Learning State Representations of Articulated Robot Arms from Visual Observations

Yunfu Deng
University of Wisconsin Madison
yunfu.deng@wisc.edu

Daniel Nikovski
Mitsubishi Electric Research Labs
nikovski@merl.com

***Abstract*—The paper presents a method for learning state representations of articulated mechanisms consisting of multiple links connected by joints but not equipped with positional encoders, from short sequences of visual observations of the mechanism collected by stationary RGB-D cameras. The method leverages recent advances in algorithms based on deep neural networks for long-term keypoint tracking in image sequences to extract tracks of matching keypoints over time, even when some of them are temporarily occluded. Because keypoint tracking is not perfectly reliable, we propose a novel method for robust clustering to discover how many rigid bodies (links) the mechanism consists of by analyzing which points move together and thus must belong to the same rigid body. Experiments with recently proposed long-term keypoint trackers demonstrate successful state estimation of the kinematic structure of articulated mechanisms entirely from camera images, effectively allowing computation of inverse kinematics and visual servocontrol for mechanisms such as robot arms, cranes, etc. even without positional encoders.**

***Index Terms*—Learning control, visual servocontrol, configuration learning**

I. Introduction

The field of robotics is undergoing a generational shift form robots that execute repeatedly the exact same motion pattern over and over again, to advanced flexible devices that adjust their operation based on rich sensory input such as images of the robot work cell obtained through cameras, as well as rich tactile information encoding contacts with manipulated parts. The interpretation of such very high-dimensional sensor observations, while opening new possibilities for advanced automation, has proven to be very laborious if done by means of manual programming. Machine learning technology holds high promise to automate the perception pipeline of robotic systems by acquiring end-to-end visuomotor control policies.

Recent advances in deep reinforcement learning (DRL) show that policies can, in principle, be trained end-to-end using large neural networks that map high-dimensional sensory inputs (e.g., images) directly to control signals [1]. In practice, however, such training typically requires millions of trials and is feasible mainly in simulation, which itself demands building highly accurate and labor-intensive environment models.

Humans and animals instead appear to learn visuomotor skills more modularly: first developing structured interpretations of the world (e.g., objects and their motion), then using these representations to learn task-specific control. Inspired by this, representation learning aims to derive compact scene representations and base control policies on them rather than on raw sensory input. Ideally, these representations correspond closely to the true dynamical state. For example, a robot arm with n degrees of freedom is most compactly represented by its n joint positions and n joint velocities, which together form the standard control state.

Numerous approaches to state representation learning (SRL) have been explored in recent years [2]. A common strategy adapts general representation learning techniques, such as autoencoders, which compress high-dimensional inputs into low-dimensional features via a bottleneck layer and reconstruct the input from this representation.

Unlike image classification – where position invariance is desirable – SRL requires preserving spatial information, motivating architectures such as Spatial Autoencoders (SAE) [3], [4]. Some methods further integrate ideas from nonlinear system identification, jointly learning state representations and system dynamics as nonlinear state-space models [5]. While effective for systems with few degrees of freedom (DoF), these approaches scale poorly as DoF increase, due to rising sample complexity and the combinatorial growth of possible configurations, which makes exhaustive sampling computationally prohibitive.

In SRL for articulated mechanisms (e.g., robots or cranes), strong inductive bias can be exploited: the system state decomposes into the states of individual rigid bodies, so sensory observations can likewise be decomposed into visual elements corresponding to those bodies [6]. Following this approach, the Object-Based Inverse Kinematics (OBIK) algorithm learns a mapping from camera images of an articulated mechanism to its configuration vector [7], [8]. It detects 3D keypoints on links, clusters them into rigid bodies via statistical testing, estimates each link's 6D pose relative to a reference image, and infers the kinematic chain by analyzing pairwise rigid body transforms during motion. Because it analyzes rigid-body motion independently, OBIK avoids the combinatorial explosion typical of many SRL

methods: any observed relative motion between bodies enables the algorithm to distinguish them and determine their order in the kinematic chain.

The empirical evaluation of OBIK assumed access to accurate keypoint tracks over the training sequence (whenever visible), obtained from the MuJoCo physics engine [9]. In real-world settings, however, keypoints must be detected and tracked across image sequences, handling occlusions through re-identification. This paper examines how practical keypoint tracking can be integrated with the OBIK algorithm. Its main contribution is the introduction into OBIK of a robust clustering algorithm for mechanism link inference that can tolerate the less-than-perfect capabilities of keypoint trackers, illustrated with an empirical verification.

Section II describes the problem the OBIK algorithm is solving and Section III summarizes the algorithm briefly. Section IV reviews methods for long-term keypoint tracking in sequences of images under occlusion and Section V describes the methods we use in the empirical investigation, itself presented in Section VII. A novel method for robust clustering of keypoint tracks is described in VI. Section VIII proposes directions for future work and concludes the paper.

II. Problem Statement

Given a sequence I_k, $k = 1, \ldots, T$, of T images of an articulated mechanism in motion, collected by a stationary camera, we want to learn a compact representation $\hat{\boldsymbol{\theta}}_k$ of the configuration of the mechanism that is functionally equivalent (for the purposes of controlling the mechanism) to the true configuration $\boldsymbol{\theta}_k$ of the mechanism consisting of all its joint positions at a given time, and a mapping/function $\hat{\boldsymbol{\theta}} = f(I)$ that will allow us to compute the mechanism's configuration given a novel image I of the mechanism at a later time, when controlling it. Here, "functionally equivalent" means that the constructed configuration $\hat{\boldsymbol{\theta}}_k$ should have a one-to-one (and preferably, linear) mapping to the true configuration $\boldsymbol{\theta}_k$. The learned mapping $\hat{\boldsymbol{\theta}} = f(I)$ essentially constitutes a form of inverse kinematics function for the mechanism, but unlike traditional inverse kinematics used in robotics control, which takes as input the Cartesian pose of the end effector of the mechanism, this form takes as input an observation (image) of the mechanism. If the function can be learned, it would allow the application of Pose-Based Visual Servocontrol (PBVS) for mechanisms with unknown forward and inverse kinematics, significantly reducing deployment time and costs of control systems for such mechanisms [10]. Applications include control of industrial manipulators, gantry cranes, 3D printers, etc. Arguably, the human brain must be using a similar learning mechanism to learn the forward and inverse kinematics of the human body and limbs, for which analytical models are not available at birth, and moreover are undergoing continuous change as the body grows over its lifetime. Our objective is to emulate this function using machine learning methods.

Our general approach is to extract sequences of reliably identifiable 3D keypoints from the sequence of images I_k and use these sequences, often called tracks in the field of computer vision, to determine how many rigid bodies exist in the scene, which keypoints belong to which rigid body, which pairs of bodies are connected by joints, what is the order of the kinematic chain of the mechanism, and what are the positions of all joints given a set of measurements of the 3D keypoints. To this end, we consider a set S of N distinctive 3D keypoints that are tracked over T time steps, resulting in measurements of the form $\mathbf{p}_{ik} = [x_{ik}, y_{ik}, z_{ik}]^T$ for $i = 1, \ldots, N$ and $k = 1, \ldots, T$. The next section summarizes the OBIK algorithm proposed previously that processes the set of keypoint tracks, and Section IV discusses methods for extracting the keypoints from images.

III. Observation-Based Inverse Kinematics

The OBIK algorithm assumes that the inertial (world) frame coincides with the camera frame, so the measurements $\mathbf{p}_{ik} = [x_{ik}, y_{ik}, z_{ik}]^T$ for $i = 1, \ldots, N$ and $k = 1, \ldots, T$ are expressed in that frame. The algorithm uses the following steps, and for further details, we refer the reader to [7], [8]:

- **Hypothesis testing whether pairs of keypoints belong to the same rigid body:** The 3D Euclidian distances d_{ijk} between all pairs of keypoints i and j are computed for all images k where both keypoints are visible. The mean distance between pairs of keypoints and its standard deviation are computed. A statistical F-test is performed to compute whether the standard deviation is significantly higher than the measurement noise, resulting in pairwise statistics q_{ij}.
- **Assignment of keypoints to rigid bodies:** The statistics q_{ij} are used as dissimilarity metrics by an agglomerative clustering algorithm with complete linkage to determine how many distinct rigid bodies $\hat{n}$ exist in the scene and which keypoints belong to which of the bodies. This step leverages the fact that keypoints belonging to the same body preserve the pairwise 3D distance between them when moving in space together.
- **Estimation of rigid body poses:** Given a reference frame with keypoints measurements in it, and a current image frame with corresponding measurements at time k, the relative pose between the current and reference frame of rigid body l, $l = 1, \ldots, \hat{n}$ is computed by means of the Kabsch-Umeyama algorithm [11], as long as at least 3 (non-collinear) keypoints assigned to body l are visible. The pose is expressed as a rigid body transform (RBT) in the world frame, defined by the rotation matrix $\mathbf{R}_{lk}$ and translation vector $\mathbf{t}_{lk}$.
- **Relative pose computation:** Using the poses in the world frame, the relative pose between all pairs of bodies l and m is computed as ${}^l\mathbf{R}_{mk} = \mathbf{R}_{lk}^T\mathbf{R}_{mk}$ and ${}^l\mathbf{t}_{mk} = \mathbf{R}_{lk}^T(\mathbf{t}_{mk} - \mathbf{t}_{lk})$, where a leading superscript l indicates the pose is expressed in the frame of body l.

- **Computation of translational relative DOF:** For a pair of bodies l and m, their relative translations ${}^l\mathbf{t}_{mk}$ are stacked in the $3 \times T$ matrix $\mathcal{T}_{lm} = [{}^l\mathbf{t}_{m1}\ {}^l\mathbf{t}_{m2}\ \ldots\ {}^l\mathbf{t}_{mT}]$ and principal component analysis (PCA) is performed on $\mathcal{T}_{lm}$ by singular value decomposition (SVD). The number of non-zero singular values is equal to the translational DoF between the two bodies.
- **Computation of rotational relative DOF:** For a pair of bodies l and m, their relative rotations ${}^l\mathbf{R}_{mk}$ are first expressed in axis-angle representation ${}^l\mathbf{r}_{mk}$, the latter are stacked in the $3 \times T$ matrix $\mathcal{R}_{lm} = [{}^l\mathbf{r}_{m1}\ {}^l\mathbf{r}_{m2}\ \ldots\ {}^l\mathbf{r}_{mT}]$, and PCA is performed on $\mathcal{R}_{lm}$ by means of SVD. The number of non-zero singular values is equal to the rotational DoF between the two bodies.
- **Computation of instantaneous centers of rotation:** For a pair of bodies l and m, if their rotation at time k is not the identity matrix, the instantaneous center of rotation ${}^l\mathbf{t}'_{mk}$ at that time is computed by solving the equation ${}^l\mathbf{t}_{mk} = \mathbf{A}\,{}^l\mathbf{t}'_{mk}$, where $\mathbf{A} = {}^l\mathbf{R}_{mk} - \mathbf{I}_3$, and $\mathbf{I}_3$ is the 3D identity matrix. The instantaneous centers of rotation are stacked in the matrix $\hat{\mathcal{T}}'_{lm} = [{}^l\hat{\mathbf{t}}'_{m1} - \boldsymbol{\tau}\quad {}^l\hat{\mathbf{t}}'_{m2} - \boldsymbol{\tau}\quad \ldots\quad {}^l\hat{\mathbf{t}}'_{mT} - \boldsymbol{\tau}]$, where each column represents the direction of a rotation center estimate relative to a reference point $\boldsymbol{\tau} = {}^l\hat{\mathbf{t}}'_{ma}$, chosen arbitrarily for some a such that $1 \leq a \leq T$. PCA on $\hat{\mathcal{T}}'_{lm}$ reveals if the instantaneous centers of rotation lie on a single straight line (when this matrix has only one non-zero singular value).
- **Discovery of single-DOF joints connecting pairs of bodies:** If neither translational nor rotational DoF exist between bodies l and m, they do not move with respect to each other. If no rotational DoF are present between bodies l and m, and exactly one translational DoF exists between them, there is a prismatic joint between them. If exactly one rotational DoF exists between the two bodies, and their instantaneous centers of rotation lie on a straight line, there is a revolute joint between them. Otherwise, no single-DoF joint exists between the two bodies.
- **Discovery of the order of the kinematic chain of the mechanism:** First, bodies that do not move with respect to the world frame are identified as stationary. Then, if a body is connected to these bodies with a single joint, it must be the first link of the mechanism. Then, a body connected to the first link by a joint must be the second link, etc., until the last link of the mechanism is identified.
- **Joint position estimation** For a revolute joint connecting a parent link l and child link m, the corresponding joint angle is computed from ${}^l\mathbf{R}_{mk}$ as the rotation angle of the matrix. If the joint is prismatic, the joint displacement is computed from ${}^l\mathbf{t}_{mk}$ as the signed length of the vector.

IV. Long-Term Keypoint Tracking

The OBIK algorithm relies critically on the ability to measure the 3D positions $\mathbf{p}_{ik} = [x_{ik}, y_{ik}, z_{ik}]^T$ of N keypoints over a sequence of T images, subject to inevitable occlusions. This means that the tracker should be able to extract N distinctive points from a training sequence and reliably establish correspondence between them over its entire span. This problem is known as long-term keypoint tracking in the computer vision community and has been the subject of intense study over the last decades, as it is an essential block in many computer vision problems [12].

Early keypoint tracking methods detected distinctive image patches (e.g., Harris or Shi–Tomasi corners) and tracked them across frames using sparse optical flow [12]. However, tracking fails under occlusion, requiring explicit re-identification. Moreover, this approach is ill-suited for OBIK, which must match keypoints in the current image to those in a potentially very different reference image – beyond the local search capability of sparse optical flow.

An alternative to sparse optical flow is to match keypoints by appearance in each frame. A well-known example is SIFT [13], which uses distinctive descriptors for robust matching across scale, rotation, and large image displacements. This makes it more suitable for OBIK, which requires matching to a potentially distant reference image. Although SIFT can be too slow for real-time use, faster alternatives such as SURF, FAST, and ORB are available, too [12], [14], [15].

Traditional keypoint methods are task-agnostic: features are chosen for general distinctiveness, without regard to their later use. In contrast, recent deep neural network (DNN) approaches learn task-specific keypoints or descriptors by optimizing tracking performance directly. For example, SuperPoint [16] trains a DNN to detect keypoints and compute descriptors, using synthetic data with known homographies to supervise correspondence learning. The resulting model is specialized for reliable matching, though still scene-independent. Similarly, the Transporter network [17] learns to detect and track a fixed number of keypoints. These methods output 2D image coordinates, with 3D positions obtainable from RGB-D data when available.

The Keypoint3D algorithm [18] learns 3D keypoints directly from multiple 2D camera views for a specific task. Because the task is defined by the reward function of a reinforcement learning (RL) problem, the learned keypoints are optimized for task performance. Experiments show that these keypoints naturally track meaningful structures, such as the limbs of a legged robot.

V. OBIK with Long-Term Tracking

The OBIK algorithm requires reliable tracks of 3D keypoints over the entire training sequence. For long-term tracking, we employ CoTracker3 [19], a recently proposed transformer-based model that tracks multiple points jointly and can handle occlusions. Unlike traditional optical flow

methods that fail upon occlusion, CoTracker3 uses cross-track attention to maintain correspondence even when points are temporarily hidden. This capability is particularly important for articulated mechanisms where self-occlusion frequently occurs during motion. We evaluate two strategies for processing multi-camera observations: independent single-view analysis and unified multi-view tracking. We use ORB keypoints during tracking [15].

A. Single-View Feature Tracking

We first consider tracking with a single calibrated RGB-D camera. Given a sequence of RGB-D images $\{I_k, D_k\}_{k=1}^{T}$ from a stationary camera with known intrinsic matrix $\mathbf{K}$ and extrinsic parameters $[\mathbf{R}|\mathbf{t}]$, we extract keypoints $\mathbf{u}_i = [u_i, v_i]^T$ in the reference frame I_1. CoTracker3 then tracks these keypoints across the sequence, producing tracks $\mathbf{u}_{ik}$ and visibility indicators $o_{ik} \in \{0, 1\}$ for $i = 1, \ldots, N$ and $k = 1, \ldots, T$.

For each visible keypoint where $o_{ik} = 1$, we obtain the corresponding depth value $d_{ik} = D_k(u_{ik}, v_{ik})$ from the depth map. Following the standard pinhole camera model, we first compute the 3D position in camera coordinates:

$$\mathbf{p}_{ik}^{cam} = d_{ik}\mathbf{K}^{-1}\begin{bmatrix} u_{ik} \\ v_{ik} \\ 1 \end{bmatrix}$$

where $\mathbf{K}^{-1}$ transforms the homogeneous pixel coordinates to normalized camera coordinates. The world coordinates are then obtained through the inverse of the camera extrinsic transformation $[\mathbf{R}_{cam}|\mathbf{t}_{cam}]$:

$$\mathbf{p}_{ik} = \mathbf{R}_{cam}^{-1}(\mathbf{p}_{ik}^{cam} - \mathbf{t}_{cam})$$

This yields a sequence of 3D keypoint measurements $\{\mathbf{p}_{ik}\}_{i=1,k=1}^{N,T}$ which serve as input to the OBIK algorithm. The algorithm then computes pairwise distance statistics, performs agglomerative clustering to identify rigid bodies, and discovers the kinematic structure as described in Section III.

However, due to the morphology of the robot arm and self-occlusions during motion, a single camera view cannot capture sufficient keypoints across all links of the mechanism. Links may become occluded as the arm moves through its workspace, resulting in incomplete tracking data and unreliable structure discovery. This limitation necessitates a multi-camera approach to ensure comprehensive coverage of the articulated mechanism.

B. Multi-View Feature Tracking

In the multi-view approach, we employ C calibrated RGB-D cameras positioned around the mechanism. Each camera $c \in \{1, \ldots, C\}$ captures a sequence of RGB-D images $\{I_k^c, D_k^c\}_{k=1}^{T}$ with known intrinsic matrix $\mathbf{K}$ (assumed identical for all cameras) and extrinsic parameters $[\mathbf{R}_{cam}^c|\mathbf{t}_{cam}^c]$.

For each camera independently, we extract keypoints in the reference frame and track them using CoTracker3, yielding tracks $\mathbf{u}_{ik}^c$ and visibility indicators o_{ik}^c for $i = 1, \ldots, N^c$ and $k = 1, \ldots, T$, where N^c denotes the number of keypoints detected by camera c. The 3D reconstruction proceeds as before:

$$\mathbf{p}_{ik}^c = (\mathbf{R}_{cam}^c)^{-1}\left(d_{ik}^c\mathbf{K}^{-1}\begin{bmatrix} u_{ik}^c \\ v_{ik}^c \\ 1 \end{bmatrix} - \mathbf{t}_{cam}^c\right)$$

The key distinction from the single-view approach is that all 3D points from all cameras are combined into a unified set for analysis:

$$S = \bigcup_{c=1}^{C}\{\mathbf{p}_{ik}^c : o_{ik}^c = 1 \text{ and } d_{min} \leq d_{ik}^c \leq d_{max}\}$$

The OBIK algorithm then processes this combined point set, computing pairwise distances between all points regardless of their camera origin. For points (i, c) and (j, c') from potentially different cameras, the distance at time k is:

$$d_{(i,c)(j,c')k} = \|\mathbf{p}_{ik}^c - \mathbf{p}_{jk}^{c'}\|$$

This multi-view formulation enables the discovery of rigid bodies that may be only partially visible from individual cameras. Points on the same rigid link maintain constant pairwise distances regardless of which camera observes them, leading to their clustering together during the agglomerative clustering step. The approach provides more robust structure discovery through redundant observations and comprehensive coverage of the mechanism's workspace.

VI. Robust Clustering of Tracked Keypoints

In its original formulation [7], OBIK used agglomerative clustering with complete linkage to group 3D keypoints into rigid bodies. This choice was motivated by the transitivity of the rigid body assumption (RBA): if points $\mathbf{p}_A$ and $\mathbf{p}_B$ satisfy it, and $\mathbf{p}_A$ and $\mathbf{p}_C$ do as well, then $\mathbf{p}_B$ and $\mathbf{p}_C$ must also satisfy it. Thus, all point pairs within a cluster must exhibit low dissimilarity, which aligns with complete linkage, where intra-cluster dissimilarity is defined as the maximum pairwise distance [20].

While complete-linkage clustering works well with precise 3D measurements [7], real-world keypoint tracking inevitably introduces noise and occasional mismatches. To address this, we propose a clustering method that relaxes the strict transitivity implied by complete linkage. Continuing with the earlier example, even if $\{\mathbf{p}_A, \mathbf{p}_B\}$ and $\{\mathbf{p}_A, \mathbf{p}_C\}$ satisfy the RBA, the algorithm should tolerate some violations between $\mathbf{p}_B$ and $\mathbf{p}_C$. Such inconsistencies can arise from noisy distance estimates or tracking errors, and the clustering method must accommodate them to remain robust.

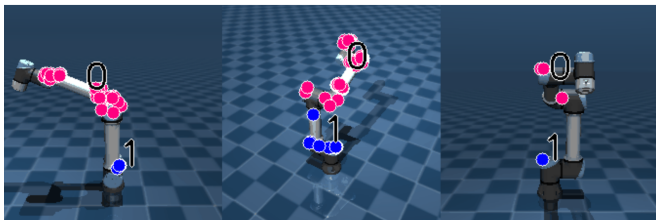

Fig. 1: An UR5 robot observed by 3 different RGB-D cameras. The tracked keypoints are shown with different color, according to their assignment to distinct bodies (magenta for body 0, blue for body 1).

We therefore propose an "almost-complete linkage" approach inspired by earlier research on clustering algorithms. In graph-theoretic terms, complete linkage partitions data into cliques – subgraphs where every pair of vertices is connected [21]. We instead use γ-quasi cliques (γQCs), where each vertex in a subgraph C of size $|C|$ is connected to at least $\gamma(|C|-1)$ others, with $\gamma \in [0,1]$. Additionally, each vertex may be allowed to miss connections to at most s others, preventing excessive dissimilarity [22]. This "almost-complete linkage" can be implemented by converting the dissimilarity matrix into a graph [23], [24] and applying a peeling algorithm to extract dense subgraphs efficiently [25].

VII. Empirical Evaluation

To test the application of CoTracker3 to keypoint tracking for use by the OBIK algorithm, we experimented with a MuJoCo simulation of the UR5 industrial robot arm, observed by 3 different RGB-D cameras (Fig. 1). The 3D position measurement noise s_{noise} was estimated by running CoTracker3 on a static scene and recording the variations in keypoint positions. On a testing set of 100 frames, following 1-DoF motion of the robot, robust estimates of the pairwise deviations s_{ij} were computed using the median absolute deviation (MAD) statistic [26]. The pairwise dissimilarity q_{ij} between two keypoints was computed as $q_{ij} = 1 - \Pr(F > F_{ij}) = CDF(F_{ij})$, where $CDF(\cdot)$ denotes the cumulative distribution function of the F-distribution, and the F-statistic is calculated as $F_{ij} = s_{ij}^2/s_{noise}^2$ [7].

Applying agglomerative clustering with complete linkage to this dataset, as proposed in [7], did not yield the expected result. For this 1-DoF motion, keypoints after the active joint should form one moving rigid body, while those before it should form a static one. However, the dendrogram in Fig. 2 consistently shows three stable clusters across most cutting thresholds, making it difficult to infer the presence of only two rigid bodies. The dissimilarity matrix in Fig. 3 explains this behavior: although two large low-dissimilarity blocks appear along the diagonal, some entries within them have values close to 1. Under the strict conditions of complete linkage, these high dissimilarities prevent the corresponding sub-clusters from merging.

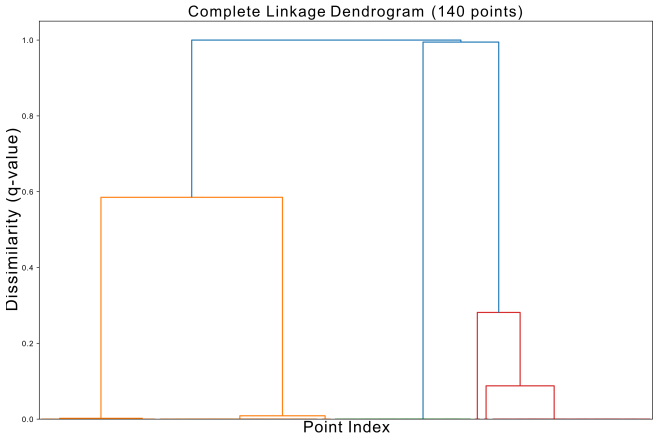

Fig. 2: Dendrogram of the agglomerative clustering algorithm using q-values.

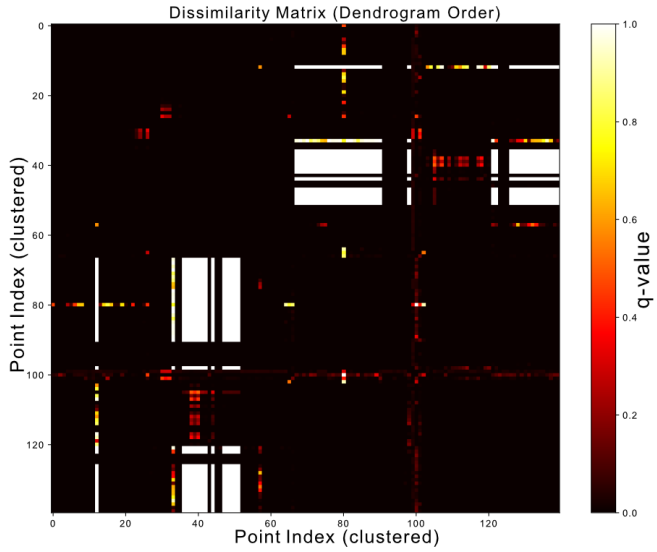

Fig. 3: Dissimilarity matrix using q-values on the training data set, after agglomerative clustering.

In contrast, the robust clustering algorithm proposed in Section VI, for a wide range of the γ parameter ($\gamma \in [0.3, 0.9]$), does produce the correct clustering into two clusters. Because the problematic high-dissimilarity values in Fig. 3 are relatively few compared to the total number of values in the sub-matrix of the diagonal block, the "almost-complete-linkage" algorithm proposed in Section VI can deal with them effectively.

After the keypoints are assigned to different bodies, the translations and rotations between them are computed as explained in Section III, by computing the SVD of the data tensors $\mathcal{T}_{lm}$, $\mathcal{R}_{lm}$, and $\hat{\mathcal{T}}'_{lm}$. (Here, we have moving body $m = 0$ and stationary body $l = 1$, see Fig. 1.) The SVD of $\mathcal{T}_{lm}$ reveals that the moving body experiences 2-DoF

translation with respect to the stationary body; this is due to the fact that the centroids of the two points undergo translational motion, and not because they are connected by a translational (prismatic) joint. The latter can be established by computing the SVD of the tensor $\hat{\mathcal{T}}'_{lm}$ of the instantaneous centers of rotation. Similarly, the SVD of $\mathcal{R}_{lm}$ produces singular values $[7.19, 0.76, 0.31]$, revealing a single rotational DoF. It should be noted that the second and third singular values are not very small, which can be attributed to the noise in keypoint positions introduced by the keypoint tracker. Still, this procedure correctly identifies the type of joint (rotational) and its axis of rotation, effectively allowing inverse kinematics to be performed from visual observations.

VIII. Conclusion and Future Work

We introduced a novel method for robust learning of state representations of articulated mechanisms composed of multiple links connected by joints, but lacking positional encoders. The method relies on short sequences of visual observations captured by stationary RGB-D cameras. Building on recent advances in deep neural network–based algorithms for long-term keypoint tracking, it extracts consistent tracks of corresponding keypoints over time, even under temporary occlusions. To identify the number of rigid bodies (links) in the mechanism, we proposed a robust clustering technique that groups points moving together, thereby revealing their membership to the same rigid body. The relaxation of the complete linkage condition implied by the rigid-body assumption turned out to be essential for the application of a keypoint tracker to this problem. The relative motion of keypoints within each body is then used to compute rigid body transforms with respect to a set of reference positions, enabling the discovery of the order of the kinematic chain. This, in turn, allows direct estimation of joint configurations (angles or displacements). Experiments with a state-of-the-art long-term keypoint tracker demonstrated successful identification of the kinematic structure of an articulated mechanism solely from camera images. In future work, we plan to apply this algorithm to state estimation and visual servocontrol of physical mechanisms such as robot arms or cranes.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat, "State representation learning for control: An overview," *Neural Networks*, vol. 108, pp. 379–392, 2018.

[3] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 512–519.

[4] R. Jonschkowski and O. Brock, "Learning state representations with robotic priors," *Autonomous Robots*, vol. 39, pp. 407–428, 2015.

[5] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "Learning deep dynamical models from image pixels," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 1059–1064, 2015.

[6] T. Kipf, G. F. Elsayed, A. Mahendran, A. Stone, S. Sabour, G. Heigold, R. Jonschkowski, A. Dosovitskiy, and K. Greff, "Conditional object-centric learning from video," *arXiv preprint arXiv:2111.12594*, 2021.

[7] D. Nikovski, "Disentangled Object-Centric Configuration Representation Learning for Articulated Robot Arms," in *International Conference on Control, Decision and Information Technologies (CoDIT)*, Split, 2025.

[8] D. N. Nikovski, "Observation-Based Inverse Kinematics for Visual Servo Control," in *22nd International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 10 2025, pp. 200–207. [Online]. Available: https://www.merl.com/publications/TR2025-153

[9] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[10] F. Chaumette, S. Hutchinson, and P. Corke, "Visual servoing," *Handbook of Robotics*, pp. 841–866, 2016.

[11] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991.

[12] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[14] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[16] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.

[17] T. D. Kulkarni, A. Gupta, C. Ionescu, S. Borgeaud, M. Reynolds, A. Zisserman, and V. Mnih, "Unsupervised learning of object keypoints for perception and control," *Advances in neural information processing systems*, vol. 32, 2019.

[18] B. Chen, P. Abbeel, and D. Pathak, "Unsupervised learning of visual 3d keypoints for control," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1539–1549.

[19] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht, "Cotracker: It is better to track together," in *European Conference on Computer Vision*. Springer, 2024, pp. 18–35.

[20] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining*, vol. 2, no. 1, pp. 86–97, 2012.

[21] J. Abello, P. M. Pardalos, and M. G. C. Resende, "On maximum clique problems in very large graphs," *External memory algorithms*, vol. 50, pp. 119–130, 1999.

[22] B. Balasundaram, S. Butenko, and I. V. Hicks, "Clique relaxations in social network analysis: The maximum k-plex problem," *Operations Research*, vol. 59, no. 1, pp. 133–142, 2011.

[23] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *IEEE Transactions on computers*, vol. 100, no. 11, pp. 1025–1034, 1973.

[24] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," *Advances in neural information processing systems*, vol. 17, 2004.

[25] M. Charikar, "Greedy approximation algorithms for finding dense components in a graph," in *International workshop on approximation algorithms for combinatorial optimization*. Springer, 2000, pp. 84–95.

[26] P. J. Rousseeuw and C. Croux, "Alternatives to the median absolute deviation," *Journal of the American Statistical association*, vol. 88, no. 424, pp. 1273–1283, 1993.