Utilizing Inpainting for Training Keypoint Detection Algorithms Towards Markerless Visual Servoing

Sreejani Chatterjee[1], Duc Doan [1], and Berk Calli[1]

***Abstract*— This paper presents a novel strategy to train keypoint detection models for robotics applications. Our goal is to develop methods that can robustly detect and track *natural* features on robotic manipulators. Such features can be used for vision-based control and pose estimation purposes, when placing artificial markers (e.g. ArUco) on the robot's body is not possible or practical in runtime. Prior methods require accurate camera calibration and robot kinematic models in order to label training images for the keypoint locations. In this paper, we remove these dependencies by utilizing inpainting methods: In the training phase, we attach ArUco markers along the robot's body and then label the keypoint locations as the center of those markers. We, then, use an inpainting method to reconstruct the parts of the robot occluded by the ArUco markers. As such, the markers are artificially removed from the training images, and labeled data is obtained to train markerless keypoint detection algorithms without the need for camera calibration or robot models. Using this approach, we trained a model for realtime keypoint detection and used the inferred keypoints as control features for an adaptive visual servoing scheme. We obtained successful control results with this fully model-free control strategy, utilizing natural robot features in the runtime and not requiring camera calibration or robot models in any stage of this process.**

I. Introduction

Robustly detecting natural features (keypoints) on a robot's body has many useful applications, from system calibration [1], [2], to robot pose estimation [3], [4] and vision-based control [5]–[7]. Training models to detect such keypoints, however, requires tedious manual image labeling processes that are prone to errors: the location of each natural keypoint on the robot needs to be marked precisely on thousands of robot images. This manual labeling process can be avoided if a calibrated camera, the robot's kinematic model, and its encoder readings are available during the data collection phase: the 3D locations of each keypoint can be calculated via forward kinematics using the robot model and the encoder readings, and these locations can be projected on the camera's image plane using camera intrinsic and extrinsic parameters. In the literature, such a strategy is utilized for training a camera calibration algorithm, the DREAM framework, presented in [8]. In our prior work [9], we have created a similar data collection pipeline: using a calibrated camera and the robot model, we calculated the 2D projections of a robot's joint locations, and used them for training keypoints in image space for vision-based robot control. Even though the labeled training data collected with our pipeline proved to be useful to achieve robust keypoint detection models [9], the data collection process require very accurate camera calibration (both intrisics and extrinsics) and precise robot kinematic models. Any small error in either of these aspects can cause erroneous labeling of the images, and consequently, inaccurate keypoint detection models.

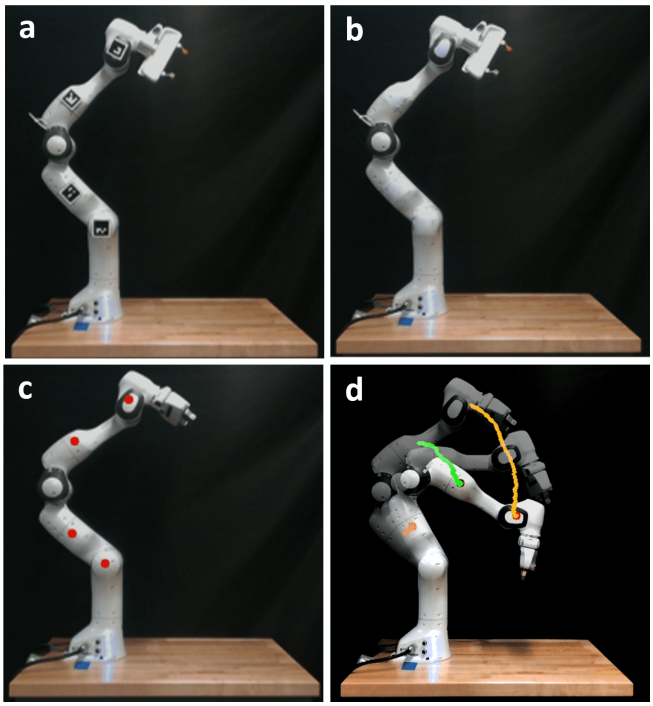

Fig. 1: The proposed method of utilizing the inpainting framework to generate visual features identifies keypoints on a robot's body in different configurations. a original image with ArUco, b image reconstructed with inpainting, c keypoints predicted on an arbitrary configuration using keypoint detector, d trajectory created by detected keypoints on different configuration of the arm in motion.

In this paper, we provide a keypoint training strategy that does not require camera calibration or robot models. In a nutshell, in the training phase, we put ArUco markers on the robot body to mark the locations of keypoints. However, we artificially remove these markers from the images using an inpainting algorithm (LaMa [10]) before feeding them to the keypoint training algorithm. As such, we know the exact locations of the keypoints thanks to the markers, but the keypoint detection model is trained with robot images

*This paper was supported in part by the National Science Foundation under grant IIS-1900953 and CMMI-1928506.

[1]S Chatterjee, [1]Duc Doan and [1]B Calli are with Department of Robotics Engineering, Worcester Polytechnic Institute, Worcester, MA 01609, USA schatterjee@wpi.edu, tvdoan@wpi.edu

"without markers" since they are artificially removed. We demonstrate that the resulting model is capable of robustly detecting keypoints on a robot in runtime. To demonstrate their utility, we further utilize these keypoints in an adaptive visual servoing algorithm to control the robot's configuration, and obtained successful results.

We would like to emphasize the practicality of our approach. The algorithm requires a set of images of the robot without markers and another set with markers. It is capable of training a model for realtime detection of natural features of the robot without requiring any manual labeling, camera calibration, robot model or encoder information. Coupled with an adaptive visual servoing algorithm, a truly model-free workflow is achieved for controlling the robot using natural visual cues. To the best of our knowledge, this whole strategy is novel in removing all the above mentioned constraints altogether.

We would also like to provide the scope and limitations of the work. We focus on detecting keypoints on a robot for planar motions. We will focus on detecting keypoints for out of plane motions in our future work. We use a uniform background for the robot in our experiments, and did not study the robustness of the algorithm to lighting conditions, different backgrounds, and occlusions. Nevertheless, we believe that this paper provides a unique strategy that can be expanded to more general conditions with our future work directions.

II. Related Work

Keypoint detection models are popular and frequently used to solve human pose estimation problem in the computer vision field. These kinds of algorithms efficiently pinpoint key human body parts such as hands, shoulders, eyes and nose in an image. There is a wealth of datasets annotated with body part keypoints for this purpose, including MPII Human Pose [11], COCO test-dev [12], and Densepose-COCO [13]. Additionally, frameworks like OpenPose [14] and DeepPose [15] offer specialized deep learning architectures for training pose detection models from keypoints. However, the existing literature lacks comprehensive datasets for keypoint detection on robotic manipulators. In [8], the authors create datasets of three rigid robotic manipulators (Panda, KUKA and Baxter) in simulation. These datasets are then used in a deep learning framework (also known as DREAM) to detect keypoints on real robots. The primary motivation for this work is to calibrate the camera extrinsics by predicting keypoints on physical robot's body. This framework does a phenomenal job in providing a one-shot calibration tool and can detect keypoints for many different static robot poses. However, the keypoint detection is not robust enough to detect keypoints continuously, which is a requirement for various applications, e.g., vision-based control and online pose detection. This unreliability of continuous detection of keypoints in a real-time setting is attributed to the fact that these algorithms are trained in simulated datasets, due to the lack of labeled real robot datasets in the literature.

In [16], a deep learning model is used to detect optimal keypoints on a robot's body again using a simulated dataset. Even though this work is further generalized and is applicable to various range of robots, it too suffers from the sim-to-real gap. Also in this paper the keypoints are detected optimally for each configuration, and does not guarantee detection in the same part of the robot in every pose or same number of keypoints. This is problematic to use in control applications as we calculate the control error between the previous and the current positions of the same keypoints in different configurations in image space.

To overcome the lack of labelled real robot data for training keypoint detection algorithms, we developed an auto-labeled data collection pipeline [9]. However, this approach requires accurate camera calibration and robot models, limiting the practicality of the approach. Moreover, it is unfeasible to utilize that approach with soft or underactuated robots, where accurate robot models are often very challenging to obtain. While in this work we only focus on rigid robots and argue the practicality of not needing a robot model or camera calibration but simply a set of images of the robot, we believe that the presented approach is a milestone towards enabling the training of keypoint detection models for soft robots in the future.

III. Methodology

The main focus of this paper is to develop a practical approach to train algorithms for detecting keypoints on the robot's body. For this purpose we utilize a state-of-the-art inpainting algorithm. The training of this algorithm is described in Section III-A.1. This algorithm is utilized to artificially remove ArUco markers from the training images of the keypoint detection model as explained in Section III-A.2. We then formed a dataset as described in Section III-A.3. The overall data collection pipeline can be described by the flowchart depicted in Fig. 2. We, then, trained a keypoint detection model as explained in Section III-B.

A. Data Collection Pipeline:

We create an automated pipeline, where we capture several images of the robot arm in arbitrary configurations with Aruco markers on designated spots and compute the keypoint location for each configuration using the steps described in the following subsections.

1) Training Phase of Inpainting Algorithm: In this section, we train a deep learning network for inpainting. Inpainting [17], in the context of computer vision and image processing, refers to the process of reconstructing lost or corrupted parts of an image. It is popular in the fields of digital image restoration and video editing. We considered two state-of-the-art inpainting methods: Generative Adversarial Networks (GANs) of CoModGAN [18] and LaMa [10]. CoModGAN [18] employs the co-modulation method to bring the strong generative capability of unconditional generators to image-to-image translation problems like inpainting. LaMa [10] relies on Fast Fourier Convolution, which works

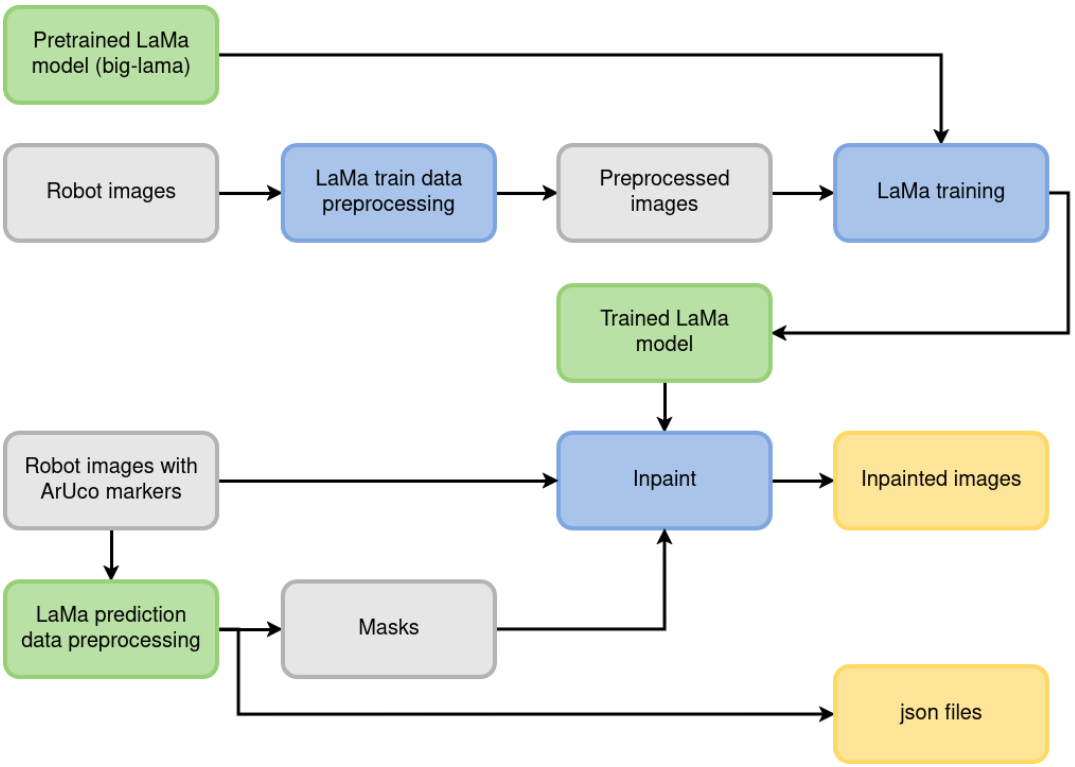

Fig. 2: Overall data collection pipeline. Green blocks denote the LaMa models, gray blocks denote the data (images), blue blocks denote the actions, and orange blocks denote the outputs of the pipeline.

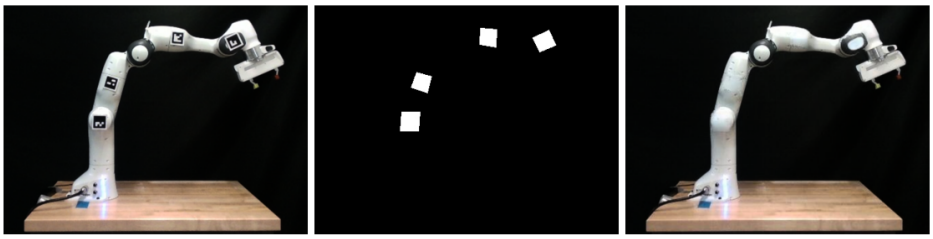

Fig. 3: These three figures depict the input for LaMa prediction and the reconstructed image as an output

well with complex structures and can scale to higher resolutions than the images used during training. We eventually chose to use LaMa for the following reasons. LaMa [10] surpasses the majority of baseline models, including state-of-the-art solutions like Deepfill v2 [19], EdgeConnect [20], HiFill [21], MADF [22] and CoModGAN [18], according to evaluations using the Learned Perceptual Image Patch similarity (LPIPS) and Frechet Inception Distance (FID) metrics. These metrics were applied to assess performance across various test mask generation strategies, such as narrow, wide, and segmentation masks. It is robust to image resolution and highly customizable via well-documented configuration files and provides great flexibility for inpainting variable data.

In their GitHub repository, LaMa [10] provides several pretrained models, but as they were not trained with robot images, they do not produce accurate enough results for our use case. To increase accuracy, we fine tune the best pretrained model named as 'big-lama', in the repository by further training it with robot images. For that purpose, we collect a significant number of images of the manipulator with no markers attached to it. We preprocess the collected images following the instructions in [10]. An important aspect of preprocessing the data is to crop it to a suitable size. For our purpose, we cropped the images to 480×480. During training run time, by default, LaMa crops the images further to a size of 256×256, which can be customized by changing specific parameters inside the LaMa repository. We changed it to 480×480 for our datasets. For further details on data preprocrocessing, refer to our data collection page at https://github.com/JaniC-WPI/KPDataGenerator. We train the customized model for 80 epochs of batch size 30.

2) Reconstruction of Images using LaMa: We collect another set of images of the manipulator with ArUco markers, placed in close proximity to the visible locations of some of the joints as shown in the first image of Fig. 3. We would like to emphasize that, in our proposed method, the user has

the flexibility to place the markers anywhere according to their individual purposes. Once the markers are placed in the desired locations on the manipulator and the images are collected, we automatically detect the marker locations in each image and create a binary mask around the markers. The RGB images and their corresponding masks are then inpainted by the newly trained LaMa model, with refinement enabled to improve the inpainting quality as described in [23]. Fig. 3 shows an example where the first image is the original image with the markers, the second image is where the markers are masked and the third image is the prediction of the LaMa trained model, which removes the ArUco markers using the inpainting method. The centers of the ArUco markers in the original images are our identified keypoint locations.

3) Dataset Generation: We save the center coordinates of the ArUco markers on all the images collected in Section III-A.2 in JSON files as keypoints annotations, together with the corresponding reconstructed images. Along with the keypoints, we also compute a bounding box with each keypoint as the center and save those in the same JSON file. The 'keypoints' field in the JSON file is of the format $[x,\ y,\ v]$, where v is the visibility of the keypoint and x and y are the pixel coordinates of the keypoint in image space. The 'bboxes' (bounding boxes) field has the format $[x-(bb\_size/2),\ y-(bb\_size/2),\ x+(bb\_size/2),\ y+(bb\_size/2)]$, where bb_size is the size of each side of the bounding box and x and y are the pixel coordinates of the keypoints as mentioned above. In our case, we set bb_size to 40 for the Franka Emika Panda arm. This constitutes the training dataset for our customized keypoint detector network described in Section III-B. We have 4 keypoints for our Panda robot along with their corresponding bounding boxes. Please refer to Fig. 4 for the corresponding image with labels on keypoints for the following sample JSON file, created from the data collected with the Panda arm using the above pipeline. Here, 'id' refers to the specific number of image in the dataset, 'image_rgb' is the name of the corresponding image. 'bboxes' are the list of bounding boxes as computed above for each keypoint and 'keypoints' are the list of keypoints for each joint:


```
{
    "id": 4136,
    "image_rgb": "004136.jpg",
    "bboxes": [
        [252.99999999999994, 294.49999999999994,
            292.99999999999994, 334.49999999999994],
        [269.9344179320318, 237.2853940708605,
            309.9344179320318, 277.2853940708605],
        [368.12763532763523, 171.1216524216524,
            408.12763532763523, 211.1216524216524],
        [450.60434782608684, 176.3014492753623,
            490.60434782608684, 216.3014492753623]],
    "keypoints": [
        [[272.99999999999994, 314.49999999999994, 1]],
        [[289.9344179320318, 257.2853940708605,  1]],
        [[388.12763532763523, 191.1216524216524, 1 ]],
        [[470.60434782608684, 196.3014492753623, 1 ]]
    ]
}
```


We obtain 4 keypoints using the above process as depicted in Fig. 4.

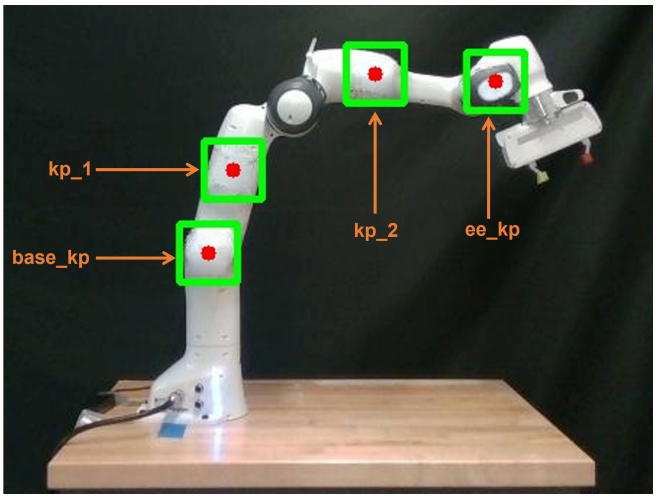

Fig. 4: Sample image from the dataset collection pipeline labeled with the keypoints, bounding boxes and respective keypoint names.

B. Network Architecture and Keypoint Prediction:

We use Pytorch vision library's [24] *keypointrcnn_resnet50_fpn* to train our dataset. The *keypointrcnn_resnet50_fpn*, which is pretrained to detect 17 keypoints in human body, is an extension of the Mask-RCNN [25] deep learning model and is state of the art for keypoint detection for human body pose estimation. We customized this model for the datasets that we created with 4 keypoints for the Panda robot. We label each keypoint/bounding box from 1 to 4, (1 being *base_kp* and 4 being *ee_kp* in Fig. 4) in the custom Pytorch Dataset class that we created for our data. The *keypointrcnn_resnet50_fpn* uses the backbone of Resnet50 with pre-trained weights of IMAGENET1K_V2 from the COCO Dataset [12]. We implemented data augmentation on our training dataset to enhance its diversity. We refined our model on our dataset employing Stochastic Gradient Descent, setting the learning rate at 0.0001 and the momentum at 0.9. Additionally, we incorporated a weight decay of 0.0005 to mitigate overfitting. We used a batch size of 4 images per batch and ran the training for 30 epochs. The evaluation and prediction process of keypoints are exactly the same as our prior work [9].

IV. Experiments and Results

In this section, we analyze the accuracy of the identification of keypoints on the Franka Emika Panda robot. We also perform control experiments using an adaptive visual servoing scheme as described in [26]. We demonstrate the noiseless and robust visual features created using our proposed method of keypoints generation, which are used to control a robotic manipulator.

A. Feature Detection Accuracy:

We take images of 20 different configurations with the Panda arm, covering most of its planar workspace. The

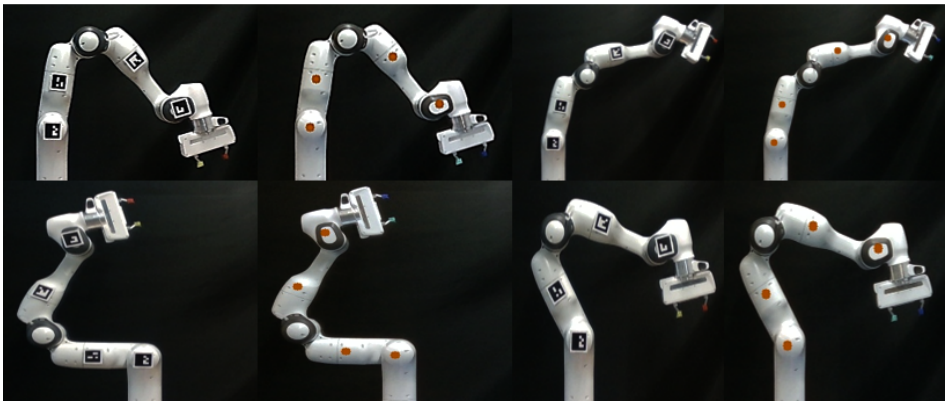

Fig. 5: The images with ArUco markers are the ground truth and the red dot on the corresponding image is the keypoint predicted by the Keypoint RCNN model trained with data generated using our present method.

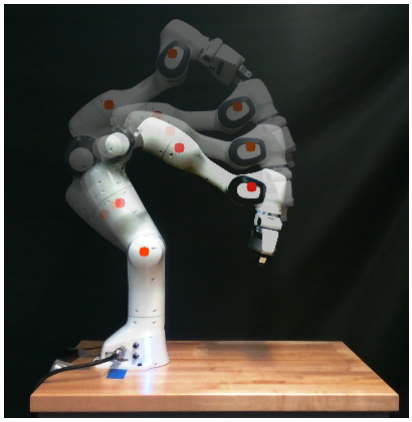

Fig. 6: Keypoint detection in continuous robot motion.

images are taken with and without markers for the same configurations. The images with the markers hold the ground truth of the keypoints' positions. We then run the images with the *keypointrcnn-resnet-50fpn* model to predict the keypoints on the robot's markerless body. We obtained 100% detection rate for detecting the features with less than 10 pixel accuracy, and 98.75% for less than 5 pixel accuracy. The average accuracy of detecting the keypoints is 2.19 pixels. The qualitative analysis of the feature accuracy is shown in Fig 5. Fig. 6 depicts the continuous detection of keypoints while the robot is in motion.

B. Transient Response

In order to assess the utility of our proposed approach, we conducted 20 vision-based control experiments by utilizing the trained keypoint detection model in real time. We used exactly the same reference configurations and adaptive visual servoing scheme as described in our prior work [9] (which requires camera calibration, robot model, and encoder readings to train the keypoint detection model) so that we can compare the control performance. We tuned the control gains for the system to the lowest rise time and also made sure the overshoot remained within 5%. We used an Intel RealSense D435i camera. Table I depicts that the visual features generated with our current method perform as reliably and efficiently as the prior keypoints based method. Fig. 8 shows the error norm and the individual feature error plot for the experiment depicted in Target 1 of Fig. 7. Please note that while the reference configurations are the same between the two experiments, the feature locations are different, since our prior work is trained to detect the joint centers, while the current work detects the locations where the markers were placed in the training phase.

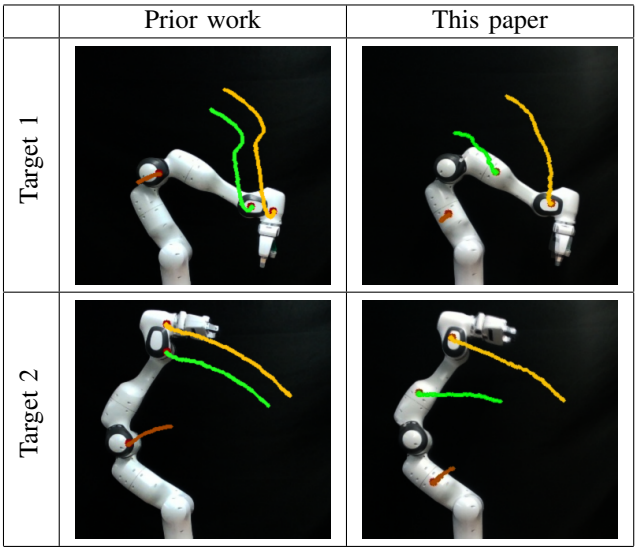

Fig. 7: Trajectory comparison between proposed and existing method's keypoints. The noiseless quality of the trajectories testify that the proposed method is almost equally robust as the prior method.

C. Repeatability:

We conducted a repeatability test for our proposed method by performing a control experiment five times using the same initial and reference positions. The outcomes are detailed in Table II. A review of the results indicates only slight variations in the rise time, settling time, and overshoot values

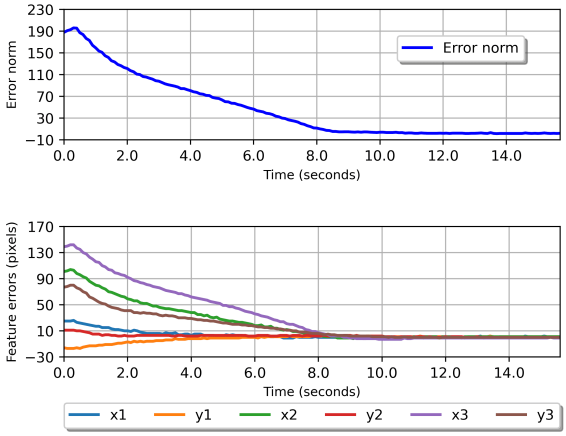

Fig. 8: Image feature error norm (top) and individual image feature errors (below).

TABLE I: Performance Comparison Summary

	Prior Method	Proposed algorithm
System Rise time (s)	6.16 ± 2.10	5.85 ± 2.9
System Settling time (s)	11.29 ± 2.42	8.31 ± 2.9
End effector Rise time (s)	5.37 ± 1.66	5.435 ± 2.65
End effector Settling time (s)	10.05 ± 3.48	7.72 ± 3.1
Overshoot (%)	2.9 ± 2.73	3.43 ± 3.28

across the five experiments.

V. Conclusion and Future Work

In this paper, we successfully generated a model that detects purely natural features on the robot's body by leveraging an inpainting method. We used ArUco markers to designate locations on the robot's body for identifying keypoints or features, and then artificially removed the markers using a deep learning inpainting framework. By doing so, we completely eliminated the need of camera calibration as well as any prior knowledge of the kinematic model of the robot. We created and open sourced the data collection pipeline using our proposed method to build large datasets for different robotic manipulators. We trained Keypoint RCNN deep learning models using these datasets. We were able to control the robot in its 2D configuration space by utilizing the keypoints inferred from the Keypoint RCNN models. Using our method, we collected a dataset for the Franka Emika Panda robot. We obtained highly accurate keypoint detection results in realtime. The keypoints generated using our proposed method were used as natural features in the image space to control the robot using an adaptive visual servoing scheme. We compared the performance of our proposed method with our prior work and we can conclusively claim that the keypoints generated using the proposed method is equally as robust and reliable as our prior method of keypoints

TABLE II: Summary of Repeatability Tests

	Proposed algorithm
System Rise time (s)	6.82 ± 0.11
System Settling time (s)	9.12 ± 0.23
End effector Rise time (s)	6.82 ± 0.11
End effector Settling time (s)	9.0 ± 0.1
Overshoot (%)	1.096 ± 0.27

generation. Additionally, to collect data using our pipeline, users have the freedom to place the markers according to their convenience, depending on the purpose at hand. In this paper, we have only showcased results of tracking keypoints for planar motion of the robotic arm. In the future, we aim to create data to control the manipulators moving out of plane. We further plan to investigate the robustness of the pipeline with cluttered background. We also aim to expand this work in identifying natural features on the body of continuum and non-rigid manipulators.

References

[1] B. Pätzold, S. Bultmann, and S. Behnke, "Online marker-free extrinsic camera calibration using person keypoint detections," in *DAGM German Conference on Pattern Recognition*. Springer, 2022, pp. 300–316.

[2] J. Lambrecht, P. Grosenick, and M. Meusel, "Optimizing keypoint-based single-shot camera-to-robot pose estimation through shape segmentation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 843–13 849.

[3] J. Bohg, J. Romero, A. Herzog, and S. Schaal, "Robot arm pose estimation through pixel-wise part classification," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3143–3150.

[4] C.-J. Liang, K. M. Lundeen, W. McGee, C. C. Menassa, S. Lee, and V. R. Kamat, "A vision-based marker-less pose estimation system for articulated construction robots," *Automation in Construction*, vol. 104, pp. 80–94, 2019.

[5] F. Chaumette, S. Hutchinson, and P. Corke, *Springer Handbook Of Robotics*. Springer, 2016, vol. Springer.

[6] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996.

[7] M. Jagersand, O. Fuentes, and R. Nelson, "Experimental evaluation of uncalibrated visual servoing for precision manipulation," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 4, pp. 2874–2880, 1997.

[8] 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 (ICRA)*. IEEE, 2020, pp. 9426–9432.

[9] S. Chatterjee, A. C. Karade, A. Gandhi, and B. Calli, "Keypoints-based adaptive visual servoing for control of robotic manipulators in configuration space," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 6387–6394.

[10] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. Lempitsky, "Resolution-robust large mask inpainting with fourier convolutions," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2022, pp. 2149–2159.

[11] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, 2014, pp. 3686–3693.

[12] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[13] R. A. Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7297–7306.

[14] Z. Cao, G. Hidalgo, T. Simon, S. E. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, pp. 172–186, 1 2021.

[15] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1653–1660.

[16] J. Lu, F. Richter, and M. C. Yip, "Pose estimation for robot manipulators via keypoint optimization and sim-to-real transfer," *IEEE Robotics and Automation Letters*, vol. 7, pp. 4622–4629, 4 2022.

[17] A. Romero, A. Castillo, J. Abril-Nova, R. Timofte, R. Das, S. Hira, Z. Pan, M. Zhang, B. Li, D. He, T. Lin, F. Li, C. Wu, X. Liu, X. Wang, Y. Yu, J. Yang, R. Li, Y. Zhao, Z. Guo, B. Fan, X. Li, R. Zhang, Z. Lu, J. Huang, G. Wu, J. Jiang, J. Cai, C. Li, X. Tao, Y.-W. Tai, X. Zhou, and H. Huang, "Ntire 2022 image inpainting challenge: Report," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2022, pp. 1150–1182.

[18] S. Zhao, J. Cui, Y. Sheng, Y. Dong, X. Liang, E. I. Chang, and Y. Xu, "Large scale image completion via co-modulated generative adversarial networks," *arXiv preprint arXiv:2103.10428*, 2021.

[19] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 4471–4480.

[20] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi, "Edgeconnect: Generative image inpainting with adversarial edge learning," *arXiv preprint arXiv:1901.00212*, 2019.

[21] Z. Yi, Q. Tang, S. Azizi, D. Jang, and Z. Xu, "Contextual residual aggregation for ultra high-resolution image inpainting," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7508–7517.

[22] M. Zhu, D. He, X. Li, C. Li, F. Li, X. Liu, E. Ding, and Z. Zhang, "Image inpainting by end-to-end cascaded refinement with mask awareness," *IEEE Transactions on Image Processing*, vol. 30, pp. 4855–4866, 2021.

[23] P. Kulshreshtha, B. Pugh, and S. Jiddi, "Feature refinement to improve high resolution image inpainting," *arXiv preprint arXiv:2206.13644*, 2022.

[24] V. Ayyadevara and Y. Reddy, *Modern Computer Vision with PyTorch: Explore deep learning concepts and implement over 50 real-world image applications*. Packt Publishing, 2020.

[25] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[26] A. Gandhi, S. Chatterjee, and B. Calli, "Skeleton-based adaptive visual servoing for control of robotic manipulators in configuration space," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2022, pp. 2182–2189.