Classification of Assembly Operations Using Recurrent Neural Networks

Björn Papenberg, Patrick Rückert and Kirsten Tracht

Abstract

Visual sensor data of manual assembly operations offers rich information that can be extracted in order to analyze and digitalize the assembly. The worker's interaction with tools and objects, as well as the spatial–temporal nature of assembly operations, makes the recognition and classification of assembly operations a complex task. Therefore, classical methods of computer vision do not provide a sufficient solution. This paper presents a recurrent neural network for the classification of manual assembly operations using visual sensor data and addresses the question as to what extent such a solution is feasible in terms of robustness and reliability. Since complex assembly operations are a combination of basic movements, four main assembly operations of the Methods Time-Measurement base operations are classified using a machine learning approach. A dataset of these four assembly operations, reach, grasp, move and release, containing RGB-, infrared-, and depth-data is used. A Convolutional Neural Network—Long Short Term Memory architecture is investigated regarding its applicability due to the spatial–temporal nature of the data.

Keywords

Machine learning · Manual assembly · Image processing

B. Papenberg (✉) · P. Rückert · K. Tracht
Bremen Institute for Mechanical Engineering (Bime), University of Bremen, Badgasteiner Str. 1, 28359 Bremen, Germany
e-mail: papenberg@bime.de

© The Author(s) 2022

T. Schüppstuhl et al. (eds.), *Annals of Scientific Society for Assembly, Handling and Industrial Robotics 2021*,
https://doi.org/10.1007/978-3-030-74032-0_25

1 Introduction

Globally acting, manufacturing companies must have efficient and versatile production structures in order to be able to meet customer demands for individually tailored products of high quality. These changing demands and shortened product lifecycles force modern production to be highly flexible [1]. This flexibility can be provided by human robot collaboration, by assigning exhausting and non-ergonomic tasks to the robot [2].

In order to enable such collaborations, it is of utmost importance to implement a visual sensor system that allows the robot to classify and analyze the assembly steps conducted by the human in real time. Industrial collaborative robots are able to detect collisions with humans or other objects and execute an emergency stop as soon as a collision is detected. While this safety mechanism forms the baseline for a safe collaboration it still requires the human to observe the robot at all times to avoid collisions, therefore burdening the human with an additional assignment. Thus, true collaboration along with a decrease in stress levels for humans can only be achieve by transferring that burden to the robot.

In this context, the presented paper addresses the question if and how assembly operations can be classified using methods of machine learning based on visual sensor data. Existing solutions generally do not classify assembly operations directly. Instead, these methods track activities in certain regions of the assembly area to draw conclusions regarding the assembly process [3, 4]. A key challenge in the classification of assembly operations using methods of machine learning is the interaction with tools and objects. Due to this interaction, conventional implementations fail, since this interaction is not represented in the used datasets [5]. In order to classify manual assembly operations efficiently, it is explored how they can be delimited and defined to enable a meaningful classification.

The assembly station and dataset proposed by Rückert et al. [6] is used for the implementation of the algorithm. Finally, the results are evaluated and interpreted in order to clarify the implementations strengths and weaknesses.

This paper contributes an outlook on the possibilities of classifying manual assembly operations using recurrent neural networks and an extension of the algorithm proposed by Rückert et al. [6].

2 Classification of Manual Assembly Operations

Existing implementations often rely on the assumption that the product is being assembled when the hands of the worker reside inside a predefined area in the work space [3, 4]. Other approaches draw bounding boxes around hands and objects and assume that the objects were interacted with, when the bounding boxes intersect [7].

While these implementations do not classify actual assembly operations, they offer insights on the challenges of the detection and classification of manual assembly operations. Therefore, it is evident that manual assembly operations need to be analyzed in detail.

The two most widely used methods to analyze manual assembly operations in such detail are REFA and Methods-Time Measurement (MTM) [8]. The time data collection according to REFA measures the actual time needed to perform certain assembly tasks and calculates the target times based on this information Verband and für Arbeitsstudien und Betriebsorganisation e.V.: Methodenlehre des Arbeitsstudiums: Teil 2Datenermittlung, Hanser, MünchenVerband and für Arbeitsstudien und Betriebsorganisation e.V.: Methodenlehre des Arbeitsstudiums: Teil 2Datenermittlung, Hanser, München [9]. Method-Time Measurement divides every manual assembly operation into several basic movements. For the finger-, hard-, and arm-system, these movements are reach, grasp, move, position and release [8]. These basic movements are defined beforehand an can be combined to represent 85% of all manual assembly operations [11].

Reaching is the movement of the fingers or hands to a specific place. Grasping involves bringing an object under control by closing the fingers. Moving an object from its initial position to its new destination describes the basic operation move. The operation position involves bringing an object into its final position at the end of its transport path. Releasing is defined as lifting control of an object by opening the fingers [10].

During the basic operations move and reach, the hand and arm are in motion. This is not the case with grasp and release, since these assembly operations primarily concern the motion of the fingers. Therefore, it is to be expected that the classification algorithm will mainly misclassify the motions move and reach as well as grasp and release, which results in a chess-like pattern of the confusion matrix. The key difference between reaching and bringing is the presence of an object in the hand. Unlike grasping, releasing is characterized by removing the fingers from an object.

Since MTM enables the detection of different motion sequences by identifying its typical characteristics and offers the prospect of deriving weaknesses in the work processes, it is more suited for the underlying approach than REFA Deuse et al. [8].

3 Machine Learning Algorithms for Activity Recognition

The recognition of hand gestures made significant breakthroughs in the last years due to methods of machine learning. While static gestures can be easily classified without machine learning, dynamic gestures prove to be more challenging, since they contain spatial–temporal information.

Recurrent Neural Networks (RNN), due to their ability to save states and thus handle spatial–temporal information, are suited for tasks where this property is needed [12]. This is enabled by the special neural connections of RNN architectures. Compared to normal feed-forward networks, where one neuron can only be connected to neurons in the next layer, a recurrent neuron can be connected to itself, neurons in the same layer and neurons in the next and previous layer, thus enabling it to store states, providing a vast number of trainable parameters [13]. Vanilla RNN-architectures are prone to the vanishing or exploding gradient problems, which are caused by the temporal links between the different

time steps [13]. Long Short-Term Memory (LSTM) is a RNN architecture that solves the vanishing and exploding gradient problem and is used in applications where spatial–temporal information has to be processed [12].

Molchanov et al. proposed a sequential architecture for the dynamic classification of hand gestures, where short video sequences were fed into a 3-Dimensional-Convolutional Neural Network (3D-CNN), extracting spatial temporal features. Afterwards, features were fed into an RNN where the spatial temporal information was processed. Multiple modalities were used to feed the network individually and combined, so that the results could be compared. With a precision of 80.3%, the depth-data provided the best individual result, while the combination of all modalities results in a precision of 83%. The robustness of the depth-data towards changing illumination might be the reason for the superior results, as suggested by the authors [14].

Lai et al. fused a CNN and RNN architecture on different architecture levels and reported an overall accuracy of 85.46% using skeleton-, and depth-data [15].

Since the recognition of hand gestures is a much more common problem than the recognizing and classifying assembly operations, considerably more datasets are available on this topic. While gesture recognition and assembly recognition are closely related, the classification of assembly operations is much more challenging, since the interaction with objects and tools might occlude parts of the hand and vice versa. Additionally, video sensors for assembly operations have to be mounted above the worker as to not hinder the assembly process. Thus recording the image sequences from a bird view perspective is necessary, thereby hindering the effective use of skeleton recognition.

4 Activity Recognition in Manual Assembly

While papers regarding the field of the recognition of manual assembly operations often focus on detecting objects and hands in certain areas inside the workspace, they generally do not detect individual assembly operations. Nonetheless, these papers offer insights regarding the challenges of the recognition and classification of assembly operations.

One challenging aspect of the recognition of manual assembly operations is processing the classification in real time. Root et al. detect the hands using the You-Only-Look-Once-v3 algorithm and surround them by a bounding box and subsequently calculates their positions [5]. They reported an accuracy of up to 89% but could not reproduce this result in a real assembly scenario due to the interaction with tools and objects [5].

Liu et al. investigated the feasibility in classifying manual assembly operations in a real production environment. In this context, they elaborated three key points that need to be addressed. They concluded that individual characteristics of each worker hinder a precise classification of assembly operations and that the visual sensors must not interfere with the assembly process. Furthermore, they mention the financial costs of the sensors. The suggested algorithm distinguishes seven different movements found in assembly

scenarios. The most important ones are classified as general assembly, reaching for assembly pieces and the recognition that no assembly takes place at specific moments in time. For the classification, a sequential architecture is proposed.

The hands are detected and their trajectory is fed into a 3D-CNN, which classifies the assembly operation with an accuracy of 89.1% Liu et al. [4].

Petruck et al. pursue a similar approach, using nine classes including general assembly and reaching for tools and objects. The movements during assembly are tracked using markers for a visual camera system, generating a numerical 27-dimensional vector, which contains the position and velocity and is fed into a CNN-architecture [3].

Andrainakos et al. propose a system that detects different objects and the hands of the assembler, drawing bounding boxes around them. Once the bounding box of an object and a hand intersect by a certain amount, the object is considered to be grabbed.

The water pump, which is used as an example, must be assembled from top to bottom. Whenever one detected object is correctly stacked on top of the other, the assembly is classified as correct [7].

5 Network Structure and Implementation

5.1 Dataset

For the experimental part of this paper, the dataset presented by Rückert et al. [6] was used. It consists of 2100 assembly operation sequences. The MTM basic operations reach, grasp, move and release are equally represented. The depth-data which was used for the training of the neural networks is stored in a resolution of 480 $\times$ 270 pixel [6]. While this dataset is small in comparison to other machine learning datasets, it is able to form the basis for a proof of concept.

5.2 Neural Network Architecture

During the training runs conducted in this paper, the depth-data was used since it highlights changes in the height of the hands and objects relative to the table. The images were scaled down to 50% of their original size in order to reduce the number or parameters in the neural network and to counteract overfitting. The initial learning rate of the Adaptive Moment Estimation (*adam*)-optimizer was set at 0.0001. Whenever the validation loss did not decrease during the training runs for five epochs, the learning rate was further decreased by a factor of five, until a minimum value of 10^{-6}. This step helps the loss converge.

The architecture of the neural network is shown in Table 1. The input layer contains no trainable parameters. As the name suggests its purpose is to bundle the input and forward it to the next layer. The input is a five-dimensional tensor. The first dimension represents

Table 1 Architecture of the neural network

layer (type)	Rows	Columns	Dimensions	Parameters
Input	106	128	32	0
ConvLSTM2D	106	128	32	40,448
Batch normalization	106	128	32	128
Max pooling 3D	53	128	32	20
ConvLSTM2D	53	64	8	11,552
Batch normalization	53	64	8	32
Dense	53	64	8	72
Flatten	1	27,136	27,136	0
Dense	1	1	4	108,548

the number of samples fed into the neural network during a discrete training step, also known as the batch size. The second dimension represents the number of individual images per sequence. Dimension three and four represent the height and width of the input frames. The last dimension represents the number of color channels. The Convolutional LSTM 2 Dimensions (ConvLSTM2D)-layer that was proposed by Shi et al. [16] extends a LSTM to have a convolutional structure, therefore enabling it to process the spatial temporal data of image sequences. The activation function of the second layer is hyperbolic tangent, while the output dimensionality is 64. In order to reduce overfitting, 50% of the neurons are dropped out at random during every epoch. A batch normalization is performed in the third layer. Thereafter a three dimensional max pooling operation is performed, reducing the height and width of the initial image in half, while the time dimension remains unchanged. Another ConvLSTM2D-layer, with a dropout of 50%, is applied after max pooling. This layer has eight output dimensions. It has to be noted that the output of this layer is a four-dimensional tensor, since the individual images are no longer represented at this point. Thereafter another batch normalization is performed. Afterwards follows the first densely connected neural layer with eight output dimensions. The second to last layer flattens the neural network to a series of individual neurons, which are then fed to the last densely connected layer. This last layer can be interpreted as a 4×1 vector, where each column represents one of the possible classes for classification. The activation function of the last layer is the SoftMax function. The neural network has 160.780 parameters in total.

5.3 Implementation

For the training of the neural network an NVIDIA RTX 2060 graphics card is used. The code was implemented in python, using the TensorFlow 2.3 framework [17].

To evaluate the model, a variant of a stratified tenfold cross validation was conducted. At first, the dataset was shuffled to randomly distribute the assembly operations. Afterwards, 20% of the dataset was reserved for the final evaluation. This test dataset was stratified so that the assembly operations were distributed evenly. The remaining 80% were divided into ten splits. Therefore, ten individual training runs were conducted. During these runs, nine splits were used for training and the remaining split was used for validation. This procedure was chosen so that a potential overfitting of the model can be evaluated. After each training run, another split was used for validation, until eventually all ten runs were completed. Each run was conducted for 30 epochs.

6 Results

The loss and accuracy of the training and validation data are depicted in Figs. 1 and 2. The training losses decay exponentially and converge at values from 0.25 to nearly zero. Because of this decrease, the training accuracy increases significantly in the first few epochs before converging near 100%. Up until the third epoch the accuracies and losses of the training and validation dataset rise and decay in unison.

The validation accuracy converges at about 60% after the fifth epoch. In the first few epochs, the validation losses generally decrease while still being very volatile. The validation loss of one of the runs even increases at this stage. While this particular run seems to be an outlier, the general phenomenon occurs due to the initial learning rate. At first, the learning rate is set at a value of 0.0001 in order to escape local minima of the loss function, which leads to volatile validation losses.

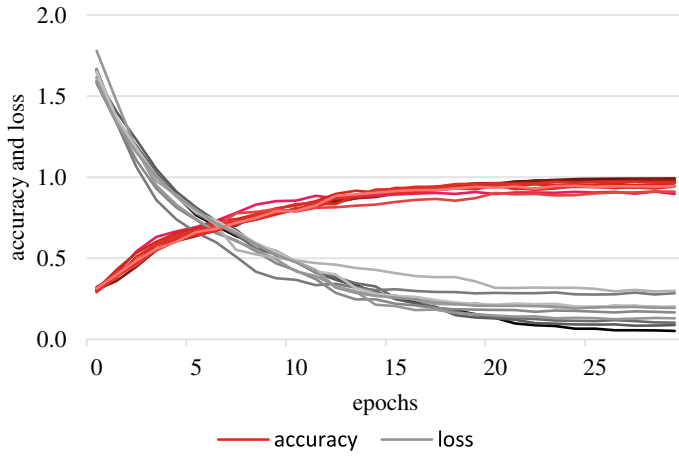

Fig. 1 Accuracy and loss of the training datasets converge to values near 1 and 0.25 to 0 for the ten different training runs

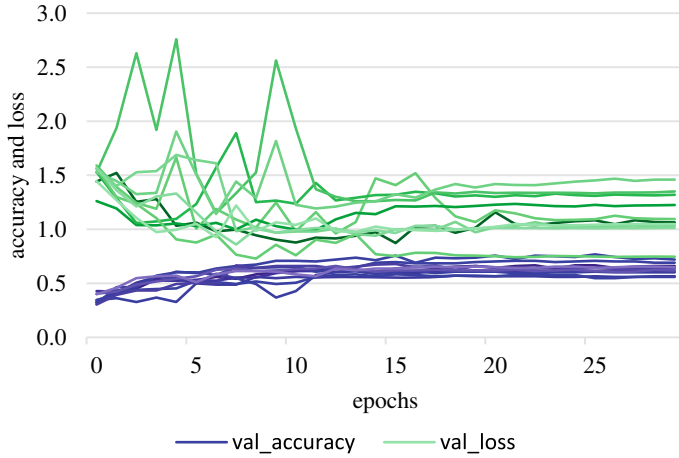

Fig. 2 Accuracy and loss of the validation datasets converge around 0.6 and 1 to 1.5 for the ten different training runs

The first reduction of the learning rate, which takes place between the tenth and eleventh epoch, significantly reduces the volatility of the validation losses. The validation loss converges at a value of 1 to 1.5 after 15 epochs. While the model does not overfit, it stops improving after the first 15 epochs.

The evaluation of the test dataset confirms the results of the validation dataset. The average accuracy is 59.6% with a standard deviation of 3.37% as shown in Table 2. The best training run has an accuracy of 64.3% while the worst run has an accuracy of 52.1%.

The confusion matrix of the average training run is shown in Fig. 3. The networks rarely predict reach/move to be grasp/release and vice versa, since those motions are fundamentally different from another. While reach/move has a dynamic course of movement, the assembly operations grasp/release show static characteristics. Although the distinction works most of the time in the case of reach/move, it does not work as well for grasp/release. Reach and move differ in the position of the tool or object after the assembly operation is finished. The difference between grasp and release is the inverted movement.

Since the algorithm is able to make a relatively clear distinction between the assembly operations reach and move, it can be assumed that it is able to implicitly detect objects. As

Table 2 Overview of the accuracy of the different training runs on the test dataset

No	1	2	3	4	5	6	7	8	9	10
Accuracy in %	64.3	61.0	61.0	56.7	59.5	63.3	52.1	59.0	61.7	57.4

Mean accuracy: 59.6%
Standard deviation: 3.37%

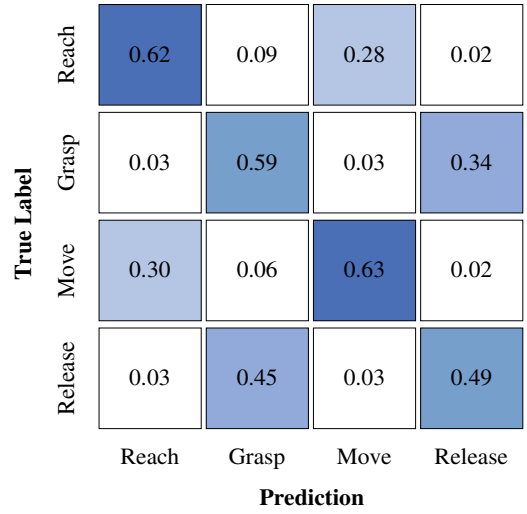

Fig. 3 Confusion matrix of the average training run. Correctly classified assembly operations are depicted on the main diagonal

the range of motion differs greatly between reach/move and grasp/release, it is deducted that the recurrent part of the neural network is able to process the spatial–temporal information contained in the image sequences very well.

7 Conclusion and Outlook

The recognition of assembly operations in an industrial manufacturing scenario is a challenging task due to the interaction with objects and tools as well as the constraints of manual assembly work stations. Due to these challenges, many approaches do not track individual assembly operations. Instead, they define classes like "general assembly" to circumvent the issue of complexity. This paper aimed to directly classify the assembly operations reach, grasp, move and release of the MTM-1 basic operations.

The dataset used in this approach consists of 2100 image sequences, which were fed into a RNN to process spatial temporal information. The average accuracy of the ten runs that were performed during the tenfold stratified cross validation is 59.6%. Most of the wrongful predictions are reach/move and grasp/release.

In order to achieve significant improvements in the training results, the dataset used must be enlarged, which allows the use of more complex neural networks. In addition, more people need to be involved in data collection so that individual characteristics have a smaller influence on the dataset and the task of classifying assembly operations can be better generalized.

Assembly operations are an interaction between the mechanic's hands and the tools or objects. The recognition and classification of these open up great potential for improvement. By adding this feature, conclusions about the assembly process can be drawn based

on the positions and movement patterns of the hands and objects, which can be directly processed by the neural network in order to significantly boost its performance. Therefore, this enhanced architecture is suited to decrease the wrong predictions for the case reach/move, since the position of the objects is detected by a sub-architecture of the neural network. Complementary to this, detecting the position of the hands as well as their orientation and trajectory results in a better performance regarding the case grasp/release, since more individual characteristics of the movements can be identified.

Acknowledgements The modular assembly system "Experimentelle Modulare Montageanlage (EMMA)", which was used in recording the data, was funded by the German Research Association (DFG).

References

1. Landherr, M.H.: Integrierte Produkt- und Montagekonfiguration für die variantenreiche Serienfertigung. Fraunhofer-Verlag, Stuttgart (2014)
2. Schröter, D.: Entwicklung einer Methodik zur Planung von Arbeitssystemen in Mensch-Roboter-Kooperation. Fraunhofer-Verlag, Stuttgart (2018)
3. Petruck, H., Mertens, A.: Using convolutional neural networks for assembly activity recognition in robot assisted manual production, In: M. Kurosu (eds), Human-Computer Interaction. Interaction in Context, LNCS, vol. 10902, pp. 381–397. Springer International Publishing, Cham (2018)
4. Liu, L., Liu, Y., Zhang, J.: Learning-based hand motion capture and understanding in assembly process. IEEE Trans. Industr. Electron. **66**(12), 9703–9712 (2019)
5. Root, M., Jauch, C.: Challenges of designing hand recognition for a manual assembly assistance system, In: Multimodal Sensing: Technologies and Applications, PROC SPIE, Munich (2019)
6. Rückert, P., Papenberg, B., Tracht, K.: Classification of assembly operations using machine learning algorithms based on visual censor data, In: 8th CIRP Conference of Assembly Technology and Systems, Procedia CIRP, Athens (2020)
7. Andrianakos, G., Dimitropoulos, N., Michalos, G., Makris, S.: An approach for monitoring the execution of human based assembly operations using machine learning. Procedia CIRP **86**, 198–203 (2019)
8. Deuse, J., Stankiewicz, L., Zwinkau, R., Weichert, F.: Automatic generation of methods-time measurement analyses for assembly tasks from motion capture data using convolutional neuronal networks—A proof of concept. In: Nunes, I.L. (ed.) Advances in Human Factors and Systems Interaction, pp. 141–150. Springer International Publishing, Cham (2020)
9. REFA Verband für Arbeitsstudien und Betriebsorganisation e.V.: Methodenlehre des Arbeitsstudiums: Teil 2 Datenermittlung, Hanser, München (1978)
10. Bokranz, R., Landau, K.: Handbuch industrial engineering: Produktivitätsmanagement mit MTM. Band 1: Konzept, 2nd ed., Schäffer-Poeschel, Stuttgart, (2012)
11. Deuse, J., Busch, F.: Zeitwirtschaft in der Montage, In: B. Lotter, H.-P. Wiendahl (Eds.): Montage in der industriellen Produktion, pp. 79–107. Springer Berlin Heidelberg (2012)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**, 1735–1780 (1997)
13. Buduma, N., Locascio, N.: Fundamentals of deep learning: designing next-generation machine intelligence algorithms. O'Reilly, Sebastopol, CA (2017)

14. Molchanov, P., Yang, X., Gupta, S., Kim, K., Tyree, S., Kautz, J.: Online detection and classification of dynamic hand gestures with recurrent 3D convolutional neural networks, In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4207–4215. Las Vegas, NV (2016)
15. Kenneth, L., Yanushkevich, S.N.: CNN+RNN depth and skeleton based dynamic hand gesture recognition, In: 24th International Conference on Pattern Recognition (ICPR), IEEE, pp. 3451–3456, Beijing (2018)
16. Shi, X., Chen, Z., Wang, H., Yeung, D.Y.: Convolutional LSTM network: a machine learning approach for precipitation nowcasting, In: 28th International Conference on Neural Information Processing Systems, vol. 1, (NIPS'15). MIT Press, pp. 802–810. Cambridge, MA (2015)
17. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin et al.: TensorFlow: A system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation, USENIX, pp. 265–283. Savannah, GA (2016)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.