# One Trajectory, One Token: Grounded Video Tokenization via Panoptic Sub-object Trajectory

Supplementary Material

## 1. More Implementation Details

We provide the complete training details in Table 1. We optimize the models using AdamW optimizer [1] with a learning rate of $10^{-4}$, a weight decay of $10^{-2}$, and mixed precision training. We adopt a cosine annealing learning rate schedule. The contrastive view (batch size) for video training is set to 256, and all models are trained for 30 epochs. We train all models with 32 NVIDIA H100 GPUs. For data augmentation, we apply a combination of random ColorJitter, Grayscale, Gaussian blur, horizontal flip, and resized cropping during training. At testing, we use only a simple resizing operation to ensure consistency.

**Hyperparameters for downstream MAP probing.** For two downstream evaluations that require MAP probing, We use AdamW optimizer with weight decay 0.5, and set learning rate to be 0.0001. We also layer-normalize the video features before providing them to the classifier. We use a batch size of 128, and we train the classifier for 12 epochs.

| Hyperparameter | Value |
|---|---|
| Trasformer size | vit-large |
| Resolution | 224 |
| Frame sampling | uniform 16 frames |
| Optimizer | AdamW |
| Base LR | $1e^{-4}$ |
| Weight decay | 0.02 |
| Optimizer momentum | $\beta_1 = 0.9, \beta_2 = 0.999$ |
| Batch size | video-256, image-4096 |
| Training epochs | 30 |
| LR schedule | cosine decay |
| Warm up epochs | 1 |
| Warm up schedule | linear warm-up |
| Random crop scale | (0.2, 1.0) |
| Random crop ratio | (3/4, 4/3) |
| Horizontal flip probability | 0.5 |
| Color jitter probability | 0.8 |
| Gaussian blur probability | 0.5 |
| Grayscale probability | 0.2 |

Table 1. hyperparameters used for pre-training.

## 2. More Architecture Details

To complement the main paper, we provide additional details on our model architecture and TokenMerge baseline's architecture.

**Trajectory Encoder.** We provide the complete architectural details of our trajectory tokenizer in table **??**. As shown, the parameter size of our tokenizer is an order of magnitude smaller compared with main transformer.

**TokenMerge Baseline.** Although the size of our trajectory encoder is very small (20M) compared with the transformer encoder (304M), to ensure that our improvements do not simply come from adding parameters, we train a model that uses exactly the same modules as TrajViT but uses a learnable token merging mechanism that does not incorporate trajectory priors. The architecture of the TokenMerge baseline is illustrated in Figure 1. We design it such that the only difference from our trajectory tokenizer is whether it incorporates trajectory priors when compressing tokens. All other architectural modules remain identical to ensure a controlled comparison. The output token number is set to be 1024 to match the average FLOPs of our model at training set (including trajectory generation).

## 3. Key Frame Detection Algorithm

We illustrate the details of our key frame detection algorithm, which ensembles three sub-detectors to ensure robust scene boundary identification. A frame is classified as a key frame if it is proposed by at least two out of the three detectors. All detectors are implemented using the Content-Aware Detector from the PySceneDetect package.

**HSV Colorspace Detector.** This detector operates in the HSV color space. Each frame is converted from RGB to HSV, and the average difference across all channels is computed frame by frame. A scene change is triggered if the difference between adjacent frames exceeds threshold 27.

**Luminance Histogram Detector.** Each frame is converted from its original color space to YCbCr, and the Y channel (luminance) is extracted. The normalized histogram of the Y channel in the current frame is then compared to that of the previous frame using the correlation method (cv2.HISTCM_CORREL). A scene change is detected if the histogram correlation between consecutive frames falls below a set threshold 0.15.

**RGB Detector.** This detector computes an intensity value for each frame by averaging the R, G, and B values across all pixels, yielding a single floating-point number. A scene cut is triggered if the intensity difference between consecutive frames exceeds threshold 12.

| Module | Detail | Output Shape | Parameter Size |
|---|---|---|---|
| Per-frame Feature Extractor | sum { ResNet18 stage 1 + linear (64→64) + resize (56×56); ResNet18 stage 2 + linear (128→64) + resize (28→56); ResNet18 stage 3 + linear (256→64) + resize (14→56); ResNet18 stage 4 + linear (512→64) + resize (7→56) } | $T \times 56 \times 56 \times 64$ | 11.6M |
| Mask Pooling | Mask pooling per trajectory (total $N$ trajectories) | $N \times T \times 64$ | 0 |
| Sinusoidal Encoder | bounding box coordinate (4) → high-dimensional embeddings (64) | $N \times T \times 64$ | 0 |
| Perceiver Resampler | Multi-head cross attention { Query: 1; Layers: 1; Heads: 8; Dim: 64 } × 2 | $N \times T \times 64$ | 8.4M |
| MLP | Linear (64→1024) × 2 | $N \times 1024$ | 0.13M |
| *Main Transformer* | *Transformer module of ViT-Large* | $N \times 1024$ | *304M* |

Table 2. Detailed architecture of our model.

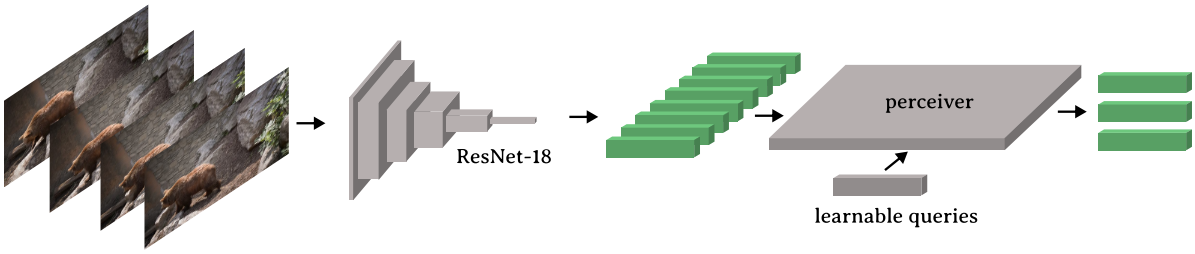


Figure 1. Architecture for TokenMerge baseline.

| Model | K400 | SSV2 | UFC-101 |
|---|---|---|---|
| ViT3D | 42.0 | **12.3** | 40.4 |
| TokenLearner | 40.9 | 11.0 | 37.8 |
| ViViT | 39.9 | 11.5 | 34.8 |
| AutoMerge | 38.4 | 10.3 | 35.4 |
| RLT | 41.0 | 10.3 | 33.7 |
| ToMe | 38.2 | 9.9 | 37.1 |
| TrajViT (ours) | **42.4** | 11.8 | **42.1** |

Table 3. **Zero-shot action classification performance.** We report top-5 accuracy.

## 4. Detailed setup in AVAv2 Spatial Temporal Detection task

We follow the setup in [2] to evaluate our model on the AVAv2 spatial-temporal action detection task. In this task, given an object's bounding box in a specific video frame, the model must predict the action associated with that object at that time instant. This requires extracting video features corresponding to the region of interest (ROI) and applying a probing head to classify the action based on the localized features. We use the same attentive probing head across all models, but the ROI pooling strategy differs depending on each model's tokenization mechanism, which we illustrate below:

**ViT3D.** ViT3D produces a spatial-temporal feature map, allowing us to use ROIAlign to extract features corresponding to the given bounding box. This setup is the same as [2]

**Our Model.** Since each output token in our model corresponds to an object trajectory, we leverage its segmentation mask at each timestep to determine its presence within the bounding box. We gather all tokens whose trajectories have at least 80% of their segmentation mask area inside the bounding box at the annotated frame.

**ViViT.** ViViT is a two-stage model, where the first stage outputs spatial features, and the second stage extracts temporal features. We handle this by pooling its spatial features using ROIAlign and selecting the corresponding temporal feature based on the annotated timestep. The final feature is obtained by concatenating the pooled spatial and temporal representations.

**TokenLearner.** We use the TokenFuser module proposed in its original paper to reproject pruned tokens back to their original spatial locations. This allows us to perform feature pooling in the same manner as ViT3D.

**RLT & ToMe.** Both RLT and ToMe dynamically merge space-time patch tokens that are identified as redundant. We reassemble the feature map by duplicating merged features back to their corresponding redundant patches, then ROI pool the reconstructed feature map in the same way as ViT3D.

**TokenMerge.** Since TokenMerge learns to merge tokens in a fully data-driven manner, it does not retain explicit spatial correspondences to the original input grid. As a result, we

| Training Data | Model | ActivityNet | | VATEX | | MSR-VTT | | Charades | |
|---|---|---|---|---|---|---|---|---|---|
| | | txt2vid | vid2txt | txt2vid | vid2txt | txt2vid | vid2txt | txt2vid | vid2txt |
| panda-2m | ViT3D | 26.33 | 27.33 | 24.74 | 44.80 | 23.75 | 48.30 | 7.11 | 7.29 |
| | TrajViT (ours) | 31.97 | 31.97 | 28.94 | 51.40 | 26.94 | 50.60 | 10.14 | 10.47 |
| panda-4m | ViT3D | 35.34 | 34.54 | 35.00 | 59.11 | 30.90 | 56.61 | 12.61 | 12.61 |
| | TrajViT (ours) | 38.62 | 38.41 | 36.19 | 61.02 | 31.71 | 60.52 | 14.81 | 14.81 |
| panda-8m | ViT3D | 38.82 | 37.46 | 40.59 | 64.46 | 34.71 | 60.83 | 17.45 | 16.00 |
| | TrajViT (ours) | 42.35 | 41.92 | 41.35 | 65.35 | 35.22 | 62.73 | 19.41 | 18.36 |

Table 4. **Full retrieval performance for pretraining video data scaling experiment.** We report results on four commonly used video retrieval datasets for both text-to-video (`txt2vid`) and video-to-text (`vid2txt`).

| Training Data | Model | ActivityNet | | VATEX | | MSR-VTT | | Charades | |
|---|---|---|---|---|---|---|---|---|---|
| | | txt2vid | vid2txt | txt2vid | vid2txt | txt2vid | vid2txt | txt2vid | vid2txt |
| panda8m | ViT3D | 37.82 | 34.54 | 39.59 | 59.11 | 33.71 | 56.51 | 15.35 | 12.61 |
| | TrajViT (ours) | 41.35 | 38.41 | 40.35 | 61.02 | 34.22 | 61.00 | 18.41 | 14.81 |
| panda8m + datacomp50m | ViT3D | 43.62 | 44.65 | 47.16 | 70.70 | 41.16 | 68.74 | 21.25 | 20.50 |
| | TrajViT (ours) | 53.57 | 53.36 | 50.38 | 75.10 | 47.38 | 79.96 | 24.80 | 22.01 |

Table 5. **Full retrieval performance for incorporating image data experiment.** We report R@5 scores on four commonly used video retrieval datasets. `txt2vid` is text-to-video retrieval and `vid2txt` is video-to-text retrieval.

| Model | ImageNet | COCO | |
|---|---|---|---|
| metrics | Top-5 Acc | img2txt R@5 | txt2img R@5 |
| ViT | 77.7 | 73.6 | 58.3 |
| TrajViT (ours) | 74.9 | 71.1 | 55.5 |

Table 6. **Performance for image-only experiments.** We report top-5 accuracy for ImageNet classification and Recall@5 for COCO retrieval (image-to-text & text-to-image).

are unable to pool features corresponding to the region of interest.

## 5. Full tables for scaling performance experiments

We provide the complete table for the scaling up experiments, which we only show the plots of average trend in the main table. Table 4 presents the performance variations of the model with the change of the scale of the training data. Table 5 presents the model's performance with images adding to training data.

## 6. Zero-shot action classification

We report zero-shot action classification performance for all models here as a complement for attentive probing action classification that shown in the main paper. We note that for video model, attentive probing that only involves vision encoder is a more accurate measure for action classification task, because the text template for action is hard to construct and likely to be out-of-distribution for text that model saw during training (e.g. put something on something). Nevertheless, as shown in table 3, our model still has competitive performance under zero-shot setting, outperforming most of baseline models.

## 7. Visualizations of generated trajectories

We show examples of generated trajectories in our training set at figure 2 and figure 3. Our pipeline allows us to generate high-quality panoptic trajectory with high efficiency. The generated segments are in detailed subobject level, allowing us to reason fine-grained interaction. The tracking is also very robust attributing to the powerful SAM2 model. We do observe occasional matching failure for the same objects between sub-clips (like frame 3$\rightarrow$ 4 in example 3), causing the same object being split into multiple trajectories.

## 8. Image only experiments

Since we mention our model can naturally be adapted to image data, it will be interesting to see its performance in image-only training as well. We therefore train our model at datacomp50M image-captioning dataset, and compare it to regular ViT model that trains in the same dataset (table 6. We found our model underperforms ViT in down-

stream evaluation of ImageNet zero-shot classification and COCO zero-shot image-text retrieval. This means our bigger gain in incorperating image data experiment for our model is primarily because our model can train at image and video together, avoiding image-then-video pipeline and the information loss when transferring 2D model's weight to 3D model. How to improve our model design to let it become also competitive in image-only domain is left to future work.

## References


[1] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[2] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022. 2

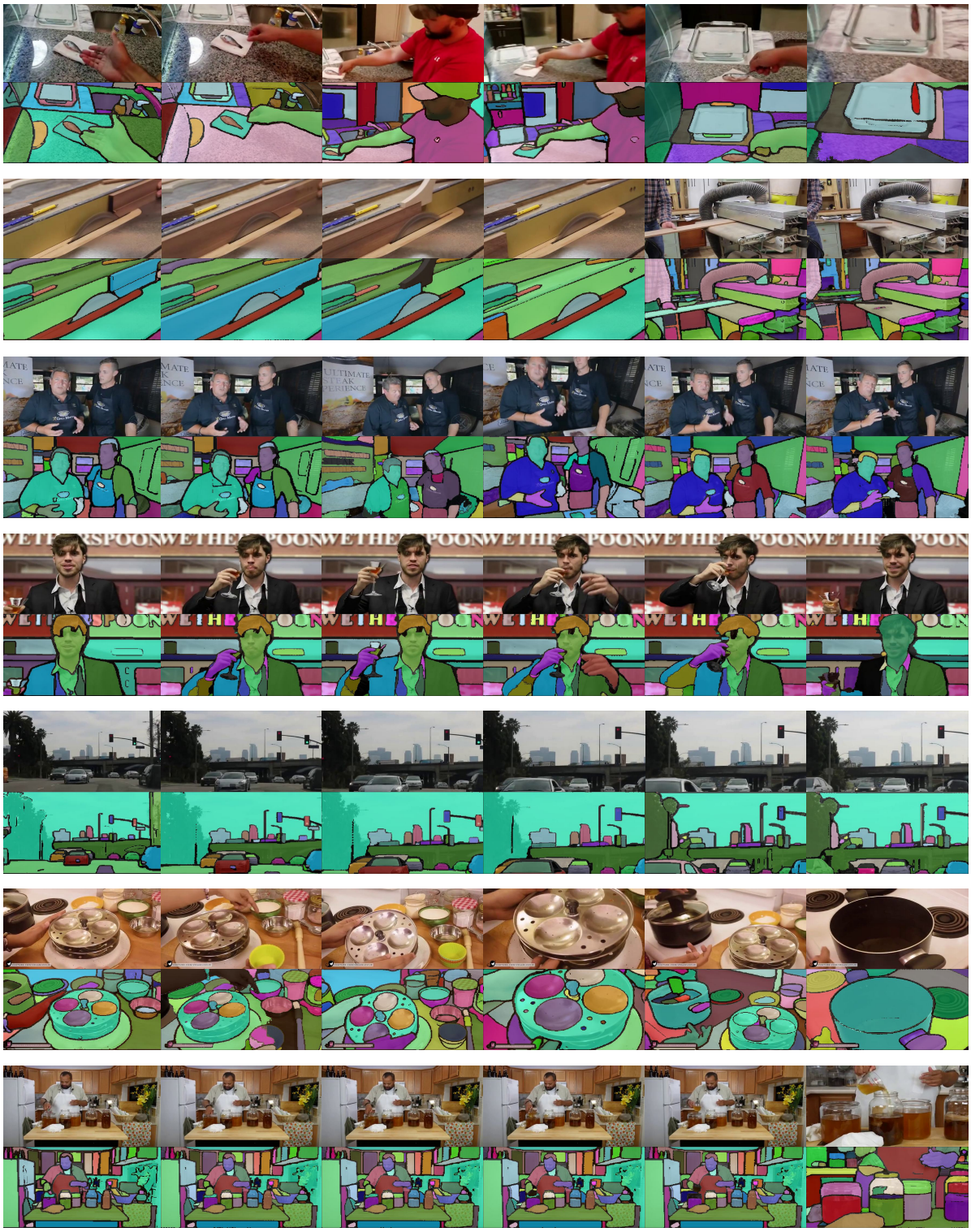

Figure 2. **Visualizations of our generated trajectories (part 1).**

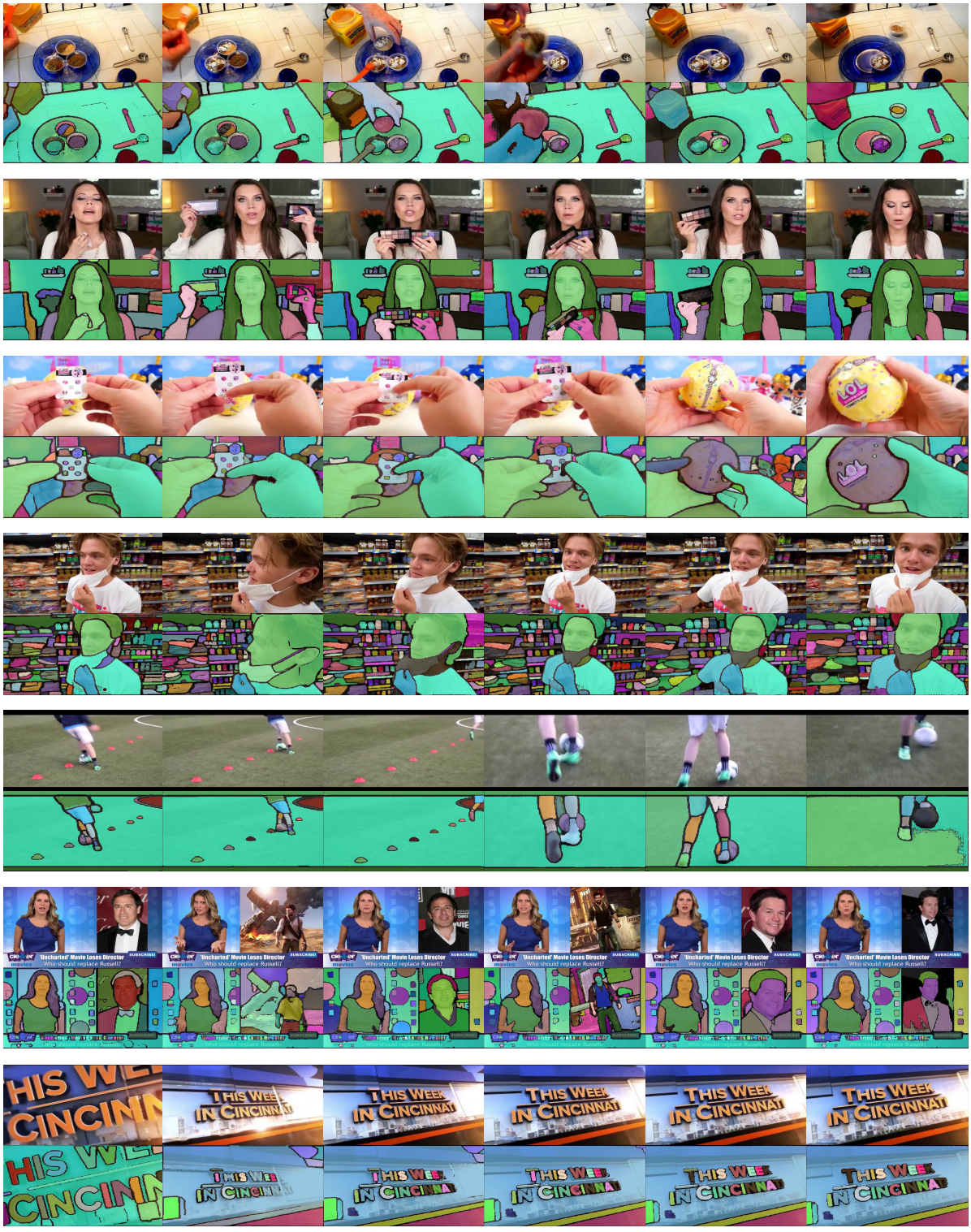


Figure 3. **Visualizations of our generated trajectories (part 2).**