Camera-based Progress Estimation of Assembly Work Using Deep Metric Learning

Takumi Kitsukawa[1], Sarthak Pathak[1], Alessandro Moro[1], Yoshihiro Harada[2], Hideo Nishikawa[2], Minori Noguchi[2], Akifumi Hamaya[2], and Kazunori Umeda[1]

***Abstract*— In this paper, a progress estimation method using deep learning is proposed to visualize the product assembly process in a factory. First, the target assembly product is detected from images acquired from a fixed-point camera installed in the factory using a deep learning-based object detection method. Next, the detection area is cropped from the image. Finally, by using a classification method based on deep metric learning on the cropped image, the progress of the product assembly work is estimated as a rough progress step. In addition, considering the similarity of features with neighboring steps when learning with deep metric learning, we propose an adaptive loss function that learns to separate features from nearby steps. In experiments, an 82 [%] success rate is achieved for the progress estimation method using deep metric learning. Furthermore, the method using the adaptive loss function achieved a success rate of 92 [%]. Experiments were also conducted to verify the practicality of a series of detection, cropping and progress estimation.**

I. Introduction

In recent years, shortage of labor in the manufacturing industry has become increasingly serious. Therefore, there is a need to improve productivity by increasing work efficiency on the production line. As a solution, automation of production lines using IoT and robots is being promoted. In particular, research on smart manufacturing is being promoted actively [1, 2]. However, automation has not progressed in high-mix, low-volume factories that produce products according to customer needs, and the majority of work is carried out manually. Such factories have not yet been converted to smart factories due to the lack of automated data capture. In particular, product assembly work requires changes to the production line each time product specifications change if automation is used. Therefore, the work is often performed manually, making automated measurement difficult. In addition, it is not desirable to install new measurement sensors at the site due to cost and labor considerations.

In response to this, it is considered effective to introduce a system that automatically manages work by acquiring images from fixed-point cameras installed in factories and estimating the progress of the work. So far, research has been conducted on motion recognition using skeletal point information of a person in a time series [3] and on progress estimation by observing hand movements from the operator's point of view [4]. However, the assembly of large equipment often requires several days of work and the assembly sequence is not defined in detail, so progress estimation using worker movements and time series data is not suitable because the movements might change from person-to-person.

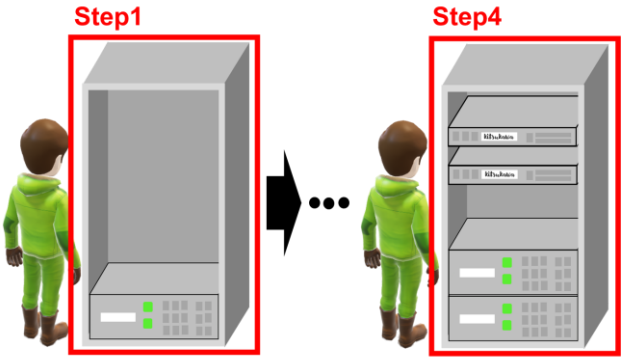

Fig. 1 Proposed method expected result

The aim of this research is to construct a system that estimates the progress of assembly work by focusing on the objects to be assembled. An overview of the proposed system is shown in Fig. 1. Progress estimation uses a method that judges which step has been reached in relation to the assembly progress step set earlier.

II. Use of Metric Learning

To construct a system that estimates which step currently belongs to a given step, the problem can be set up as a class classification problem. Deep learning methods using CNN (Convolutional Neural Networks) are highly accurate for classifying classes in images. However, the class classification in this assembly progress estimation is considered to have small differences in appearance between classes, and it is required to extract detailed feature differences and to have as much separation as possible between steps that are far apart. Therefore, the use of metric learning [5, 6], one of the machine learning methods, is considered.

Metric learning is a method for learning patterns that transforms input data into a feature space so that the samples are separated into classes based on the distance or similarity between them. The aim of metric learning is to increase the distance between samples of different classes while decreasing
the distance between samples of the same class. In particular, deep metric learning [7], which uses a multi-layer neural network to extract features when learning patterns, has also been proposed.

[1]The Course of Precision Engineering, School of Science and Engineering, Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo, Japan (Corresponding author: kitsukawa@sensor.mech.chuo-u.ac.jp)

[2]Hitachi High-Tech Solutions Corporation, 1-17-1 Toranomon, Minato-ku, Tokyo, Japan

2023 IEEE/SICE International Symposium on System Integration (SII) | 979-8-3503-9868-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/SII55687.2023.10039109

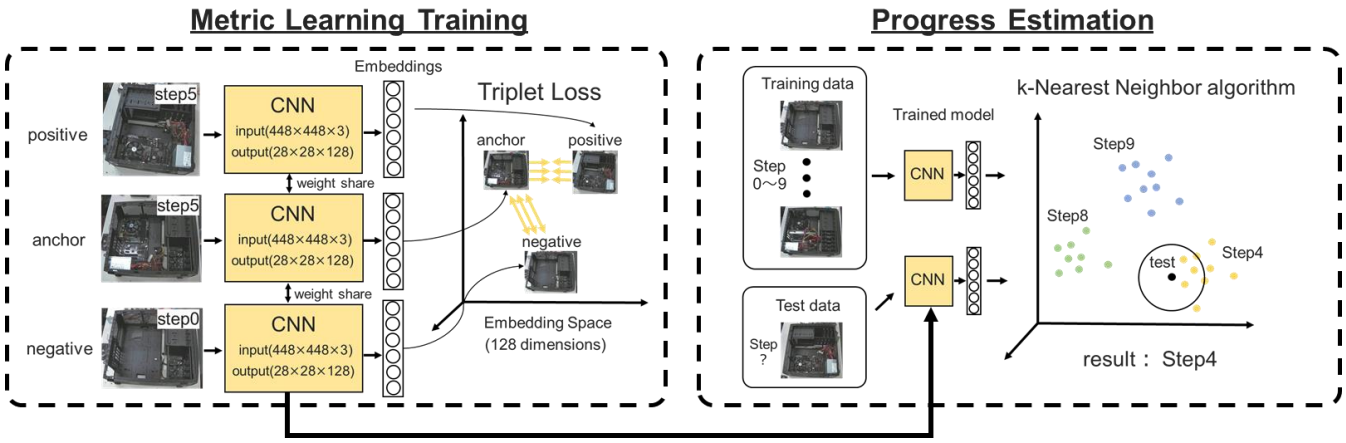

Fig. 2 Network structure of the proposed step estimation method

In general class classification, the features extracted by the CNN are passed through Fully Connected layer and converted into class affiliation probabilities using the SoftMax function. However, this means that the feature extraction network is trained without considering the distance between samples of the same class and other classes. Deep metric learning, on the other hand, can obtain discriminative features by deliberately increasing the distance between samples of other classes and decreasing the distance between samples of the same class. Therefore, it performs well when the samples of each class are small or when there are unknown classes.

Considering the practical application in an actual factory, the assembly work progress estimation using our proposed method will require only a small amount of data to learn from. Therefore, we propose the use of deep metric learning, which can learn feature differences even with small amount of data.

III. Proposed Method

A. Overview of the Proposed System

An overview of the proposed system is shown in Fig. 3. In the initial phase, an object detection model is trained on a custom dataset to be able to detect the position of the object to be assembled in the image. In addition, a step estimation model is trained to estimate the progress of the cropped image.

The flow of the system is to detect objects in the image acquired from a fixed-point camera using an object detection method, crop their positions and estimate their progress using a step estimation model based on deep metric learning.

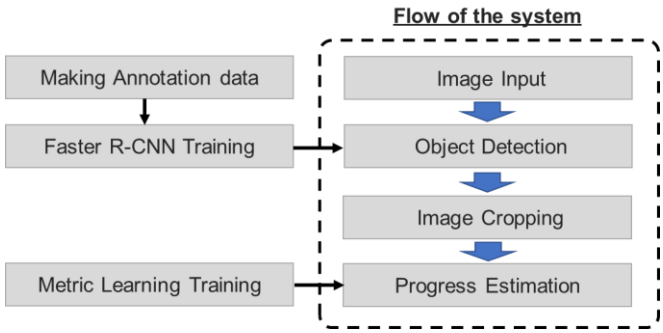

Fig. 3 Flow of the proposed system

B. Step Estimation Method

The structure of the proposed step estimation method is shown in Fig. 2. First, the steps to be judged in the assembly progress are set and training data are prepared. The method is to cut out the part of the object from the assembly video and save it separately for each step. One image is randomly selected from the training data and an anchor sample is set. The positive sample is the image from the same step as the anchor sample, and the negative sample is the image from a different step. In the example in Fig. 3, the image from step 5 is set as the anchor sample. Another image from step 5 is selected as a positive sample, and an image from a different class than the anchor sample, step 0, is selected as a negative sample. Next, these three images are input to a four-layer CNN model and then to a one-layer all-unions layer to obtain a 128-dimensional feature vector. The weights of the CNN models are shared. The parameters are updated by defining a loss function to increase the distance between the anchor sample and the positive sample and to decrease the distance between the anchor sample and the negative sample among the three feature vectors obtained. Stochastic Gradient Descent is used in the optimization algorithm to learn by mini batch.

The loss function uses Triplet Loss [8], which calculates the relative distance between the anchor, positive and negative samples as a set of data. The equation is as follows:

$$L_{triplet} = max(d_p - d_n + m,\ 0) \qquad (1)$$

where $\boldsymbol{d_p}$ is the distance between the anchor sample and the positive sample in the feature space and $\boldsymbol{d_n}$ is the distance between the anchor sample and the negative sample. $\boldsymbol{m}$ is an arbitrary constant that represents the degree of movement of the distance away/near work, called the margin. The Euclidean distance in the feature space is used to calculate the distance. In other words, by reducing this loss function, features of images at the same step can be brought closer together in feature space and features of images at different steps can be moved away from each other in feature space.

Next, the learned CNN model is used to embed each image to the metric learning feature space. First, the images used for

each step in the training are input again one by one into the trained model and embedded in the feature space. Once all the images have been converted into feature vectors, the input images to be estimated are also input into the learned model and embedded in the feature space. For these embedded data, the k-Nearest Neighbor algorithm is used to estimate which step the unknown test data belongs to. Furthermore, by judging errors as those whose distance from all the clusters in the feature space in the feature space is farther than a threshold value, it is possible to detect assembly errors in assembly work and to reduce misjudgments that occur in occlusion.

C. Adaptive Triplet Loss

The estimation of progress steps in assembly operations is considered to be prone to misjudgment of close steps due to the similar features of neighboring steps. Therefore, we propose a method to adaptively change the value of the margin of Triplet Loss to be used so that it becomes larger when the steps are close. The formula uses a normal distribution so that the margin falls smoothly from nearby to faraway steps. The proposed
adaptive margin formula is as follows:

$$m = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(n_n - n_a)^2}{2\sigma^2}\right) \cdot a \qquad (2)$$

where $\boldsymbol{\sigma^2}$ is the variance of the number of steps, $\boldsymbol{a}$ is an arbitrary constant representing the size of the margin, $\boldsymbol{n_a}$ is the number of steps in the anchor sample and $\boldsymbol{n_n}$ is the number of steps in the negative sample. The examples of adaptive margins when learning steps 1 and 5 as anchors are shown in Fig. 4.

D. Object Detection

For object detection and training data preparation, we will use Faster-R CNN [9] and Siam Mask [10], which have been validated in our research [11].

The Faster-R CNN used in object detection has a model structure that discriminates whether the contents of a rectangle are objects or background and classifies the detected regions. Unlike conventional object detection methods such as R-CNN [12] and Fast R-CNN [13], it extracts candidate object regions using a CNN (Convolutional Neural Network) structure called RPN (Region Proposal Network) [14], which can significantly reduce processing time. Object detection is enabled by fine-tuning this pre-trained model of Faster R-CNN with a custom dataset created for assembly products.

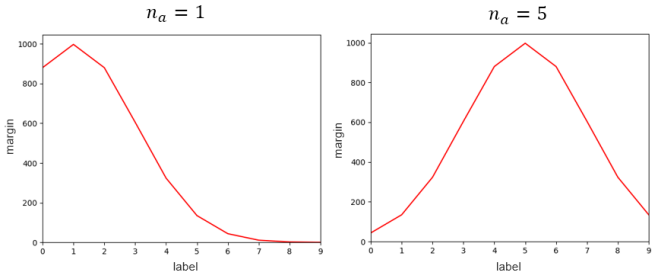

Fig. 4 Examples of adaptive margins in learning step 1 and 5

Siam Mask, used to create training data, is a mask-based object tracking method. This tracking method allows for continuous labeling of the video for training. Specifically, given the position of an object to be tracked in the first frame of the video, it will estimate the object's position in all subsequent frames. This allows for efficient creation of training datasets.

IV. EXPERIMENTS

Experiments were conducted to evaluate the proposed step estimation model and the system as a whole. A desktop PC was used as the object of the assembly product. The camera was an IO DATA network camera “Qwatch TSWR-LP” [15] and the pixel value was set to 1980×1080. Two cameras were set up in order to get more training data in a one-time assembly operation. The positions of the two set up cameras and the actual images captured are shown in Fig. 5. The camera in the close-up position is about 0.8 m away from the object and the angle is about 60° down from the horizontal. The camera in the high angle position is about 1.8 m away from the object and the angle is about 90° down from the horizontal. Our proposed method requires that the steps of the progression be set up ahead of time by yourself. The steps to be judged were set to 10 steps from step 0 to step 9. The steps are shown in Fig. 6. In the desktop PC assembly experiment, the steps to be judged were the 10 steps from step 0 to step 9. The steps are shown in Fig. 6.

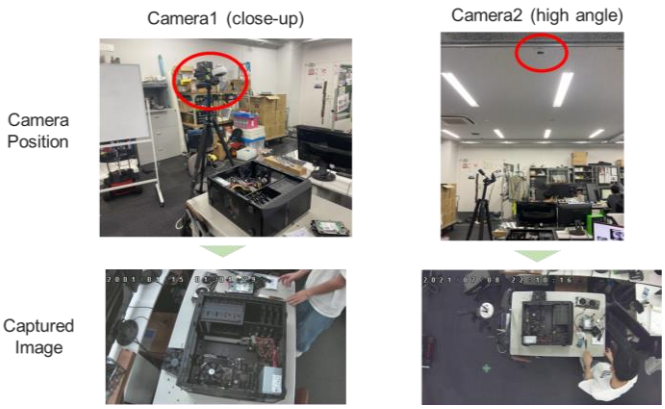

Fig. 5 The camera position that was set and the image captured

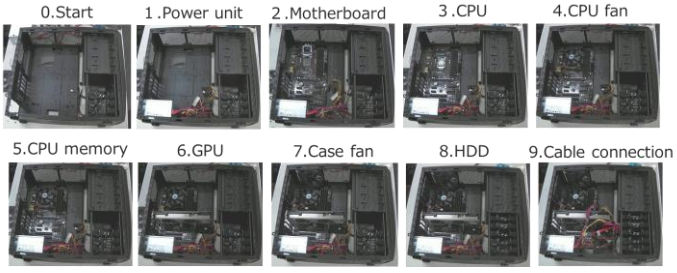

Fig. 6 The point of dividing steps

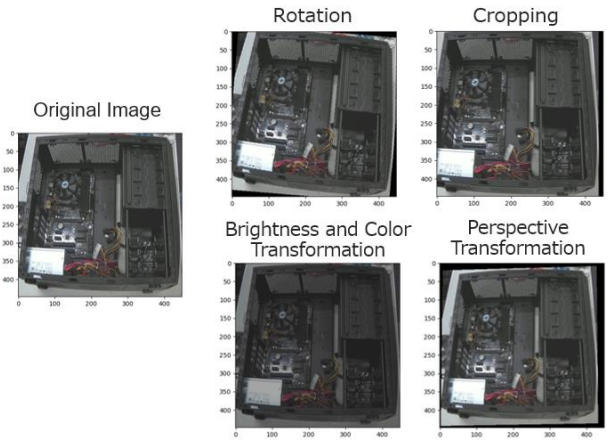

Fig. 7 Image transformation for data augmentation

A. *Experiments on Step Estimation*

Experiments were conducted to confirm the effectiveness of the step estimation model. The experimental results between the method with a constant Triplet Loss margin and the method using an adaptive margin were compared.

Experimental conditions. A total of 40 images were used for training, four at each step. To prepare the dataset, the location of the target desktop PC was cut out from each image and stored in separate folders for each step. The image size and number of convolution layers of the deep metric learning model to be implemented is the proposed model described in Chapter III. The constant k of the k-nearest neighbour method used to judge step was set to k = 4. This is because the number of training data to be input into the trained model in the progress estimation phase is four per step. In addition, as this study assumes a small amount of training data, the data was expanded by randomly adding rotation, projective transformation, colour change and partial truncation to the input images in order to avoid data bias. The actual images processed for data expansion are shown in Figure 7. The method with fixed margins and the method with adaptive margins were each trained for 10000 epochs. The results evaluated on test data are shown in Figs 8, 9 and 10.

Fig. 8 shows the success rate against the number of learning epochs. With fixed margin, 82.0 [%] and 91.8 [%] correct responses were achieved at 10000 epochs with fixed margin and adaptive margin, respectively. In addition, the success rate for fixed margin reached its head around 3000 epoch, while the success rate for adaptive margin increased gently.

Fig. 9 shows the confusion matrix of estimated and correct steps. It can be seen that both margin-fixing and adaptive margin are able to estimate steps with high accuracy. However, a closer look reveals that in the fixed margin case, misclassification with the neighbouring step occurs around steps 7, 8, and 9. In contrast, with adaptive margin, the misclassification with neighbouring steps at steps 7, 8 and 9 is reduced and the accuracy is improved.

Fig. 10 shows the 128-dimensional feature vector obtained by inputting the test data to the trained model, converted to 2 dimensions and visualised; the dimensionality reduction algorithm t-SNE [16] was used for the conversion to 2

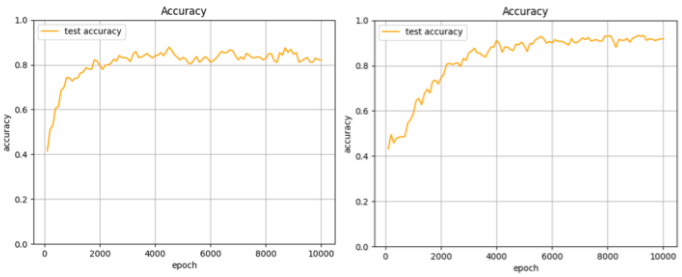

Fig. 8 Results of accuracy
Left: fixed margin, Right: adaptive margin

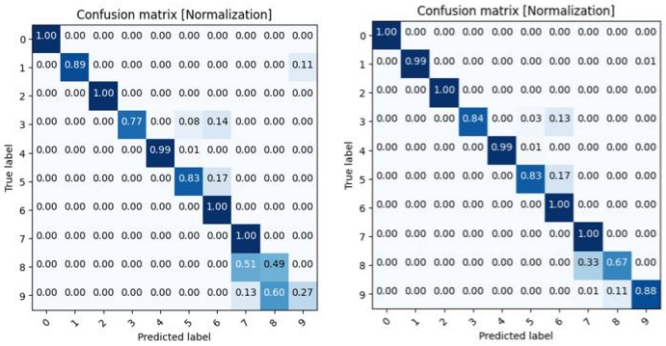

Fig. 9 Confusion matrices at the predicted and true steps
Left: fixed margin, Right: adaptive margin

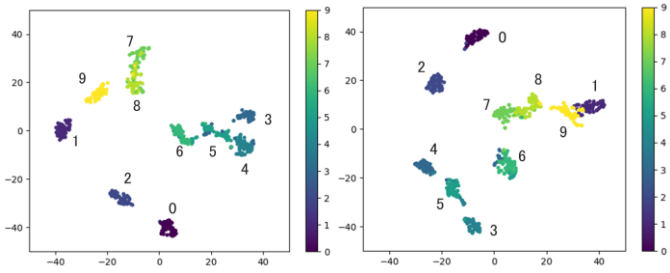

Fig. 10 Visualization of feature space using t-SNE
Left: fixed margin, Right: adaptive margin

dimensions. The colour of the plotted points indicates the correct step, and the better the results are clustered step by step, the better the results are. The present results show that steps 3, 4, 5, and 7, 8, 9 which were mixed in the fixed margin, are slightly more coherently distinguishable in the adaptive margin.

These results confirm that deep metric learning can be used to find differences in features and make step decisions with high accuracy even with small amounts of data. It was also confirmed that the proposed adaptive margin method can reduce errors with neighbouring steps and improve accuracy.

B. *Experiments on the Whole System*

Experiments were conducted on a series of steps from object detection to cropping and step estimation.

Experimental conditions. The training data for the detection was prepared by labelling approximately 1000 images with an annotation tool that was created. Object detection was implemented with Detectron2 [17], a library for deep learning, which enabled the detection of desktop PCs by fine-tuning a Faster R-CNN model pre-trained on MS COCO

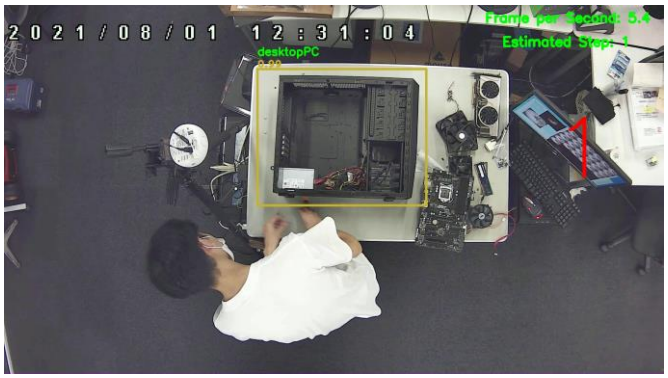

Fig. 11 The view of successful detection and step estimation

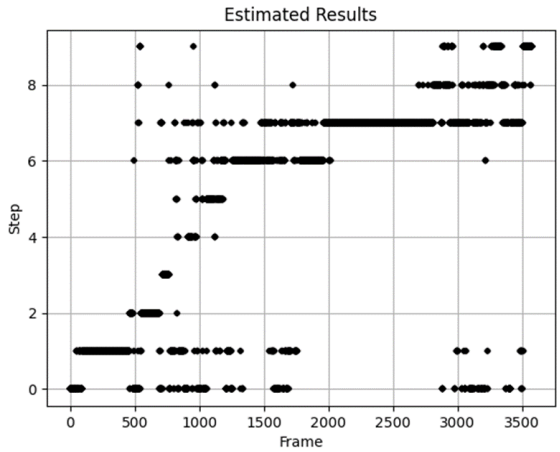

Fig. 12 Relationship between time and estimated steps

datasets [18] with a custom dataset. In order to avoid misjudgments caused by worker occlusion, a method was used to fix and log the decision step after 20 consecutive frames of the same step detection result. As a result of the experiment, correct decisions were logged at all steps. Fig. 11 shows the actual system in action. The yellow bounding box represents the detection of the object, and the red numbers on the right-hand side represent the steps that were judged.

A graph with time on the horizontal axis and estimated steps on the vertical axis is also shown in Fig. 12. The unit of the horizontal axis is the Frame number. The graph shows that the estimated step number is going up one by one, which confirms the effectiveness of the system. However, although not continuous, misjudgments of step 0 and 1 can be seen. This is thought to be since the operator's head overlaps the desktop PC and appears black, making the object look similar to steps 0 and 1. In this experiment, the method to fix the estimated step was used when the same step was judged for 20 consecutive frames. Therefore, occasional misjudgments were not a problem. The use of worker detection could further solve this problem.

C. *Evaluation of Processing Speed*

Considering the practical application of this research, the processing speed of the proposed method was measured.

Experimental conditions. The PC used for measuring had an Intel Core i7-8700 3.7GHz CPU and NVIDIA GeForce RTX 2080 GPU. The time taken for each processing when the resolution of the input image is 1920×1080 and 1280×720 is shown in Table I.

TABLE I. PROCESSING SPEED OF PROPOSED SYSTEM

	Detection [ms]	Step Estimation [ms]	Whole System [ms]	Whole System (frame rate)[fps]
1080p	188.4	11.6	202.3	4.9
720p	181.4	10.8	192.5	5.2

Note that this time the processing speed is based on the case where there is only one object. Comparing detection and step estimation, object detection takes about 16 times longer than step estimation. This is thought to be related to the fact that step estimation uses a cropped image as input and the image size is small. Although the processing time is shorter with 1280×720, there was no significant change in processing speed due to the difference in input image size. The overall system processing speed was about 5 [fps], which is considered sufficient for real-time measurements.

V. CONCLUSION

We proposed a progress estimation system for assembly work focusing on objects. Specifically, we proposed a progress estimation method that detects and crops the object and uses deep metric learning to judge the steps. We also proposed a loss function with an adaptive margin to reduce misjudgments between neighboring steps with similar features. Experiments showed the effectiveness of the method by evaluating the step estimation part and the whole system including detection.

In the experiments on the step estimation part, the method with fixed margins achieved 82.0 [%], while the method with adaptive margins achieved 91.8 [%] accuracy. Experiments on the whole system successfully achieved progress estimation in a series of detection, cropping and step estimation. The processing speed was about 5 [fps], which was sufficient for real-time measurements.

As a future work, the system is required to record the results of time-series estimation in an environment with multiple objects. The proposed system does not link IDs when multiple objects are detected. Therefore, we believe that by constructing a system that includes a tracking method, it will be possible to perform time-series connected measurements for each object.

We also believe that the proposed method can be generalized to various assembly tasks. We would like to conduct experiments on other assembly products in the future.

REFERENCES

[1] H. S. Kang, J. Y. Lee, S. S. Choi, H. Kim, J. H. Park, J. Y. Son, B. H. Kim and S. D. Noh," Smart manufacturing: Past research, present findings, and future directions," International Journal of Precision Engineering and Manufacturing-Green Technology 3, pp.111-128, 2016.

[2] J. Wang, Y. Ma, L. Zhang, R. X. Gao and D. Wu, "Deep learning for smart manufacturing: Methods and applications," Journal of Manufacturing Systems, vol.48, Part C, July 2018, pp.144-156.

[3] Hitachi, Lumada," Ensuring consistent product quality and improving productivity by work-video analysis and data visualization of the results," https://www.hitachi.com/products/it/lumada/global/en/spcon/uc_01726s/index.html, Accessed: 2022-10-29.
[4] M. Funk, A. Bachler, L. Bachler, T. Kosch, T. Heidenreich, A. Schmidt, "Working with Augmented Reality? A Long-Term Analysis of In-Situ Instructions at the Assembly Workplace," PETRA'17, 2017.
[5] E. Xing, M. Jordan, S. J. Russell, A. Ng, "Distance Metric Learning with Application to Clustering with Side-Information," Neural Information Processing Systems (NIPS), 2002.
[6] K. Q. Weinberger, J. Blitzer, L. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," Neural Information Processing Systems (NIPS), 2005.
[7] J. Lu, J. Hu, J. Zhou, "Deep Metric Learning for Visual Understanding: An Overview of Recent Advances," IEEE Signal Process. Mag. 2017, 34, pp.76–84.
[8] E. Hoffer, N. Ailon," Deep metric learning using Triplet network," International Workshop on Similarity-Based Pattern Recognition, SIMBAD 2015, pp.84-92.
[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," Neural Information Processing Systems (NIPS), 2015.
[10] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast online object tracking and segmentation: A unifying approach," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[11] T. Kitsukawa, M. Takahashi, A. Moro, Y. Harada, H. Nishikawa, M. Noguchi, A. Hamaya, and K. Umeda, "A Recognition System of AR Markers Attached to Carts in a Factory," 2022 IEEE/SICE International Symposium on System Integration (SII2022), pp.608-613, 2022.1.
[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Computer Vision and Pattern Recognition (CVPR), 2014.
[13] R. Girshick, "Fast R-CNN," IEEE International Conference on Computer Vision (ICCV), 2015.
[14] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. "High performance visual tracking with Siamese region proposal network," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[15] IO DATA Co., Ltd.: "Network Camera Qwatch TS-WRLP," https://www.iodata.jp/product/lancam/lancam/ts-wrlp/index.html, Accessed: 2022-07-22.
[16] L. van der Maaten and G. E. Hinton, "Visualizing Data using t-SNE," Journal of Machine Learning Research, vol. 9, pp.2579-2605, 2008.
[17] Facebook AI: "Detectron2," https://ai.facebook.com/tools/detectron2/, Accessed: 2022-07-22.
[18] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common objects in context," Computer Vision and Pattern Recognition (CVPR), 2015.