Learning-Based Hand Motion Capture and Understanding in Assembly Process

Liang Liu, Yong Liu, *Member, IEEE*, and Jiangning Zhang

***Abstract*—Manual assembly is still an essential part in modern manufacturing. Understanding the actual state of the assembly process can not only improve quality control of products, but also collect comprehensive data for production planning and proficiency assessments. Addressing the rising complexity led by the uncertainty in manual assembly, this paper presents an efficient approach to automatically capture and analyze hand operations in the assembly process. In this paper, a detection-based tracking method is introduced to capture trajectories of hand movement from the camera installed in each workstation. Then, the actions in hand trajectories are identified with a novel temporal action localization model. The experimental results have proved that our method reached the application level with high accuracy and a low computational cost. The proposed system is lightweight enough to be quickly set up on an embedded computing device for real-time online inference and on a cloud server for offline analysis as well.**

***Index Terms*—Assembly line monitor, hand motion capture, temporal action localization.**

I. Introduction

As GREAT progress has been achieved in Industry 4.0, most routine industry operations have been performed automatically by machines [1], [2]. However, for industries such as computers, communications, and consumer electronics, the short sales cycle and diversified product models make them unsuitable for fully automatic production. Manual and semi-automatic assemblies are still considered indispensable for manufacturing [3].

Despite the flexibility and easy implementation of manual assembly, quality control for manual assembly is much more difficult than the machine. The quality of the product will be greatly improved if the factory can monitor and manage the manual assembly process. In reality, almost all monitoring systems for assembly lines are nonintelligent, which only use surveillance cameras to record operations. These systems can replay for special situations, while it is impractical to monitor and manage the entire assembly process as it will take a lot of time and labor to view the whole video manually.

In addition, with the progress of optimization research and sensor technology, some research studies for optimizing the assembly process have been proposed, which can take a more scientific production planning [4] and achieve a more careful proficiency assessment for operators [5], [6]. All these methods can be applied to real factories only if operational data can be collected during the daily assembly process. Traditional factories obtain statistics through random manual inspections or simulation tests so that the data are limited. Due to the uncertainty of manual operation, the operating habits vary from person to person, and there are unpredictable interruptions or exceptions in manual assembly lines, which are hard to capture by traditional methods.

Overall, it is important to automatically capture and understand operations in the manual assembly process. Thus, the assembly can be monitored and managed to improve quality control of products. Furthermore, those more accurate and comprehensive process data can also be collected to optimize the assembly process.

There are three main challenges in real manual assembly lines.

1) It is hard to capture and identify the specific content of operations over a long time range as the shape and speed of the hand vary frequently, and different actions are also similar in appearance.
2) The solutions to capturing and understanding the assembly process should not interfere with the operation. Thus, additional symbolic gestures that will increase the overall time are undesirable. Similarly, it is not a good solution to place markers on the assembly line or the body of operators.
3) For manufacturers, it is critical to maintain the overall solution with a reasonable cost, so that it can be applied to dozens of workstations on assembly lines.

In this paper, a learning-based framework for hand motion capture and understanding in assembly lines is proposed. As illustrated in Fig. 1, this work focuses on real-time tracking of hand locations in assembly and identifying operational actions at the frame level. The framework can be summarized into two closely related main parts: hand motion capture and hand action understanding. Our method extracts the movement of hands

Manuscript received August 4, 2018; revised October 12, 2018; accepted November 8, 2018. Date of publication December 7, 2018; date of current version July 31, 2019.This work was supported in part by the National Key R&D Program of China under Grant 2017YFB1302003 and in part by the National Natural Science Foundation of China under Grant U1509210. *(Corresponding author: Yong Liu.)*

L. Liu and J. Zhang are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: leonliuz@zju.edu.cn; zhangzjn@qq.com).

Y. Liu is with the State Key Laboratory of Industrial Control Technology and Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: yongliu@iipc.zju.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2018.2884206

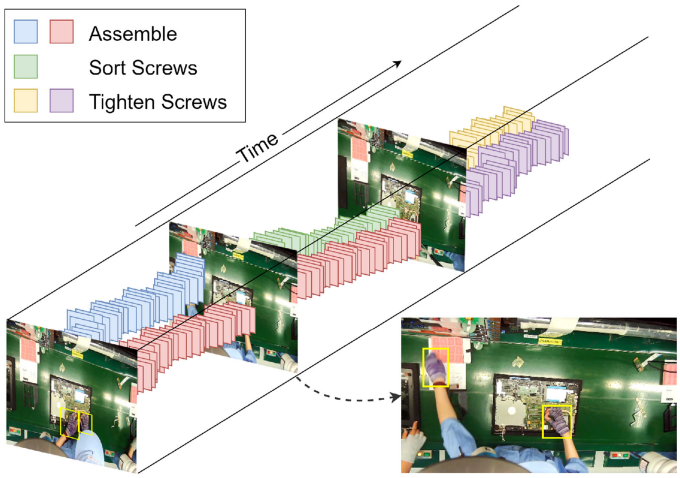


Fig. 1. Demonstration of our framework on hand motion capturing and understanding. The motion trajectories of hand in each video frame are represented by bounding boxes with recognition the action of hands. Different colors indicate different action instances.

with a novel tracking method, in which a network detects hands in the keyframes, while the tracking submodule predicts hand locations in each frame, and the results of detection and tracking are fused by a special matching algorithm. Meanwhile, a temporal action localization model is designed for simultaneously categorizing and localizing the time boundaries of atomic actions in the video segments.

Our method captures the hand position and action content of operators in real time. The hand action sequences can be concatenated to describe the operation by finite-state machines [7] in a complete operating cycle. Contrasting the actual operation with the standard operating procedure, supervisors can easily access production process information, such as if there are any operational omissions, which workstation cause drift, the valid operation time of operators in a cycle. Our method provides an effective way to obtain human data in the production process. The contributions of this paper are summarized as follows.

1) We designed a flexible and effective multiple-object tracking framework, which can capture the challenging movement of hands in the assembly process. Furthermore, the temporal action localization method we proposed can accurately detect the atomic action instances at a frame level from untrimmed video streams.
2) Our markerless framework has no influence on productivity and can detect the semantic content of the operation in the assembly videos for a long time. The structured action results in temporal and spatial can be further utilized to improve quality control or assembly line optimization.
3) The extra equipment costs of our approach are fairly low and can be set up quickly in productive factories that only require a webcam installed on each workstation. Moreover, the method is lightweight enough to be deployed on an embedded device for real-time processing.
4) The entire framework based on deep learning is learnable; thus, it can be easily extended to the new assembly lines, which have different scopes of recognition. Our approach has already been deployed on several production lines in real factories.

The remainder of this paper is organized as follows. Section II reviews related research works. Section III gives a brief instruction of data acquisition. In Section IV, the framework and details of the proposed method for hand motion capture and hand action understanding are introduced. In Section V, details of data annotation and experiments are illustrated, and the effectiveness of the proposed method is evaluated. Finally, Section VI concludes this paper.

II. Related Work

A. Motion Capture Systems

Motion capture, which has shown significant advances in recent years, is the process of recording the movement of objects or people [8], [9]. The motion capture techniques can be implemented with special sensors, such as inertial measurement units [10], [11] or multiple hardware synchronized infrared cameras [12]. However, these methods that utilize expensive devices and mainly depend on the markers equipped on the objects or environments are not suitable for manual assembly.

Recently, many markerless motion capture approaches have been developed that would not interfere with the captured object. There are many works on human or hand motion capture with low-cost depth cameras, such as Microsoft Kinect [13], [14], Intel RealSense [15], and Leap Motion sensor [16]. However, limited to the range and accuracy of hardware, they can only reconstruct coarse skeletal motion [17]. Besides, these devices are still quite expensive to be densely deployed.

With the development of deep neural networks, it has become possible for low-cost methods to capture object locations with a few ordinary cameras or even one monocular camera [17], [18]. However, even the complex methods, which can make accurate predictions in natural scenes with a lot of computational costs, are hard to handle the common problems in assembly scenes such as motion blur and occlusion in the assembly video [19], [20]. Thus, the method, which is lightweight to be deployed and is able to adapt the challenging movements, is desired in assembly.

B. Action Understanding Methods

Temporal action localization is one of the classical problems in video action understanding, which focuses on categorizing and localizing the time boundaries of action instances in untrimmed videos. Following the progress made in the action recognition task [21], [22], recent works based on two-stream networks [23] can achieve acceptable results [24], [25], but these works suffer from the heavy computation of dense optical flows; thus, they are not suitable for real-time services. Recently, several approaches based on three-dimensional (3-D) convolutional networks have been proposed. One of the most representative works is a method named CDC [26], which can output frame-level predictions of action instances by upsampling in the time dimension. However, it will lead to inconsistent

Fig. 2. Environment of data acquisition and the location of cameras.

categories between adjacent frames; thus, it requires additional postprocessing to smooth the predictions.

Moreover, most of the above methods focus on sports activities or daily activities, on which videos were collected from various sources, such as movies, YouTube, and television channels [27], [28], where action instances in different categories have obvious differences in appearance and with a long duration. There is a large gap in the performance to understand subtle actions in manual assembly scenarios, where the appearance of the foreground and background is similar.

III. Data Acquisition

In our framework, all video data came from the manual assembly lines, which produce various models of laptops in a real factory. In order to reduce the equipment costs and the impact on operators, only a web camera is installed at each workstation. The camera installation location and the data acquisition environment are shown in Fig. 2. There are no other wearable devices or sophisticated sensors.

All video data are collected from a total of 30 workstations distributed in three different assembly lines. The total video length reached 104 h. All the videos are collected at a resolution of 1920×1080, 30 frames/s. We employed experienced workers to annotate several types of data to train and validate our learning models. The specific labeling method and details will be described in Section V-A.

IV. Proposed Method

In this section, a markerless hand motion capture and understanding framework is presented. The flowchart of this framework is shown in Fig. 3. Given the video stream captured from the camera equipment in a workstation, the general analysis procedures are summarized as follows.

1) *Hand motion extraction:* Hands in keyframes will be detected by detection networks, and a tracking module is used to associate and interpolate the hand bounding boxes in consequent frames as hand trajectory extraction.
2) *Trajectory cropping:* Each trajectory of hand is trimmed and sampled into fixed-length video clips. Followed by cropping each hand area from the video clip, each hand video clip is converted into video segments with a fixed size.
3) *Temporal action localization:* Locating and classifying all hand action instances in the video segments by the temporal action localization networks.
4) *Action instance reconstruction:* Combining action instances in the video segments and removing irrelevant action instances to form a temporal and spatial description of the hand movements for the complete video.

We present the details of the proposed method as follows.

A. Hand Motion Extraction

Since the information in the captured high-resolution video is redundant, it is unfocused and computationally expensive to process the original video directly [29]. Previous works have shown that as a key part of the human visual system, visual attention mechanisms can filter irrelevant visual stimuli in order to focus more on the important parts [30]. In the manual assembly scenario, the region of hands contains most of the information in manual assembly lines that we need to focus. But it is quite a challenging task to track the nonrigidity hands with irregular movement. Thus, we introduce a detection-based "tracking-with-matching" framework for real-time hand motion extraction.

1) Hand Detection: The field of object detection has made rapid progress in recent years. Most methods can be divided into one-stage detection [31], [32] and two-stage detection [19], [33]. The former has been shown, which are more lightweight and can achieve similar performance. In our approach, we modify the original SSD networks [31], the classical one-stage detection model, with MobileNets [34] as backbone networks for feature extraction.

In short, the SSD meta-architecture has a collection of anchors overlaid on the different layers with different spatial locations, scales, and aspect ratios. The SSD detector appends convolutional detection layers to the different layers of the truncated backbone network, which allows the detections predicting at multiple scales. These extra layers are trained to make two predictions for each anchor, a discrete class prediction and a continuous position offset, by which the anchor needs to be shifted to fit the ground-truth bounding box.

We use MobileNets as backbone networks for feature extraction and adopt the original SSD implement (SSD-VGG [31]) that appends six extra layers for detection. The main reason that makes MobileNets speed up is the depthwise separable convolution (DSC) layers. The general idea behind DSC is to split convolution into a depthwise convolution and a 1×1 pointwise convolution that reduce the number of parameters and increase the speed. Consequently, we replace the following appended convolution with DSC.

It requires matching anchors with ground-truth instances when training the detection networks. For each anchor, we regard all ground-truth boxes with Jaccard overlap higher than a threshold (0.5) as matched, that is different with most detection frameworks [19], [33], which only match the anchor with maximum overlap. By increasing the abundance of samples, this modification will improve both the stability and the convergence speed of training with a minor effect on the final accuracy.

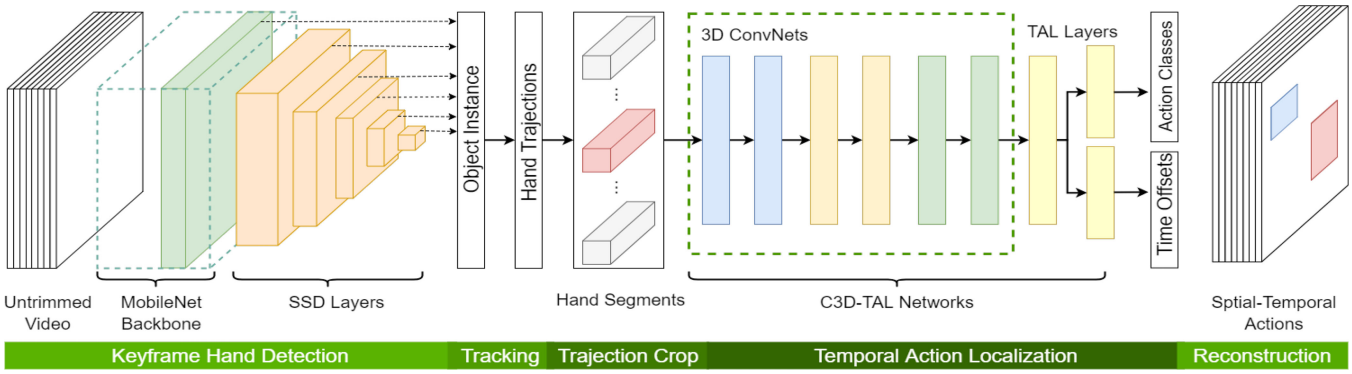


Fig. 3. Flowchart of our proposed method.

The rest of the details in our implementation are same as the standard SSD detector [31].

2) Hand Tracking: The detection model predicts hand regions in the static keyframes. In order to represent the movement of hands, we need to associate hand bounding boxes in consecutive frames. Due to the frequent occlusion and the violent movements of hands, both state-of-the-art single-object tracking algorithms (e.g., ECO [35] and KCF [36]) and multiple-object tracking algorithms (e.g., NOMT [37]) cannot perform as expected.

In our approach, we propose a simple but effective detection-based tracking framework that can be combined with the aforementioned state-of-the-art trackers, as well as weak trackers, such as Kalman filters and even linear interpolation predictions. We associate the detection results of keyframes with the frame-wise tracking results by linear assignment and interpolate the object locations in each frame with the prediction of the trackers. In particular, our framework introduces an efficient and effective state machine to deal with object occlusion and motion blur, which are the most challenging difficulties in hand motion capture.

Algorithm 1 summarizes the proposed framework. A group of trackers $\mathbb{T} = \{T\}$ is created based on the detection results of the first frame, and each tracker T has an initial state T^{state} as "*Active*" and a counter T^{uc} for recording the number of frames that the tracker does not match with any detections.

For a new frame I^t, new locations $\mathbb{L}$ and the prediction confidence score $\mathbb{S}$ of tracked objects are predicted by the active trackers $\{T_i \mid T_i^{\text{state}} = \text{Active}\}$. If the tracker is lost due to movement, obstruction, or other reason, the confidence score will be lower than the preset threshold δ. We set the state of these trackers to "*Sleep*" and keep the same location as the last frame if it exists. On the other hand, instead of frame-by-frame detection, the model predicts the hand locations $\mathbb{B}^t$ only when the current frame is a keyframe in order to reduce the total computational overhead.

The detected hand locations and tracked hand locations for the current frame are matched by the Hungarian algorithm with Jaccard overlap cost that results in a set of matching pairs $\mathbb{M}$, the unmatched detections $\mathbb{U}_{\text{Det}}$, and the unmatched trackers $\mathbb{U}_{\text{Tck}}$. We consider all unmatched detections to be new objects, and the corresponding new trackers are generated. Meanwhile, we temporarily assume that all unmatched tracker objects are missing detection due to motion blurring or occlusion, which might be detected in the next few frames. Therefore, we set the state T^{state} of the unmatched trackers to "*Wait*." Finally, the predicted locations of matched trackers are updated by the confidence-weighted average of detections, and tracking locations and the state T^{state} and the counter T^{uc} of matched trackers are reset.

Algorithm 1: Detection-based Tracking Framework.

Input: The current frame I^t, a set of trackers $\mathbb{T}$ created from the past frames, confidence threshold δ for tracking, number of maximum unmatched frames N and the optional object detection results $\mathbb{B}$.
Output: Alive trackers $\mathbb{T}$, Dead trackers $\hat{\mathbb{T}}$

```
// Get locations L = {l} and confidence score S = {s}
for T_i in T do
    l_i^t, s_i^t ← PREDICT(I^t, T_i) ;
    if s_i^t < δ then
        l_i^t ← l_i^{t-1} ;
        T_i^state ← Sleep ;
if B^t is given then
    // Matching tracking and detection.
    M, U_Det, U_Trk ← MATCH(L^t, B^t) ;
    // Create new trackers for unmatched detections.
    forall U_Det do
        Add a new tracker to the list of trackers ;
    // Try to re-track later for unmatched trackers.
    for T_i in U_Trk do
        T_i^state ← Wait ;
    // Update matched trackers with detections.
    for < l_i^t, b_j^t > pair in M do
        T_i ← UPDATE(I^t, l_i^t, b_j^t) ;
        T_i^state ← Active ;
// Remove trackers that unmatched in the last N frames.
for T_i in T do
    if T_i^uc > N then
        Pop T_i from T to T̂;
return T, T̂ ;
```

After matching, dead trackers that not matched in the last N frames will be popped from $\mathbb{T}$ to another set $\hat{\mathbb{T}}$. The trajectories of dead trackers are used for further analysis. We discard trajectories that are too short and trim the last N positions in every trajectory as they are unmatched with detections.

Our framework can be combined with many kinds of trackers. There are some differences in detail to balance the performance of tracker and detector. For high-performance trackers, such as KCF [36] and ECO [35], unmatching occurs more often when there are some mistakes in the detection. So, the trackers with state "Wait" should continue to predict in subsequent frames. In contrast, for the weak trackers, unmatching is more likely because the tracking object is lost. The waiting state trackers should keep the previous positions in the next frames as in the sleep state.

We adopt ECO and the linear KCF with HoG features as the high-performance trackers. In addition, the linear observation Kalman filters with the constant-velocity motion model and the simple linear interpolation model are used as two types of weak trackers. Note that the detected location is the observation z in Kalman filters, and the state x is modeled as

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}, \dot{r}] \tag{1}$$

where (u, v) is the center position of the bounding box, and s and r represent the area and the aspect ratio of the bounding box, respectively. $\dot{u}$, $\dot{v}$, $\dot{s}$, and $\dot{r}$ are the respective velocities in image coordinates.

The tracking module can efficiently interpolate and associate the detection results in temporal. After tracking, the movements of hand in the untrimmed video V can be formulated as a set of hand trajectories $\Phi = \{T\}$. The hand trajectory consists of a set of associated boxes $T = \{b_t \mid t_s \leq t \leq t_e\}$, where t_s and t_e are the indexes of the first frame and the last frame, respectively. b_t is the corresponding bounding box of the hand in frame I^t.

B. Hand Motion Understanding

In order to understand the semantics of hand trajectories, it is necessary to know when and where the action instances appear in each trajectory. For this purpose, we propose action classification models as baselines and temporal action localization models for more accurate detection.

1) Trajectory Cropping: Most of the deep learning models for video content analysis require fixed-length video segments as inputs, such as two-stream networks [23] or 3-D convolutional networks (C3D [38]). However, the length of hand trajectories is variable with changing hand size and location in each frame. It is necessary to segment trajectories and crop the patch of frames from the video to obtain the video segments with a fixed number of frames.

We collect a fixed number of frames for action recognition by a fixed-length temporal sliding window and then cropped all frames into image patches with the same size. More specifically, we slide the window without overlap for each trajectory on the corresponding video clip to get the segments. Then, every video segment will be cropped to ignore the unused background by some cropping strategies.

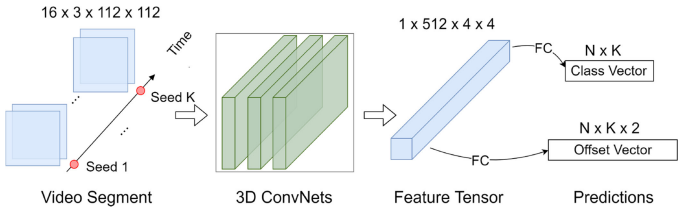


Fig. 4. Key idea of our temporal action localization method. There are two parallel fully connected layers that predict the action class and time offset of the seeds, respectively.

The most natural way is to set a fixed circumscribed cuboid of each hand trajectory segment in the video for cropping. However, it may require a large area to cover a wide-range moving hand. In order to crop out an image patch containing the hand region in each frame without losing the background information in a small area around the hand, another crop method is to crop the area of the extended hand bounding box in each hand trajectory. We crop the image patch from a rectangle region that the center of the region lies in the center of the hand bounding box. The size of the region is set slightly larger than most of the hands that appear in the field of view and adapt only when the hand appears on the border. After cropping, we scale the cropped image patch to a fixed size.

Since this crop method undermines the stationarity of background, classical methods for video processing in the fixed camera, such as background subtraction and change detection [39] cannot be used. However, this cropping method makes attention by retaining the saliency area and tracking the hand in the center of the image patch. Benefiting from the explicit attention mechanism, the deep learning models that we proposed for hand action understanding can actually work better.

2) Temporal Hand Action Localization: Generally, given a video segment, an action model predicts the global properties or frame-level results. The action models we implemented in this paper adopt a sequence of $C \times 112 \times 112$ tensors as input, where C is the number of channels determined by the type of input. The length of the segments L is set to 16, in which the frames are sampled from the original segments with a sampling interval $G = 3$.

Since the movement between two adjacent actions tends to be more intense, it is more likely that different actions break into two trajectories. It is more common that only one action instance exists in a segment. According to our frame-level annotations, each frame has a corresponding action category in the video segment. Therefore, one of the possible solutions for detecting the action instances in the video segment is to predict the unique action label, which is set to the category that appears most frequently over all frames.

We implement the video classification models based on C3D networks [38] and two-stream networks [23], respectively, as baselines. Details of implementation are shown in Table I. All the classification models are trained with a standard cross-entropy loss between the predicted action probability and the action label.

However, if there are multiple action instances in a segment, the classification models that use the most frequent category

TABLE I
BASIC MODEL ARCHITECTURE FOR VIDEO CLASSIFICATION

Layer name	Output size	C3D	Two-stream A		Two-stream B	
			RGB	Flow	RGB	Flow
conv1	$[16\times]\ 56 \times 56$	$3 \times 3 \times 3, 64$	$3 \times 3, 64$	$7 \times 7, 64$	$3 \times 3, 64$	$7 \times 7, 64$
conv2	$[16\times]\ 56 \times 56$	$3 \times 3 \times 3, 128$	$3 \times 3, 128$	$3 \times 3, 128$	$3 \times 3, 128$	$3 \times 3, 128$
conv3_x	$[8\times]\ 28 \times 28$	$[3 \times 3 \times 3, 256] \times 2$	$[3 \times 3, 256] \times 2$	$[3 \times 3, 256] \times 2$	$[3 \times 3, 256] \times 4$	$[3 \times 3, 256] \times 4$
conv4_x	$[4\times]\ 14 \times 14$	$[3 \times 3 \times 3, 512] \times 2$	$[3 \times 3, 512] \times 2$	$[3 \times 3, 512] \times 2$	$[3 \times 3, 512] \times 4$	$[3 \times 3, 512] \times 4$
conv5_x	$[2\times]\ 7 \times 7$	$[3 \times 3 \times 3, 512] \times 2$	$[3 \times 3, 512] \times 2$	$[3 \times 3, 512] \times 2$	$[3 \times 3, 512] \times 4$	$[3 \times 3, 512] \times 4$

Corresponding 2D or 3D max-pooling layers are added to each convolutional block. The first dim of output size is valid only in C3D.

as the label will have poor accuracy at the frame level. Since the actions with a shorter duration are submerged, there is an upper bound for frame-level accuracy of classification models. In order to increase the accuracy, we present a temporal action localization approach to predicting the time boundaries of hand action instance in a small granularity.

The temporal action localization model we proposed is based on 3-D convolution [38], abbreviated as "C3D-TAL," and the basic idea is inspired by the offset regression in object detection. Fig. 4 shows the key idea of our temporal action localization method. We select some frames at equal intervals from the input video segments as temporal seeds, with which we predict the action category and time boundaries. We use the whole topmost feature maps and append two fully connected layers in parallel, one of which predicts the confidence scores of action classes s of each seed, and the other predicts the offsets of the time duration $\hat{t} = \{\hat{t}_x, \hat{t}_y\}$ of the action. Notice that these two fully connected layers can be merged into one without any difference in results. We split it into two for interpretability.

Similar to the anchor mechanism described in Section IV-A1, we require matching each seed to ground-truth instance. We adopt the action class c at the seed frame as the action label of the corresponding seed. The time offsets $t = \{t_x, t_y\}$ are the difference of frame indexes between the seed frames and the frame boundaries of the seed action. Then, the loss for the seed is measured as a weighted sum of a location-based loss and a classification loss as follows:

$$L(s, \hat{t}, c, t; \theta) = L_{\text{cls}}(c, s) + \alpha \cdot L_{\text{loc}}(\hat{t}, t) \tag{2}$$

where θ are the model parameters. The weight term α is set to 1 by cross validation. The classification loss L_{cls} is the softmax loss over multiple-class confidence scores (s). The localization loss is a Smooth L1 loss between the ground-truth offsets (t) and the predicted offsets ($\hat{t}_c$) for the corresponding class c

$$L_{\text{cls}}(c, s) = -\sum_{i=1}^{K} \mathbb{1}(s_i = c) \cdot \log \frac{e^{s_i}}{\sum_{k=1}^{K} e^{s_k}}$$

$$L_{\text{loc}}(\hat{t}, t) = -\sum_{i=1}^{K} \mathbb{1}(\hat{t}_i = c) \cdot \text{Smooth-}\ell_1(\hat{t}_i - t) \tag{3}$$

where $\mathbb{1}(.)$ is the indicator function so that only the prediction that matches the ground-truth class of the seed is considered in the loss.

During testing, given a video segment, we can get the prediction of instances for all seeds. The category and the corresponding offsets of each seed are set to the class with the maximum confidence score. In addition, there might be an overlap for the predicted action instances of different seeds. The category is set to the class with the highest score for the overlapped frames.

The steps mentioned above break the original video into clips by tracking hands, and the clips are segmented and cropped into video segments. The networks predict the action instances in the segments, which can be used to reconstruct the actions in the original video by the index of frames, as the action instances in the adjacent frames of the same trajectory can be merged. Note that the frames that are predicted as "Irrelevant" are discarded as a rectify approach to handle errors in the tracking module.

V. EXPERIMENTS AND DISCUSSIONS

A. Details of Data and Experiments

As described in Section III, there are two types of annotated data to train our learning-based models: keyframe hand annotation and hand action instance annotation. For the purposes of the experiments, all the annotated data is divided into three main subsets: training, validation, and test.

1) Keyframe Hand Labeling: The hands that appear in the field of vision of the camera could belong to the operator at this workstation or someone else and may or may not wear gloves according to the different requirements of the process. We use the bounding box to label every hand in the image except for the hand with severe occlusion or motion blur. We carefully labeled 12 913 images: 9014 for training, 1950 for validation, and the remaining for testing. In order to reduce the number of similar samples, the maximum sampling frequency of the annotation is 0.1 Hz, which means that we annotate one frame from the original video every 10 s and skip over some frames with only minor changes.

2) Hand Action Instance Labeling: The training process in our framework has two stages. We train the hand detection network and extract enough trajectories in advance. For each trajectory, we have frame-level annotations that every action instance in the trajectory is annotated by the action category, the index of the start frame, and the ending frame. The hand action dataset contains six types of common assembly-related atomic action categories and a "Irrelevant" class to exclude the wrong

TABLE II
STATISTICS FOR ACTION CATEGORIES

ID	Action Category	Average Length	Number of Instance	
			Train	Valid
0	Irrelevant	54	2,481	775
1	Idle	393	1,462	545
2	Assemble	527	9,081	3,172
3	Pick Workpieces	54	1,488	486
4	Sort Screws	49	433	132
5	Tighten Screws	172	583	161
6	Scan Barcode	185	375	148

TABLE III
PERFORMANCE COMPARISON OF HAND DETECTION MODELS

	mAP	Size(MB)	GFLOPs	Speed(fps)	
				Env A	Env B
FasterRCNN_res50	96.5	115	79	18.8	-
SSD_vgg	93.3	94	31	66.3	-
Yolo2-full	88.7	202	21	107.5	3.9
Yolo2-tiny	75.4	61	2	162.3	23.8
MobileDet(Ours)	91.3	28	1	140.8	33.1

tracks that are the errors from the hand motion capture module. In particular, the class named "Idle" represents that the hand is at rest without any meaningful operation, and the class named "assemble" represents all meaningful assembly actions except the special actions contained in the categories of the dataset. Statistics for the number of action instances in the training and validation sets are given in Table II. We do not analyze the test set as if it were unknown to us, but it should be similar to other sets.

In the following experiments, we used a machine with 32-GB RAM, Intel Xeon E5-1650 v2 processor, and a Nvidia GTX Titan X (Pascal) GPU card as the server environment. On the other hand, an embedded device, namely Nvidia Jetson TX2, was used as the device-side hardware environment. We abbreviate these two environments as "Env A" and "Env B" in short, respectively.

B. Experiments for Hand Motion Extraction

For both hand detection and tracking, the main aspect to be considered is the tradeoff between the accuracy and the consumption of time and space. In this work, we evaluate the detection models on the test set based on four metrics: mAP metric, inference speed (frames/s), model size, and counting FLOPs (multiply adds). We compare our detection method "MobileDet," which is modified from the original SSD detection framework, with four state-of-the-art methods on our custom hand detection dataset. In addition, all models are under the best hyper-parameters tuned on the validation set, and the resolution of the input image is set to 300×300.

The experimental results in Table III show that all deep learning detection models except Yolo2-tiny perform quite well on our dataset. The MobileDet model results in a slight decrease in accuracy compared to the networks, which have more complicated backbone, such as ResNet50 [40] for Faster RCNN_res50 and VGG16 [41] for SSD_vgg, while it gains a considerable reduction of inference time and FLOPs. The MobileDet model even has around $1.5\times$ increase in the inference speed compared to Yolo2-tiny on the low-cost device-side environment. Besides, without any other model compression tricks, the MobileDet model is the most compact to deploy.

TABLE IV
PERFORMANCE COMPARISON OF HAND TRACKING MODELS

	Det Gap	Ave Len	Ret Accu	Speed(fps)	
				Env A	Env B
ECO	inf.	431.9	-	6.7	-
KCF	inf.	328.3	0.732	209.3	52.5
KCF+matching	10	333.8	0.859	178.2	44.0
KCF+matching	3	475.0	0.861	132.3	31.9
Kalman+matching	3	424.8	0.840	304.5	70.6
Linear+matching	3	361.6	0.822	341.7	78.7

Then, we design an evaluation method to compare the performance of different tracking methods. We randomly select 1000 video clips first and use the most accurate detection model (Faster RCNN-res50) to detect the hands in the first frame as the initial locations for tracking. After that, we run all tracking methods individually and record the results until all hand trackers are lost. Since the tracking methods have different criteria or threshold for the lost, the results of the tracking algorithm are not reliable to make a comparison. In order to objectively determine whether the tracker is on track or has lost, we perform the detection on every frame as the ground truth of hand locations and matching the detection results with the tracking results. We regard a tracker that cannot match in ten consecutive frames lost. We stop this tracker and discard the untracked frames in the results.

The main factors for the tracking module are the robustness and accuracy. Robustness depends on whether the tracker can track the target for a long time, while accuracy reflected in the ability of the tracker to independently discover that it has lost the target. Therefore, we adopt the average actually track length (Ave Len in Table IV) of trajectories as the robustness measurement, the ratio of the number of tracking frames given by the tracker to the verified number of valid tracking frames (Ret Accu in Table IV) as the accuracy measurement.

As shown in Section IV-A2, we compare various trackers with our "tracking-by-matching" framework. In addition, we also evaluate the state-of-the-art algorithms (e.g., ECO [35] and KCF [36]) separately with an infinite keyframe detection interval (Det Gap=inf in Table IV), which means that never detecting and matching hand after setting the initial locations. It is important to note that the speed evaluated here is the average for multiple objects, and the time for keyframe detection is also considered. As shown in Table IV, ECO and KCF can attain hand tracking but have poor performance in discovering lost track, and there is even no determination of loss in the original implementation of ECO. We also evaluated the performance of trackers with different keyframe detection intervals (Det Gap in Table IV) combining with our matching framework, the accuracy can be promoted even with a long interval detection. By

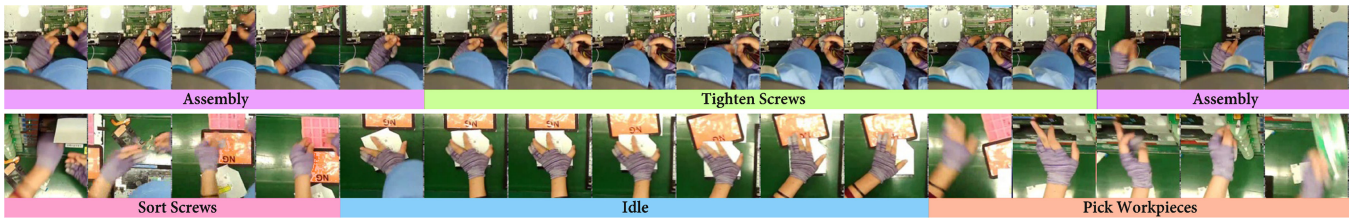


Fig. 5. Sample of two segments. Even severely occluded hand can be tracked by our method, and every image patch in the segment is cropped by the tracked hand location. Although different action categories are similar in appearance, our temporal action localization network can accurately predict all action instances.

TABLE V
PERFORMANCE COMPARISON OF CROPPING STRATEGIES

Accu_seg(%)	C3D	Two-stream A	Two-stream B
w.t. crop	71.6	76.5	77.2
circumscribe cuboid	83.3	80.8	81.1
tracking crop	93.1	79.5	82.6

TABLE VI
PERFORMANCE COMPARISON OF ACTION UNDERSTANDING MODELS

	Size(MB)	GFLOPs	Accu(%)		Speed(seg/s)	
			Seg	Frame	Env A	Env B
C3D	312	38	93.1	79.1	76.4	4.3
Two-stream A	477	5	84.5	71.3	169.8	9.0
Two-stream B	562	8	86.3	68.5	111.1	7.1
G.T. of segments	-	-	100.0	84.3	-	-
CDC	917	43	90.2	87.7	32.3	3.2
C3D-TAL(Ours)	312	39	92.6	89.1	75.2	4.1

reducing the detection interval, the length of tracks becomes longer, and even the predictor with poor performance, such as Kalman filters and linear interpolation, can achieve similar performance to ECO or KCF with a much faster speed.

Since the method of KCF tracker matching with every three-frame detection consumes almost all the computing resources to capture the movement of the hand on the embedded device, there is no spare resource for action understanding. For a compromise between speed and performance, we adopt the tracking method that utilizes Kalman filter predictors matching with every three-frame detection. This method reduces the time for motion capture by half and has similar performance.

C. Experiments for Hand Action Understanding

We evaluate the hand action understanding model with different variants. First, we compare the effects of different cropping strategies that are used to extract a fixed-size video segment for each hand track. We evaluate the classification performance of C3D and two types of two-stream networks with input of the video clips: 1) without cropping; 2) cropped by circumscribed cuboid; and 3) cropped by our tracking-based method.

Table V shows the segment-level classification accuracy of different cropping strategies on different classification models. The classification performance without cropping is poor for both C3D and two-stream networks. As the region of interest becomes smaller by cropping, the accuracy of networks rises. The strategy that crops out the region around the hands in each frame is around 10% higher than cropping in fixed positions for C3D networks. However, the promotion for the two-stream network is not obvious, since the estimation of dense optical flow is not reliable when the hand moves rapidly. Notice that the video segments cropped by tracking hands are actually based on a pseudo moving camera model; there might be slight jitter in the adjacent frames, which introduces a noise for optical flow.

The classification models actually take a role as a coarse temporal action localization method. We evaluate our temporal action localization model for more precise localization with a metric of action localization accuracy at the frame level. Since the two-stream-based methods with a high computational cost to get dense optical flow are not suitable for application, we compare our method with a recent temporal action localization model CDC [26] that is also based on the 3-D convolutional networks.

As shown in Table VI, although the class models archive a high accuracy at the segment level, even if the ground truth of the segment-level label is given, there is a large gap in a frame-level accuracy. By contrast, our temporal action localization model (C3D-TAL) with acceptable additional parameters and time cost has a much higher frame-level accuracy. As shown in Fig. 5, the temporal action localization model can handle segments with multiple action instances, while the classification method does not. Although the two-stream networks are faster than the networks based on 3-D convolution for inference, they require extra time for extracting the dense optical flow, which costs around 12 ms for a pair of frames (i.e., 180 ms for a segment) on a GPU server. It is impractical for real-time computation on a low-cost embedded device. We report the average inference speed of each model. When the sample interval $G = 2$, a segment represents 32 frames (around 1 s) in the video. Generally, there are two segments at the same time, which means we need to detect and track hands in 32 frames and predict actions for two video segments in a second. We implemented a frame data buffer for video processing pipeline, which reads the frames in a separate thread. The data are used by the tracking module and the action module in discrete steps along the dataflow path. Benefiting from our lightweight detection and tracking module, the time consumption is around 450 ms for tracking hands on the embedded device. The remaining time is sufficient for our model to predict actions.

VI. CONCLUSION

In this paper, we presented a lightweight learning-based framework for hand motion capture and action understanding,

which aims to collect the real data of operations in assembly lines for quality control, production planning, and proficiency evaluation. Particularly, a robust tracking method was designed to adapt the challenging hand movements, and a novel temporal hand action localization model was introduced to detect the content of hand operations. The experiments conducted on each part of the frameworks demonstrated the speed and accuracy of our method. Our approach that has been deployed on several production lines in real factories validated the effectiveness of the entire framework.

Our future works are as follows. Firstly, a better categorization of actions will be studied to achieve a more comprehensive action understanding. Secondly, understanding the combinations of atomic actions to detect more complex motion patterns will be more meaningful for the further analysis. Thirdly, the action model can be more compact; two-stream methods with faster dense optical flow estimation will be studied.

REFERENCES

[1] E. Avci *et al.*, "High-speed automated manipulation of microobjects using a two-fingered microhand," *IEEE Trans. Ind. Electron.*, vol. 62, no. 2, pp. 1070–1079, Feb. 2015.

[2] S. Liu, Y.-F. Li, D.-P. Xing, D. Xu, and H. Su, "An efficient insertion control method for precision assembly of cylindrical components," *IEEE Trans. Ind. Electron.*, vol. 64, no. 12, pp. 9355–9365, Dec. 2017.

[3] P. Agethen, M. Otto, S. Mengel, and E. Rukzio, "Using marker-less motion capture systems for walk path analysis in paced assembly flow lines," *Procedia CIRP*, vol. 54, pp. 152–157, 2016.

[4] M. C. O. Moreira, J.-F. Cordeau, A. M. Costa, and G. Laporte, "Robust assembly line balancing with heterogeneous workers," *Comput. Ind. Eng.*, vol. 88, pp. 254–263, 2015.

[5] L. Botti, C. Mora, and A. Regattieri, "Integrating ergonomics and lean manufacturing principles in a hybrid assembly line," *Comput. Ind. Eng.*, vol. 111, pp. 481–491, 2017.

[6] M. M. Savino, D. Battini, and C. Riccio, "Visual management and artificial intelligence integrated in a new fuzzy-based full body postural assessment," *Comput. Ind. Eng.*, vol. 111, pp. 596–608, 2017.

[7] P. Hong, M. Turk, and T. S. Huang, "Gesture modeling and recognition using finite state machines," in *Proc. 4th IEEE Int. Conf. Autom. Face Gesture Recognit.*, 2000, pp. 410–415.

[8] G. Papandreou *et al.*, "Towards accurate multiperson pose estimation in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3711–3719.

[9] Q. Wang, J. Wan, and Y. Yuan, "Locality constraint distance metric learning for traffic congestion detection," *Pattern Recognit.*, vol. 75, pp. 272–281, 2018.

[10] N. Sato and Y. Murata, "Quality control schemes for industrial production by workers' motion capture," in *Proc. 22nd Int. Conf. Adv. Inf. Netw. Appl. Workshops*, 2008, pp. 1480–1485.

[11] T. Stiefmeier, D. Roggen, G. Ogris, P. Lukowicz, and G. Tröster, "Wearable activity tracking in car manufacturing," *IEEE Pervasive Comput.*, vol. 7, no. 2, pp. 42–50, Apr–Jun. 2008.

[12] D. Thewlis, C. Bishop, N. Daniell, and G. Paul, "Next-generation low-cost motion capture systems can provide comparable spatial accuracy to high-end systems," *J. Appl. Biomech.*, vol. 29, no. 1, pp. 112–117, 2013.

[13] A. Baak, M. Müller, G. Bharaj, H.-P. Seidel, and C. Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 1092–1099.

[14] X. Wei, P. Zhang, and J. Chai, "Accurate realtime full-body motion capture using a single depth camera," *ACM Trans. Graph.*, vol. 31, no. 6, 2012, Art. no. 188.

[15] Q. Qiu, Z. Chang, M. Draelos, J. Chen, A. Bronstein, and G. Sapiro, "Low-cost gaze and pulse analysis using realsense," in *Proc. MobiHealth*, 2015, pp. 276–279.

[16] G. Marin, F. Dominio, and P. Zanuttigh, "Hand gesture recognition with leap motion and kinect devices," in *Proc. IEEE Int. Conf. Image Process.*, 2014, pp. 1565–1569.

[17] A. Elhayek *et al.*, "MARCOnI—ConvNet-based MARker-less motion capture in outdoor and indoor scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 3, pp. 501–514, Mar. 2017.

[18] X. Zhou, M. Zhu, G. Pavlakos, S. Leonardos, K. G. Derpanis, and K. Daniilidis, "MonoCap: Monocular human motion capture using a CNN coupled with a geometric prior," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018, to be published, doi: 10.1109/TPAMI.2018.2816031.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Workshop Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[20] T. Simon, H. Joo, I. A. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping." in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4645–4653.

[21] J. C. Niebles and L. Fei-Fei, "A hierarchical model of shape and appearance for human action classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.

[22] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "ActionVLAD: Learning spatio-temporal aggregation for action classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3165–3174.

[23] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Workshop Neural Inf. Process. Syst.*, 2014, pp. 568–576.

[24] L. Wang *et al.*, "Temporal segment networks: Towards good practices for deep action recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 20–36.

[25] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1933–1941.

[26] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang, "CDC: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1417–1426.

[27] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," Center for Res. Comput. Vis., Univ. Central Florida, Orlando, FL, USA, 2012.

[28] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "ActivityNet: A large-scale video benchmark for human activity understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 961–970.

[29] Q. Wang, S. Liu, J. Chanussot, and X. Li, "Scene classification with recurrent attention of VHR remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, 2018, to be published.

[30] C. Bak, A. Kocak, E. Erdem, and A. Erdem, "Spatio-temporal saliency networks for dynamic saliency prediction," *IEEE Trans. Multimedia*, vol. 20, no. 7, pp. 1688–1698, Jul. 2018.

[31] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[33] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Workshop Neural Inf. Process. Syst.*, 2016, pp. 379–387.

[34] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, arXiv:1704.04861.

[35] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6931–6939.

[36] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.

[37] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 3029–3037.

[38] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

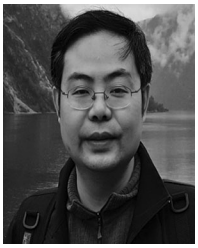
[39] Q. Wang, Z. Yuan, Q. Du, and X. Li, "GETNET: A general end-to-end two-dimensional CNN framework for hyperspectral image change detection," *IEEE Trans. Geosci. Remote Sens*, vol. 57, no. 1, pp. 3–13, Jan. 2019.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.

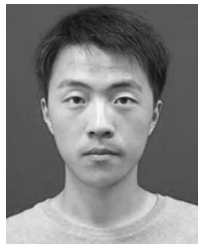
Liang Liu received the B.S. degree in communication engineering from the Zhejiang University of Technology, Hangzhou, China, in 2016. He is currently working toward the Ph.D. degree in control science and engineering with the Institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University, Hangzhou.

His current research interests include computer vision, action recognition, and geometric learning.



Yong Liu (M'11) received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2001 and 2007, respectively.

He is currently a Professor with the Institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. He has authored or coauthored more than 30 research papers in machine learning, computer vision, information fusion, and robotics. His current research interests include machine learning, robotics vision, information processing, and granular computing.



Jiangning Zhang received the B.S. degree in electronic information from Wuhan University, Wuhan, China, in 2017. He is currently working toward the M.S. degree in control science and engineering with the School of Control Science and Engineering, Zhejiang University, Hangzhou, China.

His major research interests include computer vision and temporal action recognition.