Future Optical Flow Prediction Improves Robot Control & Video Generation

Supplementary Material

Reproducibility Statement

To ensure the reproducibility of our results, we have made our code, trained model weights, and data configurations publicly available (see fofpred.github.io). Our method builds upon open-source pretrained models retrieved from the Hugging Face Hub. Detailed implementation specifics, hyperparameter settings, and experimental protocols are thoroughly documented in the main text and the Appendix.

Acknowledgments

We would like to thank the broader research team at Salesforce for support, feedback and guidance. In particular, we thank Shelby Heinecke, Juntao Tan, Srinath Meadusani, Eric Xu, Matthew Fernandez, Jim Jagielski, Joyce Zheng, and Jinxuan Xu for technical feedback and support; Jeanette Berberich and Jennifer McCallion for legal coordination; Janna Remperas, Ian Thomas, Mitra Mitchell, and Alex Dalton for logistical help and encouragement throughout the project.

Contributions

KR led the project by implementing the preliminary idea of language-driven OF prediction, building the codebase for experimentation, and performing most of the evaluations. HZ discussed all aspects of the project, contributed to several key design choices (on model architecture, OF calculation, video evaluations), and helped debug several technical issues. YF discussed multiple aspects of the robotic evaluation pipelines, helped implement the RoboTwin evaluations, and performed several robotic baseline evaluations. LY contributed to building the video evaluation pipeline, helped investigate our novelty against prior works, and discussed several aspects of our video generation pipeline. LX proposed using the unified VLM-Diffusion architecture, provided feedback on our data and training pipeline implementations, and discussed several early results. RX helped streamline early exploration of the idea, supported scaling up our training pipeline, and provided several critical feedback on project design. CX & SS provided the strategic vision for the project, oversaw the research environment, and shaped the high-level framing of the research problem. MR set the direction for robotic downstream tasks and discussed multiple aspects of the project idea, scope, and implementation. JN organized the overall project, set the research direction, and discussed all aspects of the project idea, scope, and implementation.

Appendix Contents

A. Additional Architectural Details

We provide additional details of our architecture in this section, focusing on the conditional input processing and the modifications made to the Diffusion Transformer (DiT). To recap, our overall architecture contains 3 main components: the VLM (3B parameters), VAE (83M parameters), and DiT (4B parameters). The DiT (diffusion transformer) module is the key component that we modify and train to construct our FOFPred model.

A.1. Conditional Input Processing

The core architecture utilizes two distinct conditional features: $\boldsymbol{f}_c$ (textual) and $\boldsymbol{f}_v$ (visual), which are passed through Multi-Layer Perceptrons (MLPs) to ensure dimensional compatibility with the Diffusion Transformer.

- **Textual Feature Projection:** The textual feature $\boldsymbol{f}_c$, obtained from the Qwen2.5-VL [4] Vision-Language Model (VLM), has an initial channel dimension of 2520. This feature is projected via an MLP layer to a common channel dimension D. The VLM input consists of the natural language caption c paired with the interleaved visual inputs $\boldsymbol{x}_{t-1}$ and $\boldsymbol{x}_t$.
- **Visual Feature Projection:** The visual feature $\boldsymbol{f}_v$, obtained from the Flux.1 VAE [46] encoder, has an initial channel dimension of 16. This feature is reshaped using 2×2 grid to obtain $64 = 16 \times 2 \times 2$ channel dimension vectors, followed by projected via a different MLP layer to the same common channel dimension D.

- **Initial Noise Vector:** The noise vector, sampled from the VAE latent space, is also projected into the DiT channel dimension using the same reshaping operation and visual feature project MLP layer.

We enforce the condition that the output dimensions of the two MLPs are equal to $D = 2520$ which is the DiT input channel dimension. These projected conditional features, $\hat{\boldsymbol{f}}_c$ and $\hat{\boldsymbol{f}}_v$, are simply appended to the input noise sequence of the DiT, providing comprehensive context for the diffusion process.

A.2. DiT Modifications for Temporal Modeling

Our Diffusion Transformer is based on the OmniGen architecture [94], adapted to explicitly handle the temporal nature of our sequence prediction task (future optical flow $\hat{\boldsymbol{y}}_{t+1:t+4}$ from inputs $\boldsymbol{x}_{t-1}, \boldsymbol{x}_t$).

Time-Aware 3D RoPE: We implemented a modified RoPE module to handle the required temporal dimension within the sequence. The original OmniGen RoPE provides 3-axis position encoding (L, H, W) for text length (L), image height (H), and image width (W). Our modification interprets these three axes as:

- **Axis 1 (Text/Image Shift):** Used to assign unique base position IDs for the text tokens and for each separate image (or image sequence). This axis is now leveraged to encode the **temporal offset** or frame index for the input sequences $(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t)$ and the latent noisy output $\boldsymbol{y}$.
- **Axis 2 and 3 (Image H, W):** Encode the spatial positions within each frame's patch grid.

Specifically, for multiple frames (e.g., $\boldsymbol{x}_{t-1}$ and $\boldsymbol{x}_t$ representations), the position shift variable is incremented after processing each frame. This ensures that while text tokens and $\boldsymbol{x}_{t-1}$ tokens receive their respective base position IDs, the tokens corresponding to $\boldsymbol{x}_t$ receive a unique, subsequent base position ID, which acts as a temporal index. Overall, this ensures that the tokens are timestamped with temporal position using RoPE, allowing the transformer to distinguish between and model the relationship across the consecutive time steps in both the input and output frame sequences.

Spatio-Temporal Attention: The DiT transformer blocks are updated to perform *full spatio-temporal attention* over the output tokens. This means the self-attention mechanism is configured to operate over the entire input sequence $\boldsymbol{f}_{in} = [\hat{\boldsymbol{f}}_c, \hat{\boldsymbol{f}}_v, \boldsymbol{f}_y]$, where $\boldsymbol{f}_y$ is the latent sequence of the noisy optical flow. This design choice allows the DiT to explicitly capture the motion dynamics and sequence dependencies required for accurate predictions of future optical flow frame sequences.

B. Optical Flow Representation

The visualization algorithm begins by converting the Cartesian optical flow field, denoted as $F \in \mathbb{R}^{2\times H\times W}$ with components f_x and f_y, into a polar representation. The magnitude of the flow is computed as $M = \sqrt{f_x^2 + f_y^2}$ and normalized by a scaling factor $\eta = 64.0$. This normalized magnitude, $\hat{M}$, is clamped to the range $[0, 1]$. The directional angle is derived using the four-quadrant inverse tangent function, $\phi = \arctan2(f_y, f_x)$, and is shifted by π to ensure the final angle θ lies within the interval $[0, 2\pi]$.

Subsequently, these polar coordinates are mapped to the HSV color space to generate an RGB image. The hue channel H is assigned the flow angle θ, encoding the direction of motion, while the saturation channel S is defined by the normalized magnitude $\hat{M}$. The value channel V is set to a constant maximum intensity of 1 (formulated in the algorithm as $\hat{M} + (1 - \hat{M})$) - this eliminates large color variances across consecutive frames due to outliers. Finally, the resulting HSV tensor is converted into RGB format using a differentiable color space transformation.

C. Optical Flow Calculation

The proposed method (Algorithm 1) compensates for camera motion within dense optical flow fields by leveraging the flow data itself to derive dense correspondences. Rather than relying on computationally expensive sparse feature extraction, we uniformly sample grid points from the source frame and project them into the target frame using the raw flow vectors $\mathbf{F}_{\text{raw}}$. These calculated point pairs serve as inputs for a RANSAC-based homography estimation, which computes a transformation matrix $\mathbf{H}$ representing the global camera motion. A dense "camera flow" field, $\mathbf{F}_{\text{cam}}$, is subsequently synthesized by applying $\mathbf{H}$ to the entire coordinate grid and calculating the displacement vectors. Finally, the object-centric motion is isolated by subtracting the estimated camera flow from the raw flow, such that $\mathbf{F}_{\text{obj}} = \mathbf{F}_{\text{raw}} - \mathbf{F}_{\text{cam}}$. A post-processing magnitude threshold is applied to $\mathbf{F}_{\text{obj}}$ to suppress residual noise and artifacts arising from imperfect alignment.

We reiterate that this one-time process is run offline. For our 500,000 training videos, we are able to complete processing using 4 A100 GPUs in roughly 30 hours. This timing also includes the motion-guided frame sampling (negligible time for this operation compared to relative flow calculation) that we describe in the next section.

D. Motion-Guided Frame Sampling

Training video prediction models on natural sequences requires careful data curation, as large portions of raw video may contain static scenes or imperceptible motion that contribute little to the learning process. To address this, we em-

Algorithm 1 Relative Optical Flow Calculation

Require:
$\mathbf{F}_{\text{raw}} \in \mathbb{R}^{B\times 2\times H\times W}$: Default optical flow tensors
τ_{ransac}: RANSAC reprojection threshold (default 5.0)
s: Sampling stride (default 8)
τ_{noise}: Post-compensation noise threshold (default 0.5)
Ensure:
$\mathbf{F}_{\text{comp}}$: Camera-motion compensated flow tensors

procedure CompensateFlow($\mathbf{F}_{\text{raw}}, s, \tau_{\text{ransac}}, \tau_{\text{noise}}$)
  $B, C, H, W \leftarrow \text{shape}(\mathbf{F}_{\text{raw}})$
  $\mathbf{F}_{\text{comp}} \leftarrow \text{List}()$
  $\mathcal{G} \leftarrow \{(x, y) \mid 0 \le x < W, 0 \le y < H\}$
  $\mathcal{S} \leftarrow \text{Sample}(\mathcal{G}, s)$; $\mathbf{p}_0 \leftarrow \text{Coords}(\mathcal{S})$
  for $i \leftarrow 0$ **to** $B - 1$ **do**
    $\mathbf{f}_i \leftarrow \mathbf{F}_{\text{raw}}[i]$
  Derive Correspondences
    $\mathbf{v} \leftarrow \text{Extract}(\mathbf{f}_i, \mathcal{S})$
    $\mathbf{p}_1 \leftarrow \mathbf{p}_0 + \mathbf{v}$
  Estimate Homography
    $\mathbf{H} \leftarrow \text{FindHomography}(\mathbf{p}_0, \mathbf{p}_1, \text{RANSAC}, \tau_{\text{ransac}})$
    if $\mathbf{H}$ is valid **then**
    Compute Camera Flow & Compensation
      $\mathcal{G}' \leftarrow \text{PerspectiveTransform}(\mathcal{G}, \mathbf{H})$
      $\mathbf{f}_{\text{cam}} \leftarrow \mathcal{G}' - \mathcal{G}$
      $\mathbf{f}_{\text{obj}} \leftarrow \mathbf{f}_i - \mathbf{f}_{\text{cam}}$
    Post-Compensation Thresholding
      if Threshold Enabled **then**
        $\mathbf{M} \leftarrow \|\mathbf{f}_{\text{obj}}\|_2$
        $\mathbf{f}_{\text{obj}}[\mathbf{M} < \tau_{\text{noise}}] \leftarrow 0$
      end if
      $\mathbf{f}_{\text{final}} \leftarrow \mathbf{f}_{\text{obj}}$
    else
      $\mathbf{f}_{\text{final}} \leftarrow \mathbf{f}_i$
    end if
    Append $\mathbf{f}_{\text{final}}$ to $\mathbf{F}_{\text{comp}}$
  end for
  return Stack($\mathbf{F}_{\text{comp}}$)
end procedure

Algorithm 2 Motion-Aware Frame Selection

Require: Video Sequence $V = \{I_1, I_2, \ldots, I_N\}$, Threshold τ, Top-Percentile k
Ensure: Selected Frame Indices S
Initialize selected set $S \leftarrow \emptyset$
for $t \leftarrow 1$ **to** $N - 1$ **do**
    ▷ Step 1: Spatial Downsampling for efficiency
  $I_t^{low}, I_{t+1}^{low} \leftarrow \text{Resize}(I_t, I_{t+1}, 32 \times 32)$
    ▷ Step 2: Fast Optical Flow Approximation
  $F_{LK} \leftarrow \text{LucasKanade}(I_t^{low}, I_{t+1}^{low})$
    ▷ Step 3: Compute Motion Proxy
  $M \leftarrow \|F_{LK}\|_2$ ▷ Calculate L2 norm of flow vectors
  $\mu_{proxy} \leftarrow \text{Percentile}(M, k)$ ▷ Extract top-k% value
    ▷ Step 4: Threshold Filtering
  if $\mu_{proxy} > \tau$ **then**
    $S \leftarrow S \cup \{t\}$
  end if
end for
return S

ploy a two-stage filtering pipeline, detailed in Algorithm 2, which acts as a computational gate to prioritize high-motion segments before generating expensive ground-truth labels. Calculating dense optical flow (e.g., using RAFT or PWC-Net) for every frame pair in a large-scale dataset is computationally prohibitive. To circumvent this bottleneck, we utilize a lightweight approximation strategy (Lines 4–6). We first spatially downsample all input frames to a resolution of 32×32 pixels. This reduction removes high-frequency textures and compression artifacts while preserving dominant structural motion. On these low-resolution pairs (I^{low}), we apply the Lucas-Kanade method [58]. Unlike deep learning approaches, Lucas-Kanade relies on local least-squares optimization, which converges rapidly on small spatial grids (32×32), allowing for high-throughput processing of millions of frames. To robustly distinguish meaningful scene activity from background noise, we derive a scalar motion proxy, μ_{proxy}, from the raw flow field (Lines 8–9). We compute the magnitude (L2 norm) of the flow vectors and select the top-k percentile value (k = 10) rather than the mean. This percentile-based approach ensures that the metric is driven by the moving objects within the scene, rather than being diluted by large static background regions. Frame pairs are included in the final training dataset only if this proxy value exceeds an empirical threshold τ (Lines 11–12), ensuring the model focuses on sequences with significant temporal dynamics.

The result of this motion-based frame filtering is the elimination of seemingly static regions of the video. In practice, we observe that certain static regions are removed while most frames in other motion-heavy regions remains. The resulting filtered videos still produce coherent video segments without breaking the natural temporal continuity of the motion. The low threshold for motion filtering (we use a 5 pixel length as threshold for 256×256 images) is crucial for this. We set this based on empirical observation over a randomly selected set of videos. We highlight again that tune the filtering process is important to ensure coherent frame sequences, instead of generating frame sequences that may contain large discontinuities.

E. OF Quality Evaluation

We quantitatively measure the quality of our predicted future optical flow and report these results in Table A.1. Our

Method	EPE ($\times 100$) ↓	CS ↑
LTM [69]	137	0.243
Im2Flow2Act [99]	78.3	0.0
SVD [11]	2.29	0.61
CogVideoX [101]	1.84	0.64
CogVideoX† [101]	7.64	0.52
FOFPred (Ours)	**1.48**	**0.74**

Table A.1. **OF Quality:** We report End Point Error (EPE) [5] & Cosine-Similarity (CS) between predicted & GT OF, calculated on SSv2 (val set). All models are trained to predict OF, except CogVideoX† (off-the-shelf weights) where OF is calculated from predicted RGB frames.

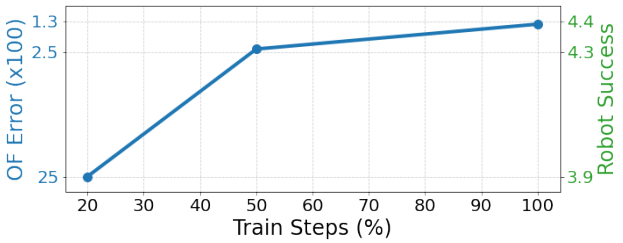

Figure A.1. **OF $\propto$ downstream:** Lower end-point-error [5] (SSv2) in predicted OF in pretraining leads to improved downstream robotic performance (AL metric on CALVIN).

model predicts better OF compared to prior work exploring language driven OF prediction [69, 99] as well as video diffusion networks adapted to predict OF (SVD [11] and CogVideoX [101]). We also explore a naive baseline of using off-the-shelf CogVideoX model (for RGB future frame prediction), and calculating OF from the prediction RGB frames. Our proposed FOFPred framework outperforms all of these baselines.

We also use different checkpoints of our FOFPred model from earlier steps of our training process (i.e. later checkpoints perform better OF prediction) to investigate the correlation between OF prediction quality and downstream task performance. We illustrate these results in Figure A.1. OF prediction error and robotic task performance (AL metric on CALVIN) are plotted. Note the separately scaled axes and flipped OF error axis, done for simpler visualization. We take this result as indication of a strong correlation between OF prediction quality and downstream robotic task performance.

F. Additional Ablations

We provide two new ablations on using dense optical flow as our representation and the significance of the motion signal passed into our robot action policy in Table A.2 and Table A.3 respectively.

Dense vs. Sparse Motion: We investigate the impact of dense vs. sparse motion representations on the downstream robot control task in Table A.2, reporting average length on the CALVIN benchmark. Training and evaluation settings are identical to our ablations in the main paper.

To isolate the value of our dense representation, we compare our method against two sparse variants: a naive baseline where our predicted future optical flows are subsampled to a 16×16 grid, and a re-implementation of the sparse trajectory model, ATM [92], trained on our dataset. The results demonstrate a distinct advantage for spatially dense motion information. Our default dense model achieves an average length of 4.39, significantly outperforming the ATM baseline (2.92) and the sub-sampled variant (1.24). This suggests that the fine-grained, pixel-level dynamics captured by dense optical flow are essential for precise robotic manipulation, whereas sparse representations fail to capture the complete motion context required for complex tasks.

Ablations on Motion Forecast Conditioning: We investigate the usefulness of motion forecasting as opposed to generic visual representations for our downstream robot control task in Table A.3. We compare our full framework against two baselines: one where motion input is entirely removed (note that our VLM input is a representation of both images and text), and another where the motion input is replaced by static visual embeddings from our VAE encoder. The results are decisive; the policy fails almost entirely without motion input (0.02) and shows only marginal improvement when provided with the static visual features (0.52). In contrast, conditioning on our predicted future optical flow embeddings yields a score of 4.39. This confirms that the predicted optical flow provides unique, critical dynamic information that static visual representations cannot substitute.

Video Generation Upper Bound: We explore a possible upper bound for our video generation results in Table 3 which uses optical flow calculated from each ground truth (GT) video in the test set=. Results in Table A.4 indicate a significant gap in the upper bound compared to both baselines and our method. Given the multi-modal output nature of OF prediction from a single frame, this upper bound should be viewed cautiously because the oracle OF (calculated from GT video) has the same mode as GT video. Predicted flows can be plausible or even optimal yet score lower simply by following a different valid mode.

G. Limitations and Future Work

Our FOFPred model currently contains several limitations. It is sensitive to the text prompt (i.e. slightly different text prompts could lead to wrong predictions). For example, changing a text "moving from right to left" $\rightarrow$ "moving left" results in wrong predictions for some samples. In addition,

Method	Motion	Avg Len ↑
Naive Baseline	Sparse	1.24 (-3.15)
ATM	Sparse	2.92 (-1.47)
Ours (default)	Dense	4.39

Table A.2. **Dense vs. Sparse Motion:** We report the average length metric (Avg Len) on CALVIN benchmark. We demonstrate the significance of our dense motion representation. The first row uses a naive sub-sampling of our predicted future optical flows to obtain motion for points on a uniform 16×16 grid on the image. This variant uses training and inference identical to ours. The second row re-implements the ATM [92] approach with training on our same data but inference similar to the original work. We highlight the clear improvements arising from our proposed method.

Method	Motion Input	Avg Len ↑
No motion input	✗	0.02 (-4.37)
Static visual input	✗	0.52 (-3.87)
Ours (default)	✓	4.39

Table A.3. **Motion Input Ablation:** We report the average length metric (Avg Len) on CALVIN benchmark. For our robot control extension, we ablate the motion input under two settings. First (row 1) we remove the motion input leaving only the state and text goal inputs. Next (row 2) we replace the motion input with an embedding (from our VAE) of the visual input. Our motion inputs in the form of future optical flow clearly leads to a performance improvement.

Method	FVD ↓	KVD ↓	LPIPS ↓
CogVideoX	78.47	12.46	30.3
Ours	75.39	11.38	28.5
Upper Bound	17.58	1.92	22.5

Table A.4. **Upper bound for downstream video generation**: Large gap since upper-bound uses GT videos for OF which has same mode in the the multi-modal output space.

the current FOFPred is a relatively large model (∼7B parameters in total) that is expensive to deploy in a real-time setting and requires considerable compute (at least 24GB of GPU) for inference.

On the other hand, our model captures diversity of motion patterns quite well and generates meaningful future optical flow in as less as 1 reverse diffusion (i.e. denoising) iteration. Across different seeds for the same frame-caption pair, our model generates a diverse distribution of mostly meaningful future optical flow: we attribute this strength to the diffusion based training. In Figure A.2, we explore the diversity of our FOFPred base model predictions, highlighting limitations as well. Across different seeds, we notice

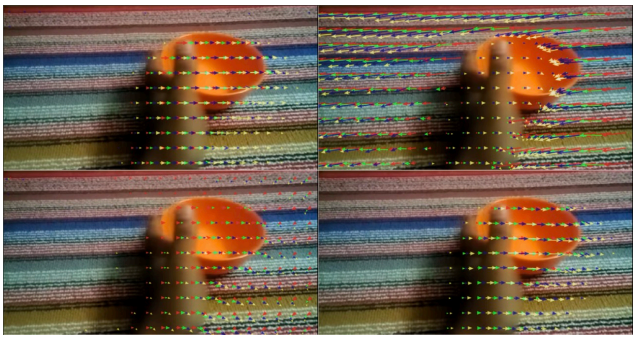

Figure A.2. **Sensitivity to seed:** We visualize 4 FOFPred predictions for the same image and prompt, "`Moving the bowl from left to right`", but using 4 different starting noise vectors for the reverse diffusion process. Notice how the upper-right corner conflates the object motion with a camera motion instead; however this camera motion does correspond to the object motion described in the provided prompt. In the lower left image, in addition to the desired object motion, we again observe a slight amount of corresponding camera motion.

unexpected camera motion sometimes. While often relevant to the provides motion prompt, this does not explicit capture the object motions that we desire from our model.

On fast convergence (during reverse diffusion), we note how our model is not distilled or trained specifically for this purpose; RGB image generation with identical architecture and training pipeline usually requires at least 20 sampling steps for meaningful generation. We hypothesize that optical flow is a less complex distribution compared to natural images (i.e. lies on a lower dimensional manifold and contains less high frequency information such as textures or patterns): this allows even a single sampling step of reverse diffusion to generate meaningful outputs.

In future work, we plan to explore training data augmentation with automated text-label re-phrasing and model distillation into light-weight architectures to address our key limitations. We also aim to further investigate and quantify the diversity aspect of FOFPred, analyze the fast convergence aspect, and explore possibility of real-time inference.

H. Visualizations

We first visualize predicted optical flow from our base FOFPred for several selected image pairs in Figure A.3. We select image pairs to highlight the *language-driven* future optical flow generation capability of our FOFPred framework.

We next visualize some videos generated with our framework for pairs of first frames and motion-focused captions obtained from the SSv2 dataset. Figure A.4 presents this qualitative comparison between the ground truth (GT) video, our T2V baseline from CogVideoX [101], and our

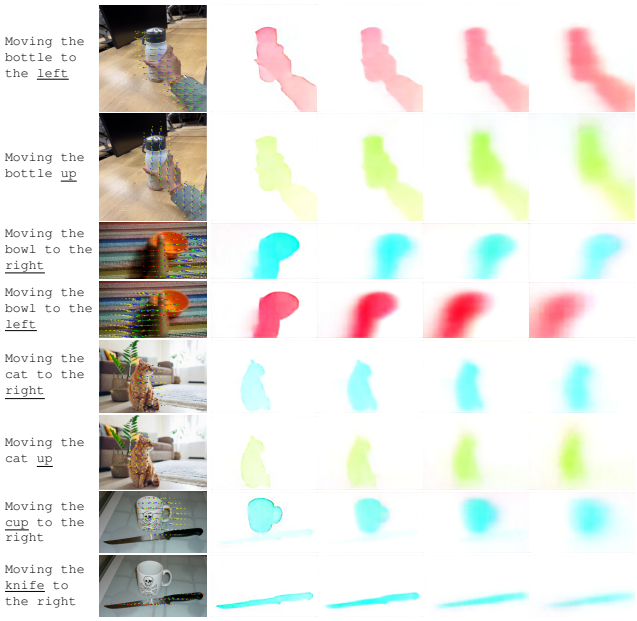

Figure A.3. **OF Quality:** We visualize our predicted future OF (4 timesteps) as sparse arrows (left: color for timestep) & raw prediction (right: color for direction). We use pairs of the same image with different prompts. Best viewed zoomed.

framework, FOFPred[1]. The samples are selected from the SSv2 validation split to demonstrate generation capabilities conditioned on the first frame (outlined in green) and a specific text goal. As observed across the samples, FOFPred consistently exhibits superior motion adherence compared to the baseline, which frequently struggles to generate significant movement or directionally accurate dynamics (e.g., the "Moving glue stick" and "Moving cycle" examples). While FOFPred successfully executes the semantic requirements of the text prompts, we also visualize a partial failure case in the second row ("Pulling toy car"). In this instance, while our model correctly adheres to the motion instruction, our extended pipeline exhibits a trade-off in visual fidelity, resulting in slight distortion of the object's appearance compared to the baseline. However, the baseline in this case moves the car in the wrong direction.

For more visualizations of our proposed FOFPred framework, checkout our website FOFPred.github.io.

[1]In our work, we explore Text-to-Video (T2V) generation using a two-stage pipeline that connects FOFPred with the existing video synthesis model Go-with-the-Flow (GWTF) [13]. GWTF is built on top of CogVideoX [101] and extends it to additionally accept a user-provided motion prompt as input.

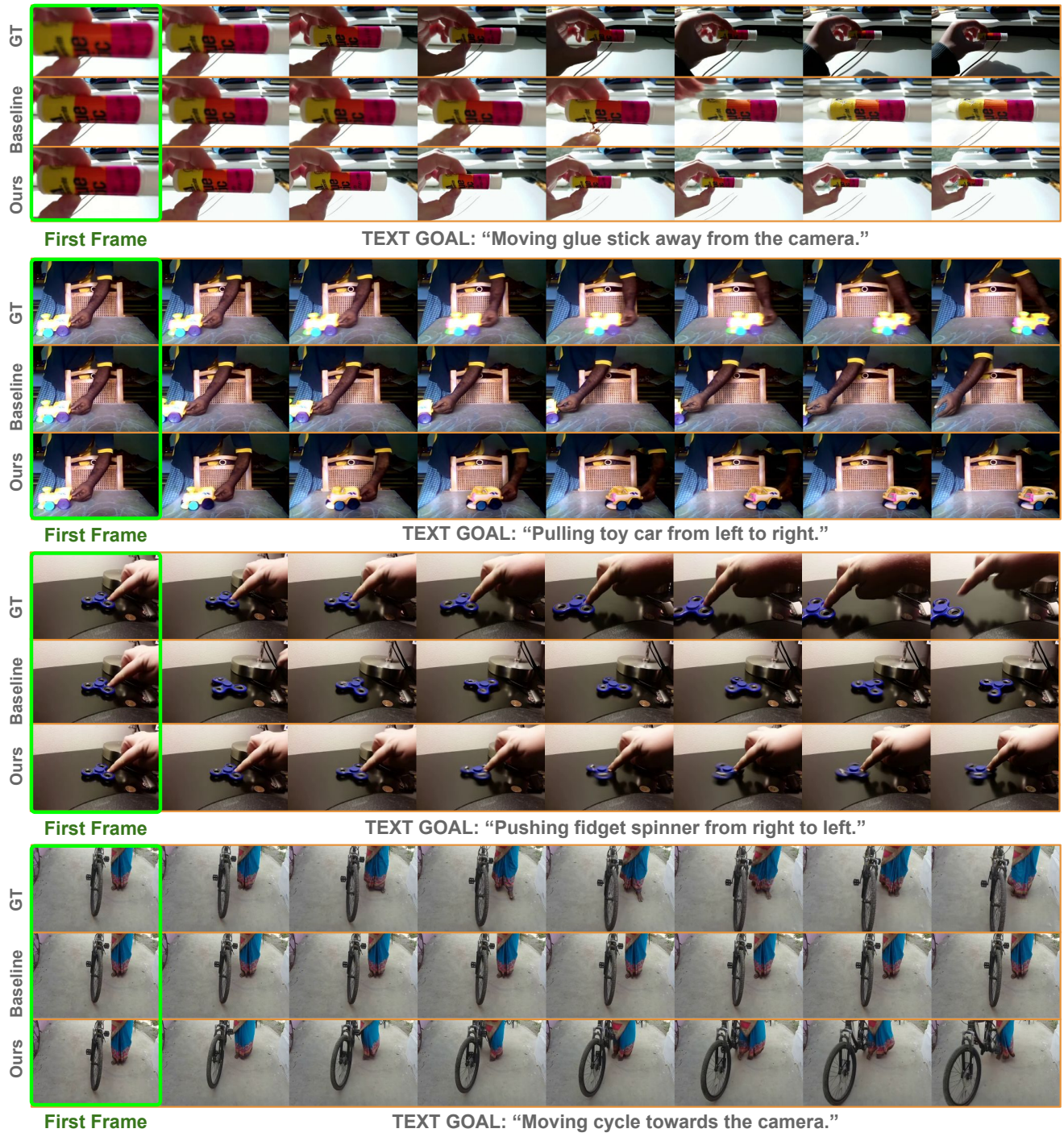

Figure A.4. **Visualization of success and failure cases for Text-to-Video (T2V) generation:** We visualize some success and failure cases for our framework over the baseline, CogVideoX [101]. Examples are drawn from the SSv2 validation split. We note that our method consistently improves motion adherence over the baseline. However, in some cases our framework distorts the visual appearance of objects although they undergo correct movement (e.g., see "toy car" in Row 2). Checkout our `FOFPred.github.io` for more visualizations.