

# Tracktention: Leveraging Point Tracking to Attend Videos Faster and Better

## Supplementary Material

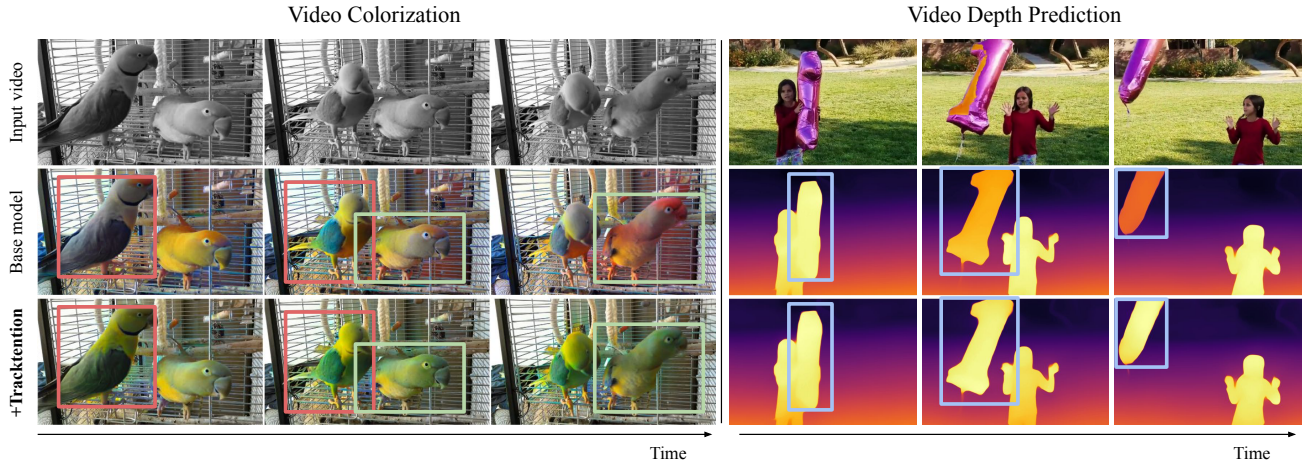


Figure 7. **Stabilization effect of our model on video prediction tasks:** *Left (Video Colorization):* The first row shows input grayscale video frames. The second row demonstrates the output of a frame-by-frame base model [22], producing inconsistent colors across frames (e.g., varying hues in pink and green boxes). The third row highlights our proposed Tracktention (+base model [22]), achieving consistent and stable colorization across frames. *Right (Video Depth Prediction):* The first row displays input video frames. The second row shows the depth predictions from a base model [62], which suffers from temporal instability. The third row presents Tracktention’s (+base model [62]) depth predictions, offering consistent and stable outputs over time.

## A. Design and Implementation Details

### A.1. Query Initialization and Point Tracking

In Tracktention, we use randomly initialized queries for the point tracking model. In Figure 8, we compare the results of two different query initialization strategies for object tracking in video: grid-initialized queries and random-initialized queries that we use. Queries (represented as larger dots with white edges) are seed coordinates in the video’s spatio-temporal space that are used to start the tracking process. Effective query initialization is crucial for maintaining complete and consistent coverage of objects throughout the video frames.

The top row illustrates the tracks obtained using a grid sampling strategy, where a uniform grid of queries is placed over the spatial dimensions of the first frame. While this method provides good initial coverage, it results in significant gaps in later frames due to motion and occlusion, leading to many areas being left untracked as the video progresses.

In contrast, the bottom row demonstrates tracks produced by our proposed random sampling method, where queries are initialized randomly in the spatio-temporal space. This approach results in more robust and complete tracking, as seen in the wider spatial distribution of tracks in the later frames.

This enhanced tracking coverage is advantageous for our tracking-based attention model, Tracktention, which depends on the completeness and density of query tracks to deliver comprehensive attention across the scene. By ensuring consistent and uniform tracking throughout the video, our random query initialization method enables Tracktention to more effectively focus on and process critical regions of interest, even in complex and dynamic scenarios. This leads to improved performance and robustness, as demonstrated in the Ablation Study presented in the main paper.

### A.2. Tracktention compared with Standard Attention Mechanisms

In Figure 9, we provide a conceptual overview comparing Tracktention to standard attention mechanisms for video processing. **Spatial-temporal attention** attends to all tokens across space and time, capturing comprehensive relationships but at a prohibitive computational cost. **Spatial attention** focuses only on spatial tokens within individual frames, ignoring temporal dependencies, while **temporal attention** processes temporal evolution at fixed spatial locations and fails when objects move across patches. While some methods combine spatial and temporal attention, attending to a different location in another frame requires traversing intermediate tokens implicitly, leading to indirect and inefficient attention pathways. These limitations

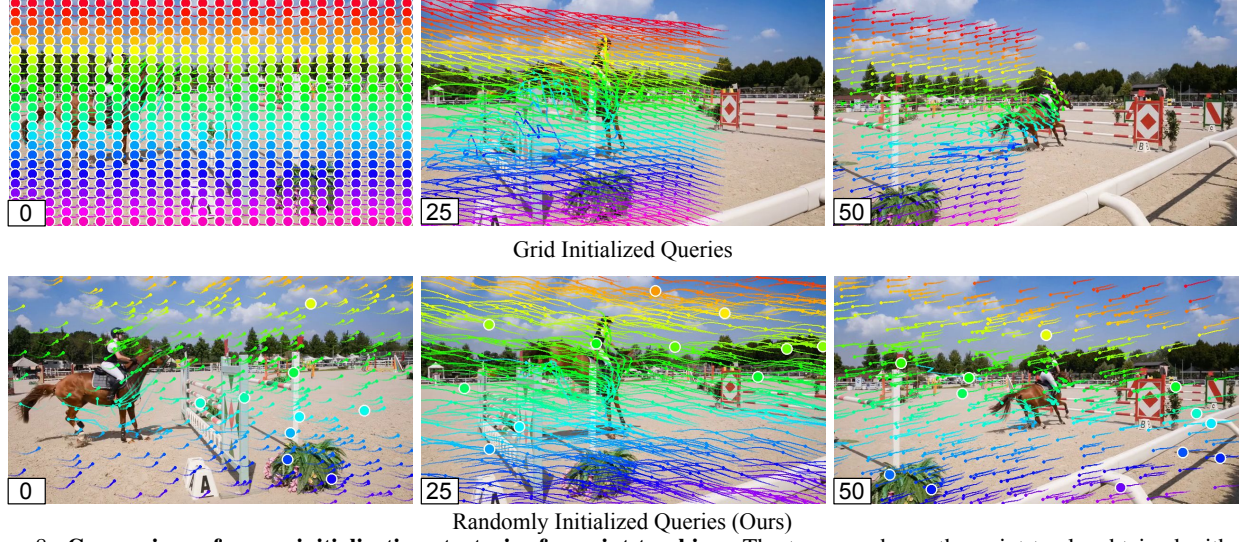


Figure 8. **Comparison of query initialization strategies for point tracking.** The top row shows the point tracks obtained with grid-initialized queries, which suffer from significant coverage loss in later frames. In contrast, the bottom row illustrates tracks obtained with our random initialization method, which maintains comprehensive coverage across the scene over time. Larger dots with white edges represent queries, which are seed coordinates used to initiate tracking in the video. Numbers at the bottom left of each frame indicate the frame index, highlighting the improved completeness of tracks produced by our approach, particularly in later frames.

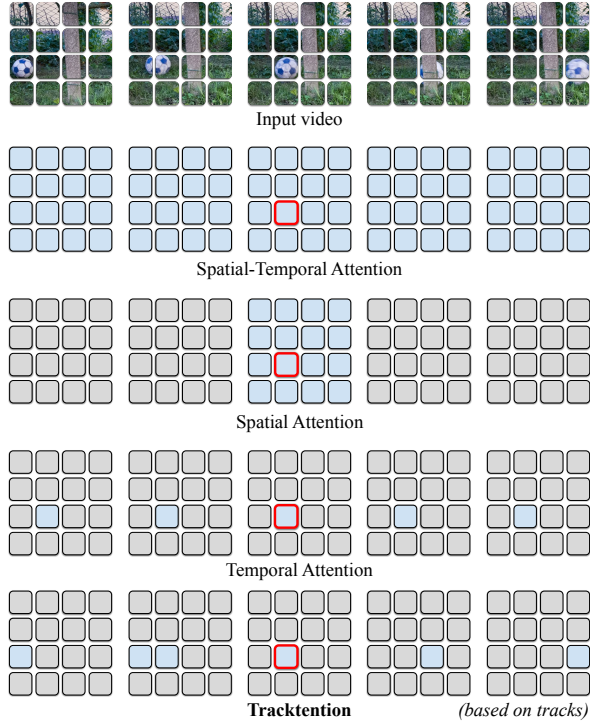


Figure 9. **Comparison of different attention mechanisms for video processing.** The red-bordered block represents the query token, while the blue-highlighted blocks indicate the range of tokens the mechanism attends to. See text for details.

hinder their ability to efficiently and effectively model dynamic video content.

Our proposed Tracktention addresses these challenges by leveraging point tracks to attend selectively to spatial-temporal tokens relevant to a query’s trajectory. Through our Attentional Sampling module, Tracktention handles cases where objects span multiple patches (*e.g.*, the second frame), ensuring fragmented representations do not disrupt attention. Additionally, it is robust to occlusion (*e.g.*, the fourth frame) by using a tracker [24] designed to handle occlusions. By combining computational efficiency, adaptability to motion, and robustness to occlusion, Tracktention significantly improves video processing in complex scenarios.

### A.2.1 Complexity Analysis

We analyze the computational complexity of our proposed Tracktention layer in comparison to standard attention mechanisms used in video processing. Let  $H$  and  $W$  denote the height and width of the frame features,  $T$  the number of frames (temporal dimension), and  $N$  the number of point tracks used in Tracktention, where  $N < HW$  and  $N \ll HWT$ .

**Spatio-temporal Attention:** Spatio-temporal attention attends to all tokens across both space and time, capturing comprehensive relationships but at a prohibitive computational cost:  $O((HWT)^2) = O(HWT \cdot HWT)$ .

This quadratic complexity over the total number of tokens  $HWT$  makes it computationally infeasible for larger



videos or higher resolutions.

**Spatial Attention:** Spatial attention operates on each frame independently, attending to all spatial tokens within a frame. The computational complexity for processing all frames is:  $O(T \cdot (HW)^2) = O(HWT \cdot HW)$ .

This is because, for each of the  $T$  frames, attention computations involve all pairs of spatial tokens, resulting in a quadratic complexity with respect to the number of spatial tokens  $HW$ .

**Temporal Attention:** Temporal attention focuses on the temporal evolution at fixed spatial locations. Each spatial position attends over the temporal dimension. The computational complexity is:  $O(HW \cdot T^2) = O(HWT \cdot T)$ .

Here, each of the  $HW$  spatial positions computes attention over all  $T$  temporal tokens at that position, leading to quadratic complexity with respect to the temporal length  $T$ .

**Tracktention Complexity:** Our proposed Tracktention leverages point tracks to attend selectively to relevant spatial-temporal tokens along a query’s trajectory. The computational complexity of Tracktention is:  $O(HWT \cdot N + T^2 \cdot N)$ .

The term  $O(HWT \cdot N)$  corresponds to the *Attentional Sampling* and *Attentional Splatting* process, where features are sampled along the point tracks from the video tokens.

The term  $O(T^2 \cdot N)$  arises from the *Temporal Transformer* operating over the point tracks, computing attention across time for each track.

**Complexity Comparison:** Since  $N$  is significantly smaller than the total number of tokens ( $N \ll HWT$ ). The complexity of Tracktention is substantially lower than that of spatio-temporal attention mechanisms.

Factorized spatial and temporal attention achieves better efficiency by separating spatial ( $O(T \times (HW)^2)$ ) and temporal ( $O(HW \times T^2)$ ) attention, resulting in a combined complexity of  $O(HWT \cdot (HW + T))$ . As  $N < (HW + T)$ , Tracktention achieves lower complexity by attending selectively to  $N$  point tracks, yielding an more efficient tool for video processing.

Furthermore, we note that the attentional sampling and splatting processes realistically focus on a local patch around each query. This allows us to further reduce the complexity of these processes from  $O(HWT \times N)$  to  $O(P^2 \times T \times N)$ , where  $P$  is a local patch size ( $P^2 \ll HW$ ). We leave this sparsity optimization for future work.

### A.3. Other Implementation Details

#### A.3.1 Video Depth Estimation Evaluation

**Data and evaluation metric.** For training, we use a combination of datasets containing both synthetic and real videos: ARKitScenes [2], ScanNet++ [65], TartanAir [57], PointOdyssey [68], DynamicReplica [23], and DL3DV [33], totaling 12,947 videos. All videos are

resized to have a short side of 336 pixels for computational efficiency. For evaluation, we use two benchmarks: RobustMVD [48], containing short clips of 8 frames with large motion, and the longer benchmark videos from DepthCrafter [20]. These evaluation datasets include a wide variety of scenes from KITTI [17], ScanNet [8], DTU [64], Tanks and Temples [28], ETH3D [47], Sintel [39], and Bonn RGB-D [41].

We use standard depth estimation metrics [62]: *Absolute Relative Difference* (AbsRel) — calculated as  $|\hat{d} - d|/d$ , where  $\hat{d}$  is the estimated depth and  $d$  is the true depth — and *Threshold Accuracy* ( $\delta_\tau$ ), the percentage of pixels satisfying  $\max(\frac{d}{\hat{d}}, \frac{\hat{d}}{d}) < \tau$ , with  $\tau$  set per the original benchmark. As in training, we calibrate each prediction to the ground truth using a scale and shift factor, shared across all frames in a video, before assessing a metric.

**Scale and shift ambiguity** Evaluating depth estimation in video sequences presents unique challenges due to the scale ambiguity inherent in monocular depth prediction, where predicted depths may differ from ground truth by a global scale and shift. Traditional methods [29, 38, 49, 59] often employ frame-wise evaluation, fitting a separate scale and shift for each frame independently:

$$\min_{s_i, t_i} \sum_{(x,y)} (D_{2,i}(x,y) - (s_i \cdot D_{1,i}(x,y) + t_i))^2,$$

where  $D_{1,i}$  is the predicted depth,  $D_{2,i}$  is the ground-truth depth for frame  $i$ , and  $s_i, t_i$  are the scale and shift for that frame. While this approach minimizes per-frame error, it can mask significant temporal inconsistencies, as it allows scale and shift to vary freely between frames, leading to flickering or unstable depth predictions in videos.

To address this issue, we adopt a video-wise evaluation method similar to DepthCrafter [20]. This approach enforces a single global scale and shift across the entire video sequence:

$$\min_{s,t} \sum_i \sum_{(x,y)} (D_{2,i}(x,y) - (s \cdot D_{1,i}(x,y) + t))^2.$$

By using consistent scaling factors  $s$  and  $t$  for all frames, the video-wise evaluation penalizes variations in predicted depth over time, providing a more accurate assessment of temporal consistency. This method highlights the model’s ability to maintain stable and coherent depth predictions across frames, which is crucial for applications requiring consistent video outputs.

#### A.3.2 Implementation Details for Video Colorization

**Evaluation metrics.** We assess the quality of the colorization using standard metrics: the *Fréchet Inception Distance* (FID), which evaluates how well the predicted colorization

matches the ground truth statistically in feature space; the *Colorfulness Score (CF)*, which quantifies color vibrancy; and the *Color Distribution Consistency (CDC)*, which measures the temporal consistency of the colorization. We also include the Peak Signal-to-Noise Ratio (PSNR), though it is generally acknowledged that this is a poor metric for evaluating colorization accurately [22].

**Integration implementation** We evaluate the effectiveness of the Tracktention layer in enhancing temporal consistency by integrating it into four image colorization models: CIC [66], IDC [67], ColorFormer [21], and DDColor [22].

**Integration with CIC and IDC** For the ConvNet-based architectures CIC and IDC, which feature encoder-decoder structures with downsampling, standard, and upsampling convolutional layers, we insert the Tracktention layer after the standard convolutional layers 5, 6, and 7. This integration allows temporal alignment at multiple stages of feature extraction, enhancing consistency across video frames.

**Integration with ColorFormer and DDColor** In ColorFormer, which utilizes a transformer backbone followed by a 4-layer U-Net decoder with residual connections, we integrate the Tracktention layer after the first three layers of the transformer backbone. Similarly, for DDColor, which employs a ConvNeXt [36] backbone with a U-Net decoder, we add the Tracktention layer after the first three layers of the backbone.

**Training Details** All models are trained using the loss functions from DDColor with the AdamW optimizer (initial learning rate  $1.6 \times 10^{-5}$ ). The learning rate is decayed using a MultiStepLR scheduler every 4k iterations starting from the 8k-th iteration, over a total of 40k iterations. Training is conducted on the YouTube-VIS [60] dataset using raw, unlabeled video data, with frames resized to  $256 \times 256$  pixels.

We do not employ temporal consistency losses such as flow warping losses used in prior works [31, 34]. By avoiding the computation of optical flow and associated warping operations, our training process remains efficient while achieving temporal consistency through the Tracktention layer integration. This shows that the Tracktention layer can effectively enhance temporal consistency in video colorization without relying on additional temporal loss functions, highlighting its flexibility and practicality.

## B. Additional Experimental Results

Here, we present additional experimental results and visual comparisons showing how the Tracktention layer enhances video consistency, stability, and accuracy, along with an analysis of input tracks’ influence.

### B.1. Influence of Input Tracks on Video Consistency

We analyzed how the selective use of specific tracks affects the temporal consistency of video colorization. By

selectively activating tracks corresponding to specific objects or regions, Tracktention guides its attention mechanism to maintain stable and consistent colorization for those targeted areas across video frames, while areas without active tracks may exhibit color inconsistencies. Figure 10 illustrates this behavior. In each case, activating tracks for a particular object (e.g., one bird, the front of the train, or the dog) results in consistent coloring for that object, while other objects without active tracks (e.g., the other bird, the back of the train, or the ball) show inconsistent coloring. This demonstrates that Tracktention can leverage selective tracks to ensure localized stability in the colorization process, even when other regions of the frame are affected.

### B.2. Additional Visual Examples

**Video Depth Estimation** Figures 11 and 12 present additional visual comparisons for video depth estimation. Similar to the results in the main paper, our model produces stable and accurate depth maps across all frames. The incorporation of the Tracktention layer enables precise temporal alignment of features, resulting in consistent and accurate depth estimation over time.

The state-of-the-art method, DepthCrafter exhibits significant errors in certain regions, particularly where complex motion or occlusions occur. DUST3R, which relies on implicit triangulation, struggles with dynamic content, leading to inaccuracies and temporal inconsistencies in the estimated depth maps.

Our base model, Depth Anything, is an image-based depth estimation model that processes frames independently. As a result, it shows inconsistent depth estimation across frames, with noticeable instability in the depth maps. By integrating the Tracktention layer into Depth Anything, we enhance temporal consistency, achieving results that are both accurate and stable throughout the video sequence.

**Automatic Video Colorization** Figures 13 and 14 provide more results on automatic video colorization. Consistent with observations in the main paper, the baseline method TCVC is unable to produce vibrant colors, resulting in desaturated and less realistic outputs.

Our base model, DDColor, performs frame-by-frame colorization, which leads to unstable and inconsistent color results. The absence of temporal coherence causes color flickering and discrepancies between frames, detracting from the overall visual quality of the video.

When we augment DDColor with our Tracktention layer, we observe a clear improvement in temporal consistency while retaining the original color vibrancy and realism. The Tracktention layer allows the model to attend to corresponding areas across time based on point tracks, ensuring smooth and coherent colorization throughout the video.



### **B.3. Video Demonstrations**

We also provide a supplementary video that explains our method and showcases the resulting video outputs. The video includes side-by-side comparisons of the base models before and after augmented with our Tracktention layer for both video depth estimation and automatic video colorization tasks. It highlights the temporal consistency and accuracy achieved by integrating the Tracktention layer into existing models. Please find the enclosed `supplementary_video.mp4` file for a visual demonstration of our method’s performance.

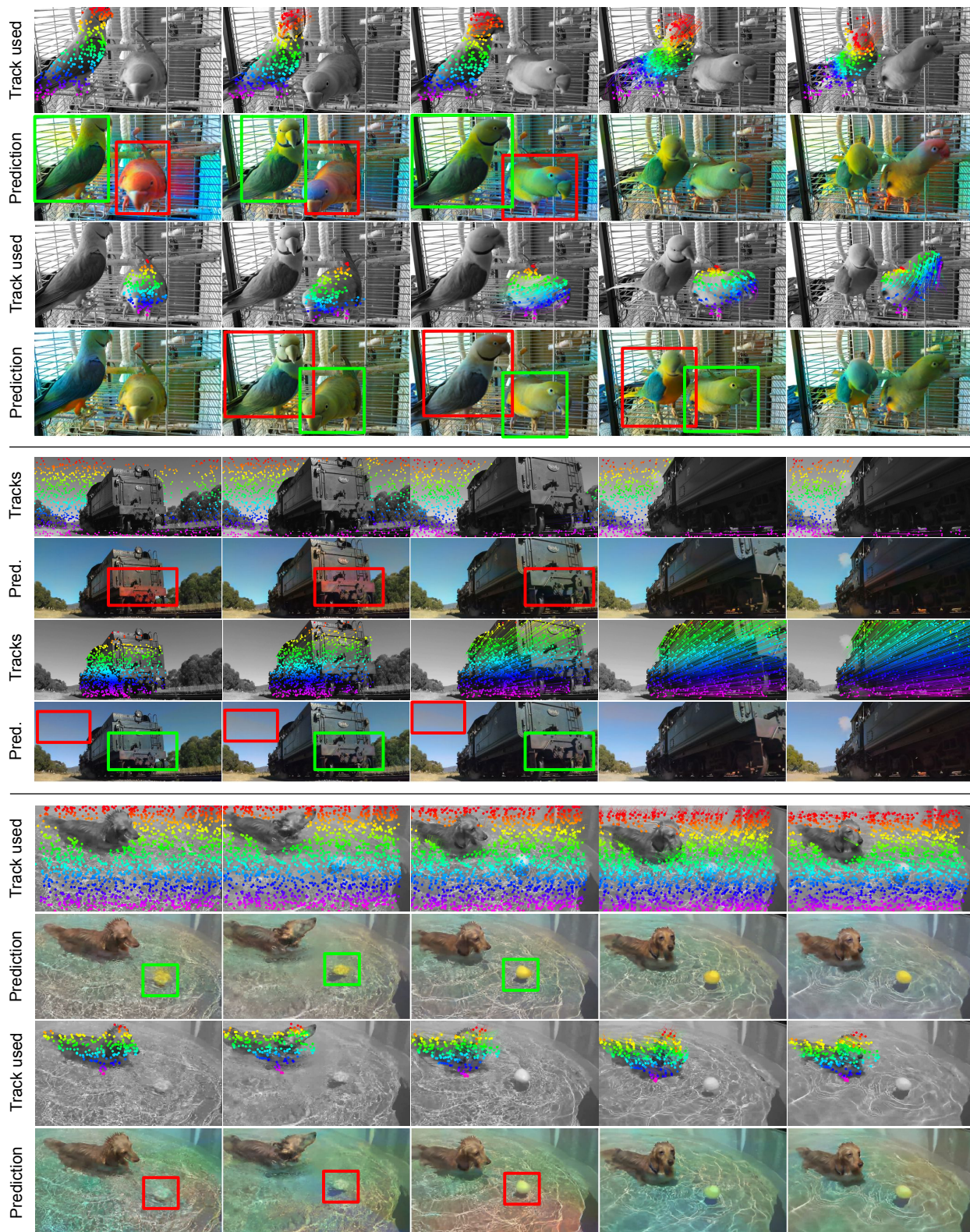


Figure 10. **The impact of selective tracks used on video colorization consistency.** In the top example, using tracks corresponding to the left bird (Row 1) results in consistent colorization of the left bird (Row 2, green box), while the right bird exhibits inconsistent colorization (red box). Conversely, using tracks for the right bird (Row 3) ensures stable colorization for the right bird (Row 4, green box), while the left bird becomes inconsistent (red box). Similar patterns are observed for the train and dog examples. Green boxes highlight regions with stable and consistent colors, while red boxes indicate inconsistent colorization.



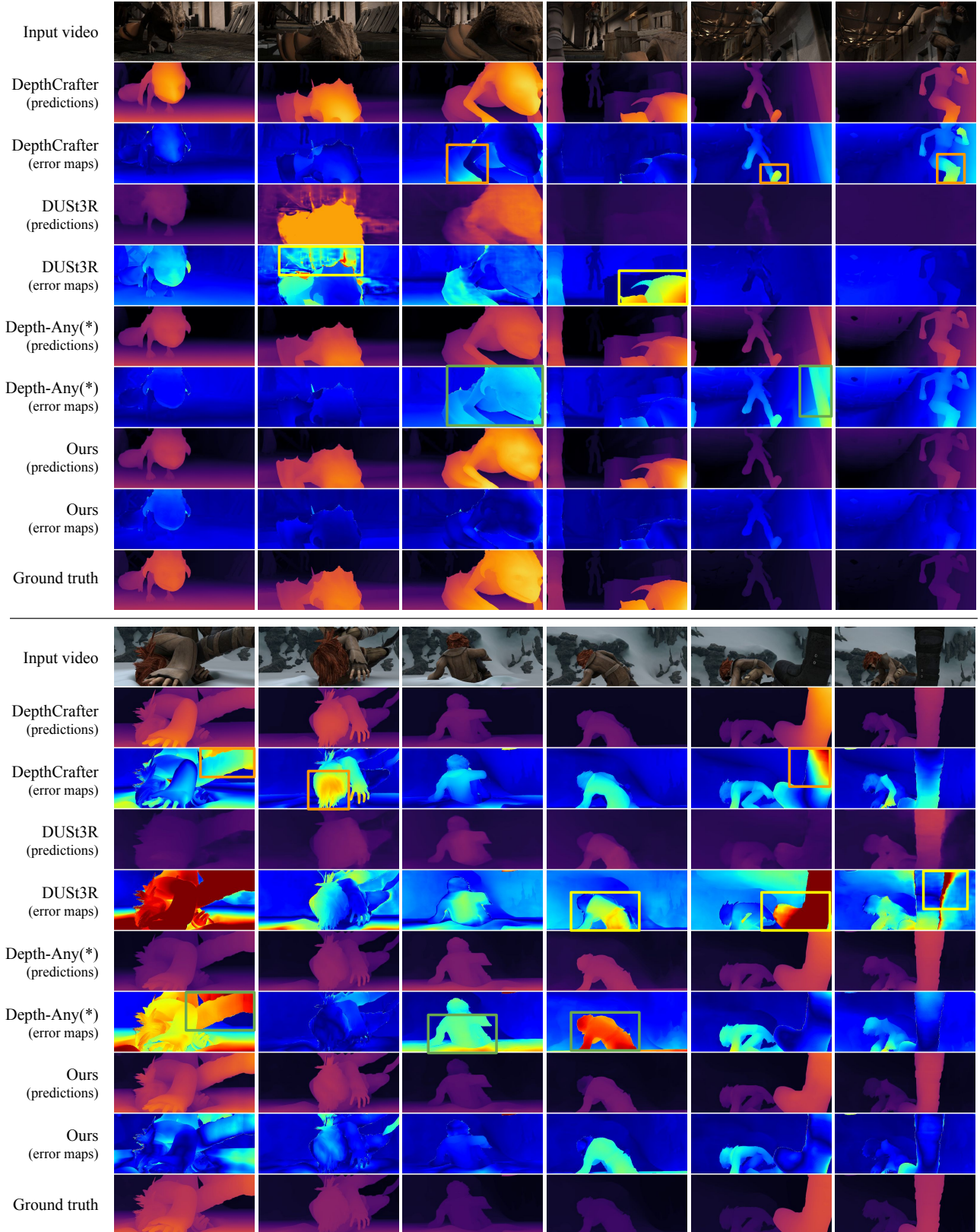


Figure 11. **Comparison of depth prediction results across different models:** DepthCrafter, DUST3R, Depth-Anything (marked with \* as the base model), and Ours (+Depth-Anything). Each row shows depth predictions, corresponding error maps, and the ground truth for the input video frames. Highlighted rectangles emphasize key issues in baseline methods: DepthCrafter exhibits significant errors in certain areas, DUST3R tends to fail in dynamic regions, and Depth-Anything produces flickering results, evident from inconsistent error patterns.



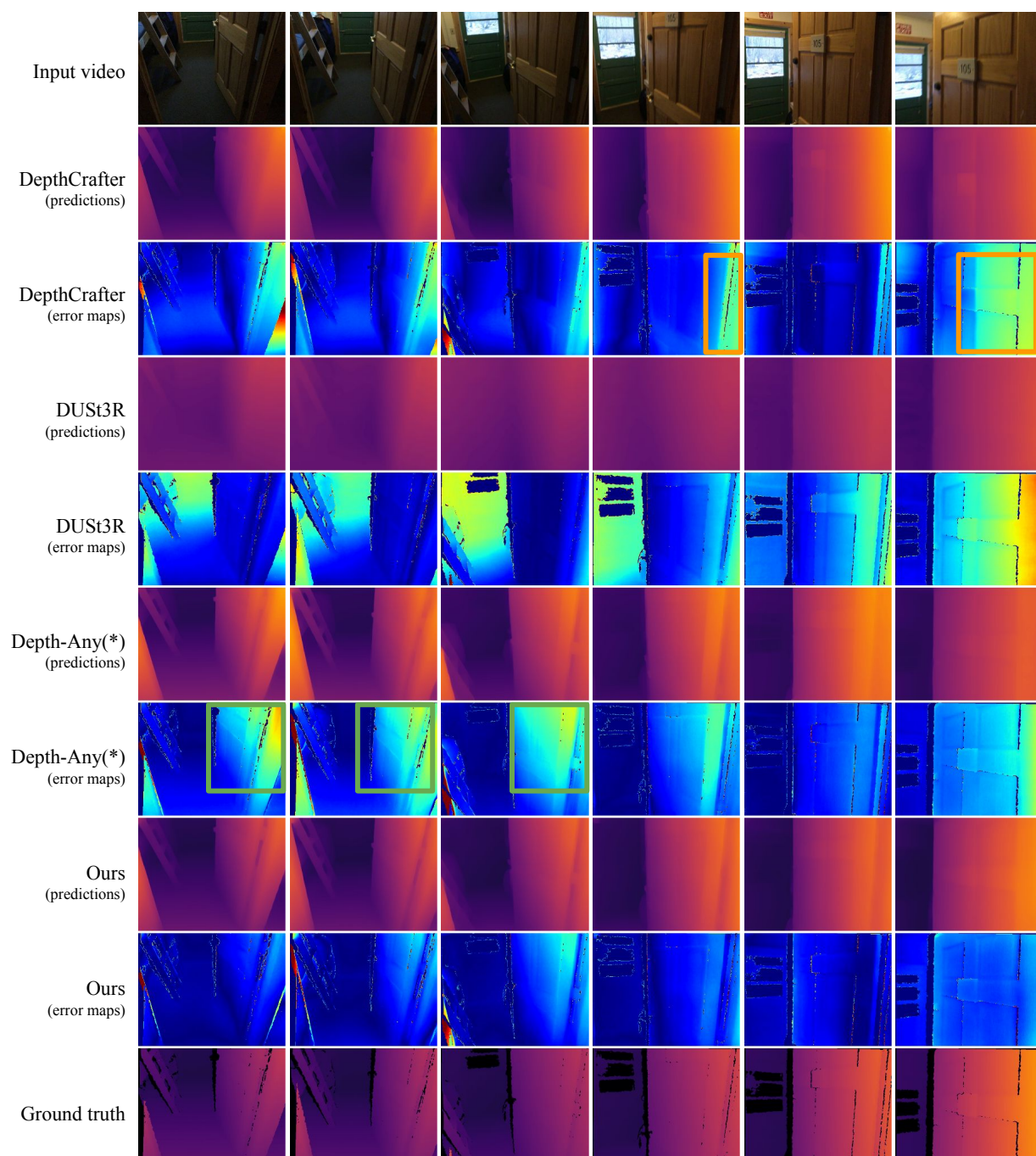


Figure 12. **Additional comparison of depth prediction results** across DepthCrafter, DUST3R, Depth-Anything (\*denotes the base model), and Ours (+Depth-Anything). The rows present depth predictions, their error maps, and ground truth for a different set of input video frames, illustrating consistency across varying scenes.

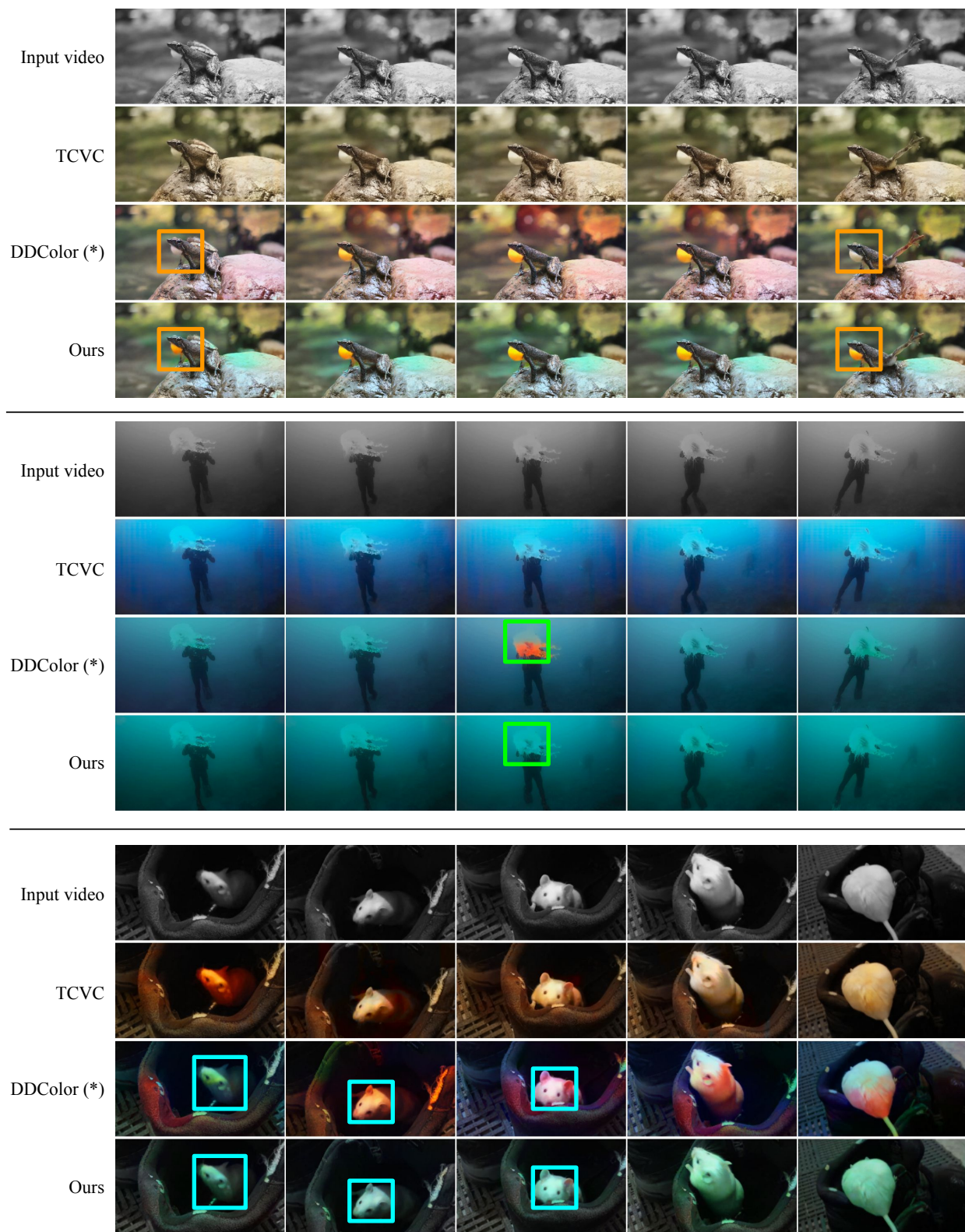


Figure 13. **Video colorization results** comparing TCVC, DDColor (\*denoting the base model), and Ours (+DDColor). The rows show the input grayscale video frames, followed by the colorized outputs from each method. Highlighted areas indicate inconsistencies in the base model (DDColor), which are resolved by our model, demonstrating its ability to produce consistent and accurate colorization.



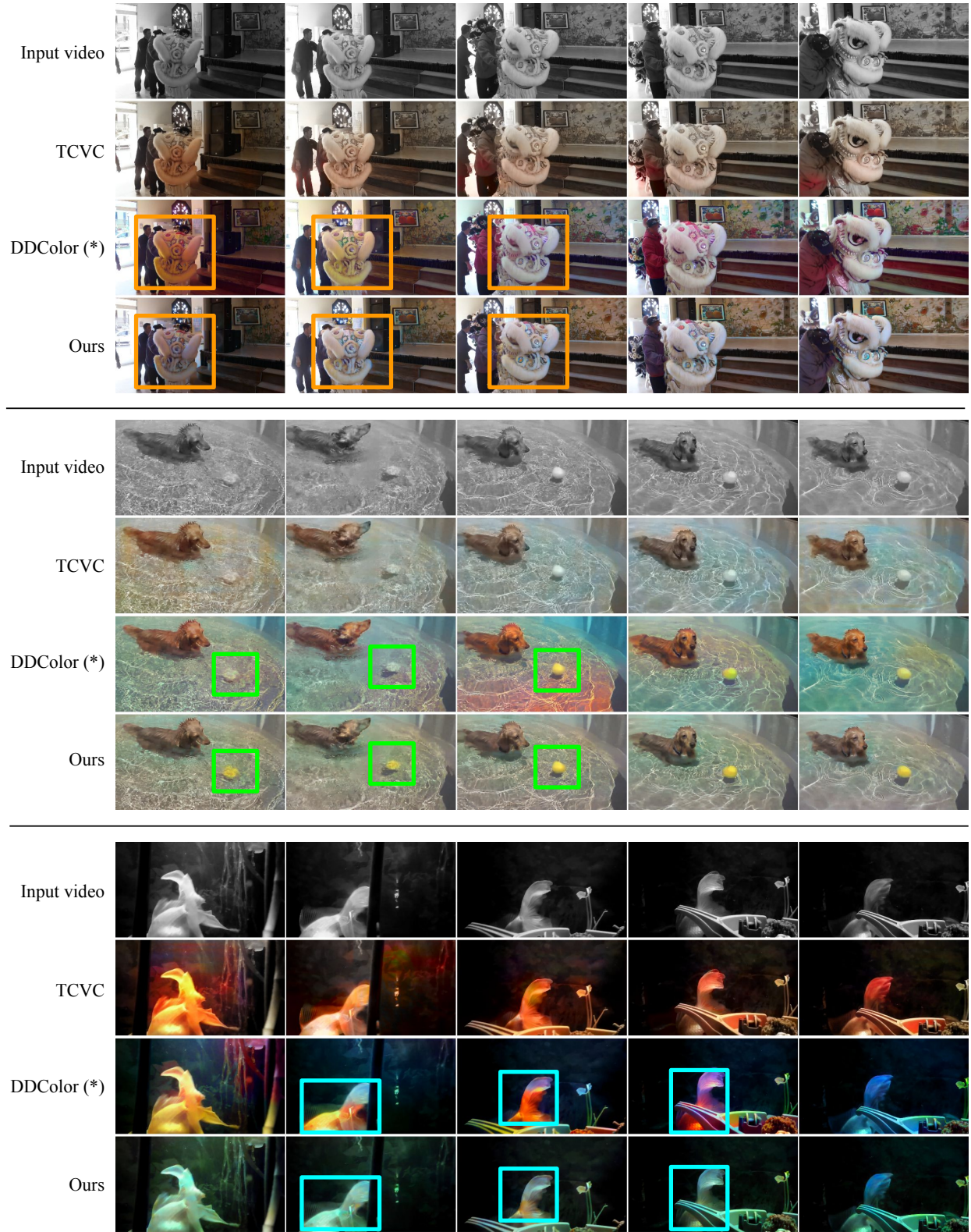


Figure 14. **Additional video colorization results** comparing TCVC, DDColor (\*denoting the base model), and Ours (+DDColor). The rows display the input grayscale video frames alongside the colorized outputs from each method. Highlighted areas pinpoint inconsistencies in the base model (DDColor), which are effectively resolved by our model, showcasing its improved consistency and color accuracy.