Toward Effective Soft Robot Control via Reinforcement Learning

Haochong Zhang[1], Rongyun Cao[1], Shlomo Zilberstein[2], Feng Wu[1], and Xiaoping Chen[1](✉)

[1] University of Science and Technology of China, Hefei 230027, Anhui, China
{solomonz,ryc}@mail.ustc.edu.cn, {wufeng02,xpchen}@ustc.edu.cn
[2] University of Massachusetts Amherst, Amherst, MA 01003-9264, USA
shlomo@cs.umass.edu

Abstract. A soft robot is a kind of robot that is constructed with soft, deformable and elastic materials. Control of soft robots presents complex modeling and planning challenges. We introduce a new approach to accomplish that, making two key contributions: designing an abstract representation of the state of soft robots, and developing a reinforcement learning method to derive effective control policies. The reinforcement learning process can be trained quickly by ignoring the specific materials and structural properties of the soft robot. We apply the approach to the Honeycomb PneuNets Soft Robot and demonstrate the effectiveness of the training method and its ability to produce good control policies under different conditions.

Keywords: Soft robot control · Reinforcement learning · PneuNets

1 Introduction

Recently, soft robotics have attracted growing attention from multiple disciplines including robotics, materials science, bionics, and AI. In contrast to hard-bodied robots [4,15], soft robots are made of soft and/or extensible materials from the surface to the motion mechanism. Soft robots have shown remarkable potential in realizing some performance which conventional rigid robots can hardly perform even after several decades of research. For example, a soft robot has an instinctive advantage on grabbing irregular, deformable or fragile objects, which is needed absolutely in domestic services. A soft robot can also fulfill manipulation friendly and safely when it works closely with humans, which is also beyond the capabilities of conventional service robots [14,19,22]. However, unlike rigid robots, which can be easily and accurately modeled and controlled using mature theories and methodologies, modeling and controlling of soft robots present new challenges to AI and robotics [11,23].

While softness can be an advantage of typical soft robots [14], it also makes them particularly vulnerable to a variety of environmental impacts [25]. In domestic service scenarios, a soft robot may interact with a wide variety of

Y. Huang et al. (Eds.): ICIRA 2017, Part I, LNAI 10462, pp. 173–184, 2017.
DOI: 10.1007/978-3-319-65289-4_17

objects in the environment, where some of the information of these objects and the robot itself cannot be effectively perceived. For example, hardness or weight of an object that the robot tries to pick up can potentially interfere with the interaction. As a result, it will seriously affect the behavior and control of the robot. Therefore, to improve the robustness of soft robot control, especially under the insufficient perception, is critical for the development of soft robots.

Even when all the information about the environment and the soft robot itself is provided, building an effective model for a particular task is still a substantial challenge [21]. It may involve very complex physical mechanics analysis that can only be established for a specific task [26]. With small changes to the task or environment, the model may be significantly different. To date, there is no principled way for modeling soft robots due to the diversity of material and structure (e.g., pneumatic, hydraulic, cables, electro-active polymers) [14]. Moreover, complex mechanics geometric models and simulator for soft robots are often difficult to adequately reflect the real characteristics of the hardware [8]. Various details that are ignored by the model tend to introduce hurdles that compromise performance [6].

To address these challenges, we propose a simple, effective and integrated reinforcement learning (RL) framework into soft robotics for the soft robot control problem. In contrast to other approaches for controlling soft robots [22], our approach has the advantage of ignoring the specific properties of the materials and partial structural characteristics of soft robots, thereby simplifying the modeling task. Besides, since we apply reinforcement learning directly to the soft robot hardware, the various features of the soft robot can also be directly reflected in the results of reinforcement learning. Hence, we can obtain a soft robot control policy that is well aligned with the actual hardware.

Although reinforcement learning has substantial advantages over existing methods, it also presents some challenges.

Firstly, reinforcement learning generally requires that the target problem be abstracted as a Markov decision process (MDP) [1]. Therefore, how to derive a set of representations that can adequately reflect the nature of soft robots, but also facilitate the use of reinforcement learning algorithms is the first problem we face. In this paper, we introduce a class of abstract representations of soft robot control problems in the representation part.

Secondly, in the training part, the performance of reinforcement learning is often dependent on an effective exploration of the problem space. Due to the presence of actions with infinite degrees of freedom and the continuous state space of soft robots [22], to obtain an accurate result, a large-scale search effort is necessary. Nonetheless, the possible loss of the robot hardware and the cost of time limit the scale of training. And since we anticipate the tasks to be handled by soft robots to become progressively more complicated, even though the performance of soft robots continues to improve, the above costs cannot be ignored. And while simulators have been used in the past to address that challenge, developing a realistic simulator of our soft robot is extremely difficult. This prevents us from relying on simulators for training. To address this, we use a

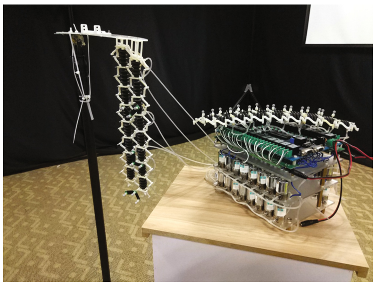

Fig. 1. Platform with soft robot arms, control circuit, air pumps and valves.

combined simulation and physical two-step training to enhance the performance of our resource-bounded physical robot training.

Intuitively, training results must be efficiently performed to ensure the overall system performance. In the experiments, we demonstrate that different execution conditions have significant impacts on the performance of the policy. This has greatly affected the effectiveness of reinforcement learning in our setting. In the execution part, we explore the possibility of policy open-loop and closed-loop execution and carefully evaluate the performance under different conditions. In order to perform open-loop control, we use the simulator to maintain an internal state to implement a "pseudo-closed-loop" control. And we rely on external sensors to achieve closed-loop control.

The remainder of this paper is organized as follows. In Sect. 2, we begin with the reinforcement learning implementation with the state representation, the assumptions and the algorithm in details. In Sect. 3, we implement and test the methods we discussed above on a physical soft robot platform to comprehensively show the performance of the methods. Section 4 provides an overview of the related work. Section 5 concludes the paper and summaries the contribution and future work.

2 Effective Soft Robot Control

As shown in Fig. 1, we use a soft robot constructed with honeycomb pneumatic network (HPN) [24]. It has many advantages, such as structural stability, flexibility, and crush-resistance. The structure of the soft robot makes it possible to carry out large-scale and powerful motion on a light self weight. Here, we try to achieve effective control of the soft robot arm. The difference between this problem with general soft robot control is that we are more concerned with the position of the soft robot arm end coordinates and the policy performance under environmental influences.

We ensure that our method is applicable to other piecewise soft robots. Under this assumption, the entire soft robot consists of relatively independent sections. Each section has the property of infinite degrees of freedom. An advantage of a piecewise soft robot is that it is easy to expand, for example by adding new sections to the end of the soft robot (Figs. 2 and 3).

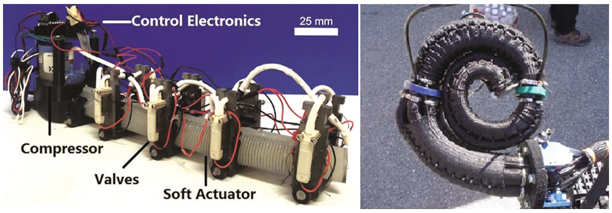

Fig. 2. Some other examples of piecewise soft robots [16,17].

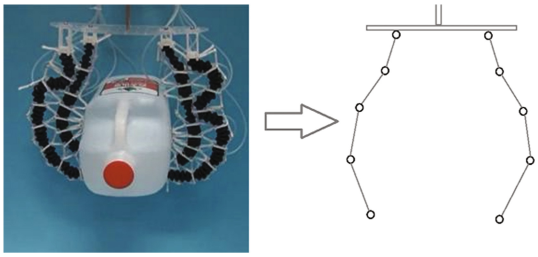

Fig. 3. Illustration of the state representation. The states consist of spatial coordinates of each point in the illustration. Each arm state has five coordinates. Section 3 describes the specific method to obtain states in real time.

2.1 The MDP Representation

To apply reinforcement learning, the first step is to abstract the problem and represent it as a Markov decision process (MDP). We describe below the fundamental MDP elements, including states (S), actions (A), and the reward function ($R(s, a, s')$).

States. A convenient representation of the state space should be able to adequately reflect the configuration of the soft robot, and must also be easy to manipulate. To balance these two factors, we use a set of predetermined points to represent the state of the piecewise soft robot. Based on the piecewise property, the state representation of the entire soft robot is the combination of the states of each section.

We use three coordinates to represent a section state, respectively, the beginning, the center and the end. The start and end points of the two adjacent sections are overlapped. So, for a soft robot connected by n sections, each state contains $n * 2 + 1$ continuous space coordinates p_n.

$$\begin{aligned} State &= \{s_1, s_2, s_3, ...\} \\ \forall s &\in State \\ s &= \{p_1, p_2, p_3, ...\} \\ &= \{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), ...\} \end{aligned} \tag{1}$$

Actions. Action abstraction is an essential expression of soft robot characteristics. Here we are also faced with a dilemma: a too simple abstraction will severely limit the capacity of soft robots, while an overly complex abstraction could compromise the model's ease of use and versatility.

Therefore, in combination with our observation of a large number of soft robots [14], we define the action as the movement of one of the motion section in discrete directions.

$$Action = \{left, right, long, short, ...\} \times \{all\ sections\} \tag{2}$$

This is an abstraction of action, not the action that can be used directly in the soft robot. For reinforcement learning, this is appropriate. But in order to implement on soft robots, we must define these actions integrated with the physical implementation of the soft robot. For cable-based soft robots, these actions mean different cable adjustments. And for pneumatic soft robot, these actions mean different air pressure adjustments. More specifically, we can increase or decrease each air bag air pressure to define motion section actions.

For a two motion mechanisms (M_a and M_b) motion section there are 4 actions: (1) **left**: M_a increases a unit pressure, M_b remains unchanged. (2) **right**: M_a remains unchanged, M_b increases a unit pressure. (3) **long**: M_a increases a unit pressure, M_b increases a unit pressure. (4) **short**: M_a decreases a unit pressure, M_b decreases a unit pressure.

In practice, "a unit pressure" may represent a unit of air pressure change, may also represent a unit of the inflating volume, but also may represent a period of inflation. The representation of an action is closely related to a particular physical implementation, in the last case "a unit" even means a unit of time.

Different action representations lead to different models. In either case, the problem of uncertainty needs to be dealt with. The uncertainty of the action is taken into account in the design of our algorithm, while the uncertainty of the observation is ignored.

Reward Function. The definition of a reward function depends on the task we are trying to complete. We define the reward function as a linear correlation function of the states in order to simplify the representation of the problem.

For example, we want to move the end of the soft robot to a specified target position p_T. We define the reward function $R(s, a, s')$ as:

$$\begin{aligned} R(s, a, s') &= distance(s, s_T) - distance(s', s_T) \qquad (3)\\ distance(s, s_T) &= |p_E - p_T| \\ &= |x_E - x_T| + |y_E - y_T| + |z_E - z_T| \end{aligned}$$

where p_E and p'_E is the end coordinate of the soft robot state s and s', s_T is the target coordinate.

2.2 The RL Algorithm

The designed learning procedure include two steps, simulation step and physical step. The main reason for doing so is to decrease physical learning cost and increase the quality of policy. Simulation trained policy can improve the efficiency of physical training. The mainly difference between these two steps is learning algorithm interaction with the simulator or real world.

Q-Learning. Based on above settings, we use function approximation Q-Learning [3] to train the control policy. Each episode, soft robot executes a sequence of actions from the initial state. On the basis of the above state, action, reward representation, we introduce a "final action" a_F. This action does not affect the shape of the robot, but indicating this episode will be stopped. The entire learning process consists of repeating episodes from the initial state to a_F.

Because of the continuous space and nonlinear setting of the problem, we map the state space into a high-dimensional linear space. This data structure can be seen as a neural network with only one hidden layer. We use the data structure to fit the $Q(s,a)$ function. The advantage of this approach lies in the simplicity of implementation and maintenance. But as the number of soft robot sections increases, the dimensions of the superposition space increase at a faster rate to keep performance.

From the definition of reward, we can see that if we do not have any *discount factor* γ or $\gamma = 1.0$, the intermediate state is not important. For this task, we want to perform as few steps action as possible. Therefore, we modify the Q function to be the mean value function associated with the step numbers.

$$\begin{aligned} Q(s_t,a_t) = Q(s_t,a_t) & \\ & +\alpha_t[\frac{1}{t+1}R(s_t,a_t,s_{t+1}) \\ & +\frac{t}{t+1}\max_a Q(s_{t+1},a) - Q(s_t,a_t)] \end{aligned} \tag{4}$$

With probability ϵ, the algorithm chooses an action at random and with probability $1-\epsilon$ choose an action $a = \arg\max_a Q(s,a)$.

Simulation Step. We developed the simulator based on the work of [6] to achieve the same effect as possible. Simulator parameters are adjusted empirically and the uncertainty of motion, hypothesis of normal distribution, is introduced. The observation of the states and the execution of the actions is relatively straightforward in the simulator.

We use the simulator to sample in the state space before reinforcement learning. We randomly execute actions and compute the Q value in each sample. By sampling the state space, we use the gradient descent algorithm to compute the weight of this network. We use the $Q(s,a)$ function trained in this way as the initial Q_0 in simulation step to reach better performance.

Physical Step. In addition to the soft robot experiment platform that we also use the OptiTrack[1] motion capture system (MCS) [5,7] as a visual sensor to observe soft robot state. Our method requires that states be fully observable, which

[1] http://optitrack.com/.

means that we ignore sensor errors. Experimental platform has two air pumps for pumping and inflatable. Air pumps with the valve opening and closing to achieve the actions we defined. More detailed information about the experimental platform will be described in the Sect. 3.

We use the simulation trained $Q(s, a)$ function as the initial Q_0 in the physical step. In addition, there are not any structural differences between the two steps in the algorithm. The main difference is the parameter setting and implementation of the algorithm.

It is also very important that we tried the reinforcement learning in physical environment only. But very unfortunately, after a long period of training, we still can not get the policy can be effectively executed. In a few cases, the policy may even fall into a non-stopable situation.

2.3 Execution and Control

How to execute effectively after learned a policy is also our main concern. We will discuss this issue from the perspective of closed-loop and open-loop (Fig. 4).

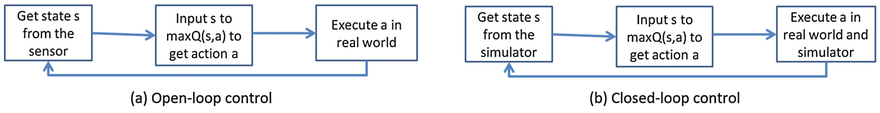
Get state s from the sensor
Input s to maxQ(s,a) to get action a
Execute a in real world
(a) Open-loop control
Get state s from the simulator
Input s to maxQ(s,a) to get action a
Execute a in real world and simulator
(b) Closed-loop control

Fig. 4. The primary difference between open-loop and closed-loop control. The closed-loop control is shown above, and the open-loop control is shown below.

Closed-loop control is relatively simple. Each time we obtain the current state of the soft robot from the actual sensor. We then input this state into the Q function to get and execute the optimal action of the policy (Figs. 5 and 6).

For open-loop control, we cannot observe the actual state of the soft robot directly. So, the simulator is used to maintain an internal state of the robot. For each execution cycle, we use the internal state as the state of the entity robot. And in each cycle, we execute the same action in both the simulation and the real world. It is easy to see that this situation lead to unavoidable errors. But for

Fig. 5. Soft Robot 3D prints hard components and EPDM (Ethylene Propylene Diene Monomer) airbags.

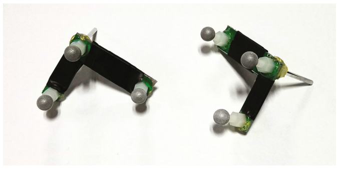

Fig. 6. The markers we put on the robot.

some scenarios where closed-loop control cannot be achieved, this compromise is worthwhile.

3 Experiments

3.1 Platform Setup

We use a 3-section soft robot arm to complete the validation of the experiment. In this experimental platform, we use two air pumps to complete the pumping and filling in each section. The control circuit of the valve enables us to achieve almost continuous action. Hence we limited each movement period to 100 ms.

The soft robot arm composed by honeycomb cells consists of soft actuator and variable hexagonal frame. This structure is a combination of soft and hard components. If stay in a relaxed state, the cell is approximately hexagonal. We use 3D printing to produce all the structural and hard components. The soft actuator is made of EPDM (Ethylene Propylene Diene Monomer) with high temperature vulcanization molding.

Another important motion error source is air-tightness of the air bag. We select the same initial state each time to reduce the impact of this error.

3.2 Policy Validation and Execution

We executed the policy we learned several times to verify the final performance. In particular, we tested the performance of different policies under different loads weight. More specifically, we selected 0 g, 10 g, 20 g, 50 g and 100 g. At each cycle, we get the current state of the soft robot arm from the MCS and then execute the action with the highest value in the RDF constructed Q function (Figs. 7 and 8).

Figure 9 shows the execution results of different environment (different load) and different experiment settings in 4 cases: (1) Use the simulator trained policy

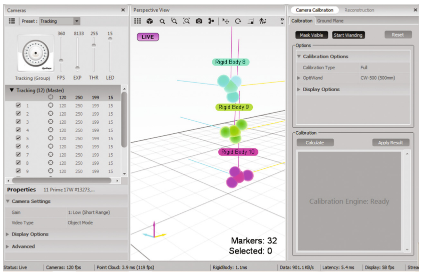

Fig. 7. This is how the soft robot looks like in motion capture system. Using this system, we can directly get the coordinates of each marker location.

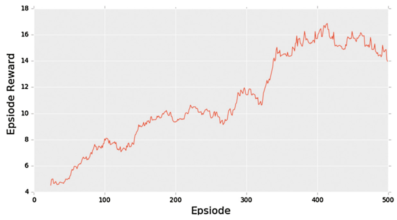

Fig. 8. Episode reward we get during 500 episodes in simulator.

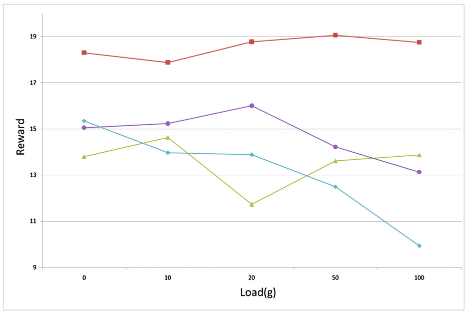

Fig. 9. The average reward of 10 times execution of different policies and different load. Which are using simulator trained policy only (●), simulator trained policy and MCS (◆), reality trained policy and MCS (▲), simulator, reality trained policy and MCS (■). Here, the reward is not Q value but the sum of the reward for each step. According to our manual measurement results, the theoretical maximum reward is about 20.15.

and maintains an internal state in simulator. (2) Use the simulator trained policy and sensors to acquire real states in real time. (3) 50 episodes reality environment trained policy and use sensors to acquire reality states in real time. (4) 50 episodes real environment trained policy based on simulator trained result and use sensors to acquire real states in real time.

In the first case above, we implement the open-loop control of the soft robot. The result is affected by 2 problems: the simulator are not fully accurate and internal state cannot effectively adapt the variety of load. Intuitively, we can not simply perform the same sequence of actions in different scenes.

In the cases 2, 3 and 4, we use the sensors to observe the soft robot's own state in real time to achieve a closed-loop control. In these cases, we improve the performance of policy execution in different loads.

In case 3, We tried to use physical robots directly to train the control policy. However, due to lack of training samples, in many cases the robot will execute a very strange action. In some cases, we observed that the execution process can not converge, so we must manually stop the execution of the policy (Fig. 10).

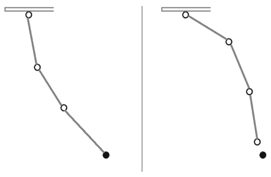

Fig. 10. Left side is what expect to achieve, i.e., the shortest sequence of actions and closest to the target. Right side shows what actually gets from our RL algorithm.

In case 4, the results are significantly better than the other cases. But compare to the theoretical maximum reward still have a certain gap. According to our analysis, this gap comes from the following factors: (1) Discretize actions cause a decrease in policy precision. (2) The learning result does not achieve the optimal solution. (3) The error of hardware of current soft robot experiment platform. More specifically, for our experimental platform, air tightness is the main source of error.

4 Related Work

Piecewise constant curvature kinematics, an approximation of the continuum and soft robot kinematics model, has been widely used in a variety of control tasks. [26] made a review of kinematic modeling for continuum robots. The use of complex curve modeling often leads problems to in sophisticated physical model analysis. But the complexity of the physical model has brought a wide range of adaptation challenges.

Because many biological systems consist of soft materials, biomimetics is often a motivation for the work on soft robots. Researchers have attempted to analysis and utilize a number of biological organs. For example, the robotic elephant trunk manipulator [10] and the OctArm continuum manipulator [18]. Based on an analysis of the morphological features of the octopus arm (connective tissue, arm density, transverse muscles, longitudinal muscles), Laschi [13], Mazzolai [17] and their groups designed the octopus soft robot arm.

To improve the accuracy of the kinematics model, [9] presented a feed-forward neural network learning method to deal with the inverse kinetics elements calculation for a cable-driven non-constant curvature soft manipulator. Because there are small differences between all the individual soft manipulators, they all had to be trained separately in order to represent the relation between the position of the robot tip and the cables' output force.

Kober and Peters [12] summaries the relationship between robotic and reinforcement learning. Several challenges like the curse of dimensionality, real-world, modeling and goal are mentioned. Moreover, MDP-based methods are widely used by other robot systems such as soccer [2,27] and disaster response [20,28].

5 Conclusion and Future Work

This paper presents a new approach for the control of a piecewise soft robot. Through the model-free reinforcement learning, we finally get the action policy for a specific task. In order to overcome the limitations of soft robot hardware, we use the simulator and physical two-step learning framework to improve the policy performance in the case of limited physical learning samples. After the policy is obtained, we use motion capture system to implement the soft robot closed-loop control, thereby enhancing the soft robot's adaptability to the environment. And we using the simulator to maintain an internal state to implement open-loop control. All the above ideas have been Implemented and verified in a physical soft robot platform.

There are many future work for soft robot control via reinforcement learning. Firstly, the value function or policy migration between the simulation and the reality needs a more solid theoretical guidance, analysis and proof. Secondly, more complex and diversified reinforcement learning methods have the potential to improve system performance. Finally, this approach can be extended to more complex forms and tasks such as obstacle avoidance and locomotion.

Acknowledgments. Feng Wu was supported in part by National Natural Science Foundation of China (No. 61603368), the Youth Innovation Promotion Association of CAS (No. 2015373), and Natural Science Foundation of Anhui Province (No. 1608085QF134).

References

1. Bai, A., Wu, F., Chen, X.: Bayesian mixture modelling and inference based thompson sampling in monte-carlo tree search. In: Proceedings of NIPS, pp. 1646–1654 (2013)
2. Bai, A., Wu, F., Chen, X.: Online planning for large markov decision processes with hierarchical decomposition. ACM Trans. Intell. Syst. Technol. **6**(4) (2015). Article No. 45
3. Baird, L., et al.: Residual algorithms: Reinforcement learning with function approximation. In: Proceedings of ICML, pp. 30–37 (1995)
4. Chen, Y., Wu, F., Shuai, W., Wang, N., Chen, R., Chen, X.: Kejia robot - an attractive shopping mall guider. In: Proceedings of ICSR, pp. 145–154 (2015)
5. Chen, Y., Wu, F., Wang, N., Tang, K., Cheng, M., Chen, X.: KeJia-LC: a low-cost mobile robot platform - champion of demo challenge on benchmarking service robots at RoboCup 2015. In: Proceedings of RoboCup, vol. 9513, pp. 60–71 (2015)
6. Cheng, B., Sun, H., Chen, X.: Evolving honeycomb pneumatic finger in bullet physics engine. Robot Intell. Tech. App. **3**, 411–423 (2015)
7. Cheng, M., Chen, X., Tang, K., Wu, F., Kupcsik, A., Iocchi, L., Chen, Y., Hsu, D.: Synthetical benchmarking of service robots: a first effort on domestic mobile platforms. In: Almeida, L., Ji, J., Steinbauer, G., Luke, S. (eds.) RoboCup 2015. LNCS (LNAI), vol. 9513, pp. 377–388. Springer, Cham (2015). doi:10.1007/978-3-319-29339-4_32
8. Duriez, C.: Control of elastic soft robots based on real-time finite element method. In: Proceedings of ICRA, pp. 3982–3987 (2013)
9. Giorelli, M., Renda, F., Ferri, G., Laschi, C.: A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space. In: Proceedings of IROS, pp. 5033–5039 (2013)
10. Hannan, M.W., Walker, I.D.: Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots. J. Robotic Syst. **20**(2), 45–63 (2003)
11. Inoue, T., Hirai, S.: Modeling of soft fingertip for object manipulation using tactile sensing. In: Proceedings of IROS, pp. 2654–2659 (2003)
12. Kober, J., Peters, J.: Reinforcement learning in robotics: a survey. In: Reinforcement Learning, pp. 579–610 (2012)

13. Laschi, C., Cianchetti, M., Mazzolai, B., Margheri, L., Follador, M., Dario, P.: Soft robot arm inspired by the octopus. Adv. Robot. **26**(7), 709–727 (2012)
14. Laschi, C., Mazzolai, B., Cianchetti, M.: Soft robotics: technologies and systems pushing the boundaries of robot abilities. Sci. Robot. (2016)
15. Lu, D., Zhou, Y., Wu, F., Zhang, Z., Chen, X.: Integrating answer set programming with semantic dictionaries for robot task planning. In: Proceedings of IJCAI (2017)
16. Luo, M., Pan, Y., Skorina, E.H., Tao, W., Chen, F., Ozel, S., Onal, C.D.: Slithering towards autonomy: a self-contained soft robotic snake platform with integrated curvature sensing. Bioinspir. Biomim. **10**(5), 055001 (2015)
17. Mazzolai, B., Margheri, L., Cianchetti, M., Dario, P., Laschi, C.: Soft-robotic arm inspired by the octopus: II. From artificial requirements to innovative technological solutions. Bioinspir. Biomim. **7**(2), 025005 (2012)
18. McMahan, W., Chitrakaran, V., Csencsits, M., Dawson, D., Walker, I.D., Jones, B.A., Pritts, M., Dienno, D., Grissom, M., Rahn, C.D.: Field trials and testing of the OctArm continuum manipulator. In: Proceedings of ICRA, pp. 2336–2341 (2006)
19. Pfeifer, R.: soft robotics - the next generation of intelligent machines. Invited talk on IJCAI (2013)
20. Ramchurn, S.D., Huynh, T.D., Wu, F., Ikuno, Y., Flann, J., Moreau, L., Fischer, J.E., Jiang, W., Rodden, T., Simpson, E., Reece, S., Roberts, S., Jennings, N.R.: A disaster response system based on human-agent collectives. J. Artif. Intell. Res. **57**, 661–708 (2016)
21. Renda, F., Giorelli, M., Calisti, M., Cianchetti, M., Laschi, C.: Dynamic model of a multibending soft robot arm driven by cables. IEEE Trans. Robot. **30**(5), 1109–1122 (2014)
22. Rus, D., Tolley, M.T.: Design, fabrication and control of soft robots. Nature **521**(7553), 467–475 (2015)
23. Shibata, M., Hirai, S.: Soft object manipulation by simultaneous control of motion and deformation. In: Proceedings ICRA, pp. 2460–2465 (2006)
24. Sun, H., Chen, X.-P.: Towards honeycomb pneunets robots. In: Kim, J.-H., Matson, E.T., Myung, H., Xu, P., Karray, F. (eds.) Robot Intelligence Technology and Applications 2. AISC, vol. 274, pp. 331–340. Springer, Cham (2014). doi:10.1007/978-3-319-05582-4_28
25. Tolley, M.T., Shepherd, R.F., Mosadegh, B., Galloway, K.C., Wehner, M., Karpelson, M., Wood, R.J., Whitesides, G.M.: A resilient, untethered soft robot. Soft Robot. **1**(3), 213–223 (2014)
26. Webster, R.J., Jones, B.: Design and kinematic modeling of constant curvature continuum robots: a review. Int. J. Robot. Res. **29**(13), 1661–1683 (2010)
27. Wu, F., Chen, X.: Solving large-scale and sparse-reward DEC-POMDPs with correlation-MDPs. In: Proceedings of RoboCup, pp. 208–219 (2007)
28. Wu, F., Ramchurn, S., Chen, X.: Coordinating human-UAV teams in disaster response. In: Proceedings of IJCAI, pp. 524–530 (2016)