Transferring Human Manipulation Knowledge to Robots with Inverse Reinforcement Learning

Hansen, Emil Blixt; Andersen, Rasmus Eckholdt; Madsen, Steffen; Bøgh, Simon

Published in:
Proceedings of the 2020 IEEE/SICE International Symposium on System Integration, SII 2020

DOI (link to publication from Publisher):
10.1109/SII46433.2020.9025873

Publication date:
2020

Document Version
Early version, also known as pre-print

Transferring Human Manipulation Knowledge to Robots with Inverse Reinforcement Learning*

Emil Blixt Hansen[1,2], Rasmus Eckholdt Andersen[1], Steffen Madsen[1], and Simon Bøgh[1]

***Abstract*—The need for adaptable models, e.g. reinforcement learning, have in recent years been more present within the industry. In this paper, we show how two versions of inverse reinforcement learning can be used to transfer task knowledge from a human expert to a robot in a dynamic environment. Moreover, a second method called Principal Component Analysis weighting is presented and discussed. The method shows potential in the use case but requires some more research.**

I. INTRODUCTION

The manufacturing industry has seen an increased demand for higher customisable products and the globalisation has brought more competitors to the market. Rapid technological development is, at the same time, decreasing product life cycles; the manufacturing industry needs new methods and technologies to overcome these challenges [1]. A result of this is Industry 4.0, which is defined as a collection of modern technologies to enhance manufacturing and reduce the time to market [2]. One of these technologies is *autonomous robots*, which can work in dynamic environments where classical robot manipulators struggle. Recent approached to cope with classical robot programming has been sought through skill-based programming principles [3][4] and task-space human-robot interaction [5]. A slaughterhouse is an example of a dynamic environment due to the variation present in meat. This can prove a challenging environment for process solutions using classical robot and automation methods [6].

A. Adaptable models

An adaptable model is a technique, which can handle dynamically changing environments. Examples of *classical* adaptable models are *potential fields* which has been used to navigating a mobile robot successfully in an unknown environment [7].

In recent years an interest in the subject of *machine learning* and especially deep machine learning utilising neural networks to predict an outcome, has become a popular buzzword and technique to use. In the realm of machine learning, three paradigms exist: supervised learning, unsupervised learning, and reinforcement learning. Unsupervised learning is typically used to find un-specified patterns in datasets. In supervised learning, a model is first trained using a dataset with known classes. The model can then be used to predict the probability that an input belongs to a given class. In reinforcement learning, prior examples of optimal actions are not given. These must thus be discovered through trial-and-error principle by getting rewards based on performance.

This paper focuses on handling dynamically changing environments with modern technologies such as machine learning and Virtual Reality (VR) system. The machine learning paradigm chosen is reinforcement learning which, in this paper, will be used to transfer human expert knowledge to a robot.

II. RELATED WORK

The neural network machine learning boom started around the middle of the 2010s, whereas the mathematics behind backpropagation, feed-forward and activation functions are some decades-old [8][9][10]. Nonetheless, the computation power and the large datasets required to get good results were not available at that time. In 2015, DeepMind published a paper on Deep Q-Learning (DQN), which showed a method to stabilise the normally unstable neural networks. Since the publication, the usage and interest in neural networks in reinforcement learning have gained traction [11].

In the field of robotics and reinforcement learning, examples of work can be found in different sub-fields of robotics. In the path-planning sub-field, a traditional Q-learning algorithm was used to control a 6 DoF manipulator [12] and solve a wire loop game [13]. Welding tasks has also been tested with deep reinforcement learning where, e.g. an Actor-Critic network (ACN) has been used for solving welding tasks [14][15]. In the robotics sub-field of pick-and-place, examples of the deep reinforcement learning algorithm Deep Deterministic Policy Gradient (DDPG) exist [16][17][18].

In this paper, we will mainly use DDPG to solve a use case by trajectory learning from a human expert.

III. USE CASE

At the industrial partner (slaughterhouse), they have a task of picking a large middle piece of meat from a table and place it on a hook located on a rod (illustrated in Figure 1). The worker is only allowed to perform the task continuously for 40 minutes due to the high strain from lifting the heavy meat and repetitive behavior. Since this is a heavy and tedious task, the industrial partner is researching the ability to use robotic solutions in this dynamic job.

A. Direct task learning

To understand the concept of transferring human expert knowledge to a robotic system, it is useful to understand what kind of knowledge is to be transferred. In this use

*This work was supported by MADE Digital

[1]Emil Blixt Hansen, Rasmus Eckholdt Andersen, Steffen Madsen, and Simon Bøgh are with the Department of Materials and Production, Aalborg University, Denmark

[2]Corresponding author: Emil Blixt Hansen ebh@mp.aau.dk

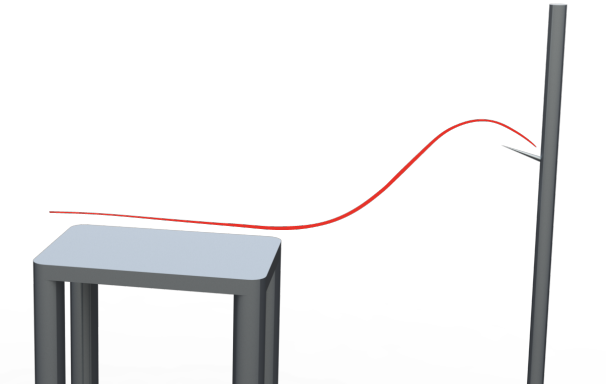

Fig. 1. The red line indicates the trajectory of the meat from the aluminium table to the hook.

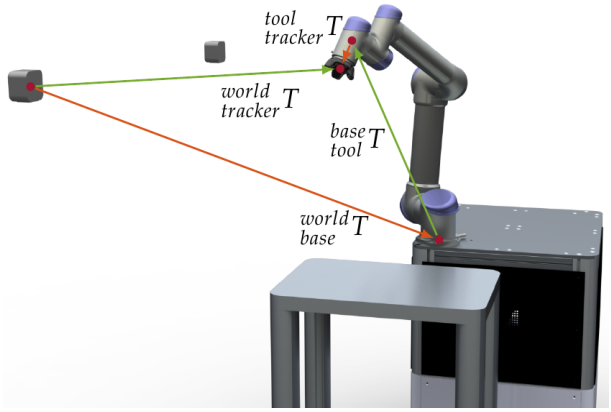

Fig. 2. The transformation matrices in the setup, where the two unknowns are the transformations ${}^{world}_{base}T$ and ${}^{tool}_{tracker}T$, illustrated with red.

case, the operator slides the meat to the edge of the table before lifting it above the hook. Gravity is then used to place the meat firmly on the hook. With this movement, the operator reduces the effort which is needed and thus limiting the risk of injuries. To better understand the correspondence of knowledge, we describe what we call *Direct Task Space Learning*. In classical teleoperation of a manipulator, the operator moves the robot, and thus the task is handled in the task space of the robot and not in the operator's task space. It is believed that information can be lost and therefore, we want to bring the robot into the expert's task space and learn directly from it.

B. Collection of human expert data

To accomplish this direct task space learning, the behaviour of the expert doing the task is needed. We can collect this behaviour with a VR system with trackers mounted on the meat and thus gathering the expert data required. We used an HTC VIVE system and performed a hand-eye calibration with the robot manipulator (Universal Robot UR5) and the HTC VIVE system, shown in Figure 2. The unknown transformation (illustrated with red lines) is then found with the hand-eye calibration method from [19]. After the calibration, it is possible to relate any point from the HTC VIVE system with the robot base.

With the calibration, ten expert trajectories were recorded, by a human expert dragging a lump of a fake piece of meat on the table and placing it on the hook.

IV. TRAJECTORY LEARNING

A deep reinforcement agent needs to explore the statespace and thus make mistakes in order to learn. It is not desirable for an agent controlling a robot to make a lot of mistakes since it is hazardous for itself and its surroundings; thus, a simulation environment is required.

A. Simulation environment

As a simulation platform Gazebo was chosen as it already has a functional ROS connection. The simulation environment enables us to speed up the training. However, it was discovered that Gazebo has some stability issues. To overcome this stability problem while remaining in the task space, we created a simple simulation environment called TCP Simulation. This simulation does only focus on the TCP coordinate in the Cartesian space, and thus no dynamics and kinematics of the robot nor the world are present. However, it is much faster than Gazebo and does not suffer from stability issues.

B. Distribution of training

Since the TCP simulation is far faster compared to Gazebo (0.003 seconds compared to 0.1 seconds, per step), but it is at the cost of lost kinematics and dynamics information, we suggest a distribution of training. This distribution is shown in Figure 3. The initial training is done in the TCP simulation till the exploration is less and the agent has started to converge. The training is hereafter moved to Gazebo and then to the real robot.

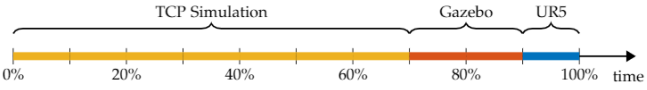

Fig. 3. The distribution of training.

C. Inverse reinforcement learning

One of the challenges of reinforcement learning is the deduction of a rewards function. In our case, the optimal policy for the agent is already given by the expert and the expert data. Therefore we can use inverse reinforcement learning to flip the problem and letting the agent find the optimal reward function given the expert data.

In this paper, we tested two inverse reinforcement learning algorithms - the first called linear inverse reinforcement learning [20] and the second known as apprenticeship learning [21]. The first approach is a linear programming problem, and the second is a quadratic programming problem. Besides the two mentioned methods, our concept of Principal Component

Analysis (PCA) weighting is also introduced and discussed. Since the state space and action space is continuous the reinforcement learning method, DDPG was chosen.

Along with the inverse reinforcement learning algorithms, the concept of viapoints is introduced as features for the agent. The features are therefore the minimum Euclidean distance to each of the points, and thus it becomes a Euclidean minimisation problem. The algorithms were tested with 2, 4 and 100 viapoints:

- **2 viapoints:** The edge of the aluminium table and the top hook.
- **4 viapoints:** The edge of the aluminium table, top of the hook, floating between table and hook, and a distraction point raised above the table.
- **100 viapoints:** 10 viapoints are sample uniformly from 10 expert trajectories.

The hyper-parameters of the DDPG agent was not changed throughout the different runs.

In Linear Inverse Reinforcement Learning (IRL), a reward function is assumed to be representable by some linear combination of features and weights. The first approach to Linear IRL is by finding a reward function for which the expected value of some observed trajectories generated by an unknown optimal policy π^* is higher than the expected value of some observed trajectories following a policy π as shown in Equation 1. [20]

$$E[V^{\pi^*}(s_0)] \geq E[V^{\pi}(s_0)] \tag{1}$$

Where s_0 is a fixed starting state. The value $V(s_0)$ is calculated as a linear combination of some static basis feature functions $\phi_i(s)$ chosen at design time. When the reward function is defined as $R = \phi_i$, then the value of a basis function is computed as shown in Equation 2.

$$V_i^{\pi}(s_0) = \sum_{t=0}^{T} \gamma^t \phi_i(s_t) \tag{2}$$

The value for a state is then a weighted sum of all the basic feature functions as shown in Equation 3.

$$V^{\pi}(s_0) = \sum_{i=0}^{k} w_i V_i^{\pi}(s_0) \tag{3}$$

Where the weights w_i are the parameters to fit such that Equation 1 is true. This gives the linear programming problem posed in Equation 4.

$$\begin{aligned} \max \quad & \sum_{i=1}^{k} \left(V^{\pi^*}(s_0) - V^{\pi}(s_0) \right) \\ \text{s.t.} \quad & |w_i| \leq 1, \quad i = \{1, \ldots, k\} \end{aligned} \tag{4}$$

The second approach to Linear IRL, called apprenticeship learning, comes from [21]. The approach is overall similar to the method presented by [20], as it also set up a linear combination of feature functions that are weighted. While the first algorithm tries to match some value of a trajectory as shown in Equation 3, the algorithm presented by [21] tries to match feature expectation *vectors* estimated as shown in Equation 5 given m trajectories.

$$\mu = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{\infty} \gamma^t \phi(s_t^{(i)}) \tag{5}$$

Equation 5 is then used to compute the feature expectation for both the expert μ_E and a policy μ_π. The idea is that by matching these feature expectations, the policy π will produce trajectories that perform as good as the expert. Another difference is that [21] puts a $\| w \| \leq 1$ constraint on the weights meaning the problem is a quadratic programming problem rather than a simpler linear one as shown in Equation 6.

$$\begin{aligned} \max \quad & w^T (\mu_E - \mu_\pi) \\ \text{s.t.} \quad & ||w||_2 \leq 1 \end{aligned} \tag{6}$$

V. RESULTS

A. Linear inverse reinforcement learning and apprenticeship learning

Since the solution to the use case is a trajectory, it can be hard to measure the performance of it relative to the trajectories generated by the expert, i.e. a trajectory does *not* explicitly need to follow the expert's trajectory to solve the case. Therefore, one of the best performance measure (besides that it reaches the goal) is a visual inspection. In Figure 4, the results related to the linear inverse reinforcement learning algorithm is shown and compared with traditional reinforcement learning. It can be seen that both methods succeeded in approaching the goal. It should be noted that the traditional reinforcement learning algorithm is the same as inverse just with the weights of the features fixed to -0.5 and thus treated equally.

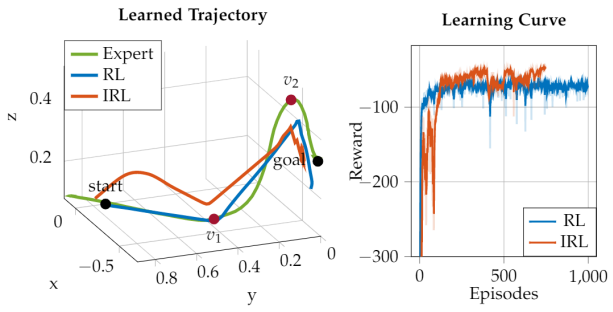

Fig. 4. A learned trajectory with two viapoints. Note that the reward function has been smoothened with a value of 0.6.

In Figure 5, the four viapoints is shown, and it can be seen that the traditional reinforcement learning does not learn to ignore the distraction point, whereas the inverse reinforcement learning does. This is expected behaviour because all of the features are weighted equally in the traditional

reinforcement learning; thus, the distraction point is just as important as the hook viapoint.

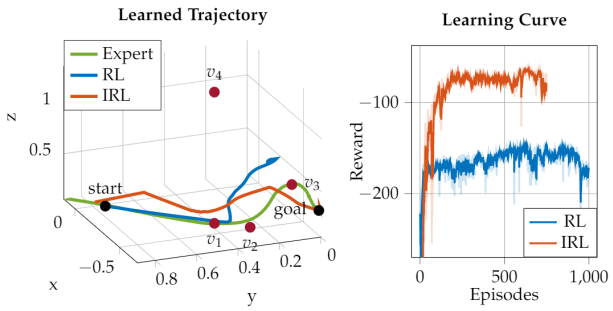

Fig. 5. A learned trajectory with four viapoints. Note that the reward function has been smoothened with a value of 0.6.

For the 100 viapoint, shown in Figure 6, the same problem can be seen. Here when the upwards motion begins, the traditional reinforcement learning starts to deviate from the expert trajectory. An explanation for this behaviour is that all points only contribute to the minimum distance, and thus the goal is overshadowed by all the points. This is not the case in inverse reinforcement learning where the agent learns to ignore more of the viapoints and therefore a higher weight is assigned for the goal point.

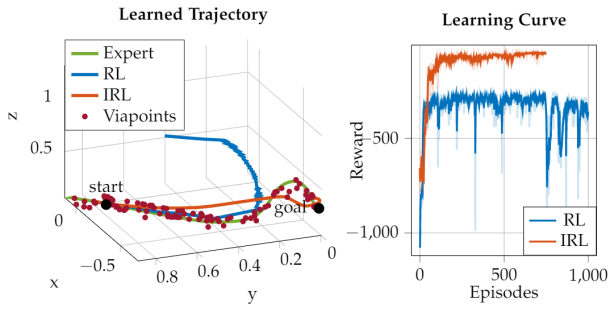

Fig. 6. A learned trajectory with 100 viapoints. Note that the reward function has been smoothened with a value of 0.6.

The usage of apprenticeship learning, i.e. the quadratic programming problem, does not seem to derive a better result as can be seen in Figure 7. Moreover, a problem related to apprenticeship learning was tendencies to produce positive weights, resulting in an increasing reward when moving farther away from the goal.

B. PCA weighting

One of the believed problems with the two tested inverse reinforcement learning methods was the selection of the viapoints and the weighting of them. A second approach is in this section proposed for weighting features, called PCA weighting. It is deemed possible that the eigenvector of the biggest eigenvalue related to the feature matrix covariance contains information about each feature contribution. Note that the feature matrix is features extracted for all expert data combined. This eigenvector thus contains the weights

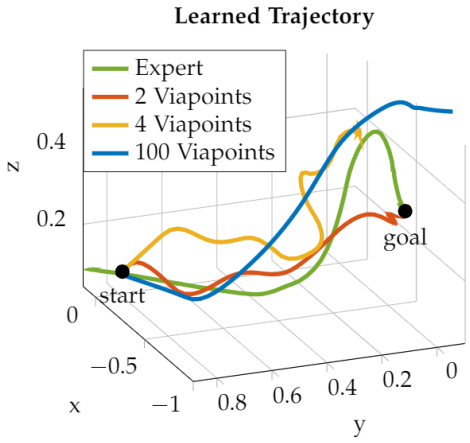

Fig. 7. An expert trajectory along with learned trajectories for 2, 4, and 100 viapoints using quadratic programming inverse reinforcement learning, from apprenticeship learning.

representing the importance of each feature. Features with the most significant variance are assumed to be of most importance. In Figure 8 the result can be inspected. It can be seen that by using PCA weighting, the agent solved the task, there are, however, still some derivation between the learned trajectories and the expert.

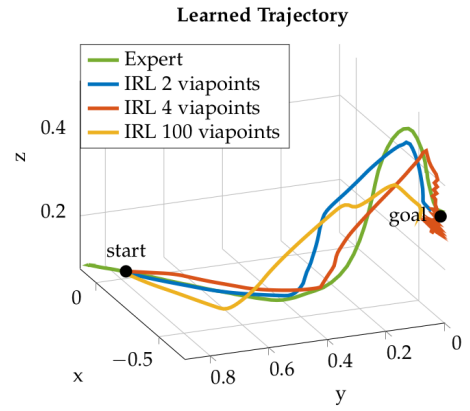

Fig. 8. An expert trajectory along with learned trajectories for 2, 4, and 100 viapoints using PCA weighting.

In the approach mentioned above, a single weighting was found for each feature. Another approach would be to have a time-step wise implementation of the PCA weighting, i.e. deriving a weight for each time-step. Let cov_w denote a weighted covariance and $F_{i,j}$ denote the the feature matrix at each time step where $i \in [t, T]$ and j is the number of features. Time depending weights can thus be found by finding the eigenvector related to the biggest eigenvalue of $cov_w(F_{i,j})$ for each time step t.

This was only tested with four viapoints, and the result is shown in Figure 9. It is shown that the line is smooth and solves the problem with the hook and approaches the goal. However, it does not reach the peaks of the expert trajectory, i.e. table edge and hook top. A reason for this could be that

a time depending weighting increases the complexity of the reward function.

For all the generated tests the methods had difficulty capturing the exact human behaviour. The believed reason for this is a combination of a large search space and the weights assigned to the features via inverse reinforcement learning.

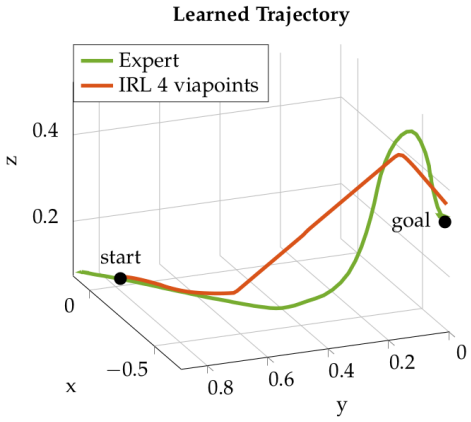

Fig. 9. An expert trajectory along with learned trajectories for 4 viapoints using time-step wise PCA weighting.

VI. CONCLUSION

One of the problems with reinforcement learning is that the reward function has to be engineered; this is what inverse reinforcement learning tries to solve. Nonetheless, the problem is then just shifted to designing and selection of features, which was also shown to be a just as hard and tedious task. The designing with viapoints showed that it did not create the best result, at least with linear inverse reinforcement learning and quadratic programming.

Our second approaches with PCA weighting showed a potential use with some visual better performance than the linear inverse reinforcement learning and apprenticeship learning. However, it was still not on the level of the expert, visually.

With this paper, we showed that it is possible to gather expert data and use it to solve a use case with inverse reinforcement learning methods. Nonetheless, it still has its caveats with not being of the same standard of the expert. For future work, efforts should be put into actually using meat and picking and placing it. Moreover, since the PCA weighting did show some potential, it should be further researched to investigate its potential.

REFERENCES

[1] Madsen, O., Schøiler, H., Møller, C,. Pedersen, T. B., Wæhrens, B. V., Remmen, A., & Vestergaard, A. (2014). Smart Production Et forskningprogram under AAU Production. Danish.

[2] Rüßmann, M., Lorenz, M., Gerbert, P., Waldner, M., Justus, J., Engel, P., & Harnisch, M. (2015). Industry 4.0: The future of productivity and growth in manufacturing industries. Boston Consulting Group, 9(1), 54-89.

[3] Andersen, R. E., Hansen, E. B., Cerny, D., Madsen, S., Pulendralingam, B., Bøgh, S., & Chrysostomou, D. (2017). Integration of a skill-based collaborative mobile robot in a smart cyber-physical environment. Procedia Manufacturing.

[4] Schou, C., Andersen, R. S., Chrysostomou, D., Bøgh, S., & Madsen, O. (2018). Skill-based instruction of collaborative robots in industrial settings. Robotics and Computer-Integrated Manufacturing, 53, 72-80.

[5] Andersen, R. S., Bøgh, S., Moeslund, T. B., & Madsen, O. (2016, August). Task space HRI for cooperative mobile robots in fit-out operations inside ship superstructures. In 2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN) (pp. 880-887).

[6] Purnell, G., & of Further, G. I. (2013). Robotics and automation in meat processing. In Robotics and Automation in the Food Industry (pp. 304-328). Woodhead Publishing.

[7] Cosío, F. A., & Castañeda, M. P. (2004). Autonomous robot navigation using adaptive potential fields. Mathematical and computer modelling, 40(9-10), 1141-1156.

[8] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. Cognitive modeling, 5(3), 1.

[9] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. Neural networks, 4(2), 251-257.

[10] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4), 303-314.

[11] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529.

[12] Park, J. J., Kim, J. H., & Song, J. B. (2007). Path planning for a robot manipulator based on probabilistic roadmap and reinforcement learning. International Journal of Control, Automation, and Systems, 5(6), 674-680.

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

[14] Günther, J., Pilarski, P. M., Helfrich, G., Shen, H., & Diepold, K. (2016). Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning. Mechatronics, 34, 1-11.

[15] Jin, Z., Li, H., & Gao, H. (2019). An intelligent weld control strategy based on reinforcement learning approach. The International Journal of Advanced Manufacturing Technology, 100(9-12), 2163-2175.

[16] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., ... & Zaremba, W. (2017). Hindsight experience replay. In Advances in Neural Information Processing Systems (pp. 5048-5058).

[17] Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018, May). Overcoming exploration in reinforcement learning with demonstrations. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 6292-6299). IEEE.

[18] Arana-Arexolaleibo, N., Anguiozar, N. U., Chrysostomou, D. & Bøgh, S. (2019), Transferring Human Manipulation Knowledge to Industrial Robots Using Reinforcement Learning. In 29th International Conference on Flexible Automation and Intelligent Manufacturing. Procedia Manufacturing.

[19] Tsai, R. Y., & Lenz, R. K. (1989). A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. IEEE Transactions on robotics and automation, 5(3), 345-358.

[20] Ng, A. Y., & Russell, S. J. (2000, June). Algorithms for inverse reinforcement learning. In Icml (Vol. 1, p. 2).

[21] Abbeel, P., & Ng, A. Y. (2004, July). Apprenticeship learning via inverse reinforcement learning. In Proceedings of the twenty-first international conference on Machine learning (p. 1). ACM.