Supplementary Material
FACT: Frame-Action Cross-Attention Temporal Modeling for Efficient Action Segmentation

Zijia Lu
Northeastern Univeristy
lu.zij@northeastern.edu

Ehsan Elhamifar
Northeastern University
e.elhamifar@northeastern.edu

In the supplementary material, we provide more details of our FACT framework, including choice of dataset (Section 1), temporal downsampling and upsampling in cross-attentions (Section 2), temporal smooth loss (Section 3), dicussion of matching loss (Section 4) and one-to-many matching algorithm (Section 5). Moreover, we also report more implementation details (Section 6), ablation study (Section 7) and qualitative results (Section 8).

1. Choice of Datasets.

We have evaluated FACT on four datasets. Beside the benchmark datasets, Breakfast and GTEA, we especially choose two challenging new datasets, EgoProceL and EPIC-Kitchen. *They feature long, complex videos, thus can validate FACT's ability for long temporal modeling.* While 50Salads[10] and Assembly101[9] are also action segmentation datasets, they lack the desired properties compared to EgoProceL and EPIC-Kitchen.

Specifically, EgoProceL has videos from *diverse tasks* (cooking, assembling PC, assembling toys and more). Yet, 50Salads and Assembly101 only focus on making salads and assembling toys, respectively. On the other hand, EPIC-Kitchen has *the longest average video lengths and the largest number of action classes (3796 actions)*, which greatly exceed those of 50Salads and Assembly101. Therefore, these properties of EgoProceL and EPIC-Kitchen lead to challenging, sophisticated temporal relations in videos, thus are the best suitable datasets for evaluating model performance on temporal modeling.

2. Temporal Downsampling and Upsampling

When the number of video frames is large, computing cross-attention can be costly. Therefore, we optionally apply temporal downsampling on frame features before cross-attention and upsampling after it to improve efficiency, as visualized in Figure 1. In the following, we use $\mathbf{F}(t)$ to denote the t-th row of a frame feature.

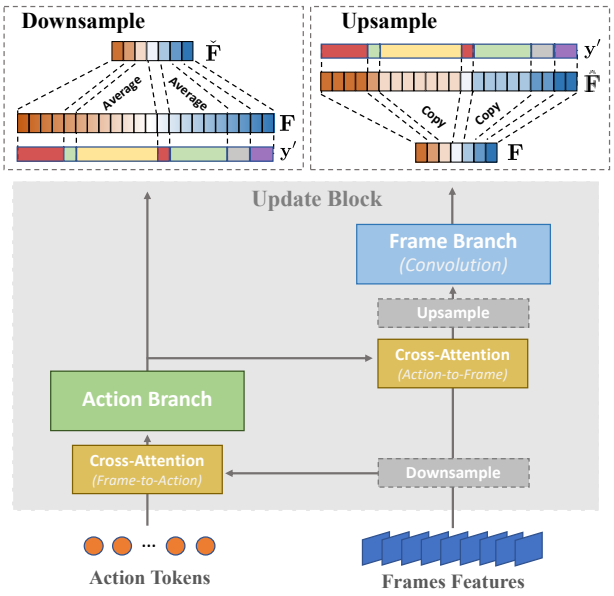

Figure 1. Visualization of Update Block with Temporal Downsampling and Upsampling.

Downsample. For a update block b, we first downsample the input of (frame-to-action) cross-attention, which is the input frame features $\mathbf{F}_{b-1}$. We do not downsample it by a fixed ratio, as it removes the features of short segments. Instead, we partition the video into segments based on the predicted framewise labels, $\boldsymbol{y}' = \text{argmax}(\mathbf{P}^f_{b-1})$, and compute one feature for each segment via average-pooling (see top-left of Figure 1). Specifically, suppose the n-th predicted segment in $\boldsymbol{y}'$ has a temporal interval $\mathcal{T}'_n$. We have

$$\check{\mathbf{F}}_{b-1}(n) = \text{average-pooling}(\{\mathbf{F}_{b-1}(t) | t \in \mathcal{T}'_n\}), \quad (1)$$

where $\check{\mathbf{F}}_{b-1}$ is the downsampled feature. Additionally, we found it is beneficial to refine the features via temporal module, $\check{\mathbf{F}}'_{b-1} = \text{GRU}(\check{\mathbf{F}}_{b-1})$. Finally, $\check{\mathbf{F}}'_{b-1}$ is the downsam-

pled input to the frame-to-action cross-attention.

Upsample. To upsample the output from (action-to-frame) cross-attention $\mathbf{F}'_b$, we make copies of the features in it (see top-right of Figure 1). We have

$$\hat{\mathbf{F}}_b(t) = \mathbf{F}'_b(n) \quad \text{for} \quad t \in \mathcal{T}'_n, \tag{2}$$

where, for each frame t in the predicted segment n, its feature is a copy of $\mathbf{F}'_b(n)$. Since $\hat{\mathbf{F}}_b$ loses the fine-grained details of each frame, we fuse it with $\mathbf{F}_{b-1}$ by a fully-connected layer, $\hat{\mathbf{F}}'_b = \mathrm{FC}(\hat{\mathbf{F}}_b, \mathbf{F}_{b-1})$. Lastly, $\hat{\mathbf{F}}'_b$ is the upsampled output of action-to-frame cross-attention.

3. Temporal Smoothing Loss

Here we present the complete formula of our temporal smoothing loss defined in Eq(13) of the paper. Following [3], we define $\mathcal{L}_{\text{smooth}}$ as

$$\begin{aligned}
\mathcal{L}_{\text{smooth}} &= w \sum_b h(\mathbf{P}^f_b) + h(\mathbf{\Lambda}^a_b) + h(\mathbf{\Lambda}^f_b) \\
&= \frac{w}{TA} \sum_b \sum_{t,a} \psi\left(\mathbf{P}^f_b(t,a), \mathbf{P}^f_b(t-1,a)\right) \\
&+ \frac{w}{TM} \sum_b \sum_{m,t} \psi\left(\mathbf{\Lambda}^a_b(t,m), \mathbf{\Lambda}^a_b(t-1,m)\right) \\
&+ \frac{w}{TM} \sum_b \sum_{m,t} \psi\left(\mathbf{\Lambda}^f_b(t,m), \mathbf{\Lambda}^f_b(t-1,m)\right),
\end{aligned} \tag{3}$$

where $\psi(x,y) = \max(\sigma, |\log x - \log y|)^2$ and σ is a smoothness threshold. Thus, $\mathcal{L}_{\text{smooth}}$ incurs penalty if the difference between predictions of two consecutive frames is too large.

4. Discussion on Matching Loss

Our matching loss finds and enforces the optimal matching between action tokens and action segments to learn action branch and cross-attentions. Alternatively, one can learn them via cross-entropy loss on the framewise prediction of action branch, $\mathbf{P}' = \bar{\mathbf{\Lambda}}^f_B \cdot \bar{\mathbf{P}}^a_B$, as defined in Section 3.1 of paper. Yet, *such method lacks explicit regulation of segment-token matching*. It can lead to one token encoding the frames of one segment and also *a portion of the frames from another segment*. Thus, it fails to ensure each segment is uniquely represented by a token to learn its fine-grained details, while our matching loss achieves so by using the optimal segment-token matching as token labels.

5. One-to-Many Segment-Token Matching

Recall that, our matching loss allows one-to-many segment-token matching, which assigns multiple segments to one token while ensures the assigned segments have the same action class. There is no existing algorithm to compute the optimal assignment. Hungarian algorithm addresses one-to-one matching. Nearest-neighbor matching, which assigns one segment to the token with the maximal matching similarity, can match segments from different classes to the same token. While this problem is NP-hard, we propose a two-step algorithm as an efficient approximation of the optimal solution. If a better matching algorithm becomes available, it will not conflict with the conclusion of our paper, as we can seamlessly leverage it in our method.

Our algorithm first performs a matching between the action tokens and action classes, then applies nearest-neighbor assignment among the matched tokens of a class and the segments of the class, hence ensuring the assigned segments of a token are from the same class. Specifically, in step one, we define the matching similarity between a token m and a class a as

$$\hat{s}(m,a) = \frac{1}{|\mathcal{D}_a|} \sum_{n \in \mathcal{D}_a} s(m,n), \tag{4}$$

where $\mathcal{D}_a$ is the set of segments of class a. Thus, it equals to the averaged matching similarity between the token and the segments of the class a. We utilize $\hat{s}(m,a)$ to match the tokens to action classes. Since the number of tokens is set to be larger than the number of action classes in a video, we ensure each class matches to at least one token while allowing some classes to match with multiple tokens. Thus, we first use Hungarian algorithm to assign one token to each class, then match the remaining unassigned tokens to classes via nearest-neighbor matching. Finally, in step two, for each class, we associate its segments to its matched tokens by nearest-neighbor matching.

Notice the complexity of our one-to-many algorithm is mainly determined by the Hungarian algorithm in step one, thus is similar to that of one-to-one algorithm. Meanwhile, the algorithm is only performed at training time to compute our losses and not required at inference.

6. Implementation Details

We train FACT on one NVIDIA Quadro RTX 6000 with Adam Optimizer and a learning rate of 0.0001. On Breakfast, experiments finish in 20 hours. For prediction generation, we set $w = 0.5$ on EgoProceL and $w = 0.75$ on other datasets. For temporal smoothing loss, we set $\sigma = 4$.

In Table 1, we list the network configuration for each dataset. For example, on Breakfast, we use 1 input block with 3 update blocks and apply temporal down/up-sampling in the $3, 4$-th blocks (last two update blocks) when transcripts are not available and only in the 4-th block when transcripts are available. Each GRU and FC in temporal down/up-sampling contain one layer. We use 4 blocks on most datasets except GTEA, as it is a small-scale dataset and using fewer blocks reduces overfitting. In principle, we maintain a simple network structure and do not extensively tune the hyperparameters.

	Number of Input Block	Number of Update Block	Down/Up-sample Blocks (without transcript)	Down/Up-sample Blocks (with transcript)
Breakfast	1	3	{3, 4}	{4}
GTEA	1	2	{3}	{3}
EgoProceL	1	3	{2, 3, 4}	{4}
EPIC-Kitchen	1	3	{2, 3, 4}	{3, 4}

Table 1. FACT network configurations for the four datasets.

On Breakfast and GTEA, we report the averaged performance over the released four train-test splits that are consistent to all prior works [1, 3, 4, 6, 11]. On EgoProceL, since the dataset does not release any train-test split, we generate one train-test split with 80% video for training and 20% for testing. EgoProceL contains videos from CMU-MMAC dataset [5] where some videos are recorded from the back of the subjects and their actions are not visible. We removed those videos and will release the train-test split with our code and model weights. On EPIC-Kitchen, we use its released one train-test split. For Breakfast, GTEA and EgoProceL, we use I3D[2] to extract framewise features to be consistent with prior works. For EPIC-Kitchen, we use the framewise features released by the dataset.

To replicate UVAST on EgoProceL and EPIC-Kitchen, we use Viterbi decoding with length model [1, 7, 8] as its alignment algorithm, as it shows the best overall performance in its paper and our experiments. To extend prior works to incorporate transcripts, we also utilize Viterbi decoding with length model as the alignment algorithm.

7. Ablation Study

7.1. Performance of Different Update Blocks

In our main experiments, we generate predictions from the last update block. In Table 2 and 3, we also evaluate the predictions of earlier update blocks on Breakfast and EgoProceL, respectively. While the predictions from the first update block have a high Acc, their F1 and Edit are low, showing the predictions have over-segmentation issue, i.e., containing many short false segments. As the frame and token features are refined with more blocks, more accurate action locations are learned, leading to the stead increase in F1 and Edit. We observed adding more update blocks will not further increase performance.

7.2. Effect of Temporal Down/Upsampling

In Table 4, we compare the effect of the Temporal Downsampling and Upsampling (TDU) on EgoProceL. Comparing row 1 and 3 shows including TDU increases F1@50 by 1.6%, as it decreases the length of frame features in cross-attention and improves both computation efficiency and learning difficulty. Meanwhile, comparing row 2 and 3 shows including GRU helps learning action ordering and increases Edit scores.

Block	F1@{10,25,50}			Edit	Acc
First	72.0	69.0	59.5	69.5	74.1
Second	74.9	72.2	62.9	71.5	74.9
Last	81.7	77.0	66.5	79.6	76.5

Table 2. Performance of different update blocks on Breakfast.

Block	F1@{10,25,50}			Edit	Acc	AccB
First	64.6	61.7	51.0	68.6	74.0	86.8
Second	67.8	64.6	53.3	70.0	75.9	87.1
Last	73.0	69.8	60.8	75.7	77.6	88.0

Table 3. Performance of different update blocks on EgoProceL.

TDU	GRU	F1@{10,25,50}			Edit	Acc	AccB
✗	✗	71.3	69.0	59.2	74.2	78.5	87.8
✓	✗	72.0	68.5	60.2	74.2	78.2	88.1
✓	✓	73.0	69.8	60.8	75.7	77.6	88.0

Table 4. Effect of Temporal Downsampling and Upsampling (TDU).

	F1@{10,25,50}			Edit	Acc
FACT	89.9	85.6	73.7	93.5	84.5
FACT + Viterbi	93.1	88.5	76.1	-	85.1

Table 5. Action Segmentation with Transcript and Viterbi decoding.

7.3. Incorporating Viterbi decoding in FACT.

For action segmentation with transcript, we extend prior works with Viterbi decoding, which finds the optimal alignment between their framewise predictions and video transcripts. In contrast, FACT efficiently computes the alignments with cross-attention. In Table 5, we show the results of also applying Viterbi decoding to FACT on Breakfast. Yet, the performance gap with and without Viterbi decoding is not large, showing our cross-attention yields accurate alignments thus removing the need of costly Viterbi decoding.

8. Qualitative Results

In Figure 2, 3, 4, 5, we visualize the segmentation results of FACT with UVAST, LTContext, DiffAct on each of our evaluation datasets. On GTEA, LTContext is not included as it does not release trained model for the dataset. On EPIC-Kitchen, since UVAST and DiffAct cannot converge

well, we replace them with ASFormer. While videos in EPIC-Kitchen are long and contain many action segments, we visualize for clips of the videos instead of whole videos to avoid cluttering. It can be observed that FACT better recognizes both the action classes and temporal locations of action segments.

References

[1] Nadine Behrmann, S. Alireza Golestaneh, Zico Kolter, Juergen Gall, and Mehdi Noroozi. Unified fully and timestamp supervised temporal action segmentation via sequence to sequence translation. In *ECCV*, 2022.

[2] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[3] Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3575–3584, 2019.

[4] Yuchi Ishikawa, Seito Kasai, Yoshimitsu Aoki, and Hirokatsu Kataoka. Alleviating over-segmentation errors by detecting action boundaries. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2322–2331, 2021.

[5] Fernando De la Torre, Jessica K. Hodgins, Adam W. Bargteil, Xavier Martin, J. Robert Macey, Alex Tusell Collado, and Pep Beltran. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. In *Robotics Institute*, 2008.

[6] Shi-Jie Li, Yazan AbuFarha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. Ms-tcn++: Multi-stage temporal convolutional network for action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[7] Z. Lu and E. Elhamifar. Weakly-supervised action segmentation and alignment via transcript-aware union-of-subspaces learning. *International Conference on Computer Vision*, 2021.

[8] A. Richard, H. Kuehne, A. Iqbal, and J. Gall. Neuralnetwork-viterbi: A framework for weakly supervised video learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[9] Fadime Sener, Dibyadip Chatterjee, Daniel Shelepov, Kun He, Dipika Singhania, Robert Wang, and Angela Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[10] Sebastian Stein and Stephen J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2013.

[11] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. Asformer: Transformer for action segmentation. In *The British Machine Vision Conference (BMVC)*, 2021.

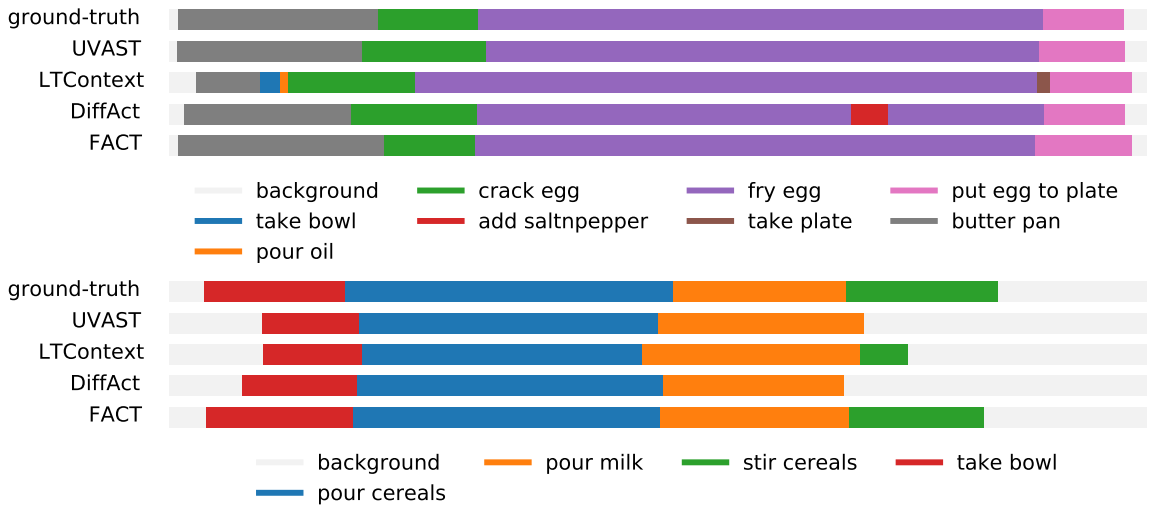

Figure 2. Visualization of Segmentation results on Breakfast

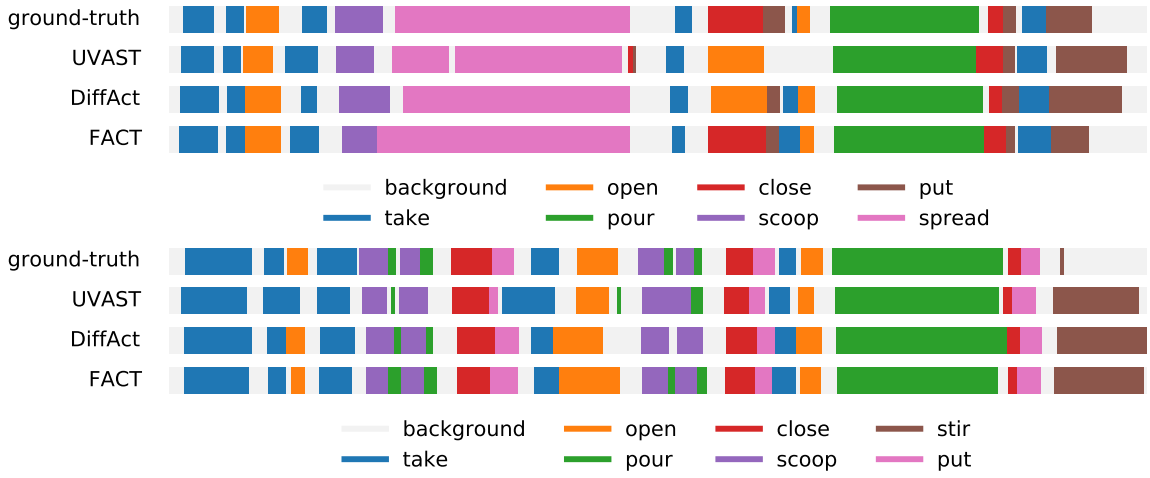

Figure 3. Visualization of Segmentation results on GTEA

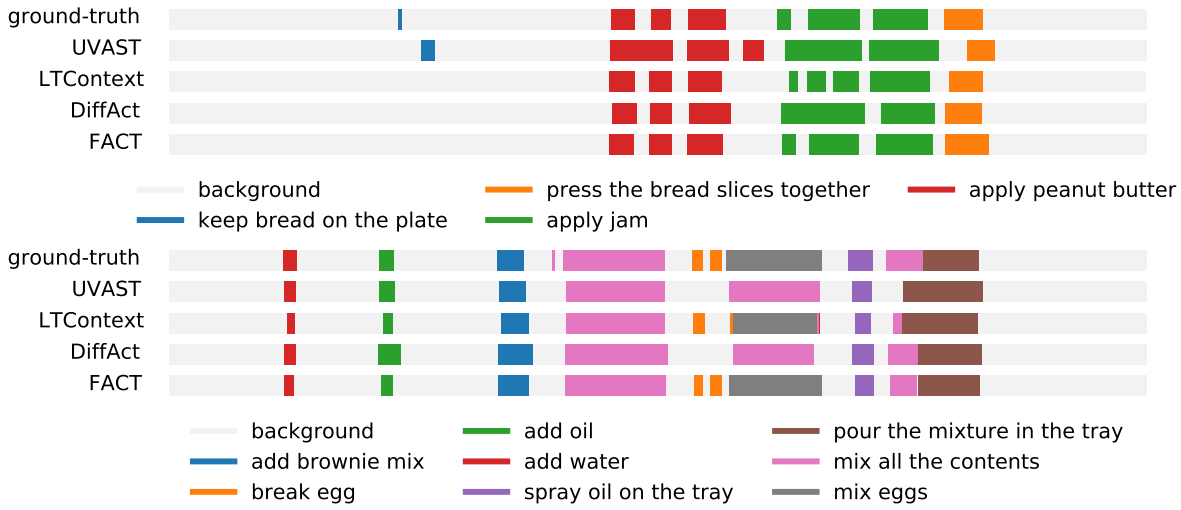

Figure 4. Visualization of Segmentation results on EgoProceL

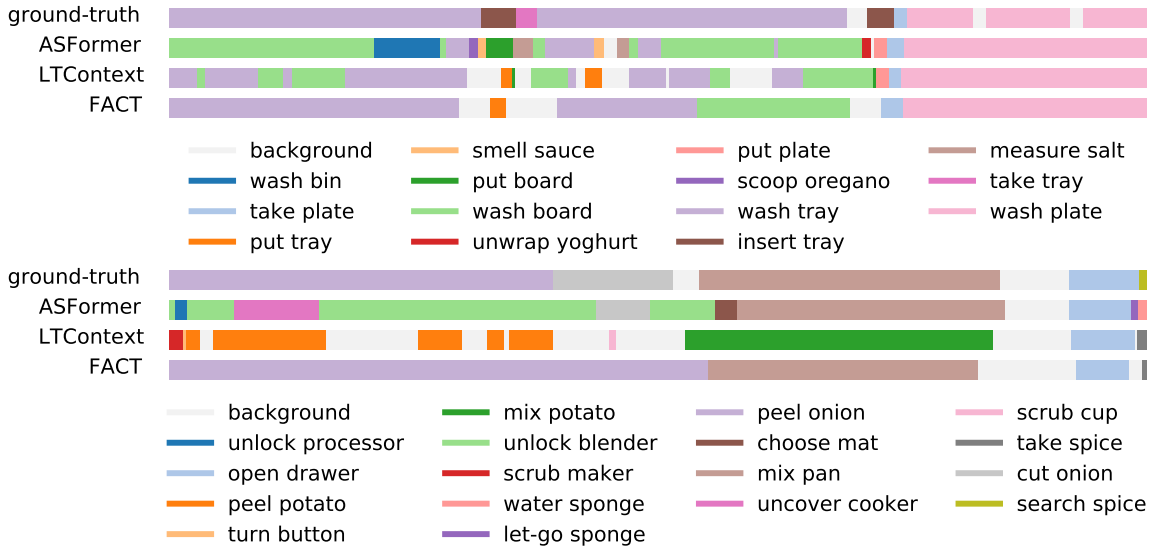

Figure 5. Visualization of Segmentation results on EPCI-Kitchen