Supplementary Material for "Video Action Segmentation via Contextually Refined Temporal Keypoints"

A. Constructing matching graphs

A.1. Details of permutation loss function

To implement the graph matching module, we simply following the deep graph matching works [6, 13] to learn the matching between the target and source graphs. Let $\mathcal{V}_s, \mathcal{V}_t$ denote the node sets of source and target graphs respectively. $|\mathcal{V}_s| = |\mathcal{V}_t| = M$. The matching is supervised by the permutation loss [10] as:

$$\mathcal{L}_{perm} = -\sum_{i\in\mathcal{V}_s, j\in\mathcal{V}_t} (S^{gt}_{i,j}\log S_{i,j} + (1-S^{gt}_{i,j})(1-\log S_{i,j})), \tag{1}$$

where $S \in [0,1]^{M\times M}$ indicates the doubly (sub-)stochastic matrix generated by Sinkhorn Propagation [1], satisfying $S\mathbf{1} = \mathbf{1}$ and $S^\top\mathbf{1} \leq \mathbf{1}$. Specifically, S is relaxed from a discrete permutation matrix X in the general quadratic assignment programming(QAP) [5] formulation:

$$\max_x \; x^\top \mathbf{Q} x \quad s.t.\; X \in \{0,1\}^{M\times M}, \quad \mathbf{H}x = \mathbf{1}, \tag{2}$$

where x is the column-wise vectorization form of X and $\mathbf{H}$ is a selection matrix ensuring each row and column of X summing to 1. $\mathbf{Q} \in \mathbb{R}^{M^2\times M^2}$ is the affinity matrix used to encodes the node (via diagonal elements) and edge (via off-diagonal elements) affinities / similarities from source / target graphs. In the image matching tasks, researchers often adopt the combination of deep visual networks such as VGG [7] and graph convolutional networks(GCN) [3] to generate the affinity matrix $\mathbf{Q}$. Similarly, we simply adopt the combination of temporal backbones and GCN to calculate the affinities of temporal nodes.

B. Refinement via Rule-Based Re-Assembling

There is an example of RA shown in Figure 1: Firstly, action points between every two adjacent boundary points are representing the same action segment, so they need to be merged into single action point. Notably, the action point with the largest confidence score is kept while its coordinate is averaged using confidence weights among other action points with the same category. Secondly, the boundary points between every two adjacent action points need to be merged. Since boundary points are category-free, the merge operation is a simple confidence-weighted average of their coordinates. Iteratively merging adjacent actions (*i.e.*, re-

Algorithm 1 Rule-Based Re-Assembling

Input: sets of action keypoints as described, S_0 and S_1;
Output: action segmentation result, $\mathcal{S}$;

for $\forall b_j \in S_1$ **do**
  find i, t that $b_j < d_i < \cdots < d_{i+t} < b_{j+1}$
  merge $(d_i, c_i) \ldots (d_{i+t}, c_{i+t})$
end for
for $\forall (d_i, c_i) \in S_0$ **do**
  find j, t that $d_i < b_j < \cdots < b_{j+t} < d_{i+1}$
  merge $b_j \ldots b_{j+t}$
end for
repeat
  for $\forall (d_i, c_i) \in S_0$ that $c_i = c_{i+1}$ **do**
    find j that $d_i < b_j < d_{i+1}$
    delete b_j
    merge $(d_i, c_i), (d_{i+1}, c_{i+1})$
  end for
until S_0, S_1 no longer change.
construct $\mathcal{S}$ by in-painting all c_i from b_j to b_{j+1} that $b_j < d_i < b_{j+1}$
return $\mathcal{S}$

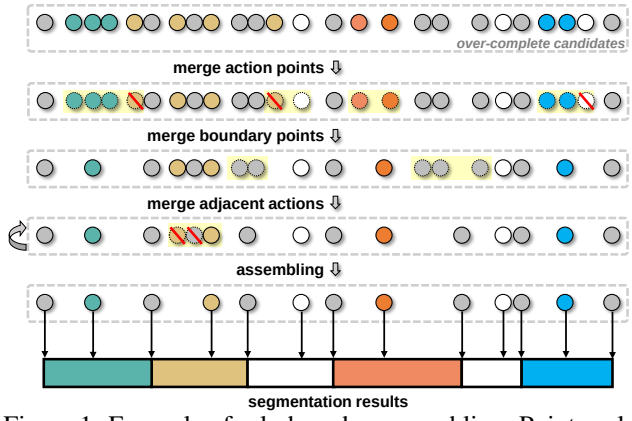
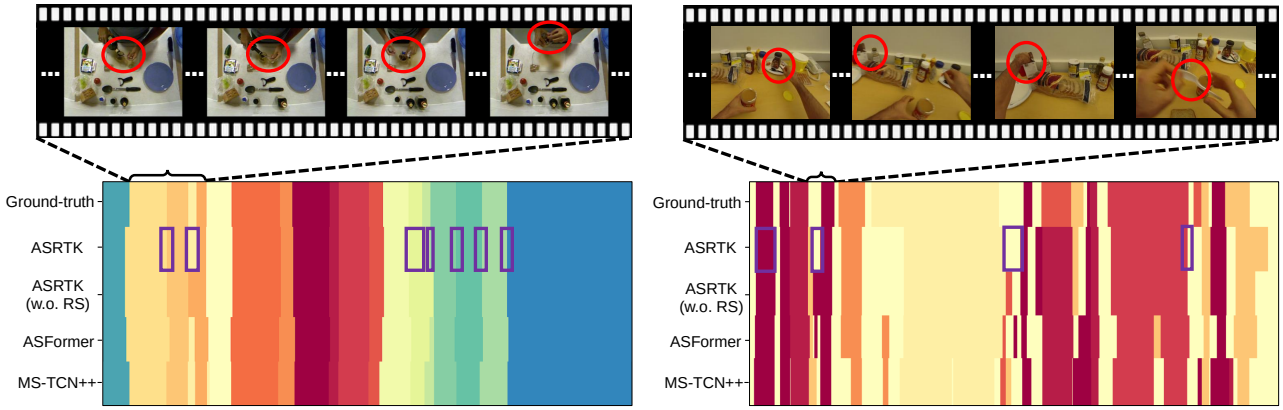


Figure 1: Example of rule-based re-assembling. Points colored in *grey* indicate boundary points, while the other points are action points.



(a) rgb-10-2 from 50salads dataset.

(b) S1-Peanut-C1 from GTEA dataset.

Figure 2: Visualization results of frame-wise classification based (MS-TCN++ [4], ASFormer [12]) and keypoints based (RTKs) methods on 50Salads and GTEA datasets. There are two parts in each subfigure: (1) the key frames of the representative clip from the video; (2) action segmentation results of ground-truth, RTK, RTK without refinement stage (RS), ASFormer, and MS-TCN++. The purple boxes highlight the better results of RTK comparing to the frame-wise classification methods. After refinement, RTK removes out-of-context keypoints and provides more reliable boundary predictions.

moving boundary points between two adjacent action points with the same category, while remaining the action point with higher confidence score) can gradually satisfy this constraint. Therefore, keep repeating this step until the points no longer change. Finally, the alternating action and boundary points are assembled to construct the segmentation results. The pseudo-code of RA is shown in Algorithm 1.

C. Modified ASFormer Backbone

This paper adopted a modified ASFormer [12] backbone for keypoints generation mainly due to reducing the parameters of model for fair comparison with the existing state-of-the-art segmentation methods. There are 2 main modifications: Firstly, the ASFormer Decoders [12] is simplified. This is because existing segmentation methods rely on this kind of architecture to alleviating over-segmentation errors progressively (similarly, Refinement Stages in MS-TCN++ [4], cascade structures in BCN [11]). However, such a structure is not effective in keypoint-based methods. As a result, we adopted simplified ASFormer Decoders [12] by reducing the number of layers and increasing the dilation factor from 2 to 4. Moreover, the kernel size of *Conv Forward* layer is increased from 3 to 5. Secondly, the keypoints generation are integrated into one shared head. The head contains multiple temporal convolutions with Instance normalization [8] and non-linear layers. These two designs of RTK could save a lot of parameters and make keypoints generator efficient.

RTK also adopted positional encoding [9], Squeeze-and-Excitation(SE) block [2] for modeling long temporal dependencies, and average pooling layers for feature smoothing. In Table 1 we study impact of these extra modules for ASFormer [12] baselines and RTK.

Method	F1@{10,25,50}			Edit	Acc
ASFormer [12]	90.1	88.8	79.2	84.6	79.7
+extra	90.4	87.9	80.5	85.1	79.8
RTK	**91.2**	**90.6**	**83.4**	**87.9**	**80.3**

Table 1: Impact of the extra modules in ASFormer and RTK on GTEA dataset.

D. Impact of the gradient flow in GM

In the visualization of the convergence of loss function (shown in Figure 3), turning off the gradient flow of the graph matching module and keypoints generator is helpful for the convergence of permutation loss of graph matching modules.

E. Visualization results

We here show how RTK alleviates the boundary misalignment and over-segmentation problems by analyzing specific visualization results. For boundary-misalignment, boundary points predicted by RTK tend to be closer to the ground truth than the baseline method, as highlighted with purple boxes in Figure 2. In addition, RTK also outperforms the baseline, especially in processing the action segments which depend on the contextual information.

An illustrating example for this point is shown in Fig-

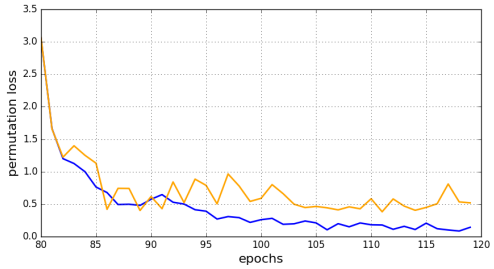


Figure 3: Permutation losses with disabling (in blue) or enabling (in orange) the gradient flow to keypoints generators in the graph matching module during training. The graph module is jointly tuned from epoch 80.

ure 2b. The frames with no interested actions are labeled as *background* (colored in yellow). For the selected video clip, the human made two attempts of taking a spoon (the corresponding action is colored in red in the bottom panel), a first failure with the right hand (since there is actually no spoon on this side) and a second successful spoon-grasping with the left hand. Most baseline methods (including RTK without RS) wrongly predict the first attempt as an action of *take*. In contrast, with more sophisticated contextual modeling, RTK with RS predicts correctly.

References

[1] Ryan Prescott Adams and Richard S. Zemel. Ranking via sinkhorn propagation. *CoRR*, abs/1106.1925, 2011.

[2] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

[3] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*. ICLR, 2017.

[4] Shi-Jie Li, Yazan AbuFarha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[5] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura Netto, Peter Hahn, and Tania Maia Querido. A survey for the quadratic assignment problem. *Eur. J. Oper. Res.*, 176(2):657–690, 2007.

[6] Michal Rolínek, Paul Swoboda, Dominik Zietlow, Anselm Paulus, Vít Musil, and Georg Martius. Deep graph matching via blackbox differentiation of combinatorial solvers. In *European Conference on Computer Vision*, volume 12373, pages 407–424, 2020.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2015.

[8] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. pages 5998–6008, 2017.

[10] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *IEEE International Conference on Computer Vision*, pages 3056–3065, 2019.

[11] Zhenzhi Wang, Ziteng Gao, Limin Wang, Zhifeng Li, and Gangshan Wu. Boundary-aware cascade networks for temporal action segmentation. In *European Conference on Computer Vision*, volume 12370, pages 34–51, 2020.

[12] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. Asformer: Transformer for action segmentation. *CoRR*, abs/2110.08568, 2021.

[13] Andrei Zanfir and Cristian Sminchisescu. Deep learning of graph matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2684–2693, 2018.