1

Vision-Based State Estimation of a Serial Manipulator

Aishani Pathak[1], James Halverson[2], Kenny DeCay Jr.[3], Anand Iyer[4], Joshua Tempelman[4], Cole Maxwell[4], Ricardo Mejia-Alvarez[5]

[1], Arizona State University, Tempe, Arizona, U.S.
[2], Michigan Technological University, Houghton, Michigan, U.S.
[3], Texas A&M University, College Station, Texas, U.S.
[4], Los Alamos National Laboratory, Los Alamos, New Mexico, U.S.
[5], Michigan State University, East Lansing, Michigan, U.S.

ABSTRACT

We present an approach for estimating the configuration of a robotic manipulator using computer vision. While sensors and encoders are built into the design of most industrial robotic arms, there are settings where integrating such components is infeasible due to constraints such as cost, complexity, or operational environment. In such cases, alternative methods of state estimation are required. This work leverages a transfer learning-based method to extract the joint configuration of a robotic arm in 3D space from images. Using a pretrained vision model as a backbone, we tune the network to extract informative feature maps corresponding to each degree of freedom (DOF) of the manipulator; a dense network subsequently infers the corresponding state vector. We propose a stage-based approach with a simplified model training pipeline to estimate the state of a serial manipulator within a real environment.

Keywords: Vision, State Estimation, Manipulator, Model, Encoderless.

Introduction

State estimation is the process of inferring the current state (e.g. position, velocity, orientation) of a dynamic system. In robotics, this allows for real-time localization where the configuration of an arm is determined through the estimation of the angular and translational components of individual joints or linkages [1, 2]. Most robotic systems accomplish this using embedded encoders, with each joint that composes an arm contains an encoder which returns its current state [3]. This state measurement is then traditionally used as the feedback element for a control system. **Figure 1** depicts a generic control system. Error is defined as the difference between the reference input angle and the output angle. A negligible error indicates completion of the system translation [4].

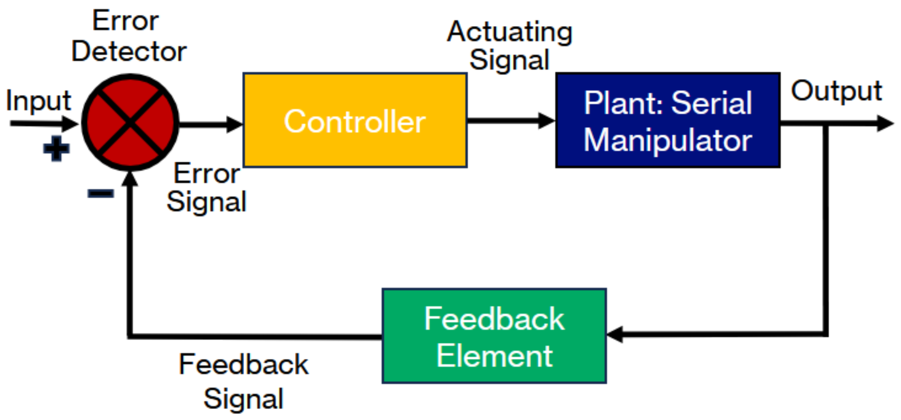


Figure 1: Basic closed-loop control system diagram.

Although encoder based state estimation is robust and has proven precision, there are numerous scenarios in which encoder reliability struggles or are infeasible to implement. For example, in high radiation environments where gamma and beta radiation interfere with encoders [5], or on-orbit servicing operations where encoder failure can lead to neighboring object damage [6] and replacement costs are prohibitive. These constraints motivate the need for alternative state estimation methods. One of the predominant alternatives is the use of vision for encoderless state estimation and control. In general, previous vision based approaches can be separated into two categories: two-dimensional (2D) and three-dimensional (3D) methods. 3D methods primarily involve the use of RGB-Depth cameras or Light Detection and Ranging (LiDAR) systems [7, 8]—these have been effective for mobile robot object tracking or end effector state estimation, but often suffer from self occlusion and high computational expense. Further information on 3D methods please reference [9–11] To avoid the limitations of 3D vision methods, this considers a 2D method. In contrast to 3D methods, 2D methods utilize traditional imagery, making them less computationally expensive, as they perform feature extraction solely in the image plane. This computational savings is essential for vision based control, where near real-time feedback is essential. There are two main 2D image based techniques for state estimation. The first is the use of fiducial markers where physical points or markings are placed onto the system. These predefined points serve as tracking markers to determine the current state. The major drawback of this method is that the accuracy of detecting these markers suffers under camera motion blur, self-occlusion when markers are blocked from the camera, and false-positive detections from camera artifacts [12]. The second is the use of keypoints, whereby non-physical points are algorithmically placed on to features that can be reliably tracked for state estimation. Although the use of keypoints is often more robust, the detection accuracy of these virtual markers can also suffer from self-occlusion, false feature matches, and environmental lighting conditions. In addition, this method is also far more computationally expensive [13].

The primary drawback of the methods described above is their lack of generalizability. The 2D methods described above are largely intended for use on a single manipulator, and do not inherently generalize well to different manipulator configurations or alternate robotic arms. Recently, advancements in machine learning have allowed for more robust robotic state estimation using 2D imagery. Machine learning, and more specifically computer vision, has shown promise in bridging this generalization gap. In this paper, we present an alternative 2D method of state estimation using machine learning based computer vision, through the implementation of a convolutional neural network (CNN) trained on planar images of a serial manipulator. Through this approach we demonstrate the ability of a CNN architecture to perform state estimation on a serial manipulator without modification to the arm itself, using purely vision. This methodology allows for future adaption of this model to alternate arms or configurations as there is no embedded information of our specific training manipulator. This work provides motivation for further research into computer vision based inference and control methods that can provide accurate and robust state estimation without the use of encoders.

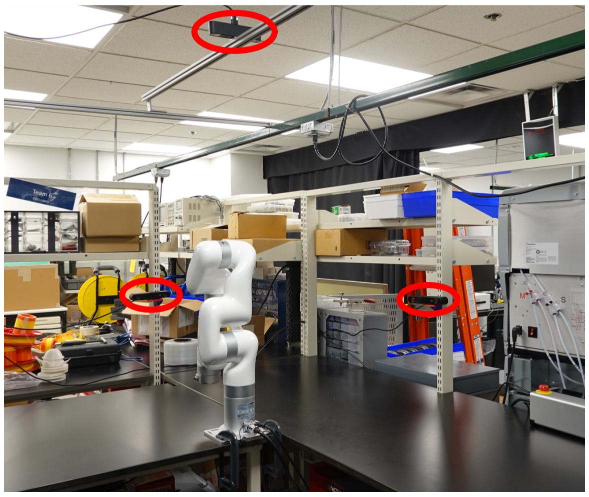


Figure 2: Real-world setup with cameras marked.

Methods

To leverage a computer vision model for state inference, large amounts of labeled data is needed for training. As such, both virtual and physical environments were created to collect data from a physical serial manipulator. To avoid the reliance on time-consuming real-world data collection, a physics-based simulation was utilized to curate a large and diverse dataset of training images. Although training with idealized data is not wholly representative of the desired physical environment, further tuning to this desired space requires much less real-world data. Specifically, we selected a transfer learning approach using ResNet-18 pretrained on ImageNet for its feature extraction abilities, with our target task being a regression over five joint angles. Finally, we considered various different training strategies to find an optimal model for this system, e.g., different loss functions, frozen ResNet parameters, shuffled batches, and weighted loss functions.

Real-World Setup

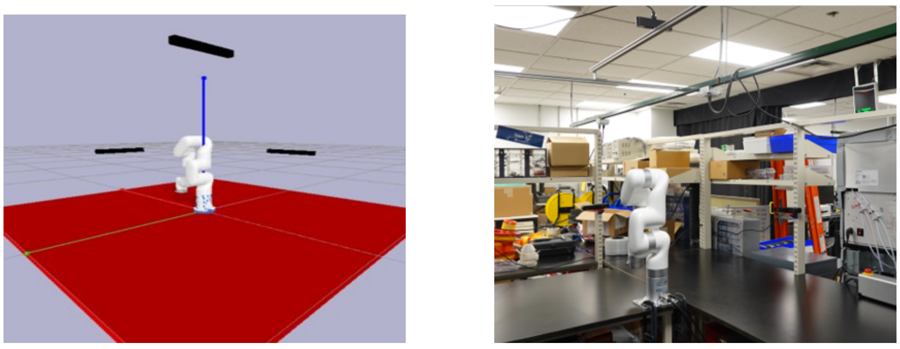
The physical environment and experimental setup collection of real-world data were carried out as follows. A monochrome UFACTORY X-ARM 6 serial manipulator was mounted to a table and observed with three RGB cameras positioned in orthogonal directions. **Figure 2** depicts the setup and three orthogonal chosen to optimize the independence of each image. Tarps were hung in the background of the manipulator with respect to each camera view to reduce distractions present in the physical environment. While this strategy reduced clutter, natural imperfections such as lighting variations and reflective surfaces remained to ensure that the physical data retained realistic challenges. Experimental images were captured using three ZED 2i RGB cameras with a 70° vertical field of view (FOV) and an aspect ratio of 16:9.

Simulation Setup

The simulation environment was implemented using PyBullet, which is a Python-based interface to the Bullet Physics Engine. PyBullet provides a lightweight simulation framework with the ability to support various robot models and perform collision detection and camera rendering. The simulation environment was configured to emulate the physical setup, replicating the manipulator and the three camera configuration. **Figure 3** shows the manipulator in both the simulated and physical environments, with the arm configuration being positioned identically and captured from the same camera angle. Moreover, the simulation also incorporated a uniform black background to maintain further consistence with the tarp background of the physical environment (not shown).

This ensured geometric and visual consistency between simulation and physical environments allowing exploration of a wide range of manipulator configurations at a low computational cost. A benefit of the simulation framework is its ability to automatically flag invalid configurations, i.e., a set of joint angles would cause the manipulator to collide with itself or with a rigid object in its environment. Using the "getContactPoints" method, which detects when collidable geometry intersect, these configurations were excluded to better represent reality and to reduce the size of the training set. Moreover, to ensure physical consistency of the PyBullet simualtions, several verification and validation strategies were employed including MATLAB Simulink, kinematic calculations, and the physical manipulator's software package.

Simulation images were captured using the same FOV utilized by the cameras in the physical setup. For reference, the three camera views at 40° and 70° vertical FOV are shown in **Figure 4**. Using an increased FOV with a constrained image size does decrease the angular resolution of the image, seen by the stretched or otherwise distorted geometry near the edges of the bottom row of images [14]. However, given the nature of a regression



(a) Simulation environment **(b)** Real environment

Figure 3: Comparison between (a) simulated and (b) real environments with identical manipulator states and viewpoints.

model and our method of preprocessing, we do not expect this smearing to negatively impact performance.

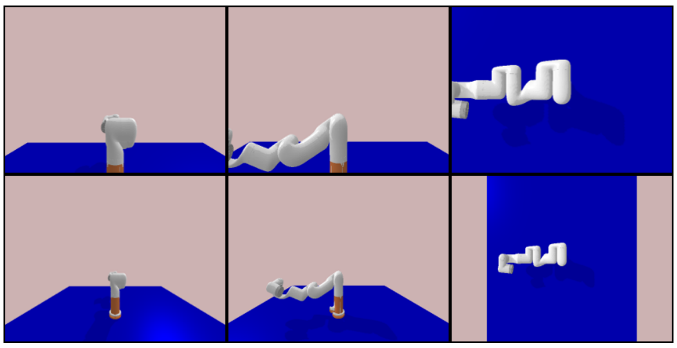


Figure 4: Orthogonal camera perspectives for 40° vertical FOV (first row) and 70° vertical FOV (second row), as seen in PyBullet.

Data Curation and Collection

Prior to data acquisition, the valid configuration were first determined. We comment that only the first five joints of the UFACTORY X-Arm 6 were considered. In the absence of an end effector, the final wrist joint was excluded due to the small physical size of the final link.

Table 1: Comparison of the manufacturers joint limitations and the modified joint limits in units of radians for the first five joints.

Joint Number	Original		Modified	
	Lower	Upper	Lower	Upper
1	-6.283	6.283	0.1	6.283
2	-2.059	2.094	-2.059	2.094
3	-3.927	0.192	-3.927	0.192
4	-6.283	6.283	0.1	6.283
5	-1.693	3.142	-1.693	3.142

Using the representative PyBullet model, joint angles were discretized at 15° increments, providing sufficient coverage of the configuration space for the model to interpolate between states. Since sampling the entire range of -2π to 2π led to duplicate configurations, the limits for these joints were constrained. Additionally, a small offset of 0.1 radians was introduced instead of a hard zero to avoid overlap with the upper bound at 2π, where the joint yet again assumed the same physical orientation as at 0 radians. These adjustments are highlighted in **Table 1**.

Data curation resulted in a total of 75,832 possible configurations. Nearly all these images were collected from the simulation environment and later used to train the ResNet-18 model to the simulation space. The dataset was augmented with 500 samples from the physical environment which were later used to tune the computer vision to real-world measurements.

Despite prior efforts to exclude invalid configurations from this set, certain configurations did not account for the path from one configuration to the next and were thus not achievable in the real-world environment. Additionally, the self-collision detection built into the physical manipulator's controller was more sensitive than the simulation's "getContactPoints" method, meaning that the controller would prevent the manipulator from achieving it. Efforts were taken to address these challenges by computing the minimum distances to collidable geometries and imposing a threshold. However, it was deemed most practical to manually reject colliding configurations from the relatively small data set.

Preprocessing

Once data was collected over both environments, training and testing datasets were preprocessed to provide standardized input to the model. Given that the manipulator only occupies a portion of each image, each orthogonal view was first center-cropped to 470 × 470 pixels. This crop size was chosen to retain regions relevant to the manipulator's motion while discarding less informative background areas. The cropping not only preserves important features but reduces the computational load as well by limiting the number of processed pixels. Following this, images were resized to 200 × 200 to match the input dimensions required by ResNet-18 and was normalized using ImageNet's expected channel-wise mean and standard deviation: mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]. Finally, the images were concatenated horizontally producing a single 200 × 600 image to form a composite representation of the manipulator from all three directions.

Alongside image preprocessing, target joint angles were encoded in sine-cosine form as $(\sin\theta, \cos\theta)$ to improve learning stability. Because raw angular representations can create discontinuities in the dataset, angles near 0 and 2π are visually almost identical but numerically appear far apart. These discontinuities can confuse a regression model, leading to poor generalization and instability in predictions. Hence, this transformation embeds angular values on the unit circle, where equivalent orientations were mapped close together providing an accurate representation space. Moreover, it improved stability during training by providing smooth target space with a bounded range $[-1, 1]$.

Model Architecture

The primary objective was to estimate the joint states of the manipulator using multi-view RGB images. To leverage existing visual representations, a transfer learning approach was employed by using a pretrained ResNet-18 model as the network backbone. The ResNet-18 model employed in this work was previously trained on ImageNet, a dataset of natural images, which provided rich feature extraction capabilities. This reduced computational cost and accelerated convergence, while still allowing the network to adapt to high level visual representations of the manipulator.

To better equip the network for the regression task, modifications were to made to the final layer of the network architecture. The single standard fully connected layer of the pretrained ResNet-18 was replaced with a regression head. This layer was implemented as a multi-layer perceptron (MLP) consisting of five fully connected layers with ReLU activations and dropout for regularization. The final layer outputs 10 values, corresponding to sine and cosine encodings of the five target joint angles. The network input consisted of RGB images from three orthogonal viewpoints concatenated channel-wise, forming a single composite image input. A visual representation of the proposed model architecture can be seen in **Figure 5**

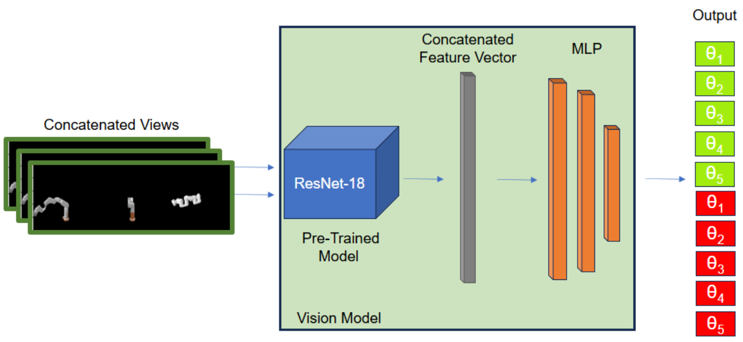


Figure 5: Overview of the proposed vision model architecture for joint angle estimation.

Training Strategies

Models were trained using the Adam optimizer with a fixed learning rate 0.0005 and a batch size of 64. A weight decay of 1×10^{-5} was applied for the simulation-only models, while a value of 1×10^{-6} was used for the hybrid-trained models to reduce overfitting. Training was performed up to 100 epochs to test convergence, with checkpoints saved at each epoch. Several variations were applied to training to find the optimal training configuration and evaluate robustness: Two loss functions were tested: mean squared error (MSE) which penalized large deviations heavily and smooth L1 Loss (Huber Loss) which acted like piecewise function behaving quadratically for smaller error and linearly for larger errors, providing a balance between characteristics of MSE and mean absolute error (MAE). Transfer learning was applied by employing ResNet-18 parameter freezing. Two modes for the ResNet-18 backbone were: Frozen and Unfrozen with the regression head trainable across both. In addition, dataloader shuffling (True vs. False) was assessed to see its impact. The combination of these techniques yielded eight total models, which were evaluated and compared to select the best performing case.

Experiment Stages

The experimental design followed a three-stage progression to address the simulation-to-real challenge and provide subsequent results for each stage.

Stage 1, simulation-only: Models were trained, validated, and tested on synthetic data that was generated in PyBullet. This stage was essential to develop a baseline and enable training on large scale data to identify the best-performing model.

Stage 2, zero-shot transfer: The trained simulation model was then only evaluated on a set of 50 real test images. This stage measured the extent of the domain gap and assessed both environments and how closely the model generalized to real environmental data with no prior learning.

Stage 3, hybrid model: The third and final stage leveraged the converged checkpoint of the simulation model as the initialization and further retrained it on real image data. This fine-tuning was applied to leverage the knowledge of the simulation environment while incorporating real data to adapt learned representations and narrow domain gap. Evaluation was conducted on the same 50 real test images to ensure consistency between Stage 2 and Stage 3.

Results

The results are organized in the following manner: simulation-only training, zero-shot transfer to real-world data, and hybrid training with simulated and real data. Although all models were trained for 100 epochs, convergence was typically achieved within the first 20 epochs, which is selected as the point of comparison for reporting results below.

Stage 1: Simulation-only

Within the simulation-only stage, several model variants configured from our training strategies were compared to identify the best-performing model.

As seen in **Figure 6**, all models achieved similar performance, with mean absolute error falling within a narrow range. Although, it was observed that the configuration using smooth L1 loss with an unfrozen ResNet-18 using non-shuffled training (L1_UnFrozen_SF) achieved the lowest error and was therefore selected for further evaluations in real and hybrid settings. To better analyze the chosen model's performance within simulation, per-joint relative error distributions and average absolute errors are reported. Joint errors were calculated using a wrapped difference formulation. This representation mapped angular differences near 0 and 2π into the range $[-\pi, \pi]$. This wrapping was primarily beneficial for joints 1 and 4, which allow for full 2π rotation, but was applied uniformly across all joints for consistency.

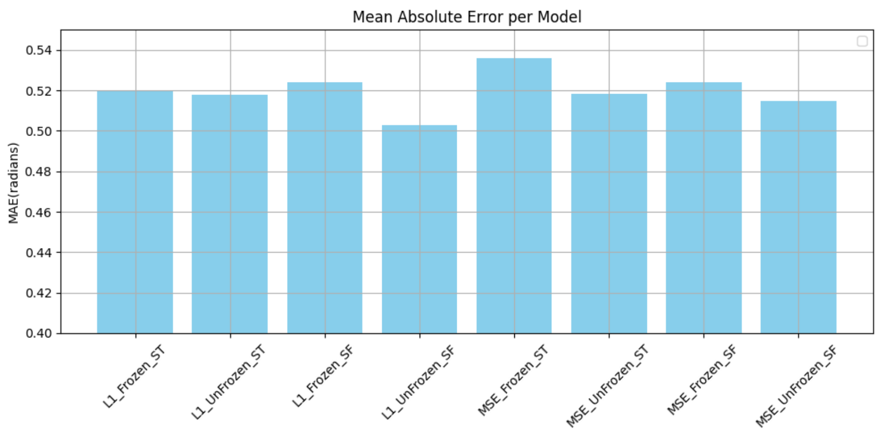


Figure 6: Comparison of mean absolute error across all eight simulation-trained models. The y-axis is narrowed to emphasize the differences among the best-performing variants.

From the average absolute errors shown in **Figure 7** and the relative error distributions shown in **Figure 8**, it it seen that joints 1-3 have more narrowed distributions with errors concentrated around zero radians. In contrast, joints 4 and 5 have more dispersed distributions indicating higher errors. A hypothesis for this outcome is that these joints are located closed to the manipulator's end-effector, a position that is prone to self-occlusions in certain configurations and in certain cases, camera occlusions. Additionally, the links between joints 4 and 5 are shorter leading to subtler changes in appearance, which may be difficult for our network to capture consistently. Similar trends are noticed in the following evaluations as well.

Stage 2: Zero-shot Transfer

Using the best-performing simulation model case, zero-shot transfer was performed. As shown in **Figure 9**, error magnitudes increase broadly across all joints as compared to the simulation results. This is expected due to the visual domain gap: real images introduce conditions such as lighting variations and partial occlusions that are not possible to replicate in simulation training. These differences in training and testing data make it difficult for

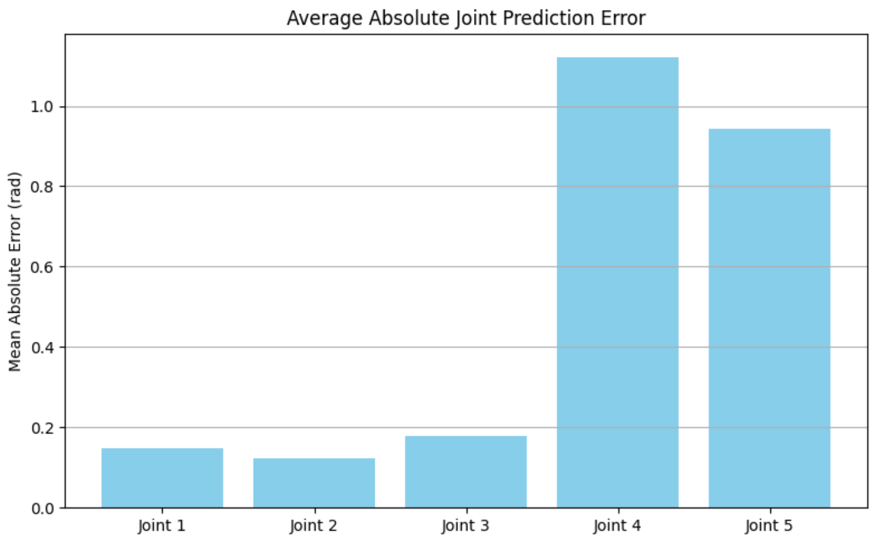


Figure 7: Mean absolute error (MAE) for each of the five joints, averaged across simulation test samples.

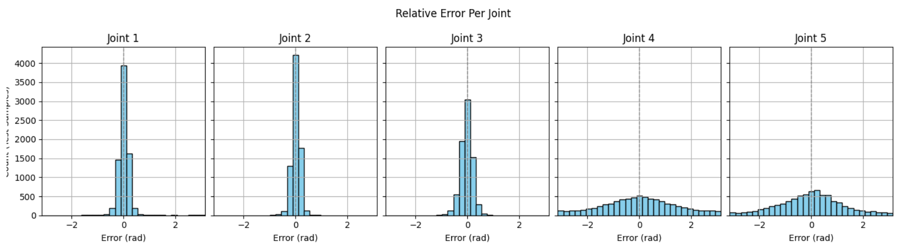


Figure 8: Relative error distributions for each of the five joints across simulation test samples.

the model to extract stable visual features. The relative error distributions in **Figure 10** further illustrate this effect. While joint 4 and joint 5 remain the most challenging, joints 1-3 indicate that the domain gap affects the entire system rather than specific joints.

In addition to quantitative errors, a qualitative comparison of the predicted against the ground truth joint values for a single configuration is shown in **Figure 11**. Generalization is visible in the predicted results, where the overall

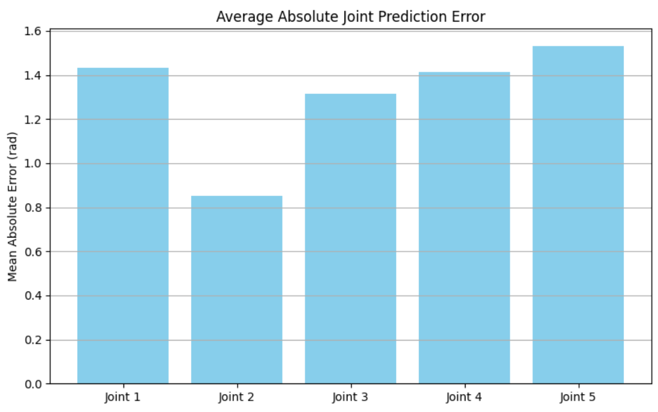


Figure 9: Mean absolute error (MAE) per joint for the simulation-trained model evaluated on real images.

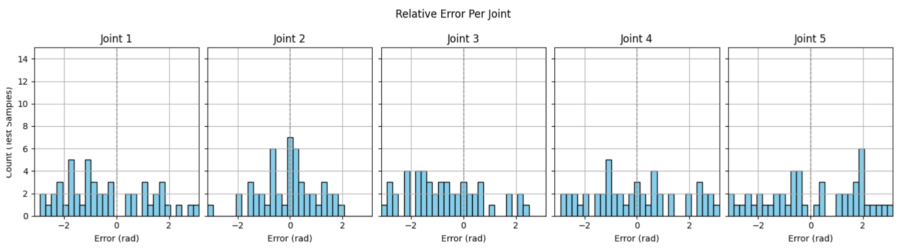
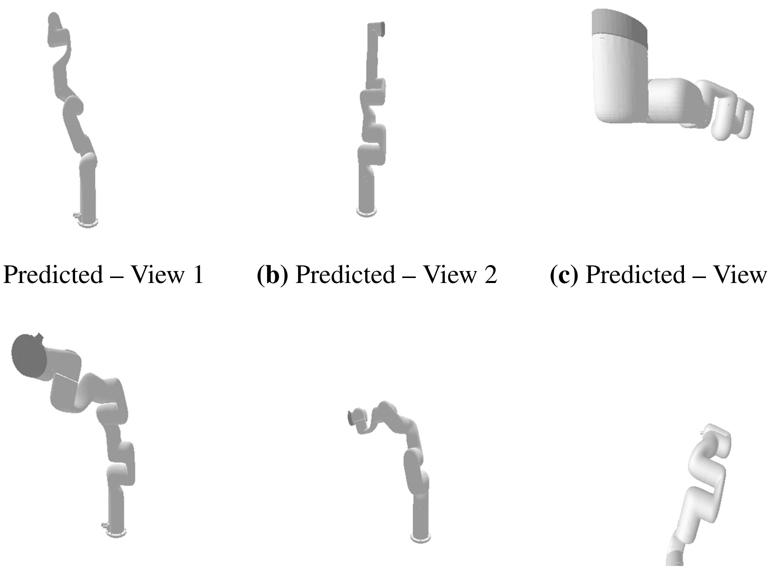


Figure 10: Relative error distributions per joint for the simulation-trained model evaluated on real images.

structure of the manipulator is recovered. However, noticeable discrepancies can be supported by actual errors per joint for this case as seen in **Table 2**. Joints 1-3 and 5 suffer relatively larger errors while joint 4 stays on the lower end yet still adding to the overall error of the results. This highlights the limitation of a zero-shot transfer method from simulation to real images: the domain gap remains persistent throughout the results preventing any consistent alignment.

Finally, **Figure 12** illustrates the displacement of the end-effector for this configuration. Although the end-effector does not play a direct role in state estimation, it provides a valuable validation technique. For the zero-shot transfer case, the distance between the ground-truth and predicted end-



(a) Predicted – View 1 **(b)** Predicted – View 2 **(c)** Predicted – View 3

(d) Ground Truth – View 1 **(e)** Ground Truth – View 2 **(f)** Ground Truth – View 3

Figure 11: Qualitative comparison of a real test configuration showing the predicted pose using the simulation model (first row) and the ground-truth (second row) from three views.

Table 2: Quantitative comparison on a per-joint basis of the same real test configuration, based on the simulation model.

	J1	J2	J3	J4	J5
Ground Truth (rad)	4.38	0.26	-1.75	1.22	0.45
Predicted (rad)	6.08	-0.49	-2.78	1.76	1.49
Error (rad)	**1.71**	**-0.75**	**-1.03**	**0.54**	**1.05**

effector is significantly large as seen by the distance line. This confirms that state estimation suffers due to domain gap.

Stage 3: Hybrid Model

Following the same strategy, the best performing simulation model was used for the hybrid approach.

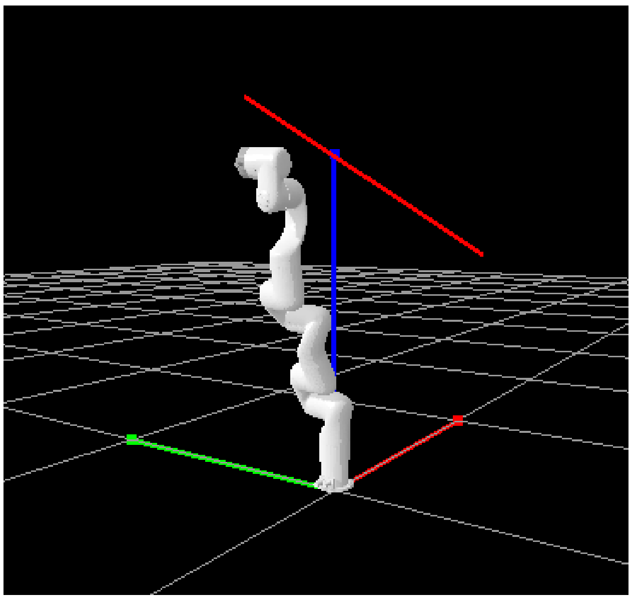


Figure 12: Qualitative comparison of end effector position where a red line marks the Euclidean distance between the real test image and the estimated configuration using the simulation model.

As shown in **Figure 13**, the mean absolute error per joint decreases overall compared to the zero-shot case. Joint 1, which exhibited the worst performance in the zero-shot case, sees a substantial reduction in error and is no longer the outlier. Joint 2 also improves considerably, maintaining its position as one of the best-performing joints but with noticeably lower error than in the zero-shot evaluation. Joint 3 remains less accurate than in the pure simulation setting, yet hybrid training reduces its error relative to the zero-shot model. Joint 4 shows little change compared to zero-shot and becomes the worst-performing joint in the hybrid evaluation. Joint 5, which carried the highest error in the zero-shot case, now drops below joint 4, reflecting a modest improvement. The relative error distributions in **Figure 14** further illustrate these trends. In the zero-shot transfer case, joint 1 exhibited an extremely dispersed distribution. Under hybrid training, its errors are now tightly centered around zero indicating a significant improvement in prediction stability. Joint 2's distribution, already among the narrowest, becomes even more compact, confirming it benefits strongly from hybrid training. Joint 3 still trails

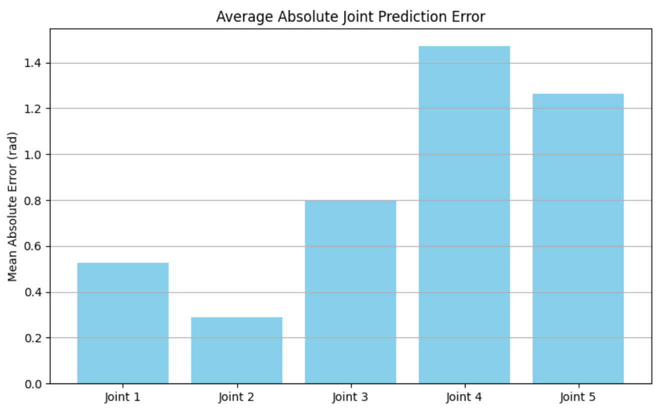


Figure 13: Mean absolute error (MAE) per joint for the hybrid-trained model on real-image test data.

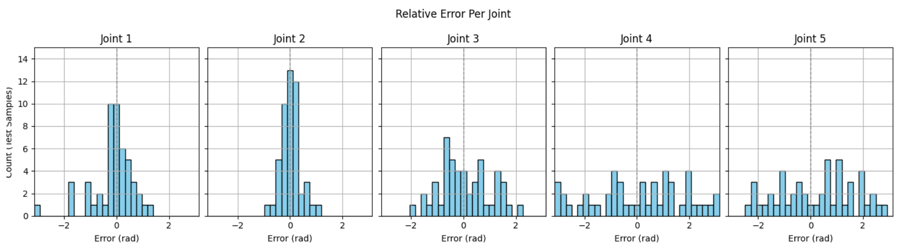
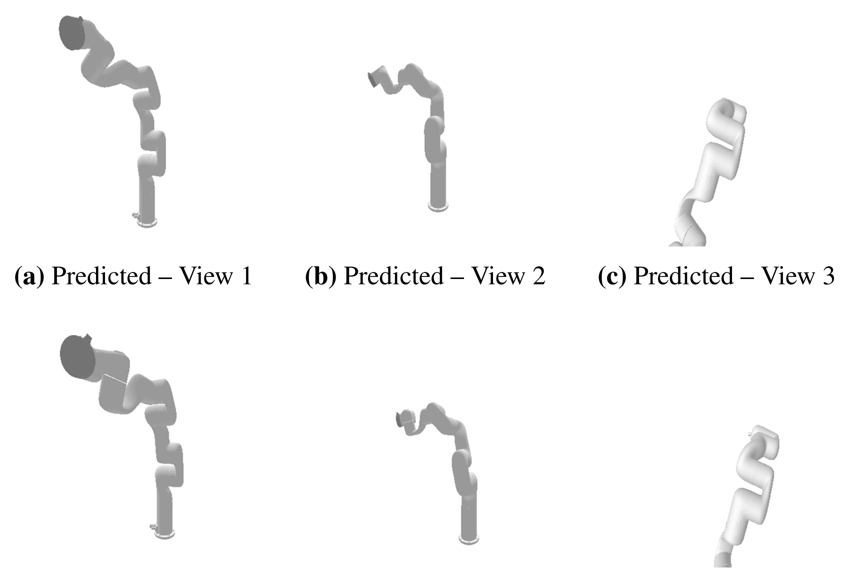


Figure 14: Relative error distributions per joint for the hybrid-trained model on real-image test data.

its performance in simulation-only testing but shows clear improvement compared to zero-shot, with a more concentrated spread around zero. In contrast, joint 4's distribution remains largely unchanged, and joint 5 continues to be the most error-prone, though with slightly reduced tails relative to the zero-shot case. Overall, the hybrid model reduces the broad dispersion seen in the zero-shot results and produces distributions that are more consistent across joints.

To provide a direct comparison with zero-shot evaluation, **Figure 15** shows the same real test configuration rendered in PyBullet, alongside **Table 3**, which summarizes the ground-truth, predicted, and error values for each joint. Visually , the hybrid model produces a pose that is a much closer replication



(**a**) Predicted – View 1 (**b**) Predicted – View 2 (**c**) Predicted – View 3

(**d**) Ground Truth – View 1 (**e**) Ground Truth – View 2 (**f**) Ground Truth – View 3

Figure 15: Qualitative comparison of a real test configuration showing the predicted pose using the hybrid model (first row) and the ground-truth (second row) from three views.

of the ground truth as seen in all orthogonal views. The per-joint errors in the table reinforce these visual results as they show strong numerical alignment in the manipulator joint angles. Together, the qualitative render and table confirm that hybrid training not only narrows the numerical error but also results in visibly improved alignment of the manipulator on real images.

Table 3: Quantitative comparison on a per-joint basis of the same real test configuration, based on the hybrid model.

	J1	J2	J3	J4	J5
Ground Truth (rad)	4.38	0.26	-1.75	1.22	0.45
Predicted (rad)	4.22	-0.03	-1.56	0.54	0.52
Error (rad)	**-0.16**	**-0.29**	**0.19**	**-0.68**	**0.07**

The final evaluation process was to compare the distance between the predicted end effector location and the ground truth. As can be seen in **Figure 16**, there is error present, noted by the red line. However, this relative distance is much less than when this same comparison was made using the simulation-only model. Therefore, the performance of the model is heightened when the variability of real-world data is introduced.

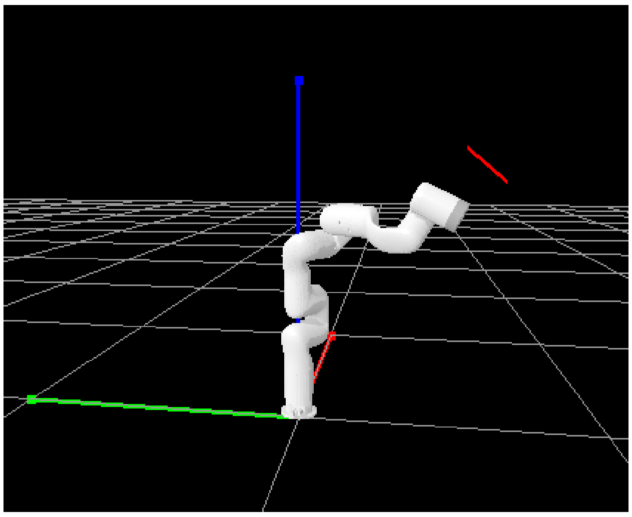


Figure 16: Qualitative comparison of end effector position where a red line marks the Euclidean distance between the real test image and the estimated configuration using the hybrid model.

Conclusion

We developed a vision-based deep learning approach for performing state estimation of robotic joint angles from multi-view RGB inputs. Our approach incorporates a neural network with a pretrained ResNet-18 backbone, which underwent a multi-stage training process. This involved first training the network on a large set of simulated data collected from a virtual 3D environment and then fine-tuning on a smaller set of real-life camera data. The network trained using this procedure outperformed models trained exclusively on either simulated or real-life data and demonstrated reasonable performance on a set of real-life test data not seen during model training. These results validate our proposed vision-based approach for robotic state estimation and suggest promising applications for additional configurations of serial manipulators beyond those examined in this work.

Future work: This methodology can be extended in numerous directions. Inclusion of the end-effector global position as a target label for the neural network could provide additional representation to the network to adapt and learn from. This strategy could also mitigate the issue of gaps in learning for shorter links across any serial manipulator that suffer subtle visual changes across configurations. Additionally, this could help the model implicitly understand manipulator kinematics. Utilizing this vision model as an feedback signal for a serial manipulator could be a potential direction to eliminate the need for encoders, a key motivation to this work.

Broader impacts: An important contribution of this work is a baseline for encoderless manipulator functionality, which has potential applications in radiation environments, space missions, and other instances where encoder feedback is not preferred. By reducing the reliance on joint encoders, building on this approach could provide a cost-effective solution for many challenges faced in the robotics space.

Acknowledgments

This research was funded by Los Alamos National Laboratory (LANL) through the Engineering Institute's Los Alamos Dynamics Summer School (LADSS). The Engineering Institute is a research and education collaboration between LANL and the University of California San Diego's Jacobs School of Engineering. This collaboration seeks to promote multidisciplinary engineering research that develops and integrates advanced predictive modeling, novel sensing systems, and new developments in information technology to address LANL mission-relevant problems.

We would also like to thank the whole of the LADSS program, and specifically Samantha Ceballes and Shannon Danforth for their excellent program organization, leadership, and mentorship.

References

[1] Park, W., Liu, Y., Zhou, Y., Moses, M., and Chirikjian, G.S. "Kinematic state estimation and motion planning for stochastic nonholonomic systems using the exponential map". *Robotica*, 26(4):419–434 (2008)ifnextchar.gobble.

[2] Talbot, W., Nubert, J., Tuna, T., Cadena, C., Dümbgen, F., Tordesillas, J., Barfoot, T.D., and Hutter, M. "Continuous-time state estimation methods in robotics: A survey" (2025)ifnextchar.gobble.

[3] Sandler, B.Z. “5 - feedback sensors”. In Sandler, B.Z., editor, *Robotics (Second Edition)*, pages 175–205. Academic Press, San Diego, second edition edition (1999)ifnextchar.gobble.

[4] Ceccarelli, M. *Fundamentals of mechanics of robotic manipulation*, volume 112. Springer (2022)ifnextchar.gobble.

[5] Reed, F.K., Ezell, N.D.B., Ericson, M.N., and Britton, C.L., Jr. “Radiation hardened electronics for reactor environments”. Technical report, Oak Ridge National Laboratory (ORNL), Oak Ridge, TN (United States) (2020)ifnextchar.gobble.

[6] Alizadeh, M. and Zhu, Z.H. “A comprehensive survey of space robotic manipulators for on-orbit servicing”. *Frontiers in Robotics and AI*, 11:1470950 (2024)ifnextchar.gobble.

[7] Han, M., Xie, B., Barczyk, M., and Bayat, A. “Image-based joint state estimation pipeline for sensorless manipulators”. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2158–2165 (2021)ifnextchar.gobble.

[8] Widmaier, F., Kappler, D., Schaal, S., and Bohg, J. “Robot arm pose estimation by pixel-wise regression of joint angles”. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 616–623 (2016)ifnextchar.gobble.

[9] Ioannidou, A., Chatzilari, E., Nikolopoulos, S., and Kompatsiaris, I. “Deep learning advances in computer vision with 3d data: A survey”. *ACM Comput. Surv.*, 50(2) (2017)ifnextchar.gobble.

[10] Shirai, Y. *Three-dimensional computer vision*. Springer Science & Business Media (2012)ifnextchar.gobble.

[11] O’ Mahony, N., Campbell, S., Krpalkova, L., Riordan, D., Walsh, J., Murphy, A., and Ryan, C. “Computer vision for 3d perception”. In Arai, K., Kapoor, S., and Bhatia, R., editors, *Intelligent Systems and Applications*, pages 788–804, Cham (2019)ifnextchar.gobble. Springer International Publishing.

[12] Kalaitzakis, M., Cain, B., Carroll, S., Ambrosi, A., Whitehead, C., and Vitzilaios, N. “Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag, aruco and stag markers”. *Journal of Intelligent & Robotic Systems*, 101(4):71 (2021)ifnextchar.gobble.

[13] Lepetit, V. and Fua, P. “Keypoint recognition using randomized trees”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479 (2006)ifnextchar.gobble.

[14] Zhang, Z., Hu, Y., Yu, G., and Dai, J. “Deeptag: A general framework for fiducial marker design and detection”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2931–2944 (2023)ifnextchar.gobble.