

[1]Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82234 Wessling, Germany firstname.lastname@dlr.de [2]School of Engineering and Design, Technical University of Munich (TUM), 85521 Ottobrunn, Germany

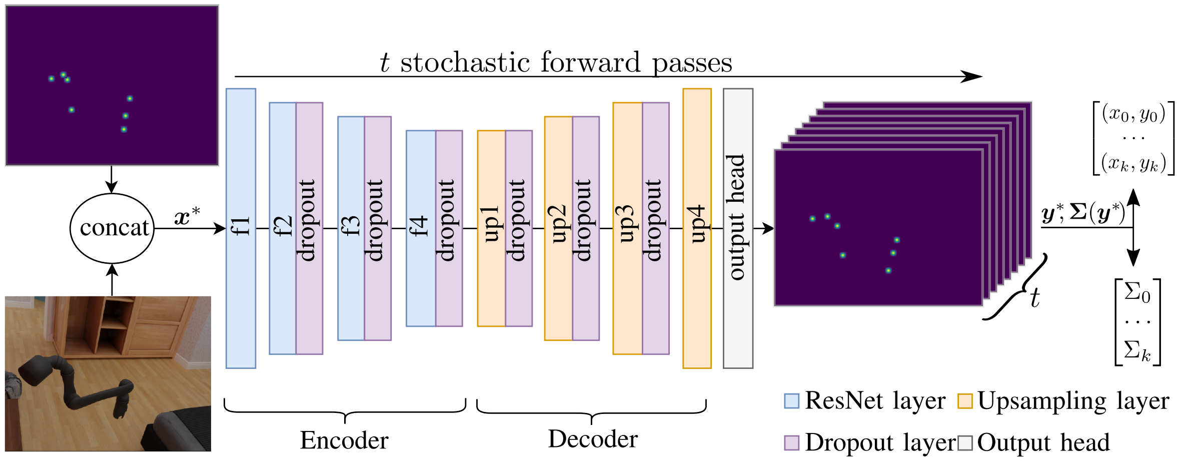

Fig. 2: The network architecture. The resulting t heatmaps per joint are stacked and constitute the belief map for uncertainty computation. Note that for visualization purposes, the heatmaps of the individual joints are combined into one image.

al. [5] compute bounding boxes around the joints of a robotic arm using image sequences. However, none of the mentioned works consider prior kinematic knowledge for robot arm detection and neither provide uncertainty estimates that accompany their result.

III. Network Architecture and Prior Kinematic Knowledge

PK-ROKED uses an hourglass network architecture to learn the keypoint representation of a robotic arm. In our case, we consider the robot joint locations as keypoints and use heatmaps of predicted keypoint locations with values roughly between 0 and 1 as network output, similar to [3].

Our network architecture is shown in Fig. 2. The network input consists of RGB images of a robot arm stacked together with the heatmaps of k joints resulting in an input with the dimension of $640 \times 480 \times (3 + k)$. The heatmaps are obtained by projecting the 3D location of the keypoints unto the camera image. These heatmaps are set to one at the keypoint pixel location and zero otherwise, afterwards a Gaussian smoothing is applied. For training, the heatmaps use the ground truth keypoint information after an additional perturbation. For inference, the heatmaps are directly obtained from the (erroneous) robot kinematics.

The encoder downsamples the input to a $40 \times 30 \times 2048$ bottleneck. We base our encoder on the first four layers of a ResNet50 [6] network, pre-trained on ImageNet.

Our decoder upsamples the bottle neck representation to a dimension of $640 \times 480 \times k$, resulting in k stacked heatmaps of the predicted keypoint locations. The final output is the keypoint locations in image coordinates, obtained by finding the location of the highest valued pixel in the output heatmaps.

The loss function is the mean squared error (MSE) measuring the Euclidean distance between a ground truth heat map and the prediction of the network.

IV. Uncertainty Computation

To use the results of PK-ROKED for fusion tasks, it is of key importance to correctly capture the uncertainty of the measurements. For this, we evaluate several approaches to capture both the model uncertainty (epistemic uncertainty) and the data uncertainty (aleatoric uncertainty). As the results are to be used with sensor fusion algorithms, mainly an extended Kalman filter (EKF), we focus on the first and second moment of the probability distribution, i. e., the mean and the covariance.

A. Monte Carlo Dropout

We estimate our model uncertainty using Monte Carlo dropout [7]. We place dropout layers at the mid-layers of the hourglass, as suggested by [8] (visualized in Fig. 2). In these dropout layers, the individual network weights are randomly set to zero based on a previously defined Bernoulli probability distribution, a technique that was initially developed as regularization method during training [9]. Dropouts however, can also be used during inference to sample posterior distributions using multiple forward passes of the same input $\boldsymbol{x}^*$. Thus, for every forward pass different entries of the weights $\hat{\boldsymbol{W}}$ are randomly set to zero, which is referred to as Monte Carlo dropout [7]. In our implementation, the dropout probability is set to $p = 0.1$. Our internal analysis shows that sampling the network output with $t = 20$ stochastic forward passes is a good trade-off between run-time efficiency and probability approximation.

The resulting keypoint locations are the mean of the image coordinates $\boldsymbol{y}_i^* = f^{\hat{\boldsymbol{W}}_i}(\boldsymbol{x}^*)$ over all t forward passes:

$$\mathrm{E}(\boldsymbol{y}^*) = \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{y}_i^*. \tag{1}$$

B. Obtaining the Covariance Matrix

In this work, we analyze several different approaches to use Monte Carlo dropout to obtain a covariance matrix that correctly represents the uncertainty of the network prediction. These methods are experimentally evaluated in Section VII-B. The methods are:

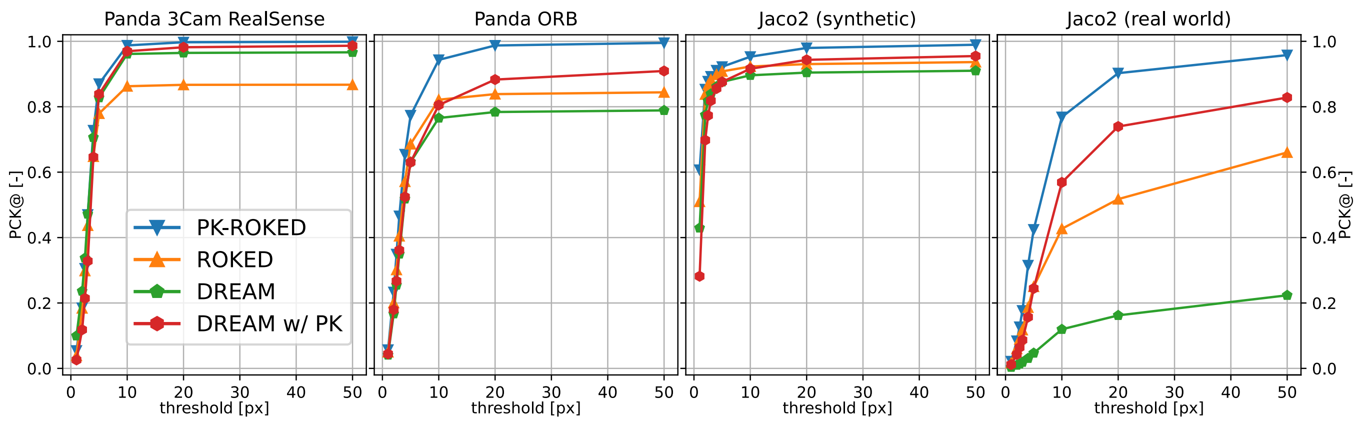

Fig. 3: Accuracy of the keypoint detection on one synthetic and three real world evaluation data sets.

1) Explicit Uncertainty Computation: According to [7], the covariance matrix $\mathbf{\Sigma}$ can be approximated as

$$\mathbf{\Sigma}(\boldsymbol{y}^*) \approx \tau^{-1}\boldsymbol{I}_D + \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{y}_i^{*T}\boldsymbol{y}_i^* - \mathrm{E}(\boldsymbol{y}^*)^T\mathrm{E}(\boldsymbol{y}^*), \qquad (2)$$

where τ^{-1} denotes the observation noise and $\boldsymbol{I}_D$ is the identity of dimension D. With the hyperparameter τ, we can encode the aleatoric uncertainty, however with the assumption that it is constant for all input data, i. e., homoscedastic aleatoric uncertainty.

Another approach is to learn the aleatoric uncertainty, following [10]. This assumes heteroscedastic aleatoric uncertainty – a change of noise in the input data. We modify our standard network with an additional output head, which predicts the aleatoric uncertainty per pixel as described by [10]. The term with τ in Eq. (2) is replaced by averaging the pixel values of the second output head around each keypoint location for all t forward passes.

2) Implicit Uncertainty Computation from Belief Maps: The heatmap of the predicted keypoint locations can also be viewed as a belief map approximating the probability of keypoint locations. This interpretation considers the heatmap of a single forward pass as a representation of the aleatoric uncertainty. By accumulating the results of all t forward passes into one heatmap, one can assume both aleatoric and epistemic uncertainty to be combined in a single belief map. The covariance (the second moment of the probability distribution) is therefore analog to the second order central image moment.

We transform the belief map into a binary image with a threshold parameter of 0.6 and directly compute the covariance matrix from the central second order image moments [11]:

$$\mathbf{\Sigma}(\boldsymbol{y}^*) \approx \begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix} \qquad (3)$$

with

$$\mu_{ij} = \sum_{u,v}(u-\bar{u})^i(v-\bar{v})^j I(u,v). \qquad (4)$$

There, u and v denote the horizontal and vertical pixel coordinates, $\bar{u}$ and $\bar{v}$ the image's centroid, and I is the pixel intensity value.

V. Training and Evaluation Data

We train and evaluate the network on two different robotic arms. In both cases, the network is trained on synthetic data and the results are evaluated on real world data.

The first robotic arm is Franka Emika's *Panda* and the corresponding data is provided by [3]. It is comprised of the synthetic training data and also consists of two different real world evaluation data sets, which we refer to as *Panda 3Cam RealSense* and *Panda ORB*.

The other data sets are specifically generated by us for the usage with the LRU rover. We render the *Jaco2* arm and the attached docking interface from its URDF representation in the tool BlenderProc [12] and apply several domain randomization techniques, such as different background scenes, varied robot and camera poses, and changing textures. During training, the images are subjected to further randomized augmentations, e. g., Gaussian noise, coarse dropout, image or channel flipping, and brightness contrast.

The real world evaluation data set is created with the LRU rover, by commanding different joint configurations and recording the arm movement using the system's head cameras; all in an indoor environment. The ground truth is obtained using a *Vicon* tracking system to track markers that are attached to the vicinity of the end-effector via the docking interface and are calibrated using a hand-eye calibration approach. Note that there is a non-observable residual calibration error on the tracking orientation, thus we consider only the ground truth of the last 3 joints for our performance evaluation.

VI. Accuracy Evaluation

We compare the accuracy of four different network architectures:

- Our architecture PK-ROKED with kinematic priors.
- Our architecture ROKED, that is PK-ROKED without using PK as additional input channels.
- The DREAM architecture [3] serving as baseline.
- We augment DREAM to consider heatmaps with kinematic priors as additional input: DREAM w/ PK.

A. *Accuracy Metric – Percentage of Correct Keypoints*

The percentage of correct keypoints (PCK) measures the accuracy of the network output over the whole evaluation data set. The $PCK@c$ is the percentage of all correctly detected keypoints within a threshold radius of c pixels around the ground truth keypoint locations.

B. *Network Accuracy Performance*

Figure 3 shows the accuracy results for the four network architectures on different data sets.

First, we evaluate the networks on the two real world Panda data sets provided by [3]. For both data sets, PK-ROKED outperforms the other algorithms, even though all achieve high accuracy rates. The Panda ORB appears to be the more challenging data set and algorithms that consider PK can handle this challenge more accurately. Interestingly, for the Panda 3Cam RealSense data set, we can reproduce the accuracy results of DREAM as reported by [3] but achieve a slightly inferior result on the Panda Orb data set.

For the synthetic Jaco2 data, all networks perform with a high accuracy and PK-ROKED achieves the best results. For the real world Jaco2 data, the performance decreases significantly compared to all other data sets. We attribute this to a strong sim-to-real gap due to reflections on the arm and clutter in the image background. However, the relative performance remains similar: Networks that consider prior kinematic knowledge outperform the other architectures and PK-ROKED provides the overall best accuracy.

In the end, two primary observations regarding the network accuracy can be made: First, the results underline our motivation to incorporate kinematic priors into robot keypoint detection algorithms and confirm the performance improvements. Second, our ROKED architecture (both with and without prior knowledge) can be considered superior to the DREAM network. Only with the Panda 3Cam RealSense data, DREAM performs better than ROKED, which we attribute to a small sim-to-real gap in the data.

VII. Uncertainty Evaluation

A. *Uncertainty Representation – Precision*

This metric allows us to evaluate the quality of uncertainty estimation. The *precision* is defined as

$$precision@s = \frac{TP}{TP + FP}, \tag{5}$$

with TP and FP denoting the number of true and false positive keypoint detections. A TP is a keypoint result that lies within a boundary defined by the covariance matrix, a FP lies outside the boundary. The precision is evaluated for boundaries that are provided as several multiples s of the covariance matrix' two-dimensional standard deviation $\boldsymbol{\sigma}$.

B. *Network Uncertainty Precision*

Figure 4 shows the precision obtained by PK-ROKED for the different covariance computation methods described in Section IV.. As reference, the corresponding curve defined for an ideal normal distribution is plotted. This means, that e. g., at a sigma multiplier $s = 1$, 68% of data points are within that boundary.

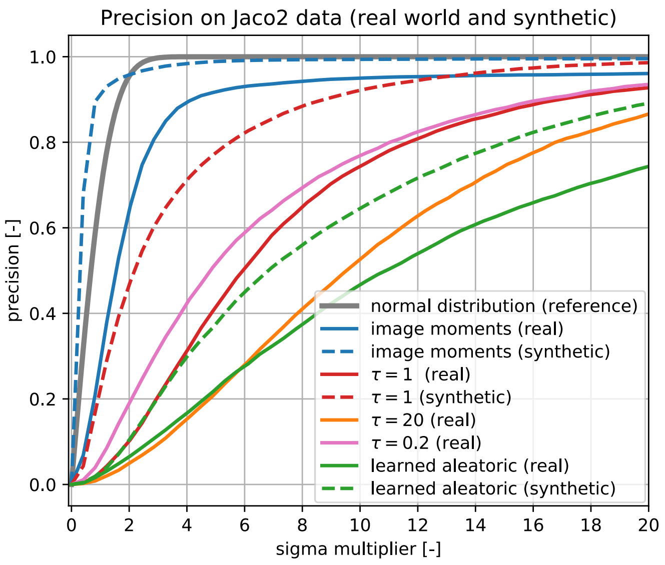

Fig. 4: Precision results for different uncertainty computation methods for the real world Jaco2 data (solid) and the corresponding synthetic data (dashed).

We evaluate the precision results of our three uncertainty computation approaches on synthetic and real data. For both data sets, it stands out that all explicit covariance computation methods fail to correctly encapsulate the uncertainty of the keypoint detection. The covariance matrices obtained by these methods are usually very overconfident.

On the other hand, obtaining the covariance matrix by the image moments on the output belief maps provides a precision very close to the reference normal distribution. We therefore argue that the stacked output belief maps over $t = 20$ forward passes are a realistic representation of the true network uncertainty. Subsequently, using the image moments to obtain a covariance matrix is therefore the most promising approach for this algorithm in combination with a downstream fusion task.

VIII. Conclusion

With PK-ROKED, we present a DL network that allows to robustly detect keypoints on robotic arms. It does not only outperform the state-of-the-art but additionally provides a realistic uncertainty estimate for the detected keypoints, such that the network output can be used in downstream sensor fusion approaches. We elaborate that using Monte Carlo dropout to create belief maps and subsequently obtain the estimation uncertainty via image moments is a reliable approach to obtain uncertainty estimates. We furthermore show that adding the available knowledge of the robot kinematics as input to the keypoint detection network significantly increases the accuracy of both evaluated networks. We therefore argue that the usage of robotic kinematic knowledge as additional network input should be considered whenever possible.

References

[1] M. J. Schuster, M. G. Müller, S. G. Brunner, H. Lehner, P. Lehner, R. Sakagami, A. Dömel, L. Meyer, B. Vodermayer, R. Giubilato, M. Vayugundla, J. Reill, F. Steidle, I. von Bargen, K. Bussmann, R. Belder, P. Lutz, W. Stürzl, M. Smíšek, M. Maier, S. Stoneman, A. F.

Prince, B. Rebele, M. Durner, E. Staudinger, S. Zhang, R. Pöhlmann, E. Bischoff, C. Braun, S. Schröder, E. Dietz, S. Frohmann, A. Börner, H. Hübers, B. Foing, R. Triebel, A. O. Albu-Schäffer, and A. Wedler, “The ARCHES space-analogue demonstration mission: Towards heterogeneous teams of autonomous robots for collaborative scientific sampling in planetary exploration,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5315–5322, 2020.

[2] L. Meyer, K. H. Strobl, and R. Triebel, “The Probabilistic Robot Kinematics Model and its Application to Sensor Fusion,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[3] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield, “Camera-to-robot pose estimation from a single image,” in *2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 9426–9432.

[4] J. Lu, F. Richter, and M. C. Yip, “Pose Estimation for Robot Manipulators via Keypoint Optimization and Sim-to-Real Transfer,” in *IEEE Robotics and Automation Letters*, vol. 7, Apr. 2022, pp. 4622–4629.

[5] J. Lambrecht, “Robust few-shot pose estimation of articulated robots using monocular cameras and deep-learning-based keypoint detection,” in *International Conference on Robot Intelligence Technology and Applications*, 2019, pp. 136–141.

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778.

[7] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,” in *International Conference on Machine Learning*. PMLR, Jun. 2016, pp. 1050–1059.

[8] A. Kendall, V. Badrinarayanan, and R. Cipolla, “Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding,” in *British Machine Vision Conference*, vol. 57, Oct. 2016, pp. 1–12.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[10] A. Kendall and Y. Gal, “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[11] J. Flusser, B. Zitova, and T. Suk, *Moments and Moment Invariants in Pattern Recognition*. Wiley, 2009.

[12] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam, “BlenderProc,” *arXiv*, 2019. [Online]. Available: http://arxiv.org/abs/1911.01911