Inferring Robot Morphology from Observation of Unscripted Movement

Neil Bell, Brian Seipp, J. Tim Oates, Cynthia Matuszek[1]

***Abstract*— Task sharing between heterogeneous robots currently requires a priori capability knowledge, a shared communication protocol, or a centralized planner. However, in practice, when two robots are brought together, the effort required to construct shared action and structure models is significant. In this paper, we describe our approach to determining the kinematic model of a robot based purely on observation of unscripted movement. We describe construction of large-scale data simulating low-cost RGB-D camera output, and application of two different RNN-based methods to the learning problem. Our results suggest that this is an efficient and effective way to determine a robot's morphological structure without requiring communication or pre-existing knowledge of its capabilities.**

I. Introduction

When multiple robots collaborate on solving some problem, it is important to consider what role, or *task*, each one will perform [1]. For more complex problem spaces, efficient robot teams are frequently heterogeneous in both structure and capability [2]. Furthermore, as collaborative robots become more ubiquitous, it becomes desirable to allow the formation of *ad hoc* teams, with compositions potentially shifting over time.

While there are a number of ways to do this, task assignment in heterogeneous groups often relies on the existence of communication or shared models of capability, either between members or on the part of a centralized planner. However, in practice, when two robots are brought together, the effort required to construct shared action and structure models is significant; in *ad hoc* teams, hand-crafting shared planning and communication solutions on a per-team basis becomes impractical. One approach to allowing such heterogeneous teams to work together scalably is to enable robots to learn about one another solely from observation.

We describe work using deep learning methods to derive a model of a robot's capabilities by estimating its physical structure from observations of its actions, updating confidence as new observations occur. We demonstrate this by continuously modeling the physical structure of an unfamiliar manipulator arm based on sensor observations. This morphology can then be used to plan an optimal strategy to achieve a cooperative goal. We demonstrate the utility of this work by considering a scenario where two robots, each with their own depth sensor, have no shared communication. The robots are instructed to accomplish the goal of relocating an object to a new position. The goal is such that cooperation between the robots is necessary. Planning and detection is inherently passive, with each robot evaluating the goal, observing the world, and building hypotheses about the capabilities of nearby robots who may help accomplish the goal. This research addresses one approach for how a robot can build confidence in a nearby robot's capabilities through observation over time and potentially use that information to plan an optimal strategy to achieve a cooperative goal.

The core contributions presented in this work are as follows. We have developed and made available a large corpus of simulated point-cloud data of a variety of manipulators of different size and degrees of freedom. We describe an effective method of processing that data for learning, and finally, we explore deep learning approaches to estimate a kinematic model of those manipulators from time-series observations. Our results make it clear that this is an effective approach and a promising area for further research.

II. Related Work

The goal of this research is to establish how robots can gain an accurate representation of other robots in their observation space. This will enable computation of useful plans to achieve a cooperative goal. Applications of cooperative robotic agents include, but are not limited to, scenarios such as exploring unknown environments [3], [4], [5], safeguarding secure facilities [6], and construction tasks [7]. Such planning requires ample knowledge about the scene and robot's potential partners in completing the task [8], and even requires on-line determination of how proactive or reactive a robot should be in performing a cooperative task [9]. Given morphology, there are a number of approaches to motion planning [10], [11], [12] and path planning [13], [14], collaborative task assignment [15], [16], [17], and behavior learning [18]. This work is also related to prior work into robot-human motion detection and mimicry,

[1]All authors are with the University of Maryland, Baltimore County. 1000 Hilltop Circle, Baltimore, MD, 21250, USA.
bell5|seipp1|oates|cmat@umbc.edu

allowing the robots to leverage pre-trained supervised learning models for optimal behavior [19].

Our research looks toward collaborative robotics by extending prior work with human-robot teaming and efficient task assignment [20], [21], as well as single-coordinator action sequencing between connected robot participants [22]. Shared planning will build on prior research into implicit communication through motion [23].

We explore initializing learning models using a supervised approach in which we use a simulator to generate a large number of arms with varying morphologies and learn a mapping of observables (i.e., simulated point clouds). We will possibly later also map to capabilities using an auto-encoder for which the inputs are depth images concatenated with a vector of link lengths padded with zeroes [24] or recurrent neural networks. Similar deep learning techniques have been applied with success in the robotic grasp problem space [25], [26], as well as generalized object recognition [27]. Prior research also demonstrates successful use of neural net-based deep learning for merged environment and behavior evaluation [28], leveraging Long Short Term Memory cells for sequential processing [29].

Our work is most similar to, and is informed by, a rich body of work on articulated pose estimation [30] and kinematic modeling [31]. Prior work proved it is possible for a robot to actively learn a kinematic model of its own manipulator using data from an attached depth sensor [32], or to perform on-line tuning of its own kinematics through internal sensing [33]. While there is also extensive research into kinematic modeling during gesture recognition [34], we are not aware of other work on estimating the capabilities of an unknown robot by learning its whole morphology through observation.

III. Data Corpus

An overview of our approach is as follows. First, we present a large-scale corpus of time-series point cloud observations showing the movement of a number of differently structured robots. Second, we describe processing this data to extract features suitable for learning a model of morphology estimation.

A. Generated 3D Motion Sequences

To understand a robot's morphology, we focus on estimating its kinematic chain. For a non-forking, non-cyclical 'arm' structure, this means determining the number and length of a sequence of connected links, as well as joint type (revolute *vs.* prismatic) and joint angle limitations. In this work we focus on identifying the first two values, which provide an outer bound on the robot's workspace. Our approach results in increasing confidence in an observed manipulator's structure over time, as sequences of observations provide more evidence. The initial probability space is uniform.

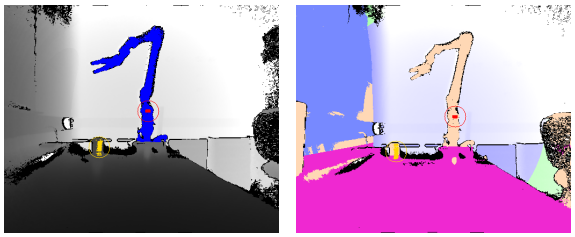
Deep learning approaches generally benefit from a significant number of examples. For this problem, such observations are difficult to obtain for a wide range of possible arms going through a variety of motions. Accordingly, we built a simulator which generated realistic observation sequences based on randomly selected robot structures and motion, represented as an unknown number of non-branching rigid links connected by joints with an unknown range of motion. The observations were designed to mimic those provided by a Kinect2 using pointclouds as described in section III-B [35].

The simulator varied the following parameters for generating and animating robot movements: the link count, n; the link diameter, d; and the individual link lengths, described below. The simulator randomly generated robot structures according to those parameters, using rotational joints, animating movements around a normal vector at each joint. A robot with n links has links numbered l_0, representing the base, to l_{n-1} representing the end effector, and joints numbered j_0 to j_{n-2}. The animation logic selected a joint at random and rotated it in either a clockwise or counter-clockwise direction for a minimum number of frames before reselecting a joint at random.

To cover a range of structures while bounding classification complexity, the simulator randomly selected a link count in the range $[2, 6]$. Each link received a randomly generated length in the range $[1000, 7500]$ units (units are approximately 0.1 millimeters), with a randomly generated radius in the range $[200, 400]$. These values were chosen to provide a reasonable simulation starting point. Similar to most available non-industrial manipulators, all generated robots' links were sequential and non-branching. Future work could extend the simulator to produce movement of more complex robot structures for greater generality.

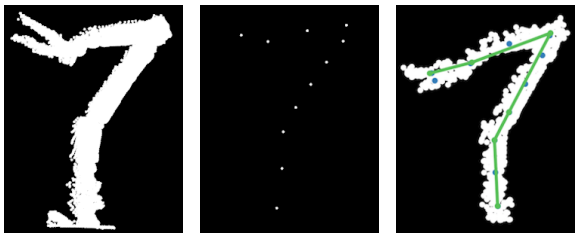
At each frame in the animation sequence, the simulator renders a cylindrical point cloud around each link in the robot structure, then prunes any points occluded from the fixed camera location by some part of the robot (that is, self-occluded points). Culling the occluded points was intended to mimic what a statically positioned Kinect2 sensor experiences, as it can't include rear-facing surfaces in the point cloud. Lastly, the simulator generated a representative "skeleton" of the point cloud using k-means clustering to greatly reduce the number of points a classifier must process. This skeletonization process is described in detail below.

We set the simulator to generate 100 frames per sequence, saving the skeleton and selected metadata for each sequence into a single file. Each frame in a sequence is comprised of the frame number, the ordered link lengths, the ordered vertex locations of the link endpoints (with joints located at internal vertices), or-



(a) (b)

Fig. 1. Obtaining the point cloud. (a) Original Kinect 3-D; (b) RANSAC segmentation.



(a) (b) (c)

Fig. 2. Application on a JACO2 arm using a Kinect 2. (a) Original point cloud; (b) centroids after k-means decomposition; (c) hypothesized morphology after one frame.

dered joint types, joint normals for calculating rotations reliably, the camera location for calculating occlusion, and the skeleton point locations.

TABLE I

VIDEO SEQUENCES FOR CHAINS OF DIFFERENT LENGTHS.

Number of links	2	3	4	5	6
Number of video sequences	2,011	2,001	2,007	2,019	2,011

For this research, we generated a total of 10,049 sequences of 100 frames, subdivided as shown in table I. This corpus is available publicly for unrestricted download.[1] In the near future, it is our intention to also make the associated full point clouds available.

For a sufficiently representative corpus, the size of unprocessed point clouds generated is unwieldy, and the total set of points is redundant for structural identification. Accordingly, these sequences are reduced as follows. Given a single observation, the point cloud is skeletonized as described in section III-B. This decomposition process generates a minimal set of points which accurately trace the shape of the robot. These points represent features used to predict hypothetical structures.

B. Data Preprocessing

The application runs a series of filtering steps on each point cloud frame to segment out the manipulator. Figure 1(a) shows a 3-D gray-scale rendering of Kinect2 sensor output of a Kinova JACO2 robot arm attached to a table. The blue highlight on the robot arm is the result of detecting the arm, and provides the classifier with observation data. The Kinect2 publishes 3D point clouds and RGB images.

We run the point cloud through a series of Random Sample Consensus (RANSAC [36]) filters to prune extraneous points. First, we attempt to detect and remove any large planes present, as these are likely to represent walls or tables. Figure 1(b) shows the results of RANSAC's detection of the wall to the left of the robot (dark blue) and the table on which the JACO2 robot and Kinect2 sensor are sitting (magenta). The remaining peach-colored portions are points which remain in the dataset. We then remove smaller contiguous 'blobs.' The remaining large contiguous sets of points are assumed to be the robot. This approach is simple and effective in our test scenarios. More complex segmentation approaches could be easily substituted for more natural scenes, which may be both dynamic and cluttered.

The remaining points are represented in a Cartesian coordinate space anchored at the Kinect2's depth sensor. Post-filtering, this cloud generally contains many thousands of points, most of which are redundant for the classification task. This level of granularity may be useful for other tasks where greater fidelity is needed. However, for the purposes of this research, we ease downstream computational complexity by performing the k-means simplification step described in section III-A, which reduces points while still accurately representing the robot's core structure.

We generate this skeleton by running unsupervised k-means clustering after plane removal, keeping only the points representing the centroids of each cluster. We used the elbow method for empirically determining an optimal value for k, beginning with $k = 5$ and incrementally increasing k until the cost evaluation identified significantly diminished improvement.

Figure 2 shows a progression between (a) the JACO2's post-RANSAC point cloud, to (b) the skeleton points obtained through k-means clustering, to (c) how output from a trained classifier produces a hypothesis of the actual kinematic model. Ideally, hypotheses will represent the observed robot without overestimating complexity, while optimally fitting the original point cloud. Because the Kinect2 can only see the surfaces of the robot that are facing the sensor, the centroids are located on the near surfaces rather than inside the robot's links.

Each frame received by our C++ application from the Kinect2 through the Robot Operating System (ROS) was processed by the steps above, leaving only the skeleton points as the final "observation" to be used by the downstream models. We hypothesize that this skeleton is sufficient for determining a robot's structure and length,

[1] https://tiny.cc/iral-arm-corpus/simulated_robot_clouds.tar.gz

which may suffice for gross scene evaluation and the eventual path planning for task completion.

IV. Approach

In this section, we describe the application of two different learning approaches: a recurrent neural network (RNN) that estimates the complete kinematic chain, and a pipelined RNN-based approach that first estimates degrees of freedom (links), followed by estimating their lengths. Both RNN-based approaches had advantages and disadvantages. Ultimately, the pipelined approach with specialized models shows the most promise.

A. Complete-Structure RNN Prediction

Due to the complexity of observations of robot movement from a depth sensor, we chose to use neural network models rather than a hidden Markov model (HMM). The potential branching factor and non-determinism behind a robot's movement implies a likely intractable HMM implementation. Intuitively, a sufficient state machine that could capture arbitrary movement sequences may not generalize well to new structures. In this section, we discuss the reasoning for using a recurrent neural network (RNN) made up of Long Short Term Memory (LSTM) cells to capture the sequential nature of the observations.

We implemented an RNN capable of evaluating a sequence of observations rather than outputting a prediction one frame at a time. We trained the RNN to determine if sequential observations avoided the inherent weakness of the single-frame observations. An RNN's applicability to sequential time series data may avoid failures resulting from sensor noise, occlusion, and other causes.

The simulator described in section III-A outputs observation sequences and the ground truth about the robot structure from which the sequences were generated. We used LSTM cells for each of the RNN's hidden layers consisting of a width equal to the number of nodes per layer. We used the rectified linear unit (RELU) activation function for the entire network. The cost evaluation was given by L2 Euclidean distance between the predicted and true labels.

We first tune hyper-parameters on the RNN model to determine an optimal number of hidden layers and an appropriate number of nodes per layer. We intended to determine a network shape that reduced complexity and still generalized to new data. Figure 3 shows the reduction in test cost following each training epoch for selected parameter permutations.

Of note, networks with one or two hidden layers appeared to fail to learn enough about the input data. Also interesting was the behavior of the network with two layers and 180 nodes per layer. It appeared to learn up to an asymptote before suddenly discovering a route to a better optimum at epoch 38. This drastic behavior change was something we hoped to avoid, and instead decided to use the three layer network with 140 nodes per layer, as this configuration provided smooth and consistent learning behavior.

B. RNN Pipelining

The initial approach with the RNN models was to train a single model capable of receiving an observation and predicting both the number of links and the length of each link. We later attempted a pipelined approach such that these two predictions became decoupled and occurred systematically. We trained a classifer to only predict the correct number of links for the robot, outputting a one-hot vector of the format $\{l_2 = 0, l_3 = 1, l_4 = 0, l_5 = 0, l_6 = 0\}$ with all values equal to 0 except for a single 1, with this example signifying that the specific observation was representative of a robot with three links ($l_3 = 1$). We then trained several multi-output regression models, one for each of the possible number of links. Our goal was to first have the one-hot classifier determine the correct number of links before passing the same observation to the appropriate second-phase model which could then estimate the lengths of those predetermined links. By specializing the models, we hoped to more accurately predict the correct number of links and their lengths.

V. Experimental Results

Our first round of experiments were with the holistic RNN model, where a single learner was fed observation sequences, each of 100 frames. The model was trained to output a prediction vector of the format $[n, l_0, l_1, l_2, l_3, l_4, l_5]$, representing the number of links and their predicted link lengths. Links with sufficiently small lengths are dropped from the kinematic model. Figure 4 shows the performance of the multi-learner RNN model in terms of decrease in test error loss (red)

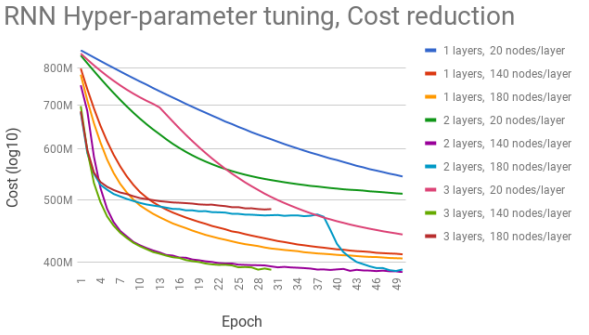


Fig. 3. Improvement during hyper-parameter tuning of the RNN. Permutations of parameters shown are chosen to provide insights into how various structural categories perform.

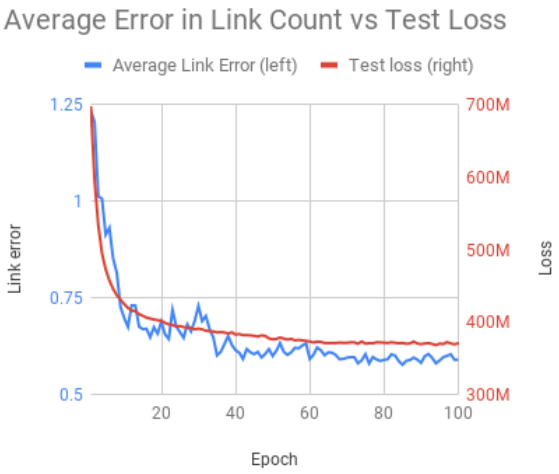


Fig. 4. Performance of RNN multi-learner. Red shows link count error and blue shows test loss after each training epoch.

as well as the improvement in average link count error across all test cases for each training epoch. The loss against the test batch decreased each epoch as expected, with a significant rate of improvement until it slowed around epoch 20.

To calculate the average link count error shown in Figure 4 (blue), the number of predicted links were summed together across all test instances, before being subtracted from the same calculation against the true label. The calculation of the average link count error for epoch i with t test observations is as follows, where $truth_i$ and $pred_i$ are the number of links for the true label and predicted labels respectively for observation i:

$$error_i = \frac{|\sum_{n=1}^{t} truth_n - \sum_{n=1}^{t} pred_n|}{n}$$

Figure 4 shows that test loss significantly decreased (red), however it took longer for the model's link count accuracy to converge (blue). After 80 training epochs, the tested average link count difference was still approximately 0.6 links, showing difficulty in accurately predicting discrete link counts. This result led us to decouple the classification logic into the pipelined approach explained in section IV-B, because we hypothesized that specialized models would perform better when only trained to perform a single specific task.

The pipelined RNN system consisted of two separate layers of ordered learners. First, we trained a one-hot classifier on the entire simulated data corpus in order to output a discrete link-count prediction. Second, we trained individual models to receive the same observation as the one-hot classifier and output a vector of link-lengths. At a high level, the individual length prediction models performed very well, but the overall accuracy was limited by the model predicting the number of links.

Figure 5 shows the prediction accuracy for each of the possible discrete link counts. This shows how many observations the one-hot learner correctly predicted by link count. This accuracy of this initial number of links model varied depending on the true number of links, showing the best performance at the extremes (two and six-link manipulators). In practice, robots with three, four, and five links are more difficult to differentiate during a motion sequence. In cases where relatively few bent joints are observed—that is, links are in a straight line—it is still possible to make informed predictions about extremely long or extremely short arms. Our intention is to further investigate the performance and training of the first stage of the model, initially by exploring the use of more dynamic sequences of motion.

We then trained five models, each seeing only data consisting of one of the five link-count values ($l = \{2, 3, 4, 5, 6\}$). Figure 6 plots the test loss after

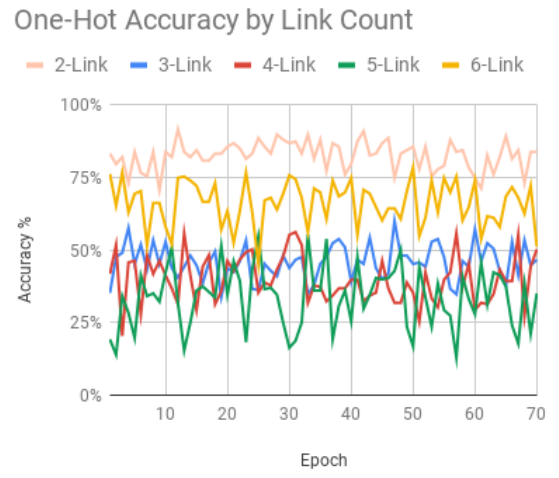


Fig. 5. Performance of RNN one-hot link count classifier. Each line shows the prediction accuracy for a particular link count.

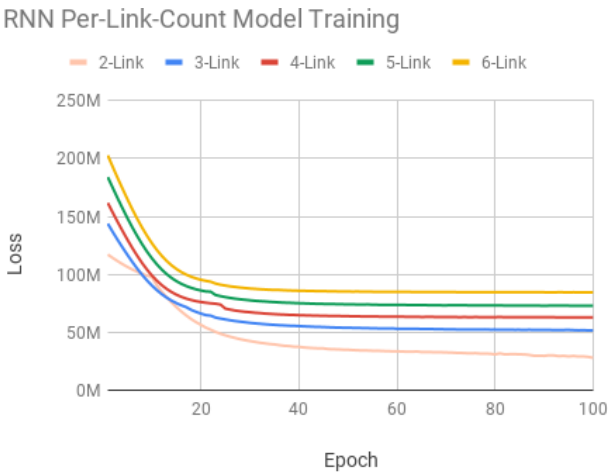


Fig. 6. Test loss reduction while training RNN per-link-count models, one line per model.

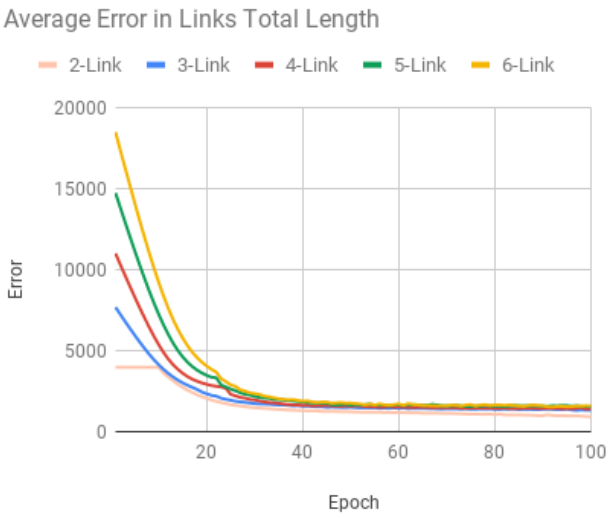


Fig. 7. Mean discrepancy between true and predicted total length.

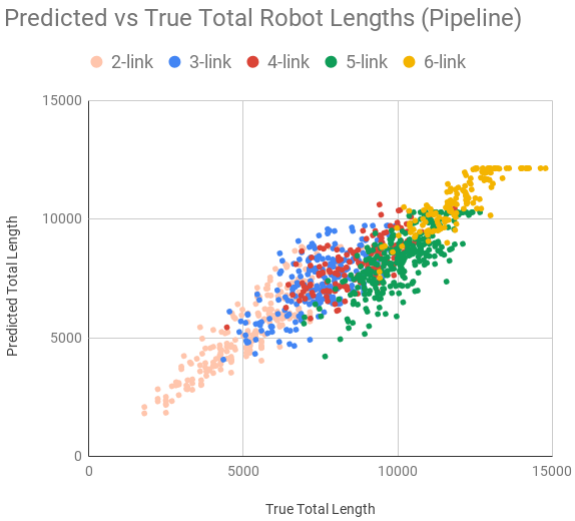


Fig. 8. Relationship between a simulated robot's true total length versus the post-pipeline prediction.

each of 100 training epochs. Each of the models showed improvement until hitting an asymptote, although the two-link model did continue to improve beyond where the others leveled out. The two-link learner showed slower learning during the first few epochs before it found a faster improvement gradient.

Next, we focused on capturing the ability of the per-link-count models to accurately predict the total length of the observed robot. Figure 7 shows the reduction of the average difference between the true and predicted robot lengths after each epoch. The calculation for this graph is as follows:

$$diff_i = \frac{|\sum_{n=1}^{t} len(truth_n) - \sum_{n=1}^{t} len(pred_n)|}{n}$$

Despite the two-link learner making no progress early, Figure 7 shows that all learners eventually converged to a difference of approximately 1,000. We expected to see an asymptote around that level resulting from how the k-means skeletonizing dispersed centroids around the robot's point cloud. For n total skeleton points s_0 to s_{n-1} ordered from the robot's base to the end-effector, s_0 can't be located at the extreme end of the base as the centroid must be in the middle of the points in its cluster. Nor can s_{n-1} be at the tip of the end-effector since it would be drawn inwards by the points in its cluster. Consequently the skeleton points can't fully capture the entire length of the robot and this floor appears. With higher values of k for k-means, one might expect this discrepancy to decrease as skeleton centroids can more closely approach the extreme ends of the robot.

Finally, Figure 8 shows the result of end-to-end performance of the pipelined approach described in section IV-B. The graph compares each robot's true total length (x) to the total length of the prediction (y) following the one-hot classification and specialized model evaluation. The points are colored based on their respective predicted link counts, showing a trend up and to the right following the growth of a robot's potential total length. The accuracy of the pipeline approach is shown by the near-linear relationship between the true and predicted lengths. There appears to be a ceiling in the final 6-link series which is worth further investigation; however, Figure 8 demonstrates that our approach can find useful morphologies through observation of unscripted movement.

VI. Conclusion and Future Work

In practice, even robots with shared capability information attempting to accomplish a shared goal can require significant effort. Disconnected, heterogeneous robotic systems can be made more adaptable and resilient by using a robust learning process. Two depth-sensing robots who share a cooperative goal can learn about each other by observation and use that morphological information to model capabilities.

In this paper we use supervised learning over observation of a specific, common structure, the non-branching kinematic chain. We demonstrate that it is possible for a recurrent neural network to use motion sequences to determine a robot's structure with some accuracy, providing baseline context to a robot about the size of its potential partner's workspace. This method for determining morphology from observation demonstrates the promise of this area of research. In the future, we will explore related subproblems. Specifically, the behavior of the one-hot classifier stage of the pipeline warrants additional study and experimentation, as does formalizing a benchmark for the upper bounds on classification accuracy. Our future work will continue to focus on the broad questions surrounding how robotic agents

with no initial shared structural knowledge can execute cooperative tasks successfully.

REFERENCES

[1] M. J. Matarić, G. S. Sukhatme, and E. H. Østergaard, "Multi-robot task allocation in uncertain environments," *Autonomous Robots*, vol. 14, no. 2-3, pp. 255–263, 2003.

[2] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[3] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, April 2000, pp. 476–481 vol.1.

[4] J. A. Mendez-Polanco and A. Munoz-Melendez, "Collaborative robots for indoor environment exploration," in *2008 10th International Conference on Control, Automation, Robotics and Vision*, Dec 2008, pp. 359–364.

[5] M. N. et al., "Collaborative mapping of an earthquake damaged building via ground and aerial robots," in *Field and Service Robotics*, vol. 92, Dec 2014, pp. 33–47.

[6] Y. Guo, L. Parker, and R. Madhavan, "Towards collaborative robots for infrastructure security applications," *The 2004 International Symposium on Collaborative Technologies and Systems*, 2004.

[7] E. Gambao, M. Hernando, and D. Surdilovic, "A new generation of collaborative robots for material handling," in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, vol. 29. Vilnius Gediminas Technical University, Department of Construction Economics & Property, 2012, p. 1.

[8] A. Farinelli, L. Iocchi, D. Nardi, and V. A. Ziparo, "Task assignment with dynamic perception and constrained tasks in a multi-robot system," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 1523–1528.

[9] J. Baraglia, M. Cakmak, Y. Nagai, R. Rao, and M. Asada, "Initiative in robot assistance during collaborative task execution," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2016, pp. 67–74.

[10] N. E. Du Toit and J. W. Burdick, "Robotic motion planning in dynamic, cluttered, uncertain environments," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 966–973.

[11] N. Xi, "Event-based motion planning and control for robotic systems," 1993.

[12] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt." Institute of Electrical and Electronics Engineers, 2011.

[13] V. J. Lumelsky, S. Mukhopadhyay, and K. Sun, "Dynamic path planning in sensor-based terrain acquisition," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 4, pp. 462–472, Aug 1990.

[14] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[15] N. Miyata, J. Ota, T. Arai, and H. Asama, "Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment," *IEEE transactions on robotics and automation*, vol. 18, no. 5, pp. 769–780, 2002.

[16] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Aaai/Iaai*, 2000, pp. 852–858.

[17] M. M. Veloso, P. Stone, and M. Bowling, "Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer," in *Sensor Fusion and Decentralized Control in Robotic Systems II*, vol. 3839. International Society for Optics and Photonics, 1999, pp. 134–142.

[18] J. Auerbach and J. C. Bongard, "How robot morphology and training order affect the learning of multiple behaviors," *2009 IEEE Congress on Evolutionary Computation*, pp. 39–46, 2009.

[19] K. Yamane, M. Revfi, , and T. Asfour, "Synthesizing object receiving motions of humanoid robots with human motion database," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[20] J. Shah, J. Wiken, B. Williams, and C. Breazeal, "Improved human-robot team performance using chaski, a human-inspired plan execution system," in *Proceedings of the 6th International Conference on Human-robot Interaction*, ser. HRI '11. New York, NY, USA: ACM, 2011, pp. 29–36. [Online]. Available: http://doi.acm.org/10.1145/1957656.1957668

[21] M. C. Gombolay, R. A. Gutierrez, S. G. Clarke, G. F. Sturla, and J. A. Shah, "Decision-making authority, team efficiency and human worker satisfaction in mixed human–robot teams," *Autonomous Robots*, vol. 39, no. 3, pp. 293–312, 2015.

[22] M. Gombolay, R. Wilcox, and J. Shah, "Fast scheduling of multi-robot teams with temporospatial constraints," 2013.

[23] R. A. Knepper, C. I. Mavrogiannis, J. Proft, and C. Liang, "Implicit communication in a joint action," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '17. New York, NY, USA: ACM, 2017, pp. 283–292. [Online]. Available: http://doi.acm.org/10.1145/2909824.3020226

[24] A. Kumar and T. Oates, "Connecting deep neural networks with symbolic knowledge," *30th Int'l Joint Conference on Neural Networks*, 2017.

[25] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

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

[27] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 681–687.

[28] K. Noda, H. Arie, Y. Suga, and T. Ogata, "Multimodal integration learning of robot behavior using deep neural networks," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 721 – 736, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889014000396

[29] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2786–2793.

[30] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *Computer Vision and Pattern Recognition*, 2011.

[31] L. Sciavicco and B. Siciliano, *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012.

[32] A. Broun, C. Beck, T. Pipe, M. Mirmehdi, and C. Melhuish, "Building a kinematic model of a robots arm with a depth camera," in *FIRA-TAROS 2012, LNAI 7429*, 2012, pp. 105–116.

[33] C. Kim, S. C. Ryu, and P. E. Dupont, "Real-time adaptive kinematic model estimation of concentric tube robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 3214–3219.

[34] R. B. Dan and P. Mohod, "Survey on hand gesture recognition approaches," *structure*, vol. 15, p. 17, 2014.

[35] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (pcl)," in *International Conference on Robotics and Automation*, 2011.

[36] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.