Decoupled Motion Expression Video Segmentation

Supplementary Material

Here we compare the performance of DMVS on different VIS methods in Sec. 6. In Sec. 7, we present PyTorch-style pseudo-code for DMVS and LMPM to clearly demonstrate our architecture design. Also, more visualizations on MeViS dataset are provided in Sec. 8.

6. Performance Comparison of Different VIS

We compare the performance of DMVS on different VIS methods in Tab. 10. Mask2Former-VIS [7] is an offline VIS model based on a simple implementation of Mask2Former. Its results on the VIS datasets are weaker than VITA, and its performance on MeViS is also lower than DMVS based on VITA. CTVIS [50] is the state-of-the-art VIS model, and its performance on the VIS dataset is superior to VITA. But there is no advantage on MeVIS mainly because as an online model, there are no video queries, only frame queries interact with text information, and the relevant modules of DMVS cannot function effectively.

7. Pseudo-code Comparison

As shown in Tab. 9 and Tab. 11, we present PyTorch-style pseudo-code for DMVS and LMPM to clearly demonstrate our architecture design.

Table 9. PyTorch-style pseudo-code of LMPM.

```
def LMPM(video, text):
    # froze
    text_encoder.froze()

    # Mask2Former
    frame_queries = []
    mask_features = []
    for frame in video:
        feature = backbone(frame)
        fq, mf = mask2former(feature)
        frame_queries.append(fq)
        mask_features.append(mf)

    # Text Encoder
    word_features, sen_embed= text_encoder(text)

    # VITA Encoder
    frame_queries = vita_encoder(frame_queries)

    # VITA Decoder
    motion_queries = sen_embed /
                .unsqueeze(0).repeat(k, 1)
    motion_queries = vita_decoder(
            motion_queries, frame_queries)

    return motion_queries
```

Table 10. Performance Comparison of Different VIS.

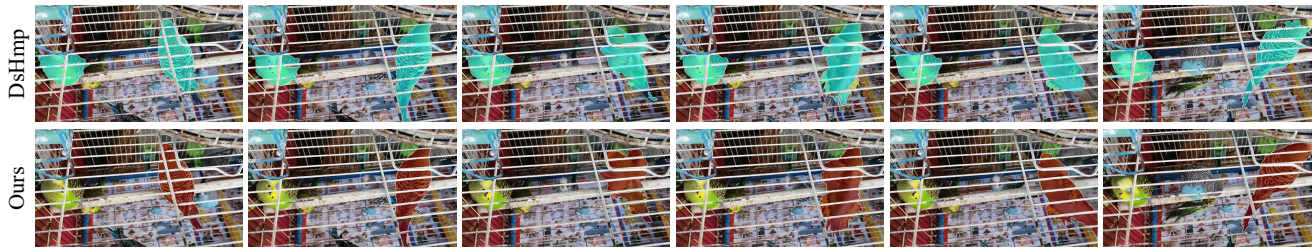
Dataset	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
Mask2Former-VIS [7]	46.4	42.8	50.0
CTVIS [50]	46.8	43.2	50.4
VITA [19]	**48.6**	**44.2**	**52.9**

Table 11. PyTorch-style pseudo-code of DMVS.

```
def DMVS(video, text):
    # froze
    mask2former.froze()
    vita.froze()
    text_encoder.froze()

    # Mask2Former
    frame_queries = []
    mask_features = []
    for frame in video:
        feature = backbone(frame)
        fq, mf = mask2former(feature)
        frame_queries.append(fq)
        mask_features.append(mf)

    # VITA
    video_queries, class_scores = \
                vita(frame_queries)

    # Text Encoder
    word_features = text_encoder(text)
    motion_cues, static_cues, sen_embed = \
               text_decoupler(word_features)

    # Motion Expression Encoder
    motion_expression = dmvs_encoder(
        motion_cues, static_cues, sen_embed,
        frame_queries, video_queries)

    # Motion Query Decoder
    motion_queries = TopK(
            video_queries, class_scores, k)
    motion_queries = dmvs_decoder(motion_queries,
            motion_expression, word_features)

    return motion_queries
```

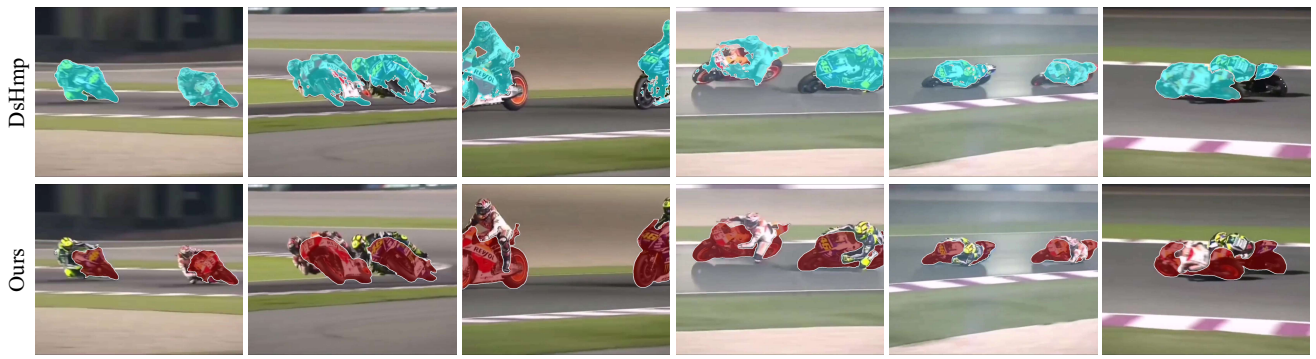
8. Visualization Results

As shown in Fig. 5, we present more visualization results on MeViS [10]. For example, DsHmp [18] does not understand "slightly moving towards the right", but instead segment all the birds standing on the pole. DMVS accurately understands motion expressions and segment the specified objects.



(a) *"The bird standing on the pole, slightly moving towards the right"*

(b) *"The two girls moving towards the left"*



(c) *"A back-and-forth between two motorcycles as they take turns leading the way"*

Figure 5. Qualitative comparison of our method to the main counterpart DsHmp [18] on the MeViS [10] val set.