Regular Paper

Target Area Extraction for Object Recognition from Gesture Trajectory Detected Using YOLO

Tsukasa Kudo†

†Faculty of Informatics, Shizuoka Institute of Science and Technology, Japan
kudo.tsukasa@sist.ac.jp

Abstract -

Advances in mobile cameras such as smart glasses and object recognition technologies have made it possible to extract and utilize various types of information seen while working or walking. And, when the object is small in the camera's view, it is necessary firstly to perform object detection to extract the object area. Recently, object detection methods using deep learning have been actively studied, and their applications have spread to various fields. However, using deep learning requires preparing training data for each object, which is challenging when various objects are targeted. In this study, instead of these deep learning–based object detection methods, a method of extracting the target area from the trajectory of a gesture detected using You Only Look Once (YOLO) is proposed. An advantage of this method is that the YOLO model can be trained using the training data of only a specific body part, such as a fingertip. Furthermore, the experiments show that, contrary to the previously mentioned object detection methods, even the target area and video frame without special features can be extracted according to the purpose such as extracting an arbitrary area of an image.

Keywords: YOLO, gesture detection, object recognition, transfer learning, object detection

1 INTRODUCTION

Recently, object recognition for videos and images has been actively studied. Its applications are expanding into various fields, such as automatic car driving and immigration control using face recognition. And with the development of the Internet of Things (IoT), especially edge computing that performs computations as close to data sources as possible, its application field is expanding to mobile cameras such as smart glasses [15]. During object recognition, when the target in the image is small, the target area must first be extracted using object detection. For example, face detection using the Haar-like feature is performed during face recognition to extract the face area as a bounding box, and then face recognition is performed on this face area [20].

With the advancements in deep learning, various methods for object detection have been actively studied [12]. For example, You Only Look Once (YOLO) detects the target as a bounding box and simultaneously recognizes the target [17]. Furthermore, videos can be processed in real-time by YOLO [21]. However, these deep learning–based methods require training data for each target, so it is difficult to apply them to fields where various objects are targeted.

As such a field, the author has been working on the automation of inventory management in machine factories. The aim of this research is to automatically determine the target parts and their inventory number from the videos of smart glasses worn by workers. However, since there are thousands of different parts in such factories, the above-mentioned challenge of preparing the training data arose [4].

In this paper, a method of extracting the target area from the trajectory of a gesture detected using YOLO is proposed. The gesture is performed so that, for example, the target area is a closed area surrounded by its trajectory. Hence, the target area extracted from this trajectory can be the same as the object detection result. Notably, since gestures can be performed by a specific body part, such as a fingertip, this method requires only one type of training data for various object detections.

The detected trajectory usually includes erroneously detected points (subsequently called noises) and extra gesture sections. Therefore, in this method, the noises are removed using the median of nearby points, and the target section or feature point is indicated by the stop motion of the gesture. Furthermore, the experiments on actual use cases show that the arbitrary area of an image and frame of a video can be extracted. Consequently, object recognition can be done using simple methods, such as optical character recognition (OCR) and template matching.

The remainder of this paper is organized as follows. Section 2 presents related work and the aims of this study, and Sec. 3 proposes a method for extracting the target area from a gesture trajectory. Section 4 shows the implementation of the proposed method and the extraction of the target area through experiments. Section 5 shows the applications and evaluations of this method, and Sec. 6 discusses the evaluation results. Finally, Sec. 6 concludes this paper.

2 RELATED WORKS AND AIM OF THIS STUDY

In a machine factory, parts are stored in bulk containers as shown in Fig. 1 (3). However, as can be seen from Fig. 1 (3), the number of parts cannot be visually counted, which has been a problem in inventory management. For this problem, the authors have shown that deep learning can be used to estimate the number from images of containers with practical accuracy and that computer graphics (CG) can be used to efficiently prepare training data [3], [7].

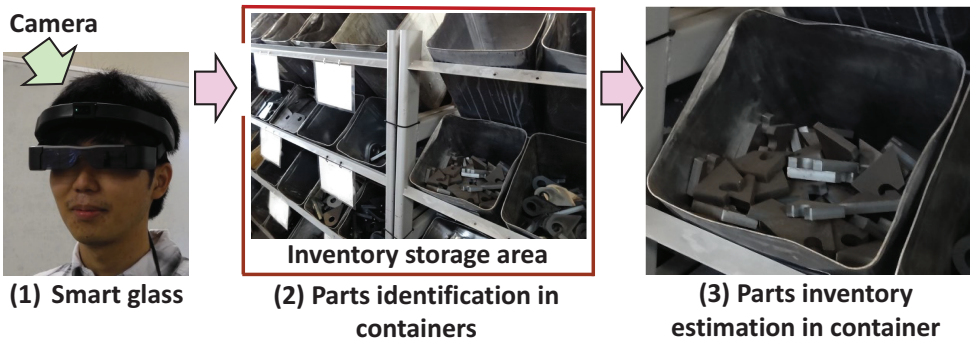


Figure 1: Inventory management of parts with smart glass

However, in such factories, there are usually several thousand types of parts stored dispersedly in the inventory storage areas in the factory. To manage such widespread objects, methods combining smart glasses and object recognition such as deep learning have been proposed and systems have been studied to assist factory workers and the visually impaired [14], [22].

So, I focused on the fact that the parts inventory fluctuates due to replenishment and picking (i.e., taking out the parts to be used) by workers, and attempted to automatically extract inventory information from videos continuously captured by smart glasses worn by workers shown in Fig. 1 (1). Concretely, object recognition with deep learning was used to identify the worker's entry into the inventory storage area, the target container, and parts from the videos, as shown in Fig. 1 (2). However, since the parts are small in the camera's view, the accuracy of the discrimination degraded [4]. In such a case, the target area must be extracted using object detection prior to object recognition.

Various methods for joint object detection and recognition have been proposed. Faster R-CNN performed them in a lump through collective end-to-end training of both models [18], and YOLO executed them with a single neural network to improve efficiency [17]. Regarding differently scaled objects, SSD could process them collectively [11]. RetinaNet improved the efficiency by introducing the Feature Pyramid Network (FPN) and improving the loss function [9], [10]. Subsequently, M2Det further improved accuracy and efficiency by introducing the new FPN and loss function [23].

Among these methods, YOLO has been improved repeatedly through version upgrades, with several models currently available [2]. YOLO has a high detection efficiency and accuracy and can be applied to real-time object detection and recognition [21].

However, since the above methods use deep learning, training data (i.e., ground truth) comprising images and the correct labels for model training must be prepared. For example, YOLO requires ground truth comprising the images and their corresponding labels of bounding box positions and object classification for each target. Here, object classification indicates the object type to be recognized. Therefore, this creates a significant burden in preparing training data when various objects are to be detected.

In the above-mentioned machine factory, since there are thousands of types of parts, it is not practical to apply object detection using deep learning. So, I focused on the fact that the operator picks up the parts when working, and tried a method of using optical flow for parts detection. However, it was found that the detection accuracy degraded when the background was complex [5]. In addition, using Cycle-GAN, which is a kind of generative adversarial network, I showed that the contour of a container can be detected [6]. However, this method did not cover the detection of parts in containers.

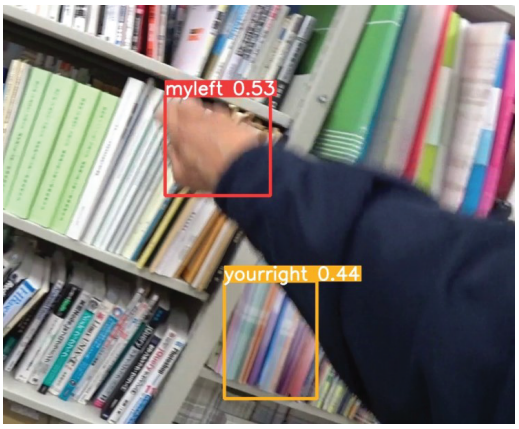


Figure 2: Object detection result using YOLO

The motivation for this study is the idea that the target area of an arbitrary object can be extracted by detecting the trajectory of a specific object, such as the tip of a hand indicated by a gesture in a video. Using this idea, I expected that randomly stored parts in containers shown in Fig. 1 (3) can be detected.

By the way, trained models and training data for YOLO for various objects, such as the coco dataset, are available on the Internet [1], [2], [8]. Therefore, YOLO can be easily used for real-time object detection and recognition from videos when targeting specific objects. That is, this trajectory detection can be performed in real-time by YOLO targeting only one object, and training the model is easy.

Furthermore, several applications have been proposed for hand gesture recognition using deep learning, such as conversation and device control [13], [16], [19]. However, application studies on extracting the target area, such as object detection, could not be obtained.

Figure 2 shows the resulting image of object detection for the right hand by YOLO from a video frame. The detected

objects are indicated by bounding boxes, and the target type and estimation accuracy of the object is indicated above the upper side of each box. For the detected boxes, my left hand (myleft) was detected with an accuracy of 53% (0.53) on the upper side. However, there is falsely detected noise on its lower side, and its accuracy is 44%.

The aims of this study are stated below. The first is to develop a method that accurately extracts the target area from the trajectory of gestures detected using YOLO. To achieve this, the noise shown in Fig. 2 and the extra gesture section should be removed to improve the accuracy of the detected trajectory. The second is to demonstrate the effectiveness of this method. For example, this method requires an extra action (a gesture) compared to object detection using YOLO directly. On the other hand, in this method, only a single body part is required for model training, regardless of the number of target object types; this method has the advantage of extracting arbitrary areas for the purposes, such as areas without special features. Hence, to clarify what application fields are effective based on actual business use cases.

3 TARGET AREA EXTRACTION METHOD FROM GESTURE TRAJECTORY

Two steps are used to extract the target area using gesture trajectory: (1) accurate trajectory detection, for which noises and extra gesture sections are removed, and (2) target area extraction from the detected gesture trajectory.

3.1 Gesture Trajectory Detection Method

This method uses the trajectory of a gesture by a specific body part. Moreover, it indicates the target gesture section with stop motions. For example, to indicate the area of some books shown in Fig. 2, the gesture is paused when the hand moves to the target book; subsequently, it is moved along the outline of the books. Finally, it is paused at the end of the outline and moved away. The gesture section between these two pauses is used to extract the area of the book. In the following description, the upper left corner of the bounding box (Fig. 2) is used as the gesture indication for the detected point.

The following procedure detects gesture trajectory: (1) valid points are selected from the detected points using YOLO, (2) noise is removed using the medians of nearby points, and (3) the target section is extracted as described above.

3.1.1 Valid Point Selection

In this method, there is a maximum of one valid point in each frame. To select this valid point from the detected points, the following condition is adopted. Its accuracy is the highest in the frame and greater than the threshold (i.e., the specified value). If each pair of the detected point coordinates and accuracies of the i-th frame is indicated by c_{ij} and a_{ij}, the valid point with a coordinate s_i in Eq. (1) is selected.

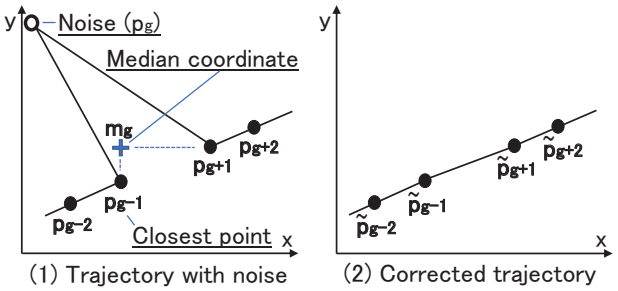


Figure 3: Noise elimination using median coordinates

$$s_i = \begin{cases} c_{ik} & (\exists k(a_{ik} \geq V \wedge a_{ik} = max(\{a_{ij}\}))) \\ \emptyset & (\forall k(a_{ik} < V)) \end{cases} \quad (1)$$

Here, $\emptyset$ indicates that the frame does not have a valid point; V indicates the threshold; $\{a_{ij}\}$ indicates the set of a_{ij}.

In Fig. 2, if $V = 0.4$, then the upper-left coordinate of the upper bounding box is selected because its accuracy is the highest in the two boxes and greater than the threshold. However, if $V = 0.6$, this frame is ignored.

3.1.2 Noise Elimination Using Median Coordinates

For each frame, the median coordinates are calculated with the valid points of the nearby frames to eliminate noises. Let $p_j(j = 1, 2, 3, \cdots)$ be the ordered set of coordinates with eliminating s_i if $s_i = \emptyset$ from $\{s_i\}$, namely the set of s_i in Eq. (1). Figure 3 (1) shows an example where p_g is a noise.

The median of p_g is constructed using the section before and after the index g. Let R_g indicate the set of indices of this section, and let p_{Rx} and p_{Ry} indicate the set of x-coordinates and y-coordinates, respectively. The median of this section is defined as follows:

$$m_g = (median(p_{Rx}), median(p_{Rx}))$$

Here, *median* is the function to get the median value of the coordinates. In Fig. 3, the coordinate m_g with the median of each of the x–y coordinates is selected.

The noises are eliminated using these median coordinates. The median coordinate $\tilde{m}_g$ of the frame g is defined as follows:

$$\tilde{m}_g = \{p_h | \exists h(dist(p_h, m_g) = min(dist(p_n, m_g)) \wedge \forall n \in R_g)\} \quad (2)$$

$dist(p_n, m_g)$ indicates the distance between p_n and m_g. Conversely, $\tilde{m}_g$ denotes the coordinate $p_r(\exists r \in R_g)$ of the closest point to the median coordinate m_g. Hence, the median trajectory is defined as the set of points $\tilde{p}_g$ shown in Eq. (3).

$$\tilde{p}_g = \begin{cases} p_g & (\tilde{m}_g = p_g) \\ \emptyset & (\tilde{m}_g \neq p_g) \end{cases} \quad (3)$$

That is, if p_g is not the closest coordinate to m_g in R_g, then it is converted to $\emptyset$, that is, it is eliminated; otherwise, p_g is

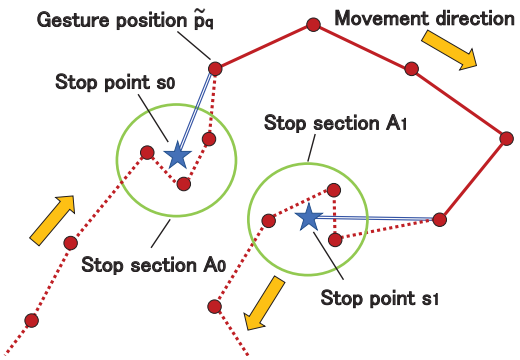


Figure 4: Target gesture section extraction way

adopted for $\tilde{p}_g$. Thus, noises are eliminated from the median trajectory.

In Fig. 3, the closest coordinate to the median $\tilde{m}_g$ is p_{g-1}, so the coordinate $\tilde{p}_g$ is set to $\emptyset$ and eliminated. The other coordinates are set as $\tilde{p}_n = p_n(r \in R_g)$. Consequently, a trajectory is corrected using the median coordinate, as shown in Fig. 3 (2).

3.1.3 Extraction of Target Section Using Stop Points

Let $q(q = 1, 2, 3, \cdots)$ be the coordinates of the q-th point in the set of $\tilde{p}_g$ denoted in Eq. (3), namely points other than noises. To extract the target section of a gesture, two stop motions are detected. The stop section is defined as the section where the distance between $\tilde{p}_q$ and $\tilde{p}_{q+1}$ is below or equal to a specified threshold H. This is the section in which the gesture is almost stopped and is shown as A_0 and A_1 in Fig. 4.

k-th stop section is expressed as follows:

$$A_k = \{\tilde{p}_q | (dist(\tilde{p}_{q+1} - \tilde{p}_q) < H) \wedge (\tilde{p}_q \notin A_l), l < k\} \quad (4)$$

Here, $dist$ is the same as in Eq. (2). Note that the threshold H is used because actual fingertip gestures cause slight fluctuations, even when the fingertip is stopped.

For the stop section A_k, stop point s_k is defined as the point whose coordinates are the simple average of the coordinates of all points in A_k. Figure 4 shows that the point $\tilde{p}_q \in A_k$ in the stop section is replaced with the stop point s_k, and the points before and after the stop section indicated by the dashed lines are eliminated.

Consequently, the target section of the gesture trajectory (subsequently called the detected trajectory) is extracted. This is indicated by the solid and double lines in Fig. 4.

3.2 Target Area Extraction Method Using Gesture Trajectories

In this section, the target area extraction method from the detected trajectories is shown. First, the basic target area extraction method, which extracts a specific area from the image same as object detection, is shown. Second, the following application methods are shown. The first is extracting frames for which gestures cannot be used, such as close-up photography; the second is extracting as a bounding box, which can also be used for tilt correction.

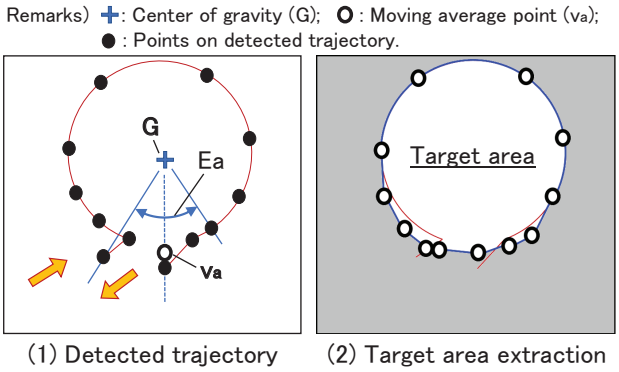


Figure 5: Target area extraction method using moving averages

3.2.1 Basic Target Area Extraction Method

The detected trajectory is accompanied by fluctuations of gesture motion. To compensate for these fluctuations, a moving average of the coordinates along the angle from the center of gravity is created. Similarly, doubles and omissions near the start and end of the detected trajectory are also compensated.

The x–y coordinate of the center of gravity G is obtained as an average of the x-coordinate and y-coordinate of the points on the detected trajectory, as shown in Fig. 5 (1). Subsequently, the coordinate of each point is transformed into a pair $\hat{p}_a = (\theta_a, r_a)$ of angle and distance from G. Let E_a be the angle section to calculate the moving average, and define the moving average v_a by Eq. (5).

$$v_a = (\theta_a, \bar{r}_a) \qquad (\theta_a \in E_a) \quad (5)$$

Here,

$$\bar{r}_a = (\sum r_a)/n$$

n is the number of coordinates contained in E_a, and it is five in Fig. 5 (1).

Consequently, $\bar{r}_a$ is the moving average of the distances between G and points on the detected trajectory along the angle. By connecting the coordinates of this moving average set $\{v_a\}$ along the angle, the target area is extracted, as shown in Fig. 5 (2).

3.2.2 Close-up Target Frame Extraction Method

Since the labels, such as those attached to fixed assets and products, are small, an enlarged image must be obtained by close-up photography in order to use OCR. In this case, the target area should not be specified. However, the close distance from the camera makes it difficult to indicate the target frame with the gesture. To solve this problem, a target frame extraction method that combines camera movement and gestures is proposed.

Figure 6 shows that the camera is held still to view target (1) and zoom-in (a) for a close-up of the label (2). Subsequently, zoom-out (b) is performed to indicate the target frame by gestures (3). Consequently, the target frame where the label was taken was between this zoom-in and zoom-out. This operation can be performed using the mobile camera by approaching and leaving.

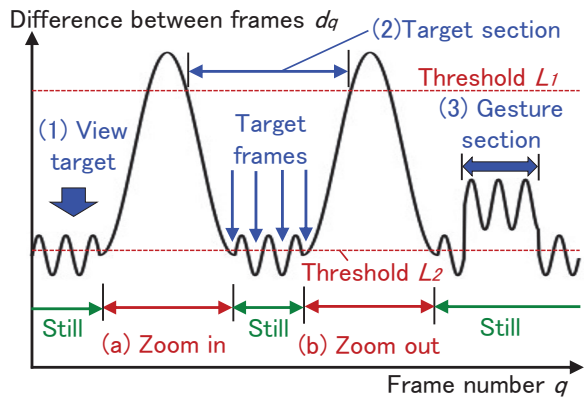


Figure 6: Target frame extraction method using differences between frames and gesture section

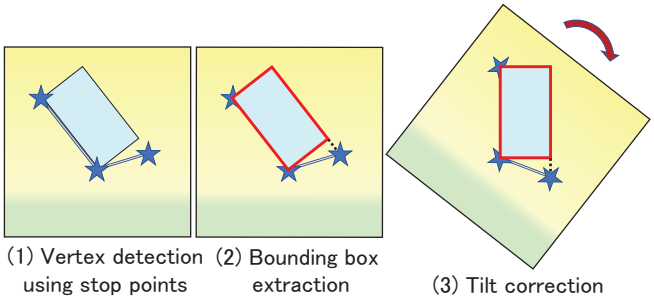


Figure 7: Extraction method as a bounding box with tilt correction

The vertical axis of Fig. 6 shows the variation of the difference d_q between frames along the frame number q in this operation. Since the field of view changes in the zoom-in and zoom-out sections, the difference between frames increases. Conversely, the difference is small in the other still section, but small fluctuations remain due to camera shakes for a wearable camera.

Set two threshold values L_1 and L_2 to extract the target frame based on the gesture section (3). L_1 is the threshold for detecting zoom-in and zoom-out; L_2 is the threshold for detecting still frames. The gesture section (3) is detected based on the stop points by the procedure shown in Fig. 4. Subsequently, the target section (2) is detected as the section between zoom-in and zoom-out, based on L_1. Finally, for the target section (2), the target frame is detected as the frame with the smallest difference d_q in each section where d_q is below L_2, as shown in Fig. 6.

3.2.3 Bounding Box Extraction Method

A method to extract the target area as a bounding box, similar to other object detection methods, is shown. In addition, the box's tilt can be corrected with this method. This tilt correction can be applied to rectangular target areas that may be placed at an angle, such as books on a bookshelf. Consequently, it is expected that the discrimination accuracy through template matching and others will be improved.

This method uses stop points; the gesture is sequentially stopped at three vertices, as shown in Fig. 7 (1). And, the bounding box can be estimated by these stop points. When correcting the tilt, the edge that will become the bottom edge is specified first to indicate the direction of the rotation.

Figure 7 (2) shows that the longer of the two edges (subsequently called long edge), which should be accurate according to the tilt, is selected. Moreover, the other edge is estimated as a line segment perpendicular to the long edge. The remaining edges are estimated as parallels of these edges. Hence, the bounding box is estimated with these edges.

If the number of stop areas exceeds three, the three areas are selected based on the number of points $\tilde{p}_q$ on the detected trajectory included in each stop area. When the box's tilt must be corrected, the image is rotated so that the long edge is vertical or horizontal, according to the above-mentioned specified order of the edges, as shown in Fig. 7 (3).

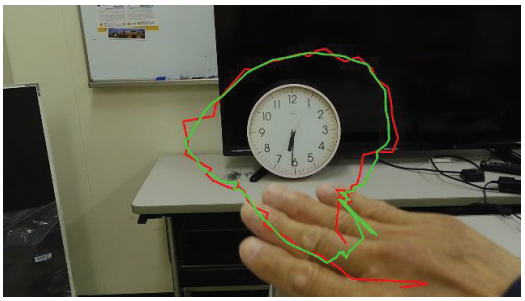


Figure 8: Indication accuracy of hand gestures

4 IMPLEMENTATION AND EXPERIMENTS

4.1 Implementation of Gesture Trajectory Detection Method

An experimental system was constructed to evaluate the proposed target area extraction method. The PC used has a CPU of i9-10850K 3.6 GHz, 64 GB memory, and a GPU of Geforce RTX 3090 with 24 GB memory. In addition, the Logitech Web camera C920n was directly connected to the laptop computer, and videos were shot at a resolution of $1,920 \times 1,080$ pixels and 30 fps. This system was implemented on Windows 10, Python Ver. 3.8.13 as the program and Pytorch Ver. 1.7.1 with CUDA Ver. 11.5 to use YOLO, OpenCV-Python Ver. 4.5.5.64 for image and video manipulation, and TensorBoard Ver. 2.8.0 to analyze the model training results. YOLOv5s, a highly efficient model of YOLO, was implemented by adding the necessary functions to the publicly available program [2]. Similarly, the publicly available "Egohand Dataset" [1] was used for the training data, by which the model to detect each of own and opponent's left and right hands is trained, as shown in Fig. 2. It comprised 4,800 data taken at 48 locations, of which 3,840 were used as training data and 960 as validation data.

As a preliminary experiment, this trained model was used to evaluate the extraction accuracy of the target area from the gesture trajectory of the right hand. Figure 8 shows the results, where the red line shows the median trajectory and the yellow–green line shows the detected trajectory. Note that

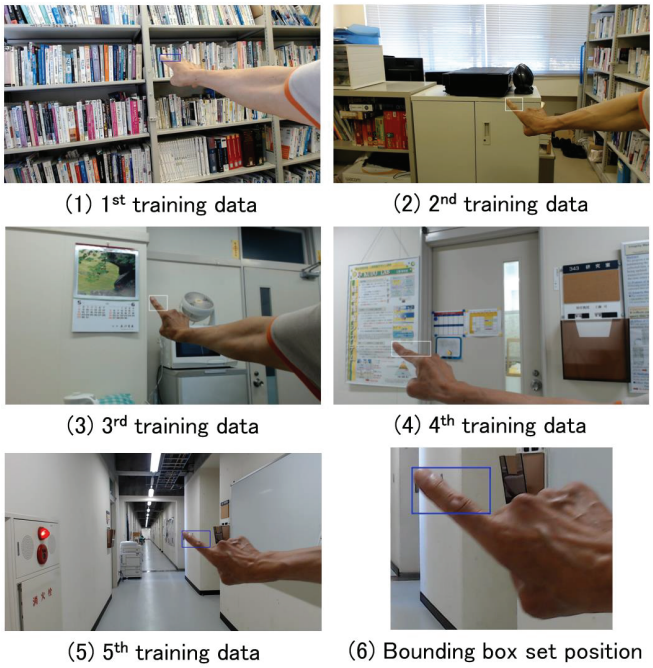


Figure 9: Training data of fingertip for transfer learning

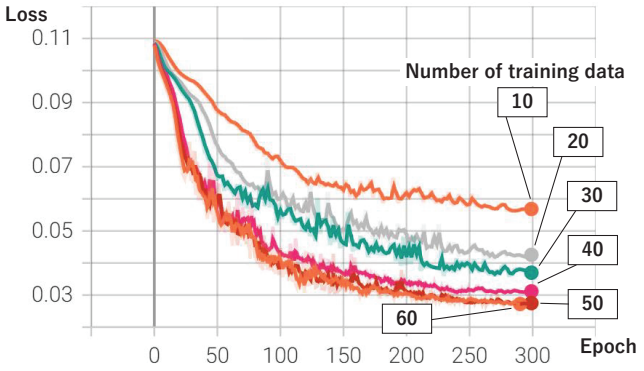


Figure 10: Changes in validation loss in transfer learning

the extraction of the target gesture section was omitted in this experiment.

Consequently, it was observed that the hand was too large to indicate the target area accurately and that the indication accuracy was insufficient.

4.2 Experiments on Improving Indication Accuracy Using Transfer Learning

For the issue mentioned in Sec. 4.1, the training efficiency of transfer learning utilizing this existing model was evaluated to efficiently use an arbitrary body part or instruction tool. A total of 75 images, 15 for each of the five different environments, were shot, as shown in Figs. 9 (1–5), and a bounding box label was created for each fingertip, as shown in Fig. 9 (6). This box includes the fingertip, and the lower right is the lowest position near the fingertip. In this experiment, the fingertip was pointed to the upper left. The data were divided into 60 training data and 15 validation data.

In YOLO's transfer learning, the number of layers to be

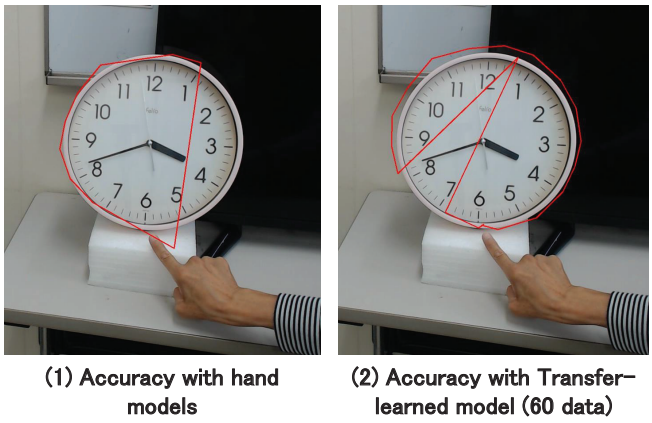


Figure 11: Detected trajectory of gesture using hand and transfer-learning model

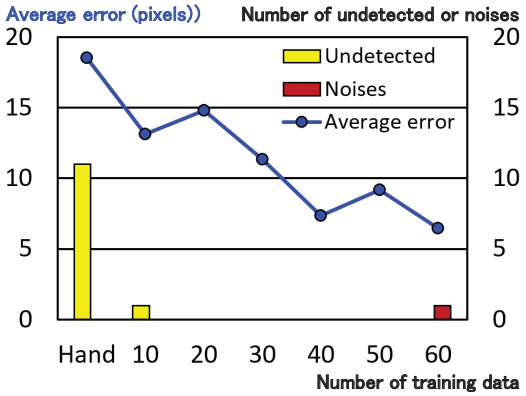


Figure 12: Accuracy improvement with an increase in transfer-learning data

fixed can be specified by the "freeze" parameter. In this experiment, it was set to 10, corresponding to the "backbone" of YOLO, where features were extracted. The number of training data was from 10 to 60 for every 10, and the number of validation data was fixed to 15. The number of training epochs was 300. Figure 10 shows the transition of the validation loss of the bounding box position obtained through TensorBoard. The numbers in the boxes indicate the number of training data. For 60 training data, the verification loss converged, so training was completed before 300 epochs.

Figure 10 shows that the accuracy improves as the training data increases, and the accuracy improvement converges at about 50 data.

Subsequently, 20 test data were prepared for the clock used in the preliminary experiment, as shown in Fig. 8, and evaluated the effectiveness of transfer learning. Figure 11 (1) shows the connecting result of the fingertip points detected by YOLO using the model before transfer learning, that is, the hand model; (2) shows the result using the model after transfer learning with 60 data.

In Fig. 11 (1), since some points are not detected, the right side is missing. In Fig. 11 (2), though there is one point that is far from the original position (a noise), this point can be removed in the median trajectory.

Figure 12 shows the number of undetected points and noises

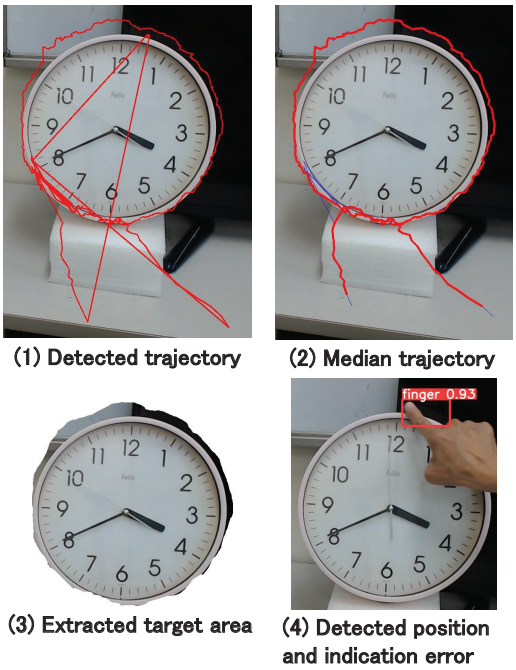
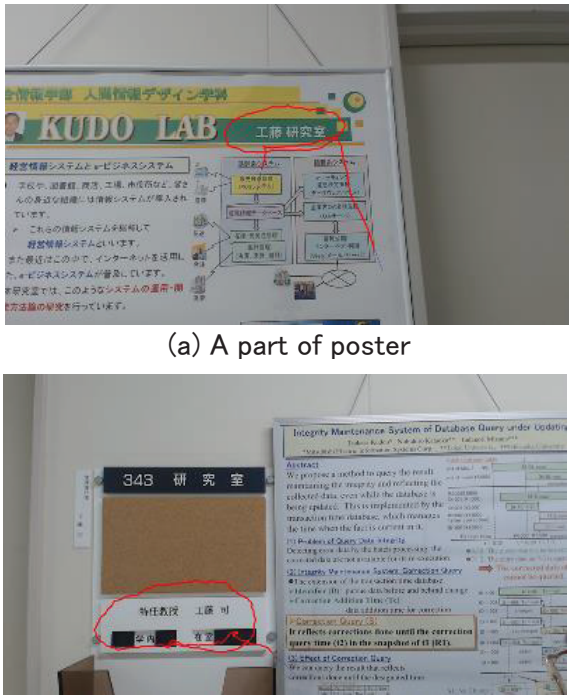


Figure 13: Accuracy evaluation of target area extraction using fingertip gestures



(a) A part of poster

(b) Laboratory nameplate

Figure 14: Evaluation target for basic target area extraction

and the average error of detected coordinates in pixels for each number of transfer-learning data. Note that "hand" indicates the case where the original hand model was used. Undetected points and noises were excluded from the calculation of the average error.

While more than half of the points are not detected in the "hand" model before transfer learning, this number is reduced to one point after transfer learning with 10 data. This indicates that transfer learning with even a small number of data (about 10) can improve detection accuracy. As in Fig. 10, the accuracy improved as the number of training data increased, but in this experiment, the improvement in accuracy almost converged at 40 data.

In this experiment, the target area was extracted using the basic target area extraction method mentioned in Sec. 3.2.1. The accuracy threshold V in Eq. (1) was set to 0.4; the number of points to calculate the median and the moving average was set to five. Note that the end points of the trajectory were omitted for the median trajectory, and the next point was calculated with three points. In addition, for the stop area detection, the threshold H in Eq. (4) was set to 1% of the narrower axis of the screen (108 pixels).

Figure 13 shows the results. (1) shows the trajectory of the points detected by YOLO with the highest accuracy for each frame; (2) shows the median trajectory; (3) shows the extracted target area based on the detected trajectory where the outside of the target area was set to white.

Compared Fig. 13 to Fig. 8, the accuracy of the target area is improved using the fingertip. In addition, the area untargeted for detection at the bottom of the clock could be removed using the detected trajectory. However, the outline of the upper part of the clock deviated from the actual outline. Thus, I investigated the detected points and observed that the indicated position of the fingertip deviated, as shown in Fig. 11 (4). This was caused by the viewpoint error between the human eye and the camera because a fixed camera was used.

5 APPLICATIONS AND EVALUATIONS OF PROPOSED METHOS

To evaluate the effectiveness of the proposed method in actual business, object recognition accuracies were evaluated in the three cases assumed in Sec. 3.2. In addition, the settings are similar to the experiments shown in Fig. 13.

5.1 Evaluation of Basic Target Area Extraction: Recognition of Bulletin Boards

As the evaluation target of the basic target area extraction method mentioned in Sec. 3.2.1, the poster and the nameplate of the laboratory shown in Figs. 14 (a) and (b), respectively, were used, and OCR on the extracted target area was performed. The red line in Fig. 14 shows the median trajectory of the gesture.

First, the same procedure shown in Fig. 13 was used to extract the target area shown in Fig. 15 (1). Subsequently, the area was binarized, as shown in (2), to eliminate the influence of color. In this case, the area of the poster in 14 (a) is inverted because the text is white. This binarization was performed using the $threshold$ method of OpenCV-Python, and the binarization threshold was set to 127 (the middle value). Finally, character recognition was performed using OCR from the binarized image shown in (2). OCR was implemented using Tesseract ver. 5.2.0 and OpenCV-Python's pyocr class. These gestures and procedures were performed twice.

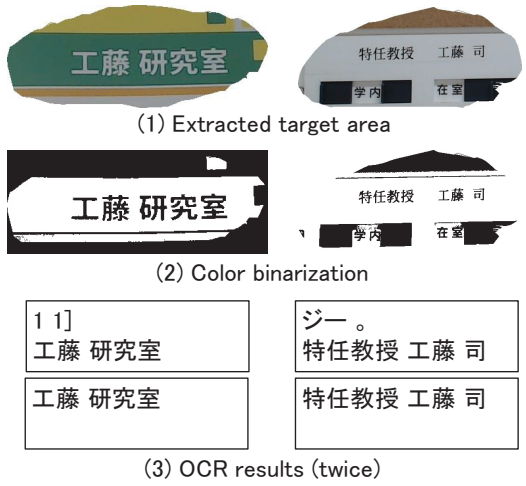


Figure 15: Procedure and results of object recognition by OCR

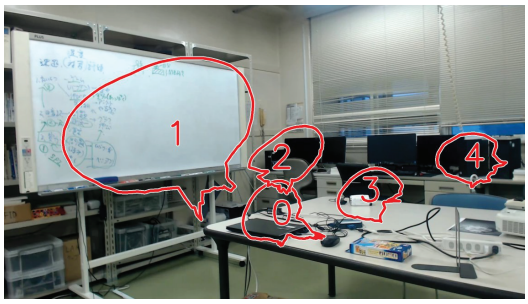


Figure 16: Indication results of fixed-asset locations

Hence, Fig. 15 (3) shows that although extra characters were included, the target characters could be recognized in all cases.

5.2 Target Frame Extraction: Fixed-Asset Location Management

To evaluate the target frame extraction method mentioned in Sec. 3.2.2, the method was applied to fixed-asset location management. The fixed-asset audit verifies the location of the assets and the fixed-asset label attached to each asset. In addition, the fixed-asset management database, which manages detailed information, such as fixed-asset names and administrators, can be retrieved with the fixed-asset number on the fixed-asset label. This experiment verified the possibility of information extraction from the video.

To record the location of the asset, a fixed camera was used, and the target area was indicated sequentially with the gestures, following the same procedure shown in Fig. 13. However, in this application, it was necessary to clarify the items' location, so the target area in Fig. 5 (2) was not extracted, but only the detected trajectory was drawn. In addition, the order in which each asset was indicated was described, in each trajectory.

Consequently, Fig. 16 shows that the position of each asset could be recorded only by gestures. Moreover, since this method does not require precise positions such as the outline of the targets, gesture indications were easy regardless of the type or view of the target.

Next, the recognition accuracy of fixed-asset labels was evaluated. Since the fixed-asset label needs to be taken by close-up photography, the target frames were extracted following the method mentioned in Sec. 3.2.2, and OCR was performed on the frames. Table 1 shows that a PC, a smart glass (Glass), and a tablet were used. Moreover, as shown in Fig. 17 (2), the fixed-asset number (BB170090) and registration date (20171109) in the second line were recognized.

In this experiment, the target object was taken along the procedure as shown in Fig. 6. That is, it was taken at a wide angle (1); then, the camera was moved closer to take the fixed asset (2); finally, the camera was moved away again, and the frame was indicated by gesture (3). The difference between frames (hereinafter, difference) in Fig. 6 was calculated using the OpenCV-Python methods. The frames were converted to monochrome images using the $cvtColor$ method, and each difference image between adjacent frames using the $absdiff$ method. Subsequently, the difference was calculated by Eq. (6).

$$d_i = \sum_{j=0}^{255} n_{ij} * j/N \tag{6}$$

Here, i is the frame number; j is the luminance; n_{ij} is the number of pixels with luminance j in the frame number i; N is the number of pixels in each frame. The variation d_i is divided by N for normalization.

Thresholds L_1 and L_2 are set to 10 and 5, respectively. From the gesture section (3) to backward, the section where ten consecutive frames exceed L_1 is detected as the zoom-out section. Similarly, with this zoom-out section as the starting point, a section where 15 consecutive frames were below L_1 was detected as the target section (2). Figure 18 shows the transition of d_i along the frame number and the detected results.

The frames with the minimum d_i for each section below L_2 in the target section were extracted to perform OCR to recognize the fixed-asset numbers and registration dates. The implementation of OCR was similar to Sec. 5.1. However, binarization was also omitted, since the target area extraction was omitted. Fixed-asset numbers began with "BB" in this case. Hence, strings matching the following regular expression were extracted for the OCR results.

$$\backslash nBB\backslash S\{6\}\backslash s * \backslash S\{8\}\backslash n$$

That is, the target line is surrounded by newline characters and comprises eight characters, starting with BB, one or more spaces, and eight characters. Additionally, in order to correct OCR errors, at most one space was allowed in the strings of each fixed-asset number ($BB\backslash S\{6\}$) and registration date ($\backslash S\{8\}$), and it was removed from the OCR results.

Table 1 shows the recognition results under this condition. In the "Result" column, "◯" indicates the correct result, and "NG" indicates the incorrect result. The "Target" column shows the number of frames with the target minimum difference in the target section, and the "Ratio" column shows the ratio of the frames that satisfy the above conditions.

Table 1: Target object in experiment

Target	Asset num.	Number	Result	Reg. date	Number	Result	Target	Ratio
PC	BB210104	9	◯	20211214	9	◯	20	45%
Glass	BB170090	8	◯	20171109	8	◯	16	50%
Tablet	BB15Z061	15	◯	20151022	18	◯	20	90%
	BB152061	3	NG					

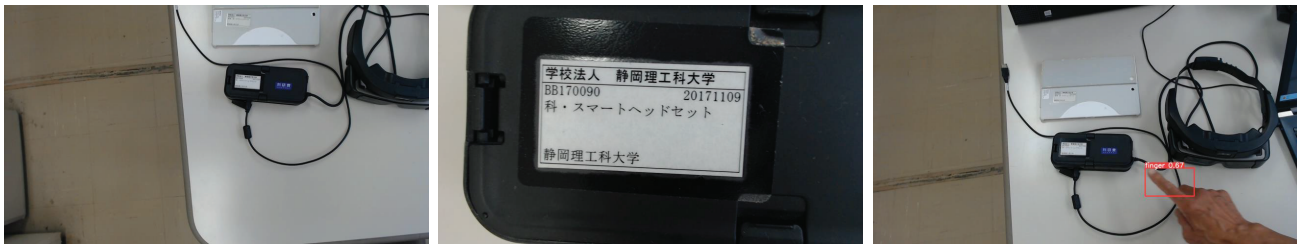


Figure 17: Target object-taking procedure to indicate target frame

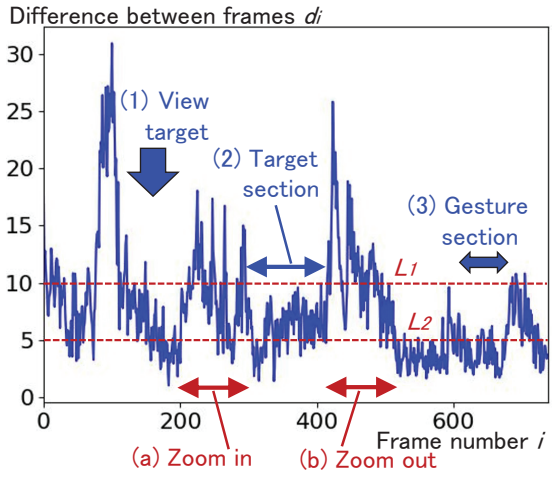


Figure 18: Transition of difference between frames for Glass

Consequently, while "Z" was misrecognized as "2" in 3 out of 18 fixed-asset numbers of the tablet, the others were correctly recognized. However, the percentage of frames satisfying the above conditions differed depending on the case.

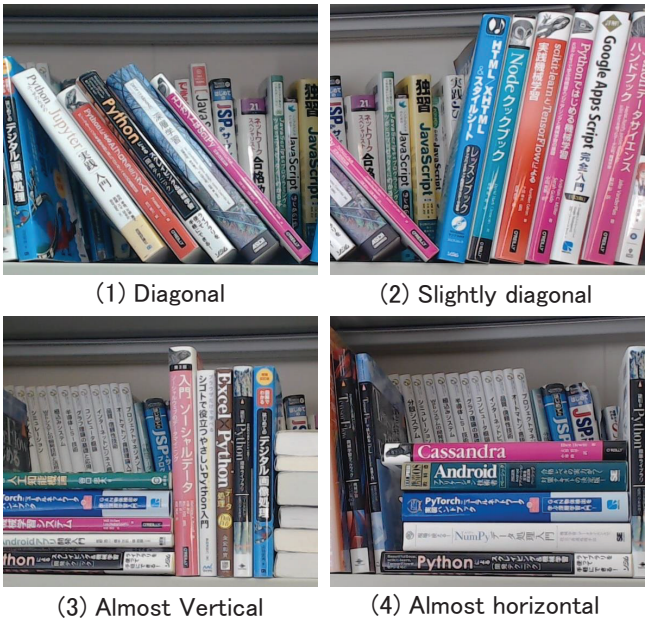


Figure 19: Evaluation target for book recognition with tilt correction

5.3 Bounding Box Extraction and Tilt Correction: Book Recognition

In order to evaluate the bounding box extraction method with tilt correction, as shown in Sec. 3.2.3, 20 books arranged at four different tilts were used, as shown in Fig. 19. Moreover, the effect of tilt correction on object recognition using template matching was evaluated. The $matchTemplate$ function of OpenCV-Python was used for template matching, and the $normalized\ cross-correlation\ matching\ method$ was used for the matching method.

First, to create a template image, each target book was manually cut out from the image in Fig. 19 so that it was arranged vertically, as shown in Fig. 20 (1). Next, as shown in Figs. 20 (2) and (3), the target book was extracted as a bounding box with and without tilt correction, as mentioned in Sec. 3.2.3, and placed in a white image of 700×700 pixels, respectively. The green rectangles in Figs. 20 (2) and (3) are the results of template matching shown below.

For each image, template matching was performed between the template image shown in Fig. 20 (1) and each of the images shown in (2) and (3) to obtain the matching position and accuracy. Here, when the tilt was corrected, as shown in Fig. 20 (2), the template image was generally matched to the correct position. Similarly, other images were generally matched to the correct position when the tilt was corrected. However,

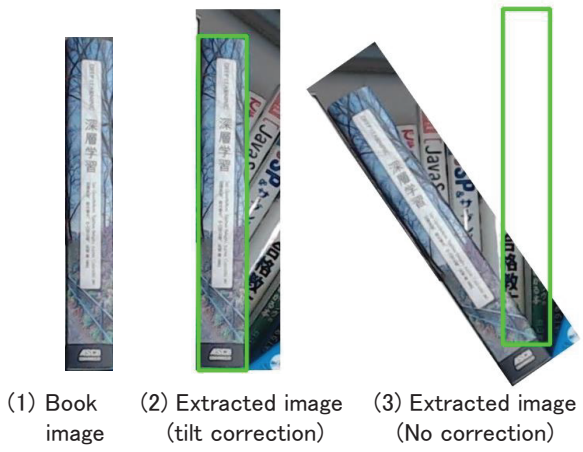


Figure 20: Book template-matching results

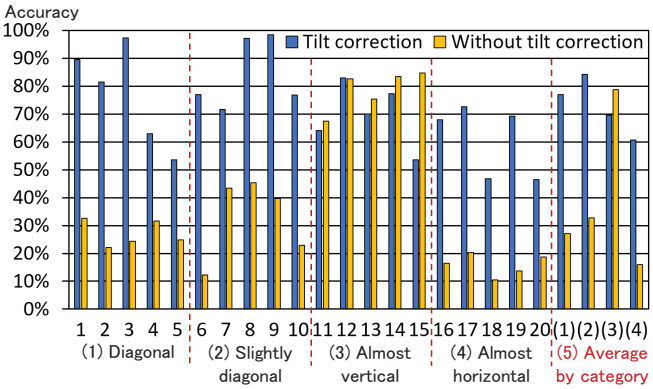


Figure 21: Evaluation of template-matching accuracy improvement through tilt correction

as shown in Fig. 20 (3), there was an error in the matching position of the template images in the case of this book.

Figure 21 shows the evaluation results of the matching accuracy for all books. The numbers (1) to (4) on the horizontal axis correspond to the numbers of the arrangement types in Fig. 19; (5) shows each average of the cases of (1) to (4). The accuracy of the tilt correction case was advanced for the tilted books, as shown in (1), (2), and (4) in Fig. 21. However, the accuracy of the no tilt correction case was advanced for the books placed almost vertically, as shown in (3). Furthermore, the accuracy was different for each book.

To analyze this phenomenon, template-matching positions were verified for typical cases. The maximum accuracy of the tilt correction case (horizontal axis number: 9) is shown in Fig. 22 (1); the minimum accuracy (also 20) is shown in (2), and the case where the tilt correction result deteriorated compared to the no tilt correction result (also 15) is shown in (3). In each case, the left image was the template image, and the right image was the extracted image. (3) was the only case of no tilt correction. In (1), the extracted image included the entire area of the target book, while in (2), the left side of the book was missing. In (3), the bounding box (extracted target area) was tilted due to an error in the gesture indication, although the books were placed vertically. Consequently, the accuracy deteriorated with tilt correction.

Figure 22: Influence analysis of gesture indication error on template-matching accuracy

The results show that the target area extraction with tilt correction is effective for template matching accuracy (recognition accuracy) for the target placed diagonally. Conversely, the gesture indication error caused an accuracy deterioration, especially when the target was placed vertically.

6 DISCUSSION

In this study, a method to extract the target area by gesture trajectory detection using YOLO was proposed, and its effectiveness was evaluated through several experiments. First, since only a specific part of the body, such as the fingertip, was used for gestures, it was easy to prepare the training data. Second, practical accuracy was achieved with 60 training data through transfer learning. This shows that this method can be easily applied to any body part and indication device according to the purpose.

However, the gesture trajectory detection was performed on videos and noises, and extra sections were included, as shown in Figs. 2 and 13 (1). Hence, this study showed that the noises were removed using medians, and the target sections were extracted using stop points. Consequently, as shown in Fig. 13 (3), it was possible to extract the target area without these influences. Furthermore, experiments showed that the target area could be extracted without being affected by noise, even in actual use cases.

Though object detection using deep learning can be performed automatically, this method requires the action of gestures. Conversely, this method can extract the target areas more flexibly, as shown by the actual application cases in Secs. 5.1 and 5.3. Any part of an image with no special features can be extracted, as shown in Figs. 14 and 15 (1). Furthermore, the target area can be extracted as a bounding box, and its tilt can be corrected when necessary, as shown in Fig. 20. In addition, as shown in Table 1, Figs. 15 (3) and 21, by extracting the target area in this method, object recognition using simpler recognition methods, such as OCR or template matching, is possible.

Though the experiments in this study were conducted in a laboratory, they targeted the use case of the factory operation

shown in Fig. 1. First, the entry of workers to the inventory storage area, also the target container, can be identified by the recognition of the bulletin boards shown in Sec. 5.1. Second, since there are thousands of containers in the factory, the picking process requires instructions not only for the containers to be picked but also for the optimal route. This can be indicated by the method shown in Sec. 5.2 as shown in Fig. 16. Similarly, images of containers or container labels can be extracted and stored for confirmation during picking as shown in Fig. 17. Third, the images of the parts in the factory are stored in a database. Thus, parts in containers can be efficiently discriminated by using the tilt-corrected bounding box and template matching described in Sec. 5.3.

With the spread of IoT and mobile cameras, long tasks and operations are recorded through videos, and it is important to record the data to specify the target frame and area. Section 5.2 showed the method for this, by which the area and the target frame can be extracted. In addition, the frame can be specified even for close-up photography using gestures and differences between frames. For example, when inspecting a building while constantly taking video with the smart glass shown in Fig. 1 (1), the location of abnormalities can be recorded by gestures without operating the camera, as shown in Figs. 16 and 17 (2).

Conversely, when using a fixed camera, as shown in Figs. 13 (4) and 22, errors of gesture indication occur due to a viewpoint error between a human and a camera. For this problem, the use of the above-mentioned smart glass effectively performs gestures while checking the position displayed on the screen is considered. However, there is a challenge for such a mobile camera, since it moves, as shown in Fig. 18. To construct the trajectory of the target area shown in Fig. 5 (2), it is necessary to correct the angle, position, and magnification between frames. Therefore, applying this method to mobile cameras is the next challenge.

7 CONCLUSIONS

When performing object recognition with mobile cameras such as smart glasses, it is often necessary to perform object detection beforehand. In object detection, various studies using deep learning have been actively conducted. However, since it is necessary to prepare training data for each target, it is difficult to prepare training data when there are various targets.

To solve this problem, a method for extracting the target area from the gesture trajectory detected using YOLO is proposed. Furthermore, through experiments, it was shown that practical accuracy can be achieved from less training data by transfer learning and that targets can be extracted flexibly according to purposes.

Future studies will focus on improvements in the accuracy of gesture indication using mobile cameras.

REFERENCES

[1] S. Bambach, S. Lee, D. Crandall, and C. Yu, "Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions," IEEE Int. Conf. Computer Vision (ICCV), pp. 1949-1957 (2015), https://public.roboflow.com/object-detection/hands (referred May 19, 2022).

[2] G. Jpcher, et.al. "YOLOv5," https://github.com/ultralytics/yolov5 (referred May 17, 2022).

[3] T. Kawanaka, and T. Kudo, "Inventory satisfaction discrimination method utilizing images and deep learning," Procedia Computer Science, Vol. 126, pp. 937–946 (2018).

[4] T. Kudo, "A Proposal for Article Management Method Using Wearable Camera," Procedia Computer Science, Vol. 176, pp. 1338–1347 (2020).

[5] T. Kudo, "Moving Object Detection Method for Moving Cameras Using Frames Subtraction Corrected by Optical Flow," Int. J. Informatics Society, Vol. 13, No. 2, pp. 79–91 (2021).

[6] T. Kudo, "Contour Detection Method by CycleGAN Using CG Images as Ground Truth," Int. J. Informatics Society, Vol. 14, No. 1, pp. 3–12 (2022).

[7] T. Kudo, "Application Method of Deep Learning Model Trained with CG Images to Real Images," Procedia Computer Science, Vol. 192, pp. 1484–1493 (2021).

[8] T. Y. Lin, et.al., "Microsoft coco: Common objects in context," European Conf. Computer Vision, pp. 740–755 (2014).

[9] T.Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 2117–2125 (2017).

[10] T.Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection." Proc. IEEE Int. Conf. Computer Vision, pp. 2980–2988 (2017).

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," European Conf. Computer Vision, pp. 21–37 (2016).

[12] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikáinen, "Deep learning for generic object detection: A survey," Int. J. Computer Vision, Vol. 128, No. 2, pp. 261–318 (2020).

[13] A. Mujahid, M. J. Awan, A. Yasin, M. A. Mohammed, R. Damaševičius, R. Maskeliūnas, and K. H. Abdulkareem, "Real-time hand gesture recognition based on deep learning YOLOv3 model," Applied Sciences, Vol. 11, No. 9, 4164 (2021).

[14] M. Mukhiddinov, and J. Cho, "Smart glass system using deep learning for the blind and visually impaired," Electronics, Vol. 10, No. 22, p. 2756 (2021).

[15] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," ACM Computing Surveys (CSUR), Vol. 54, No. 8, pp. 1–37 (2021).

[16] M. Oudah, A. Al-Naji, and J. Chahl, "Hand gesture recognition based on computer vision: a review of techniques, " J. Imaging, Vol. 6, No. 8, 73 (2020).

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," Proc. IEEE Conf. Computer Vision and Pattern

Recognition, pp. 779–788 (2016).
[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," Advances in Neural Information Processing Systems, pp. 91–99 (2015).
[19] Y. Shi, Y. Li, X. Fu, K. Miao, and Q. Miao, "Review of dynamic gesture recognition," Virtual Reality & Intelligent Hardwar, Vol. 3, No. 3, pp. 183–206 (2021).
[20] P. Viola, and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proc. 2001 IEEE Computer Society Conf. Computer Vision and Pattern Recognition, Vol. 1, pp. 511–518 (2001).
[21] Z. Wang, L. Jin, S. Wang, and H. Xu, "Apple stem/calyx real-time recognition using YOLO-v5 algorithm for fruit automatic loading system," Postharvest Biology and Technology, Vol. 185, No. 111808 (2022).
[22] J. Wolfartsberger, J. Zenisek, and N. Wild, "Data-driven maintenance: combining predictive maintenance and mixed reality-supported remote assistance," Procedia Manufacturing, Vol. 45, pp. 307–312 (2020).
[23] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai, and H. Ling, "M2det: A single-shot object detector based on multi-level feature pyramid network," Proc. AAAI Conf. Artificial Intelligence, Vol. 33, pp. 9259–9266 (2019).

(Received: October 29, 2022)
(Accepted: December 18, 2022)

Tsukasa Kudo received his BSc and ME from Hokkaido University in 1978 and 1980, and Dr. Eng. from Shizuoka University in 2008. In 1980, he joined Mitsubishi Electric Corp. He was a researcher of parallel computer architecture and engineer of business information systems. Since 2010, he has been a professor at Shizuoka Institute of Science and Technology. His current research interests include deep learning and database applications. He is a member of IEIEC and IPSJ.