Lasers in Manufacturing Conference 2023

Monitoring and control of laser-based additive manufacturing using convolutional neural nets and reinforcement learning

Frederic Cousin[a,*], Vishnuu Jothi Prakash[a], Julian Ulrich Weber[a], Ingomar Kelbassa[a]

[a]*Fraunhofer Research Institution for Additive Manufacturing Technologies IAPT, Am Schleusengraben 14, 21029 Hamburg, Germany*

Abstract

Directed Energy Deposition (DED) processes using industry robots enable the production of large structures through additive manufacturing technology. One major challenge hindering the industrialization of DED processes is online process monitoring and control. Melt pool characteristics are one of the key identifiers regarding the stability of DED processes. Typically, these characteristics are monitored visually by camera systems using standard image processing toolboxes. To improve the robustness, the image processing of melt pools has been replaced by a convolutional neural net.

Furthermore, a control strategy, focusing on laser power and processing speed, has been introduced through a Reinforcement Learning Framework. Wherein, a second neural net, predicting melt pool characteristics, has been used as the simulation environment. Based on this, the reward metric for the reinforcement system has been determined by comparing target melt pool characteristics with the predicted ones.

Keywords: Laser-based Processing; Reinforcement Learning; Convolutional Neural Net; Artificial Intelligence; Additive Manufacturing

1. Introduction

In order to produce large structures through additive manufacturing technology, the directed energy deposition process (DED) is a suitable option. Here, an industrial robot can be used to handle the deposition head. In general, one distinguishes between two different material delivery options in DED processes, namely powder feeding and wire feeding. Typically, a laser or electron beam is used as the heat source in the process. The heat impact caused by the heat source on the powder or the wire as well as the movement of the deposition result in certain melt pool characteristics. These in turn determine the geometry and the material characteristics of the produced part (Gibson et al., 2015). Hence, monitoring the melt pool characteristics and using the monitoring results as control parameters can improve the process quality. For this purpose, this paper shows how melt pool characteristics can be monitored by using a convolutional neural net (CNN). Furthermore, a reinforcement learning concept has been introduced, in order to use the monitored

information to control and therefore improve the process. The paper is structured as follows. Initially, the experimental setup is described, then, the monitoring concept based on a CNN is explained. How one can take advantage of the monitoring results using a reinforcement learning framework is described in section four. Finally, conclusion and outlook are presented, which outline the next steps to implement the reinforcement learning agent in the process.

2. Experimental setup

In this section, the experimental setup as well as the data acquisition concept are described to explain how the necessary data was collected during the process.

2.1. Physical setup

The physical setup consists of three major parts, which are combined in a TruLaser Robot 5020 Cell:

- LaserTruDisk 6001 as the energy source
- KUKA KR60HA with controlling unit KRC 2 as multi-axis kinematic system
- Laser Metal Deposition head with rotary powder feeding unit

For the sake of completeness, it has to be mentioned, that all trials used for this paper were processed with the following material and process parameters:

Table 1. Process parameter

Material	Powder Flow (g/min)	Laser Power (W)	Processing Speed (m/s)
Ti6Al4V	13.0	1450	0.01

2.2. Data acquisition

To monitor the melt pool characteristics, an industrial camera (Basler acA1920-40um) was integrated with a beam-splitter into the beamline. Through this, the melt pool can be nearly continuously (41 fps) observed. For the later training of the CNN, a dataset consisting of many melt pool images is necessary. Hence, an application is needed which saves a certain amount of melt pool images during the process. Therefore, a Python application (Van Rossum and Drake, 2009) was developed, which runs on an industrial computer and saves the input image of the industrial camera every 0.5 seconds on an external device. This setup is schematically visualized in Fig. 1.

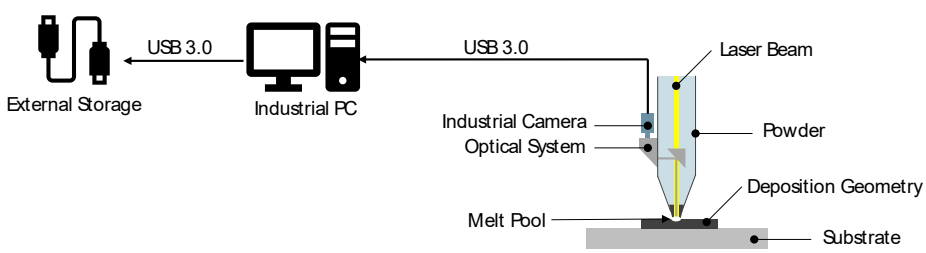

Fig. 1. Data acquisition setup

3. Melt pool monitoring

In order to monitor the melt pool, its characteristics need to be quantified. For this purpose, the geometrical characteristics are extracted from the melt pool images by a traditional computer vision application. Once this was done, a training data set, consisting of images as inputs and geometrical characteristics as labels, was created. This data set was used to train a CNN to replace the traditional computer vision application.

3.1. Labeling

During multiple trials, 1221 melt pool images were collected. These images were labeled with geometric characteristics of the melt pool by fitting an ellipse on the melt pool contour. This results in five values, namely the x-position, y-position, width, height and angle of the ellipse. The geometrical meaning of these values is graphically depicted in Fig. 2.

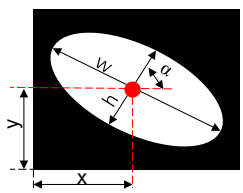

Fig. 2. Ellipse geometry

The application that detects the ellipse has been written in Python by using the image processing package Open-CV (Bradski, 2000). The general structure of the application is shown in the following pseudocode. Additionally, the different steps of image processing are visually represented with one example image.

```
images = Load_images()
For each image: (1)
        image = Threshold_filter()  (2)
        image = Area_of_interest()  (3)
        contours = Find_countours()  (4)
        For each contour:
                area = Contour_area()  (5)
                if area > lower_limit and area < upper_limit_
                        Fit_ellipse()  (6)
                        Save_ellipse_geometry()
```

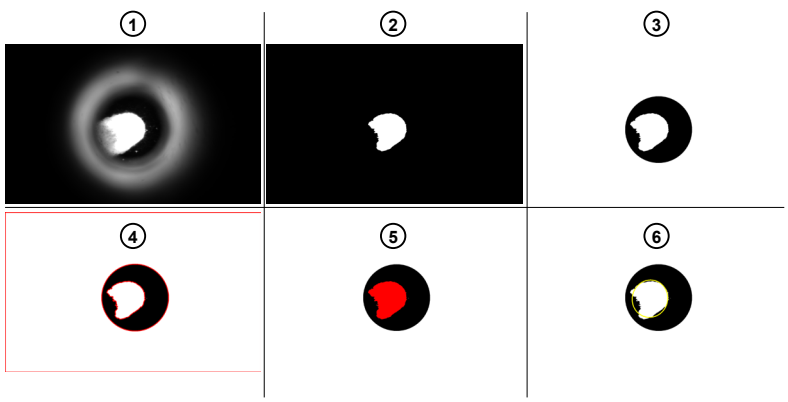

Fig. 3. Pseudocode traditional computer vision system

Due to the deterministic character of the traditional computer vision application, the robustness is limited. Especially the usage of thresholds in steps two and three is prone to error. Two potential error types can be named here:

- If the exposure of the image is too low, the threshold filter will return a black image, wherein no further detection can be made
- If the melt pool is bigger than the area of interest, the melt pool will be detected as too small

It can be observed that both error types frequently occur in the printing process due to varying laser power, processing speed as well as part geometry. To overcome this, either an advanced traditional computer vision application can be developed or a CNN can be used. The CNN can be more robust than the given traditional approach because it works more intuitively. Additionally, CNNs require less fine-tuning and are more flexible because, through re-training, they can be used also for other applications (O'Mahony et al., 2020). How the architecture of the convolutional neural net looks and how it was trained are mentioned in the following sections.

3.2. Convolutional neural net architecture

The architecture of the developed CNN consists of seven different layers, wherein the three typical layers for a CNN architecture, namely convolutional layers, pooling layers, and fully connected layers are included (LeCun et al., 2015). The eponymous layer type for CNN is the convolutional layer. The main reason why convolutional layers are used instead of fully connected layers for image processing tasks is the high number of weights that are necessary for fully connected layers. Instead of interpreting single pixels independently, convolutional layers use a sliding window that scans local regions in the input field. In this manner, features can be effectively detected independent of their location in the image. The size of the sliding window is defined by the kernel of the convolutional layer, whereas the third dimension defines the number of output channels (feature maps) (Albawi et al., 2017). Max-pooling layers also down-sample the input image by simply taking the maximum value of the applied kernel in the layer. Additionally, they improve position invariance (Scherer et al., 2010). The flattening layer unfolds the feature maps to build a one-dimensional vector (Chen et al., 2020). The last layer type is a fully connected layer.

Regarding the ellipse detection, the fully connected layer results in five values, namely the x-position, y-position, width, height and angle of the ellipse. In all convolutional layers, the state-of-the-art rectified linear unit function (ReLU) was used as an activation function (Nair and Hinton, 2010). Whereas a simple linear activation was applied in the fully connected layer. It has to be mentioned that before the images were processed by the CNN, they were pre-processed. Here a downscaling of 30%, as well as a Min-Max normalization of the grey values in the range of 0 to 255, took place. In order to get a better idea of how the structure of the CNN looks like, its architecture is graphically depicted in the following Fig. 4. Here the different layers are visualized with their main characteristics.

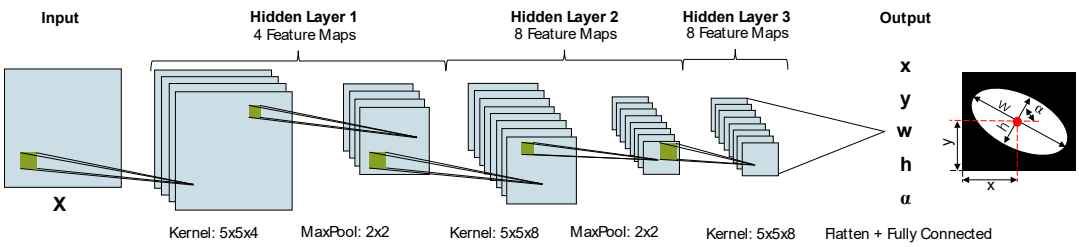

Fig. 4. CNN architecture

3.3. Training

For the training of the CNN, the initial dataset was split randomly into three different parts, namely training, validation and test. The shares are distributed as shown in the following table.

Table 2. Datasplit

Dataset	Total	Training	Validation	Test
Number of images	1221 (100%)	854 (≈70%)	242 (≈20%)	125 (≈10%)

The training part was used for the actual training of the CNN through the backpropagation algorithm. In order to check the generality of the learning process, the validation part was processed after each training epoch to check the performance on samples that were not used for the training. The last part, namely the test part, was used after the training of the CNN to check the performance on samples that were not involved in the training process (Bishop et al., 2006).

The CNN was trained over 10 epochs with a batch size of one. During training, the mean squared error was used as the loss criterion as well as the Adam-optimizer as the optimizer (Zhang, 2018). To track the training, the root mean squared error (RMSE) was used (Chai and Draxler, 2014). In this application, the RMSE describes the deviation in pixels of all predicted values from the labels. It can be computed through the formula given in (1) wherein n is the number of samples, x the pixel error of the x-position, y the pixel error of the y-position, w the pixel error of the width, h the pixel error of the height and α the pixel error of the angle. Each of the errors is simply computed by subtracting the actual value (determined by the traditional image processing) from the predicted value (2).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i + y_i + w_i + h_i + \alpha_i)^2} \quad (1)$$

$$e_i = |Predicted_i - Actual_i| \quad (2)$$

In Fig. 5 the training process is visualized through the RMSE on the training part and the validation part. As one can see in this Fig., the RMSE decreases already strongly after the first epoch. After that, just smaller improvements can be observed.

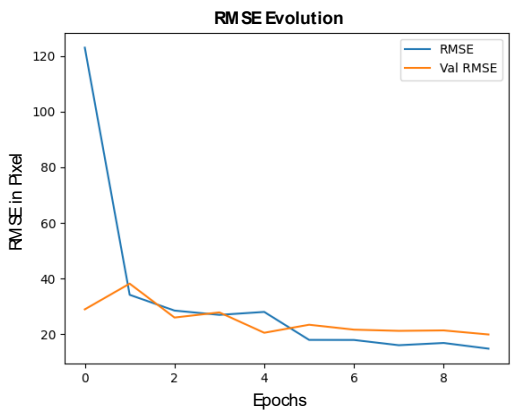

Fig. 5. Training process

As stated in Fig. 6, the CNN is replacing the traditional computer vision well. There, the performance of the different output features of all training samples is visualized. It can be observed that the CNN performs the worst in predicting the angle of the melt pool ellipse. But since all RMSEs are comparatively small, the CNN can be seen as a good approximation of the traditional computer vision system. To underline this, a predicted ellipse is graphically depicted in one sample image together with the ellipse detected by the traditional vision system used for labeling. In this test data sample, just a minimal difference between the ellipses is observable.

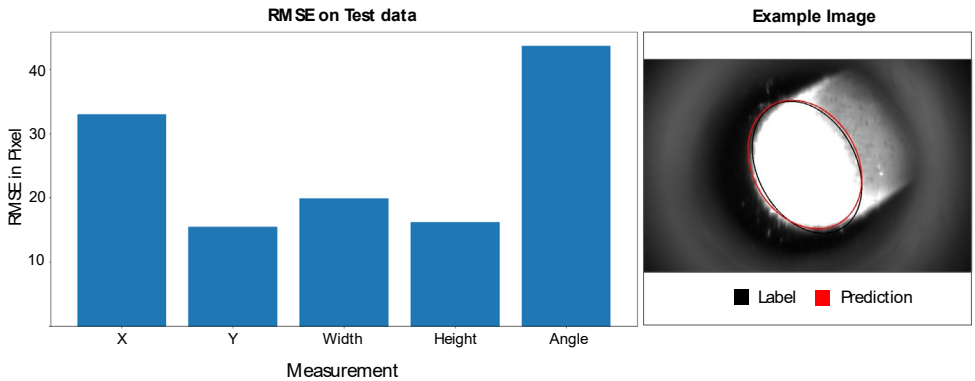

Fig. 6. Performance on test data

In order to justify the robustness improvement through the usage of a CNN instead of the traditional computer vision, a melt pool image with a lower exposure time was proceeded with both systems (error type one). Due to the deterministic character of the traditional computer vision system of the threshold filter in step two, the ellipse detection fails as soon as the exposure of the melt pool fall below the threshold, which results in a black image (see Fig. 7). Whereas, the CNN can still detect the ellipse, which shows that the CNN is more robust against varying exposure conditions.

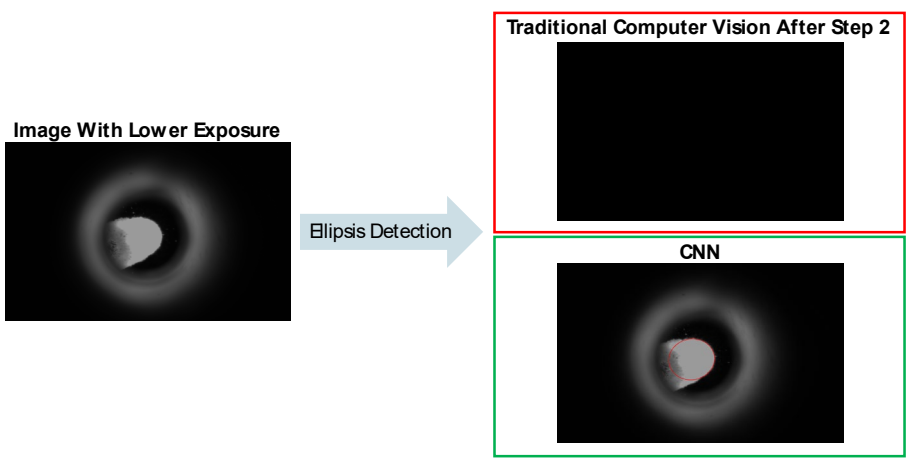

Fig. 7. Robustness improvement

4. Reinforcement learning concepts

Once the geometrical information of the melt pool can be automatically extracted, this information should be used to control the process. For this purpose, a Reinforcement Learning (RL) concept, inspired by Dharmawan, 2020 is introduced in this section. In general, RL systems consist of three main parts which interact with each other, namely an agent, a simulation environment (model-based RL) and a reward function (Kealbling et al., 1996). The simulation environment should approximate the working space of the agent as well as possible. Regarding the described DED process, the simulation environment should be represented by a regression model, which is able to predict the melt pool characteristics based on the process parameters (e.g. laser power, processing speed, layer, working distance, etc.). Here, a neural net or any other regression model can be used to approximate the process. In order to train the agent, a random state vector of process parameters (with layer equal to one) is initially provided to the agent. Based on this state vector the agent performs an action. The action choice depends on the policy of the agent (Sutton and Barto, 2018). The chosen action modifies the initial state vector by, for example, increasing the laser power. The modified state vector is then processed by the simulation environment to predict the resulting melt pool characteristics. These melt pool characteristics can then be compared with some target melt pool characteristics through a reward function. In this way, the agent gets feedback about the chosen action. This feedback is taken by a learning algorithm to improve the agent policy (Li, 2017). After that, the layer is increased by one to simulate the printing characteristic by getting a new state vector as input for the agent. This procedure is repeated until a max layer limit is reached. If the layer limit is reached, a new random state is initialized to cover the whole process space as well as possible. A graphical description of this learning process is depicted in Fig. 8.

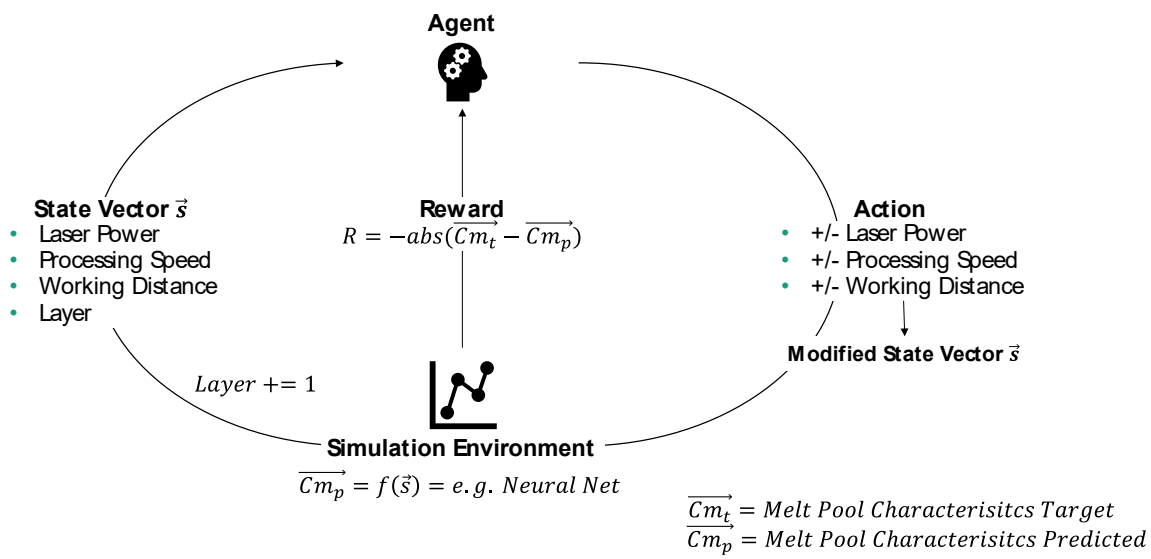

Fig. 8. Reinforcement learning concept

5. Conclusion and outlook

In Summary, this paper introduced a setup that can be used to monitor melt pool characteristics in a DED process. It was shown how the robustness of the ellipse detection can be improved by replacing a traditional computer vision system with a CNN which was trained based on the traditional system. Additionally, an RL concept was introduced, in which the extracted melt pool characteristics are acting as target variables in the reward function.

In order to implement and test the introduced RL concept some further steps are necessary. First of all, a cyber-physical system is needed to be able to generate a dataset to train the regression model for the simulation environment of the RL concept. Additionally, the RL agent has to be trained with the help of the regression model. As a last step, the RL Agent must be implemented in the cyber-physical system to be able to control the process and therefore ideally improve the print quality.

Acknowledgments

This research was funded by the German Federal Ministry of Education and Research within the funding program KI4KMU in the project KiRo3D (grant number: 01IS21048B).

References

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 international conference on engineering and technology (ICET) (pp. 1-6). Ieee.

Bishop, Christopher M., and Nasser M. Nasrabadi. Pattern recognition and machine learning. Vol. 4. No. 4. New York: springer, 2006.

Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. Geoscientific model development, 7(3), 1247-1250.

Chen, H., Chen, A., Xu, L., Xie, H., Qiao, H., Lin, Q., & Cai, K. (2020). A deep learning CNN architecture applied in smart near-infrared analysis of water pollution for agricultural irrigation resources. Agricultural Water Management, 240, 106303.

Dharmawan, A. G., Xiong, Y., Foong, S., & Soh, G. S. (2020, May). A model-based reinforcement learning and correction framework for process control of robotic wire arc additive manufacturing. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4030-4036). IEEE.

Gibson, Ian & Rosen, David & Stucker, Brent. (2015). Additive manufacturing technologies: 3D printing, rapid prototyping, and direct digital manufacturing, second edition. 10.1007/978-1-4939-2113-3

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. Journal of artificial intelligence research, 4, 237-285.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

Li, Y. (2017). Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).

O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., ... & Walsh, J. (2020). Deep learning vs. traditional computer vision. In Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1 (pp. 128-144). Springer International Publishing.

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In Artificial Neural Networks–ICANN 2010: 20th International Conference, Thessaloniki, Greece, September 15-18, 2010, Proceedings, Part III 20 (pp. 92-101). Springer Berlin Heidelberg.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace

Zhang, Z. (2018, June). Improved adam optimizer for deep neural networks. In 2018 IEEE/ACM 26th international symposium on quality of service (IWQoS) (pp. 1-2). Ieee.