TDFLOW: EFFICIENT AND ACCURATE OPTICAL FLOW ESTIMATION USING TRUNCATED DIFFUSION MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models have recently emerged as a powerful generative technique for diverse tasks in computer vision and robotic policy learning. Their extension to optical flow estimation represents a promising direction, enabling more accurate motion estimation for robot localization. Nevertheless, the heavy computational cost arising from the large number of denoising steps in vanilla diffusion policies poses a major obstacle to practical efficiency and real-world applications. To address this challenge, we propose a novel *truncated diffusion policy*, named TDFlow, that leverages initial flow estimates as anchors and shortens the diffusion steps by learning the denoising process directly from an anchored Gaussian distribution to the flow output, rather than the original pure Gaussian distributions. Furthermore, we design a *Conditional Feature-Space Warping Diffusion Decoder* for efficient flow refinement, conditioned by the scene context and flow anchors. Compared with existing approaches, our TDFlow achieves state-of-the-art performance on three optical flow datasets: MPI Sintel, Spring, and KITTI benchmarks, and further demonstrates a superior zero-shot generalization property on KITTI. Due to our designed truncated policy, our model requires only one twentieth of the denoising steps compared to the vanilla diffusion policy, while running up to $4.2\times$ faster in the reported implementation-level comparison with other diffusion-based methods.

1 INTRODUCTION

Optical flow estimation, which computes a dense, per-pixel 2D displacement field between successive frames, remains a cornerstone low-level vision task. Accurate flow fields can significantly support various high-level applications, including scene reconstruction (Ma et al., 2022; Zuo & Deng, 2023), video-based action recognition (Sun et al., 2018b; Piergiovanni & Ryoo, 2019), video frame interpolation (Xu et al., 2019; Liu et al., 2020), and autonomous driving (Menze & Geiger, 2015; Janai et al., 2020).

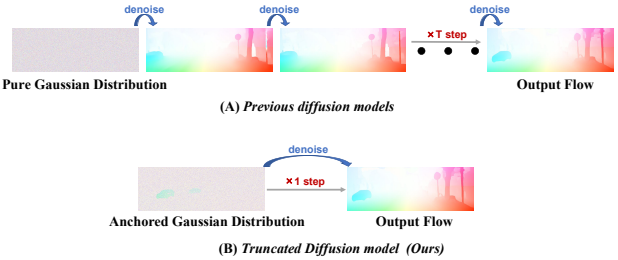
With the development of deep learning, research for optical flow estimation witnesses a transition from discriminative methods to generative ones. Early approaches typically formulate the task as a discriminative regression problem (Hur & Roth, 2019; Sun et al., 2018a; Teed & Deng, 2020; Zhao et al., 2020a;b). Following the pioneering work of FlowNet (Dosovitskiy et al., 2015), recent research has focused on improving feature encoders to extract more discriminative representations (Deng et al., 2023; Xu et al., 2022; Zhao et al., 2022), and strengthening decoders with recurrent updates (Teed & Deng, 2020) or attention-based aggregation (Jiang et al., 2021; Luo et al., 2022a; Huang et al., 2022). However, discriminative optical flow methods commonly suffer from two inherent limitations. First, by regressing a single flow field, they neglect the underlying uncertainty and probabilistic nature of optical flow, which are fragile under complex dynamics. Second, previous discriminative methods struggle with real-world degradations like occlusions, motion blur, and illumination changes, which corrupt feature matching and degrade performance.

In contrast, recent generative approaches (Saxena et al., 2023; Luo et al., 2024) learn a probabilistic joint distribution over flow fields, naturally modeling uncertainty and multiple motion hypotheses, thereby offering greater robustness in challenging scenarios (Luo et al., 2024). Although Saxena et al. (2023) first introduce a diffusion-based generative approach to optical flow, their use of a generic diffusion framework incurs a high inference cost and lacks flow-specific design. To overcome these limitations, FlowDiffuser (Luo et al., 2024) designs a task-specific diffusion process: it corrupts the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) **Truncated diffusion policy.** We inject limited noise into the anchor flow to form an anchored Gaussian distribution, enabling one-step denoising.



(b) **Diffusion-policy comparison.** FlowDiffuser samples flow from a standard Gaussian and applies T denoising steps. Our anchored diffusion starts around a flow estimate and recovers the target in a single step.

Figure 1: **Comparison of diffusion policies for optical flow.** Left: illustration of the proposed truncated diffusion policy. Right: comparison between FlowDiffuser and our Anchored Diffusion.

ground-truth flow with Gaussian noise and learns to denoise it iteratively. However, two critical drawbacks still remain: (1) The multi-step DDIM (Song et al., 2021a) schedule incurs substantial inference latency (see Table 1), impeding real-time deployment; and (2) Early diffusion iterations, seeded from pure noise, fail to recover high-frequency motion details essential for precise flow estimation as shown in Fig. 1b. These issues not only inflate computation but also contradict the fine-grained accuracy demands of optical flow. This raises a fundamental question: *Can we design a generative framework that retains the robustness of diffusion models while meeting the efficiency and detail-sensitivity requirements of optical flow estimation?*

To address the aforementioned drawbacks, we propose **TDFlow**, the first truncated denoising diffusion framework enabling fast optical flow estimation. Optical flow commonly exhibits structured, image-dependent patterns, while a standard diffusion model starting from Gaussian noise requires multiple time-consuming denoising steps to recover these patterns. To take advantage of more informative initial priors, we propose an *anchored diffusion policy* by applying only a small Gaussian perturbation centered on the predicted initial flow anchors. Specifically, it is implemented by truncating the diffusion schedule by introducing a small portion of Gaussian noise around the prior anchors, as shown in Fig. 1a. By harnessing the multi-mode distributional expressivity of the diffusion model, the proposed truncated diffusion policy effectively spans the potential flow manifold, where the required denoising steps are reduced from 20 in a vanilla diffusion policy to just 1. This enables a substantial speedup for the optical flow estimation.

Table 1: **Efficiency comparison of diffusion modules.** RTX 4090, batch 1, 376 × 1248 inputs. Implementation-level measurements.

Metric	FlowDiffuser	TDFlow
Denoising decoder	Cost volume	Warping
Feature resolution	1/8	1/2
Denoising updates	4	1
One-step latency	53 ms	**9 ms**
Diffusion latency	210 ms	**9 ms**
Total latency	290 ms	**69 ms**
Peak inference memory	2.924 GB	**1.210 GB**

To enhance the interaction with conditional scene context, we also propose a Conditional Feature-Space Warping Diffusion Decoder conditioned on scene context and flow anchors. Our proposed TDFlow formulates optical flow estimation as a generative diffusion process – By anchoring the stochastic dynamics to predicted flow priors, this anchored diffusion encourages exploration of diverse latent paths, enhancing generalization to unseen motion patterns as in Table 3 (b). Furthermore, our proposed truncated diffusion policy has a strong compatibility when integrated into existing methods, leading to a substantial performance improvement in Table 4.

The main contributions of this work are:

- We design a novel truncated diffusion policy that substantially reduces the computational burden of directly applying a vanilla diffusion framework to optical flow estimation.

- We design a feature-warping-based diffusion decoder that efficiently fuses conditional scene context with anchor flow estimates for more precise motion refinement.
- Our TDFlow delivers a strong accuracy–efficiency trade-off across MPI Sintel, Spring, and KITTI benchmarks. It achieves 0.96/1.84 EPE on Sintel Clean/Final, 3.283 1px on Spring, and 3.34 Fl-all on KITTI. Meanwhile, TDFlow maintains high efficiency, reducing the reported end-to-end runtime from 290 ms to 69 ms on one NVIDIA RTX 4090 GPU in the implementation-level comparison with FlowDiffuser.

2 Related Work

2.1 Optical Flow Estimation

Traditional optical flow methods view optical flow estimation as a global optimization that maximizes pixel-wise visual similarity via coarse-to-fine warping schemes. This approach is theoretically grounded by Brox et al. (2004) and exemplified in classic works on robust variational and multigrid formulations (Zach et al., 2007; Chen & Koltun, 2016). In recent years, deep learning has revolutionized the field (Sun et al., 2018a; Zhao et al., 2020b; Hui et al., 2018; Teed & Deng, 2020; Sui et al., 2022; Sun et al., 2022; Deng et al., 2023; Huang et al., 2022; Shi et al., 2023; Weinzaepfel et al., 2022; 2023; Xu et al., 2022; 2023; Leroy et al., 2023; Saxena et al., 2023; Jahedi et al., 2024; 2023; Luo et al., 2022a; Zheng et al., 2022; Zhao et al., 2022; Luo et al., 2023; Jung et al., 2023; Luo et al., 2024; Zhou et al., 2024; Morimitsu et al., 2025), which can be grouped into two paradigms: direct regression models that infer the flow field in a single pass (Dosovitskiy et al., 2015; Weinzaepfel et al., 2022; 2023; Saxena et al., 2023; Leroy et al., 2023; Xu et al., 2022) and iterative refinement methods that progressively update an initial estimate, often leveraging compact cost volumes to handle large displacements, resulting in superior accuracy (Sun et al., 2018a; Teed & Deng, 2020; Huang et al., 2022; Shi et al., 2023; Luo et al., 2024; Morimitsu et al., 2025; Zhou et al., 2024).

Building on these two foundational paradigms, recent works have introduced a variety of complementary mechanisms to further enhance flow estimation. Graph-based reasoning frameworks (Luo et al., 2022b) explicitly model spatial relationships between pixel regions, while attention-driven modules (Jiang et al., 2021; Luo et al., 2022a) adaptively weight features to capture long-range context. Holistic motion aggregation techniques (Jiang et al., 2021; Luo et al., 2022b) further integrate multi-scale cues for global consistency. Recently, Saxena et al. (2023) repurposed standard diffusion processes for optical flow generation, mitigating problems such as neglected uncertainty modeling in discriminative models but introducing substantial computational overhead. To address the trade-off between performance and efficiency, FlowDiffuser (Luo et al., 2024) implements a diffusion architecture tailored specifically to flow tasks, but still requires multiple denoising steps and iterative refinements, which introduce heavy computational consumption and slow inference speed as shown in Table 1.

2.2 Diffusion Policy in Computer Vision

Diffusion-based methods have become ubiquitous in many fields, including image synthesis (Yao & Wang, 2025; Yao et al., 2024; Rombach et al., 2022; Song et al., 2021b; Zhu et al., 2024), robot policy learning (Chi et al., 2023; Janner et al., 2022), etc. Regarding image synthesis, DDIM (Song et al., 2021a) extends DDPM (Ho et al., 2020) by defining a non-Markovian sampling process that yields high-fidelity outputs with far fewer steps. Flow matching techniques (Lipman et al., 2023; Liu et al., 2022) streamline generation by learning continuous transport maps directly. TDPM (Zheng et al., 2023) accelerates inference by initializing denoising from an implicit intermediate distribution instead of pure Gaussian noise. For the robotic policy learning, DiffusionPolicy (Chi et al., 2023) reveals diffusion models' capacity to represent multimodal, high-dimensional action distributions in robotic control. A range of follow-up studies have further applied diffusion frameworks across tasks such as stationary manipulation (Ze et al., 2024), autonomous navigation (Sridhar et al., 2024), and quadruped locomotion (Stamatopoulou et al., 2024). However, transferring a vanilla diffusion policy to optical flow estimation introduces stringent requirements for rapid inference and dynamic scene adaptation. In this work, we propose a novel truncated diffusion policy to address these challenges. Unlike prior diffusion-based approaches (Saxena et al., 2023; Luo et al., 2024) that initialize reverse

inference from pure Gaussian noise, our method employs a truncated diffusion policy and an efficient diffusion decoder to deliver high-speed, accurate optical flow estimation.

3 METHOD

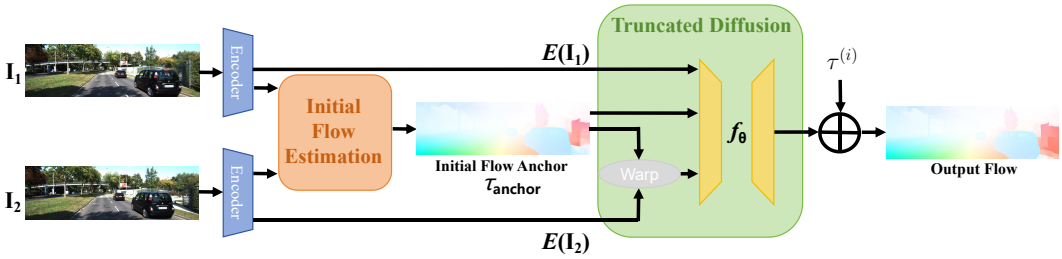


Figure 2: **The overall architecture of our method.** TDFlow begins by extracting features and producing an initial flow anchor τ_{anchor} using the WAFT-style recurrent update module (Wang & Deng, 2025). These anchors are then fed into a conditional feature-space warping diffusion decoder, which performs a single-step denoising pass to yield the output flow.

We first review optical flow estimation and conditional diffusion in Appendix B, then discuss the efficiency bottleneck of previous diffusion-based flow estimators. As shown in Fig. 2, TDFlow extracts high-resolution scene features and predicts an initial flow anchor τ_{anchor} using the recurrent update module in WAFT (Wang & Deng, 2025), as detailed in Sec. 3.1. Instead of denoising from pure Gaussian noise, TDFlow perturbs this anchor within a truncated diffusion interval and performs only one reverse update, as described in Sec. 3.2. Sec. 3.3 introduces our Conditional Feature-Space Warping Diffusion Decoder, which aligns noisy flow states with scene features for residual refinement.

3.1 INITIAL FLOW ANCHOR GENERATION

Feature Extraction Module. Our encoder $E(\cdot)$ combines a frozen ViT-S variant from DepthAnythingV2 (Yang et al., 2024) with a ResNet-18 (He et al., 2016). We extract the full-resolution feature map from the DepthAnythingV2 head, concatenate this feature with the original image, and feed the concatenated representation into ResNet-18 for further refinement. The ResNet output is then upsampled through three deconvolutional layers to obtain half-resolution feature maps $E(\mathbf{I}_1)$ and $E(\mathbf{I}_2)$.

Initial Flow Estimation. To obtain the initial flow anchor, we adopt a tailored DPT-Small (Ranftl et al., 2021) backbone following WAFT (Wang & Deng, 2025). Specifically, we merge the upsampled image features and project them into the initial hidden state with a 1×1 convolution. Since the input features have already been downsampled, we adjust the DPT patch size accordingly. During both training and inference, we apply four DPT-based update iterations to produce the initial flow anchor τ_{anchor}.

Implementation and initialization details are provided in Appendix A.

3.2 TRUNCATED DIFFUSION POLICY

Real-world optical flow is highly structured and strongly conditioned on image content. Therefore, starting the diffusion process from pure Gaussian noise is unnecessarily difficult for flow estimation, especially when a reasonable initial motion estimate is available. We introduce a truncated diffusion policy that samples noisy flow states from an anchor-centered Gaussian distribution. This converts diffusion-based flow estimation from full-sample generation into local stochastic refinement around an image-conditioned flow prior.

We distinguish two quantities throughout this section. The first is T_{trunc}, which denotes the maximum forward noising timestep used to perturb the anchor. It controls the noise strength of the anchored distribution. The second is K, which denotes the number of reverse denoising updates used at inference. In TDFlow, $K = 1$ by default, while T_{trunc} is selected by validation and set to 20 in our default configuration.

Training. Given the initial flow anchor τ_{anchor}, we sample a truncated timestep $i \in \{1, \dots, T_{\text{trunc}}\}$ and add Gaussian perturbation around the anchor:

$$\tau^{(i)} = \sqrt{\bar{\alpha}_i}\,\tau_{\text{anchor}} + \sqrt{1-\bar{\alpha}_i}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}). \tag{1}$$

Compared with the vanilla forward process that eventually approaches pure Gaussian noise, this truncated process keeps the noisy sample close to the anchor. As a result, the denoising model only needs to learn a local correction from an anchor-centered noisy state to the target flow, which makes one-step denoising feasible.

The denoising decoder f_θ predicts a residual flow $\Delta\hat{\tau}$ from the noisy flow state, timestep embedding, and image-conditioned features:

$$\Delta\hat{\tau} = f_\theta\big(\tau^{(i)},\, t,\, z\big), \tag{2}$$

where t denotes the timestep embedding as in (Luo et al., 2024), and the condition z consists of the two feature maps $E(\mathbf{I}_1)$ and $E(\mathbf{I}_2)$. The final denoised flow is obtained by residual prediction:

$$\hat{\mathbf{f}} = \tau^{(i)} + \Delta\hat{\tau}. \tag{3}$$

Inference. At inference, we draw one noisy flow sample from the same anchor-centered truncated distribution and apply a single reverse denoising update, i.e., $K = 1$. The decoder predicts the residual $\Delta\hat{\tau}$, and the final optical flow $\hat{\mathbf{f}}$ is computed according to Eq. 3. Since the reverse process contains only one learned denoising pass, the diffusion-stage inference cost is that of a single decoder evaluation.

3.3 Conditional Feature-Space Warping Diffusion Decoder

The truncated diffusion policy requires a denoising network that can effectively exploit the current stochastic flow state $\tau^{(i)}$. Rather than simply appending $\tau^{(i)}$ as an input tensor, we design a *Conditional Feature-Space Warping Diffusion Decoder* (CFSW) to convert it into state-conditioned alignment evidence. Since $\tau^{(i)}$ is sampled around the anchor, it provides a local motion hypothesis for aligning the two frames. CFSW uses this hypothesis to warp the second-frame feature toward the first-frame feature, while the timestep embedding informs the decoder of the perturbation strength. Thus, although feature warping is a classical operation in optical flow, here it serves as a lightweight interface between the anchored diffusion state and one-step residual denoising.

Given a noisy flow sample $\tau^{(i)} \in \mathbb{R}^{2\times h\times w}$, we construct a regular pixel grid $P \in \mathbb{R}^{2\times h\times w}$, where each coordinate $p \in P$ denotes a location in the first frame. We then sample the second-frame feature map $E(\mathbf{I}_2) \in \mathbb{R}^{h\times w\times d}$ at the displaced coordinate $p + \tau_p^{(i)}$. Using bilinear sampling, the warped feature is defined as

$$\widetilde{E}_2^{(i)}(p) = \text{Bilinear}\big(E(\mathbf{I}_2),\, p + \tau_p^{(i)}\big), \quad p \in P, \tag{4}$$

where $\tau_p^{(i)}$ is the noisy flow vector at position p. We denote this warped feature as $\text{Warp}(\tau^{(i)})_p = \widetilde{E}_2^{(i)}(p)$, and form the decoder input as $\Psi^{(i)} = \text{Concat}\big(E(\mathbf{I}_1), \text{Warp}(\tau^{(i)})\big)$. With the timestep embedding $\phi(t)$ injected following (Luo et al., 2024), a modified DPT decoder predicts the residual flow as $\Delta\hat{\tau} = D_\theta\big(\Psi^{(i)}, \phi(t)\big)$, corresponding to $f_\theta(\tau^{(i)}, t, z)$.

3.4 Loss Function

In our truncated diffusion policy, instead of predicting the noise ϵ as in DDPM (Ho et al., 2020), we follow the sample-prediction formulation (Tevet et al., 2023) and directly supervise the predicted flow sample. Following SEA-RAFT (Wang et al., 2024), we optimize a two-component Mixture-of-Laplace (MoL) negative log-likelihood loss. At the diffusion decoder and each refinement layer, the network predicts MoL parameters $M \in \mathbb{R}^{h\times w\times 6}$, which are upsampled to the full image resolution via convex upsampling (Teed & Deng, 2020; Wang et al., 2024) before computing the loss.

Let $\mathbf{f}^{\text{gt}}$ denote the ground-truth flow and Ω denote the set of valid pixels. For each pixel p, the MoL distribution models the likelihood of $\mathbf{f}_p^{\text{gt}}$ under the predicted flow $\hat{\mathbf{f}}_p$ with two Laplace components:

$$\mathcal{L}_{\text{MoL}} = -\frac{1}{|\Omega|}\sum_{p\in\Omega}\log\sum_{m=1}^{2}\pi_{p,m}\,\text{Laplace}\big(\mathbf{f}_p^{\text{gt}};\, \hat{\mathbf{f}}_p,\, \mathbf{b}_{p,m}\big), \tag{5}$$

where $\pi_{p,m}$ and $\mathbf{b}_{p,m}$ are the mixture weight and scale predicted from M, respectively. The final training objective sums the MoL losses from the diffusion output and the intermediate refinement predictions following (Wang et al., 2024).

4 EXPERIMENT

We evaluate our approach on the Sintel (Butler et al., 2012), KITTI (Geiger et al., 2013), and Spring (Mehl et al., 2023) benchmarks. Adhering to prior work (Teed & Deng, 2020; Huang et al., 2022; Wang et al., 2024; Morimitsu et al., 2025), our benchmark training corpus comprises FlyingChairs (Dosovitskiy et al., 2015), FlyingThings (Mayer et al., 2016), HD1K (Kondermann et al., 2016), Sintel, KITTI, Spring, and TartanAir (Wang et al., 2020). We report the standard metrics: endpoint error (EPE), one-pixel outlier rate (1px), percentage of flow outliers (Fl), and the weighted area under the curve (WAUC). Please refer to (Richter et al., 2017; Mehl et al., 2023; Geiger et al., 2013; Morimitsu et al., 2025) for formal definitions.

4.1 TRAINING PROTOCOL

Benchmark results use dataset-specific fine-tuning and hidden test splits; TDFlow's zero-shot results use C+T optical-flow training without TartanAir or target flow fine-tuning. Baseline protocols are in Appendix D.2. Our schedules and image pretraining are detailed in Appendix A; initialization controls are reported in Appendix E.2.

4.2 BENCHMARK RESULTS

a) Quantitative Comparison on Spring Dataset. Table 2 summarizes the Spring benchmark results. TDFlow achieves strong performance among the listed methods, with 95.210 WAUC, and 0.363 EPE. Compared with WAFT (Wang & Deng, 2025), it reduces 1px and Fl while improving WAUC. Figure 5 shows the corresponding qualitative comparison.

b) Quantitative Comparison on Sintel & KITTI Dataset. Table 3 (a) reports results on MPI-Sintel (Butler et al., 2012) and KITTI (Geiger et al., 2013). TDFlow obtains 0.96/1.84 EPE on Sintel Clean/Final and 3.34/2.01 Fl-all/Fl-noc on KITTI All/Non-Occ among the listed methods. As shown in Fig. 3, TDFlow also produces cleaner motion boundaries and fewer artifacts than FlowDiffuser and DPFlow, benefiting from high-resolution feature preservation and anchor-guided one-step denoising.

Table 2: **Spring benchmark comparison.** Lower 1px/EPE/Fl and higher WAUC are better. Colors mark the first three ranks, including ties. *: Spring-team submissions.

Method	1px↓	EPE↓	Fl↓	WAUC↑
FlowNet2 (Ilg et al., 2017)*	6.710	1.040	2.823	90.907
SpyNet (Ranjan & Black, 2017)*	29.963	4.162	12.866	67.150
PWC-Net (Sun et al., 2018a)*	82.270	2.288	4.889	45.670
RAFT (Teed & Deng, 2020)*	6.790	1.476	3.198	90.920
GMA (Jiang et al., 2021)*	7.074	0.914	3.079	90.722
FlowFormer (Huang et al., 2022)*	6.510	0.723	2.384	91.679
GMFlow (Xu et al., 2022)*	10.355	0.945	2.952	82.337
CroCoFlow (Weinzaepfel et al., 2023)	4.565	0.498	1.508	93.660
RPKNet (Morimitsu et al., 2024)	4.809	0.657	1.756	92.638
Win-Win (Leroy et al., 2023)	5.371	0.475	1.621	92.270
SEA-RAFT(M) (Wang et al., 2024)	3.686	0.363	1.347	94.534
MemFlow (Dong & Fu, 2024)	4.482	0.471	1.416	93.855
StreamFlow (Sun et al., 2024)	4.152	0.467	1.424	94.404
DPFlow (Morimitsu et al., 2025)	3.442	0.340	1.311	94.980
WAFT (Wang & Deng, 2025)	3.418	0.340	1.280	94.663
TDFlow (Ours)	3.283	0.363	1.219	95.210

Table 3: **Quantitative evaluation on Sintel and KITTI.** (a) Hidden test splits after fine-tuning. (b) Cross-dataset evaluation without target flow fine-tuning. Baseline protocols and references are in Appendix D.2. Lower is better. First, second, and third ranks include ties.

(a) Benchmark comparison

Method	Sintel		KITTI	
	Clean	Final	Fl-all	Fl-noc
PWC-Net+	3.45	4.60	7.72	4.91
RAFT	1.61	2.86	5.10	3.07
GMFlowNet	1.39	2.65	4.79	2.75
FlowFormer	1.20	2.12	4.68	2.69
AnyFlow	1.23	2.44	4.41	2.69
MatchFlow(G)	1.16	2.37	4.63	2.77
CroCoFlow	1.09	2.44	3.64	2.40
RPKNet	1.31	2.65	4.64	2.71
SEA-RAFT(L)	1.31	2.60	4.30	–
FlowDiffuser	1.02	2.03	4.17	2.82
SAMFlow	1.00	2.08	4.49	–
DPFlow	1.04	1.97	3.56	2.12
WAFT	1.09	2.34	3.42	2.04
TDFlow (Ours)	0.96	1.84	3.34	2.01

(b) Zero-shot evaluation

Method	Sintel		KITTI	
	Clean	Final	EPE	Fl-all
FlowNet2	2.02	3.14	10.10	30.4
GMFlowNet	1.14	2.71	4.24	15.4
FlowFormer	1.01	2.40	4.09	14.7
FlowFormer++	0.90	2.30	3.93	14.2
CCMR+	0.98	2.36	–	12.9
MatchFlow(G)	1.03	2.45	4.08	15.6
SEA-RAFT(L)	1.19	4.11	3.62	12.9
AnyFlow	1.10	2.52	3.76	12.4
SAMFlow	0.87	2.11	3.44	12.3
FlowDiffuser	0.86	2.19	3.61	11.8
DPFlow	1.02	2.26	3.37	11.1
FlowSeek	1.03	2.18	3.31	11.2
WAFT	1.00	2.15	3.10	10.3
TDFlow (Ours)	0.87	2.09	2.87	9.3

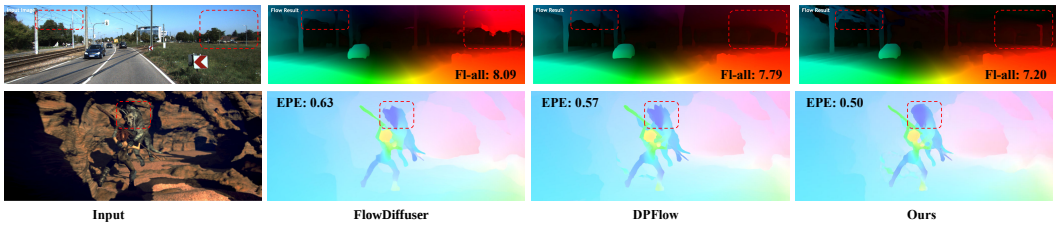


Figure 3: **Qualitative benchmark comparison.** FlowDiffuser (Luo et al., 2024), DPFlow (Morimitsu et al., 2025), and TDFlow on KITTI and Sintel. These official-submission examples show motion boundaries and local artifacts.

4.3 Zero-Shot Evaluation

Following prior work (Teed & Deng, 2020; Huang et al., 2022; Sun et al., 2018a), we adopt the zero-shot protocol: the model is trained only on FlyingChairs (Dosovitskiy et al., 2015) and FlyingThings (Mayer et al., 2016), then evaluated on the *training* splits of Sintel (Butler et al., 2012) and KITTI (Geiger et al., 2013). As summarized in Table 3 (b), TDFlow exhibits strong cross-dataset generalization, achieving 0.87/2.09 on Sintel Clean/Final and 2.87/9.3 on KITTI EPE/Fl-all. These results demonstrate strong cross-dataset generalization without target-dataset optical-flow fine-tuning.

4.4 Analysis of Anchor-Guided Truncated Diffusion Policy

Since TDFlow starts from an initial flow anchor τ_{anchor}, we analyze whether its improvement mainly comes from stronger initialization or from the proposed anchor-guided truncated diffusion policy. The variant *TDFlow w.o. TDP (anchor only)* removes the truncated diffusion policy and directly evaluates the predicted anchor flow. We also reinterpret the compatibility study in Table 4 as an anchor-quality analysis, where GMA (Jiang et al., 2021), KPA-Flow (Luo et al., 2022a), SKFlow (Sun et al., 2022), and DPFlow (Morimitsu et al., 2025) provide anchors of different accuracy levels.

Across all anchor generators, TDFlow consistently improves the corresponding baselines, showing that the method is not tied to a particular initialization quality. Compared with FlowDiffuser (Luo et al., 2024) under the same anchor families, TDFlow also achieves larger gains, suggesting that the

Table 4: **Anchor compatibility.** C+T training. PG (%) is the mean relative error reduction over each dataset's two metrics (Eq. 6).

Method	Sintel			KITTI		
	Clean	Final	PG	EPE	Fl-all	PG
GMA	1.30	2.74	–	4.69	17.10	–
FlowDiffuser-G	1.26	2.59	4.3	4.32	15.60	8.3
TDFlow-G	**1.23**	**2.57**	**5.8**	**4.29**	**15.20**	**9.8**
KPA-Flow	1.28	2.68	–	4.46	15.90	–
FlowDiffuser-K	1.24	**2.45**	5.9	3.97	14.70	9.3
TDFlow-K	**1.18**	2.47	**7.8**	**3.92**	**14.50**	**10.5**
SKFlow	1.22	2.46	–	4.27	15.50	–
FlowDiffuser-S	1.18	2.39	3.1	3.91	14.20	8.4
TDFlow-S	**1.16**	**2.35**	**4.7**	**3.77**	**13.70**	**11.7**
DPFlow	1.02	2.26	–	3.37	11.10	–
FlowDiffuser-D	1.01	2.24	0.9	3.33	10.90	1.5
TDFlow-D	**0.98**	**2.21**	**3.1**	**3.14**	**10.60**	**5.7**

Table 5: **Ablation study.** C+T training. The no-noise row keeps the same anchor and CFSW decoder; noise is removed in training and inference.

Configuration	Sintel		KITTI	
	Clean	Final	EPE	Fl-all
#1: Components and matched noise control				
Anchor only	1.03	2.32	3.23	10.75
Same decoder, no noise	1.01	2.29	3.21	10.63
Without CFSW	0.93	2.14	3.02	9.87
TDFlow (default)	**0.87**	**2.09**	**2.87**	**9.30**
#2: Denoiser evaluations K				
2	0.87	2.09	2.88	9.30
3	0.88	2.10	2.87	9.30
#3: Truncation timestep T_{trunc}				
10	0.92	2.14	2.92	9.90
30	0.88	2.10	2.90	9.60

Default: $K = 1, T_{\text{trunc}} = 20$.



Figure 4: **Refinement beyond the anchor.** The anchor-only prediction (middle) has more moving-object outliers than TDFlow (right), complementing the controlled quantitative ablations.

benefit comes from anchor-centered truncated denoising rather than a generic diffusion refinement module.

The anchor-only comparison in Fig. 4 shows the same trend; the corresponding analysis is retained in Appendix E.2.

For both TDFlow and FlowDiffuser, performance gain (PG) is the average relative error reduction over the two metrics of each dataset:

$$\text{PG } (\%) = \frac{100}{2} \sum_{j=1}^{2} \frac{M_j^{\text{base}} - M_j^{\text{refined}}}{M_j^{\text{base}}}. \tag{6}$$

The metrics are Clean/Final EPE for Sintel and EPE/Fl-all for KITTI. The PG ranges are 3.1–7.8% on Sintel and 5.7–11.7% on KITTI, rounded once to one decimal. Additional integration details and controls are in Appendices D.3 and E.

4.5 Ablation Study

Effect of Designs in Denoising Decoder. Table 5 (#1) ablates the key components of TDFlow. The anchor-only baseline directly evaluates τ_{anchor} without truncated diffusion refinement and is consistently worse than the full model, showing that the gain is not merely from initialization. Removing either the Conditional Feature-Space Warping decoder (CFSW) or the noise injection in the Truncated Diffusion Policy (TDP) degrades performance on both Sintel and KITTI. With the same decoder and anchor, removing noise gives 1.01/2.29 on Sintel and 3.21/10.63 on KITTI; Appendix E.1 details this matched control. Removing CFSW increases KITTI EPE from 2.87 to 3.02 and Sintel

Clean/Final EPE from 0.87/2.09 to 0.93/2.14, confirming the benefit of flow-state-guided feature alignment. The full model performs best, demonstrating that CFSW and TDP jointly complement the anchor flow.

Denoising Step Number. Table 5 (#2) studies the effect of the denoising step number K. Increasing K from the default 1 to 2 or 3 brings negligible accuracy changes on both Sintel and KITTI. This indicates that the anchored noisy state is already close to the target flow distribution, so TDFlow can produce stable predictions with only one denoising update. Additional steps mainly introduce extra computation while providing limited accuracy improvement, supporting the efficiency motivation of our truncated diffusion design.

Truncated Diffusion Step. Table 5 (#3) studies $T_{\text{trunc}} \in \{10, 20, 30\}$ on Sintel and KITTI. A smaller truncation step injects insufficient perturbation and limits the model's ability to explore alternative local flow hypotheses, while a larger step moves the noisy state too far from the anchor and makes refinement harder. The setting $T_{\text{trunc}} = 20$ achieves the lowest errors, suggesting a good balance between introducing useful stochasticity and preserving reliable anchor-centered motion structure.

4.6 Efficiency and Inference Stability

Inference implementation and measurement. We measure TDFlow on one RTX 4090, with batch size 1 and 376×1248 inputs. Table 1 reports 69 ms end-to-end latency and 1.210 GB peak inference memory. With one denoiser update, one-step and diffusion-stage latency are both 9 ms. The full comparison (Table 11) and measurement protocol are in Appendix F.

Ten-seed inference stability. With the checkpoint fixed, ten inference-noise seeds give Sintel Clean/Final EPE of $0.870 \pm 0.003/2.090 \pm 0.006$ and KITTI EPE/Fl-all of $2.870 \pm 0.005/9.300 \pm 0.008$. The maximum standard deviation across anchors and ablations is 0.009. These statistics measure inference-noise variation, not independent training runs.

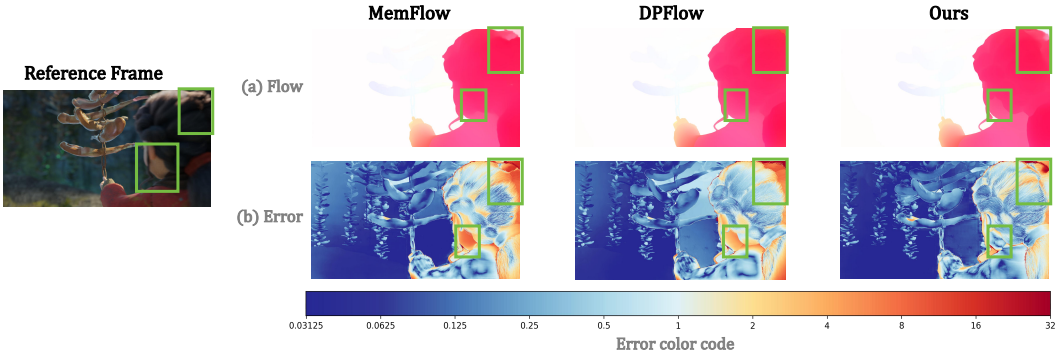


Figure 5: **Spring qualitative comparison.** MemFlow (Dong & Fu, 2024), DPFlow (Morimitsu et al., 2025), and TDFlow. The highlighted regions show flow boundaries and error maps; TDFlow reduces local errors in these examples.

5 Conclusion

In this work, we presented TDFlow, a novel optical flow estimation framework that integrates an anchor-guided truncated diffusion policy. Compared to previous diffusion-based optical flow estimators, our method reduces the sampling procedure from multiple denoising steps to only one step, substantially improving inference efficiency. In addition, instead of relying on expensive cost-volume construction, we design a conditional feature-space warping decoder that aligns image features using the noisy flow state and performs efficient residual refinement. This design preserves the generative modeling advantage of diffusion while better satisfying the accuracy and latency requirements of optical flow estimation. TDFlow achieves state-of-the-art performance on multiple benchmarks, including Sintel, KITTI, and Spring, while maintaining strong efficiency. Extensive experiments further demonstrate the effectiveness, compatibility, and generalization capability of our method.

AI USE STATEMENT

Generative AI was used to assist with writing and presentation, including language polishing, reorganizing existing manuscript content and reported results, checking aggregate-metric arithmetic, and LaTeX/table formatting. It was not used to run experiments or generate experimental measurements. The authors are responsible for the final text, claims, figures, and results.

ETHICS STATEMENT

TDFlow may support motion estimation for robotics, video understanding, and autonomous-driving research. Benchmark accuracy is not evidence of safe real-world deployment. Safety-critical use requires system-level validation, robustness testing under distribution shifts, and appropriate human oversight. Public datasets and pretrained assets are described in Appendix I; this submission does not redistribute those datasets or upstream checkpoints.

REPRODUCIBILITY STATEMENT

Section 3 specifies the anchored noising process, decoder, and loss. Section 4.1 separates benchmark and zero-shot protocols, while Appendix A and Appendix A.3 provide architecture and training details. Table 5 and Appendix E describe the ablations and additional controls. Table 11, Section 4.6, and Appendix F report inference settings and repeated-measurement statistics. Further integration details, complexity analysis, and qualitative results appear in the appendices.

REFERENCES

Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part IV 8*, pp. 25–36. Springer, 2004.

Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pp. 611–625. Springer, 2012.

Qifeng Chen and Vladlen Koltun. Full flow: Optical flow estimation by global optimization over regular grids. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4706–4714, 2016.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

Changxing Deng, Ao Luo, Haibin Huang, Shaodan Ma, Jiangyu Liu, and Shuaicheng Liu. Explicit motion disentangling for efficient optical flow estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9521–9530, 2023.

Qiaole Dong and Yanwei Fu. Memflow: Optical flow estimation and prediction with memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19068–19078, 2024.

Qiaole Dong, Chenjie Cao, and Yanwei Fu. Rethinking optical flow from geometric matching consistent perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1337–1347, 2023.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766, 2015.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. In *European Conference on Computer Vision*, pp. 668–685. Springer, 2022.

Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8981–8989, 2018.

Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2462–2470, 2017.

Azin Jahedi, Maximilian Luz, Marc Rivinius, Lukas Mehl, and Andrés Bruhn. Ms-raft+: High resolution multi-scale raft. *International Journal of Computer Vision*, pp. 1–22, 2023.

Azin Jahedi, Maximilian Luz, Marc Rivinius, and Andrés Bruhn. Ccmr: High resolution optical flow estimation via coarse-to-fine context-guided motion reasoning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 6899–6908, 2024.

Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and trends® in computer graphics and vision*, 12(1–3):1–308, 2020.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *ICLR*, 2022.

Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard Hartley. Learning to estimate hidden motions with global motion aggregation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9772–9781, 2021.

Hyunyoung Jung, Zhuo Hui, Lei Luo, Haitao Yang, Feng Liu, Sungjoo Yoo, Rakesh Ranjan, and Denis Demandolx. Anyflow: Arbitrary scale optical flow with implicit neural representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5455–5465, 2023.

Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrulis, Alexander Brock, Burkhard Gussefeld, Mohsen Rahimimoghaddam, Sabine Hofmann, Claus Brenner, et al. The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 19–28, 2016.

Vincent Leroy, Jerome Revaud, Thomas Lucas, and Philippe Weinzaepfel. Win-win: Training high-resolution vision transformers from two windows. *arXiv preprint arXiv:2310.00632*, 2023.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

Xiaozhang Liu, Hui Liu, and Yuxiu Lin. Video frame interpolation via optical flow estimation with image inpainting. *International Journal of Intelligent Systems*, 35(12):2087–2102, 2020.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Ao Luo, Fan Yang, Xin Li, and Shuaicheng Liu. Learning optical flow with kernel patch attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8906–8915, 2022a.

Ao Luo, Fan Yang, Kunming Luo, Xin Li, Haoqiang Fan, and Shuaicheng Liu. Learning optical flow with adaptive graph reasoning. In *AAAI*, 2022b.

Ao Luo, Fan Yang, Xin Li, Lang Nie, Chunyu Lin, Haoqiang Fan, and Shuaicheng Liu. Gaflow: Incorporating gaussian attention into optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9642–9651, 2023.

Ao Luo, Xin Li, Fan Yang, Jiangyu Liu, Haoqiang Fan, and Shuaicheng Liu. Flowdiffuser: Advancing optical flow estimation with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19167–19176, 2024.

Zeyu Ma, Zachary Teed, and Jia Deng. Multiview stereo with cascaded epipolar raft. In *European Conference on Computer Vision*, pp. 734–750. Springer, 2022.

Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4040–4048, 2016.

Lukas Mehl, Jenny Schmalfuss, Azin Jahedi, Yaroslava Nalivayko, and Andrés Bruhn. Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4981–4991, 2023.

Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3061–3070, 2015.

Henrique Morimitsu, Xiaobin Zhu, Xiangyang Ji, and Xu-Cheng Yin. Recurrent partial kernel network for efficient optical flow estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 4278–4286, 2024.

Henrique Morimitsu, Xiaobin Zhu, Roberto M Cesar, Xiangyang Ji, and Xu-Cheng Yin. Dpflow: Adaptive optical flow estimation with a dual-pyramid framework. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 17810–17820, 2025.

AJ Piergiovanni and Michael S Ryoo. Representation flow for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9945–9953, 2019.

Matteo Poggi and Fabio Tosi. Flowseek: Optical flow made easier with depth foundation models and motion bases, 2025.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12179–12188, 2021.

Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4161–4170, 2017.

Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2213–2222, 2017.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Saurabh Saxena, Charles Herrmann, Junhwa Hur, Abhishek Kar, Mohammad Norouzi, Deqing Sun, and David J Fleet. The surprising effectiveness of diffusion models for optical flow and monocular depth estimation. *Advances in Neural Information Processing Systems*, 36:39443–39469, 2023.

Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1599–1610, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021b.

Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *ICRA*, 2024.

Maria Stamatopoulou, Jianwei Liu, and Dimitrios Kanoulas. Dippest: Diffusion-based path planner for synthesizing trajectories applied on quadruped robots. *arXiv preprint arXiv:2405.19232*, 2024.

Xiuchao Sui, Shaohua Li, Xue Geng, Yan Wu, Xinxing Xu, Yong Liu, Rick Goh, and Hongyuan Zhu. Craft: Cross-attentional flow transformer for robust optical flow. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 17602–17611, 2022.

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8934–8943, 2018a.

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. *IEEE transactions on pattern analysis and machine intelligence*, 42(6):1408–1423, 2019.

Shangkun Sun, Yuanqi Chen, Yu Zhu, Guodong Guo, and Ge Li. Skflow: Learning optical flow with super kernels. *Advances in Neural Information Processing Systems*, 35:11313–11326, 2022.

Shangkun Sun, Jiaming Liu, Huaxia Li, Guoqing Liu, Thomas Li, and Wei Gao. Streamflow: streamlined multi-frame optical flow estimation for video sequences. *Advances in neural information processing systems*, 37:9205–9228, 2024.

Shuyang Sun, Zhanghui Kuang, Lu Sheng, Wanli Ouyang, and Wei Zhang. Optical flow guided feature: A fast and robust motion representation for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1390–1399, 2018b.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, pp. 402–419, 2020.

Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023.

Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4909–4916. IEEE, 2020.

Yihan Wang and Jia Deng. Waft: Warping-alone field transforms for optical flow. *arXiv preprint arXiv:2506.21526*, 2025.

Yihan Wang, Lahav Lipson, and Jia Deng. Sea-raft: Simple, efficient, accurate raft for optical flow. In *European Conference on Computer Vision*, pp. 36–54. Springer, 2024.

Philippe Weinzaepfel, Vincent Leroy, Thomas Lucas, Romain Brégier, Yohann Cabon, Vaibhav Arora, Leonid Antsfeld, Boris Chidlovskii, Gabriela Csurka, and Jérôme Revaud. Croco: Self-supervised pre-training for 3d vision tasks by cross-view completion. *Advances in Neural Information Processing Systems*, 35:3502–3516, 2022.

Philippe Weinzaepfel, Thomas Lucas, Vincent Leroy, Yohann Cabon, Vaibhav Arora, Romain Brégier, Gabriela Csurka, Leonid Antsfeld, Boris Chidlovskii, and Jérôme Revaud. Croco v2: Improved cross-view completion pre-training for stereo matching and optical flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17969–17980, 2023.

Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8121–8130, 2022.

Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, Fisher Yu, Dacheng Tao, and Andreas Geiger. Unifying flow, stereo and depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic video interpolation. *Advances in Neural Information Processing Systems*, 32, 2019.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 2024.

Jingfeng Yao and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. *arXiv preprint arXiv:2501.01423*, 2025.

Jingfeng Yao, Cheng Wang, Wenyu Liu, and Xinggang Wang. Fasterdit: Towards faster diffusion transformer training without architecture modification. *arXiv preprint arXiv:2410.10356*, 2024.

Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition: 29th DAGM Symposium, Heidelberg, Germany, September 12-14, 2007. Proceedings 29*, pp. 214–223. Springer, 2007.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

Chengqian Zhao, Cheng Feng, Dengwang Li, and Shuo Li. Optical flow-auxiliary multi-task regression network for direct quantitative measurement, segmentation and motion estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020a.

Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I Chang, Yan Xu, et al. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6278–6287, 2020b.

Shiyu Zhao, Long Zhao, Zhixing Zhang, Enyu Zhou, and Dimitris Metaxas. Global matching with overlapping attention for optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17592–17601, 2022.

Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders. In *ICLR*, 2023.

Zihua Zheng, Ni Nie, Zhi Ling, Pengfei Xiong, Jiangyu Liu, Hao Wang, and Jiankun Li. Dip: Deep inverse patchmatch for high-resolution optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8925–8934, 2022.

Shili Zhou, Ruian He, Weimin Tan, and Bo Yan. Samflow: Eliminating any fragmentation in optical flow with segment anything model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 7695–7703, 2024.

Lianghui Zhu, Zilong Huang, Bencheng Liao, Jun Hao Liew, Hanshu Yan, Jiashi Feng, and Xinggang Wang. Dig: Scalable and efficient diffusion models with gated linear attention. *arXiv preprint arXiv:2405.18428*, 2024.

Yiming Zuo and Jia Deng. View synthesis with sculpted neural points. In *11th International Conference on Learning Representations, ICLR 2023*, 2023.

A Implementation and Training Details

We provide implementation and training details, conditional diffusion background, complexity analysis, expanded benchmark comparisons, and additional qualitative results.

A.1 Feature Encoder, Fusion, and Initialization

For each normalized RGB image I, DAv2-S (Yang et al., 2024) produces a full-resolution 32-channel feature D. Concatenating $[D, I]$ gives 35 channels. A custom stride-2 stem, ResNet-18 (He et al., 2016), and three deconvolution blocks produce a 64-channel half-resolution feature R. Fusion is

$$E(I) = \mathrm{Conv}_{1\times1}([R, \mathrm{Down}_2(D)]) \in \mathbb{R}^{64\times H/2\times W/2}. \tag{7}$$

The two images share encoder weights. The complete DAv2-S branch is frozen under the default initialization. ResNet-18 starts from ImageNet-1K weights and is fine-tuned; the custom stem, deconvolutions, and fusion layer are learned from scratch.

The DPT-Small anchor estimator (Ranftl et al., 2021; Wang & Deng, 2025) uses pretrained ViT-S blocks that are fine-tuned, with task-specific patch embedding, decoder, and prediction heads learned from scratch. Four recurrent updates produce the anchor before the separate CFSW denoising update. The model does not initialize from a pretrained WAFT optical-flow checkpoint. Table 9 separately randomizes each pretrained component and then all three together; randomized components are trainable.

A.2 Anchor and Decoder Interpretation

The anchor is used as the center of the truncated diffusion distribution rather than as the final prediction. Therefore, TDFlow does not require the anchor to be perfectly accurate; instead, the subsequent denoising decoder learns to correct local residual errors around the anchor. This formulation also makes the truncated diffusion policy compatible with different initial flow estimators, as empirically evaluated in Table 4. Initialization and implementation details are provided in Appendix A.

Different from a deterministic warping refinement module, CFSW recomputes the aligned feature from the stochastic diffusion state and is trained to map anchor-centered noisy hypotheses to the clean flow. Since the truncated distribution keeps $\tau^{(i)}$ close to the anchor, the decoder mainly estimates local residual corrections, making half-resolution feature-space warping a memory-efficient alternative to dense cost-volume decoding.

This probabilistic loss encourages accurate flow prediction while allowing the decoder to represent spatially varying uncertainty, especially near motion boundaries and occluded regions.

A.3 Training Schedules

The full training corpus for benchmark submissions comprises FlyingChairs (Dosovitskiy et al., 2015), FlyingThings (Mayer et al., 2016), HD1K (Kondermann et al., 2016), MPI Sintel (Butler et al., 2012), KITTI (Geiger et al., 2013), Spring (Mehl et al., 2023), and TartanAir (Wang et al., 2020). The zero-shot C+T protocol is separate and uses only FlyingChairs and FlyingThings for optical-flow training.

Pretraining and Fine-Tuning for Benchmark Submissions. We first pretrain on TartanAir (Wang et al., 2020) for 300k steps (batch size 32, learning rate 4×10^{-4}). This is followed by fine-tuning on FlyingChairs (Dosovitskiy et al., 2015) for 50k and then on FlyingThings (Mayer et al., 2016) for 200k steps, maintaining the same hyperparameters. For final submissions, we use a batch size of 16 unless otherwise noted below and lower the learning rate to 1×10^{-4}. Specifically, the KITTI submission undergoes an additional 5k-step fine-tuning on KITTI (training) (Geiger et al., 2013). The Sintel submission is further adapted by fine-tuning on a union of FlyingThings, HD1K, KITTI (training), and Sintel (training) datasets for 200k steps. For Spring submissions, we train two variants on Spring (training) (Mehl et al., 2023) for 200k steps each: a 1080p model (batch size 8) and a 540p model (batch size 16).

Zero-Shot Evaluation. Our zero-shot protocol pretrains the models on FlyingChairs for 50k steps, then fine-tunes them on FlyingThings for 200k steps, using a batch size of 16 and a learning rate of

1×10^{-4}. Unless otherwise specified, all diagnostic variants follow the same training and evaluation protocol, while the anchor-only variant removes the truncated diffusion policy at evaluation.

B Conditional Diffusion Background

Task Formulation. Given two consecutive images $\mathbf{I}_1$ and $\mathbf{I}_2$, optical flow predicts a dense motion field τ that maps pixels in $\mathbf{I}_1$ to their correspondences in $\mathbf{I}_2$:

$$\tau = \mathrm{F}_\Phi(\mathbf{I}_1, \mathbf{I}_2), \tag{8}$$

where Φ denotes the network parameters.

Conditional Diffusion Model. Let $\tau^{(0)}$ be the clean flow and $\tau^{(i)}$ its noisy state at diffusion step i. The forward process is defined as

$$q\big(\tau^{(i)} \mid \tau^{(0)}\big) = \mathcal{N}\big(\tau^{(i)}; \sqrt{\bar{\alpha}_i}\tau^{(0)}, (1-\bar{\alpha}_i)\mathbf{I}\big), \tag{9}$$

where $\bar{\alpha}_i = \prod_{s=1}^{i} \alpha_s$, $\alpha_s = 1 - \beta_s$, and $\{\beta_s\}_{s=1}^{T}$ is the noise schedule. A conditional model $f_\theta(\tau^{(i)}, z, i)$ predicts the clean sample from $\tau^{(i)}$ under condition z. Vanilla inference samples $\tau^{(T)} \sim \mathcal{N}(0, \mathbf{I})$ and repeatedly applies

$$p_\theta\big(\tau^{(i-1)} \mid \tau^{(i)}, z\big) = \mathcal{N}\big(\tau^{(i-1)}; \mu_\theta(\tau^{(i)}, z, i), (1-\alpha_i)\mathbf{I}\big), \tag{10}$$

for $i = T, T-1, \ldots, 1$, where μ_θ is the predicted mean. Thus,

$$p_\theta\big(\tau^{(0)} \mid z\big) = \int p\big(\tau^{(T)}\big) \prod_{i=1}^{T} p_\theta\big(\tau^{(i-1)} \mid \tau^{(i)}, z\big) \mathrm{d}\tau^{1:T}. \tag{11}$$

Limitations of Previous Diffusion-based Flow Estimator. FlowDiffuser (Luo et al., 2024) applies diffusion denoising to optical flow, but its cost-volume decoder still requires multiple reverse passes. As shown in Table 1, the published configuration uses four denoising passes, causing 210 ms diffusion latency, 290 ms total inference time, and 2.924 GB peak inference memory. TDFlow reduces this cost with an anchor-centered truncated diffusion policy and a lightweight half-resolution feature-space warping decoder. With a single denoising update ($K = 1$), our implementation reaches 9 ms diffusion latency, 69 ms total inference time, and 1.210 GB peak inference memory in the reported implementation. This comparison includes differences in model design and inference implementation; component-wise effects are analyzed in the ablation studies.

Noise-conditioned recovery objective. The perturbation is applied to a *predicted anchor*, whereas supervision is the *ground-truth flow*. TDFlow thus learns a conditional recovery map over a local distribution of anchor perturbations, rather than the exact reverse of a ground-truth corruption chain. With $g_\theta(\tau^{(i)}, z, i) = \tau^{(i)} + f_\theta(\tau^{(i)}, \phi(i), z)$, the learning objective has the form

$$\mathcal{L}_{\mathrm{TDFlow}} = \mathbb{E}_{i,\epsilon}\Big[\ell\Big(g_\theta(\tau^{(i)}, z, i), \mathbf{f}^{\mathrm{gt}}\Big)\Big]. \tag{12}$$

Unlike deterministic refinement of a clean anchor, this objective exposes the decoder to perturbed hypotheses. The matched no-noise control in Appendix E.1 tests this distinction directly. A single update is an empirical operating choice, not a claim that a full diffusion reverse process is unnecessary in general. The following interpretation connects the perturbation objective with local sensitivity.

Local-regularization interpretation. Equation 12 averages supervised recovery over noisy anchor states. As a local interpretation, consider a squared-error surrogate and a locally linear approximation to the complete recovery map g_θ. Small Gaussian input perturbations then suggest a sensitivity penalty of the schematic form

$$\mathcal{L}_{\mathrm{noisy}} \approx \mathcal{L}_{\mathrm{clean}} + \lambda \| J_{g_\theta}(\tau_{\mathrm{anchor}}) \|_F^2, \tag{13}$$

where J_{g_θ} is the Jacobian with respect to the input flow and λ depends on perturbation magnitude. This is an explanatory approximation, not an exact identity for TDFlow's MoL objective or finite noise schedule. It motivates reduced local sensitivity but does not prove convergence, calibrated uncertainty, or a benefit for arbitrarily corrupted anchors. The controlled no-noise and initialization studies provide empirical evidence for the recovery formulation; they do not establish equivalence to a full generative diffusion model.

C Additional Analysis of Warping and Cost Volume

C.1 Why Warping Fits Anchor-guided Diffusion Refinement

The decoder receives a coarse motion hypothesis rather than starting correspondence search from scratch. Its task is to correct local inconsistencies using image evidence. Given the current flow f_{cur}, warping aligns second-frame features by bilinear sampling:

$$\mathrm{Warp}(f_{\mathrm{cur}})_p = E(I_2)_{p+(f_{\mathrm{cur}})_p}. \tag{14}$$

This representation conditions residual prediction directly on the current hypothesis. A cost volume instead explicitly represents candidate correspondences, which is useful when initialization is unreliable. TDFlow uses the anchor to narrow the correspondence region before refinement; warping avoids repeatedly enumerating this extra search dimension. This rationale is specific to local recovery and does not make cost volumes unnecessary for optical flow in general.

C.2 Complexity Comparison

For $h \times w$ features with d channels, full all-pairs correlation computes a d-dimensional similarity for every pair of spatial locations:

$$\mathrm{Cost}_{\mathrm{full}} = \mathcal{O}((hw)^2 d). \tag{15}$$

A partial cost volume restricts candidate search to radius r around the current estimate:

$$\mathrm{Cost}_{\mathrm{partial}} = \mathcal{O}(hw(2r+1)^2 d). \tag{16}$$

Warping bilinearly samples one d-dimensional feature per location, with a constant-size interpolation neighborhood:

$$\mathrm{Cost}_{\mathrm{warp}} = \mathcal{O}(hwd). \tag{17}$$

The reduction concerns the matching primitive, not the complete network: feature extraction, the anchor backbone, and decoder operations remain. It explains why dense half-resolution alignment can be practical without proving an end-to-end speedup over every correlation-based method. Tables 11 and 12 report complete-implementation profiles.

C.3 Advantages of High-Resolution Indexing

Motion boundaries, thin structures, and occluded regions require fine spatial alignment. Coarse feature lookup can mix neighboring motions and blur boundary corrections. Half-resolution warping preserves the lookup at a finer scale, supplying more localized evidence to the residual decoder. This is complementary to noise-conditioned training: warping controls how image evidence is retrieved, while the truncated policy controls which flow states the network learns to recover from. The CFSW removal and matched no-noise controls test these contributions separately. The qualitative comparisons illustrate challenging regions but do not isolate resolution as the sole cause of improvement.

D Complete Benchmark and Compatibility Comparisons

D.1 Full Spring Comparison

Table 2 reports the complete Spring benchmark comparison. Figure 5 provides a qualitative example.

D.2 Sources for Compact Main-paper Baselines

Table 6 identifies the methods reported in Table 3. Each method retains its reported benchmark or zero-shot evaluation protocol.

Configuration-specific TartanAir use. TartanAir use before target-benchmark evaluation may occur as a separate pretraining stage or within the optical-flow training mixture. The following training details apply to the displayed configurations, not to every checkpoint of a method. CroCoFlow

Table 6: **Baseline references for Table 3.** Methods appearing in both panels share the corresponding reference.

Method	Reference
PWC-Net+	Sun et al. (2019)
RAFT	Teed & Deng (2020)
GMFlowNet	Zhao et al. (2022)
FlowFormer	Huang et al. (2022)
AnyFlow	Jung et al. (2023)
MatchFlow(G)	Dong et al. (2023)
CroCoFlow	Weinzaepfel et al. (2023)
RPKNet	Morimitsu et al. (2024)
SEA-RAFT(L)	Wang et al. (2024)
FlowDiffuser	Luo et al. (2024)
SAMFlow	Zhou et al. (2024)
DPFlow	Morimitsu et al. (2025)
WAFT	Wang & Deng (2025)
FlowNet2	Ilg et al. (2017)
FlowFormer++	Shi et al. (2023)
CCMR+	Jahedi et al. (2024)
FlowSeek	Poggi & Tosi (2025)

includes TartanAir in its flow-training mixture (Weinzaepfel et al., 2023). Win-Win reports TartanAir-augmented benchmark training and subsequent Spring fine-tuning (Leroy et al., 2023). The displayed SEA-RAFT and WAFT benchmark configurations also use TartanAir (Wang et al., 2024; Wang & Deng, 2025). TDFlow benchmark results follow our TartanAir pretraining schedule in Appendix A.3.

Cross-dataset configurations. Table 3(b) retains the C+T-only SEA-RAFT(L) row (1.19/4.11 on Sintel and 3.62/12.9 on KITTI) (Wang et al., 2024), the C+T-only WAFT row (Wang & Deng, 2025), and our C+T-only TDFlow result. In contrast, the displayed FlowSeek result (1.03/2.18 and 3.31/11.2) is the TartanAir-pretrained FlowSeek(L) configuration (Poggi & Tosi, 2025). Thus cross-dataset evaluation means no target-dataset flow fine-tuning, not identical pretraining data for all methods. Other image-level pretraining is separate from optical-flow training. All accuracy values are retained from their reported configurations.

D.3 Full Compatibility Comparison

Table 7 provides inference means and standard deviations for TDFlow with GMA, KPA-Flow, SKFlow, and DPFlow anchors. Main-paper Table 4 gives all base and refined means with uniformly recomputed PG.

Table 7: **Per-anchor inference variability.** Means and standard deviations over ten noise seeds with each checkpoint fixed. The corresponding base results and corrected PG values appear in main-paper Table 4.

Variant	Sintel Clean	Sintel Final	KITTI EPE	KITTI Fl-all
TDFlow-G	1.230 ± 0.004	2.570 ± 0.005	4.290 ± 0.006	15.200 ± 0.009
TDFlow-K	1.180 ± 0.003	2.470 ± 0.004	3.920 ± 0.005	14.500 ± 0.008
TDFlow-S	1.160 ± 0.003	2.350 ± 0.004	3.770 ± 0.005	13.700 ± 0.008
TDFlow-D	0.980 ± 0.002	2.210 ± 0.003	3.140 ± 0.004	10.600 ± 0.007

Different integration strategies. FlowDiffuser-G/K/S results are taken from the original FlowDiffuser work (Luo et al., 2024). Its recurrent denoising decoder and hidden-state denoising are integrated into the corresponding RAFT-like backbones while retaining their feature, context, correlation, and update components; these variants still initialize noise-to-flow generation from Gaussian noise. FlowDiffuser-D is the corresponding DPFlow-based implementation. TDFlow-G/K/S/D instead takes each base prediction as an image-conditioned anchor and applies the truncated policy with CFSW. Therefore, this comparison tests compatibility across method families, not a decoder-only ablation with identical inputs and backbones. The matched decoder-only control is Table 5.

Uniform aggregation. For example, TDFlow-G's Sintel gain is $\frac{1}{2}[(1.30 - 1.23)/1.30 + (2.74 - 2.57)/2.74] \times 100 = 5.8\%$ after rounding. The same formula is used for every FlowDiffuser and TDFlow row. KITTI averages relative EPE and Fl-all reductions, not raw values with different units. PG is neither the maximum single-metric improvement nor an average of absolute errors.

E Additional Controlled Experiments

The following experiments provide detailed controls for the recovery, initialization, and anchor analyses. All follow C+T optical-flow training and evaluation on the labeled Sintel and KITTI training splits.

E.1 Noise-conditioned Recovery and Deterministic Refinement

Table 8: **Controlled noise-conditioned refinement.** C+T training; the deterministic and stochastic refinements use the same anchor and CFSW decoder. Noise is removed from both training and inference in the deterministic row. Lower is better.

Configuration	Sintel Clean	Sintel Final	KITTI EPE	KITTI Fl-all
Anchor only	1.03	2.32	3.23	10.75
Same CFSW decoder, no noise	1.01	2.29	3.21	10.63
TDFlow, with noise	**0.87**	**2.09**	**2.87**	**9.30**

Table 8 separates the anchor, added decoder, and noise-conditioned learning. Keeping the same CFSW decoder but removing noise changes KITTI EPE/Fl-all only from 3.23/10.75 to 3.21/10.63. Anchored noise improves these to 2.87/9.30, a 10.6% EPE and 12.5% relative Fl-all reduction over the deterministic control. Sintel Clean/Final improves from 1.01/2.29 to 0.87/2.09. The gain therefore cannot be explained only by adding the decoder. The earlier submission's separately reported w/o-TDP diagnostic is recorded in Appendix H.1; it is not pooled with this control.

E.2 Initialization and Anchor Choice

Tables 9 and 10 report the initialization and anchor controls. We provide their detailed interpretation below.

Table 9: **Initialization controls.** C+T training. Randomized branches are trainable; all three are randomized in the last row.

Initialization	Sintel		KITTI	
	Clean	Final	EPE	Fl-all
Default pretrained	0.87	2.09	2.87	9.30
Random DAv2-S	0.88	2.11	2.89	9.35
Random ResNet-18	0.89	2.10	2.88	9.36
Random anchor ViT-S	0.87	2.10	2.88	9.33
Fully from scratch	0.89	2.12	2.89	9.38

Table 10: **Changing the anchor.** C+T training, fixed refinement architecture. This isolates anchor choice, not inference latency.

Anchor / refinement	Sintel		KITTI	
	Clean	Final	EPE	Fl-all
WAFT, anchor only	1.00	2.15	3.10	10.30
Proposed, anchor only	1.03	2.32	3.23	10.75
WAFT + TDFlow	0.87	2.11	2.91	9.33
Proposed + TDFlow	0.87	2.09	2.87	9.30

Pretrained initialization. Table 9 randomizes the depth branch, ResNet-18, and anchor ViT-S separately and jointly. Fully from-scratch training obtains 0.89/2.12 on Sintel and 2.89/9.38 on KITTI, only 0.02/0.03 and 0.02/0.08 above the default. Thus the benefit is not primarily explained by pretrained weights in these controls. Randomized branches are trainable; the default pretrained depth branch is frozen.

Changing the anchor. Table 10 holds the truncated policy and CFSW architecture fixed while switching anchors. WAFT is a stronger standalone anchor, yet the refined models finish close: 2.91/9.33 versus 2.87/9.30 on KITTI. Aggregate EPE alone does not characterize suitability for

refinement. Different spatial error patterns and decoder compatibility are plausible explanations, not isolated conclusions from this table. Dependence on anchor quality remains: local recovery improves the prior but does not erase its errors.

Anchor-only qualitative analysis. The controlled ablation in Table 5 further supports this conclusion. The anchor-only baseline is clearly worse than the full TDFlow model, and removing the truncated diffusion policy consistently degrades performance on both Sintel and KITTI. The qualitative comparison in Fig. 4 shows the same trend: without truncated diffusion, the model produces more outliers around moving objects and motion boundaries, whereas full TDFlow yields cleaner and lower-error flow estimates. These results show that TDFlow acts as an anchor-conditioned residual refinement mechanism, rather than merely relying on the initial flow anchor.

Scope of the controls. The matched no-noise experiment changes the learning input distribution while retaining the refinement architecture. The initialization study changes pretrained weights, and the anchor study changes the initial flow estimator. They probe different sources of improvement and are not substitutes for an accuracy–latency ablation under one fixed implementation.

F Additional Profiling and Measurement Details

All profiled runtimes use a single RTX 4090, batch size 1, and 376×1248 inputs. We exclude data loading, disk I/O, and visualization, while including the complete model forward pass. Ten warm-up iterations precede 50 CUDA-Event measurements. For memory, peak allocated statistics are reset before each forward run; `torch.cuda.max_memory_allocated()` is recorded over five runs. Standard deviations describe these repeated measurements, not variation across hardware platforms.

Table 11: **End-to-end runtime, peak inference memory, and KITTI zero-shot accuracy.** RTX 4090, batch size 1, 376×1248 inputs. Implementation-level measurements. Runtime/memory are means $\pm$ standard deviations; accuracy retains C+T results. Measurement details: Appendix F.

Method	Runtime (ms)↓	Peak memory (GB)↓	KITTI EPE↓	KITTI Fl-all↓
RAFT	53.000 ± 1.782	0.530 ± 0.0005	5.04	17.40
StreamFlow	259.000 ± 3.914	1.525 ± 0.0000	3.85	12.60
VideoFlow-MOF	256.000 ± 6.979	2.337 ± 0.0208	3.89	14.20
MemFlow	214.000 ± 5.132	1.228 ± 0.0004	3.88	13.70
WAFT	131.000 ± 0.120	1.260 ± 0.0385	3.10	10.30
FlowDiffuser	286.000 ± 1.302	2.924 ± 0.0012	3.61	11.80
TDFlow	69.000 ± 0.494	1.210 ± 0.0404	**2.87**	**9.30**

End-to-end efficiency and accuracy. Table 11 reports 69.000 ± 0.494 ms runtime and 1.210 ± 0.0404 GB peak inference memory for TDFlow, with the lowest KITTI zero-shot EPE/Fl-all of 2.87/9.30 among the listed entries. RAFT has lower runtime and memory but higher errors of 5.04/17.40. These are implementation-level comparisons; additional profiling and controls are in Appendix F.

TDFlow runtime and memory. Tables 1 and 11 consistently report 69 ms total runtime and 1.210 GB peak allocated inference memory; the updated decoder latency is 9 ms. With one denoiser evaluation, one-step and complete diffusion-stage latency coincide. We do not infer individual encoder or anchor timings by subtracting this decoder measurement from the end-to-end total. Peak inference memory is distinct from training memory: the previously reported 13.9/5.8 GiB training-memory pair is not the quantity used in the efficiency tables.

Measurement series and interpretation. The 290 ms FlowDiffuser total in Table 1 is the original module-comparison measurement; 286.000 ± 1.302 ms in Table 11 is the subsequent repeated end-to-end profile. These are separately reported measurements, not interchangeable rounding conventions. Accordingly, the $4.2\times$ speedup in the abstract uses $290/69 \approx 4.20$ from Table 1; the repeated profile

in Table 11 gives $286/69 \approx 4.14$. The peak-memory entries in both tables use the repeated-inference measurements. Table 11 summarizes implementation-level results with method-specific inference settings; it does not establish a same-precision architectural speedup. Its accuracy columns retain the reported C+T results; the source experiments do not establish that these scores were all re-evaluated under the profiled inference settings. A paired precision/accuracy check is needed for that stronger interpretation.

Table 12: **Additional same-precision inference profiling.** RTX 4090, batch size 1, 376×1248 inputs. Inference precision is held fixed across methods, with consistent optimization of compatible attention operations. These are separate reported measurements from Table 11.

Method	Runtime (ms)↓	Peak inference memory (GB)↓
SEA-RAFT(L)	$\mathbf{43.186 \pm 3.325}$	0.546 ± 0.003
DPFlow	102.757 ± 7.810	$\mathbf{0.378 \pm 0.001}$
WAFT	73.392 ± 1.497	1.150 ± 0.054
TDFlow	69.000 ± 0.494	1.210 ± 0.040

Additional profiling comparison. Table 12 retains the complementary same-precision comparison. The measured WAFT/TDFlow runtimes are 73.392/69.000 ms, a 6.0% reduction, while TDFlow uses approximately 0.060 GB more peak memory. SEA-RAFT(L) remains faster and DPFlow uses less memory. These results provide additional context for the broader implementation-level comparison without attributing its runtime gaps solely to truncated diffusion.

Table 13: **Extended end-to-end profiling.** Entries retain their reported implementation-level measurements; rows without a $\pm$ term have no reported standard deviation.

Method	Runtime (ms)	Peak memory (GB)
RAFT (Teed & Deng, 2020)	53.000 ± 1.782	0.530 ± 0.0005
SEA-RAFT(L) (Wang et al., 2024)	44.086	0.616
StreamFlow (Sun et al., 2024)	259.000 ± 3.914	1.525 ± 0.0000
VideoFlow-MOF	256.000 ± 6.979	2.337 ± 0.0208
MemFlow (Dong & Fu, 2024)	214.000 ± 5.132	1.228 ± 0.0004
DPFlow (Morimitsu et al., 2025)	91.969	0.436
WAFT (Wang & Deng, 2025)	131.000 ± 0.120	1.260 ± 0.0385
FlowDiffuser (Luo et al., 2024)	286.000 ± 1.302	2.924 ± 0.0012
TDFlow	69.000 ± 0.494	1.210 ± 0.0404

Table 13 retains the broader reported profiles, including SEA-RAFT(L) and DPFlow. Entries without a $\pm$ term do not have a reported standard deviation. Multi-frame methods use a different temporal-input setting from two-frame TDFlow. These measurements complement, rather than replace, the seven-method comparison in Table 11.

G Additional Qualitative Comparisons

The following examples complement the quantitative tables and main-paper visualizations. They compare TDFlow with MemFlow (Dong & Fu, 2024) and DPFlow (Morimitsu et al., 2025), emphasizing motion boundaries, thin structures, and occluded regions. In the highlighted areas, TDFlow gives cleaner flow predictions and more compact error regions. These qualitative observations are consistent with the local refinement and high-resolution feature-alignment analysis.

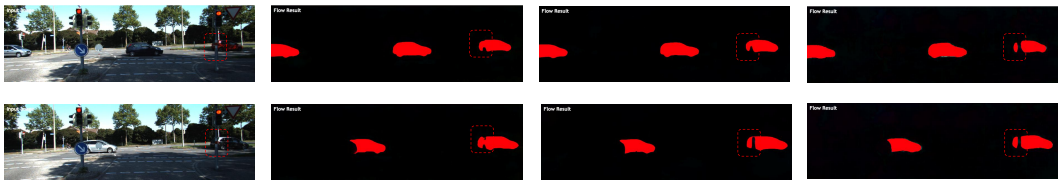


Figure 6: **KITTI qualitative comparison.** MemFlow (Dong & Fu, 2024), DPFlow (Morimitsu et al., 2025), and TDFlow on highlighted challenging regions. TDFlow better preserves object boundaries and reduces local outliers in these examples. Best viewed by zooming in.

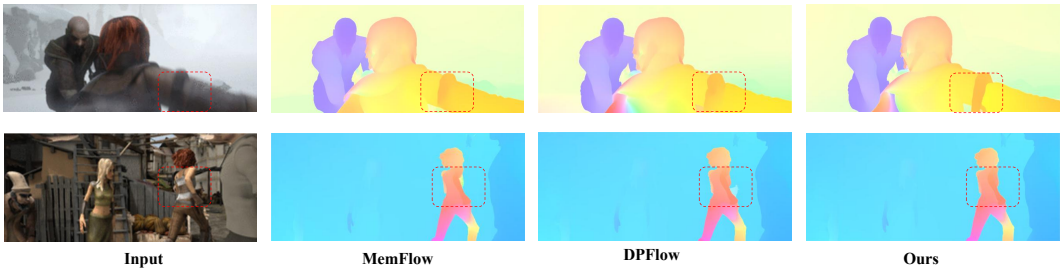


Figure 7: **Sintel qualitative comparison.** MemFlow (Dong & Fu, 2024), DPFlow (Morimitsu et al., 2025), and TDFlow on challenging examples. TDFlow preserves sharper flow boundaries and reduces errors in the highlighted regions. Best viewed by zooming in.

H Limitations and Future Work

Local recovery, not unrestricted generation. TDFlow perturbs an image-conditioned anchor and directly supervises target recovery. The no-noise control supports this learning strategy, but does not establish a mathematical reverse process for ground-truth diffusion, calibrated uncertainty, or explicit multimodal prediction. The ten-seed study measures stochastic *inference* variation only, not training-seed variance or statistical significance across independently trained networks.

Anchor quality and denoising count. The method improves each tested anchor family, but final errors remain dependent on initialization. One denoising evaluation works well for the tested default; the current experiments do not jointly sweep anchor quality, truncation depth, and reverse-step schedules. The lack of benefit for $K > 1$ should not be generalized to weak anchors or full diffusion from pure noise. A controlled study of those interactions would better separate local correction from iterative generative recovery.

Efficiency and deployment. The implementation-level comparison to FlowDiffuser changes denoising count, warping, feature resolution, and inference implementation. It does not isolate truncation as the sole cause of acceleration. In the complementary same-precision comparison (Table 12), the latency difference to WAFT is modest and TDFlow uses slightly more peak memory. SEA-RAFT(L) is faster and DPFlow uses less memory. The reported accuracy tables and timing tables are not a verified paired-precision accuracy–latency operating point; additional accuracy evaluation under the profiled settings is needed. Module timings and implementation-level results must not be used as architecture-only comparisons.

Scope. The study covers supervised two-frame optical flow on established datasets. It does not demonstrate closed-loop autonomous-driving safety, temporal consistency over long videos, or deployment on resource-constrained devices. Generic pretraining and different baseline training recipes are disclosed, but are not eliminated from every benchmark comparison. Extending the framework to temporal modeling and downstream-task evaluation remains future work.

H.1 Original Submission Diagnostic and Matched No-Noise Control

The original submission included a row labeled "DF.-w/o TDP" with Sintel Clean/Final EPE of 0.97/2.17 and KITTI EPE/Fl-all of 3.18/10.46. The subsequent matched no-noise experiment in Table 8 instead reports 1.01/2.29 and 3.21/10.63, keeping the anchor and CFSW architecture fixed while removing noise during learning and inference. These are retained as separately reported experiments. Only the explicitly matched no-noise configuration is used to quantify the effect of noise in the main-paper control. The earlier diagnostic is not pooled with it or treated as a repeated measurement of that configuration.

I Existing Assets and Licenses

We use only publicly available datasets, benchmarks, and pretrained/model assets, and credit their original creators through citations in the main paper. All assets are used for academic research and benchmark evaluation. We do not redistribute any original dataset, benchmark data, or upstream pretrained checkpoint in this submission.

Datasets and benchmarks. FlyingChairs (Dosovitskiy et al., 2015) and FlyingThings3D (Mayer et al., 2016) are used following their official research-use terms. In particular, FlyingChairs and the Scene Flow dataset collection, including FlyingThings3D, are provided for research purposes only and prohibit commercial use. MPI Sintel (Butler et al., 2012) is used under the official MPI-Sintel dataset and benchmark terms, while the underlying Sintel movie assets from the Blender Foundation are released under the Creative Commons Attribution 3.0 license. KITTI (Geiger et al., 2013) is distributed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 license. HD1K (Kondermann et al., 2016) is used through its official benchmark suite and under its benchmark access terms. Spring (Mehl et al., 2023) is based on Blender Foundation movie assets released under Creative Commons Attribution 4.0 terms. TartanAir (Wang et al., 2020) is used under the Creative Commons Attribution 4.0 license for the dataset, while its toolkit is released under the MIT license.

Pretrained and model assets. For feature extraction, we use DepthAnythingV2-Small (Yang et al., 2024), whose Small model is released under the Apache-2.0 license. Other components, including ResNet (He et al., 2016) and DPT (Ranftl et al., 2021), are used as cited model architectures or standard public implementations under their respective upstream terms. We do not redistribute upstream pretrained weights unless explicitly permitted by their licenses.

Compliance. All external assets are used only for academic research, benchmarking, and reproducibility. Third-party datasets and pretrained checkpoints are not included in the manuscript source package. Upon acceptance, any released code will preserve the required attribution and will not redistribute external assets beyond their original license terms.