Temporal Video Alignment Based on Integrating Multiple Features by Adaptive Weighting

Taiki Sato, Yutaka Shimada, Yukinobu Taniguchi
Tokyo University of Science, Graduate School of Engineering, Japan

***Abstract*—This paper proposes a method for establishing temporal correspondence between two different videos of the same manufacturing processes to enable visual comparison of the work in progress to support the transfer of expertise and skills. We extract two features from work objects and the hand motions of workers from the videos and align the two videos by integrating these features. For integrating the features, we propose a method that adaptively weights the two features to obtain the distance between frames. Using pairs of videos of PC assembly and cooking tasks, we show that our method offers better frame-by-frame alignment accuracy than the methods that employ each feature separately.**

Keywords—Video alignment, Feature integration, SIFT, DP matching

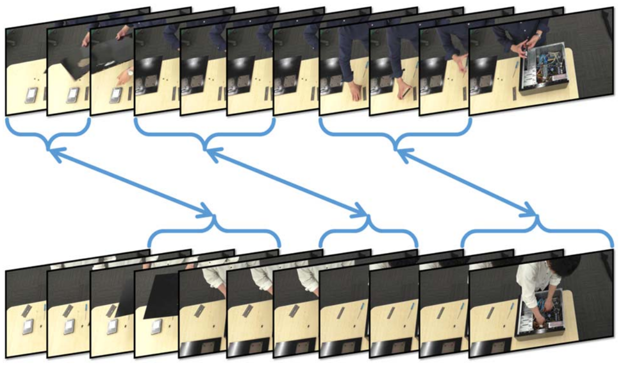


Fig. 1: Conceptual diagram of the video alignment method. We aim at finding scenes of common tasks from two different videos of different workers performing the same manufacturing process.

I. Introduction

Given the rapid advance of globalization in recent years, a key research goal is to develop tools that can assist in the transfer of expertise and skills from experts to non-experts who speak different languages. Watching videos of experts and novices is one of most basic non-verbal approaches for acquiring skills and avoiding problems. However, it is not easy for non-experts to identify the differences in actions between experts and non-experts just by watching videos. The differences are most easily discovered by watching the same tasks, but too much time is needed to find such scenes. This paper introduces an imaging tool to support the acquisition of expertise and skills and proposes a method of temporally aligning multiple videos of the same manufacturing process (Fig.1).

The video alignment problem has been studied in several fields. Higuchi et al. [1] proposed an alignment method based on the hand motion of workers; it establishes a frame-by-frame correspondence between two different videos captured by head mounted cameras. To establish the correspondence, they use the DP (Dynamic Programming) matching algorithm [2]. Lu et al. [3] also proposed a motion based video alignment method that uses the trajectories of the objects of interest by using a tracking method. Gupta et al. [4] focused on human motion and proposed a temporal alignment method that aligns a query video and a human motion capture sequence. Papazoglou et al. [5] proposed an object-appearance-based video alignment method. In their approach, object images from the same viewpoint are detected by using a convolutional neural network, thus achieving frame-by-frame temporal alignment. Kappeler et al. [6] proposed a video copy detection method using an improved DP matching algorithm that reduces the run-time of alignment and saves memory. Their method can align two copies of the same video, one of which has been partially altered by compression, insertion of extra-scenes, overlaid captions, and deletion of scenes. Bojanowski et al. [7] proposed a video alignment method that requires each scene in the video to be tagged with text data representing scene content.

In this paper, we focus on features obtained from work objects, because the work objects are direct indicators of the work being performed in many cases. We propose a temporal alignment method that establishes frame-by-frame correspondence between two different videos without text data by integrating the features obtained from work objects and the hand motions of workers. Experiments show that our method can more accurately align the videos than the methods that use each feature separately.

II. Related Work

In [1], trajectories of feature points are first extracted from hand regions of a worker in the video by using dense trajectories [8]. After that, directions and displacements are calculated from the start and end points of each extracted trajectory, and their histogram is calculated from all frames in the video. Throughout this paper, we call this histogram the motion feature. The two videos are then aligned frame by frame so that the sum of inter-frame distances between motion features is minimized according to the DP matching algorithm. Unfortunately, while this method concentrates on hand motion it fails to detect subtle differences in hand motion. We avoid this problem by also extracting features from work objects.

III. Proposed Method

The proposed method establishes frame-by-frame temporal correspondence between two videos by integrating features extracted from both hand motion of workers and work objects. Our method consists of the following steps: (i) Extract object and motion features from each frame in each video. (ii) Integrate the two features, and compute inter-frame distances between the two videos. (iii) Establish the frame-by-frame temporal correspondence between the two videos by DP matching. Note that we do not track work objects because their trajectories are similar to the trajectories obtained from hand motions. We extract only static features from the work objects in each frame.

Motion feature extraction. In step (i), we first extract motion features from each frame by calculating the trajectories of feature points obtained from hand regions as in [1]. The difference from [1] is that we compute the displacement vector by using the maximum norm of each trajectory. For example, to obtain a displacement vector from a trajectory starting at the $(x(t), y(t))$th pixel in the t-th frame, the maximum norm of this vector is $\max_{\tau} \sqrt{(x(t)-x(t+\tau))^2+(y(t)-y(t+\tau))^2}$, where $1 \leq \tau \leq L$ and L is the number of feature points in the trajectory. To obtain the motion feature of the t-th frame, we gather trajectories passing through the t-th frame and generate the motion feature from the displacement vectors of these trajectories. Let $\boldsymbol{m}(t) = \frac{1}{C_m(t)S}(m_1(t), m_2(t), \cdots, m_{N_m}(t))$ be the histogram, namely the motion feature, of the t-th frame, where $C_m(t)$ is the number of feature points in the t-th frame, S is the diagonal length of the frame, N_m is the number of bins, and $m_j(t)$ is the sum of the norms of the displacement vectors whose angles from the horizontal axis are included in $\left[\frac{2\pi(j-1)}{N_m}, \frac{2\pi j}{N_m}\right]$ $(1 \leq \mathrm{j} \leq N_m)$.

Object feature extraction. We next extract object features from work objects by computing the SIFT feature [9] from the hand regions of workers for each frame. The hand regions of workers are extracted by skin color detection. By computing dictionaries of No visual words, we obtain a Bag-of-Features (BoF) representation for each frame in each video. Let $\boldsymbol{o}(t) = \frac{1}{C_o(t)}(o_1(t), o_2(t), \cdots, o_{N_o}(t))$ be the BoF histogram of the t-th frame, namely the object feature, where $C_o(t)$ is the number of SIFT keypoints in the t-th frame, and $o_k(t)$ is the number of k-th visual words in the t-th frame.

Feature integration. In step (ii), we integrate the object feature and the motion feature to allow determination of the distance, $D^{\mathrm{AB}}(i,j)$, between the i-th frame in video A and the j-th frame in video B as follows:

$$D^{\mathrm{AB}}(i,j) = w_{ij}^{\mathrm{AB}} d\left(\boldsymbol{o}^{\mathrm{A}}(i), \boldsymbol{o}^{\mathrm{B}}(j)\right) + (1-w_{ij}^{\mathrm{AB}}) d\left(\boldsymbol{m}^{\mathrm{A}}(i), \boldsymbol{m}^{\mathrm{B}}(j)\right) \tag{1}$$

where w_{ij}^{AB} is the weight, $\boldsymbol{o}^{\mathrm{X}}(i)$ and $\boldsymbol{m}^{\mathrm{X}}(i)$ are the object and motion features of the i-th frame in the video $\mathrm{X} \in \{\mathrm{A}, \mathrm{B}\}$ respectively, and $d(\boldsymbol{u}, \boldsymbol{u}')$ is the Euclidean distance between histograms $\boldsymbol{u}$ and $\boldsymbol{u}'$ whose sizes are the same. In step (iii), the DP matching algorithm is used to align the two videos frame by frame on the basis of the distance given by (1).

To determine the value of weight w_{ij}^{AB} in (1), we propose two strategies. In the first strategy, the weight is constant, $w_{ij}^{\mathrm{AB}} = w$ for all pairs of i and j. In this case, we train weight w by using training data to maximize the accuracy of alignment.

In the second strategy, for each frame in a video, we count the number of similar frames across the video, that is neighboring frames in the feature space, to determine the value of weight w_{ij}^{AB}. The second strategy comes from the idea that if the number of neighbors of the t-th frame is large, this frame is likely to cause confusion when we look for its neighbors in the other video, because there are many false neighbors of this frame. We use this idea in determining weight w_{ij}^{AB}. Let $\mathcal{M}^{\mathrm{X}}(t) = \{j \mid d(\boldsymbol{m}^{\mathrm{X}}(t), \boldsymbol{m}^{\mathrm{X}}(j)) < \varepsilon_m \wedge j = 1, \cdots, T^{\mathrm{X}}\}$ be a set of ε-neighbors of the t-th frame in video X calculated from the distance between the motion features, and $\mathcal{O}^{\mathrm{X}}(t) = \{j \mid d(\boldsymbol{o}^{\mathrm{X}}(t), \boldsymbol{o}^{\mathrm{X}}(j)) < \varepsilon_o \wedge j = 1, \cdots, T^{\mathrm{X}}\}$ be calculated from the distance between the object features, where ε_m and ε_o are thresholds and T^{X} is the total number of frames in video $\mathrm{X} \in \{\mathrm{A}, \mathrm{B}\}$. When we calculate $\mathcal{M}^{\mathrm{X}}(t)$ and $\mathcal{O}^{\mathrm{X}}(t)$, the frames from $t-\frac{r}{2}$ to $t+\frac{r}{2}$ are excluded, because the distance between temporally neighboring frames are likely to be short and these frames are true nearest neighbors of the t-th frame. Let $\|\mathcal{M}^{\mathrm{X}}(t)\|$ and $\|\mathcal{O}^{\mathrm{X}}(t)\|$ be the number of elements in $\mathcal{M}^{\mathrm{X}}(t)$ and $\mathcal{O}^{\mathrm{X}}(t)$, respectively. Weight w_{ij}^{AB} is defined by:

$$w_{ij}^{\mathrm{AB}} = 1 - \left(\gamma_{ij}^{\mathrm{AB}}\right)^{-1}\left\{\frac{1}{T^{\mathrm{A}}}\|\mathcal{O}^{\mathrm{A}}(i)\| + \frac{1}{T^{\mathrm{B}}}\|\mathcal{O}^{\mathrm{B}}(j)\|\right\},$$

$$\gamma_{ij}^{\mathrm{AB}} = \frac{1}{T^{\mathrm{A}}}(\|\mathcal{M}^{\mathrm{A}}(i)\| + \|\mathcal{O}^{\mathrm{A}}(i)\|) + \frac{1}{T^{\mathrm{B}}}(\|\mathcal{M}^{\mathrm{B}}(j)\| + \|\mathcal{O}^{\mathrm{B}}(j)\|).$$

Larger numbers of ε-neighbors $\|\mathcal{O}^{\mathrm{A}}(i)\|$ and $\|\mathcal{O}^{\mathrm{B}}(j)\|$ as measured by the object features trigger smaller values of weight w_{ij}^{AB}, because it means that the object features are less effective. We determine ε_m and ε_o by applying the Otsu thresholding method [10] to distance distributions of $d\left(\boldsymbol{m}^{\mathrm{A}}(i), \boldsymbol{m}^{\mathrm{B}}(j)\right)$ and $d\left(\boldsymbol{o}^{\mathrm{A}}(i), \boldsymbol{o}^{\mathrm{B}}(j)\right)$.

IV. Experimental conditions

We conducted experiments to evaluate the performance of our video alignment method.

Datasets. In the experiments, we evaluated our methods on two datasets. One holds videos of the assembly of a tower PC, by using a camera installed above the work space (see Fig. 2 and TABLE 1(a) for details). The other, a salad dataset [11], holds videos of people making a salad taken by using camera installed above the cooking space. TABLE 1(b) shows details of the cooking task. In this paper, we assume that each pair of videos show the same order of work. In the case of the PC assembly videos, the order of tasks is the same as TABLE 1(a), while the order of tasks in the cooking videos differs from the order shown in TABLE 1(b). In the cooking

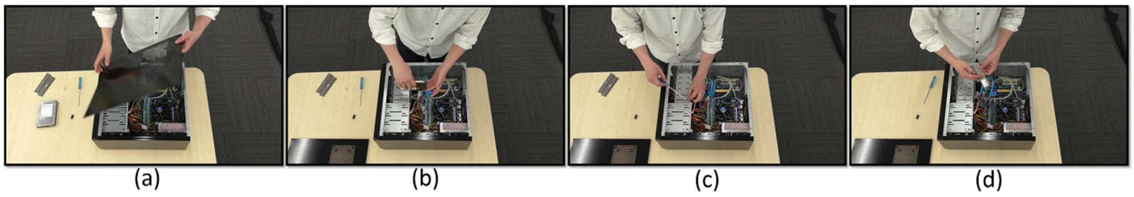


Fig. 2: Examples of frames in a video of PC manufacturing tasks. (a) Opening the PC case. (b) Inserting the HDD. (c) Screwing of the HDD to the PC case. (d) Inserting the memory.

TABLE 1: List of working tasks

	(a) PC manufacturing tasks		(b) salad cooking tasks
1	Removing screws of the case	1	Preparation
2	Opening the PC case	2	Cut the cabbage
3	Inserting the HDD	3	Cut the cheese
4	Screwing of the HDD to the PC case	4	Cut the cucumber
5	Plugging the cable into the HDD	5	Cut the tomato
6	Inserting the memory	6	Serve on the dish
7	Closing the PC case	7	After cooking
8	Screwing the PC case		

video case, the order of tasks is classified into three types: (i) $1\rightarrow2\rightarrow3\rightarrow4\rightarrow5\rightarrow6\rightarrow7$, (ii) $1\rightarrow2\rightarrow5\rightarrow3\rightarrow4\rightarrow6\rightarrow7$, (iii) $1\rightarrow2\rightarrow5\rightarrow4\rightarrow3\rightarrow6\rightarrow7$, where the numbers correspond to those in TABLE 1(b). The number of videos of type (i), (ii) and (iii) are 2, 4 and 3, respectively. In the experiments, we choose a pair of two videos that belong to the same type, and thus the maximum number of pairs of cooking videos is ${}_2C_2 + {}_4C_2 + {}_3C_2 = 10$.

Evaluation procedures. We compare the performance of our alignment method with those of methods that use the object and the motion features separately. In the experiments, we set parameter r to 25% of the total number of frames in each video, the number of visual words of BoF, N_o, to 500, and the number of bins of the motion feature N_m to 8. In the case of the PC assembly videos, the number of videos was 12, where 6 videos were training data and the rest were used to evaluate the performance of our method. Video length was 2 - 4 minutes. In the case of the cooking videos, the number of videos was 9, and all 9 videos were used to evaluate the performance of our method. The training data is the same as that used for the PC assembly tasks. All videos had encoded frame rate of 5fps.

Let t_s^{X} be the ground truth, namely the starting time of the s-th work task in video X ($s = 1, \cdots, n$), where n is the total number of tasks for each type of video. If the t_s^{A}-th frame in video A is matched to the $t_s^{\mathrm{B}'}$-th frame in video B, we estimate that work task s started at time $t_s^{\mathrm{B}'}$. The alignment error is defined by: $\frac{1}{n}\sum_{s=1}^{n}\left|t_s^{\mathrm{B}'} - t_s^{\mathrm{B}}\right|$. In the experiments, we calculate the alignment errors for all pairs of test videos, namely ${}_6C_2$ patterns. We repeated this procedure 30 times by randomly reclassifying the 12 videos as training and testing videos, and the alignment accuracy was evaluated by the average of the ${}_6C_2 \times 30$ alignment errors for the cases of PC assembly videos and $({}_2C_2 + {}_4C_2 + {}_3C_2) \times 30$ alignment errors for the cooking videos. As the baseline, we adopted the linear alignment method that simply estimates the starting time of each work task by $t_s^{\mathrm{B}'} = \frac{T^{\mathrm{B}}}{T^{\mathrm{A}}} t_s^{\mathrm{A}}$, where we assume that A is the experts' video and B is the non-experts' video.

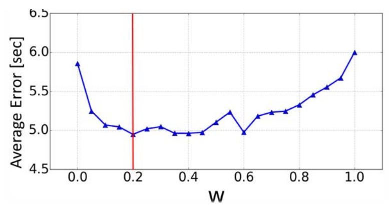


Fig. 3: values of the alignment errors versus weight $\boldsymbol{w}$.

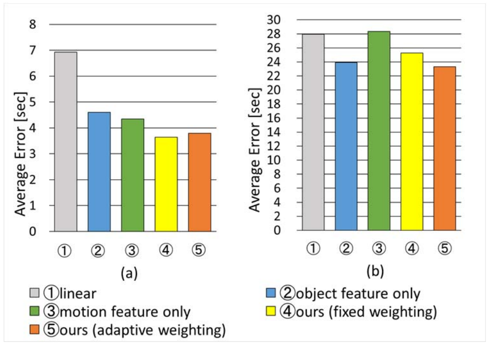


Fig. 4: Average values of the alignment errors in the case of (a) PC manufacturing tasks, (b) salad dataset [11].

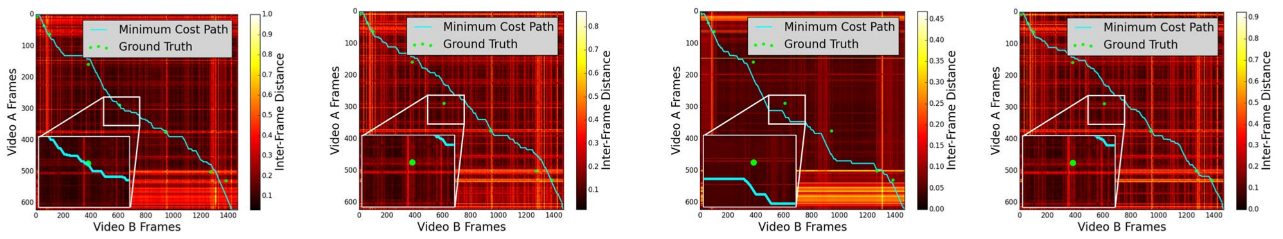
In addition, we also compared our method with dense trajectories [8], a technique that uses four types of descriptors, trajectories, HOG, HOF and MBH, which are obtained along with the trajectories extracted from the hand regions of workers. By computing dictionaries of 500 visual words for these features separately, we obtained BoF representations for each feature and concatenated these BoF representations for each frame in all videos.

In our method with fixed weighting, we set weight w using the training videos, so that the alignment error was minimized. In order to find weight w, the value of w was changed from 0 to 1 in 0.05 increments.

Fig. 3 shows the average values of the alignment errors for each fixed value of weight w. From Fig. 3, as the value of w is changed, the average errors of our method with fixed weighting range from about 4.9 to 6 seconds, which implies that in our method with fixed weighting it is imperative to adjust fixed weight w. In the following assessments, fixed weight w was set to 0.2.

V. Results

Fig. 4 shows the average values of the alignment errors. As shown in Fig. 4(a), in the case of the PC assembly videos, our methods (integrated object and motion features) have lower average errors than the other methods. We also calculate the average value of the alignment errors by using dense trajectories. The average value, 8.26 sec, shows the worst performance. This is mainly because the method detects



(a) ours (adaptive weighting) (b) ours (fixed weighting) (c) object feature only (d) motion feature only

Fig. 5: Example of the minimum cost path obtained by the DP matching algorithm.

trajectories only when hand motions are unsubtle, so that it tends to fail to detect object features in the presence of subtle hand motions, such as screwing or plugging small parts. On the other hand, our method can find object features even when the hand motions are subtle. This indicates that using features obtained from work objects is highly effective in offsetting hand motion detection failures. From Fig. 4(a), our method with fixed weighting has slightly smaller average error than adaptive weighting. However, the difference is slight, and the method with adaptive weighting is training free in that it does not require weight w to be optimized for each task or camera configuration. For these reasons, the method with adaptive weighting is effective in aligning two different videos of the same manufacturing process.

Fig. 4(b) shows the results for the case of the cooking videos. Because of the low resolution of the cooking videos, both object and motion features are not fully extracted from the videos, so the average errors are large in all methods. However, our method with adaptive weighting achieved the lowest average error of all methods. On the other hand, the method with motion feature yielded poor performance. One reason is that the positions of the working objects in the individual videos, such as knives and bowls, differs video to video. These irregular patterns in the positions of the working videos significantly degrade effectiveness of the motion feature: for example, if knife position is switched from left-hand side to right-hand side, the direction of the hand motion in one video is the inverse direction of that in the other video. In this case, the motion based method does not work. On the other hand, our method with adaptive weighting automatically determines which feature is more effective and integrates the object and motion features with reasonable weighting. For this reason, our method with adaptive weighting shows superior performance. From the results of these experiments, our method with integrating object and motion features can be a useful and effective temporal alignment method.

Fig. 5 shows an example of the distance matrixes whose (i, j)th element is the distance between the i-th frame of video A and the j-th frame of video B yielded by the methods tested. The colors in Fig. 5 show the distances and the blue solid lines are the minimum cost paths obtained by the DP matching algorithm. From Fig. 5, the paths deviate greatly from the ground truth (the green filled points) that is the starting time of the work task 5, except for that of our method with adaptive weighting. Because fixed weight w is set to 0.2, the motion feature mainly impacted the results in the method with fixed weighting. For this reason, the alignment results obtained by the method with fixed weighting and the method that used only the motion feature were almost the same. On the other hand, in the method with adaptive weighting, the minimum cost path overlapped the ground truth because the object feature and the motion feature were adaptively selected by our method.

VI. Conclusion

In this paper, we proposed a video alignment method that establishes the frame-by-frame correspondence between two videos of the same work process by adaptively weighting object and motion features. The innovations of our study are as follows: 1) we proposed video alignment method that establishes temporal correspondence between two different videos by integrating object and motion features, 2) our method has higher performance than the methods that use only the object or motion feature, 3) our method is training free when we determine the value of weight w_{ij}^{AB} adaptively based on neighboring frames in the feature space. Although we assume that the videos demonstrate the same sequence of work tasks, some change in task order might occur in reality. Thus, aligning two different videos that contain different task orders is an important future work.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP17K06608.

References

[1] K. Higuchi, R. Yonetani, and Y. Sato, *IEICE Technical Report*, vol. 115, no. 414, pp. 9–14, 2016 (in Japanese).

[2] H. Sakoe and S. Chiba, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[3] C. Lu and M. Mandal, *IEEE Transactions on Multimedia*, vol. 15, no. 1, pp. 70–82, 2013.

[4] A. Gupta, J. He, J. Martinez, J. J. Little, and R. J. Woodham, *IEEE Winter Conference on Applications of Computer Vision*, pp. 1–9, 2016.

[5] A. Papazoglou, L. Del Pero, and V. Ferrari, *The Asian Conference on Computer Vision*, pp. 273–288, 2016.

[6] A. Kappeler, M. Iliadis, H. Wang, and A. K. Katsaggelos, *IEEE International Conference on Image Processing*, pp. 3324–3328, 2016.

[7] P. Bojanowski, R. Lajugie, E. Grave, F. Bach, I. Laptev, J. Ponce, and C. Schmid, *IEEE International Conference on Computer Vision*, pp.4462–4470, 2015.

[8] H. Wang and C. Schmid, *IEEE International Conference on Computer Vision*, pp.3551–3558, 2013.

[9] D. G. Lowe, *IEEE International Conference on Computer Vision*, vol. 2, pp.1150–1157, 1999.

[10] N. Otsu, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[11] S. Stein and S. J. McKenna, *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp.729–738, 2013.