DOI: 10.2507/32nd.daaam.proceedings.090

EXPLAINABLE ARTIFICIAL INTELLIGENCE FOR ROBOT ARM CONTROL

Simon Schwaiger, Mohamed Aburaia, Ali Aburaia & Wilfried Woeber

This Publication has to be referred as: Schwaiger, S[imon]; Aburaia, M[ohamed]; Aburaia, A[li] & Woeber, W[ilfried] (2021). Explainable Artificial Intelligence For Robot Arm Control, Proceedings of the 32nd DAAAM International Symposium, pp.0640-0647, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-33-4, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/32nd.daaam.proceedings.090

Abstract

In this paper, we investigate reinforcement learning model explainability through a pick and place task. Two robots with three degrees of freedom learned to solve the pick and place task in simulation as well as reality. To investigate the explanatory factors implicitly learned by the models, we derive robot parameters, i.e., the length of the robot segments. To overcome the black box nature of reinforcement learning models and provide a physical explanation of the results, the robot dimensions are derived from the learned reinforcement learning model and compared to the real dimensions. The hypothesis in the presented work is that converged reinforcement learning models must learn the robot parameters implicitly in order to learn a task. This transforms black box models into white box models, where each model's decisions can be interpreted. Our experiments show that robot parameters can be derived from learned models and that the chosen reinforcement learning model implicitly learns physical context. In order to create robust and trustworthy AI systems for intelligent factories, we suggest that a physical interpretation of all black box models must be done.

Keywords: Explainable Artificial Intelligence; XAI; Reinforcement Learning; Movement Control; Parameter Extraction

1. Introduction

In automation, motion planning is required for robots to perform tasks. Motion planning algorithms depend on physical models of robots as well as knowledge about the environment, such as obstacles and the goal pose. Information of the robot model, such as robot type and configuration, the physical properties and transforms between the robot coordinate frames need to be known in order to enable adequate motion planning and execution.

Accurate motion planning is made increasingly difficult by unknown robot parameters as well as a changing environment, creating new problems that a robot needs to solve [1], [2]. While artificial intelligence (AI) based approaches, such as reinforcement learning (RL) [3], can be used to automate planning procedures and partially overcome these problems, the models resulting from RL methods are so-called black box models [4], which must be turned into white box models for explainable and reliable application. Creating such explanation through physical context, however, still poses an open problem [5]. Motivation for solving this problem comes from the dependency of machine learning algorithms on data representation [6] in order to be successful.

In AI, reaching a level of so-called "usable intelligence" [7] is done in multiple stages, with explanation of the data in context of an application domain especially being a challenge. In order to meet this challenge, the field of explainable AI (XAI) engages with the concept of the interpretability of AI, aiming to enable models to be understood by a human through explanation and interpretation [8]. Understanding the functionality of a model allows for justification of its decisions, control of model behaviour, improvement of model architecture and discovery of new strategies for problem solving [4]. In the context of robotics, XAI can be used for feasibility explainability of task- and motion planning [4], enabling inspection of possible solutions to planning problems.

This paper tackles the identification of latent explanatory factors implicitly learned by RL methodologies for robot path planning problems. We trained a three degree of freedom robotic arm to perform a task using reinforcement learning and extract robot parameters from the model during application on the real robot. To this end, we initially trained a model in simulation. Afterwards, the model is adopted based on information from the real-world during deployment on the real robot. Finally, after the learning procedure, we investigate the transition models and the robot's kinematics to derive the physical characteristics of the joints. We hypothesise that physical context, i.e., the robot's kinematic structure, is learned implicitly.

This study is structured as follows. Chapter 2 depicts the current state of the art regarding RL applications and provides an overview of strategies of creating XAI solutions. Chapter 3 describes used methods, including an introduction to the concept of RL, as well as calculation of the kinematic structure of the tested robots. In chapter 4, results of the experiments performed on both robots are presented and discussed in chapter 5. A summary of the contents of the paper is given in chapter 6 and an outlook on future work is provided.

2. State of the Art

Reinforcement learning engages with the process of mapping the state of a system to an action in order to maximise a numerical reward [3]. Tasks can be formulated as an interaction between an agent and an environment, called the Markov-Decision process [3]. The goal of RL is to find a policy that determines which action the agent should take in a certain state. This concept provides a computational approach to understanding and automating goal-centric learning and decision making [3]. While RL models based on policy gradients [9], [10] and deep Q-networks (DQN) [11] have been applied to benchmarks, such as Atari games [12], continuous control based on Deep Deterministic Policy Gradients (DDPG) [13] has been used for practical RL implementations, such as process control [14], robot arm control [2] and motion planning for industrial robots [15].

XAI is part of the third wave of AI with one of the objectives of this wave being the precise creation of algorithms that explain themselves [4]. Used approaches for explaining black box models either look into the model itself or assume the model to be complex, trying to explain behaviour without knowing and altering the inner works of the model [4]. The first category is especially suitable when a model is fundamentally designed to be explainable, however, the more complex a model is, the more difficult this kind of explanation is to achieve. Since increased model accuracy often requires complex prediction methods [16], this results in a trade-off between model interpretability and accuracy [17].

The interpretation can be either done in a global manner [18], [19], explaining the entire reasoning for a model's decisions, or locally, based on explanation of a single decision [20]. Local explanations are commonly used for understanding image classification models [21], where the area most influential for the model's decision is highlighted [22], [23]. However, few work is done in the context of XAI in robotics. Explanatory factor extraction for mobile robots based on sparse Gaussian processes is done in [24]. This study examines black box models using motion data for mobile robot control. Using kinematics, vehicle parameters are derived from the black box model, transforming the black box model into a white box model.

3. Methods

In order to achieve a computationally efficient system, a robot with three degrees of freedom was used in a discrete environment. The environment is assumed to be fully observable in order to enable the use of policy and value iteration for computing a policy that moves the robot's tool centre point (TCP) to a desired location, while avoiding collisions with itself and obstacles. The environment and problem are formulated in a universal way in order to enable the use of multiple robots as well as multiple reinforcement learning methods for training. The computed model is globally explained by calculating the kinematics from application of the model on the real robot, showing that the kinematics are implicitly learned.

This allows for the black box model to be turned into a white box model by disclosing the internal design and explain the model in physical context. Similarly to the methodology applied in [24], we use robot kinematics and trained RL movement models to derive robot parameters and thus unveil explanatory factors implicitly learned by the RL agent. Joint state data as well as the resulting TCP position are measured during application of the trained model on the robot. Based on this data, robot parameters representing the real robot are calculated using kinematics. A full overview of the resulting software solution is provided in Fig. 4.

3.1 Reinforcement Learning

For application of RL, the problem is formulated as a Markov Decision process [3]. Environment and agent are created based on a simulated robot (see Fig. 1). The environment is fully observable and discrete with the state being based on the robot's joint positions. A positive reward is given to the agent whenever the robot's TCP reaches the desired position and a negative reward is given when the robot collides with itself or any obstacle. A transition function is created based on the simulation, which maps the resulting next state for each possible action-state pair to the associated reward (success, failure or neutral). Based on this function, a policy is computed using policy and value iteration [3], [25], determining what action the agent takes in each state in order to maximise reward. The experiments are based on two robots, one of which has three rotational joints and the other with two rotational and one translational axes.

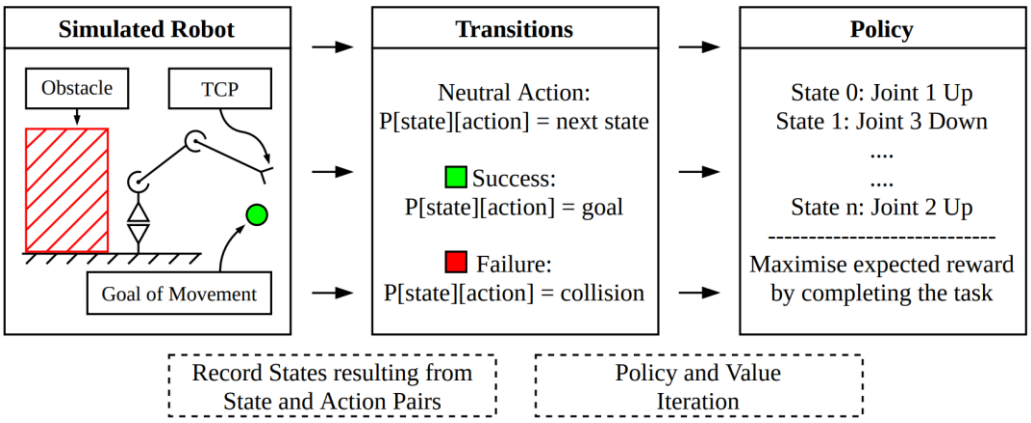

Fig. 1. The underlying structure of the RL model consists of transitions, created from a virtual version of the robot. The policy is computed based on these transitions with the goal of maximising reward. Reward is given upon the TCP reaching the desired destination, while a negative reward is given upon collisions.

3.2 Extraction of the first Kinematic Model

In order to show that the RL model implicitly learns the robot kinematics, robot parameters are calculated using TCP position and joint state data during application of the model on the real robot. Robot 1 is a three degree of freedom robotic arm with three rotatory axes based on the Hebi X-Series [26] platform (see Fig. 2). TCP displacement along the y-axis (when $\theta_1 = 0$) is combined into one metric, noted as o_1, resulting in four unknown robot parameters (l_1, l_2, l_3, o_1). The forward transform is not formulated based on the Denavit-Hartenberg notation [27], since the resulting equation system was not solvable due to the high complexity of the equations. Instead, the transform is established geometrically.

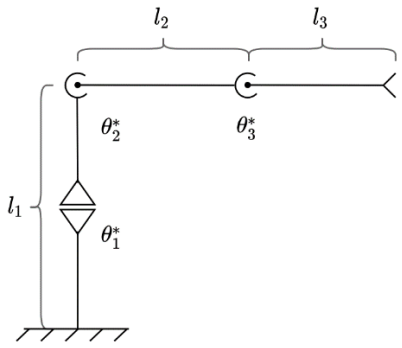

Fig. 2. Kinematic structure of the first robot.

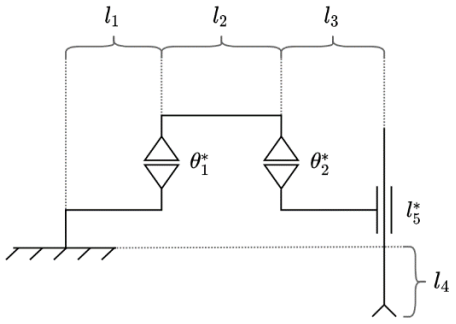

Fig. 3. Kinematic structure of the second robot.

Assuming $\theta_1 = 0$, terms for z and x-coordinates of an intermediary point P, which is offset by o_1 from the TCP position, are formulated ((2) and (1)). The resulting term for P_z is used directly, while the term for P_x is projected onto the xy-plane with the inclusion of θ_1. The resulting term (3) describes the coordinates of P. Since this offset is orthogonal to the first joint's x-axis, θ_1 can be used to project the offset onto the xy-plane and add it to the intermediary point.

$$P_x(\theta_1 = 0) = l_3 \cdot \cos(\theta_2 + \theta_3) + l_2 \cdot \cos(\theta_2) \quad (1)$$

$$P_z = l_1 + l_2 \cdot \sin(\theta_2) + l_3 \cdot (\theta_3 + \theta_2) \tag{2}$$

$$\begin{pmatrix} TCP_x \\ TCP_y \\ TCP_z \end{pmatrix} = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} + \begin{pmatrix} o_1 cos\left(\theta_1 + \frac{\pi}{2}\right) \\ o_1 sin\left(\theta_1 + \frac{\pi}{2}\right) \\ 0 \end{pmatrix} = \begin{pmatrix} cos(\theta_1)[l_3 cos(\theta_2 + \theta_3) + l_2 cos(\theta_2)] \\ sin(\theta_1)[l_3 cos(\theta_2 + \theta_3) + l_2 cos(\theta_2)] \\ l_1 + l_2 \cdot sin(\theta_2) + l_3 \cdot (\theta_3 + \theta_2) \end{pmatrix} + \begin{pmatrix} o_1 cos\left(\theta_1 + \frac{\pi}{2}\right) \\ o_1 sin\left(\theta_1 + \frac{\pi}{2}\right) \\ 0 \end{pmatrix} \tag{3}$$

When substituting the known joint states, (3) results in a linear system, allowing for a simple determination of the unknown variables, since o_1 can be calculated as the orthogonal distance between the first joint's x-axis and the TCP position. o_1, the joint states and the TCP positions can be substituted into the equation system, resulting in a system that has three unknown variables and three equations, making the system solvable.
In order to improve the reliability of the parameter calculation, the TCP position for $\theta_1 = \theta_2 = \theta_3 = 0$ (see (4)) is measured as well and substituted into one of the lines in (3), as seen in (5).

$$\theta_1, \theta_2, \theta_3 = 0 \rightarrow (TCP_x) = (l_3 + l_2) \tag{4}$$

$$\begin{pmatrix} TCP_x \\ TCP_x(\theta_1, \theta_2, \theta_3 = 0) \\ TCP_z \end{pmatrix} = \begin{pmatrix} cos(\theta_1)[l_3 \cdot cos(\theta_2 + \theta_3) + l_2 \cdot cos(\theta_2)] + o_1\left(\theta_1 + \frac{\pi}{2}\right) \\ l_3 + l_2 \\ l_1 + l_2 \cdot sin(\theta_2) + l_3 \cdot (\theta_3 + \theta_2) \end{pmatrix} \tag{5}$$

Robot parameters are calculated by first calculating o_1 as the orthogonal distance between TCP and the first joint's x-axis. The TCP positions are measured on the robot during operation, while θ_1, θ_2 and θ_3 are determined using the built-in feedback from the used servo motors. The joint states, as well as o_1 and the TCP positions are substituted into the equation system, resulting in a linear system, with three terms and three unknown variables. A linear solver is used to determine the parameters l_1 to l_3.

3.3 Extraction of the second Kinematic Model

Robot 2 is a three degree of freedom robot with two rotating and one translational axis based on the Makeblock smart servo [28] platform. In a similar manner to the first robot, the transform from base to TCP is calculated geometrically in order to obtain a linear equation system for determination of the robot parameters l_1 to l_4 (see Fig. 3). X and y-coordinates of the TCP are expressed based on positions of the first two joints. Due to the third joint being a translational axis, the TCP position along the z-axis is determined by the displacement of the robot's linear axis (l_5), as well as l_4. The transformation from base to TCP is depicted in (6).

$$\begin{pmatrix} TCP_x \\ TCP_y \\ TCP_z \end{pmatrix} = \begin{pmatrix} l_3 \cdot cos(\theta_1 + \theta_2) + l_2 \cdot cos(\theta_1) + l_1 \\ l_3 \cdot sin(\theta_1 + \theta_2) + l_2 \cdot sin(\theta_1) \\ l_4 + l_5 \end{pmatrix} \tag{6}$$

Since the robot has four unknown parameters, (6) is not solvable without additional information. Therefore, the TCP position's x-coordinate is measured for $\theta_1 = \theta_2 = l_5 = 0$ and formulated as seen in (7). In combination with the forward transform, this results in a solvable linear equation system (8) that allows for calculation of the robot parameters (l_1 to l_4).

$$\theta_1, \theta_2 = 0 \rightarrow (TCP_x) = (l_3 + l_2 + l_1) \tag{7}$$

$$\begin{pmatrix} TCP_x \\ TCP_y \\ TCP_z \\ TCP_x(\theta_1, \theta_2, l_5 = 0) \end{pmatrix} = \begin{pmatrix} l_3 \cdot cos(\theta_1 + \theta_2) + l_2 \cdot cos(\theta_1) + l_1 \\ l_3 \cdot sin(\theta_1 + \theta_2) + l_2 \cdot sin(\theta_1) \\ l_4 + l_5 \\ l_3 + l_2 + l_1 \end{pmatrix} \tag{8}$$

The robot parameters are calculated by measuring TCP positions, as well as θ_1, θ_2 and l_5 on the robot during operation and substituting them into (8). The resulting equation system is then solved for the robot parameters using a linear solver.

3.4 Used Software Packages

The robot operating system (ROS Noetic) [29] provides communication between different software components and manages coordinate frame transformations. The built-in software Rviz and the Rviz Visual Tools plugin [30] are used for visualisation of the virtual robot's training procedure.

OpenAI Gym [31] is used as the base framework for reinforcement learning since it provides a standardised way of formulating the RL problem as well as an interface between agent and environment. Even though the OpenAI Gym software offers multiple implemented environments, a custom environment for controlling the tested robots was implemented. Due to the standardised nature of the OpenAI Gym framework, any reinforcement learning algorithm written for discrete OpenAI Gym environments can be used with the implemented software solution.

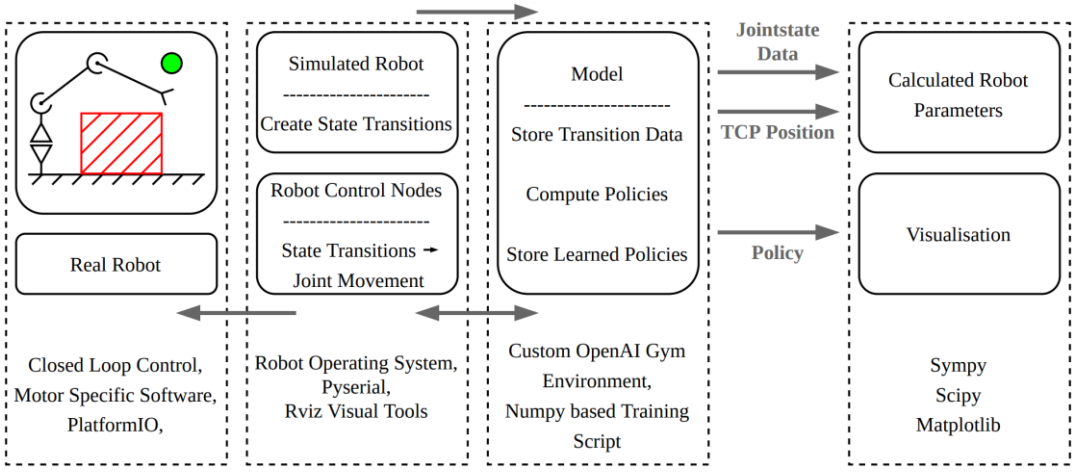

Fig 4. Multiple openly available software packages have been used for implementation of the system.

Since robot 1 is based on the HEBI-Robotics X-series platform [26], the provided Hebi ROS, C++ and Python libraries are used for robot control. Robot 2 is based on the Makeblock platform. Therefore, the Makeblock Servo Library [28], Pyserial [32] and PlatformIO [33] are used to control the real robot from the RL application. Sympy, a symbolic mathematics library for Python [34] and the linear algebra module of Scipy [35] are used for calculating the robot kinematics. Data is stored and handled using Numpy [36].

4. Experimental Results

In order to measure the error introduced by the robot parameter calculation, a pick and place task was learned in simulation on both robots. Simulated joint state and TCP position data was sampled ten times during operation at different robot poses. Based on this data, the robot parameters were calculated for each pose based on the kinematics presented in chapter 3. The calculated robot parameters are compared to the actual parameters used for simulation, in order to determine the accuracy of the implemented kinematics solution. Table 1 shows the parameters of the simulated robot, as well as average discrepancy between the calculated and real robot parameters in absolute and percent values.

Robot 1				
	l1	**l2**	**l3**	**o1**
Robot len.[m]	0.086	0.175	0.252	0.043
Avg. dev.[m]	0.00014	0.00043	0.00043	1.48E-05
Avg. dev.[%]	0.16311	0.2469	0.17115	0.03481

Robot 2				
	l1	**l2**	**l3**	**l4**
Robot len.[m]	0.047	0.093	0.073	0.042
Avg. dev.[m]	1.65E-16	2.62E-16	1.26E-16	1.39E-17
Avg. dev.[%]	3.51E-13	2.82E-13	1.73E-13	3.30E-14

Table 1. Data from simulations of both robots was used to measure the average error introduced by calculation of the robot parameters.

In order to show the system's ability to correct an inaccurate virtual robot model, a simulation was set up for both robots with robot parameters that intentionally deviate from the real robots. The use case depicted by this experiment intends for a model of a robot with partially unknown parameters to be trained in an inaccurate simulation. The model can then be corrected using data from the real robot, gained during deployment.

Point	Joint State			TCP pos.[m]		
P1	0.229	0.324	0.109	1.06	0.59	-0.88
P2	0.222	-0.234	0.11	-0.68	1.02	-1.52
P3	0.17	0.393	0.107	1.26	0.29	-0.4

Point	Robot Param.			
P1	0.0862	0.1747	0.2603	0.0414
P2	0.0855	0.175	0.26	0.0424
P3	0.0859	0.1739	0.2611	0.0417
Actual	0.086	0.175	0.26	0.0425

Table 2. Calculated robot parameters of robot 1 for the measured points. TCP_x (θ_1, θ_2, θ_3=0) was measured as 0.435m.

Three TCP positions (P_1, P_2, P_3) were measured by hand on each robot and used with the respective joint state data to calculate the robot parameters. The TCP coordinates and corresponding joint state data, as well es the calculated and actual robot parameters are depicted in in tables 2 and 3 for each respective robot. Based on the data in tables 2a and 3a, robot parameters were calculated using the kinematics presented in chapter 3. The resulting robot parameters as well as the actual parameters used for training an accurate representation of the real robots are depicted in tables 2b and 3b.

Point	Joint State			TCP pos.[m]		
P1	-0.5061	0.9599	-0.048	0.194	-0.011	-0.09
P2	0.6283	0.6283	-0.043	0.145	0.124	-0.0855
P3	-0.8029	-1.1694	0	0.083	-0.135	-0.0425

Point	Robot Param.			
P1	0.0469	0.0908	0.0753	-0.042
P2	0.0469	0.0936	0.0726	-0.0425
P3	0.0457	0.0947	0.0727	-0.0425
Actual	0.047	0.093	0.073	-0.0425

Table 3. Calculated robot parameters of robot 2 for the measured points. TCP_x (θ_1, θ_2=0) was measured as 0.213m.

5. Discussion

The presented system is able to train a model based on a simulated robot and use data from deployment on the real robot to correct the trained model. Table 1 shows that extraction of the robot parameters introduces an absolute error in the range of (3.3E−14m; 1.4E10−4m). This error, however, is smaller than other possible sources of inaccuracy (e.g., inaccurate manual measuring of the TCP positions or deviations resulting from closed loop motor control). The accuracy achieved in the practical implementation of the system allows for the completion of a pick and place task using robot 1. The robot trajectory is planned by the RL agent, who is able to adapt to different obstacles that need to be avoided. In practice, this allows for the system to be flexible in terms the surroundings of deployment as well as in terms of the problem that needs to be solved. Thus, we accept our hypothesis since the reinforcement learning model did learn the kinematic structure implicitly.

6. Summary and Outlook

In this paper, a system for autonomously fulfilling a pick and place task of three degree of freedom robots was implemented and tested. The system is capable of determining the robot's trajectory using reinforcement learning (RL), reaching the desired TCP position while avoiding obstacles as well as correcting the learned model by means of explainable artificial intelligence (XAI). This allows for a model to be learned in simulation and corrected afterwards, using data gained from the real world in order to achieve a model that is accurate to the real robot. The system was tested on two robots of different configurations, in simulation and reality. Future work on application of XAI on robot control is possible regarding multiple aspects of XAI due to the novelty of the topic. Explanation techniques, such as the one presented in this paper, are inherently only able to explain a certain type of model.

A generalised approach for explanation of robot behaviour still poses an open problem [4]. Furthermore, the trade-off between model accuracy and model explainability [17] still prevents models that are not designed to be explainable from being fully interpretable.

7. Acknowledgments

This work was supported by the city of Vienna (MA23 – Economic Affairs, Labour and Statistics) through the research project AIAV (MA23 project 26-04).

8. References

[1] Vecerik, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T. and Riedmiller, M. (2017). Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. arXiv preprint arXiv:1707.08817.

[2] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P. and Zaremba, W. (2017). Hindsight experience replay. arXiv preprint arXiv:1707.01495.

[3] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction, A Bradford Book, 978-0-262-03924-6, Cambridge, MA, USA.

[4] Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE access, 6, 52138-52160.

[5] Pearl, J., & Mackenzie, D. (2018). The book of why: the new science of cause and effect. Basic books, 978-0-465-09760-9, New York, NY, USA.

[6] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence, 35(8), 1798-1828.

[7] Holzinger, A. (2018). From machine learning to explainable AI. In 2018 world symposium on digital intelligence for systems and machines (DISA) (pp. 55-66). IEEE.

[8] Anjomshoae, S., Najjar, A., Calvaresi, D., & Främling, K. (2019). Explainable agents and robots: Results from a systematic literature review. In 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019 (pp. 1078-1088). International Foundation for Autonomous Agents and Multiagent Systems.

[9] Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems (pp. 1057-1063).

[10] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014, January). Deterministic policy gradient algorithms. In International conference on machine learning (pp. 387-395). PMLR.

[11] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.

[12] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 .

[13] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[14] Spielberg, S., Tulsyan, A., Lawrence, N. P., Loewen, P. D., & Gopaluni, R. B. (2020). Deep Reinforcement Learning for Process Control: A Primer for Beginners. arXiv preprint arXiv:2004.05490.

[15] Meyes, R., Tercan, H., Roggendorf, S., Thiele, T., Büscher, C., Obdenbusch, M., Brecher, C., Jeschke, S. and Meisen, T. (2017). Motion planning for industrial robots using reinforcement learning. Procedia CIRP, 63, 107-112.

[16] Breiman, L. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). Statistical science, 16(3), 199-231.

[17] Sarkar, S., Weyde, T., Garcez, A., Slabaugh, G. G., Dragicevic, S., & Percy, C. (2016). Accuracy and interpretability trade-offs in machine learning applied to safer gambling. In CEUR Workshop Proceedings (Vol. 1773). CEUR Workshop Proceedings.

[18] Yang, C., Rangarajan, A., & Ranka, S. (2018). Global model interpretation via recursive partitioning. In 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS) (pp. 1563-1570). IEEE.

[19] Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N. and Lee, S.I. (2020). From local explanations to global understanding with explainable AI for trees. Nature machine intelligence, 2(1), 56-67.

[20] Ljubobratovic, D., Guoxiang, Z., Brkic Bakaric, M., Jemric, T., & Matetic, M. (2020). Predicting Peach Fruit Ripeness Using Explainable Machine Learning, Proceedings of the 31st DAAAM International Symposium, pp.0717-0723, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-29-7, ISSN 1726-9679, Vienna, Austria

[21] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

[22] Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K. R. (2010). How to explain individual classification decisions. The Journal of Machine Learning Research, 11, 1803-1831.
[23] Woods, W., Chen, J., & Teuscher, C. (2019). Adversarial explanations for understanding image classification decisions and improved neural network robustness. Nature Machine Intelligence, 1(11), 508-516.
[24] Wöber, W., Novotny, G., Mehnen, L., & Olaverri-Monreal, C. (2020). Autonomous Vehicles: Vehicle Parameter Estimation Using Variational Bayes and Kinematics. Applied Sciences, 10(18), 6317.
[25] Russell, S., & Norvig, P. (2002). Artificial intelligence: a modern approach, Prentice Hall Press, 978-0-13-604259-4, One Lake Street Upper Saddle River, NJ, USA.
[26] https://www.hebirobotics.com/, (2021). HEBI Robotics, X-series actuator, Accessed on: 2021-10-15
[27] Angeles, J., & Angeles, J. (2002). Fundamentals of robotic mechanical system, Springer Verlag, 978-3-319-01850-8, NY, USA.
[28] https://github.com/Makeblock-official/Makeblock-Libraries, (2020). Makeblock-official, Makeblock-libraries: Arduino library for makeblock electronic modules, Accessed on: 2021-10-15
[29] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A.Y. (2009). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5).
[30] https://github.com/PickNikRobotics/rviz visual tools, (2020). PickNikRobotics, Rviz visual tools: C++ api wrapper for displaying shapes and meshes in rviz, Accessed on: 2021-10-15
[31] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. ArXiv Preprint ArXiv:1606.01540.
[32] https://pypi.org/project/pyserial, (2020). C. Liechti, pyserial 3.4, Accessed on 2021-10-15
[33] https://platformio.org/, (2020), PlatformIO labs, A professional collaborative platform for embedded development platformio, Accessed on: 2021-10-15
[34] Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, Rocklin M, Kumar A, Ivanov S, Moore JK, Singh S, Rathnayake T, Vig S, Granger BE, Muller RP, Bonazzi F, Gupta H, Vats S, Johansson F, Pedregosa F, Curry MJ, Terrel AR, Roučka Š, Saboo A, Fernando I, Kulal S, Cimrman R, Scopatz A. (2017) SymPy: symbolic computing in Python. PeerJ Computer Science, 3, e103, https://doi.org/10.7717/peerj-cs.103
[35] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J. and Van Der Walt, S.J. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature methods, 17(3), 261-272.
[36] Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J. and Kern, R. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.