euspen's 20th International Conference & Exhibition, Geneva, CH, June 2020

www.euspen.eu

Fixture design using reinforcement learning

Dennis Wee Keong Neo [1], Darren Wei Wen Low [2], A. Senthil Kumar [2], Kui Liu [1]

[1] *Joining and Machining Group, Singapore Institute of Manufacturing Technology, 73 Nanyang Drive, Singapore 637662*
[2] *Department of Mechanical Engineering, National University of Singapore, 9 Engineering Drive 1, Singapore 117576*

dennis_neo@SIMTech.a-star.edu.sg

Abstract

This paper aims to explore the use of machine learning techniques, specifically reinforcement learning, as a tool to realise an optimised fixture design. In response to a fixturing environment, the adjustments to the reinforcement learning process in the exploration phase has been studied. Different learning exploration approaches have also been demonstrated with a case study to validate its performance in a reward rich environment. These successful results would further credit the capability of the reinforcement learning technique as an effective automated fixture design tool in generating well-designed fixtures at a short period of time without any human intervention.

Computer-aided design; fixture design; artificial intelligence, reinforcement learning, learning exploration approach

1. Introduction

Fixtures are an integral aspect of manufacturing, providing essential workpiece locating and clamping elements prior to subsequent processes. Having well-designed fixtures are crucial in achieving consistent and accurate manufacturing outcomes. However, designing such fixtures is a challenging task due to many engineering considerations and optimisation strategies. Conventionally, fixture design relies on the vast heuristic experience of human designers [1], demanding years of apprenticeship to learn. Reliance on these skilled professionals is essential in creating optimal fixtures, which results in costly fixtures. Moreover, the experience-heavy nature of fixture design causes a significant knowledge gap for junior fixture design engineers, which can limit the effectiveness of their contributions. Much research has therefore been done on automating fixture design, which would potentially reduce design costs, human error and lead time.

Case-based reasoning (CBR) has been the predominant approach in automating fixture design [2,3,4,5] which leverages on matching a given workpiece to similar proven fixturing designs, subsequently providing the necessary design adjustments. However, CBR needs a proper indexing of a design library for design retrieval and evaluation, and final adjustments to work [6]. Another limitation is that its reliance of extensive and comprehensive fixture design libraries to produce fixturing solutions limits the ability of CBR when processing unique and unorthodox workpieces. If a given workpiece is significantly different from those found in the library, CBR may not be able to produce a valid result [7]. In other words, although CBR has shown to be effective in solving experienced-based problems through inference from a library, it is unable to adapt to significantly different situations. This inflexible nature of CBR therefore limits its potential in handling edge cases and CBR ultimately still depends on fixture designers to provide high-quality examples.

On the other hand, rule-based reasoning (RBR) has also been studied in generating fixture designs [1,8,9]. RBR utilises a selection of defined rules to convert geometric information into suggested positions for fixture locating elements. RBR's main disadvantage is the difficulty in accurately and comprehensively defining the rules needed to encompass all possible fixture designs. Additionally, using too little rules would result in poor coverage of possible fixturing problems, whereas too many rules would make the code significantly complex [10]. A combination of CBR and RBR [11] has also been attempted to combine the benefits of both methods in generating fixture design solutions. However, it still suffer from inheriting their individual limitations as discussed previously.

Notwithstanding the facts that automatic fixture design using CBR and RBR has been well researched, critical assumptions in their fundamental mechanisms limits their practicality for fixture design application. Machine learning has also been explored in automating fixture design [12]. However, much of reported work remains conceptual.

This paper presents the use of reinforcement learning to generate optimal fixturing solutions. Through a proposed reinforcement learning driven fixture design (RL-FD) framework, reinforcement learning was used to generate optimised fixturing solutions. In response to the fixturing environment, adjustments to the reinforcement learning process in the exploration phase is studied. A case study is presented, comparing a conventional exploration method with an adjusted one. Two different agents show improved average results over time, with the adjusted exploration model exhibiting faster performance.

2. Framework of reinforcement learning based fixture design

A reinforcement learning driven fixture design (RL-FD) has been developed as a novel approach to automate the fixture design decisions. There are five key necessities needed for crucial reinforcement learning function, as follows:

a) Generating workpiece silhouette,
b) Automatic generation of initial locators,
c) Physics simulation and state representation,
d) Action space and reward feedback, and

e) Convolutional neural network (CNN)

In this proposed RL-FD architecture as shown in Figure 1, a 2D silhouette of a given workpiece is firstly generated since the outer surface of workpiece tends to be more important for locating and clamping in any 3-dimensional fixture system. 2D silhouette also simplifies the physics and improves compatibility with already available CNNs.

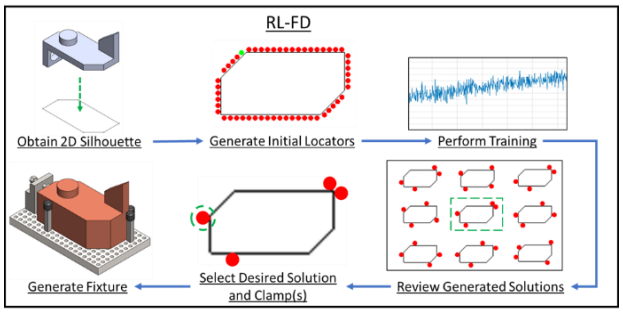

Figure 1. Proposed RL-FD framework for generating optimal fixture

In the second step, the initial locator positions are generated to properly constrain a particular workpiece silhouette. Workpiece silhouette geometries (see Figure 2) are defined using the X-and Y-coordinates of the polygon edges. The list of given coordinates can be used to generate the initial locators around the silhouette using the following equations:

$$\theta = \arctan 2\left(\frac{y_b - y_a}{x_b - x_a}\right) \quad (1)$$

$$n = \left[0, \frac{\sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}}{2r}\right] \quad (2)$$

$$x_n = x_A + r(2n+1)(cos\theta),$$
$$y_n = y_B + r(2n+1)(sin\theta) \quad (3)$$

$$x'_n = x_n + r(\cos(90-\theta)),$$
$$y'_n = y_n - r(\sin(90-\theta)) \quad (4)$$

Where (x_a, y_a) and (x_b, y_b) represents the initial and subsequent silhouette geometrical coordinates, respectively. (x_n, y_n) are coordinates between A and B, where n is a range of integers defined by Eq. (2). (x'_n, y'_n), which is centre of each circular locator can then be determined using Eq. (4). The conversion occurs until all the edges of the given polygon has been acted upon. These final converted coordinates are used to automatically populate the necessary initial locator positions based off the given geometry.

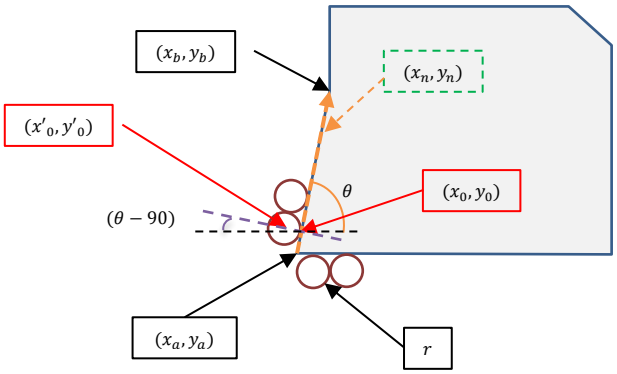

Figure 2. Locator geometries and locations coordinates

Next, a pybox2d, a physics rigid body simulation library [13,14] would be utilized for physics interactions in a silhouette-locator environment. With this environment, the fixture's ability to constrain the workpiece from moving and rotating will be evaluated, as presented in Figure 3. In the proposed framework, it only considers constrains in 2D directions and hence Trans_Z is not considered. In response to Rot_X and Rot_Y, typically the fixture has locators on the bottom which, when included with the locators/clamps as generated by the proposed framework, should theoretically restrict Rot_X and Rot_Y.

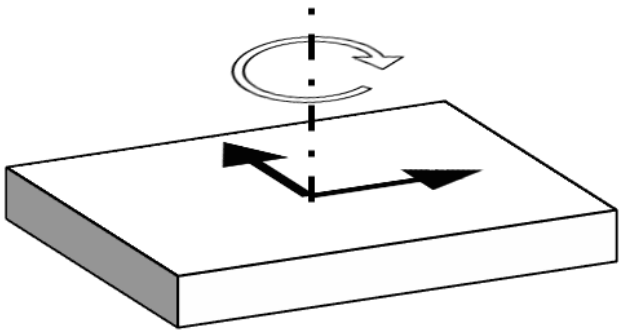

Figure 3. Evaluation of fixture's ability to constrain the workpiece in the two linear and one rotational directions.

A reinforcement learning agent, which is made using CNN, will be trained against this environment. CNNs are convenient in this study as a 2D pixel array can be used as an input. The CNN used in this study is shown in Figure 4. This neural network [15,16], which was originally designed for use on Atari game environments. Changes include widening the neural network input layer to incorporate the larger pixel array and adjustments to grey scaling.

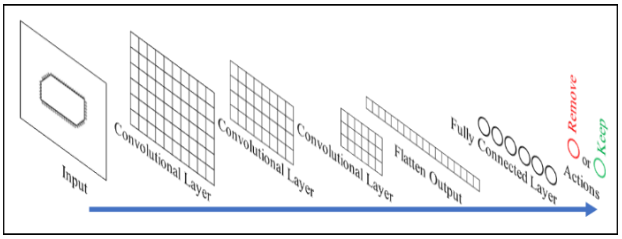

Figure 4. Convolutional neural network, CNN for decision-making process to perform either *Keep* or *Remove* action. (Not drawn to scale)

There are only two actions available to the modified agent, which are either a remove or keep locator action. These two actions represent the absolute minimum necessary for proper interactions with the physics simulation, which helps to speed up the learning process. To encourage the agent to reach a more optimal fixture with the least number of locators, positive reward will be given for each successful remove action performed. Successful removals are evaluated by applying a series of forces and torques on the workpiece silhouette, as explained in Figure 3. The evaluation simply identifies significant linear or angular movement beyond acceptable limits, which is indicates that the workpiece is no longer secure and therefore the fixture is invalid.

With the rewards given for acceptable locator removal, the agent should progressively get better at fixturing decision making. Lastly, the agent would be able to create highly optimised fixturing solutions, which will be stored and listed out to the user for selection. The selected solution can then be used to automatically generate a computer-aided design (CAD) fixture.

3. Case study

This study utilises a workpiece as illustrated in Figure 5 as a case study. With the developed RL-FD framework, a feasible

fixture would be generated and employed for machining the top surface of this component.

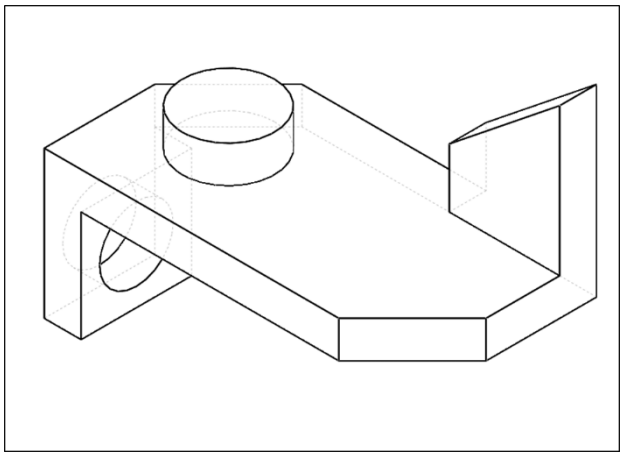

Figure 5. Selected workpiece as a case study

A key consideration of reinforcement learning is the incorporation of an exploration method, which provides adequate and representative training on all possible actions available to the agent. At the start of any learning process, the agent would pick actions from this exploration function instead of the neural network. Overtime, dependence on this exploration mechanism would decrease and instead the neural network would be used to make decisions.

A widely adopted exploration method is selecting actions randomly, with each possible action having an equal chance of being selected. For the environment in this paper, the agent only needs to decide between keep and remove. The exploration mechanism of these two actions would therefore be as illustrated in Figure 6a, where both actions would be chosen 50% of the time.

However, in the proposed fixture design environment an ideal agent would perform the remove action frequently in order to maximise its cumulative reward. Therefore, this study would also compare a chance-adjusted model whereby the remove and keep actions are performed 75% and 25% of the time respectively as shown in Figure 6b. This adjustment should theoretically allow the agent to learn quicker by bringing it closer towards an ideal state, while also maintaining sufficient representation of the keep action.

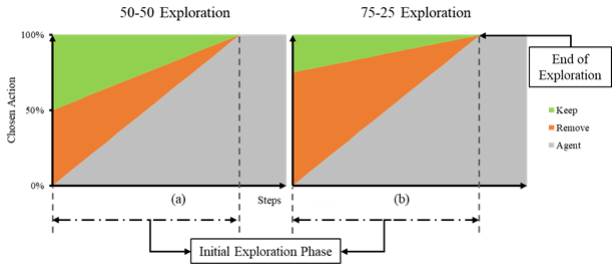

Figure 6. Exploration mechanism (a) shows equal chance of either keep or remove actions in initial exploration phase (b) shows 75% chance of remove and 25% keep in initial exploration phase.

Both of different agents would be trained on this environment until 5,000,000 steps are performed. A step represents as a single action performed by the agent. 5,000,000 steps was chosen as a basis of comparison between both agents as preliminary runs show stable results prior to this step count. Both agents are:

a) 50-50 Agent: Performs 50% Remove and 50% Keep when exploring
b) 75-25 Agent: Performs 75% Remove and 25% Keep when exploring

$$Average\ Final\ Reward = \frac{\sum Final\ Rewards}{Number\ of\ runs} \quad (5)$$

The average final rewards will be used as a means of evaluating learning performance and is calculated at every step using Eq. (5). Final rewards are recorded when the minimum number of active constraints is achieved, or when the solution is not valid. In both cases, the physics environment is reset after.

As shown in Figure 7, both agents show performance improvements over time This demonstrates their capability in learning from the fixture design environment and progressive improvements in making better fixture design decisions. However, the 50-50 Agent initially started with a noticeably lower average reward and took significantly longer to reach higher scores as compared to the 75-25 Agent. This observation can be attributed by a more optimal exploration phase, resulting in favourable learning performance as exhibited by the 75-25 Agent.

Using the same computer in both instances, faster learning was observed from the 75-25 Agent as compared to the 50-50 Agent. The 75-25 Agent also obtained the optimal fixture (highest reward) in 20 hours, as compared to 35 hours from the 50-50 Agent. The time reduction of 43% exhibited by the 75-25 Agent is preferable, resulting in an overall faster fixture optimization process which consequentially reduces fixture lead times.

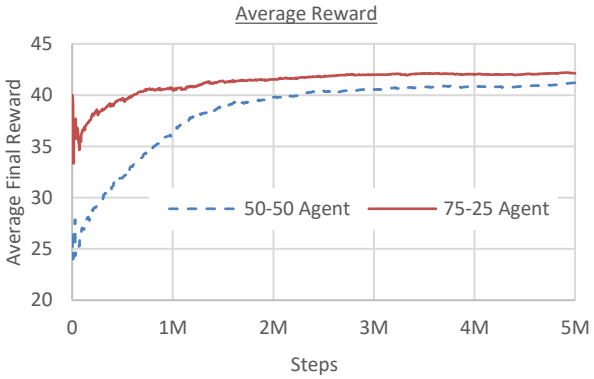

Figure 7. Average final rewards obtained during training of the agent. A single step refers to either a Remove or Keep action.

Generated solutions that receive high cumulative rewards are saved and listed out as presented in Figure 9. The presented solutions have been successfully passed the fixturing test as described in Section 2. Therefore, they represent the valid and optimal fixturing solutions for the given workpiece.

However, in a real-world scenario, there are other considerations that designers might have to account for (i.e. restrictions with the space available, worker ergonomics and cost). The proposed framework simply generates a list of optimised and practical fixtures.

After considering several factors such as cost, ergonomics and space constraints, the designer would have to make a decision as to which of the optimised fixtures are used. Upon selecting any of the fixturing solutions in Figure 9, following which, the user will able be required to select which of the given locators should be changed into clamping elements. A completed CAD model from one of the solutions can then be generated as shown in Figure 9.

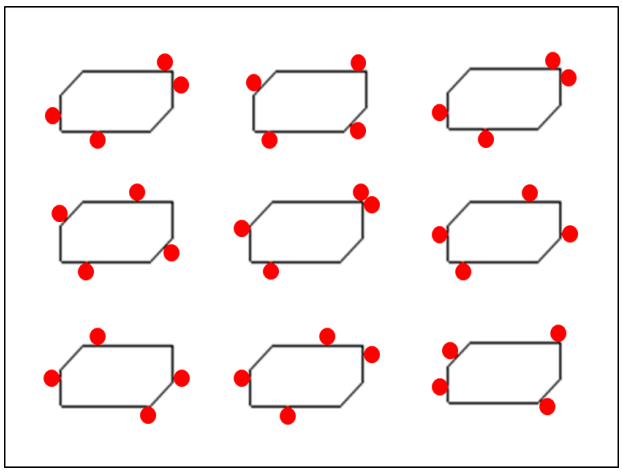

Figure 9. Selected training results generated from training agent

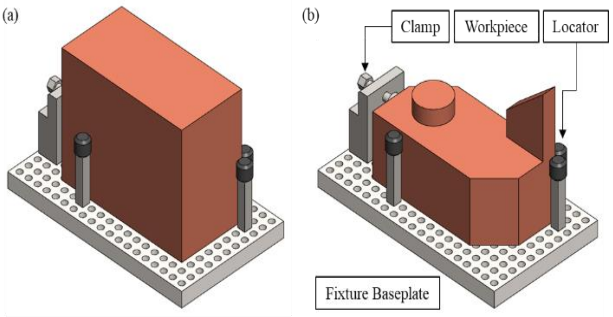

Figure 8. Generated fixture (a) Raw workpiece prior to machining. (b) Machined top surface of workpiece after intermediate machining processes

4. Conclusions

Fixture design relies heavily on the experience and awareness of the multitude of engineering considerations. It is therefore very vital and essential to automate this process, which can reduce the reliance on experienced designers and thereby improving the overall efficiency. Historical attempts in automated fixture design has largely adopted the use of CBR and RBR, which exhibits certain fundamental limitations in its practical use. This paper demonstrates the use of an automated RL-FD framework using reinforcement learning.

The framework was described and demonstrated through a case study. Reinforcement learning was found to be capable of training a neural network in making better fixture design decisions over time. The optimisation of the exploration mechanism has shown to be effective in improving the training time. Further research is onoing to encompass more complex workpieces using non-silhouette-based approaches.

References

[1] Nee A Y C and Kumar A S 1991. A Framework for an Object/Rule-Based Automated Fixture Design System. *CIRP Annals* **40** 147-151

[2] Kumar A S and Nee A Y C 1995. A framework for a variant fixture design system using case-based reasoning technique. *Manu. Sci. Eng.*, ASME, **3** 763-775

[3] S. H. Sun and J. L. Chen 1996. A fixture design system using case-based reasoning. *Eng. App. Art. Intel.* **9** 533-540.

[4] W. Li, P. Li, and Y. Rong 2002. Case-based agile fixture design. J. Mat. Proc. Tech. **128** 7-18

[5] H. Hashemi, A. M. Shaharoun, and I. Sudin. A case-based reasoning approach for design of machining fixture. *Int. J. Adv. Manu. Tech.* **74**, no. 1, pp. 113-124, 2014.

[6] D. McSherry 2002. The Inseparability Problem in Interactive Case-Based Reasoning. Research and Development in Intelligent Systems XVIII, London, , pp. 109-122: Springer London.

[7] P. S. Szczepaniak and A. Duraj 2018. Case-Based Reasoning: The Search for Similar Solutions and Identification of Outliers. Complexity, **12**, 9280787.

[8] Dong X, DeVries W R and Wozny M J 1991. Feature-Based Reasoning in Fixture Design. *CIRP Annals* **40** 111-114

[9] A. S. Kumar, A. Y. C. Nee, and S. Prombanpong 1992. Expert fixture-design system for an automated manufacturing environment. *CAD*, **24**, 316-326.

[10] J. Prentzas and I. Hatzilygeroudis 2007. Categorizing approaches combining rule-based and case-based reasoning. *Expert Systems* **24** 97-122

[11] Zhang F P, Wu D, Zhang T H, Yan Y, and Butt S I 2018. Knowledge component-based intelligent method for fixture design. *Int. J. Adv. Manu. Tech.* **94** 4139-4157

[12] Kumar A S, Subramaniam V and Tan B T 2000. Conceptual design of fixtures using machine learning techniques. *Int. J. Adv. Manu. Tech.* **16** 176-181,

[13] E. Catto. (2011). Box2D. https://github.com/erincatto/Box2D

[14] K. Lauer. (2011). pybox2d. https://github.com/pybox2d/pybox2d

[15] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M 2013. Playing Atari with Deep Reinforcement Learning. Technical Report arXiv:1312.5602[cs.LG], *Deepmind Technologies*.

[16] Mnih V. et al. 2015 Human-level control through deep reinforcement learning. *Nature* **518** 529–533