Camera-to-Robot Pose Estimation from a Single Image

Timothy E. Lee[1,2], **Jonathan Tremblay**[1], **Thang To**[1], **Jia Cheng**[1],
Terry Mosier[1], **Oliver Kroemer**[2], **Dieter Fox**[1], and **Stan Birchfield**[1],
[1]NVIDIA: {jtremblay, thangt, jicheng, tmosier, dieterf, sbirchfield}@nvidia.com
[2]Carnegie Mellon University, The Robotics Institute: {timothyelee, okroemer}@cmu.edu

***Abstract*— We present an approach for estimating the pose of an external camera with respect to a robot using a single RGB image of the robot. The image is processed by a deep neural network to detect 2D projections of keypoints (such as joints) associated with the robot. The network is trained entirely on simulated data using domain randomization to bridge the reality gap. Perspective-n-point (PnP) is then used to recover the camera extrinsics, assuming that the camera intrinsics and joint configuration of the robot manipulator are known. Unlike classic hand-eye calibration systems, our method does not require an off-line calibration step. Rather, it is capable of computing the camera extrinsics from a single frame, thus opening the possibility of on-line calibration. We show experimental results for three different robots and camera sensors, demonstrating that our approach is able to achieve accuracy with a single frame that is comparable to that of classic off-line hand-eye calibration using multiple frames. With additional frames from a static pose, accuracy improves even further. Code, datasets, and pretrained models for three widely-used robot manipulators are made available.**[1]

I. INTRODUCTION

Determining the pose of an externally mounted camera is a fundamental problem for robot manipulation, because it is necessary to transform measurements made in camera space to the robot's task space. For robots operating in unstructured, dynamic environments—performing tasks such as object grasping and manipulation, human-robot interaction, and collision detection and avoidance—such a transformation allows visual observations to be readily used for control.

The classic approach to calibrating an externally mounted camera is to fix a fiducial marker (*e.g.*, ArUco [1], ARTag [2], or AprilTag [3]) to the end effector, collect several images, then solve a homogeneous linear system for the unknown transformation [4]. This approach is still widely used due to its generality, flexibility, and the availability of open-source implementations in the Robot Operating System (ROS). However, such an approach requires the somewhat cumbersome procedure of physically modifying the end effector, moving the robot to multiple joint configurations to collect a set of images (assuming the marker-to-wrist transformation is not known), running an off-line calibration procedure, and (optionally) removing the fiducial marker. Such an approach is not amenable to online calibration, because if the camera moves with respect to the robot, the entire calibration procedure must be repeated from scratch.

A more recent alternative is to avoid directly computing the camera-to-robot transform altogether, and instead to rely on an implicit mapping that is learned for the task at hand. For example, Tobin *et al.* [5] use deep learning to map RGB images to world coordinates on a table, assuming that the table at test time has the same dimensions as the one used during training. Similarly, Levine *et al.* [6] learn hand-eye coordination for grasping a door handle, using a large-scale setup of real robots for collecting training data. In these approaches the learned mapping is implicit and specific to the task/environment, thus preventing the mapping from being applied to new tasks or environments without retraining.

We believe there is a need for a general-purpose tool that performs online camera-to-robot calibration without markers. With such a tool, a researcher could set up a camera (*e.g.*, on a tripod), and then immediately use object detections or measurements from image space for real-world robot control in a task-independent manner, without a separate offline calibration step. Moreover, if the camera subsequently moved for some reason (*e.g.*, the tripod were bumped accidentally), there would be no need to redo the calibration step, because the online calibration process would automatically handle such disturbances.

In this paper, we take a step in this direction by presenting a system for solving camera pose estimation from a single RGB image. We refer to our framework as *DREAM* (for Deep Robot-to-camera Extrinsics for Articulated Manipulators). We train a robot-specific deep neural network to estimate the 2D projections of prespecified keypoints (such as the robot's joints) in the image. Combined with camera intrinsics, the robot joint configuration, and forward kinematics, the camera extrinsics are then estimated using Perspective-n-Point (PnP) [7]. The network is trained entirely on synthetic images, relying on domain randomization [5] to bridge the reality gap. To generate these images, we augmented our open-source tool, NDDS [8], to allow for scripting robotic joint controls and to export metadata about specific 3D locations on a 3D mesh.

This paper makes the following contributions:

- We demonstrate the feasibility of computing the camera-to-robot transformation from a single image of the robot, *without fiducial markers*, using a deep neural

[1]Work was completed while the first author was an intern at NVIDIA.
[1]https://research.nvidia.com/publication/2020-03_DREAM

network trained only on synthetic data.
- We show that the resulting accuracy with a single real image frame is comparable to that of classic hand-eye calibration using multiple frames. For a static camera, accuracy further improves with multiple image frames.
- Quantitative and qualitative results are shown for three different robot manipulators (Franka Emika's Panda, Kuka's LBR iiwa 7 R800, and Rethink Robotics' Baxter) and a variety of cameras.

The simplicity of our approach enables real-time pose estimation on a desktop computer with an NVIDIA Titan Xp GPU, without manual optimization.

II. APPROACH

An externally mounted camera observes n keypoints $\mathbf{p}_i \in \mathbb{R}^3$ on various robot links. These keypoints project onto the image as $\mathbf{k}_i \in \mathbb{R}^2$, $i = 1, \ldots, n$. Some of these projections may be inside the camera frustum, whereas others may be outside. We consider the latter to be invisible/inaccessible, whereas the former are visible, regardless of occlusion.[2] The intrinsic parameters of the camera are assumed known.

Our proposed two-stage process for solving the problem of camera-to-robot pose estimation from a single RGB image frame is illustrated in Fig. 1. First, an encoder-decoder neural network processes the image to produce a set of n belief maps, one per keypoint. Then, Perspective-n-Point (PnP) [7] uses the peaks of these 2D belief maps, along with the forward kinematics of the robot and the camera intrinsics, to compute the camera-to-robot pose, ${}^{R}_{C}T$. Note that the network training depends only on the images, not the camera; therefore, after training, the system can be applied to any camera with known intrinsics. We restrict $n \geq 4$ for stable PnP results.

A. Network Architecture

Inspired by recent work on object pose estimation [9], [10], [11], we use an auto-encoder network to detect the keypoints. The neural network takes as input an RGB image of size $w \times h \times 3$, and it outputs an $\alpha w \times \alpha h \times n$ tensor, where $w = 640$, $h = 480$, and $\alpha \in \{1, \frac{1}{2}, \frac{1}{4}\}$, depending on the output resolution used. This output captures a 2D belief map for each keypoint, where pixel values represent the likelihood that the keypoint is projected onto that pixel.

The encoder consists of the convolutional layers of VGG-19 [12] pretrained on ImageNet. The decoder (upsampling) component is composed of four 2D transpose convolutional layers (stride = 2, padding = 1, output padding = 1), and each layer is followed by a normal 3×3 convolutional layer and ReLU activation layer. We also experimented with a ResNet-based encoder, *viz.*, our reimplementation of [13], with the same batch normalization, upsampling layers, and so forth as described in the paper.

The output head is composed of 3 convolutional layers ($3\times$ 3, stride = 1, padding = 1) with ReLU activations with 64, 32, and n channels, respectively. There is no activation layer after the final convolutional layer. The network is trained using an L2 loss function comparing the output belief maps with ground truth belief maps, where the latter are generated using $\sigma = 2$ pixels to smooth the peaks.

B. Pose Estimation

Given the 2D keypoint coordinates, robot joint configuration with forward kinematics, and camera intrinsics, PnP [7] is used to retrieve the pose of the robot, similar to [14], [15], [9], [10], [11]. The keypoint coordinates are calculated as a weighted average of values near thresholded peaks in the output belief maps (threshold = 0.03), after first applying Gaussian smoothing to the belief maps to reduce the effects of noise. The weighted average allows for subpixel precision.

C. Data Generation

The network is trained using only synthetic data with domain randomization (DR). Despite the traditional challenges with bridging the reality gap, we find that our network generalizes well to real-world images, as we will show in the experimental results. To generate the data we used our open-source NDDS tool [8], which is a plugin for the UE4 game engine. We augmented NDDS to export 3D/2D keypoint locations and robot joint angles, as well as to allow control of the robot joints.

The synthetic robot was placed in a simple virtual 3D scene in UE4, viewed by a virtual camera. Various randomizations were applied: 1) The robot's joint angles were randomized within the joint limits. 2) The camera was positioned freely in a somewhat truncated hemispherical shell around the robot, with azimuth ranging from $-135°$ to $+135°$ (excluding the back of the robot), elevation from $-10°$ to $75°$, and distance from 75 cm to 120 cm; the optical axis was also randomized within a small cone. 3) Three scene lights were positioned and oriented freely while randomizing both intensity and color. 4) The scene background was randomly selected from the COCO dataset [16]. 5) 3D objects from the YCB dataset [17] were randomly placed, similar to flying distractors [18]. 6) Random color was applied to the robot mesh.

Figure 2 shows representative images from synthetic datasets generated by our approach. Domain randomized (DR) datasets were used for training and testing; datasets without domain randomization (non-DR) were used for testing to assess sim-to-sim generalization. The DR training data was generated using the publicly available robot models, whereas the non-DR test data used models that were artistically improved to be more photorealistic.

III. EXPERIMENTAL RESULTS

In this section we evaluate our DREAM method both in simulation and in the real world. We seek to answer the following questions: 1) How well does the synthetic-only training paradigm transfer to real-world data? 2) What is the impact of various network architectures? 3) What accuracy can be achieved with our system, and how does it compare to traditional hand-eye calibration?

[2]The network learns to estimate the positions of occluded keypoints from the surrounding context; technically, since the keypoints are the robot joints (which are inside the robot links), they are always occluded.

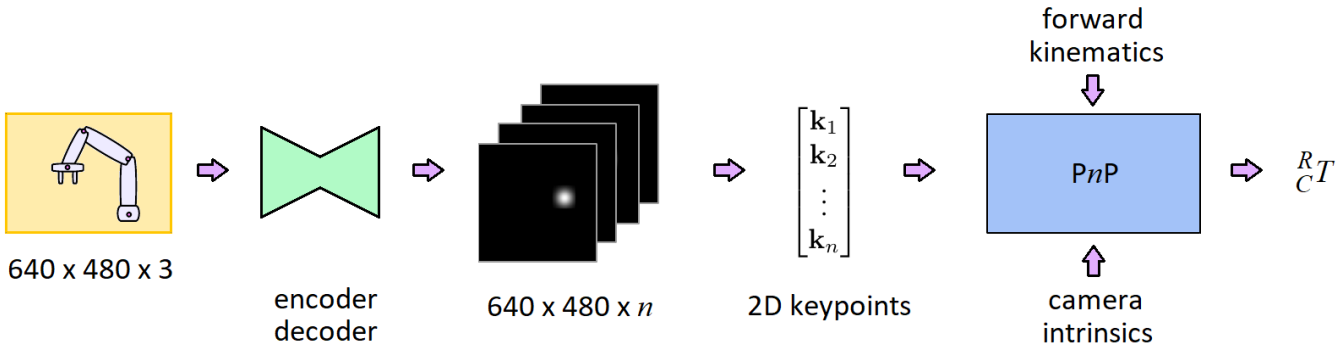

Fig. 1. The DREAM framework. A deep encoder-decoder neural network takes as input an RGB image of the robot from an externally-mounted camera, and it outputs n belief maps (one per keypoint). The 2D peak of each belief map is then extracted and used by PnP, along with the forward kinematics and camera intrinsics, to estimate the camera-to-robot pose, ${}^{R}_{C}T$.

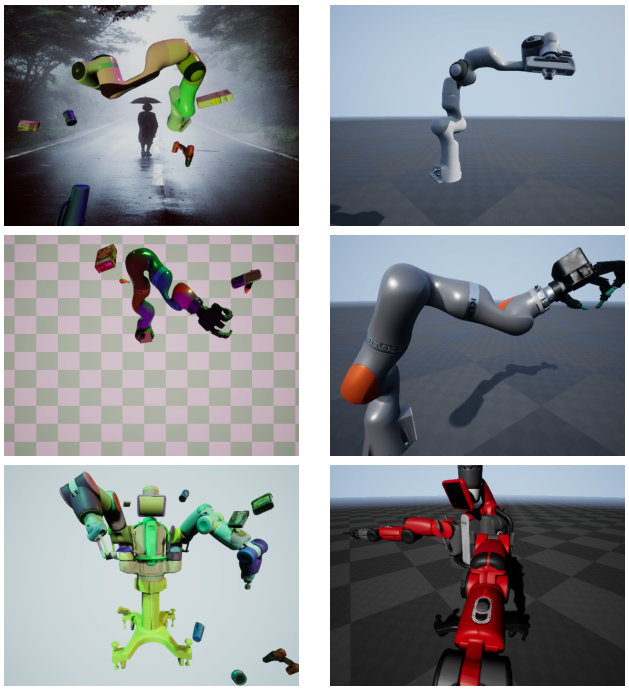

Fig. 2. Synthetic training images for the three robot models: Franka Panda (top), Kuka LBR with Allegro hand (middle), and Rethink Baxter (bottom).

A. Datasets

We collected several real-world datasets in our lab using various RGBD sensors. Since our DREAM algorithm processes only RGB images, the depth images were captured solely to facilitate ground truth camera poses via DART [19], a depth-based articulated tracking algorithm. DART was initialized manually and monitored in real-time during data collection to ensure tracking remained stable and correct, by viewing the projection of the robot coordinate frames onto the camera's RGB images via the ROS visualization tool (RViz).

Panda-3Cam. For this dataset, the camera was placed on a tripod aimed at the Franka Emika Panda robot arm. The robot was moved to five different joint configurations, at which the camera collected data for approximately five seconds each, followed by manual teleoperation to provide representative end effector trajectories along cardinal axes, as well as along a representative reaching trajectory. During data collection, neither the robot base nor the camera moved. The entire capture was repeated using *three different cameras* utilizing different modalities: Microsoft XBox 360 Kinect (structured light), Intel RealSense D415 (active stereo), and Microsoft Azure Kinect (time-of-flight). All images were recorded natively at 640×480 resolution at 30 fps, except for the Azure Kinect, which was collected at 1280×720 and downsampled to 640×480 via bicubic interpolation. This dataset consists of 17k total image frames divided approximately equally between the three cameras.

Panda-Orb. To evaluate the method's ability to handle a variety of camera poses, additional data was captured of the Panda robot. The RealSense camera was again placed on a tripod, but this time at 27 different positions in a roughly *orbital* motion around the robot (namely, 9 different azimuths, ranging from roughly -180° to +180°, and for each azimuth two different elevations approximately 30° and 45°, along with a slightly closer depth at 30°). For each camera pose, the robot was commanded using Riemannian Motion Policies (RMPs) [20], [21] to perform the same joint motion sequence of navigating between 10 waypoints defined in both task and joint space. The dataset consists of approximately 40k image frames.

B. Metrics

Metrics were computed on both 2D and 3D. For the 2D evaluation, the percentage of correct keypoints (PCK) [22] below a certain threshold was calculated, as the threshold varied. All keypoints whose ground truth was within the camera's frustum were considered, regardless of whether they were occluded. For 3D evaluation of the accuracy of the final camera-to-robot pose, the average distance (ADD) [23], [24] was calculated, which is the average Euclidean distance of all 3D keypoints to their transformed versions, using the estimated camera pose as the transform. ADD is a principled way, based upon Euclidean geometry, to combine rotation and translation errors without introducing

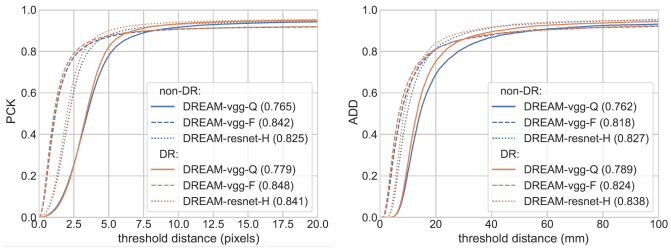

Fig. 3. PCK (left) and ADD (right) results for three different variants of our DREAM network on the two simulated datasets. The numbers in parentheses are the area under the curve (AUC).

an arbitrary weighting between them. As with PCK, the percentage of keypoints with ADD lower than a threshold was calculated, as the threshold varied. For ADD, all keypoints were considered. In both cases, averages were computed over keypoints over all image frames.

C. Training and Simulation Experiments

For comparison, we trained three versions of our DREAM network. As described earlier, the architecture uses either a VGG- or ResNet-based encoder, and the decoder outputs either full (F), half (H), or quarter (Q) resolution. Each neural network was trained for 50 epochs using Adam [25] with 1.5e-4 learning rate and 0.9 momentum. Training for each robot used approximately 100k synthetic DR images. The best-performing weights were selected according to a synthetic validation set.

As a baseline, Fig. 3 compares these versions on two simulated datasets, one with domain-randomization (DR) and the other without (non-DR). The improvement due to increasing resolution is clear, but different architectures have only minimal impact for most scenarios.

D. Real-world Experiments

Results comparing the same three networks on the Panda-3Cam dataset are shown in Fig. 4. Encouragingly, these results show that our training procedure is able to bridge the reality gap: There is only a modest difference between the best performing network on simulated and real data.

For 3D, it is worth noting that the standard deviation of ground truth camera pose from DART was 1.6 mm, 2.2 mm, and 2.9 mm, respectively, for the XBox 360 Kinect (XK), RealSense (RS), and Azure Kinect (AK) cameras, respectively. The degraded results for the Azure Kinect are due to DART's sensitivity to noise in the time-of-flight-based depth, rather than to DREAM itself. On the other hand, the degraded results for XBox 360 Kinect are due to the poor RGB image quality from that sensor.

Ultimately, the goal is to be able to place the camera at an arbitrary pose aimed at a robot, and calibrate automatically. To measure DREAM's ability to achieve this goal, we evaluated the system on the Panda-Orb dataset containing multiple camera poses. These results, alongside those of the previous experiments, are shown in Tab. I. For this table we used DREAM-vgg-F, since the other architectures performed similarly as before.

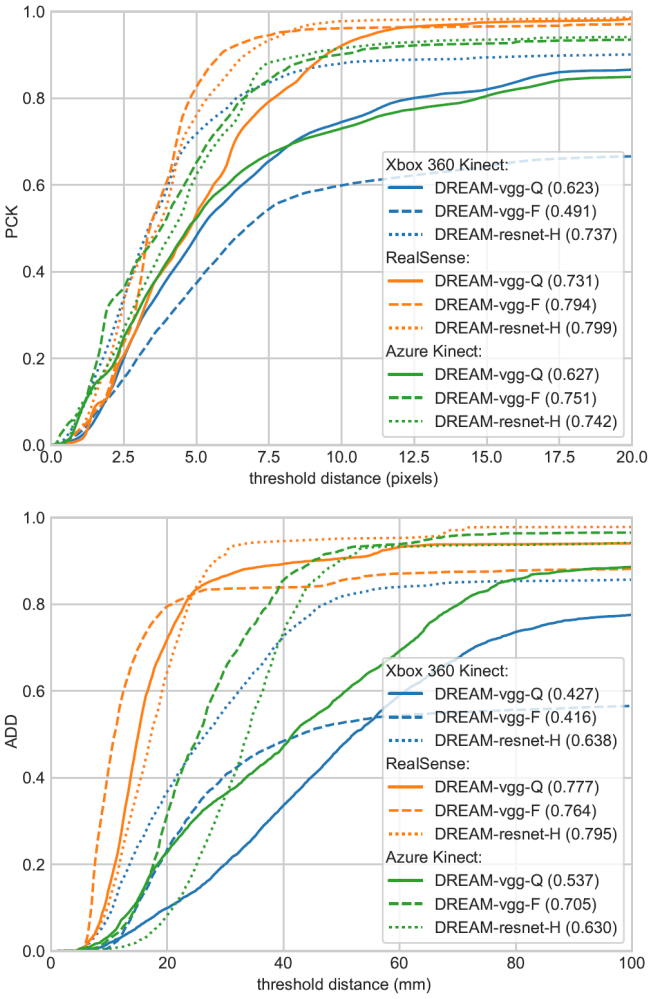

Fig. 4. PCK (top) and ADD (bottom) results on the real Panda-3Cam dataset.

We also trained DREAM on the Kuka LBR iiwa and Baxter robots. While the former is similar to the Panda, the latter is more difficult due to symmetry that causes the two arms to be easily confused with one another. To alleviate this problem, we had to restrict the azimuth range from -45° to +45°. Although we did not perform quantitative analysis on either robot, qualitatively we found that the approach works about the same for these robots as it does for the Panda. The detected keypoints overlaid on images of the three robots, using three different RGB cameras, are shown in Fig. 5. Although in principle the keypoints could be placed anywhere on the robot, we simply assign keypoints to the joints according to each robot's URDF (Unified Robot Description Format).

E. Comparison with Hand-Eye Calibration

The goal of our next experiment was to assess the accuracy of DREAM versus traditional hand-eye calibration (HEC). For the latter, we used the `easy_handeye` ROS package[3]

[3]`https://github.com/IFL-CAMP/easy_handeye`

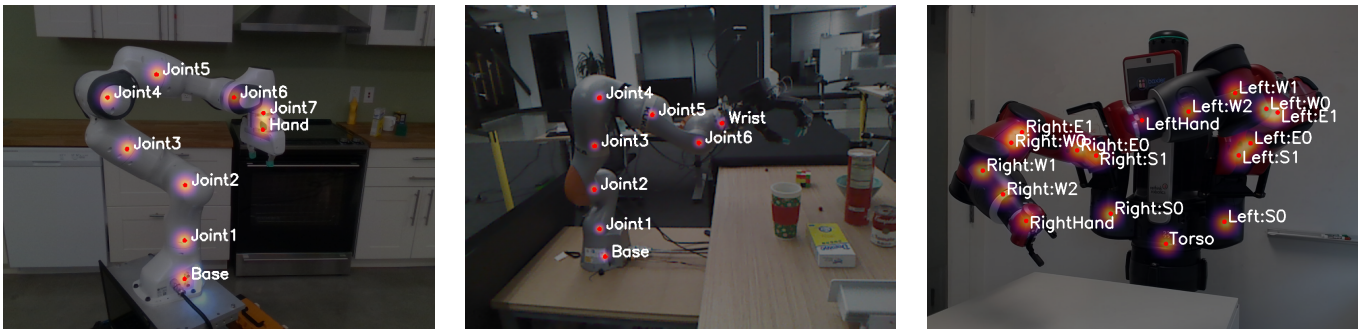

Fig. 5. Keypoint belief maps (red dots indicate peaks) detected by DREAM in RGB images of three different robots (taken by three different cameras). From left to right: Franka Emika Panda (Intel RealSense D415), Kuka LBR iiwa (Logitech C920 webcam), and Rethink Baxter (cell phone camera).

TABLE I

RESULTS AT DIFFERENT THRESHOLDS OF THE SAME NETWORK (DREAM-VGG-F) ON VARIOUS DATASETS (AND CAMERA SENSORS). ALL BUT THE LAST ROW ARE TAKEN FROM FIGS. 3 AND 4.

	PCK @ (pix)			ADD @ (mm)		
Dataset	2.5	5.0	10.0	20	40	60
Sim. DR	0.79	0.88	0.90	0.81	0.88	0.90
Sim. non-DR	0.77	0.87	0.90	0.80	0.88	0.90
Panda-3Cam (XK)	0.15	0.37	0.59	0.23	0.48	0.54
Panda-3Cam (RS)	0.24	0.83	0.96	0.80	0.83	0.87
Panda-3Cam (AK)	0.36	0.65	0.90	0.32	0.86	0.94
Panda-Orb (RS)	0.28	0.67	0.83	0.57	0.77	0.80

to track an ArUco fiducial marker [1] attached to the Panda robot hand.

The XBox 360 Kinect was mounted on a tripod, and the robot arm was moved to a sequence of $M = 18$ different joint configurations, stopping at each configuration for one second to collect data. Neither the camera nor the base of the robot moved. The fiducial marker was then removed from the hand, and the robot arm was moved to a different sequence of M joint configurations. The joint configurations were selected favorably to ensure that the fiducial markers and keypoints, respectively, were detected in the two sets of images. As before, DART with manual initialization was used for ground truth.

Although our DREAM approach works with just a single RGB image, it can potentially achieve greater accuracy with multiple images by simply feeding all detected keypoints (from multiple frames) to a single PnP call. Thus, to facilitate a direct comparison with HEC, we applied DREAM to $m \geq 1$ images from the set of M images that were collected. Similarly, we applied HEC to $m \geq 3$ images from the set. Both algorithms were then evaluated by comparing the estimated pose with the ground truth pose via ADD. For both algorithms, we selected $\binom{M}{m}$ possible combinations when evaluating the algorithm on m images, to allow the mean, median, min, and max to be computed. To avoid unnecessary combinatorial explosion, whenever this number exceeded $N = 2500$, we randomly selected N combinations rather than exhaustive selection.

Results of this head-to-head comparison are shown in Fig. 6. Note that HEC is unable to estimate the camera pose when $m < 3$, whereas DREAM works with just a single image. As the number of images increases, the estimated pose from both DREAM and HEC improves, depending somewhat on the robot configurations used. In all cases, DREAM performs as well or better than HEC. (Note, however, that HEC results would likely improve from manually, rather than randomly, selecting image frames.)

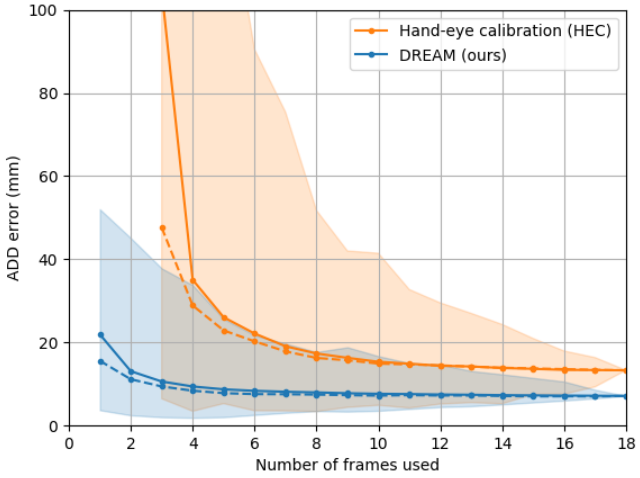

Fig. 6. DREAM vs. HEC, measured by ADD as a function of the number of image frames used for calibration. Shown are the mean (solid line), median (dashed line), and min/max (shaded area), computed over different image combinations. DREAM requires only a single image frame but achieves greater accuracy with more images.

F. Measuring Workspace Accuracy

In the final experiment we evaluated the accuracy of DREAM's output with respect to the workspace of the robot. The RealSense camera was placed on a tripod facing the Panda robot reaching toward an adjustable-height table on which were placed five fiducial markers. A head-to-head comparison of the camera poses computed by DART, DREAM, and HEC was conducted by transforming each fiducial marker's pose from the camera's frame to the robot's frame by applying each algorithm's camera-to-robot pose estimate. The robot was then commanded to move the end effector to a target position defined 10 cm directly above

TABLE II

EUCLIDEAN ERROR BETWEEN THE ROBOT'S ACTUAL REACHED POSITION AND THE COMMANDED POSITION, USING THE CAMERA POSE ESTIMATED BY EACH OF THE THREE METHODS.

	DART	HEC	DREAM (ours)
camera	depth	RGB	RGB
no. frames	1	10	1
min error (mm)	10.1	**9.4**	20.2
max error (mm)	44.3	51.3	**34.7**
mean error (mm)	**21.4**	28.2	27.4
std error (mm)	12.3	14.2	**4.7**

the marker, to avoid potential collision. This process was repeated for ten target positions (5 markers, 2 table heights). The Euclidean distance between the end effector's position in 3D was measured for each algorithm. Note that in this experiment DART was not considered to be ground truth, but rather was compared against the other methods.

Results are shown in Tab. II. Even though DREAM is RGB-only, it performs favorably not only to HEC but also to the depth-based DART algorithm. This is partly explained by the fact that the extrinsic calibration between the depth and RGB cameras is not perfect. Note that DREAM's error is similar to that of our previous work [14] upon which it is based, where we showed that an error of approximately 2 cm for object pose estimation from RGB is possible.

IV. PREVIOUS WORK

Relationship to previous work is considered in this section.

Object pose estimation. In robotics applications, it is not uncommon for objects to be detected via fiducial markers [26], [27], [28]. Even so, there is growing interest in the problem of markerless object pose estimation in both the robotics and computer vision communities [14], [23], [29], [9], [24], [11], [10], [30], [15], building upon work in keypoint detection for human pose estimation [31], [32], [13], [33], [34]. Recent leading methods are similar to the approach proposed here: A network is trained to predict object keypoints in the 2D image, followed by PnP [7] to estimate the pose of the object in the camera coordinate frame [14], [10], [35], [11], [15], [36], or alternatively, a deformable shape model is fit to the detect keypoints [37]. Indeed, our approach is inspired by these methods. Our approach builds upon the findings of Peng *et al.* [10], who showed that regressing to keypoints on the object is better than regressing to vertices of an enveloping cuboid. Other methods have regressed directly to the pose [24], [30], but care must be taken not to bake the camera intrinsics into the learned weights.

Robotic camera extrinsics. Closely related to the problem of estimating the camera-to-object pose (just described) is that of estimating the camera-to-robot pose. The classic solution to this problem is known as hand-eye calibration, in which a fiducial marker (such as ArUco [1], ARTag [2], AprilTag [3], or otherwise known object) is attached to the end effector and tracked through multiple frames. Using forward kinematics and multiple recorded frames, the algorithm solves a linear system to compute the camera-to-robot transform [38], [39], [40]. Similarly, an online calibration method is presented by Pauwels and Kragic [41], in which the 3D position of a known object is tracked from nonlinear optimization over multiple frames.

An alternate approach is to move a small object on a table, command the robot to point to each location in succession, then use forward kinematics to calibrate [42]. However, the accuracy of such an approach degrades as the robot moves away from the table used for calibration. Aalerund *et al.* [43] present a method for calibrating an RGBD network of cameras with respect to each other for robotics applications, but the camera-to-robot transforms are not estimated.

For completeness, we mention that, although our paper addresses the case of an externally mounted camera, another popular configuration is to mount the camera on the wrist [44], for which the classic hand-eye calibration approach applies [41]. Yet another configuration is to mount the camera on the ceiling pointing downward, for which simple 2D correspondences are sufficient [45], [46], [42].

Robotic pose estimation. Bohg *et al.* [47] explore the problem of markerless robot arm pose estimation. In this approach, a random decision forest is applied to a depth image to segment the links of the robot, from which the robot joints are estimated. In follow-up work, Widmaier *et al.* [48] address the same problem but obviate the need for segmentation by directly regressing to the robot joint angles. Neither of these approaches estimate the camera-to-robot transform.

The most similar approach to ours is the simultaneous work of Lambrecht and Kästner [49], [50], in which a deep network is also trained to detect projected keypoints, from which the camera-to-robot pose is computed via PnP. A key difference is that our network is trained only on synthetic data, whereas theirs requires real and synthetic data. In other recent work, Zuo *et al.* [51] also present a keypoint-based detection network. But rather than use PnP, nonlinear optimization directly regresses to the camera pose and the unknown joint angles of a small, low-cost manipulator. The network is trained using synthetic data, with domain adaptation to bridge the reality gap.

V. CONCLUSION

We have presented a deep neural network-based approach to compute the extrinsic camera-to-robot pose using a single RGB image. Compared to traditional hand-eye calibration, we showed that our DREAM method achieves comparable accuracy even though it does not use fiducial markers or multiple frames. Nevertheless, with additional frames, our method is able to reduce the error even further. We have presented quantitative results on a robot manipulator using images from three different cameras, and we have shown qualitative results on other robots using other cameras. We believe the proposed method takes a significant step toward robust, online calibration. Future work should be aimed at filtering results over time, computing uncertainty, and incorporating the camera pose into a closed-loop grasping or manipulation task.

ACKNOWLEDGMENTS

We gratefully acknowledge Karl van Wyk, Clemens Eppner, Chris Paxton, Ankur Handa, and Erik Leitch for their help. Many thanks also to Kevin Zhang and Mohit Sharma (Carnegie Mellon University).

REFERENCES

[1] S. Garrido-Jurado, R. M. noz Salinas, F. J. Madrid-Cuevas, and M. J. Marn-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[2] M. Fiala, "ARTag, a fiducial marker system using digital techniques," in *CVPR*, 2005.

[3] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *ICRA*, 2011.

[4] I. Fassi and G. Legnani, "Hand to sensor calibration: A geometrical interpretation of the matrix equation $AX{=}XB$," *Journal on Robotics Systems*, vol. 22, no. 9, pp. 497–506, 2005.

[5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IROS*, 2017.

[6] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[7] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EP*n*P: An accurate O(*n*) solution to the P*n*P problem," *International Journal Computer Vision*, vol. 81, no. 2, 2009.

[8] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, and S. Birchfield, "NDDS: NVIDIA deep learning dataset synthesizer," 2018, https://github.com/NVIDIA/Dataset_Synthesizer.

[9] S. Zakharov, I. Shugurov, and S. Ilic, "DPOD: Dense 6D pose object detector in RGB images," *arXiv preprint arXiv:1902.11020*, 2019.

[10] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *CVPR*, 2019.

[11] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6D object pose estimation," in *CVPR*, 2019.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[13] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *ECCV*, 2018.

[14] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *CoRL*, 2018.

[15] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6D object pose prediction," in *CVPR*, 2018.

[16] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *CVPR*, 2014.

[17] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB object and model set: Towards common benchmarks for manipulation research," in *ICAR*, 2015.

[18] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *CVPR Workshop on Autonomous Driving*, 2018.

[19] T. Schmidt, R. A. Newcombe, and D. Fox, "DART: Dense articulated real-time tracking," in *Robotics: Science and Systems (RSS)*, 2014.

[20] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," in *arXiv:1801.02854*, 2018.

[21] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "RMPflow: A computational graph for automatic motion policy generation," in *WAFR*, 2018.

[22] J. Tremblay, T. To, A. Molchanov, S. Tyree, J. Kautz, and S. Birchfield, "Synthetically trained neural networks for learning human-readable plans from real-world demonstrations," in *ICRA*, 2018.

[23] 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 *ACCV*, 2012.

[24] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *RSS*, 2018.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[26] C. Liu and M. Tomizuka, "Robot safe interaction system for intelligent industrial co-robots," in *arXiv:1808.03983*, 2018.

[27] C. H. Kim and J. Seo, "Shallow-depth insertion: Peg in shallow hole through robotic in-hand manipulation," in *ICRA*, 2019.

[28] N. Tian, A. K. Tanwani, J. Chen, M. Ma, R. Zhang, B. H. K. Goldberg, and S. Sojoudi, "A fog robotic system for dynamic visual servoing," in *ICRA*, 2019.

[29] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," in *WACV*, 2017.

[30] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *ECCV*, 2018.

[31] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[32] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *CVPR*, 2017.

[33] W. Li, Z. Wang, B. Yin, Q. Peng, Y. Du, T. Xiao, G. Yu, H. Lu, Y. Wei, and J. Sun, "Rethinking on multi-stage networks for human pose estimation," *arXiv preprint arXiv:1901.00148*, 2019.

[34] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *CVPR*, 2019.

[35] D. J. Tan, N. Navab, and F. Tombari, "6D object pose estimation with depth images: A seamless approach for robotic interaction and augmented reality," in *arXiv 1709.01459*, 2017.

[36] A. Dhall, D. Dai, and L. V. Gool, "Real-time 3D traffic cone detection for autonomous driving," in *IEEE Intelligent Vehicles Symp.*, 2019.

[37] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-DoF object pose from semantic keypoints," in *ICRA*, 2017.

[38] F. C. Park and B. J. Martin, "Robot sensor calibration: solving $AX{=}XB$ on the Euclidean group," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 717–721, 1994.

[39] J. Ilonen and V. Kyrki, "Robust robot-camera calibration," in *Proceedings, International Conference on Advanced Robotics*, 2011.

[40] D. Yang and J. Illingworth, "Calibrating a robot camera," in *BMVC*, 1994.

[41] K. Pauwels and D. Kragic, "Integrated on-line robot-camera calibration and object pose estimation," in *ICRA*, 2016.

[42] D. Park, Y. Seo, and S. Y. Chun, "Real-time, highly accurate robotic grasp detection using fully convolutional neural networks with high-resolution images," in *arXiv:1809.05828*, 2018.

[43] A. Aalerud, J. Dybedal, and G. Hovland, "Automatic calibration of an industrial RGB-D camera network using retroreflective fiducial markers," *Sensors*, vol. 19, no. 7, 2019.

[44] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," in *RSS*, 2018.

[45] A. Feniello, H. Dang, and S. Birchfield, "Program synthesis by examples for object repositioning tasks," in *IROS*, 2014.

[46] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *RSS*, 2017.

[47] J. Bohg, J. Romero, A. Herzog, and S. Schaal, "Robot arm pose estimation through pixel-wise part classification," in *ICRA*, 2014.

[48] F. Widmaier, D. Kappler, S. Schaal, and J. Bohg, "Robot arm pose estimation by pixel-wise regression of joint angles," in *ICRA*, 2016.

[49] J. Lambrecht and L. Kästner, "Towards the usage of synthetic data for marker-less pose estimation of articulated robots in RGB images," in *International Conference on Advanced Robotics (ICAR)*, 2019.

[50] J. Lambrecht, "Robust few-shot pose estimation of articulated robots using monocular cameras and deep-learning-based keypoint detection," in *Intl. Conf. on Robot Intell. Techn. and Applications (RITA)*, 2019.

[51] Y. Zuo, W. Qiu, L. Xie, F. Zhong, Y. Wang, and A. L. Yuille, "CRAVES: Controlling robotic arm with a vision-based economic system," in *CVPR*, 2019.