Learning to Assemble Objects with a Robot Swarm (Extended Abstract)

Gregor H.W. Gebhardt
Computational Learning for Autonomous Systems,
Technische Universität Darmstadt

Kevin Daun
Computational Learning for Autonomous Systems,
Technische Universität Darmstadt

Marius Schnaubelt
Computational Learning for Autonomous Systems,
Technische Universität Darmstadt

Alexander Hendrich
Computational Learning for Autonomous Systems,
Technische Universität Darmstadt

Daniel Kauth
Computational Learning for Autonomous Systems,
Technische Universität Darmstadt

Gerhard Neumann
Lincoln Centre for Autonomous Systems, School of Computer Science, University of Lincoln

1. INTRODUCTION

Nature provides us with a multitude of examples that show how swarms of simple agents are much richer in their abilities than a single individual. This insight is a main principle that swarm robotics tries to exploit. In the last years, large swarms of low-cost robots such as the Kilobots [6] have become available. This allows to bring algorithms developed for swarm robotics from simulations to the real world. Recently, the Kilobots have been used for an assembly task with multiple objects [7]: a human operator controlled a light source to guide the swarm of light-sensitive robots such that they successfully assembled an object of multiple parts. However, hand-coding the control of the light source for autonomous assembly is not straight forward as the interactions of the swarm with the object or the reaction to the light source are hard to model.

In this paper, we investigate how to learn a policy that controls the light source such that the Kilobots solve the assembly task autonomously. We subdivide the assembly process in two sub-tasks: the high-level assembly plan is given by waypoints for each object while the learned low-level object movement policy controls the trajectory execution by guiding the Kilobots with the light source.

Learning to move objects is a complex task as we have to coordinate a large number of agents, which results in many state variables. Eventough we need information about the configuration of the swarm (i.e. the positions of the agents) in our state representation, it is not important which individual of the swarm is at which position and, furthermore, our policy should be as well invariant to the number of agents in the swarm. Hence, we represent the state of the swarm as a distribution over the agent locations embedded into a reproducing kernel Hilbert space [8]. Our reinforcement learning algorithm is based on the recently introduced actor-critic relative entropy policy search (AC-REPS) algorithm [9] and learns a nonparametric Gaussian process (GP) policy for controlling the light source. We evaluate our approach in simulation on assembly tasks with different object shapes.

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

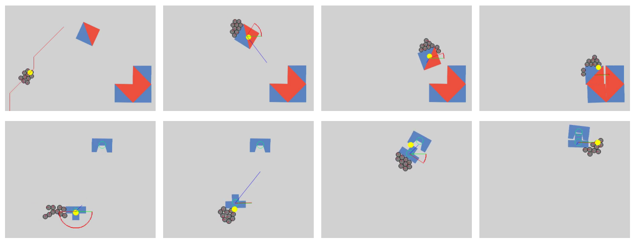


Figure 1: We propose a new reinforcement learning approach to enable a robot swarm (grey circles) to assemble objects (top: squares, bottom: T- and C-shape). The swarm is guided by a light cone (yellow circle) which follows a learned control policy.

2. SOLVING AN ASSEMBLY TASK

Our solution to the assembly task consists of three components: (1) a path planning strategy to guide the swarm around the objects and to arrange them for the next pushing task, (2) an assembly policy that describes how the parts should move such that they in the end form a whole unit, and (3) an object movement policy that realizes basic movements of an object part indirectly by controlling a light source that guides the robots.

Learning the Object Movement Policy.

We use AC-REPS [9] to learn the continuous, nonparametric object movement policies from samples. This model-free reinforcement learning approach based on relative entropy policy search (REPS) [5] consists of the following three steps: (1) Estimate the Q-function using the observed state transitions with least-squares temporal difference learning (LSTD). (2) Improve the policy by maximizing the expected Q-value for the sampled data using REPS. (3) Obtain a continuous policy by matching a weighted Gaussian process.

We learn multiple policies to control the light source which differ in their objective: one policy for pure translation in positive x-direction, one for pure counterclockwise (ccw) rotation of the object, and arbitrary many policies for different ratios w of combined translation and rotation. The reward function reflects the setting of the control parameter $w \in [0, 1]$ which sets the ratio of translation and rotation. The function rewards a ccw rotation for $w = 1$ and a translation in x direction for $w = 0$. For values $w \in]0, 1[$ the terms are weighted accordingly. Later, we will apply the object movement policies to arbitrary target directions and orientations by rotating and flipping the state of the world.

We represent the state relative to the center of mass of the object part that we want to push. Given a swarm configuration with n agents and their relative positions $\boldsymbol{b}_i = (x_i, y_i)$ as well as the relative light position $\boldsymbol{l} = (x_l, y_l)$, we define the state vector as $\boldsymbol{s} := [\boldsymbol{l}, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$. The action vector $\boldsymbol{a} = (a_x, a_y)$ contains the desired shift of the center of the light cone.

To learn the object movement policy independently from the number of individuals in the swarm, we represent the swarm as distribution embedded into a reproducing kernel Hilbert space (RKHS). Let $\mathcal{H}$ be an RKHS of functions, uniquely defined by a positive definite kernel function $k(x, x') := \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$ [2]. Here, the feature mappings $\phi(x)$ are in general intrinsic to the kernel functions and might map into an infinite dimensional feature space. We can embed now a marginal distribution $p(X)$ as the expected feature mapping of its random variable [8] $\mu_X := \mathbb{E}_X [\phi(X)]$. In practice we need to estimate the embedding from samples as $\hat{\mu}_X = \frac{1}{m} \sum_{i=1}^{m} k(x_i, \cdot)$.

Thus, the state of the swarm relative to the center of mass of the part we want to push is represented as $\mu(\boldsymbol{k}) = \frac{1}{n} \sum_{i=1}^{n} k(\boldsymbol{b}_i, \cdot)$, where k is a kernel function (e.g., the Gaussian kernel) and the $\boldsymbol{b}_i$ are the locations of the individual agents. With this representation, we can compute the difference between two swarm distributions by computing the squared difference of their embeddings

$$\frac{1}{n^2} \sum_{\substack{i=1\\j=1}}^{n,n} k(\boldsymbol{b}_i, \boldsymbol{b}_j) - \frac{2}{nm} \sum_{\substack{i=1\\j=1}}^{n,m} k(\boldsymbol{b}_i, \boldsymbol{b}'_j) + \frac{1}{m^2} \sum_{\substack{i=1\\j=1}}^{m,m} k(\boldsymbol{b}'_i, \boldsymbol{b}'_j),$$

with the two swarm configurations $\boldsymbol{b}$ and $\boldsymbol{b}'$ with n and m individuals, respectively. Based on this distance metric, we define a feature vector using radial basis functions (RBF) centered around our samples. The remaining state-action variables (position and desired shift of the light) are compared by a squared distance. We can now combine these two distance measures into an exponential product kernel based on which we define the feature functions for approximating the Q-function and for matching the weighted GP as the new policy.

Assembly Policy and Path Planning.

The assembly policy contains the construction information stored as a list of oriented way points with required accuracies for each object. These way points are processed consecutively by applying either the object movement policy or the path planning strategy. The desired translation-rotation ratio $w_{\text{des}} = e_{\text{rot}}/(e_{\text{trans}} + e_{\text{rot}})$ obtained from the rotational and translational error determines which of the learned object movement policies is executed.

After reaching a way point, the swarm has to be arranged at the next object. We use a path planning strategy to guide the swarm around objects and arrange them for the next pushing task. This strategy is a combination of A* with potential fields.

A* is a heuristic search algorithm commonly applied for graph search problems [3]. The algorithm selects which node n_s to expand by minimizing the sum of the cost we have to spend for reaching node n_s from the start and a heuristic lower bound to the cheapest costs from n_s to the goal state.

Potential fields [4] are a fast planning method for mobile robots. The robots move along a hypothetical force field, being attracted towards the goal position and repulsed away from the obstacles. In our approach, we use the repulsive potential in the cost term for the path segments $c(n_{s_1}, n_{s_2})$ of A*.

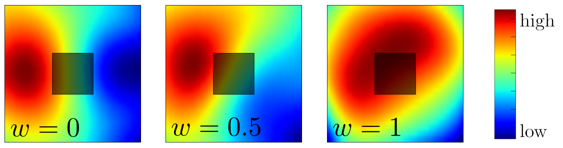
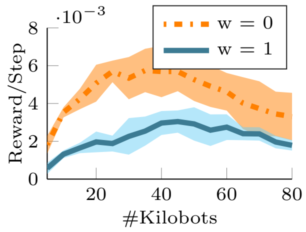
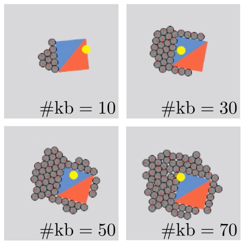


Figure 2: The value function plots around the object depict how our approach successfully adapts to different settings of translation-rotation-ratio w.



(a) Up to 50 agents, the performance benefits from the larger swarm size.



(b) If the swarm gets too big, it spreads around the object and disturbs the movement.

Figure 3: Performance of a policy learned with 15 individuals applied to different swarm sizes.

3. EXPERIMENTAL RESULTS

To conduct our experiments, we developed a 2D Kilobot simulator based on Box2D [1]. By applying a heuristic sampling strategy for the initial state of the kilobots, we assure a stable learning process. We learned the object movement policy with 15 Kilobots for a square object over 20 iterations. Figure 2 shows the value functions for three different settings of the translation-rotation ratio w after 15 iterations.

To evaluate how well the learned policy generalizes to different swarm sizes, we applied policies learned with 15 individuals to different numbers of Kilobots. Figure 3a shows how the average reward per step benefits from a growing swarm size up to about 50 individuals. As shown in Figure 3b, the reward drops with a bigger swarm size since the individuals start to disperse around the object and disturb or block the movement.

In combination with hand-coded assembly policies and the path planning strategies described above, we applied the learned object movement policies to different assembly tasks. The top row in Figure 1 depicts the assembly process of composing a larger square out of four small squares. In a second task, the kilobots had to assemble a C-shaped object and a T-shaped object. This process is depicted in the bottom row of Figure 1. This second task also demonstrates that the learned object movement policies—to a certain extend—can be transferred to different object shapes.

REFERENCES

[1] Box2d, a 2d physics engine for games. http://box2d.org/.

[2] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, pages 337–404, 1950.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[4] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[5] J. Peters, K. Mülling, and Y. Altun. Relative entropy policy search. In *AAAI*. Atlanta, 2010.

[6] M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera, and R. Nagpal. Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robotics and Autonomous Systems*, 62(7):966–975, 2014.

[7] M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal. Collective transport of complex objects by simple robots: Theory and experiments. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, 2013.

[8] A. Smola, A. Gretton, L. Song, and B. Schölkopf. A hilbert space embedding for distributions. In *In Algorithmic Learning Theory: 18th International Conference*, pages 13–31. Springer-Verlag, 2007.

[9] C. Wirth, J. Fürnkranz, and G. Neumann. Model-free preference-based reinforcement learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.