RESIDUAL MOTION SPLATTING FOR CONTINUOUS SPACE-TIME VIDEO SUPER-RESOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous space-time video super-resolution produces a high-resolution, high-frame-rate video from a low-resolution, low-frame-rate one at arbitrary spatial scales and temporal instants. Classical spatial multi-frame super-resolution has shown that detail beyond a single frame's sampling limit can be recovered by aligning complementary observations with sub-pixel precision and accumulating them on a finer grid. Existing learning-based methods largely depart from this principle: some align and fuse frames only at low resolution before regressing a continuous high-resolution representation, while others interpolate each frame to the output resolution before splatting them. In both cases, the per-frame samples are resampled before fusion, so their sub-pixel positions are lost before frames are combined. Restoring this principle in a learned continuous model raises two challenges: estimating sub-pixel motion at unobserved times, and handling holes and occlusions when splatting a coarse grid onto a finer one. We address the first by predicting per-pixel continuous trajectories that refine linear optical-flow paths with smooth residual motion, using features integrated over the full video sequence. We address the second by directly splatting consecutive low-resolution features in a learned latent space onto a high-resolution grid at their predicted sub-pixel positions with predicted visibility weights, before a decoder resolves remaining ambiguities. This makes temporal interpolation and multi-frame super-resolution a single operation. Our method improves both reconstruction fidelity and perceptual quality on multiple standard benchmarks over the state of the art, while running up to an order of magnitude faster.

1 INTRODUCTION

Spatial frequencies above a single frame's Nyquist limit become recoverable when several observations of the same scene fall at distinct locations between integer pixel positions and accumulate on a finer grid. Classical multi-frame super-resolution (Tsai & Huang, 1984; Irani & Peleg, 1991; Farsiu et al., 2004) is built on this mechanism, and video supplies motion that places every frame with a different spatial phase. Unlike conventional methods (Haris et al., 2020; Xiang et al., 2020; Xu et al., 2021), which fix the spatial and temporal upscaling rates s and r once and for all at training time, continuous space-time video super-resolution (C-STVSR) algorithms learn a single model that can be used at inference time for any values of s and r, including ones not seen during training.

Recent C-STVSR models depart from the classical mechanism of splatting low-resolution pixels directly to their high-resolution sub-pixel positions. Implicit neural representation (INR) methods predict high-resolution representations of each frame *before* splatting them to the interpolated time steps (Chen et al., 2022; 2023b; Kim et al., 2025); recent sinusoidal representation based methods predict over spatio-temporal voxels a parametric representation of a high-resolution video with no explicit motion model (Becker et al., 2026); Gaussian splatting based methods rasterise primitives whose position, spread and colour are regressed from features fused under linear motion (Shi et al., 2026). None of them carries the original low-resolution samples to their own sub-pixel positions under an explicit motion model. We reinstate this mechanism, which SPMC (Tao et al., 2017) introduced for fixed-scale video super-resolution at observed frames, and extend it to unobserved instants: a single splat maps the encoder features onto their sub-pixel coordinates on the high-resolution grid, along continuous trajectories predicted per location.

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.

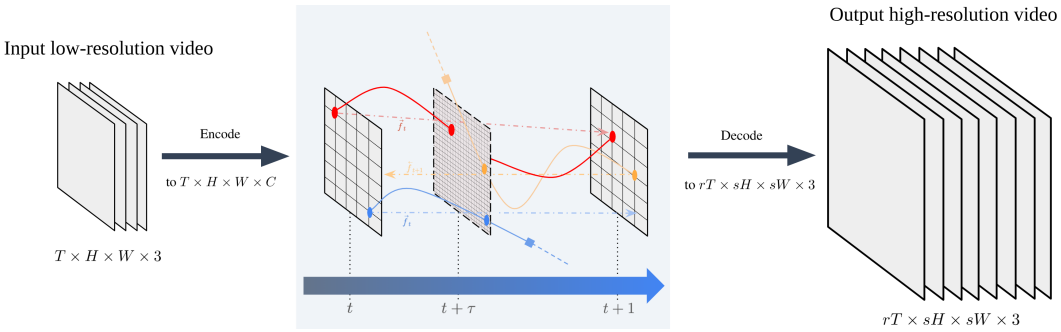


Figure 1: **Overview.** A video encoder turns the low-resolution input into features that carry context from the whole sequence. For each keyframe pair, every low-resolution feature is splatted onto the high-resolution grid at time $t + \tau$ (fine dashed grid) along a trajectory (solid) that corrects the linear optical-flow motion (dash-dotted) with a learned spline, and is weighted by a predicted visibility. Points visible in both keyframes (red) are fused from both, forward from t and backward from $t + 1$; points that disappear (blue) or appear (yellow) between the keyframes are taken from the frame in which they are visible, squares marking where visibility changes. A decoder maps the fused features to RGB frames.

Our method works in four stages: (1) a video encoder produces latent features that capture temporal context over multiple frames; (2) given off-the-shelf forward and backward optical flow estimates, cubic splines with learned parameters are used to predict sub-pixel residual motions at every low-resolution location in temporally interpolated frames; (3) visibility estimates are also predicted at each location to account for effects such as occlusion and disocclusion; (4) the learned features are splatted on the high-resolution interpolated frames, and mapped back onto pixel values by a learned decoder.

In this formulation, restoration fidelity depends on trajectory and visibility accuracy. While classical spatial super-resolution pipelines estimate motion between discrete observed frames, continuous temporal models must predict motion at unobserved intermediate times. To model complex motion, we refine coarse linear flow estimates by predicting residual splines. Leveraging feature maps over a broad temporal context enables the network to adjust trajectories according to scene contents. Unlike closed-form quadratic approaches (Xu et al., 2019; Liu et al., 2020) that impose a uniform constant-acceleration assumption everywhere, our model can learn varying acceleration dynamics. Our contributions are:

- We bring back the classical multi-frame super-resolution principle in a learned continuous model. Low-resolution features are splatted directly onto the high-resolution grid at their own sub-pixel positions, at any instant between input frames, and weighted by a visibility predicted together with their trajectory. Nothing is upsampled or warped at high resolution before fusion, so multi-frame super-resolution and temporal interpolation become a single operation. Existing C-STVSR methods instead fuse either before upsampling or after interpolating each frame to the output resolution.
- Accumulating observations this way requires sub-pixel motion at instants that were never sampled. At each low-resolution coordinate, we predict a continuous cubic B-spline residual on pretrained optical flow, from features propagated across the input sequence. An inference-time experiment (Appendix A) shows that this trajectory benefits from context beyond the two frames around the queried instant: with four frames it recovers most of the full-sequence gain and exceeds a closed-form quadratic model given the same frames.
- On GoPro and Adobe240, our method improves both fidelity and perceptual quality over the state of the art. It generalises to unseen space-time scales, is the best C-STVSR method on arbitrary-scale video super-resolution, in video frame interpolation and is the most robust to unseen input noise. Because no stage runs at high resolution before the splat, it is 6–11$\times$ faster than the C-STVSR methods with public weights.

2 Related Work

Classical Multi-Frame Super-Resolution. Super-resolution from several shifted observations predates learned methods. Frequency-domain registration and reconstruction (Tsai & Huang, 1984), iterative back-projection (Irani & Peleg, 1991) and robust regularised estimators (Farsiu et al., 2004) all rest on the same premise: the observations sample the scene at distinct sub-pixel offsets, and accumulating them on a finer grid recovers aliased frequencies. Burst photography applies the same principle with kernel regression on a fine grid (Wronski et al., 2019), and early learned VSR kept it explicit: the SPMC layer (Tao et al., 2017) maps low-resolution pixels onto the high-resolution grid at their flow-displaced sub-pixel positions, performing motion compensation and upsampling in one step. All of these estimate motion only between frames they actually observe, at a fixed output scale, and degrade sharply where motion estimation fails, which is one reason later learned methods replaced explicit accumulation with implicit feature fusion (Chan et al., 2022a; Liang et al., 2022). In this work, we combine the strengths of both paradigms: we leverage implicit temporal feature extraction (Liang et al., 2022) to construct rich context representations, which are then used to predict robust non-linear trajectories for explicit sub-pixel splatting at unobserved instants.

Fixed-Scale Space-Time Video Super-Resolution. Video Super-Resolution (VSR) and Video Frame Interpolation (VFI) traditionally address spatial and temporal upsampling as independent tasks. VSR frameworks aggregate information across frames with recurrent or transformer backbones (Wang et al., 2019; Chan et al., 2021a; 2022a; Liang et al., 2022; Shi et al., 2022; Tran et al., 2025), aligning them with optical flow, deformable attention (Chan et al., 2021b; 2022a) or cross-attention (Xu et al., 2024). VFI methods synthesise intermediate frames along the temporal axis using flow-based backward warping (Jiang et al., 2018; Bao et al., 2019; Sim et al., 2021) or kernel-based motion compensation (Niklaus et al., 2017). Unified fixed-scale Space-Time Video Super-Resolution (STVSR) models (Haris et al., 2020; Xiang et al., 2020; Xu et al., 2021) fuse the two into single-stage architectures, avoiding the error propagation and flicker of cascaded pipelines. All of them are bound to pre-defined upsampling factors, through fixed pixel-shuffle heads (Shi et al., 2016) or rigid temporal modulation blocks: the output scale and frame rate are decided when the model is trained.

Arbitrary-Scale Spatial Super-Resolution. Continuous spatial scaling was pioneered through implicit neural representations (Chen et al., 2021; Lee & Jin, 2022), which parameterise images as continuous functions over coordinate grids using MLPs or local Fourier bases (Pak et al., 2023). Continuous 2D Gaussian Splatting (Peng et al., 2025; Chen et al., 2025) has since emerged as an efficient alternative, rendering arbitrary spatial grids by forward rasterisation of fitted primitives rather than by point-wise implicit coordinate queries. Extending continuous evaluation to video, Arbitrary-Scale Video Super-Resolution (AVSR) frameworks (Li et al., 2024; Shang et al., 2024) incorporate temporal guidance features, but address spatial scaling only and cannot synthesise intermediate frames.

Continuous Space-Time Video Super-Resolution (C-STVSR). VideoINR (Chen et al., 2022) scales both axes with decoupled spatial and temporal implicit functions, but predicts a backward flow at unobserved times: that field cannot be measured, and at occlusion boundaries it is unconstrained by the observations. MoTIF (Chen et al., 2023b) replaces it with forward trajectories and softmax splatting (Niklaus & Liu, 2020), regressing the displacement at time t from a SIREN; BF-STVSR (Kim et al., 2025) keeps that pipeline and gives the time axis a B-spline basis. Both learn the trajectory outright, unlike the closed-form fits of the VFI lineage, which are cheap and stable on smooth motion but fixed in order: linear (Jiang et al., 2018), quadratic with an acceleration term recovered from a third flow (Xu et al., 2019), and least-squares-rectified quadratic (Liu et al., 2020). Both also evaluate the trajectory at the high-resolution output coordinates, so the splat carries N_{HR} feature tokens, and BF-STVSR estimates it from the two successive frames. A pair fixes a displacement and says nothing about how it changes: acceleration is not observable, so whatever curvature the spline expresses comes from the learned prior rather than from the input. Other approaches have been explored such as using 3D Fourier Fields in V^3 (Becker et al., 2026) to map space-time signals globally to bypass explicit motion, at the cost of detail under large displacement which can exhibit chunk-boundary artefacts from the voxel partitioning their memory footprint requires, or Gaussian splatting approaches such as GS-STVSR (Shi et al., 2026) which regresses Gaussian primitives from features already fused under linear motion rather than transporting the observations. In our work, we instead predict the trajectory at each of the N_{LR} low-resolution coordinates, from features prop-

agated across the whole sequence, and as a residual on the flow estimate rather than as the full displacement.

3 METHOD

Given a low-resolution image sequence $y \in \mathbb{R}^{T\times H\times W\times 3}$ with T RGB frames of size $H \times W$, a spatial upscale factor s and a temporal upscale factor r, C-STVSR aims to reconstruct high-resolution frames at continuous positions in space and time.

Our method solves this problem in four stages. We first estimate bidirectional optical flow across the input RGB sequence with an off-the-shelf estimator (Teed & Deng, 2020); motion in both temporal directions is needed to capture asymmetric dynamics and to resolve regions occluded in either the past or the future. These flows guide feature propagation in a video super-resolution backbone (Liang et al., 2022), which returns latent features $Y \in \mathbb{R}^{T\times H\times W\times C}$: every frame is encoded at its original spatial resolution but carries context from the whole sequence (Section 3.1). For each feature map Y_t, a network predicts (from its neighbouring features maps Y_{t+1} and Y_{t-1}), at every low-resolution position and in both temporal directions, two continuous functions of time: a B-spline correction to the flow-induced linear motion (Section 3.2), and a visibility weight modelling whether the content is still visible at that instant or has been occluded. Both are predicted once per input frame, independently of the output frame rate, and are then evaluated at the intermediate times $t + \tau$ and $t - 1 + \tau$; to produce r frames per interval we take $\tau \in \{\frac{j}{r}\}_{j=0}^{r-1}$. A splat then places the encoded features onto a fixed $4H \times 4W$ grid at the positions their corrected trajectories reach at time τ, weighted by the predicted visibility (Section 3.3), so that temporal interpolation and multi-frame super-resolution happen in a single operation. A decoder finally maps the fused features to pixels, and for $s \neq 4$ the RGB frame is bicubically interpolated up or down to reach the factor s (Section 3.4). Two properties characterise this design. Every quantity is estimated in the low-resolution source domain, so the trajectory and visibility networks run at the input resolution. And the output is continuous in both dimensions: time enters through the spline evaluation at any τ in $[0, 1]$ and space through the final resampling, so one trained model serves arbitrary space-time upsampling.

3.1 VIDEO-AWARE FEATURE EXTRACTION

Estimating interframe trajectories from only two adjacent temporal frames restricts motion modeling to constant velocity. Without access to broader temporal context, the network cannot observe higher-order motion dynamics such as acceleration or curvature; as a result, any non-linear trajectory it predicts relies on learned inductive biases rather than observed evidence. While parameterizations such as the B-spline motion model in BF-STVSR (Kim et al., 2025) provide an effective continuous trajectory formulation, estimating its parameters from adjacent two-frame windows remains a key bottleneck. We retain the continuous B-spline parameterization but estimate it from features that aggregate long-range temporal context and use it to correct flow-based linear motion estimates. Enlarging the temporal context for the trajectory estimator improves reconstruction quality indicating that the additional frames provide usable motion evidence (Appendix A). We adopt RVRT (Liang et al., 2022), a recurrent video super-resolution backbone, to propagate features bidirectionally across the entire sequence using optical flow maps estimated with RAFT (Teed & Deng, 2020). We replace RVRT's original fixed-scale upsampling block with our proposed residual motion multi-frame splatting.

By convention, we use in the rest of this presentation right and left arrows over the corresponding symbols to denote quantities associated with the forward and backward temporal directions, for example $\vec{f}_t$ and $\overleftarrow{f}_t$ for the flows from the frame at time t to the frame at times $t + 1$ (forward) and $t - 1$ (backward) respectively.

3.2 TRAJECTORY AND VISIBILITY ESTIMATION FROM PROPAGATED FEATURES

In the forward direction, a point located at position u_t in the feature map Y_t is mapped by our forward model onto the point

$$\vec{u}_{t+\tau} = u_t + \tau\, \vec{f}_t(u_t) + \vec{s}\big(\tau, \lambda, \vec{\mu}_t(u_t)\big), \tag{1}$$

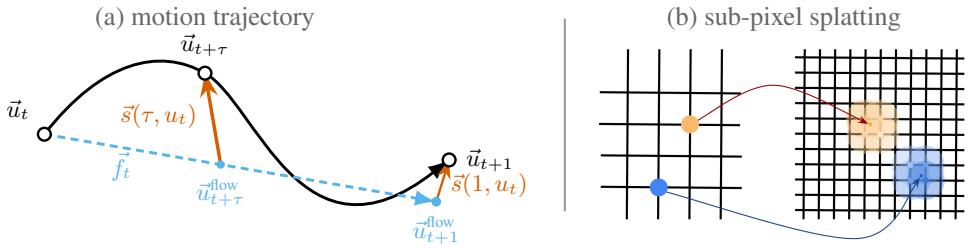


Figure 2: (a) The full forward trajectory decomposed as the linear motion estimate from the optical-flow $\vec{f}_t$ corrected by a B-spline residual $\vec{s}$. (b) An illustration of the sub-pixel splat-and-upscale operation. We splat the content of a low-resolution pixel (left-grid) to a sub-pixel position in the high-resolution image (right-grid). The 4 neighbours of the landing position receive a contribution depending on its distance to the landing position and each is extended by a Gaussian kernel.

at time $t+\tau$ with τ in $[0,1]$, where $\vec{s}$ is the forward residual B-spline (we write $\overleftarrow{s}$ for the backward residual B-spline), parameterised by τ, its bandwidth λ taken constant over all pixel locations and time intervals and learned with the other model parameters, and the coefficients $\vec{\mu}_t(u_t)$ combine the two-dimensional control points and the scalar knots associated with the spline, and depend on time t and pixel location u_t. To make sure the trajectories start at the correct locations, we design the splines to not contribute at the origins ($\tau = 0$ for forward and $\tau = 1$ for backward). The control points and knots are computed for all locations in a single forward pass of the network g_θ, as a $H \times W \times K \times 3$ tensor

$$\vec{\mu}_t = g_\theta(Y_t, Y_{t+1}, \vec{f}_t). \tag{2}$$

Although g_θ operates on the pair (Y_t, Y_{t+1}), both feature maps carry long-range sequence context. Unlike methods with limited temporal support that rely on rigid assumptions (Xu et al., 2019) or learned motion priors (Chen et al., 2023b; Kim et al., 2025), our splines predict plausible trajectories from observed multi-frame dynamics (Appendix A). By formulating these splines as a residual correction to linear optical flow, the network focuses purely on modeling non-linear motion as well as corrects any flow estimation errors (at $\tau = 1$), Figure 2 (a) illustrates this well.

Symmetrically, the backward trajectory is $\overleftarrow{u}_{t-1+\tau} = u_t + (1-\tau)\,\overleftarrow{f}_t(u_t) + \overleftarrow{s}\big(\tau, \lambda, \overleftarrow{\mu}_t(u_t)\big)$, where the backward coefficients are estimated from the backward feature context and flow field by the same network $\overleftarrow{\mu}_t = g_\theta(Y_t, Y_{t-1}, \overleftarrow{f}_t)$.

These positions define where the feature point $Y_t(u_t)$ will be splatted in the time interpolated high-resolution feature maps $X_{t+\tau}$ and $X_{t-1+\tau}$.

Splatting features from several source frames onto a single target grid is a many-to-one map and whenever two contradicting sources, such as an occluder and the surface behind it, land on the same target pixel their contributions are accumulated and the result is a ghost. Resolving this requires a weight per low-resolution input feature deciding which should contribute at every moment in time (Appendix B). Following BF-STVSR (Kim et al., 2025), which predicts its splatting importance jointly with the trajectory, we read the weights from the trajectory as a learned linear projection of the B-spline basis with a final sigmoid layer $\vec{w}_t(u_t, \tau) = P_\varphi(\tau, \lambda, \vec{\mu}_t(u_t))$ and $\overleftarrow{w}_t(u_t, \tau) = P_\varphi(\tau, \lambda, \overleftarrow{\mu}_t(u_t))$. These are $H \times W \times 1$ tensors with values in $(0, 1)$ and weight the features during splatting.

3.3 Splatting video features onto the high-resolution grid

Once the forward and backward trajectories are estimated, we use them to place the low-resolution features Y_t and Y_{t+1} directly onto a fixed high-resolution grid of scale factor 4, producing the fused feature map $X_{t+\tau}$ that the decoder turns into pixels. The input features live on the low-resolution grid of $N_{LR} = HW$ points with normalized coordinates $\{u_n\}_{n=1}^{N_{LR}} = \{(\frac{i}{H}, \frac{j}{W}) : i \in \{0, \dots, H-1\},\ j \in \{0, \dots, W-1\}\}$, and the target features on the high-resolution grid of $N_{HR} = 16HW$ points with coordinates $\{v_n\}_{n=1}^{N_{HR}} = \{(\frac{i}{4H}, \frac{j}{4W}) : i \in \{0, \dots, 4H-1\},\ j \in \{0, \dots, 4W-1\}\}$. We splat to a single $\times 4$ grid, the scale the model is trained at, and resample the decoded result to reach the requested scale (Section 3.4). Three reasons motivate this choice. First,

splatting to a larger grid brings little: each output pixel then receives fewer observations, especially with only two source frames, and the gain of multi-frame reconstruction diminishes as the scale factor grows (Baker & Kanade, 2002; Lin & Shum, 2004; Wronski et al., 2019). Second, for scales below $\times 4$, the finer grid places observations at least as precisely as needed, and filtering it down to the target grid causes no loss of accuracy in our experiments (Table 5). Third, a single scale lets the decoder learn one sampling density rather than several.

In the following we only describe the forward direction but the backward direction is symmetric. Each trajectory is a set of points $\{\vec{u}_{n,t+\tau},\ \forall \tau \in [0,1]\}$ that depends on the source frame Y_t and the starting low-resolution position u_n. Each point of the trajectory generally falls between four neighbouring high-resolution discrete points rather than on one of them, this is the sub-pixel landing position the trajectory is designed to predict. To respect this sub-pixel placement, we distribute the source feature $Y_t(u_n)$ over these four neighbors using bilinear weights, so each neighbor's share depends on its distance to the predicted landing point. Due to the small number of source pixels compared to the number of possible target positions, bilinear weights alone leave holes between splatted samples. We therefore widen each sample's footprint by composing the bilinear kernel with a Gaussian of standard deviation $\sigma_s = 1$ high-resolution pixel, so that target pixels between samples still receive a contribution. Figure 2(b) illustrates the coverage of the resulting kernel κ.

The fused feature at every high-resolution position v is the sum, over both source frames Y_t and Y_{t+1}, of every source point's feature weighted by the visibility maps $\vec{w}$ and $\overleftarrow{w}$ and by the kernel κ evaluated at the distance to its predicted landing position:

$$X_{t+\tau}(v) = \sum_{n=1}^{N_{LR}} \vec{w}(u_n,\tau) Y_t(u_n) \kappa\big(v - \vec{u}_{n,t+\tau}\big) + \sum_{n=1}^{N_{LR}} \overleftarrow{w}(u_n,\tau) Y_{t+1}(u_n) \kappa\big(v - \overleftarrow{u}_{n,t+\tau}\big). \quad (3)$$

The first term accumulates features splatted from frame t along its forward trajectory, the second from frame $t{+}1$ along its backward trajectory. For simplicity, we do not write the contribution normalization where we divide by the total weight landing at v, so $X_{t+\tau}(v)$ is a weighted average of all contributions. This operation builds interpolated frames at continuous times and upscales them in a single pass, adding detail restoration beyond what the video encoder provides. It follows the same principle as classical multi-frame super-resolution (Tsai & Huang, 1984), now in an end-to-end learned latent feature space evaluated at any continuous time $t + \tau$.

3.4 Decoding

A decoder D_ϕ maps the splatted feature map $X_{t+\tau}$ to the high-resolution RGB frame:

$$x_{t+\tau} = D_\phi(X_{t+\tau}) \in \mathbb{R}^{3\times 4H\times 4W}. \quad (4)$$

Splatting and decoding operate at a fixed factor of 4 (Section 3.3); any target scale s is reached by bicubic up or downsampling of $x_{t+\tau}$ in RGB space, so observations are accumulated at their sub-pixel positions before any interpolation.

3.5 Training objective

Let $\mathcal{D} = \{(y^b, x^b)\}_{b=1}^{B}$ denote a batch of B video sequences, where y^b represents the low-resolution input video and x^b denotes the corresponding ground-truth high-resolution video. Each video x^b consists of $N = (r(T-1)+1)s^2HW$ pixel values with $T \times H \times W \times 3$ the shape of y^b and r the temporal scale factor and s the spatial scale factor (we train at $s = 4$ and $r = 8$).

We evaluate the reconstruction error using the Charbonnier distance (Lai et al., 2017):

$$\forall (z_1, z_2) \in \mathbb{R}^2,\ \ell(z_1, z_2) = \sqrt{(z_1 - z_2)^2 + 10^{-6}}, \quad (5)$$

such that the empirical reconstruction risk $R_{\text{rec}}(\Theta)$ over the batch is defined as:

$$R_{\text{rec}}(\Theta) = \frac{1}{BN} \sum_{b=1}^{B} \sum_{n=1}^{N} \ell\big(x^b[n], f(y^b;\Theta)[n]\big)\,, \quad (6)$$

where $f(y^b;\Theta)$ denotes the reconstructed video output parameterized by network weights Θ we optimize.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Implementation. We follow (Becker et al., 2026) and use their hyper-parameter choices for the RVRT (Liang et al., 2022) video encoder, with the embedding dimension set to 90 and 12 attention heads, and train it from scratch end-to-end. We choose to encode the features to have 64 hidden channels. For the trajectory estimator g_θ we use a three-layer SIREN (Sitzmann et al., 2020) with layer dimensions set to 64,64, 256. Following the choice of (Kim et al., 2025), we choose $K = 64$ for the B-spline basis cardinal. We use a Residual U-Net (Zhang et al., 2018c) to parameterize the decoder network D_ϕ.

Training Details. We train the model end-to-end and optimise the model parameters with AdamW($\beta_1 = 0.9, \beta_2 = 0.99$) and use Cosine Annealing scheduler over 500.000 iterations with an initial learning rate of 10^{-4}. A training sample is an input sequence of 14 frames at 64×64 with a fixed temporal scale factor of $r = 8$ and spatial scale factor of $s = 4$. We choose a batch size of 3 and train on a single RTX PRO 6000 Blackwell GPU. The pretrained RAFT flow estimator is kept frozen throughout the first 90% of the iterations and resumes with a ten times smaller learning rate. This first lets the trajectory model train stably and then allow the optical-flow model to get specialized for the task in an end-to-end learning setting.

Datasets. We train our model using the Adobe240 (Su et al., 2017) dataset, a set of 133 RGB videos captured at 240 frames-per-second. Following the split from (Chen et al., 2022) that is used in the works that followed (Chen et al., 2023b; Kim et al., 2025; Shi et al., 2026; Becker et al., 2026) we have 100 videos for training, 16 for validation and 17 for testing. The dataset contains various scenes and camera motion, and is the standard training set for continuous space-time video super-resolution. Low-frame-rate inputs are synthesised by keeping one frame in eight such that the newly consecutive frames contain 30 fps-like motion. We also use Vid4 (Liu & Sun, 2011) and GOPRO (Nah et al., 2017) datasets as test sets for the spatio-temporal task and the REDS (Nah et al., 2019) dataset for the spatial-only task.

4.2 C-STVSR PERFORMANCE

We evaluate first in the training setting ($\times 4$ space, $\times 8$ time), then on unseen scales. Spatial SR is better posed than temporal SR: motion provides sub-pixel-shifted views of the same content, whereas intermediate instants are never observed. A model can only predict a plausible trajectory, and pixel metrics penalise a sharp but offset frame more than a blurred one. We therefore report PSNR computed over luminance (PSNR-Y) alongside LPIPS (Zhang et al., 2018a) (SSIM over luminance is reported in Appendix E). FPS is measured on full GoPro frames (Appendix E.1).

Our method is best on GoPro and Adobe240 on both subsets and metrics (Table 1), improving on V^3 by up to 0.6 dB and on V^3-Large by up to 0.5 dB with 20% fewer parameters. The gain is largest in LPIPS: 12–17% lower than V^3, whereas V^3's $\approx$2 dB gain over BF-STVSR brought only 3–10%. Replacing the trajectory at inference changes PSNR by up to 1.5 dB but LPIPS by under 0.006 (Appendix A): the trajectory controls *where* content lands, our reconstruction stage (sub-pixel splatting and the HR decoder) *how sharply* it is rendered.

Trajectory errors therefore cost PSNR, not sharpness; in one clip our sharper crops trail V^3 in PSNR until a 2–6 pixel shift is compensated (Appendix F). V^3 shares our RVRT encoder and RAFT fine-tuning, pointing to the reconstruction stage as the source of the gap, and trains with far more compute (2.5M iterations, batch 16, 16 GPUs vs. 500k, batch 3, 1 GPU). On Vid4, whose inputs are twice as far apart as in training, our method is on par with V^3. We are an order of magnitude faster than V^3 and about $6\times$ faster than MoTIF and BF-STVSR.

Extreme scale factors. Following (Chen et al., 2023b; Kim et al., 2025), we test combinations of spatial and temporal scales on GoPro, and add a two-step baseline that runs EMA-VFI (Zhang et al., 2023) followed by LIIF (Chen et al., 2021). Table 2 shows that our method is best for every combination, including temporal scales up to twice and spatial scales up to three times those used in training. We evaluate V^3 with the authors' official inference code and released weights.

Arbitrary spatial and temporal scales. On arbitrary-scale video SR (Appendix D.1), our method outperforms all C-STVSR methods in both PSNR and SSIM at every scale from $\times 2$ to $\times 8$, and surpasses the dedicated ST-AVSR (Shang et al., 2024) in PSNR at $\times 2$ and $\times 8$ despite being trained

Table 1: C-STVSR at $\times 4$ spatial scale, $\times 8$ temporal (GoPro, Adobe240) and $\times 2$ temporal (Vid4). We report PSNR-Y↑ / LPIPS↓. *Center* evaluates frames 1, 5, 9 of each 9-frame sequence, *Average* all output frames. Methods marked † have no public weights, so we report their published PSNR-Y only. We highlight the best result in **bold** and the second best with underline.

Method	# Par.	FPS	Vid4	GoPro *Center*	GoPro *Average*	Adobe *Center*	Adobe *Average*
MoTIF	12.6M	2.5	25.76 / 0.3024	31.04 / 0.1554	30.04 / 0.1587	30.63 / 0.1408	29.82 / 0.1457
BF-STVSR	13.5M	2.2	25.84 / 0.2931	31.17 / 0.1507	30.22 / 0.1543	30.83 / 0.1313	30.12 / 0.1350
GS-STVSR†	12.7M	—	26.04 / —	31.33 / —	30.35 / —	31.13 / —	30.35 / —
V^3	13.7M	1.3	26.76 / **0.2649**	32.96 / 0.1458	32.26 / 0.1485	32.91 / 0.1186	32.29 / 0.1223
V^3- Large†	20.6M	—	26.82 / —	33.09 / —	32.36 / —	33.08 / —	32.45 / —
Ours	16.5M	**14.5**	**26.87** / 0.2760	**33.40 / 0.1261**	**32.54 / 0.1306**	**33.55 / 0.0986**	**32.78 / 0.1032**

Table 2: Performance comparison on the C-STVSR baselines for extreme scale factors on GoPro dataset. Results are evaluated using PSNR-Y (dB) and SSIM-Y metrics. All frames are interpolated by a scaling factor specified on the table and metrics calculated across all interpolated frames. We highlight the best result in **bold** and the second best with underline.

Temporal Scale	Spatial Scale	EMA-VFI + LIIF	VideoINR	MoTIF	BF-STVSR	V^3	Ours
×6	×4	30.64 / 0.8850	30.78 / 0.8954	31.56 / 0.9064	31.70 / 0.9083	33.79 / 0.9394	**34.33 / 0.9460**
	×6	28.17 / 0.8126	25.56 / 0.7671	29.36 / 0.8505	29.45 / 0.8520	30.99 / 0.8867	**31.34 / 0.8951**
	×12	24.85 / 0.7052	24.02 / 0.6900	25.81 / 0.7330	25.80 / 0.7295	26.66 / 0.7538	**26.80 / 0.7589**
×12	×4	27.90 / 0.8263	27.32 / 0.8141	27.77 / 0.8230	28.07 / 0.8287	28.77 / 0.8432	**29.12 / 0.8540**
	×6	26.49 / 0.7748	24.68 / 0.7358	26.78 / 0.7908	27.07 / 0.7963	27.79 / 0.8160	**28.08 / 0.8276**
	×12	24.16 / 0.6918	23.70 / 0.6830	24.72 / 0.7108	24.88 / 0.7104	25.56 / 0.7301	**25.74 / 0.7381**
×16	×4	26.56 / 0.7904	25.81 / 0.7739	25.98 / 0.7758	26.39 / 0.7840	26.68 / 0.7837	**27.10 / 0.7993**
	×6	25.54 / 0.7503	23.86 / 0.7123	25.34 / 0.7527	25.81 / 0.7619	26.14 / 0.7675	**26.40 / 0.7804**
	×12	23.68 / 0.6828	22.88 / 0.6659	23.88 / 0.6923	24.22 / 0.6955	24.67 / 0.7084	**24.79 / 0.7160**

on another dataset. In the temporal-only setting (Appendix D.2), it leads V^3 by 2.4 dB PSNR and, on the synthesized frames alone, beats the dedicated interpolators RIFE (Huang et al., 2022) and EMA-VFI (Zhang et al., 2023) by 2.2 dB.

Robustness to degradations. Appendix D.3 evaluates inputs corrupted by Gaussian noise, and by noise followed by H.264 compression, although no method is trained on degraded inputs. Our method outperforms all C-STVSR methods at every noise level and loses only 0.15 dB up to $\sigma{=}0.01$, against 0.5 dB for V^3. Under compression, all methods are on par.

Qualitative results. Figures 3 and 4 show frames from the Adobe240 test set at intermediate times between two input frames, none of which the model sees. Under fast motion (Figure 3), prior methods either duplicate the moving object or blur it; under parallax (Figure 4), they lose a thin background object seen through the foreground. Our method keeps both sharp and at the position shown in the ground truth. Appendix F shows a fast-moving bus, where the other methods also lose thin structures such as a street light and power lines (Figure 7). A video comparing the different methods on the $\times 4$ spatial $\times 8$ temporal task is provided in the supplementary materials.

5 Conclusion

Discussion. We showed that the principle of classical multi-frame super-resolution, accumulating several frames at their sub-pixel positions on a finer grid, carries over to learned continuous space-time video super-resolution. What we splat, however, are not raw observations but features that a video encoder has propagated across the whole sequence, so each splatted sample already carries temporal context beyond its own frame before a decoder maps the fused features to pixels. This combines the two paradigms that learned video super-resolution had separated: implicit feature propagation supplies temporal context, while explicit sub-pixel accumulation keeps each sample at its own position until frames are fused. As a result, a single splat both interpolates in time and gathers detail from neighbouring frames, improving fidelity and perceptual quality at a fraction of the computational cost. Natural next steps are a visibility head decoupled from the motion basis, and more samples per target pixel, by splatting more frames or several trajectories per pixel (Hu et al., 2022), to go beyond our fixed $\times 4$ splat, which is limited by the two frames it combines (Baker

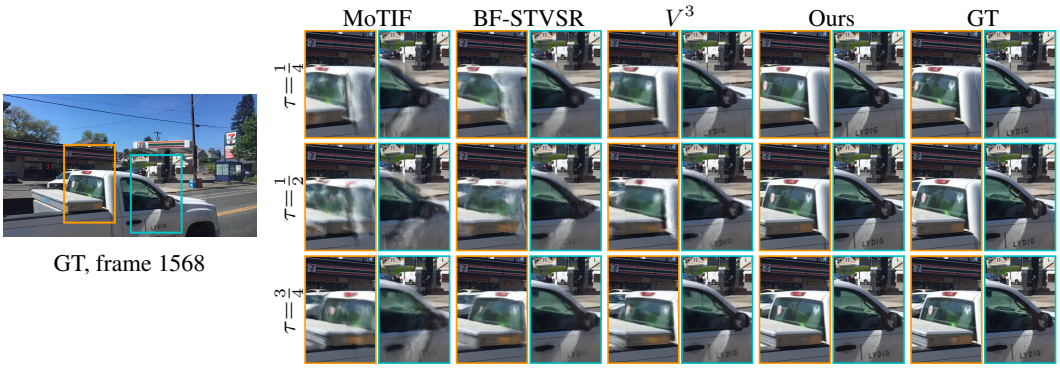


Figure 3: Fast motion on Adobe240 IMG_0015 ($\times$4 space, $\times$8 time), at the intermediate times between input keyframes 1564 and 1572. A truck drives across the frame. Ours keeps the back of the truck sharp and the "LYDIG" text on its door readable at every intermediate time. V^3 blurs the truck and adds straight lines where the borders between its space-time processing blocks show in the output, while MoTIF and BF-STVSR produce several overlapping copies of it. Recovered details (or lack thereof) may be better appreciated by zooming on the figure.

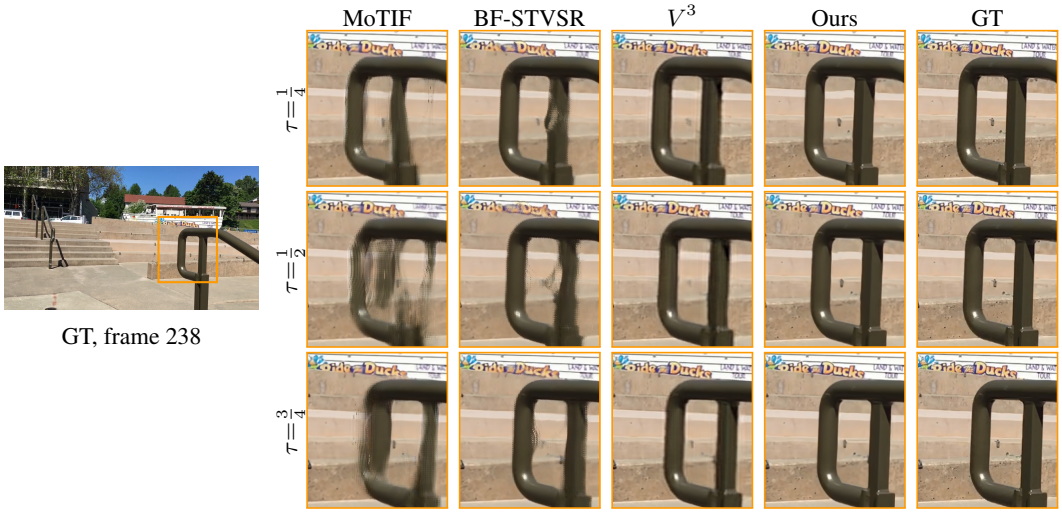


Figure 4: Parallax on Adobe240 IMG_0004a ($\times$4 space, $\times$8 time), at the intermediate times between input keyframes 234 and 242. The scene pans around with a rail in the foreground and a staircase in the background is visible through the railing loop. Our method keeps the bar intact and correctly pans the background at every intermediate time. V^3 blurs the rail and background through the loop, BF-STVSR breaks the bar where it overlaps the background, and MoTIF smears the two layers together. Recovered details (or lack thereof) may be better appreciated by zooming on the figure.

& Kanade, 2002; Wronski et al., 2019). Edge-aligned splat kernels (Wronski et al., 2019) could also fill holes without blurring sub-pixel detail.

Limitations. (1) Visibility is read from the trajectory's B-spline basis, following BF-STVSR (Kim et al., 2025), so it is as smooth in time as the motion and tied to the point's own trajectory rather than to its occluder. This limits its gain (Appendix B) and leaves errors at the boundaries of moving objects. (2) Under complex motion, trajectories can place content a few pixels off (Appendix F), which costs PSNR even when the frame is sharp. (3) Like all evaluated methods, ours does not recover detail from compressed inputs (Appendix D.3).

AI USE STATEMENT

The research direction, method, experiments and conclusions of this work are the authors' own. We used generative AI tools as a discussion partner to help organise and refine our ideas, to suggest experiments and ablations, which we then designed, ran and checked ourselves; every reported result comes from these experiments. We also used them to help implement parts of the code, to search for related work, to obtain feedback on the manuscript, and to draft and edit parts of the text. All AI-assisted code was reviewed and tested by the authors, and all text was checked and revised by the authors. The remaining tasks listed as requiring disclosure (synthetic data generation, mathematical claims or proofs, dataset cleaning, translation) are not applicable. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work uses only publicly available video benchmarks (Adobe240, GoPro, REDS and Vid4), under their respective licenses and terms of use. No new data was collected and no human subjects were involved. While spatial detail is recovered by aggregating sub-pixel observations across neighbouring frames, intermediate frames are synthesized at time stamps that were never captured. The motion they depict follows a learned trajectory model and is not observed. In applications where frame content carries evidential weight, such as forensic, medical or surveillance analysis, synthesized intermediate frames should be clearly distinguished from captured ones and interpreted with appropriate validation.

REPRODUCIBILITY STATEMENT

The method is described in Section 3: the encoder in Section 3.1, the trajectory and visibility estimation in Section 3.2, with the B-spline construction in Appendix C, the splatting in Section 3.3 and the decoder in Section 3.4. Architecture hyperparameters, the training objective and schedule, the hardware and the dataset splits are given in Section 4.1. The evaluation protocol for each benchmark is given with its results: the *Center* and *Average* subsets in Section 4.2, arbitrary-scale SR in Appendix D.1, frame interpolation in Appendix D.2, the noise and compression settings, including the codec parameters, in Appendix D.3, the additional metrics and the VMAF encoding in Appendix E, and runtime measurement in Appendix E.1. The inference-time trajectory study is described in Appendix A, and the visibility ablation, including the definition of the occluded areas, in Appendix B. We re-evaluated every baseline with public weights using its official code under this protocol; where our results matched the published ones, we report the published values.

REFERENCES

Simon Baker and Takeo Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1167–1183, 2002.

Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Alexander Becker, Julius Erbach, Dominik Narnhofer, and Konrad Schindler. Continuous space-time video super-resolution with 3D Fourier fields. In *International Conference on Learning Representations (ICLR)*, 2026.

Kelvin C. K. Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. BasicVSR: The search for essential components in video super-resolution and beyond. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021a.

Kelvin C. K. Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Understanding deformable alignment in video super-resolution. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021b.

Kelvin C. K. Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. BasicVSR++: Improving video super-resolution with enhanced propagation and alignment. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022a.

Kelvin C. K. Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. Investigating tradeoffs in real-world video super-resolution, 2022b.

Du Chen, Liyi Chen, Zhengqiang Zhang, and Lei Zhang. Generalized and efficient 2D Gaussian splatting for arbitrary-scale super-resolution. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Hao-Wei Chen, Yu-Syuan Xu, Min-Fong Hong, Yi-Min Tsai, Hsien-Kai Kuo, and Chun-Yi Lee. Cascaded local implicit transformer for arbitrary-scale super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023a.

Yi-Hsin Chen, Si-Cun Chen, Yi-Hsin Chen, Yen-Yu Lin, and Wen-Hsiao Peng. MoTIF: Learning motion trajectories with local implicit neural functions for continuous space-time video super-resolution. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023b.

Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Zeyuan Chen, Yinbo Chen, Jingwen Liu, Xingqian Xu, Vidit Goel, Zhangyang Wang, Humphrey Shi, and Xiaolong Wang. VideoINR: Learning video implicit neural representation for continuous space-time super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Mengyu Chu, You Xie, Jonas Mayer, Laura Leal-Taixé, and Nils Thuerey. Learning temporal coherence via self-supervision for GAN-based video generation. *ACM Transactions on Graphics*, 39(4):75:1–75:13, 2020.

Duolikun Danier, Fan Zhang, and David Bull. FloLPIPS: A bespoke video quality metric for frame interpolation. In *Picture Coding Symposium (PCS)*, 2022.

Sina Farsiu, M. Dirk Robinson, Michael Elad, and Peyman Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344, 2004.

Muhammad Haris, Greg Shakhnarovich, and Norimichi Ukita. Space-time-aware multi-resolution video enhancement. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Qiqi Hou, Abhijay Ghildyal, and Feng Liu. A perceptual quality metric for video frame interpolation. In *European Conference on Computer Vision (ECCV)*, 2022.

Ping Hu, Simon Niklaus, Stan Sclaroff, and Kate Saenko. Many-to-many splatting for efficient video frame interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *European Conference on Computer Vision (ECCV)*, 2022.

Michal Irani and Shmuel Peleg. Improving resolution by image registration. *CVGIP: Graphical Models and Image Processing*, 53(3):231–239, 1991.

Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Yuxuan Jiang, Ho Man Kwan, Tianhao Peng, Xiaoqing Zhu, Joel Sole, and David Bull. HIIF: Hierarchical encoding based implicit image function for continuous super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Eunjin Kim, Hyeonjin Kim, Kyong Hwan Jin, and Jaejun Yoo. BF-STVSR: B-Splines and Fourier—best friends for high fidelity spatial-temporal video super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep Laplacian pyramid networks for fast and accurate super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Jaewon Lee and Kyong Hwan Jin. Local texture estimator for implicit representation function. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1929–1938, 2022.

Yinxiao Li, Pengchong Jin, Feng Yang, Ce Liu, Ming-Hsuan Yang, and Peyman Milanfar. Comisr: Compression-informed video super-resolution. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Zekun Li, Hongying Liu, Fanhua Shang, Yuanyuan Liu, Liang Wan, and Wei Feng. SAVSR: Arbitrary-scale video super-resolution via a learned scale-adaptive network. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. Toward a practical perceptual video quality metric. Netflix Technology Blog, 2016. URL `https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652`.

Jingyun Liang, Yuchen Fan, Xiaoyu Xiang, Rakesh Ranjan, Eddy Ilg, Simon Green, Jiezhang Cao, Kai Zhang, Radu Timofte, and Luc Van Gool. Recurrent video restoration transformer with guided deformable attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Zhouchen Lin and Heung-Yeung Shum. Fundamental limits of reconstruction-based superresolution algorithms under local translation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):83–97, 2004.

Ce Liu and Deqing Sun. A Bayesian approach to adaptive video super resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

Yihao Liu, Liangbin Xie, Li Siyao, Wenxiu Sun, Yu Qiao, and Chao Dong. Enhanced quadratic video interpolation. In *European Conference on Computer Vision Workshops (ECCVW)*, 2020.

Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. NTIRE 2019 challenge on video deblurring and super-resolution: Dataset and study. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

Byeonghyun Pak, Jaewon Lee, and Kyong Hwan Jin. B-Spline texture coefficients estimator for screen content image super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Long Peng, Anran Wu, Wenbo Li, Peizhe Xia, Xueyuan Dai, Xinjie Zhang, Haoze Sun, Renjing Pei, Yang Wang, Yang Cao, and Zheng-Jun Zha. Pixel to Gaussian: Ultra-fast continuous super-resolution with 2D Gaussian modeling. In *International Conference on Learning Representations (ICLR)*, 2025.

Wei Shang, Dongwei Ren, Wanying Zhang, Yuming Fang, Wangmeng Zuo, and Kede Ma. Arbitrary-scale video super-resolution with structural and textural priors. In *European Conference on Computer Vision (ECCV)*, 2024.

Mingyu Shi, Xin Di, Long Peng, Boxiang Cao, Anran Wu, Zhanfeng Feng, Jiaming Guo, Renjing Pei, Xueyang Fu, Yang Cao, and Zheng-Jun Zha. GS-STVSR: Ultra-efficient continuous spatio-temporal video super-resolution via 2D Gaussian splatting. *arXiv preprint arXiv:2604.18047*, 2026.

Shuwei Shi, Jinjin Gu, Liangbin Xie, Xintao Wang, Yujiu Yang, and Chao Dong. Rethinking alignment in video super-resolution transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. XVFI: eXtreme video frame interpolation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. Deep video deblurring for hand-held cameras. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, 2020.

Dinh Phu Tran, Dao Duy Hung, and Daeyoung Kim. VSRM: A robust Mamba-based framework for video super-resolution. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Roger Y. Tsai and Thomas S. Huang. Multiframe image restoration and registration. In *Advances in Computer Vision and Image Processing*, volume 1, pp. 317–339. JAI Press, 1984.

Xintao Wang, Kelvin C. K. Chan, Ke Yu, Chao Dong, and Chen Change Loy. EDVR: Video restoration with enhanced deformable convolutional networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

Bartlomiej Wronski, Ignacio Garcia-Dorado, Manfred Ernst, Damien Kelly, Michael Krainin, Chia-Kai Liang, Marc Levoy, and Peyman Milanfar. Handheld multi-frame super-resolution. *ACM Transactions on Graphics*, 38(4):28:1–28:18, 2019.

Xiaoyu Xiang, Yapeng Tian, Yulun Zhang, Yun Fu, Jan P. Allebach, and Chenliang Xu. Zooming Slow-Mo: Fast and accurate one-stage space-time video super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Gang Xu, Jun Xu, Zhen Li, Liang Wang, Xing Sun, and Ming-Ming Cheng. Temporal modulation network for controllable space-time video super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Kai Xu, Ziwei Yu, Xin Wang, Michael Bi Mi, and Angela Yao. Enhancing video super-resolution via implicit resampling-based alignment. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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

Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T. Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019.

Guozhen Zhang, Yuhan Zhu, Haonan Wang, Youxin Chen, Gangshan Wu, and Limin Wang. Extracting motion and appearance via inter-frame attention for efficient video frame interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018a.

Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b.

Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018c.

A IMPORTANCE OF TEMPORAL INFORMATION FOR MOTION ESTIMATION

We examine whether the trajectory estimator uses temporal context beyond the two frames bracketing the interpolated instant. To test this, we vary only the temporal context seen by the trajectory estimator g_θ at inference. For each keyframe pair, we run a second encoder pass restricted to T keyframes centred on the pair, and feed these features to g_θ; the splines it predicts then drive the splatting step. All other features, including the splatted ones, keep the full input window and are therefore identical across all cases. To isolate the trajectory, we use the variant of our model without visibility estimation (uniform splatting weights, Appendix B), trained under the same setting as the full model. We compare against two closed-form references built on the same optical flow: *Linear Motion*, which scales the flow along straight lines, and *Quadratic Motion*, the constant-acceleration model of QVI (Xu et al., 2019), which uses the same four frames as $T{=}4$.

Figure 5 reports the results. With only a two frame context, our learned trajectory already improves over linear motion on every clip. Performance then jumps once each keyframe of the pair sees a neighbour on its far side: at $T{=}4$, our trajectory recovers 96% of the full-window gain over linear motion, against 52% for QVI given the same frames, and outperforms it on every video. From $T{=}6$ on, results start to have diminishing returns. The trained estimator therefore relies on context beyond two frames.

Across all trajectory models, including linear and quadratic motion, LPIPS stays between 0.1318 and 0.1373, while PSNR varies by up to $1.48\,\text{dB}$ (Table 3): replacing the trajectory at inference leaves perceptual quality essentially unchanged, and the differences in PSNR reflect where content is placed rather than how it is rendered.

Figure 5: PSNR ↑ on interpolated GoPro frames ($\times 4$ spatial, $\times 8$ temporal) as a function of the number of frames used to encode the features passed to the trajectory estimator. The splatted features always come from the full sequence. *Linear Motion*: linear motion from optical flow. *Quadratic Motion*: the 4-frame constant-acceleration model of QVI (Xu et al., 2019). *Full Window*: the unmodified model.

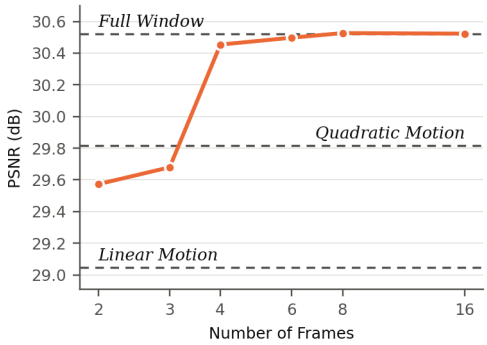


Table 3: PSNR↑ / LPIPS↓ on interpolated GoPro frames ($\times 4$, $\times 8$) for each trajectory model of Figure 5. We notice in **bold** the best values and in *italic* the worst.

Trajectory	PSNR	LPIPS
Linear Motion	*29.05*	0.1344
Quadratic Motion	29.82	*0.1373*
Ours, $T{=}2$	29.57	0.1342
Ours, $T{=}4$	30.45	**0.1318**
Ours, $T{=}6$	30.50	0.1327
Ours, $T{=}8$	**30.53**	0.1329
Ours, $T{=}16$	30.52	0.1329
Ours, Full Window	30.52	0.1329

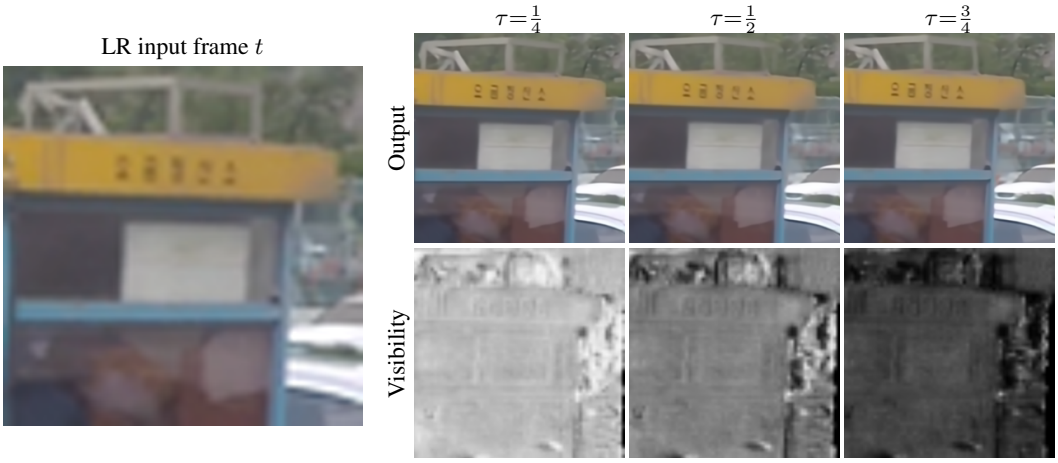


Figure 6: **Visibility over time.** Top: our output at three intermediate times between input frames t and $t{+}1$. Bottom: the visibility weight of the features of frame t, splatted along their trajectories to the output time $t{+}\tau$, so it shows where frame t contributes to each output (bright: high weight). Motion boundaries such as object edges are down weighted the most. As τ grows, the weight decreases and frame t contributes less.

B Visibility Maps

When features from two input frames are warped to the same intermediate time, the frames can disagree about what is visible: an object may hide part of the background in one frame and reveal it in the other. Without visibility weights, the overlapping features are simply averaged. Because this averaging happens in feature space, the decoder can still resolve many of these conflicts, which already makes the model fairly robust to it. Visibility weights make the choice explicit, letting the model rely on whichever frame actually sees each region. Averaged over whole frames, the improvement is therefore small but consistent across all datasets (Table 4), also because occlusions cover only a small part of each image. We therefore also evaluate only on occlusion boundary areas selected where a RAFT flow based forward-backward consistency check fails. In these occluded areas alone, the gain is about twice as large, which is where the mechanism is expected to act.

Table 4: **Effect of visibility weighting** (PSNR-Y, dB). *Uniform*: overlapping warped features are simply averaged. Both models share the same architecture and training setup. The last column scores only the occluded areas of GoPro.

	GoPro	Adobe240	Vid4	GoPro (occluded areas)
Uniform	32.47	32.69	26.78	27.75
Visibility	**32.54**	**32.78**	**26.87**	**27.90**

C B-Spline curves

We use the cubic cardinal B-spline β^3, obtained by recursive convolution of the box function $\beta^0(x) = \mathbf{1}_{[-\frac{1}{2},\frac{1}{2})}(x)$ as $\beta^n = \beta^{n-1} * \beta^0$:

$$\beta^3(x) = \begin{cases} \frac{2}{3} - x^2 + \frac{|x|^3}{2}, & |x| < 1, \\ \frac{(2-|x|)^3}{6}, & 1 \le |x| < 2, \\ 0, & |x| \ge 2. \end{cases} \tag{7}$$

It is non-negative, supported on $[-2, 2]$ and C^2. A trajectory is built from K atoms, each translated by a knot ξ_k, dilated by λ_k and weighted by a two-dimensional control coefficient c_k in $\mathbb{R}^2$ (one component per image axis, sharing the knot). Any linear combination of such atoms is C^2 in time, so every trajectory has continuous velocity and acceleration.

The uncentered spline is:

$$\rho(\tau;\lambda,\mu_t(u)) = \sum_{k=1}^{K} c_{t,k}(u)\,\beta^3\big(\lambda_k\,(\tau-\xi_{t,k}(u))\big) \in \mathbb{R}^2, \tag{8}$$

and the residual used in the trajectories of Section 3.2 subtracts its value at $\tau = 0$, the time of the source frame Y_t:

$$s(\tau;\lambda,\mu_t(u)) = \rho(\tau;\lambda,\mu_t(u)) - \rho(0;\lambda,\mu_t(u)). \tag{9}$$

Hence $s(0) = 0$ and every trajectory starts exactly at u_t, whereas $s(1)$ is left free, so the residual can correct the optical-flow endpoint $u_t + \vec{f}_t(u_t)$. Unlike scaling the residual by τ, centering pins the start point without damping the correction near it. The forward residual is $\vec{s}(\tau,\lambda,\vec{\mu}_t(u)) = s(\tau;\lambda,\vec{\mu}_t(u))$. The backward residual $\overleftarrow{s}(\tau,\lambda,\overleftarrow{\mu}_t(u)) = s(1-\tau;\lambda,\overleftarrow{\mu}_t(u))$ is read at $1-\tau$, the time elapsed since its source frame, so it vanishes at $\tau = 1$, where the backward trajectory starts. The visibility weights of Section 3.2 follow the same convention: the backward weight $\overleftarrow{w}_t$ is read at $1-\tau$.

D Additional Experiments

D.1 Arbitrary-Scale Video Super-Resolution

Following (Becker et al., 2026), we evaluate our method on the arbitrary-scale video super-resolution task (temporal factor 1). We compare against C-STVSR methods (Chen et al., 2022; 2023b; Kim et al., 2025; Becker et al., 2026), the arbitrary-scale image SR methods LTE (Lee & Jin, 2022), CLIT (Chen et al., 2023a) and HIIF (Jiang et al., 2025) applied per frame on an RDN (Zhang et al., 2018b) backbone, and the dedicated arbitrary-scale video SR method ST-AVSR (Shang et al., 2024), on the 30 REDS (Nah et al., 2019) validation videos (100 frames at 1280×720) with metrics computed on RGB following (Shang et al., 2024).

Table 5 shows that our method is the best C-STVSR model at every scale, with the largest margins at $\times 2$ and $\times 3$. This holds even though these outputs are obtained by resampling our $\times 4$ reconstruction (Section 3.4), which supports splatting on a fixed grid. Unlike ST-AVSR, which is trained on REDS (Shang et al., 2024), all C-STVSR methods, including ours, are trained on Adobe240 and evaluated here across datasets. Despite this, our method surpasses ST-AVSR in PSNR at $\times 2$ and $\times 8$ and remains within 0.6 dB of it at the other scales. Results of competing methods are taken from (Becker et al., 2026).

Table 5: Spatial video super resolution (AVSR) on the REDS validation set (PSNR-RGB↑ / SSIM-RGB↑). Methods marked with † indicate arbitrary-scale *image* SR applied frame-by-frame for comparison; ‡ marks an arbitrary-scale *video* SR method. We highlight the best result in **bold** and the second best with underline.

Method	$\times 2$	$\times 3$	$\times 4$	$\times 6$	$\times 8$	# Par.
Bicubic	31.51 / 0.911	26.82 / 0.788	24.92 / 0.713	22.89 / 0.622	21.69 / 0.574	—
RDN-LTE †	34.73 / 0.943	30.73 / 0.866	28.75 / 0.804	26.56 / 0.718	25.24 / 0.669	22.5 M
RDN-CLIT †	34.63 / 0.942	30.63 / 0.865	28.63 / 0.801	26.43 / 0.714	25.14 / 0.661	37.7 M
RDN-HIIF †	34.57 / 0.942	30.59 / 0.864	28.60 / 0.799	26.44 / 0.712	25.15 / 0.659	23.2 M
ST-AVSR ‡	36.91 / **0.969**	**33.41/0.937**	**31.03 / 0.897**	**27.89 / 0.812**	26.04 / **0.746**	27.9 M
VideoINR	31.59 / 0.900	30.04 / 0.852	28.13 / 0.791	25.27 / 0.687	23.46 / 0.619	11.3 M
MoTIF	31.03 / 0.898	30.44 / 0.862	28.77 / 0.807	25.63 / 0.698	25.12 / 0.664	12.6 M
BF-STVSR	34.72 / 0.946	31.17 / 0.881	29.11 / 0.820	26.78 / 0.728	25.40 / 0.668	13.5 M
V^3	36.53 / 0.963	32.31 / 0.908	29.92 / 0.849	27.39 / 0.754	25.96 / 0.690	13.7 M
V^3- Large	36.70 / 0.964	32.53 / 0.911	30.13 / 0.853	27.54 / 0.760	26.10 / 0.696	20.6 M
Ours	**37.09** / 0.967	33.13 / 0.921	30.49 / 0.863	27.63 / 0.764	**26.11** / 0.698	16.5M

D.2 Video Frame Interpolation

We evaluate pure temporal upsampling on Adobe240 by feeding 30 fps sequences at full spatial resolution ($s = 1$) and synthesizing 240 fps output ($r = 8$). Besides the C-STVSR methods, we

Table 6: Temporal-only VSR with $\times 1$ spatial scale and $\times 8$ temporal scale on Adobe240. We report PSNR / SSIM on RGB outputs. "Interp." averages only the 7 synthesized frames between consecutive inputs. †Frame-interpolation methods copy input frames unchanged, where C-STVSR methods reconstruct the keyframes.

Method	All frames	Interp. only
VideoINR	24.45 / 0.712	–
MoTIF	28.09 / 0.843	–
BF-STVSR	29.37 / 0.867	–
V^3	<u>33.43</u> / <u>0.936</u>	–
RIFE†	41.81 / 0.923	30.64 / 0.911
EMA-VFI†	42.16 / 0.929	<u>31.04</u> / <u>0.918</u>
Ours	**35.79 / 0.950**	**33.21 / 0.943**

compare against two video frame interpolation (VFI) methods, RIFE (Huang et al., 2022) and EMA-VFI (Zhang et al., 2023), using their official pretrained weights, trained on Vimeo-90K Xue et al. (2019) whereas all C-STVSR methods, ours included, are trained on Adobe240. VFI methods return the input frames unchanged, while C-STVSR methods reconstruct them, so the two families are compared on different columns of Table 6. Over all output frames, our method leads every C-STVSR method, by 2.4 dB PSNR over V^3. On the seven synthesized frames between consecutive inputs, it surpasses RIFE and EMA-VFI by 2.6 and 2.2 dB PSNR.

D.3 C-STVSR UNDER DEGRADATIONS

Following (Becker et al., 2026), we evaluate how C-STVSR methods trained without degradations, apart from the space-time downscaling, handle noise and compression. Since the codec settings of (Becker et al., 2026) are not specified, we re-run the study for all methods under a common protocol. Zero-mean Gaussian noise of standard deviation σ is added to the input frames normalised to $[-1, 1]$, which are then clipped and quantised to 8 bits. In the compressed setting, the same noisy frames are encoded as an H.264 stream (libx264, CRF 23, `medium` preset, YUV 4:2:0) containing only the low-frame-rate input. Only the input is degraded; outputs are compared with the clean ground truth.

All methods are fairly robust to noise (Table 7), but not equally so. The three splatting-based methods, MoTIF, BF-STVSR and ours, lose at most 0.17 dB up to $\sigma{=}0.01$ and degrade by almost the same amount at every noise level, whereas V^3, which decodes from the features of the nearest input frame without warping them, loses 0.5 dB at $\sigma{=}0.01$. Since V^3 and our method share the same RVRT encoder, this points to the reconstruction stage. Splatting forms each output feature as a weighted average of observations from several frames, which share the scene content but carry independent noise, so the average keeps the content and partly cancels the noise. As a result, our margin over V^3 grows from 0.11 dB on clean inputs to 0.62 dB at $\sigma{=}0.025$. At $\sigma{=}0.05$, all methods lose 1.3–1.6 dB.

Compression dominates the degradation (Table 8): all methods lose more than 2 dB and end up on par. H.264 quantises the high-frequency transform coefficients most coarsely, so at this rate much of the fine detail that multi-frame super-resolution relies on is removed from the input, and blocking and ringing artefacts are added in its place. A model trained only on clean inputs has no way to tell these artefacts apart from real content, so it reconstructs them rather than removing them. Recovering detail from compressed video requires training on compressed inputs, as done by dedicated compressed-VSR methods (Li et al., 2021; Chan et al., 2022b); none of the evaluated methods, ours included, is trained this way. The codec also explains why the noise level has almost no further effect: quantisation removes most of the added noise along with the detail, so every noise level produces nearly the same encoded input.

Table 7: Robustness to Gaussian input noise on Vid4 ($\times 4$ spatial, $\times 2$ temporal). We mark the best result in **bold** and the second best underlined.

Method	Clean	σ=0.005	σ=0.0075	σ=0.01	σ=0.025	σ=0.05
MoTIF	25.76	25.70	25.66	25.59	25.14	24.45
BF-STVSR	25.84	25.80	25.75	25.69	25.26	24.52
V^3	<u>26.76</u>	<u>26.61</u>	<u>26.42</u>	<u>26.24</u>	<u>25.69</u>	<u>25.21</u>
Ours	**26.87**	**26.79**	**26.78**	**26.72**	**26.31**	**25.57**

Table 8: Robustness to noise followed by H.264 compression on Vid4 ($\times 4$ spatial, $\times 2$ temporal). All methods are on par.

Method	σ=0.005	σ=0.0075	σ=0.01	σ=0.025	σ=0.05
MoTIF	23.56	23.55	23.56	23.53	23.47
BF-STVSR	23.63	23.62	23.62	23.60	23.53
V^3	23.88	23.87	23.87	23.85	23.79
Ours	23.89	23.89	23.88	23.87	23.81

E Additional metrics

E.1 Runtime measurement.

FPS is measured during the GoPro evaluation itself, on full-resolution test clips (320×180 input, 1280×720 output) rather than crops, on a single RTX 6000 Ada GPU. Each clip is processed in temporal windows, and only the model's forward pass is timed: frame loading, host-to-device transfer, metric computation and output writing all fall outside the timer. A warm-up before the first clip absorbs one-off costs such as compilation and autotuning. Context frames recomputed at window boundaries are included in the model time (for V^3 and Ours). FPS is the total number of output frames divided by the accumulated model time over the whole test set.

V^3 reports a speed advantage over prior C-STVSR methods, measured on $14\times80\times80$ crops. We could not reproduce this advantage on full frames: its tiled encoder and per-patch decoder must then cover the entire 320×180 input, and its throughput drops to 1.3 FPS.

E.2 Perceptual and temporal coherence metrics

Following BF-STVSR (Kim et al., 2025), Table 9 reports four additional video metrics, computed offline on the saved output frames with the same protocol. VFIPS (Hou et al., 2022) and FloLPIPS (Danier et al., 2022) are perceptual metrics designed for frame interpolation: VFIPS is computed on non-overlapping 12-frame segments, and FloLPIPS on all consecutive frame pairs after conversion to 8-bit YUV 4:2:0. tOF (Chu et al., 2020) measures temporal consistency as the difference between the optical flows of the output and ground-truth videos. VMAF (Li et al., 2016) is a video quality metric fitted to human opinion scores; as in BF-STVSR, both the output and the ground truth are first encoded with H.264 (`libx264`, default quality, YUV 4:2:0, 30 fps) and the decoded videos are scored, so that the metric reflects the video as it would actually be delivered.

Our method is best on every metric except tOF on GoPro, where it is within 0.007 of V^3 (0.377 vs. 0.370), and it is best on Adobe240 (0.395 vs. 0.401). Temporal coherence is therefore on par with V^3, even though V^3 represents the video with space-time Fourier fields, which vary smoothly in time by construction, whereas we synthesise each frame by splatting along predicted trajectories. The perceptual metrics favour our method more clearly: FloLPIPS is 11–12% lower than V^3 on both datasets, consistent with the LPIPS gains of Table 1.

Video encoding for VMAF measurement.

Following the protocol from (Kim et al., 2025), we encode both the reconstructed and the ground-truth sequences to H.264 with `libx264` at 30 frames-per-second in `yuv420p`, at ffmpeg's default rate factor, and compute VMAF on the luma plane of the *decoded* streams. The codec is therefore part of the measurement for this metric: the score reflects the reconstruction as it would be delivered, not the network output in isolation.

Table 9: Performance comparison on the one-stage C-STVSR baselines on GoPro and Adobe240 datasets. Results are evaluated using VFIPS (Hou et al., 2022) (negated so that higher is better), FloLPIPS (Danier et al., 2022), tOF (Chu et al., 2020), and VMAF (Li et al., 2016) metrics. All frames are interpolated by a factor of $\times 4$ in the spatial axis and $\times 8$ in the temporal axis. In **bold** and underline we respectively indicate the first and second best methods.

Method	GoPro				Adobe			
	VFIPS↑	FloLPIPS↓	tOF↓	VMAF↑	VFIPS↑	FloLPIPS↓	tOF↓	VMAF↑
VideoINR	81.13	0.151	0.519	57.96	81.15	0.145	0.574	67.08
MoTIF	81.89	0.156	0.517	59.82	81.61	0.144	0.607	68.40
BF-STVSR	83.01	0.151	0.480	61.06	84.04	0.132	0.498	70.82
V^3	88.21	0.141	**0.370**	70.88	88.97	0.113	0.401	78.40
Ours	**88.50**	**0.126**	0.377	**71.76**	**90.24**	**0.099**	**0.395**	**79.40**

E.3 Standard luminance SSIM and the RGB-PSNR

We share Table 10 that contains the standard SSIM computed on luminance as well as a PSNR computed on RGB frames. We recall that evaluating metrics on the only luminance channel is permissive to several artifacts such as color bleeding as the necessary information to discern them is contained in the chromatic channels that are ignored.

Table 10: C-STVSR at $\times 4$ spatial scale, $\times 8$ temporal (GoPro, Adobe240) and $\times 2$ temporal (Vid4). PSNR-RGB↑ / SSIM-Y↑. Methods marked † have no public weights, so we report their published SSIM-Y only. We highlight the best result in **bold** and the second best with underline.

Method	Params (M)	Vid4	GoPro *Center*	GoPro *Average*	Adobe *Center*	Adobe *Average*
MoTIF	12.6	24.23 / 0.774	29.52 / 0.888	28.53 / 0.877	29.07 / 0.884	28.27 / 0.875
BF-STVSR	13.5	24.35 / 0.777	29.67 / 0.890	28.74 / 0.880	29.31 / 0.888	28.60 / 0.880
GS-STVSR†	12.7	— / —	— / 0.892	— / 0.882	— / 0.891	— / 0.883
V^3	13.7	25.23 / 0.819	31.49 / 0.923	30.77 / 0.919	31.39 / 0.922	30.73 / 0.917
V^3- Large†	20.6	— / —	— / 0.925	— / 0.921	— / 0.924	— / 0.919
Ours	16.5	**25.34 / 0.826**	**31.95 / 0.929**	**31.07 / 0.924**	**32.06 / 0.931**	**31.26 / 0.925**

F Additional results

IMG_0012 (Figure 7) contains difficult motion: the camera pans non-uniformly while a large bus passes close to it. Our results look clearly sharper than those of V^3, but their PSNR is lower at $t{=}\frac{1}{2}$: 15.3–19.9 dB for ours against 18.8–22.1 dB for V^3 over the three crops, whereas our LPIPS is about half of theirs (0.190–0.223 against 0.385–0.484). The reason is position, not appearance: our output is shifted by 2–6 pixels from the ground truth, and PSNR strongly penalizes a small shift of sharp edges, while a blurred output is penalized less. To measure this, we search, for each crop and each method, the integer translation within ± 12 pixels that maximizes PSNR against the ground truth, and compute PSNR after applying it. Our PSNR then rises to 27.2–30.0 dB, while V^3 only reaches 19.0–22.8 dB, since shifting cannot recover blurred detail. On IMG_0015 (Figure 3) and IMG_0004a (Figure 4) the best shift is zero, and our method has the highest PSNR without any correction.

The same pattern appears at the dataset level (Table 1): on GoPro and Adobe240 at $\times 8$, our PSNR-Y gain over V^3 is moderate (0.28–0.64 dB), while our LPIPS is 12–17% lower. This is consistent with sharp outputs that PSNR penalizes for small displacements but that a perceptual metric, which is more tolerant to such shifts, rewards.



Figure 7: Qualitative comparison on Adobe240 IMG_0012 ($\times$4 space, $\times$8 time), at the intermediate times between input keyframes 1720 and 1728. MoTIF and BF-STVSR break the bus front into fragmented patches and V^3 blurs it with a visible seam; ours preserves the window frames and the back of the bus. Thin structures are the hardest case: the baselines lose the streetlight and most of the power lines (MoTIF and BF-STVSR drop the pole entirely at $\tau=\frac{1}{4}$), whereas ours keeps the pole, the lamp arm and the individual wires at every intermediate time. Recovered details (or lack thereof) may be better appreciated by zooming on the figure.