Revisiting Learning-based Video Motion Magnification for Real-time Processing

Anonymous authors
Paper under double-blind review

Abstract

Video motion magnification is a technique to capture and amplify subtle motion in a video that is invisible to the naked eye. The deep learning-based prior work successfully models outstanding quality better than conventional signal processing-based ones. However, it still lags behind real-time performance, which prevents it from being extended to various online systems. In this paper, we revisit the first learning-based model and present experimental analyses, in particular on the identification of redundant components, the insertion of spatial bottlenecks, and the trade-off relationship between channel reduction and layer addition. By integrating the findings of each experiment, we present a real-time, deep learning-based motion magnification model that achieves a computational speed ranging from a minimum of **2.7**× to a maximum of **34.9**× **faster** than existing learning-based methods, while maintaining comparable generation quality to prior arts. To the best of our knowledge, this is the first learning-based motion magnification model that runs in real-time on Full-HD resolution videos even without ad hoc quantization.

1 Introduction

Subtle motions, almost invisible to humans' naked eyes, are typically missed in the video. However, sensing such motions from the video is an irreplaceable modality for important applications, *e.g.*, visualizing the health status of infrastructures (Chen et al., 2014; 2015b;a; 2017) and buildings (Cha et al., 2017), sound (Davis et al., 2014), the motion of hot air (Xue et al., 2014), and medical signals like a human pulse (Balakrishnan et al., 2013; Tveit et al., 2016; Janatka et al., 2018). Video motion magnification (Liu et al., 2005; Wu et al., 2012; Wadhwa et al., 2013; 2014; Oh et al., 2018) is a technique that captures and amplifies such small motions to make them recognizable to the human eyes.

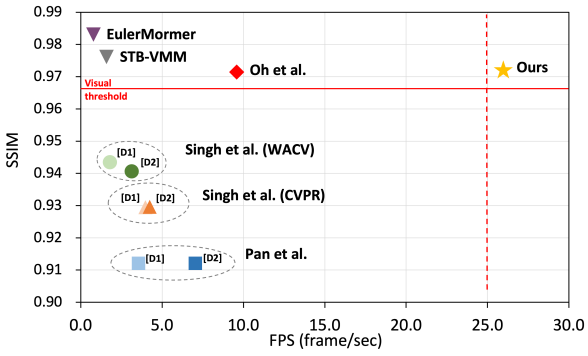


Figure 1: **Computational speed comparison of learning-based video motion magnification models.** Only our model exceeds the standard frame rate of real-time application (*i.e.*, 25 FPS), while achieving favorable motion magnification quality. See Table 3 for details.

Video motion magnification is initially pioneered by signal processing based methods (Wu et al., 2012; Wadhwa et al., 2013; 2014; Zhang et al., 2017; Takeda et al., 2018; 2019; 2020; 2022). Unfortunately, these methods share fundamental limitations derived from signal processing theories, as proved in Wu et al. (2012); Wadhwa et al. (2013), *e.g.*, noise sensitivity, magnification bounds, and limitations to occlusion/disocclusion of objects. Oh et al. (2018) tackle these limitations and demonstrate breakthrough results, by proposing the first learning-based model and the synthetic training dataset. Upon this success, subsequent works also take the learning based approach (Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023; Pan et al., 2024; Gao et al., 2022) and the synthetic training dataset from Oh et al. (2018) as their supervision signal (Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023; Gao et al., 2022).

Despite the success of these learning-based methods, the prior methods do not meet real-time processing. As shown in Fig. 1, their computational speed is still below 10 frames per second (FPS) for Full-HD resolution videos even on decent GPUs. This prevents it from being practically used in various online applications, *e.g.*, structural vibration monitoring (An & Lee, 2022) and robotic surgery (Fan et al., 2021), unattainable unless real-time. In the existing real-time applications, the speed requirement is dealt with in an ad hoc way, *e.g.*, reducing input resolution, or by assuming that the core motion magnification algorithm would run in real-time (Fan et al., 2021).

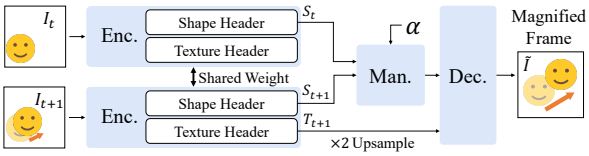
In this regard, we aim to develop a real-time motion magnification model by revisiting Oh et al. (2018) as a baseline. We first introduce the experimental analyses for the components of the baseline, examining their importance and functionality. The insights identified from these experimental analyses of the baseline model can be summarized as follows: 1) The encoder functions well even with a single linear layer, 2) Adding a spatial bottleneck on the decoder makes the model fast, and 3) The reduction of channels has a detrimental effect on the output quality, but this can be compensated by the addition of layers on the decoder.

Integrating these insights, we develop a real-time video motion magnification model , which achieves a computational speed ranging from at least **2.7**$\times$ to a maximum of **34.9**$\times$ **faster** than existing learning-based methods (Oh et al., 2018; Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023; Pan et al., 2024), while maintaining generation quality comparable to the baseline (Oh et al., 2018) and the recent method based on the transformer architectures (Lado-Roigé & Pérez, 2023) in both quantitative and qualitative evaluations. To the best of our knowledge, this is the first learning-based motion magnification model that runs in real-time on Full-HD (FHD; 1920×1080) resolution videos with a capability of high quality synthesis.

2 Related Work

Liu et al. (2005) pioneered Lagrangian video motion magnification, which relies on explicit motion estimation by optical flow and image warping. Later, Wu et al. (2012) coined the Eulerian approach that exploits intensity changes as a way of motion representation (Freeman et al., 1991a), and have become the mainstream of motion magnification; thus, we focus on reviewing this line of work. The Eulerian approaches (Wu et al., 2012; Wadhwa et al., 2013; 2014; Zhang et al., 2017; Takeda et al., 2018; Oh et al., 2018; Takeda et al., 2019; 2020; 2022; Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023; Pan et al., 2024) mainly consist of three stages: (a) Extracting a motion representation of each frame by a spatial decomposition; (b) Temporal filtering to capture motion of interest and amplifying the filtered signals with optional denoising; and (c) Recomposing the magnified frames from the amplified representations. Those works can be categorized according to main contributions: motion representation (Wu et al., 2012; Wadhwa et al., 2013; 2014; Oh et al., 2018; Takeda et al., 2020; Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023; Pan et al., 2024) in (a) and (c), and temporal or denoising filters (Zhang et al., 2017; Takeda et al., 2018; 2019; 2022) in (b). Pillars of the motion magnification work established motion representations. Wu et al. (2012) model Eulerian motion representation by a first-order Taylor expansion, which results in applying Laplacian pyramid decomposition as a spatial decomposition. Subsequent works (Wadhwa et al., 2013; 2014) employ alternative wavelet filters, such as complex steerable pyramid (Freeman et al., 1991b). These hand-designed spatial decompositions are derived from traditional signal processing techniques, including the theories of polynomial, Fourier, and wavelet series, and show elegant modeling of motion as a shift of intensity signal. However, the modeling assumptions inherited by those signal processing theories limit their working regimes to pure translational motion and often yield artifacts in the regions where the theories do not support, *e.g.*, newly appearing or disappearing signals that frequently appear in real-world near occlusion/disocclusion of objects. Further, the magnitudes of their magnified motions are theoretically bounded as proved by themselves (Wu et al., 2012; Wadhwa et al., 2013).

To address these, Oh et al. (2018) propose the first learning-based approach that models motion representations by convolutional neural networks (CNN) and learns a spatial decomposition for motion magnification. They demonstrate significantly fewer artifacts, more robustness against noise and occlusion/disocclusion, and the linearity between the input amplification factor and the resulting magnified motion, which previous signal processing-based methods suffered from. Based on the development Oh et al. (2018), modifications in network architecture have been explored (Singh et al., 2023b; Lado-Roigé & Pérez, 2023; Pan et al., 2024;



Module	Input	Network Components	Output
(a) Enc.	$[H, W, 3]$	Conv× 2, Resblk× 3, (d), (e)	See (d), (e)
(b) Man.	$[H/2, W/2, 32]$	Conv× 2, Resblk× 1	$[H/2, W/2, 32]$
(c) Dec.	$[H/2, W/2, 64]$	Conv× 2, Resblk× 9	$[H, W, 3]$
(d) Shape header	$[H/2, W/2, 32]$	Conv× 1, Resblk× 2	$[H/2, W/2, 32]$
(e) Texture header	$[H/2, W/2, 32]$	Conv× 1, Resblk× 2	$[H/4, W/4, 32]$

Figure 2: **Overall architecture and specification of the baseline.** The baseline (Oh et al., 2018) consists of three modules: the encoder (Enc.), manipulator (Man.), and decoder (Dec.). Encoders are weight-shared. Given the two input frames, I_t and I_{t+1}, the encoder takes each frame and outputs shape representation, S_i, and texture representation, T_i. The manipulator takes S_t and S_{t+1} and magnifies the motion by multiplying the amplification factor α. The decoder reconstructs the magnified frame $\tilde{I}$.

Wang et al., 2024a;b) to enhance the video frame generation performance of the learning-based method. However, all these endeavors have been underexplored in the computational perspective and are not affordable to real-time processing.

Distinguished from the above line of motion representations, another series of works propose temporal filters (Zhang et al., 2017; Takeda et al., 2018; 2022). They deal with degradation caused by large motion, noise, or drift, by designing dedicated temporal filters that change motion frequency of interest. Thus, their methods are effective for their targeted motions but unknown for other scenarios. Also, they are mainly built on the motion representation of Wadhwa et al. (2013); thereby, they share the same limitations, a bit mitigated though. Therefore, their scope is independent of the motion representation works (Wu et al., 2012; Wadhwa et al., 2013; 2014; Oh et al., 2018; Takeda et al., 2020; Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023; Pan et al., 2024; Wang et al., 2024a;b), including ours.

Recently, the interests in real-time motion magnification have been increased. Fan et al. (2021) present a robot surgery module that localizes blood vessels by magnifying pulsatile motion, but the method does not meet the real-time requirement despite their online application scenario. Singh et al. (2023a) propose a lightweight version of the baseline (Oh et al., 2018), but they fall far short of the standard frame rate for real-time applications (*i.e.*, 25 FPS) while exhibiting noticeable quality degradation.

In this regard, we aim to develop a real-time motion magnification model. To address this, we revisit and analyze Oh et al. (2018) as a baseline for two main reasons. First, as we can see in Fig. 1, Oh et al. (2018) demonstrate the fastest computation speed among existing learning-based methods while maintaining favorable quality, making it a logical starting point. Second, many recent learning-based models (Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023; Gao et al., 2022; Wang et al., 2024a;b) still adopt the training procedures and datasets introduced by Oh et al. (2018), establishing the work as the standard baseline.

3 Preliminary

In this section, we present preliminary backgrounds of the video motion magnification and the learning-based motion magnification and setup *the baseline*.

3.1 Video Motion Magnification

We first briefly describe the video motion magnification problem. To give intuition, we explain with an image intensity profile f undergoing motion field $\delta(\mathbf{x}, t)$ over time t as: $I(\mathbf{x}, t) = f(\mathbf{x}+\delta(\mathbf{x}, t))$, where $I(\mathbf{x}, t)$ is the observed intensity of an image (frame) at position $\mathbf{x}$ and time t. Then, the goal of motion magnification is to synthesize the motion magnified image $\tilde{I}(\mathbf{x}, t)$ as:

$$\tilde{I}(\mathbf{x}, t) = f(\mathbf{x} + (1 + \alpha)\delta(\mathbf{x}, t)), \tag{1}$$

where α is the amplification factor. In reality, the underlying image $f(\cdot)$ and motion $\delta(\cdot)$ are hidden and complicatedly entangled; thus, those manipulation requires decomposition of motion $\delta(\cdot)$ from $f(\cdot)$ which is fundamentally challenging.

To address this, as mentioned in Sec. 2, Eulerian motion magnification techniques (Wu et al., 2012; Wadhwa et al., 2013; 2014; Oh et al., 2018; Takeda et al., 2020; Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023)

apply spatial decompositions, *e.g.*, Laplacian pyramids (Wu et al., 2012) or CNN (Oh et al., 2018), to transform the image $I(\mathbf{x}, t)$ to a more favorable form to approximately separate motion information, *e.g.*, $I(\mathbf{x}, t) \rightarrow T(\mathbf{x}) + S(\delta(\mathbf{x}, t))$, where $T(\cdot)$ and $S(\cdot)$ represent texture and shape representations (Oh et al., 2018), respectively. Conceptually, the texture $T(\cdot)$ and the shape $S(\cdot)$ can be understood as proxy representations of the underlying profile $f(\cdot)$ and the residual depending on motion $\delta(\cdot)$, respectively.

The following procedure, extracting motion of interest on $S(\delta(\mathbf{x}, t))$ from the sum $T(\mathbf{x}) + S(\delta(\mathbf{x}, t))$, is typically implemented by temporal bandpass filters under the assumption that the motion signal of interest is within the passband of the filters. After extracting $S(\delta(\mathbf{x}, t))$, we can manipulate it to be $(1+\alpha) \cdot S(\delta(\mathbf{x}, t))$ and add it back to $T(\mathbf{x})$, which allows to have the motion magnified image as

$$\tilde{I}(\mathbf{x}, t) \approx R[T(\mathbf{x}) + (1 + \alpha) \cdot S(\delta(\mathbf{x}, t))], \tag{2}$$

where $R[\cdot]$ denotes necessary reconstruction operations corresponding to respective spatial decompositions depending on the methods (Wu et al., 2012; Wadhwa et al., 2013; 2014; Oh et al., 2018; Takeda et al., 2020; Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023) that revert back to image domain.

3.2 Baseline: Learning-based Motion Magnification

Our goal is to build an efficient model based on the deep motion magnification by Oh *et al.* (Oh et al., 2018). The nature of modeling the motion magnification functions (*i.e.*, spatial decomposition, temporal filtering and magnification, and frame reconstruction) is reflected in the baseline as well, and those correspond to the encoder, manipulator, and decoder, respectively. For convenience, we call the input features of the decoder as *motion representations*.

We denote the baseline model as $\mathcal{G}(\cdot)$ and $I(\mathbf{x}, t) = I_t$ for simplicity. The model is learned to synthesize a magnified image $\tilde{I}(\mathbf{x}, t)$ from consecutive frames $\{I_t, I_{t+1}\}$ and an amplification factor α as input: *i.e.*, $\tilde{I}(\mathbf{x}, t) = \mathcal{G}(I_t, I_{t+1}, \alpha)$. Also, it can be generalized for multi-frames as $\mathcal{G}(\{I_t\}_{t=1}^{N}, \alpha)$ during inference time, where N denotes a window size (refer to Fig. 2 for details).

One of the fundamental challenges in this field is that, unfortunately, there exist no ground-truth motion magnified videos in real world to train the model. To address this issue, Oh et al. (2018) propose a synthetic training dataset, which is still adopted by most of the recent works (Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023; Gao et al., 2022).

4 Experimental Setting

Before proceeding with the development, we analyze the baseline method to gain a better understanding of the learning-based motion magnification method. This section provides a detailed account of the experimental settings employed in Architecture Design (Sec. 5) and Results (Sec. 6).

4.1 Training and Evaluation Details

For training, we employ the synthetic dataset proposed by the baseline (Oh et al., 2018). This dataset includes pairs of consecutive frames, ground-truth magnified frames, and amplification factors. We split the dataset into training and validation sets, consisting of $95,000$ and $5,000$ data samples, respectively.

We utilize the validation set of the training dataset to evaluate the magnification quality of the models. In terms of metric, we use image similarity measures, *e.g.*, the Structural Similarity Index Measure (SSIM). We also use Learned Perceptual Image Patch Similarity (LPIPS)[1] (Zhang et al., 2018) to evaluate perceptual aspects. We also measure architectural parameters (Params), floating-point operations per second (FLOPs), and frames per second (FPS) for inference wall-clock time on a single NVIDIA RTX 3090 GPU.

4.2 Calibration on Quality Evaluation Metric

Since similarity metrics (e.g., SSIM) measured on synthetic data often exhibit a domain gap with human perception in real-world scenarios, we conducted a human study to establish a reliable quality standard.

[1]We measure LPIPS using the VGG16 network (Simonyan & Zisserman, 2014).

We asked 20 participants to rate the visual similarity of magnified real videos compared to reference frames on a 0–5 scale. By correlating these scores with SSIM values across various model variants, we identified that a score of "3" (Adequate Similarity) corresponds to an SSIM of approximately 0.966. Based on this, we establish a *visual threshold* of 0.966 as the primary criterion for acceptable generation quality. Detailed experimental setups, participant instructions, and calibration results are provided in Appendix F.

4.3 Tests for Noise Handling

Effective noise handling is critical for separating subtle motion from artifacts (Wu et al., 2012). Following prior protocols (Oh et al., 2018; Byung-Ki et al., 2024), we conduct two complementary tests on synthetic data: a *noise test* and a *subpixel test*. The noise test evaluates robustness against varying noise levels. However, since a model might achieve high SSIM by merely denoising (replicating inputs) rather than magnifying motion, the subpixel test is essential. It verifies the capability to isolate and magnify minute motions (ranging from 0.01 to 1 pixel) to a target of 10 pixels, ensuring the model accurately amplifies motion instead of simply suppressing small changes.

5 Architecture Design

We analyze the baseline model (Oh et al., 2018) to identify redundant components and architectural bottlenecks, which helps clarify the crucial characteristics for the video motion magnification task. Our analyses are threefold: 1) learning-to-remove redundant components (Sec. 5.1), 2) spatial bottleneck for frame reconstruction (Sec. 5.2), and 3) channel reduction with adding layers (Sec. 5.3). By integrating these insights, we introduce a real-time learning-based motion magnification architecture (Sec. 5.4).

5.1 Learning-to-Remove Redundant Components

Oh et al. (2018) reported an interesting finding that linear approximations of their shape encoder resemble manually-designed linear filters in signal processing (Wu et al., 2012; Wadhwa et al., 2013), *e.g.*, directional edge detector, Laplacian operator, and corner detector. From the observation of resemblance with linear filters, we pose a question, "Do we really need non-linear layers in the encoder?". This motivates us to conduct a main component ablation experiment to analyze the importance or redundancy of neural components. By identifying these components, we can find better trade-offs between computational efficiency and quality, facilitating the deployment in resource-constrained environments without compromising prediction accuracy.

Inspired by Dror et al. (2021) and Huang & Wang (2018), we design a *learning-to-remove method* that uses a learnable switch parameter to force a layer to transition to a desirable state, specifically, without unnecessary components. The method begins with parameterizing around the component of interest as:

$$F(x) = (1-\omega)A_o(x) + \omega A_t(x), \quad 0 \leq \omega \leq 1, \tag{3}$$

where ω is a learnable switch parameter, $A_o(\cdot)$ and $A_t(\cdot)$ are element functions, *e.g.*, layers. $A_o(\cdot)$ is an original function standing for a pre-existing state, and $A_t(\cdot)$ a target function standing for a desirable state after the removal, *i.e.*, a much simpler element function. Initially, the learnable parameter ω is set to zero, so that $F(x) = A_o(x)$. If the learnable parameter ω approaches to 1, then $F(x) \to A_t(x)$. Along with this parameterization, we aim to remove and analyze unnecessary components while preserving the task quality. We fine-tune the target architecture, *i.e.*, the baseline, with the learnable parameter ω by minimizing the following objective function,

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \lambda_{rm}\mathcal{L}_{rm}, \tag{4}$$

where $\mathcal{L}_{task}$ denotes the original motion magnification task loss, $\mathcal{L}_{rm}$ the bias term, and λ_{rm} a loss weight. We define the term $\mathcal{L}_{rm}$ as:

$$\mathcal{L}_{rm} = \frac{1}{K}\sum\nolimits_{k\in K}(1-\omega_k^p), \tag{5}$$

where p is a hyperparameter regulating the gradient of near $\omega = 1$, and K the number of components. Equation (5) induces the parameter ω close to 1 during optimization; thereby minimizing $\mathcal{L}_{total}$ finds the parameters $\{\omega\}$ that balance the task quality and and component removal. By fine-tuning the whole neural network with $\mathcal{L}_{total}$, we distinguish which components are redundant to perform the task. After converged,

Table 1: **Experiments with the component removal in residual blocks for each module.** We experiment removal of each component (*i.e.*, ReLU, skip-connection of a residual block, and a whole residual block) module-by-module (*i.e.*, the encoder, the manipulator, and the decoder). All the experiments are conducted using a pre-trained baseline model (300 epochs). The cross mark ✗ indicates that the components are completely removed, and the triangle symbol △ denotes that a few components survive.

Module	ReLU	Skip	Block	# Params [K] ↓	FLOPs [G] ↓	Quality Metric	
						SSIM ↑	LPIPS ↓
Baseline	–	–	–	967	41.3	0.980	0.180
Encoder	✗	–	–	967	41.3	0.979 (-0.001)	0.187 (+0.007)
	–	✗	–	967	41.3	0.977 (-0.003)	0.195 (+0.015)
	–	–	✗	838 (-129)	37.6 (-3.7)	0.978 (-0.002)	0.186 (+0.006)
Manipulator	✗	–	–	967	41.3	0.979 (-0.001)	0.181 (+0.001)
	–	✗	–	967	41.3	0.980 (-0.000)	0.181 (+0.001)
	–	–	✗	948 (-19)	40.6 (-0.7)	0.979 (-0.001)	0.182 (+0.002)
Decoder	△	–	–	967	41.3	0.966 (-0.014)	0.252 (+0.072)
	–	△	–	967	41.3	0.971 (-0.009)	0.231 (+0.051)
	–	–	△	524 (-443)	25.0 (-16.3)	0.945 (-0.035)	0.317 (+0.137)

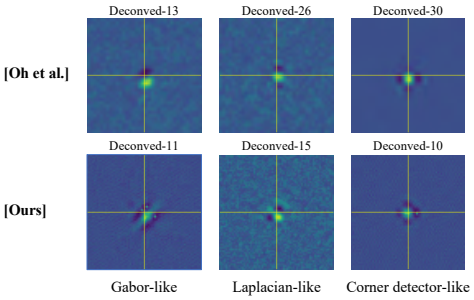


Figure 3: **Estimated impulse response of shape encoder.** We visualize the estimated kernels of the shape encoder in our (linear) model and the baseline (Oh et al., 2018) (non-linear) model. The kernels of the baseline encoder (top) and our deep linear neural encoder (bottom) have similar functionality to the conventional linear filters.

we explicitly discard the components whose resulting ω is larger than a threshold τ. Subsequently, we train the processed model using only $\mathcal{L}_{task}$ to achieve the highest quality under the filtered configuration.

Using the learning-to-remove method, we conduct an experiment that can reveal which module is relatively unimportant and can be substituted with a linear one. Nine configurations were established by applying the learning-to-remove method to only one module (encoder, manipulator, or decoder) and one type of component (activation functions, skip connections, or residual blocks) for each configuration. In this experiment, we only investigate the components in residual blocks since they accounts for over 84% of the parameters in the baseline model. The experimental results are summarized in Table 1 and we explain those results in the following paragraphs. In addition, we also show that the learning-to-remove method can be successfully adapted to recent video motion magnification architectures (Lado-Roigé & Pérez, 2023; Wang et al., 2024a) and another task with inhomogeneous architectures such as image super resolution (Lim et al., 2017). Please see Appendix A for the detailed results.

Encoder Removing all ReLUs and skip-connections of the residual blocks in the encoder results in *a negligible drop in quality*. We further find that the complete removal of the non-linearity from the encoder does not show a significant difference in quality. We visualize the estimated impulse response of the shape encoder of the baseline and ours in Fig. 3, showing that our linear encoder behaves similarly to the deep non-linear baseline (Oh et al., 2018). Moreover, dropping out of all the residual blocks significantly reduces the number of parameters and FLOPs, while resulting in negligible drops in quality.

Manipulator The result for the manipulator is analogous to the encoder; *i.e.*, all the components in the residual block can be removed with almost no quality loss. As a separate test, we manually remove the ReLU behind the convolutional layer of the manipulator, which is outside of the residual block. This results in quality degrading with no gain in computational costs. This is consistent with the report by the baseline that the manipulator without ReLU blurs strong edges and is more prone to noise.

Decoder Unlike the encoder and manipulator, removing any component from the decoder noticeably reduces quality, with SSIM dropping below the visual threshold when elements like ReLU or residual blocks are removed; even if some components remain intact. This highlights the critical role, *frame reconstruction*, of the decoder's components in maintaining output quality.

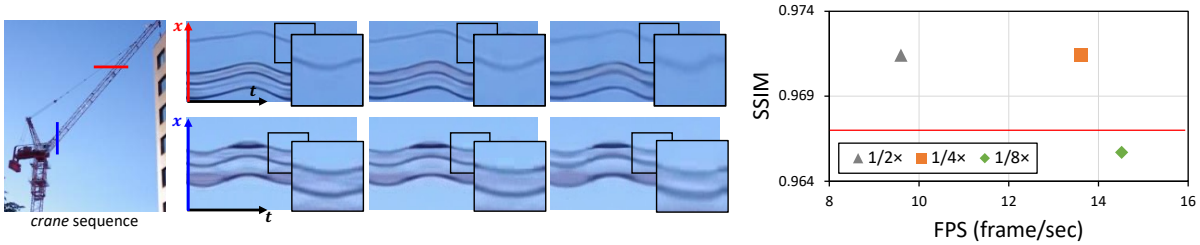


Figure 4: **[Left] Qualitative results varying the spatial resolution of the representations, [Right] Quantitative results varying the spatial resolution of the representations.** We choose one or two lines in the original frame and plot an **x**-t slice for *crane*, *drum*, and *guitar* sequences. $\frac{1}{2}\times$ denotes the baseline model (Oh et al., 2018). We notice that the differences of qualitative results between $\frac{1}{2}\times$ and $\frac{1}{4}\times$ are almost unnoticeable. In contrast, $\frac{1}{8}\times$ shows more blurry results than the others. We also measure SSIM and FPS results varying the different downsampling factors and plot the SSIM versus FPS trade-off graph. The $\frac{1}{4}\times$ downsampling significantly gains FPS with marginally sacrificing SSIM.

5.2 Spatial Bottleneck for Frame Reconstruction

The decoder is crucial for reconstructing high-resolution outputs from compressed features. As established in Sec. 5.1, the decoder is significantly more sensitive to component removal than the encoder or manipulator. We attribute this to the decoder's requirement for a large receptive field to effectively integrate spatial information for intricate magnified outputs. Reducing model depth (*i.e.*, removing layers or blocks) shrinks this receptive field, thereby limiting the decoder's capability to integrate information over larger spatial extents, which is essential for high-quality motion magnification.

The baseline (Oh et al., 2018) downsamples the spatial resolution of motion representation by $\frac{1}{2}\times$ at the beginning of the encoder, and upsamples it at the end of the decoder. This spatial bottleneck is designed to decrease memory footprint and increase the receptive field size, which is crucial for effective frame reconstruction in motion magnification. However, this reduction in feature size can result in information loss, adversely affecting the quality of the motion-magnified frame.

To address this trade-off between computational efficiency and magnification quality, we develop two alternative architectures with downsampling rates of $\frac{1}{4}\times$ and $\frac{1}{8}\times$, respectively. We then measure the trade-offs between SSIM and FPS. As shown in Fig. 4, the $\frac{1}{4}\times$ spatial resolution shows comparable qualitative results and SSIM to the baseline's $\frac{1}{2}\times$, while performing 1.85× faster FPS than the baseline. The $\frac{1}{8}\times$ downsampled representation further accelerates the computational speed, but suffers significant quality drops producing artifacts and much blurrier magnified frames compared to the $\frac{1}{4}\times$ one. Given these findings, the $\frac{1}{4}\times$ downsampled representation appears to be the better choice for achieving an efficient model. It offers a substantial gain of computational speed at the expense of acceptable drops in quality, making it a viable solution for balancing computational efficiency with magnification quality. We provide further detailed analyses of noise handling capabilities and the impact of downsampling motion representations in Appendix E.

5.3 Channel Reduction with Adding Layers

While reducing channel dimensions is a common strategy for lightweight networks (Singh et al., 2023a;b), naïvely applying it to the baseline (Oh et al., 2018) leads to significant degradation in generation quality. To address this, we propose increasing the network depth (d_D) while reducing channel dimensions (c_D) to maintain constant FLOPs. This approach aims to compensate for the quality loss by increasing the receptive field size. We conduct a controlled experiment varying d_D and c_D inversely.

Table 2: Controlled experiment on the impact of varying layer depth d_D in the decoder. Underlined: baseline configuration. Bold: best SSIM, LPIPS.

d_D	c_D	**FLOPs (G)**	**SSIM ↑**	**LPIPS ↓**
3	112	42.3	0.962 (−0.009)	0.232 (+0.041)
6	80	42.7	0.969 (−0.002)	0.202 (+0.011)
<u>9</u>	<u>64</u>	41.3	<u>0.971</u>	<u>0.191</u>
12	56	42.1	0.972 (+0.001)	0.186 (-0.005)
15	48	40.0	0.973 (+0.002)	0.183 (-0.008)
36	**32**	41.5	**0.975** (+0.004)	**0.172** (-0.019)
144	16	41.6	0.974 (+0.003)	0.175 (-0.016)

Table 2 indicates that networks with deeper depth generally achieve higher SSIM and lower LPIPS values. We empirically observe that extreme reduction ($[d_D, c_D] = [144, 16]$) performs less effectively than $[36, 32]$, suggesting that a minimum channel dimension is required. Based on this, we adopt the $[36, 32]$ configuration

Structural change	FLOPs [G] ↓	FPS ↑	SSIM ↑	LPIPS ↓
Baseline (Oh et al., 2018)	41.3	9.6	0.971	0.190
+ 1/4× Spatial Resolution	24.6	13.6	0.971	0.203
+ Single Linear Encoder	20.5	16.8	0.970	0.208
+ Scaling d_D and c_D	20.5	16.7	0.973	0.196
+ Scaling C_{upsample}	12.8	21.4	0.971	0.206
+ Scaling c_E and $c_{\text{downsample}}$	9.86	25.8	0.971	0.211
+ Perceptual Loss (proposed)	**9.86**	**25.8**	**0.972**	**0.163**

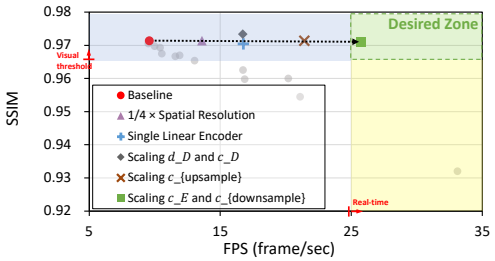


Figure 5: **Ablation study on structural change in model architecture.** We progressively impose structural changes on the baseline model (Oh et al., 2018), reaching our proposed lightweight model. Frames Per Second (FPS) is calculated for input frames of resolution 1920×1080, while the other metrics are measured on synthetic data with 384×384. We denote by d_D the decoder depth, by c_D, c_E the channel dimensions, and by $c_{\text{upsample}}, c_{\text{downsample}}$ the up/downsampling layer dimensions.

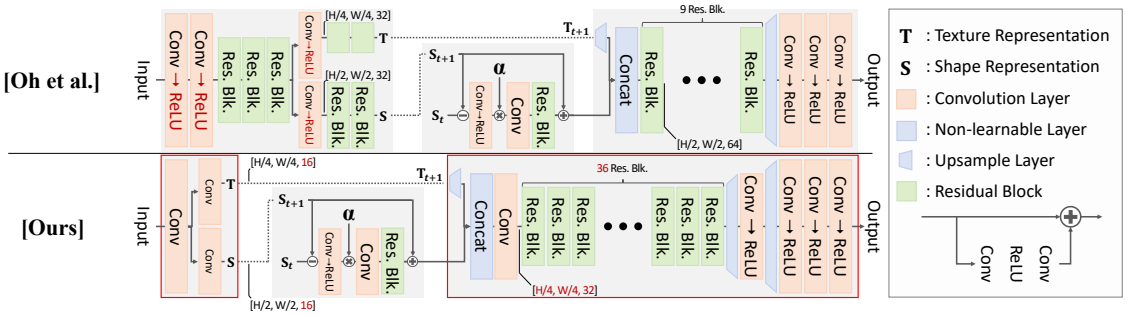


Figure 6: **Overall architecture of the proposed networks.** We present a real-time deep learning-based motion magnification model. The dimension of representations are denoted as [height, width, channel].

and proportionally reduce the encoder channels ($32 \rightarrow 16$) to maximize computational efficiency without significant quality loss.

5.4 Proposed Architecture

As illustrated in Fig. 5, we ablate the architecture including three key changes: (i) we downsample the spatial resolution of the latent motion representation in the decoder; (ii) we simplify the encoder to a single linear layer[2]; and (iii) we scale the decoder's residual blocks in layer depth (d_D) and channel dimension (c_D), keeping c_E fixed. Consistent with the controlled experiment in Table 2, varying d_D and c_D alone has little impact on FLOPs. However, reducing the channel dimension of the upsampling convolutions c_{upsample} to align it with the decoder residual width c_D, yielding additional FLOPs savings. Similarly, we can also reduce c_E and downsampling convolutions $c_{\text{downsample}}$ without significant quality degradation.

In addition, we add perceptual loss using VGG16 (Zhang et al., 2018) to the training loss to improve perceptual quality[3].

Through the integration of the findings, we present a deep learning-based motion magnification model as shown in Fig. 6, achieving a real-time computation speed and show favorable magnification quality.

6 Results

In this section, we assess the quality and effectiveness of our proposed model by comparing it with previous methods. Initially, we evaluate our model's quantitative performance on the synthetic dataset proposed by the baseline (Oh et al., 2018) (Sec. 6.1). In addition, we examine the methods' ability to handle noise and magnify subpixel motion (Sec. 6.3). Next, we present the qualitative results of our model and other motion

[2]After removing all nonlinear activations and residual blocks in the encoder, the remaining CNN layers naturally collapse into one layer, slightly reducing FLOPs without compromising quality.

[3]While Oh et al. (2018) reported that adding the perceptual loss does not improve SSIM, we found that using the perceptual loss is effective qualitatively as we shall see in the result.

magnification methods across various aspects: (1) Qualitative results on real video samples (Sec. 6.2), (2) Compatibility with temporal filter (Sec. 6.2), and (3) Physical accuracy of motion magnification (Sec. G).

6.1 Quantitative Analysis

We conduct a quantitative evaluation of our model and other learning-based methods regarding computational efficiency and the quality of generated image frames. Table 3 shows that only our proposed model surpasses the standard frame rate of real-time application (*i.e.*, 25 FPS), while achieving favorable motion magnification quality. Our model achieves computational speed ranging from at least 2.7× to a maximum of 34.9× faster than other learning-based methods (Oh et al., 2018; Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023; Pan et al., 2024; Wang et al., 2024a;b). Overall, our method achieves the second-best LPIPS score among the methods, while having a lower parameter count compared to the baseline (Oh et al., 2018) and the recent methods (Lado-Roigé & Pérez, 2023; Pan et al., 2024; Wang et al., 2024a;b).

Table 3: **Quantitative results.** We provide the detailed quantitative comparison aligned with Fig. 1. Frames Per Second (FPS) is calculated for input frames of resolution 1920 × 1080, while the other metrics (*i.e.*, Number of parameters, FPS, and SSIM) are for resolution 384 × 384. For studies that provide both a main model and a lightweight variant, we denote the main model as *D1* and the lightweight model as *D2*. **Bold** indicates the best performance, and underlined indicates the second-best performance.

Model		FPS ↑	# Params [M] ↓	SSIM ↑	LPIPS ↓
Oh et al. (2018)		9.6	0.967	0.971	0.190
Pan et al. (2024)	(D2)	7.0	17.288	0.912	0.298
Singh et al. (2023b)	(D2)	4.2	**0.054**	0.930	0.308
Singh et al. (2023b)	(D1)	4.0	0.117	0.930	0.279
Pan et al. (2024)	(D1)	3.5	17.288	0.912	0.298
Singh et al. (2023a)	(D2)	3.1	0.185	0.941	0.308
FD4MM (Wang et al., 2024b)		2.2	1.47	–	–
Singh et al. (2023a)	(D1)	1.8	1.814	0.943	0.265
STB-VMM (Lado-Roigé & Pérez, 2023)		1.6	31.335	0.977	0.176
EulerMormer (Wang et al., 2024a)		0.7	1.51	**0.984**	**0.107**
Ours		**25.8**	0.731	0.972	0.163

In contrast, the other methods either lag behind ours in both generation quality and computational speed (Singh et al., 2023a;b; Pan et al., 2024), or exhibit comparable generation quality but are significantly slower (Oh et al., 2018; Lado-Roigé & Pérez, 2023; Wang et al., 2024a). Specifically, Singh et al. (2023a) achieve the lowest number of parameters compared to other methods, but it does not translate into increased computational speed.

6.2 Motion Magnification on Real Video Samples

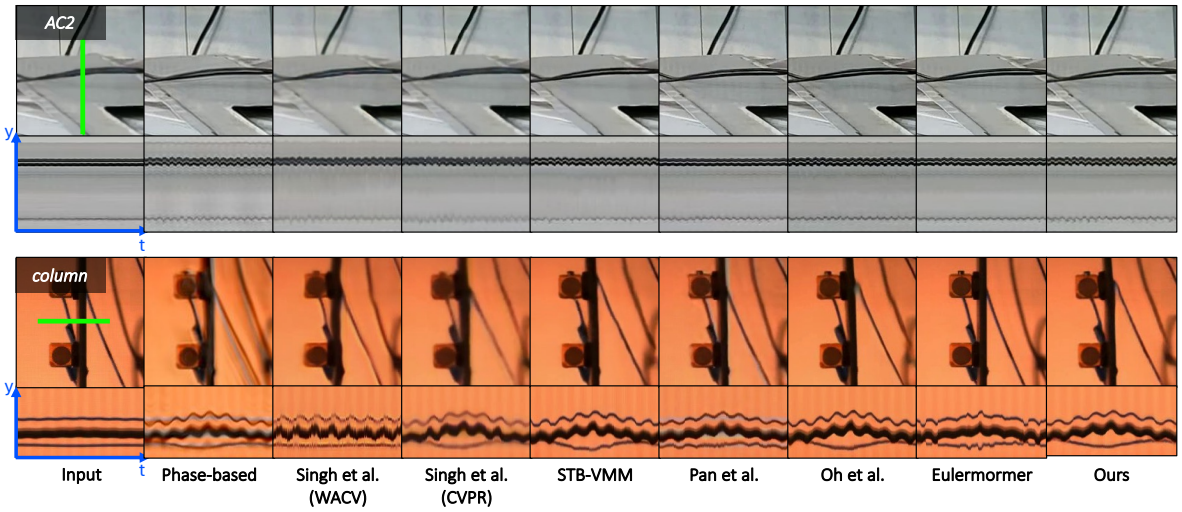
We test our method on real-world videos, including mechanical vibrating units (AC2) and structural components (Column, Wheel), to demonstrate practical applicability.

Figure 7 includes two general processing modes in learning-based motion magnification: *Static*, which uses a fixed reference (X_1, X_t), and *Dynamic*, which employs a sliding reference (X_{t-1}, X_t). In both cases, the model magnifies the difference between the reference frame and X_t frame without advanced temporal filters. Figure 7 also includes the case when applying temporal filters (*e.g.*, FIR, butterworth).

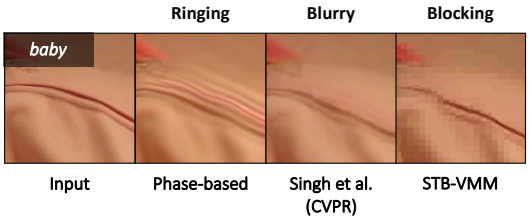
Advantages of learning-based approaches When comparing the phase-based method (Wadhwa et al., 2013) with other learning-based methods (Oh et al., 2018; Pan et al., 2024; Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023; Wang et al., 2024a;b), a notable disparity in generation quality becomes apparent. Learning-based methods produce relatively higher-quality results and demonstrate superior handling of edge cases, such as occlusion or dis-occlusion, compared to the phase-based method, which exhibits noticeable ringing artifacts and produces blurry results.

Comparison with the baseline Our model shows comparable generation quality of magnified frames with the baseline model (Oh et al., 2018), while running in real-time on Full-HD videos. Our model also captures the edges clearly and shows no ringing artifacts as well as the baseline.

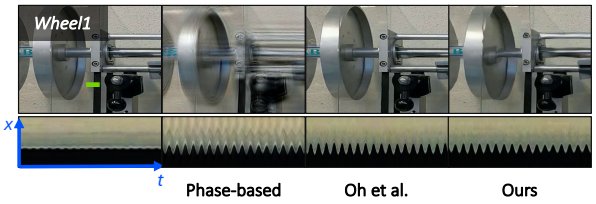
Comparison with other learning-based methods Overall, many learning-based methods (Pan et al., 2024; Singh et al., 2023a;b), except for recent transformer-based architectures (Lado-Roigé & Pérez, 2023; Wang et al., 2024a;b), tend to produce results with more noise, blur, and ringing artifacts compared to our model and the baseline. Notably, STB-VMM occasionally exhibits blocking artifacts, when processing lower-resolution video sequences such as a *baby* sequence with a resolution 960×544. The artifacts may arise from its use of a patch-wise processing technique, a characteristic of Vision Transformer architecture (Dosovitskiy



(a) Qualitative comparison across models (Top: *static*, Bottom: *dynamic*)



(b) Examples for several artifacts



(c) Applying temporal filter

Figure 7: **Motion magnification results.** (a) Qualitative comparison across models (Top: *Static* mode, Bottom:*Dynamic* mode), (b) Examples for several artifacts, and (c) Results applying temporal filter. Ours successfully magnifies the given video sequences without causing any artifacts even without temporal filters, and also shows compatibility with temporal filters. Configurations can be found in Appendix B.

et al., 2020; Liu et al., 2021). In contrast, our method is less susceptible to these artifacts and degradation, while uniquely achieving real-time processing speed in Full-HD among all the learning-based methods.

Compatibility with Temporal Filter Although our model is capable of producing fine results without relying on temporal filtering, incorporating such filtering can further reduce artifacts caused by unwanted motions, as suggested in Oh et al. (2018). By directly applying temporal filtering to motion representations, we effectively isolate only the desired motion within a temporal context, before feeding it into the decoder. Our model reconstructs a clear periodic signal, demonstrating its compatibility with temporal filters, similar to the baseline (Oh et al., 2018). Phase-based method (Wadhwa et al., 2013) can also capture periodic signal, however, it tends to produce severe ringing artifacts.

6.3 Noise and Subpixel Test

We perform a quantitative evaluation comparing our model and other motion magnification methods (Wadhwa et al., 2013; Oh et al., 2018; Singh et al., 2023a;b; Lado-Roigé & Pérez, 2023; Pan et al., 2024) in handling subpixel motion and noise, following the same evaluation setup as described in Sec. E or the baseline (Oh et al., 2018). As shown in Fig. 8, our model demonstrates noise handling capability comparable or superior to STB-VMM (Lado-Roigé & Pérez, 2023) and the baseline modelin both small motion and large motion noise tests. Notably, in the small motion case, ours surpasses STB-VMM (Lado-Roigé & Pérez, 2023) as the noise factor becomes greater than 1.

Singh *et al.* (Singh et al., 2023a;b) exhibit distinctive noise characteristics that differ from other motion magnification methods. Specifically, they show weaker noise handling capability under a noise factor of 0.1 in the small motion noise test. While it may appear that their methods outperform others as the noise factor increases, this perception is deceptive. As shown in Fig. 8 [Bottom], Singh *et al.* (Singh et al.,

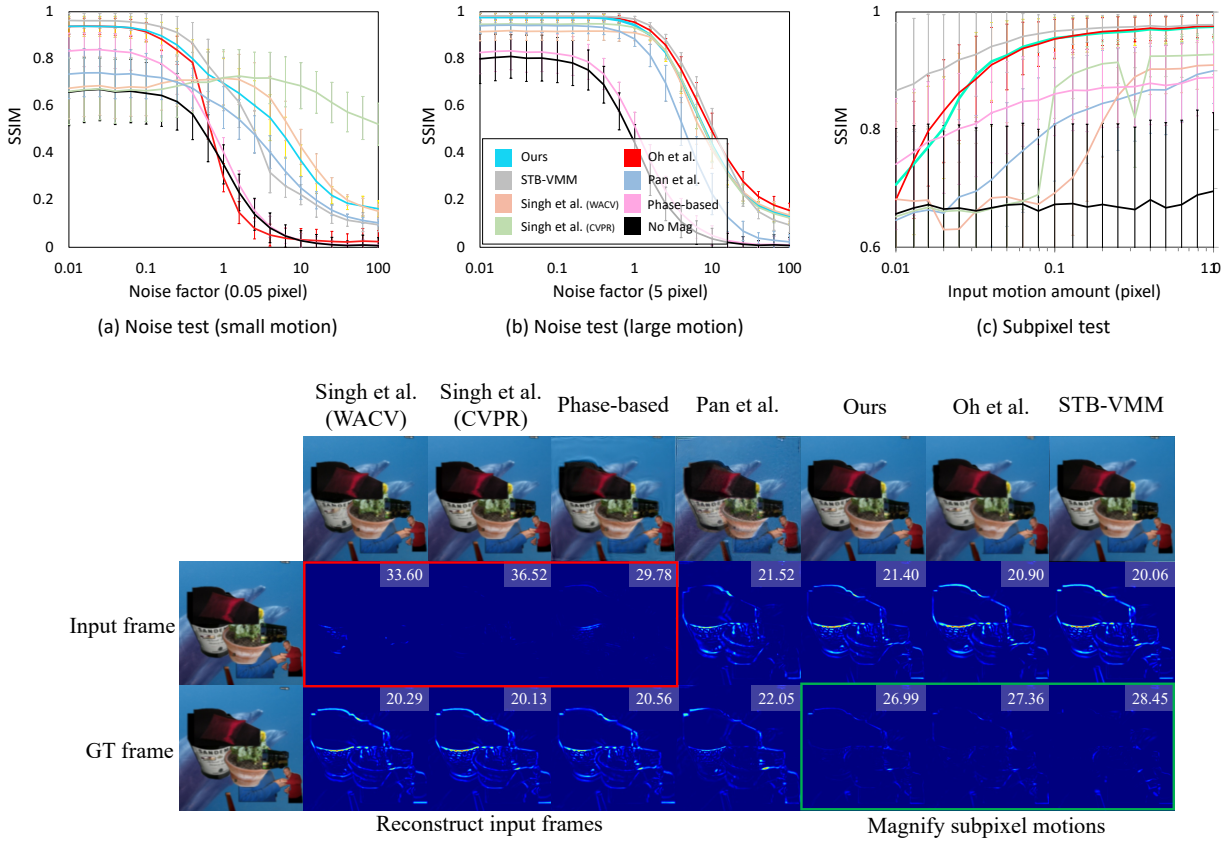


Figure 8: **Noise test and subpixel motion test** [Top] (a, b) Noise test with small motion of 0.05px and large motion of 5px, (c) Sub-pixel motion test. [Bottom] shows an error map and PSNR metric between the ground-truth magnified frame and the magnified frame produced by each method. This example is one of the test pairs for a noise factor of 0.158 in [Top] (a). Our model exhibits strong noise handling, particularly surpassing STB-VMM (Lado-Roigé & Pérez, 2023) in small motion noise tests as the noise factor increases. In contrast, Singh *et al.* (Singh et al., 2023a;b) show weaker performance, often replicating the input image rather than magnifying motion. The subpixel test further confirms the superiority of our model in maintaining magnification quality across all methods.

2023a;b) produce output frames that closely resemble the input frames rather than the ground truth (GT) frames. It suggests these methods often merely replicate the input image instead of effectively magnifying small motions. On the other hand, our model, along with Oh et al. (2018) and STB-VMM (Lado-Roigé & Pérez, 2023), shows considerably smaller errors in comparison to the GT frames, indicating a more successful magnification of small motions.

The subpixel test confirms this; unlike prior arts that fail to isolate small motions, our model maintains magnification quality comparable to the baseline (Oh et al., 2018) and STB-VMM (Lado-Roigé & Pérez, 2023), while outperforming others (Wadhwa et al., 2013; Singh et al., 2023a;b; Pan et al., 2024).

6.4 Physical Accuracy

To measure the physical accuracy of various motion magnification methods, we simulate a 20 Hz sinusoidal vibration in a laboratory setting using a vibration generator following Byung-Ki et al. (2024). An accelerometer attached to the generator measures the peak amplitude of acceleration (m/s^2), which is then converted into sinusoidal displacement waves in both real-world (m) and pixel (px) units using the formula $\mu = a/\omega^2$ where ω is the vibration frequency. Then, we overlay the converted sinusoidal displacement into the image plane using the formula of pixel displacement $k(t) = \frac{f}{Lv}s(t)$, where f is the focal length, L the distance

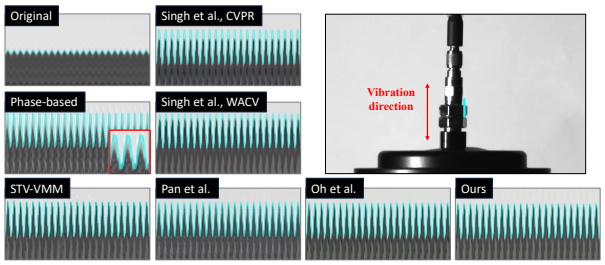


Figure 9: **Physical accuracy on motion magnification.** The vibration simulator is set to generate a periodic vibration with the frequency 21 Hz. Our model captures the frequency of the vibration accurately and magnify the sinusoidal periodic signal with fewer artifacts, showing comparable quality to the baseline, which suggests physical accuracy of their method. We use Finite Impulse Response (FIR) filter with a frequency band of lower bound 17 Hz and upper bound 25 Hz, and the amplification factor α of 20.

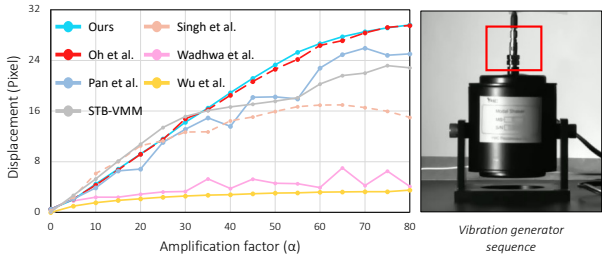


Figure 10: **Comparison of linearity of motion magnification according to amplification factor α.** Learning-based methods (Pan et al., 2024; Singh et al., 2023a; Lado-Roigé & Pérez, 2023) including the baseline (Oh et al., 2018) and ours show linear magnification of input motion according to the amplification factor α, whereas signal processing based methods including Wu et al. (2012) and Wadhwa et al. (2013) show distorted and attenuated motion magnification. The peak-to-peak displacements are estimated by Kanade-Lucas-Tomasi tracking algorithm (Tomasi & Kanade, 1991).

from the camera to the vibrator, and v the sensor's per-pixel size. The conversion parameters are detailed in Appendix B.

As demonstrated in Fig. 9, our method aligns well with the 20× magnified video, preserving the sinusoidal pattern and edge details comparably to the baseline method (Oh et al., 2018). In contrast, the phase-based method (Wadhwa et al., 2013) introduces significant ringing effects and fails to accurately reconstruct movements. Singh et al. (2023a) also struggle with accurately capturing the sinusoidal signal, indicating inferior separation of motion of interest from the background.

We further verify the linearity between the amplification factor and the resulting motion magnitude. As shown in Fig. 10, signal processing-based methods (Wu et al., 2012; Wadhwa et al., 2013; 2014; Takeda et al., 2020) exhibit severe attenuation as the amplification factor increases, failing to preserve physical fidelity. In contrast, both our model and the baseline (Oh et al., 2018) demonstrate clear linearity up to an 80× amplification factor, ensuring reliable motion quantification. These results validate the potential of our method as a reliable analysis tool.

7 Conclusion

In this paper, we propose the first learning-based motion magnification model running in real-time on Full-HD videos, generating high-quality magnification results. Previous learning-based motion magnification methods either lag behind ours in both generation quality and computational speed, or exhibit comparable generation quality but significantly slower computational speed. These challenges stem from a lack of understanding of which parts and characteristics of a motion magnification model play crucial roles in the target task, which has been barely explored before. We experimentally analyze components of the baseline in terms of their importance and functionality on the human-calibrated quality evaluation metric. Integrating insights from the analyses, we develop a real-time motion magnification model, which achieves a computational speed ranging from at least **2.7**× to a maximum of **34.9**× **faster** than existing learning-based methods while maintaining generation quality comparable to the baseline and state-of-the-art model. We hope that our analyses can serve as a guide for developing learning-based motion magnification model in the future. Further discussions regarding temporal filtering trade-offs and practical deployment limitations, such as handling large objects and camera motions, are provided in Appendix H.

References

Jae Young An and Soo Il Lee. Phase-based motion magnification for structural vibration monitoring at a video streaming rate. *IEEE Access*, 10:123423–123435, 2022.

Guha Balakrishnan, Fredo Durand, and John Guttag. Detecting pulse from head motions in video. In *CVPR*, pp. 3430–3437, Portland, OR, USA, 2013. IEEE. doi: 10.1109/CVPR.2013.440.

Kwon Byung-Ki, Oh Hyun-Bin, Kim Jun-Seong, Hyunwoo Ha, and Tae-Hyun Oh. Learning-based axial video motion magnification. In *ECCV*, 2024.

Y-J Cha, Justin G Chen, and Oral Büyüköztürk. Output-only computer vision based damage detection using phase-based optical flow and unscented kalman filters. *Engineering Structures*, 132:300–313, 2017.

Justin G Chen, Neal Wadhwa, Young-Jin Cha, Frédo Durand, William T Freeman, and Oral Buyukozturk. Structural modal identification through high speed camera video: Motion magnification. *Topics in Modal Analysis I, Volume 7*, 7:191–197, 2014.

Justin G Chen, Neal Wadhwa, Young-Jin Cha, Frédo Durand, William T Freeman, and Oral Buyukozturk. Modal identification of simple structures with high-speed video using motion magnification. *Journal of Sound and Vibration*, 345:58–71, 2015a.

Justin G Chen, Neal Wadhwa, Frédo Durand, William T Freeman, and Oral Buyukozturk. Developments with motion magnification for structural modal identification through camera video. In *Dynamics of Civil Structures, Volume 2*, pp. 49–57. Springer, Cham, Switzerland, 2015b.

Justin G Chen, Abe Davis, Neal Wadhwa, Frédo Durand, William T Freeman, and Oral Büyüköztürk. Video camera–based vibration measurement for civil infrastructure applications. *Journal of Infrastructure Systems*, 23(3):B4016013, 2017.

Abe Davis, Michael Rubinstein, Neal Wadhwa, Gautham J. Mysore, Frédo Durand, and William T. Freeman. The visual microphone: Passive recovery of sound from video. *ACM Transactions on Graphics (SIGGRAPH)*, 33(4), jul 2014. ISSN 0730-0301.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Amir Ben Dror, Niv Zehngut, Avraham Raviv, Evgeny Artyomov, Ran Vitek, and Roy Jevnisek. Layer folding: Neural network depth reduction using activation linearization. *arXiv preprint arXiv:2106.09309*, 2021.

Wenkang Fan, Zhuohui Zheng, Wankang Zeng, Yinran Chen, Hui-Qing Zeng, Hong Shi, and Xiongbiao Luo. Robotically surgical vessel localization using robust hybrid video motion magnification. *IEEE Robotics and Automation Letters*, 6(2):1567–1573, 2021.

Brandon Y Feng, Hadi Alzayer, Michael Rubinstein, William T Freeman, and Jia-Bin Huang. 3d motion magnification: Visualizing subtle motions from time-varying radiance fields. In *CVPR*, 2023.

William T Freeman, Edward H Adelson, and David J Heeger. Motion without movement. *ACM Transactions on Graphics (SIGGRAPH)*, 25(4):27–30, 1991a.

William T Freeman, Edward H Adelson, et al. The design and use of steerable filters. *IEEE TPAMI*, 13(9): 891–906, 1991b.

Sicheng Gao, Yutang Feng, Linlin Yang, Xuhui Liu, Zichen Zhu, David Doermann, and Baochang Zhang. Magformer: Hybrid video motion magnification transformer from eulerian and lagrangian perspectives. In *BMVC*, pp. 444, London, UK, 2022. BMVA Press. URL `https://bmvc2022.mpi-inf.mpg.de/444/`.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, pp. 544–560, Cham, Switzerland, 2020. Springer.

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *ECCV*, 2018.

Mirek Janatka, Ashwin Sridhar, John Kelly, and Danail Stoyanov. Higher order of motion magnification for vessel localisation in surgical video. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part IV 11*, pp. 307–314. Springer, 2018.

Ricard Lado-Roigé and Marco A. Pérez. Stb-vmm: Swin transformer based video motion magnification. *Knowledge-Based Systems*, 269(7):110493, 2023.

Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR*, pp. 1132–1140, Honolulu, HI, USA, 2017. IEEE.

Ce Liu, Antonio Torralba, William T Freeman, Frédo Durand, and Edward H Adelson. Motion magnification. *ACM TOG*, 24(3):519–526, 2005.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Tae-Hyun Oh, Ronnachai Jaroensri, Changil Kim, Mohamed Elgharib, Fr'edo Durand, William T Freeman, and Wojciech Matusik. Learning-based video motion magnification. In *ECCV*, pp. 663–679, Cham, Switzerland, 2018. Springer International Publishing.

Zhaoying Pan, Daniel Geng, and Andrew Owens. Self-supervised motion magnification by backpropagating through optical flow. *Advances in Neural Information Processing Systems*, 36, 2024.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Jasdeep Singh, Subrahmanyam Murala, and G Kosuru. Lightweight network for video motion magnification. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2040–2049, Los Alamitos, CA, USA, 2023a. IEEE Computer Society.

Jasdeep Singh, Subrahmanyam Murala, and G. Sankara Raju Kosuru. Multi domain learning for motion magnification. In *CVPR*, pp. 13914–13923, Vancouver, BC, Canada, 2023b. IEEE. URL `https://doi.org/10.1109/CVPR52729.2023.01337`.

Shoichiro Takeda, Kazuki Okami, Dan Mikami, Megumi Isogai, and Hideaki Kimata. Jerk-aware video acceleration magnification. In *CVPR*, pp. 1769–1777, Salt Lake City, UT, USA, 2018. IEEE.

Shoichiro Takeda, Yasunori Akagi, Kazuki Okami, Megumi Isogai, and Hideaki Kimata. Video magnification in the wild using fractional anisotropy in temporal distribution. In *CVPR*, pp. 1614–1622, Long Beach, CA, USA, 2019. IEEE.

Shoichiro Takeda, Megumi Isogai, Shinya Shimizu, and Hideaki Kimata. Local riesz pyramid for faster phase-based video magnification. *IEICE Transactions on Information and Systems*, 103(10):2036–2046, 2020.

Shoichiro Takeda, Kenta Niwa, Mariko Isogawa, Shinya Shimizu, Kazuki Okami, and Yushi Aono. Bilateral video magnification filter. In *CVPR*, pp. 17348–17357, New Orleans, LA, USA, 2022. IEEE.

Carlo Tomasi and Takeo Kanade. Detection and tracking of point. *IJCV*, 9:137–154, 1991.

Daniel Myklatun Tveit, Kjersti En2gan, Ivar Austvoll, and Øyvind Meinich-Bache. Motion based detection of respiration rate in infants using video. In *ICIP*, pp. 1225–1229, Phoenix, AZ, USA, 2016. IEEE.

Neal Wadhwa, Michael Rubinstein, Frédo Durand, and William T Freeman. Phase-based video motion processing. *ACM TOG*, 32(4):1–10, 2013.

Neal Wadhwa, Michael Rubinstein, Frédo Durand, and William T Freeman. Riesz pyramids for fast phase-based video magnification. In *IEEE International Conference on Computational Photography (ICCP)*, pp. 1–10, Santa Clara, CA, USA, 2014. IEEE, IEEE.

Fei Wang, Dan Guo, Kun Li, and Meng Wang. Eulermormer: Robust eulerian motion magnification via dynamic filtering within transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 5345–5353, Washington, DC, USA, 2024a. AAAI Press.

Fei Wang, Dan Guo, Kun Li, Zhun Zhong, and Meng Wang. Frequency decoupling for motion magnification via multi-level isomorphic architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18984–18994, Seattle, WA, USA, 2024b. IEEE.

Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, and William Freeman. Eulerian video magnification for revealing subtle changes in the world. *ACM TOG*, 31(4):1–8, 2012.

Tianfan Xue, Michael Rubinstein, Neal Wadhwa, Anat Levin, Fredo Durand, and William T Freeman. Refraction wiggles for measuring fluid depth and velocity from video. In *ECCV*, pp. 767–782, Cham, Switzerland, 2014. Springer International Publishing.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pp. 586–595, Salt Lake City, UT, USA, 2018. IEEE.

Yichao Zhang, Silvia L Pintea, and Jan C Van Gemert. Video acceleration magnification. In *CVPR*, pp. 502–510, Honolulu, HI, USA, 2017. IEEE.

A Generality of Learning-to-remove Framework

A.1 Applying for image super resolution model

We apply our learning-to-remove method to an image super resolution model to verify its generality. The MDSR (multi-scale super resolution) model (Lim et al., 2017) mainly composes of 1) three scale specific pre-processing branches (*i.e.*, *an encoder*) which have two residual blocks respectively and 2) the main branch (*i.e.*, *a decoder*) which has 16 residual blocks and processes the representation regardless of the scale. As shown in Table A1, we discover that no noticeable PSNR drop occurs in the model while achieving notable parameter reduction, even if the residual blocks in the encoder of MDSR are totally removed. On the other hand, removing all 16 residual blocks in the decoder leads to significant quality degradation while reducing parameters relatively smaller than removing the encoder module. From these observations, we can investigate the redundancy of neural components of each module in the image super resolution model; *i.e.*, the encoder, whose functionality is scale specific pre-processing, can deal with the task without any heavy residual blocks. These results demonstrate the potential applicability of our learning-to-remove method beyond the motion magnification task.

Table A1: **Our framework applied to image super resolution task.** We eliminate the residual blocks of the encoder and decoder module in the multi-scale SR (MDSR) network, respectively. The network has three scale specific pre-processing modules (*i.e.*, the encoder).

Model	**PSNR [dB]** ↑			**Params [M]** ↓
	2×	3×	4×	
MDSR	35.61	31.65	29.57	3.23
- Encoder	35.61	31.64 (-0.01)	29.57	2.00 (-38.1%)
- Decoder	34.93 (-0.68)	31.02 (-0.63)	28.99 (-0.58)	2.05 (-36.5%)

Table A2: **Component removal in attention-based blocks for each module of EulerMormer (Wang et al., 2024a).** We experiment removal of attention-based blocks module-by-module. All experiments are conducted using pre-trained weights. ✗ indicates completely removed, and △ denotes a few components survive.

Module	Block	# Params [M] ↓	FLOPs [G] ↓	Quality Metric	
				SSIM ↑	LPIPS ↓
Original	–	1.51	56.0	0.984	0.107
Encoder	✗	1.35 (-0.16)	50.2 (-5.8)	0.982 (-0.002)	0.116 (+0.009)
Manipulator	✗	1.43 (-0.08)	53.1 (-2.9)	0.982 (-0.002)	0.119 (+0.012)
Decoder	△	0.54 (-0.97)	20.3 (-35.7)	0.952 (-0.032)	0.267 (+0.160)

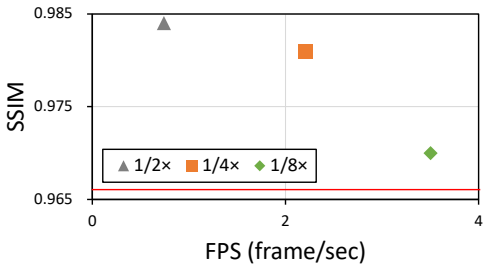


Figure A1: **Quantitative results varying the spatial resolution on EulerMormer (Wang et al., 2024a).** $\frac{1}{2}\times$ denotes the original model. We plot the SSIM versus FPS trade-off graph for different downsampling factors. The $\frac{1}{4}\times$ downsampling gains FPS with marginally sacrificing SSIM.

Table A3: **Component removal in attention-based blocks for each module of STB-VMM (Lado-Roigé & Pérez, 2023).** We experiment removal of attention-based blcoks module-by-module (*i.e.*, the encoder, the manipulator, and the decoder). All the experiments are conducted using a pre-trained weights. The cross mark ✗ indicates that the components are completely removed, and the triangle symbol △ denotes that a few components are survived. In STB-VMM, the computational cost (FLOPs) is heavily concentrated in the decoder's upsampling convolution layer, accounting for 94.8%. This concentration stems from the layer's inefficient design (`ConvTranspose2d` with 8×8 transposed convolution). To facilitate interpretation of the results, we also report FLOPs excluding the upsampling layer (see the *FLOPs wo. up_conv [G]* column in the table.

Module	Block	# Params [M] ↓	FLOPs [G] ↓		Quality Metric	
			w. up_conv	wo. up_conv	SSIM ↑	LPIPS ↓
Original	–	31.3	366.9	19.0	0.978	0.178
Encoder	✗	18.6 (-12.7)	361.8 (-5.1)	13.9 (-5.1)	0.978	0.174 (-0.004)
Manipulator	✗	30.2 (-1.1)	364.6 (-2.3)	16.7 (-2.3)	0.978	0.174 (-0.004)
Decoder	△	24.9 (-6.4)	364.4 (-2.5)	16.5 (-2.5)	0.962 (-0.016)	0.247 (+0.069)

A.2 Applying our framework on recent architectures

We further apply the learning-to-remove method to two recent models—STB-VMM (Lado-Roigé & Pérez, 2023) and EulerMormer (Wang et al., 2024a). As you can see in Table A2 and Table A3, we observe the same pattern as in the main paper: the encoder is less sensitive to removal than the decoder. Additionally, for EulerMormer, we conduct spatial bottleneck experiments. From Figure A1, we can find that the $\frac{1}{4}$ resolution setting provides a favorable trade-off, consistent with the observation from Sec. 5.2[4]. These results highlight that our insights extend beyond a single baseline and remain valid across different architectures.

B Hyperparameter setup

We give the parameters we used for each video. We conduct both experimental cases, motion magnification with temporal filter and without temporal filter to verify the compatibility and generality of our method. We set the amplification factor α, temporal filter type and frequency band as given in Table A4 and Table A5. We also set hyperparameters for acquiring pixel displacement in the physical accuracy test as given in Table A6.

[4]STB-VMM (Lado-Roigé & Pérez, 2023) is excluded from spatial bottleneck experiment due to memory footprint issues

Table A4: Hyperparameter for processing with temporal filters.

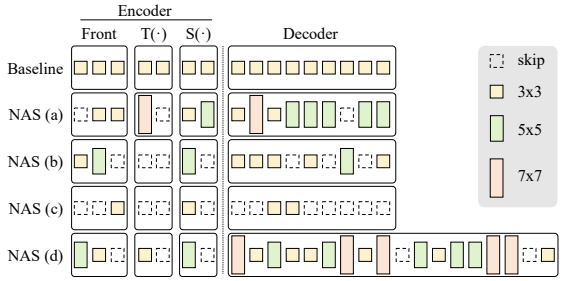
Sequence	Amp. Factor	FPS	Temporal Filter	Freq. Band (Hz)
crane	75×	24	FIR	0.2 - 0.25
drum	10×	1900	FIR	74 - 78
guitar	25×	600	FIR	72 - 92
Wheel1	20×	120	FIR	20 - 30
baby	20×	30	Butterworth	0.01 - 1

Table A5: Hyperparameter for standard mode.

Sequence	Amp. Factor	mode
baby	20 ×	Static
AC2	8 ×	Static
column	7 ×	Dynamic
gun	8 ×	Dynamic

Table A6: Hyperparameters for acquiring pixel displacement.

Hyperparameters	Unit	Value
Vibration frequency ω	Hz	20
Peak amplitude of acceleration a	m/s^2	4.11
Camera-to-vibrator distance L	m	2
Focal length f	mm	100
Per-pixel sensor size v	μm	5.86



Constraints				Results		
Enc.	Dec.	Ch.	Max. FLOPs [G]	Trainable	SSIM	FLOPs [G]
7	9	8	10	✓	0.824	4.96
		16	10	✓	0.887	9.95
		32	20	✓	0.884	19.86
	18	8	10	✓	0.836	9.34
		16	-	✗	-	-
		32	-	✗	-	-

Figure A3: **[Left] Found configurations by the NAS experiments, [Right] Constraints and results of the NAS experiments.** NAS (a) - (d) of [Left] correspond to the row 1 - 4 of [Right]. [Left] From the found configurations, we observe that several residual blocks in the decoder are removed, in contrast, our proposed model discards only the residual blocks in the encoder and the manipulator. [Right] Constraints are summarized as follows: Maximum number of residual blocks in the encoder (Enc.) and the decoder (Dec.), fixed channel dimension of the decoder (Ch.) and maximum FLOPs of the architecture (Max. FLOPs). The channel dimension of the encoder is set as half of the channel of the decoder. Training is failed in the two experiments due to gradient exploding. We refer to Guo et al. (2020) for designing the NAS experiments.

C Comparison to NAS Results

We conduct experiments to find an efficient model by adopting one of the one-shot NAS methods (Guo et al., 2020), following the same evaluation setting described in Sec. 4.1. The search space used in the NAS experiments includes the kernel size and the number of residual blocks in the encoder and decoder. We set the constraints for each NAS experiment as [max layer depth of the encoder, of the decoder, channel dimension of the decoder, max FLOPs]: (a) [7,9,8,10G], (b) [7,9,16,10G], (c) [7,9,32,20G], and (d) [7,18,8,10G].

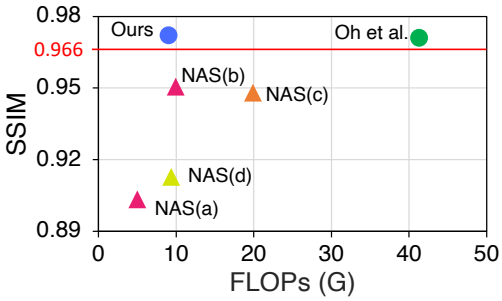


Figure A2: **Quantitative comparison of our proposed model and NAS-searched models.**

The SSIM and FLOPs of the NAS models are compared with our model in Fig. A2. All the architectures found by NAS show significant quality degradation, lying on the poor trade-off between SSIM and FLOPs compared to our proposed model. Specifically, NAS (b) shows the best SSIM among all the NAS models but still is below 0.966, the visual acceptability threshold. These results support that an efficient yet effective model could not be found straightforwardly through standard search methods.

Figure A3 shows the searched architectures which achieve the best quality under fixed FLOPs constraints. The search space of the NAS includes the kernel sizes and allows skipping the residual block. When we increase the maximum number of residual blocks (*i.e.*, layer depth) in the search space, a few NAS experiments fail to converge. We find that even the best architecture searched on the trainable configuration cannot approach the optimal trade-off between the quality (SSIM) and the computational costs (FLOPs) in comparison to our proposed architecture.

Table A7: Effect of increasing encoder layer depth while maintaining FLOPs. The first row corresponds to the baseline (Oh et al., 2018).

Enc	Dec	FLOPs [G] ↓	# Params [K] ↓	Quality Metric	
				SSIM ↑	LPIPS ↓
7	**9**	**41.3**	**967.0**	**0.971**	**0.190**
13	8	41.7	1,004 (+37.0)	0.970 (-0.001)	0.193 (+0.003)
22	6	40.8	1,023 (+56.0)	0.968 (-0.003)	0.208 (+0.018)
28	5	41.2	1,060 (+93.0)	0.966 (-0.005)	0.212 (+0.022)

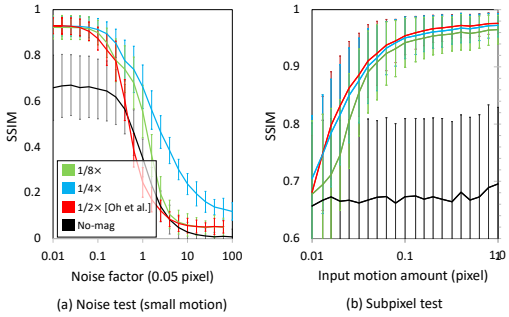


Figure A4: **(a) Noise test and (b) Subpixel motion test for downsampled motion representation.**

D Encoder with Deeper Layer

We observe that a deeper decoder achieves better generation quality due to its need for a large receptive field (see Sec. 5.3 of the main paper). To evaluate whether a larger receptive field in the encoder enhances magnification quality, we conduct an additional experiment: training the encoder with increased depth and the decoder with reduced depth. We ensure similar computational costs (FLOPs) across configurations for a fair comparison. As indicated in Table A7, the configurations with a deeper encoder and shallower decoder exhibit significant degradation in SSIM and LPIPS. This indicates that enlarging the encoder's receptive field does not improve the generation of magnified frames.

E Effect of Downsampling Motion Representations

To see the noise handling ability of the architectures with the downsampled representations, we perform two test scenarios, *i.e.*, the noise test and the subpixel test, following the evaluation setup as described in Sec. 3.2. In Fig. A4, the $\frac{1}{4}\times$ downsampled representation demonstrates improved SSIM in the noise test and comparable SSIM in the subpixel motion test compared to the $\frac{1}{2}\times$ and $\frac{1}{8}\times$ downsampled representations. These results confirm the effectiveness of $\frac{1}{4}\times$ downsampled representation in handling noise, making it a robust choice for noise-sensitive applications.

F Calibration on Quality Evaluation Metric

When we evaluate the models using image similarity measures, *e.g.*, SSIM, a remaining challenge is that we are still not sure how well the metrics align with human perception. Due to the absence of real ground truth videos, prior work follows the evaluation method suggested by Oh et al. (2018), where similarity metrics including SSIM are measured with a synthetic evaluation subset. However, due to the small domain gaps between the training and evaluation data, the SSIM values obtained from the synthetic evaluation are biased towards being very high. Moreover, models with subtle differences in high SSIM values (*e.g.*, 0.93 and 0.96) on the synthetic evaluation exhibit noticeable quality differences in real videos as perceived by humans. To address this problem, we attempt a simple calibration of the quality evaluation metric, *i.e.*, SSIM, by conducting a human study on the relationship between SSIM measured on synthetic data and visual quality for real data through real video examples.

We conduct a two-fold user study: (i) variants of the baseline model (Oh et al., 2018), and (ii) a comparative study across four models, including ours. We first train four architecture variants (Models A, B, C, and E) by altering the entire channel dimensions of the baseline model (Oh et al., 2018) (Model D). These models show different SSIM scores (ranging from 0.9320 to 0.9744) on the synthetic dataset. Using these models, we magnify real video sequences (*e.g.*, crane and column in Fig. A5 [Left]).

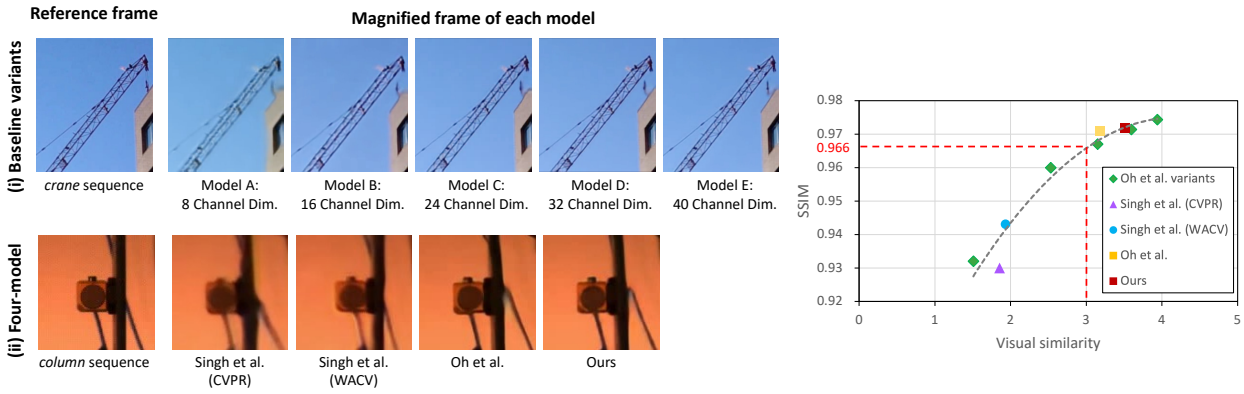


Figure A5: **[Left] Samples of the magnified frame from each model used for the human study. [Right] Relationship between SSIM and visual similarity scored by humans.** We conduct a two-fold user study: (i) variants of the baseline model (Oh et al., 2018), and (ii) a four-model (Singh et al., 2023a;b; Oh et al., 2018) comparison, including ours. We develop four architecture variants (Models A, B, C, and E) by altering the entire channel dimensions of the baseline model (Oh et al., 2018) (Model D) from 8 to 40. These models have different SSIM scores (ranging from 0.9320 to 0.9744) on the synthetic dataset according to channel dimensions. We use them as reference models for calibration to see the relationship between the metric in the synthetic data and the perceptual quality in real data. We ask 20 participants to assess the visual similarity between the input reference frame and the magnified frames generated by each models. We observe that a score of "3" (Adequate Similarity) corresponds to an SSIM of 0.966 by fitting a quadratic trend line to the (SSIM, human scored visual similarity). Based on this, we set a visual threshold of above 0.966 as the primary criterion for determining the acceptability of a proposed architecture, establishing a standard for visually acceptable quality in motion magnification models.

To extend the evaluation beyond baseline variants, we also conduct human study on across several methods. We select models whose SSIM on the synthetic dataset fall in a mid-range—neither too low nor too high (Singh et al., 2023a;b; Oh et al., 2018) including ours proposed model.

Prior to the human studies, participants are trained on examples of the motion magnification task to become acquainted with the task. Then, we ask 20 participants to rate visual similarity between the input reference frame and the magnified frame generated by each case, with a description "Please rate the visual similarity between the input reference image and the magnified frames on a scale of 0 to 5, where each score corresponds to No, Minimal, Moderate, Adequate, Strong, and Perfect similarity to the reference, respectively." Each study involves 10 samples; some samples differ across studies because certain models could not use temporal filtering.

The results from the study indicate that a score of "3" (Adequate Similarity) corresponds to an SSIM of approximately 0.966 (see Fig. A5 [Right]). Based on this observation, we establish a *visual threshold* level of above *0.966* as a primary criterion for determining the acceptability of a proposed architecture. This allows us to establish a threshold for visually acceptable generation quality that a motion magnification model should adhere to.

G Computation speed test on various hardware

As summarized in Table A9, we benchmark our model on three GPUs—RTX 3090 (server), RTX 3070 (desktop), and RTX 2080 Super (laptop), consistently achieving a 2.7× speed-up and reaching 25.7 FPS at 1080p (on the server). As summarized in Table A8, the model sustains real-time motion magnification ($\geq$24 FPS) at substantially higher input resolutions than Oh et al. (2018) on all devices, increasing the maximum processable pixels by 4× (RTX 3090/3070) to 5× (RTX 2080 Super).

Beyond offline benchmarks, we implemented a live webcam demo (540p) on the RTX 2080 Super laptop, which achieves real-time performance in an interactive setting. We include a live demonstration video of this

Table A8: **Maximum input resolution each GPU can process in real time ($\geq$ 24 FPS).** The maximum resolution is selected from the set $\mathcal{R} = \{240\text{p}, 360\text{p}, 480\text{p}, 540\text{p}, 720\text{p}, 1080\text{p}\}$.

GPU	Max Resolution $\geq$ 24 FPS	
	Baseline	**Ours**
RTX 3090 (server)	540p	1080p
RTX 3070 (desktop)	360p	720p
RTX 2080 Super (laptop)	240p	540p

Table A9: **Throughput at Full-HD (1080p) resolution.** Speed-up is the performance gain of our method over the baseline (Oh et al., 2018).

GPU	FPS @ 1080p		Speed-up
	Baseline	**Ours**	
RTX 3090 (server)	9.4	25.7	2.73×
RTX 3070 (desktop)	3.9	10.6	2.72×
RTX 2080 Super (laptop)	2.8	7.6	2.71×

code in the supplementary video. This demonstrates that the efficiency gains translate to practical usage scenarios. We will release the demo code upon acceptance.

H Discussion and limitations

In this section, we cover the temporal-filter trade-offs, real-world deployment challenges (large object motion and camera motion), and the limitations of our study.

Strengths of the structural efficiency We focus on developing a structurally efficient architectural design. Thus, our approach is orthogonal to dynamic pruning or quantization, which could be applied on top of our model if desired. For example, if quantization achieves a 2× acceleration, the overall speed-up would be multiplied, yielding approximately $2.7 \times 2 = 5.4\times$. This highlights that the architectural design we focus on provides a significant contribution.

Trade-off using temporal filter While temporal filtering is beneficial for suppressing noise and unwanted motion so that amplification can focus on the motion-of-interest and yield smoother results, it also increases computational cost and may unintentionally attenuate or remove small parts of the target motion. Therefore, when applying temporal filtering, it is crucial to choose the frequency band and filter type appropriately for the given scenario.

Large object motion For large object motion, magnifying such motion is not the scope of video motion magnification, as the primary goal of this technique is to reveal and visualize subtle motions that are otherwise difficult to perceive. In practice, large motions can be suppressed using complementary techniques such as temporal frequency filtering, depending on the frequency band of interest. More advanced temporal filtering approaches, such as Video Acceleration Magnification (Zhang et al., 2017) and Jerk-Aware Video Acceleration Magnification (Takeda et al., 2018), have also been proposed to specifically address large-magnitude motions.

Camera motion For camera motion, since video motion magnification inherently amplifies all types of motion, camera motion is also magnified. Unlike large object motion, this cannot be separated from object motion in current 2D motion magnification methods, and we therefore consider it a limitation of the existing framework. Recently, a 3D motion magnification approach (Feng et al., 2023), extended the setting to handle moving cameras using time-varying radiance fields, but they require multi-view data and NeRF-based representations, making the setup fundamentally different from 2D video motion magnification.

Limitations While we provide a comprehensive analysis on the baseline, some configuration choices (e.g., channel reduction and layer depth) may appear architecture-specific. In this context, we would like to position our results as design guidelines: careful treatment of spatial bottlenecks, channel widths, and module sensitivities can systematically improve efficiency across motion-magnification models. There is also a need for further investigation into an in-depth examination of training datasets and pipelines. Finally, although our model achieves real-time throughput, further optimization is needed for high-FPS capture scenarios.