HyperCUT: Video Sequence from a Single Blurry Image using Unsupervised Ordering

Bang-Dang Pham[1,*]   Phong Tran[1,2,*]   Anh Tran[1]   Cuong Pham[1,3]   Rang Nguyen[1]   Minh Hoai[1,4]

[1]VinAI Research, Vietnam   [2]MBZUAI, UAE   [3]Posts & Telecommunications Inst. of Tech., Vietnam   [4]Stony Brook University, USA

{v.dangpb1, v.anhtt152, v.hoainm}@vinai.io   cuongpv@ptit.edu.vn   the.tran@mbzuai.ac.ae

[*]Equal contribution

Abstract

We consider the challenging task of training models for image-to-video deblurring, which aims to recover a sequence of sharp images corresponding to a given blurry image input. A critical issue disturbing the training of an image-to-video model is the ambiguity of the frame ordering since both the forward and backward sequences are plausible solutions. This paper proposes an effective self-supervised ordering scheme that allows training high-quality image-to-video deblurring models. Unlike previous methods that rely on order-invariant losses, we assign an explicit order for each video sequence, thus avoiding the order-ambiguity issue. Specifically, we map each video sequence to a vector in a latent high-dimensional space so that there exists a hyperplane such that for every video sequence, the vectors extracted from it and its reversed sequence are on different sides of the hyperplane. The side of the vectors will be used to define the order of the corresponding sequence. Last but not least, we propose a real-image dataset for the image-to-video deblurring problem that covers a variety of popular domains, including face, hand, and street. Extensive experimental results confirm the effectiveness of our method. Code and data are available at https://github.com/VinAIResearch/HyperCUT.git

1. Introduction

Motion blur artifacts occur when the camera's shutter speed is slower than the object's motion. This can be studied by considering the image capturing process, in which the camera shutter is opened to allow light to pass to the camera sensor. This process can be formulated as:

$$y = g\left(\frac{1}{\tau}\int_0^\tau x(t)dt\right) \approx g\left(\frac{1}{N+1}\sum_{k=0}^N x_k\right), \quad (1)$$
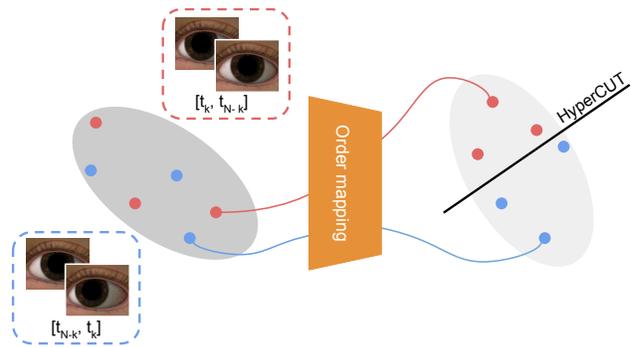

Figure 1. We tackle the order ambiguity issue by forcing the frame sequence to follow a pre-defined order. To find such an order, we map the frame sequences into a high dimensional space so that they are separable. The side (left or right of the hyperplane) is used to define the order of the frame sequence.

where y is the resulting image, $x(t)$ is the signal captured by the sensor at time t, g is the camera response function, and τ is the camera exposure time. For simplicity, we omit the camera response function in the notation. The image y can also be approximated by averaging $N+1$ uniform samples of the signal x, denoted as x_k with $k = \overline{0, N}$. For long exposure duration or rapid movement, these samples can be notably different, causing motion blur artifacts.

Image deblurring seeks to remove the blur artifacts to improve the quality of the captured image. This task has many practical applications, and it has been extensively studied in the computer vision literature. However, existing methods often formulate the deblurring task as an image-to-image mapping problem, where only one sharp image is sought for a given blurry input image, even though a blurry image corresponds to a sequence of sharp images. The image-to-image approach can improve the aesthetic look of a blurry image, but it is insufficient for many applications, especially the applications that require recovering the motion of objects, e.g., for iris or finger tracking. In this paper, we tackle the another important task of image-to-video deblurring, which we will refer to as *blur2vid*.

Image-to-video deblurring, however, is a non trivial task that requires learning a set of deblurring mapping functions $\{f_k\}$ so that $f_k(y) \approx x_k$. A naive approach is to minimize the squared difference between the predicted sharp image and the ground truth target, i.e.,

$$f_k = \underset{f}{\arg\min} \, \mathbb{E}_{x,y} ||f(y) - x_k||_2^2. \qquad (2)$$

However, this approach suffers from the order-ambiguity issue [5]. Considering Eq. (1), the same blurry image y is formed regardless of the order of the corresponding sampled sharp frames. For example, both $\{x_0, ..., x_N\}$ and the reversed sequence are valid solutions. Thus, x_k, x_{N-k}, and possibly other x_h's are valid 'ground truth' target for $f_k(y)$. Thus, optimizing Eq. (2) will lead to a solution where f_k is mapped to the average of x_k and x_{N-k}. This issue has also been observed in the work of [13] for future video frame prediction. This also explains why most existing deblurring methods cannot be directly used to recover any frame other than the middle one. To tackle this issue, Jin *et al.* [5] introduced the order-invariant loss, Eq. (3), which computed the total loss on frames at symmetric indexes (i.e., k and $N-k$). However, this loss does not fully resolve the issue of having multiple solutions, as will be demonstrated in Sec. 2.2.

This paper proposes a new scheme to solve the order ambiguity issue. Unlike the order-invariant loss [5] or motion guidance [30], we solve this problem directly by explicitly assigning which frame sequence is backward or forward. In other words, each sequence is assigned an order label 0 or 1 so that its label is opposite to the label of its reverse. Then the ambiguity issue can be tackled by forcing the model to learn to generate videos with the order label "0". We introduce **HyperCUT** as illustrated in Fig. 1 to find such an order. Specifically, we find a mapping $\mathcal{H}$ that maps all frame sequences into a high-dimensional space so that all pairs of vectors representing two temporal symmetric sequences are separable by a hyperplane. We dub this hyperplane HyperCUT. Each frame sequence's order label is defined as the side of its corresponding vector w.r.t. the hyperplane. We find the mapping $\mathcal{H}$ by representing it as a neural network and training it in an unsupervised manner using a contrastive loss.

Previously, there existed no real $blur2vid$ dataset, so another contribution of this paper is the introduction of a new dataset called Real $blur2vid$ (**RB2V**). RB2V was captured by a beam splitter system, similar to [18, 29, 31]. It consists of three subsets for three categories: street, face, and hand. We will use the last two to demonstrate the potential applications of the $blur2vid$ task in motion tracking.

In short, our contributions are summarized as follow:

- We introduce HyperCUT which is used to solve the order ambiguity issue for the task of extracting a sharp video sequence from a blurry image.

- We build a new dataset for the task, covering three categories: street, face, and hand. This is the first real and large-scale dataset for image-to-video deblurring.
- We demonstrate two potential real-world applications of image-to-video deblurring.

2. Related Work

2.1. Image deblurring

Image deblurring is a classical task in low-level computer vision. In the past, the blur kernel was assumed to be linear and uniform, and the blur model can be formulated as: $y = x * k + \eta$, where k is the blur kernel, x is the sharp image, $*$ denotes convolution operator, η is the white noise, and y is the corresponding blurry one. The main approach was to find a good prior for either the sharp images [1, 7, 8, 15, 24] or the blur kernel [12] space. However, the complexity of the optimization involved in these methods, along with their reliance on linear and uniform assumptions, renders them unsuitable for generalizing to real-world blur scenarios.

Thanks to the advance of deep neural networks in the past few years, the community has witnessed a significant leap in the deblurring field. Deep learning-based models do not make any explicit assumption on the blur operator nor on the sharp image space. Instead, they can learn to deblur using large-scale datasets. Zamir *et al.* [26] proposed a multi-stage architecture, where contextual information was learned in the earlier stages. In contrast, the whole input image was processed without any downsample operator to extract fine spatial details in the last stage. Tao *et al.* [20] employed a multi-scale recurrent network that deblurred the input image in a multi-scale and recurrent manner. Kupyn *et al.* [9, 10] introduced generative adversarial networks [3] for the deblurring task to make the deblurred image more realistic. However, the performance of deep deblurring models degrades significantly when the blur operator does not appear in the training set [22].

2.2. Recovery of multiple sharp frames

Jin *et al.* [5] were the first to introduce a model that took a blurry input y and produced multiple sharp frames $x_0, \ldots, x_6$. They trained seven networks $f_0, \ldots, f_6$, each corresponded to a sharp frame output. Their method was also the first to point out the order-ambiguity issue: finding a set of sharp frames given a single blurry input is an ill-posed problem since the generation of y is independent of the order of the sharp frames. [5] addressed this by introducing the order-invariant loss:

$$\mathcal{L}_{OI} = \sum_{k=0}^{2} (|\,||f_k(y) - f_{6-k}(y)|| - ||x_k - x_{6-k}||\,| \\ + |\,||f_k(y) + f_{6-k}(y)|| - ||x_k + x_{6-k}||\,|). \quad (3)$$
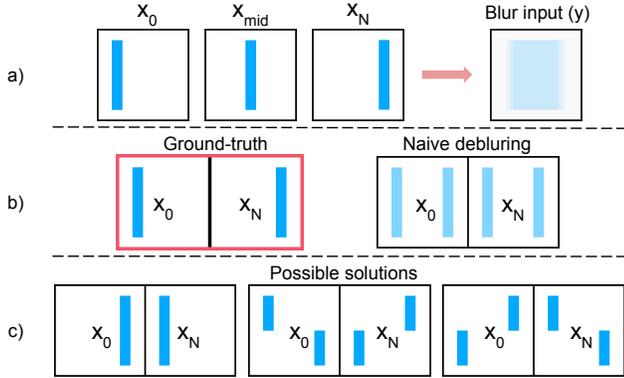
Figure 2. **Toy example.** Row (a) depicts the formation of a blurry image y from a sharp sequence. We consider the task of recovering border frames (x_0, x_N) from y. The ground-truth label is provided in the first column of Row (b). Due to the order-ambiguity issue, normal regression networks often return the blurry result as in the second column of Row (b). Order-invariant loss [5] accepts both the correct solution and three other ones in Row (c). Our proposed method only returns the correct solution (red-box).

However, this loss does not fully address the issue as illustrated in Fig. 2. Row (a) shows the formation of the blurry image y from a sequence of sharp images. Consider the sub-task of recovering the end frames (x_0, x_N) from the blurry one. The ground truth solution for this task is shown in the first column of Row (b). Due to the order-ambiguity issue, normal regression networks often return the blurry solution as in the second column of Row (b), in which each predicted frame is an average of the ground-truth pair. By applying the order-invariant loss [5], one can get sharp prediction outcomes. However, besides the correct frame pair, it also accepts three other solutions in Row (c). The first solution is just different from the ground truth by the frame order, while the other two are obviously wrong.

Purohit *et al.* [17] proposed a recurrent architecture that could be extended to generate any number of sharp frames without increasing the number of parameters. They trained a pair of recurrent video encoder and decoder to reconstruct a set of N continuous sharp frames. The encoder was then replaced by a blurred image encoder to form a network that could generate N sharp frames from a single blurry image. All these methods rely on the order-invariant loss [5] to avoid the order-ambiguity issue. However, this loss does not fully address the issue. The models proposed by [16, 23] do not suffer from order ambiguity, but they need additional data from an event camera.

Recently, Zhong *et al.* [30] proposed a different approach to solving the order ambiguity issue. Instead of focusing on the training loss, they converted the ill-posed $blur2vid$ task into a nearly deterministic one-to-one mapping problem by using motion guidance as an additional input. This motion guidance input was generated from the blurry image by a conditional Variation Autoencoder such that it was unique

for each solution. However, the model largely depended on the quality of the motion guidance and consequently failed when the human-annotated data was not available or the estimated optical flow was inaccurate. In addition, the motion guidance was built upon handcrafted heuristics and might not hold for every case, especially for complicated motion.

2.3. Real blur datasets

To train deep deblurring models, many large-scale sharp-blur pair datasets have been proposed. Tao *et al.* [20] introduced the GOPRO dataset, which consists of more than 1000 pairs of sharp images captured by a high-speed GOPRO 4 Hero Black and their corresponding synthetic blurry images. They generated blurry images by mimicking the blur generation process as described in Eq. (1). Nah *et al.* [14] proposed the REDS dataset with a similar synthesis method, but with more pairs, higher quality, and a different camera response function choice. [19] proposed a human-aware deblurring dataset that focused on human movements. Tran *et al.* [22] used a blur encoder to transfer blur operator from existing datasets to another sharp frame set. Zhong *et al.* [30] proposed B-Aist++ dataset, which was synthesized from dancing videos [11] to simulate complex human body movements.

Since deep deblurring models are highly overfitted to the blur operator used in the training dataset [22], real-image deblurring datasets are critical. Therefore, many have been introduced over the past few years [18, 29, 31]. These datasets were captured by a system that consists of high and low shutter speed cameras. Two cameras were placed on two sides of a beam splitter to capture the same scene. Existing datasets used for image/video deblurring are not sufficient to train $blur2vid$ models. Jin *et al.* [5] built a synthetic dataset by using seven consecutive frames and their average as ground-truth and input, respectively. To the best of our knowledge, there was no real-image deblurring dataset for the $blur2vid$ task.

3. Methodology

This section describes the proposed method. We assume there is training data of the form $\{(y^i, x_0^i, \ldots, x_N^i)\}_{i=1}^M$, where M is the number of training samples, and each training sample consists of a blurry image y^i and $N + 1$ sharp images. Our goal is to train neural networks that can recover all the sharp images from the blurry one.

3.1. HyperCUT order

One approach for the $blur2vid$ task is to pose it as multiple image-to-image deblurring tasks and train a separate network for each task. It means that for each target frame index $k \in [0..N]$, we train a network f_k to predict x_k^i from y^i by optimizing:
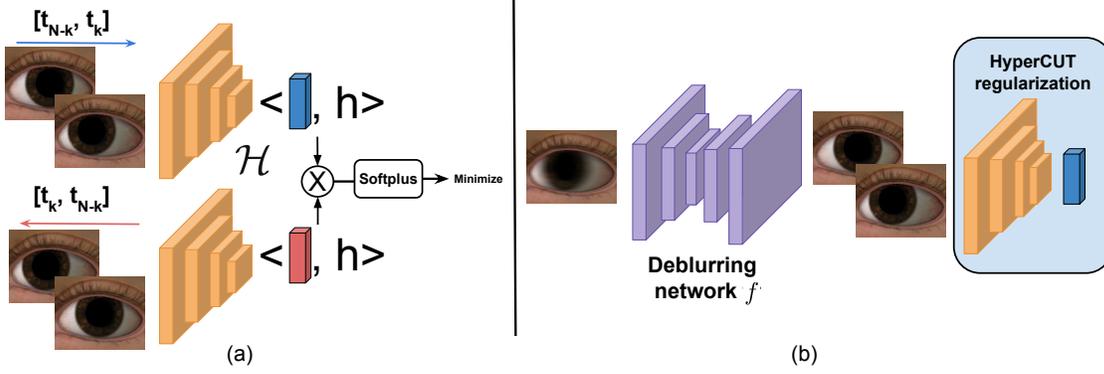
Figure 3. The overall architecture of our framework. (a) Given a frame sequence, we optimize the function $\mathcal{H}$ by forcing the sides of the vectors representing it and its reverse to be different, according to a fixed hyperplane h. (b) Having $\mathcal{H}$, we can use it as a regularization to make the deblurring network only predict frame sequences of the same side of h, thus solving the order-ambiguity issue.

$$\mathcal{L}_{rec} = \frac{1}{M} \sum_{i=1}^{M} \|f_k(y^i) - x_k^i\|. \qquad (4)$$

Unfortunately, this naive approach fails to produce a sharp output. Empirically, we observe that this approach tends to output an image that is close to $\frac{(x_k + x_{N-k})}{2}$. Conceptually, it is known that the output of f_k will converge to the average of all sampled targets used for training, which include both x_k and x_{N-k} due to the order-ambiguity issue.

Let h be a fixed hyperplane in a high-dimensional space. We want to find a mapping $\mathcal{H} : \mathbb{R}^{2 \times H \times W \times C} \to \mathbb{R}^d$ such that $\mathcal{H}\left([x_k^i, x_{N-k}^i]\right)$ and $\mathcal{H}\left([x_{N-k}^i, x_k^i]\right)$ are on different sides of h. In other words:

$$\left\langle \mathcal{H}\left([x_k^i, x_{N-k}^i]\right), h \right\rangle \left\langle \mathcal{H}\left([x_{N-k}^i, x_k^i]\right), h \right\rangle < 0, \qquad (5)$$

To find the mapping $\mathcal{H}$, we represent it by a neural network as shown in Fig. 3. The objective function is to minimize the left hand side of Eq. (5):

$$\mathcal{L}_h = \frac{1}{M} \sum_{i=1}^{M} softplus(\left\langle \mathcal{H}\left([x_k^i, x_{N-k}^i]\right), h \right\rangle$$
$$\times \left\langle \mathcal{H}\left([x_{N-k}^i, x_k^i]\right), h \right\rangle), \quad (6)$$

where $softplus(t) = \log(1 + e^x)$ is used to give less penalty for correct prediction (when the forward and backward sequences are on different sides of h).

In this work, we use a standard residual network [4] with a fully connected layer so that the output is a vector of length n, where n is a hyperparameter of the network. We sample a random hyperplane h in this n-dimensional space and fix it during training.

3.2. Addressing order-ambiguity with HyperCUT

The function $\mathcal{H}$ can be combined with other losses and used as a regularization to solve the order ambiguity issue
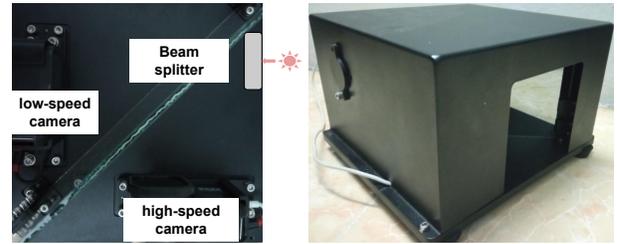

Figure 4. Beam splitter camera system: interior (top-view) and exterior (side-view).

as shown in Fig. 3b. Specifically, we force the vector corresponding to the output of the deblurring network to lie on only one side of the hyperplane. This can be done by adding to the training loss the following HyperCUT regularization:

$$\mathcal{R}_{hyp}(f) = \frac{1}{M} \sum_{i=1}^{M} \sum_{k=0}^{\lfloor N/2 \rfloor} \left\langle \mathcal{H}\left(f_k(y^i), f_{N-k}(y^i)\right), h \right\rangle \quad (7)$$

where $\mathcal{H}$ is the pretrained network described in Sec. 3.1 and it is frozen during the training of deblurring networks. This regularization enforces all synthesized pairs to stay on the "negative side" of the hyper-plane h. It can be combined with other losses. The final loss for model training will be:

$$\mathcal{L}(f) = \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}_D(f(y^i)) + \alpha \mathcal{R}_{hyp}(f), \qquad (8)$$

where $\mathcal{L}_D$ can be any $blur2vid$ loss, such as regular L_2 loss [30] or order-invariant loss [5], and α is the weight of the HyperCUT regularization. This loss is differential w.r.t. f and can be optimized using any gradient-based optimizer.

4. Real $blur2vid$ (RB2V) Dataset

Due to the difficulty of collecting paired blurry and sharp video sequences, previous deblurring works utilized syn-
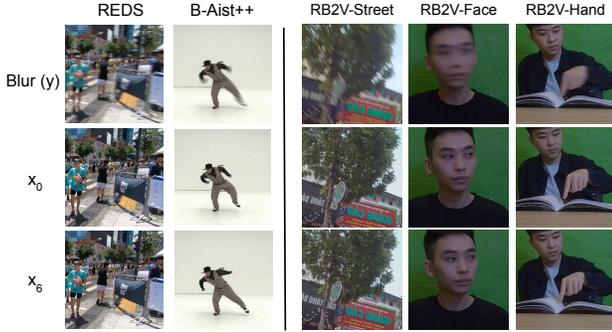
Figure 5. Comparing $blur2vid$ datasets. Sample images from the proposed RB2V dataset are on the right, which are real images as opposed to synthetic ones of existing datasets on the left.

Data subset	#data samples	
	Train	Test
RB2V-Street	9000	2053
RB2V-Face	8000	2157
RB2V-Hand	12000	4722

Table 1. Statistics of our in-the-wild $blur2vid$ dataset.

thetic datasets [14, 20]. Although these datasets are formed by mimicking the camera process (Eq. (1)), there is a significant gap between synthetic and real blur [22]. Recently, beam splitter deblurring datasets [18, 29] have been proposed and brought remarkable advances in image deblurring. However, such kinds of datasets are not available for the $blur2vid$ task. This poses the need for the collection of a new dataset for this task. This section describes our data collection procedure.

4.1. Data collection

Following recent works [18, 29, 31], we built a beam-splitter camera system for collecting real data. This system had two GoPro Hero-8 cameras and a beam splitter as shown in Fig. 4. One camera captured videos at 25fps, and the second camera captured at 100fps.

We used this beam splitter system to collect three categories of data: street, hand, and face, which will be referred to as RB2V-Street, RB2V-Hand and RB2V-Face respectively. For the RB2V-Street dataset, we captured various scenes with different and non-uniform moving objects, creating a diverse dataset with many blur types and intensities. As for RB2V-Face and RB2V-Hand, we captured data for 25 objects in an laboratory environment with a green screen, which was then be replaced with a random static background. For each face or hand object, we captured it with different distances and movements. We divided the datasets into disjoint training and testing sets. Tab. 1 reports the statistics of this dataset, and Fig. 5 provides some data samples in the last three columns.

4.2. Data processing

Spatial alignment. Although we tried our best to position the cameras to capture exactly the same scene, there might still be some misalignment between the captured images. To correct for the misalignment, we calibrated the cameras and performed homography mapping.

Temporal upsampling. The low-speed camera was four times slower than the high-speed one, so each blurry frame corresponded to four sharp ones. Since the previous works typically used seven, we temporally upsampled the frame sequence captured by the 100fps camera by a factor of two. After this interpolation step, each blurry frame corresponded to seven sharp frames, including four original frames and three interpolated ones. We used [2] as the interpolation module.

Color correction and temporal alignment. Let $C_{x,y}(z)$ denote the color correction algorithm that applies the correction matrix calculated from a reference pair $\{x, y\}$ to an image z. Details of this algorithm are given in the supplementary materials. Also denote y_i^{fake} as the synthetic blurry frame generated from the consecutive sharp frames $\{x[i], x[i+1], ..., x[i+6]\}$ by temporally upsampling this set to a higher frame rate as in [14] and average all of them. We interpolated two extra frames in between for each consecutive frames, so the number of frames in the upsampled sequence was 19; this helped the synthetic blurry image be more realistic. To find the sharp sequence that corresponded to y, denoted as $\mathcal{X} = x_0, x_1, ..., x_6$, we needed to find a color correction map C^* and a position p so that

$$\mathcal{X} = \{C^*(x[p]), C^*(x[p+1]), ..., C^*(x[p+6])\} \quad (9)$$

To find C^* and p, we found the seven consecutive sharp frames such that the "fake" blurry image generated by them after color correction was the closest to the real blurry one. If the camera response function g was linear, we have:

$$y \approx \frac{\sum_{i=0}^{N} C^*(x[i])}{N+1} = C^*\left(\frac{\sum_{i=0}^{N}(x[i])}{N+1}\right) = C^*\left(y_i^{fake}\right).$$

From the above equation, if we apply C^* to y_i^{fake}, y_i^{fake} will become y. This observation suggests that C^* can be approximated by $C_{y_i^{fake},y}$.

In summary, the position p was found by optimizing:

$$p = \arg\min_{i} PSNR\left(C_{y_i^{fake},y}\left(y_i^{fake}\right), y\right). \quad (10)$$

The sharp-image sequence $\mathcal{X}$ was taken as the set $\{C_{y_p^{fake},y}(\{x[p]\}), C_{y_p^{fake},y}(\{x[p+1]\}), ..., C_{y_p^{fake},y}(\{x[p+6])\}\}$. More details are given in the supplementary materials.

Model		1st	2nd	3rd	4th	5th	6th	7th
[5]		20.65	22.63	24.20	23.50	24.20	22.63	20.65
[5] + HyperCUT	REDS	**22.87**	**24.88**	**26.29**	**25.10**	**26.29**	**24.88**	**22.86**
[17]		22.78	24.47	26.14	**31.50**	26.12	24.49	22.83
[17] + HyperCUT		**26.75**	**28.30**	**29.42**	29.97	**29.41**	**28.30**	**26.76**
[17]		26.99	27.99	29.45	**32.08**	29.55	28.06	27.04
[17] + HyperCUT	RB2V	**28.29**	**29.20**	**30.43**	**32.08**	**30.53**	**29.22**	**28.25**

Table 2. pPSNR scores (dB) between predicted frames and the ground-truth ones on the synthetic $blur2vid$ REDS dataset and our proposed real $blur2vid$ dataset (we get the average result in all categories including hand, face and street).
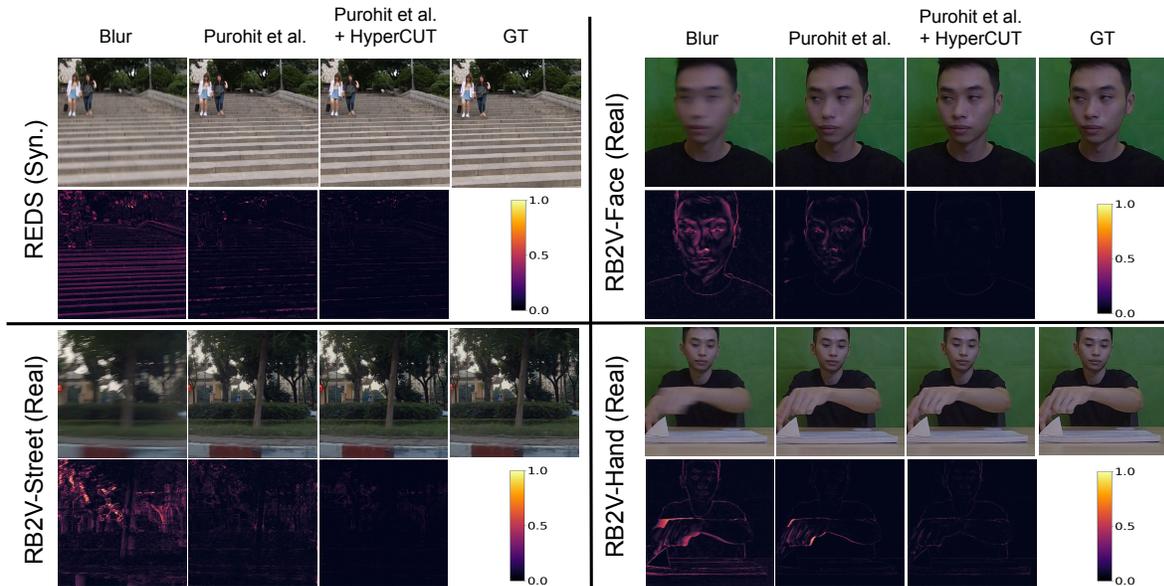

Figure 6. Qualitative results for the first frame prediction on different $blur2vid$ datasets: REDS, RB2V-Street, RB2V-Face, and RB2V-Hand. From left to right on each dataset: blurry input, the result of [17], our prediction result applying HyperCUT, and the ground truth. Each result includes a color image and an error heatmap.

5. Experiments

We compare the proposed ordering scheme applying to the publicly available $blur2vid$ model proposed by [5, 17] on both the existing and the proposed RB2V datasets. In addition, to study the contribution of our proposed mapping, we examine HyperCUT on [30] with the same settings on the B-Aist++ dataset [11, 27].

5.1. Dataset preparation

Synthetic datasets. We used the 120fps set of the REDS dataset [14] to synthesize the training and the testing set (the first row of Fig. 5). Specifically, for every four consecutive frames in the set, we interpolated one intermediate frame between each consecutive pair using CDFI model [21], form a sequence of seven frames, and generated the corresponding blurry image. This allowed us to compare our method with the model proposed by [5], which fixed the number of frames per sequence to seven. In addition,

we synthesized another testing set using Vimeo90K [25]. Compared to REDS, this dataset had many more scenes but provided only three frames per data point. Hence, it was suitable for the ablation studies, which required only ground truth on two border frames, but unsuitable for other evaluations. From the three original frames, we interpolated two frames in the middle of each consecutive pair, forming a 7-frame sequence and generating the corresponding blurry image by the same procedure. As for the B-Aist++ dataset, we used the augmentation and setting proposed in the original paper that croped the main character using a given bounding box to compare the results from our method and their model.

RB2V dataset. We also evaluated the models on our proposed real $blur2vid$ dataset RB2V on all the three categories. For each category, we re-trained our model and the baselines and tested on the testing set of the same category.

5.2. Implementation details

All the models used in the experiments were trained using the Adam optimizer [6]. Training our model took roughly one day for 100 epochs on a single Nvidia A100 GPU. For fair comparison, we re-trained the baseline model on both synthetic and our real datasets.

5.3. Order Accuracy of HyperCUT

We trained our HyperCUT model in both synthetic and real datasets to regularize the corresponding $blur2video$ task. In all experiments, we used output vectors of length $n = 128$. For evaluation, we proposed new metrics that overcame the limitations of existing ones in analyzing the effectiveness of our scheme:

• hit: is the ratio of frame pairs (x_k, x_{N-k}) that satisfy:

$$\langle \mathcal{H}([x_k, x_{N-k}]), h \rangle \langle \mathcal{H}([x_{N-k}, x_k]), h \rangle < 0$$

• con: measures the **con**sistency of frame pairs in each sequence in the HyperCUT space. It computes the ratio that the pairs $(x_1, x_7), (x_2, x_6)$, and (x_3, x_5) are in the same side of the hyperplane h.

As can be seen in Tab. 3, our proposed self-supervised model can extract the ordering information effectively in all mentioned datasets. The trained models achieve almost perfect scores in all metrics. We use t-SNE to visualize representation vectors in Fig. 7. It can be observed that the mapped vectors are perfectly split into two clusters in both REDS and RV2B-Street datasets.
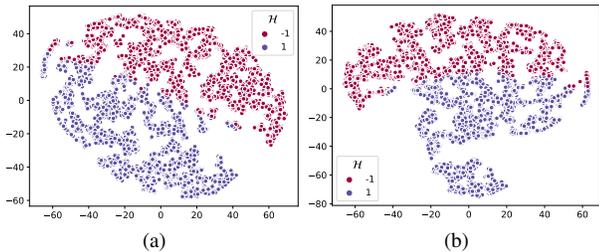

(a)   (b)

Figure 7. The t-SNE visualization of HyperCUT ordering mapping on (a) RV2B-Street (Real) and (b) REDS (Synthetic) datasets. We use -1 and 1 to represent each side of hyperplane.

Dataset	hit	$con@2$	$con@3$
REDS	95.7	96.5	94.4
B-Aist++	97.5	95.6	91.2
RB2V-Face	94.4	96.7	92.1
RB2V-Hand	98.6	97.0	96.7
RB2V-Street	98.7	98.3	96.8

Table 3. The experiments of HyperCUT on five datasets with hit (%) and con (%) rates. We denote $con@X$ as the consistency rate of HyperCUT ordering for two frame pairs $(x_i, x_{N-i}), (x_j, x_{N-j})$ (when X = 2) and for 3 frame pairs $(x_i, x_{N-i}), (x_j, x_{N-j}), (x_k, x_{N-k})$ (when X = 3) in the same side of the hyperplane.

5.4. HyperCUT Regularization

Synthetic datasets. We studied the HyperCUT regularization with the methods proposed by [5, 17] and [30] on the REDS and B-Aist++ datasets, using a default weight $\alpha = 0.2$.

With the REDS dataset, we re-trained the models proposed in [5] and [17] with their original loss functions and with our proposed HyperCUT add-on. For evaluation, since $\mathcal{L}_{OI}$ accepts any frame ordering, we define a paired-based PSNR, denoted as pPSNR, that computes the maximum average of PSNR scores between the regressed and ground-truth symmetric frame pair in forward and backward order. Specifically, given the output $(x'_0, x'_1, ..., x'_N)$ and the ground-truth $(x_0, x_1, ..., x_N)$, pPSNR can be computed as:

$$pPSNR_k(x, x') = max(PSNR(x'_k, x_k),$$
$$PSNR(x'_k, x_{N-k})) \quad (11)$$

Quantitative results are given in Tab. 2, where we use pPSNR scores to measure the performance of the models. Our HyperCUT-based models provide stable performance on all frames and consistently outperform the compared models on all six border frames with 1-4 point gaps in pPSNR scores. Since the backbone used in [5] is weak and outdated, from now on, we will focus on the models with the more recent and stronger backbone of [17]. We notice that the performance gap caused by HyperCUT regularization increases when moving to the boundary frames x_0 and x_7. The compared model, however, performs better at the center frame with an exceptionally high pPSNR. This model performs poorly on border frames, meaning that its loss concentrates on improving the quality of the middle frame. Our model, on the other hand, has a balance in improving all frames together. While our pPSNR score on the middle frame is not as high, we can easily improve it by deploying an extra, normal image deblurring network. The border frames, on the other hand, can only be learned effectively with our proposed HyperCUT regularization. An example is shown in the top left of Fig. 6. The result produced by our model is sharper and closer to the ground truth.

In addition, with the benchmark proposed by Zhong *et al*. [30] on the B-Aist++ dataset, we re-train the model with the same setting as the original model for a fair comparison, evaluating by average pPSNR, pSSIM, and pLPIPS[28] metrics, in which pSSIM and pLPIPS are defined similar to pPSNR. We denote these metrics as $p\overline{PSNR}$, $p\overline{SSIM}$, and $p\overline{LPIPS}$, respectively. As can be seen in Tab. 4, the result with HyperCUT regularization dominates the one reported from the paper as well as the reproduced version. The score difference between the two versions of the original method also reveals the instability of the motion guidance module.

RB2V dataset. We also ran evaluation on our proposed RB2V dataset. We trained the models using the training set

Method	[30] (from paper)	[30] (reproduced)	[30] + HyperCUT
$\mathcal{P}_1$	19.97 / 0.860 / 0.089	20.58 / 0.890 / 0.068	22.16 / 0.901 / 0.102
$\mathcal{P}_3$	22.44 / 0.890 / 0.068	21.21 / 0.899 / 0.063	23.31 / 0.915 / 0.062
$\mathcal{P}_5$	23.49 / 0.911 / 0.060	22.48 / 0.903 / 0.061	23.81 / 0.920 / 0.060

Table 4. **Quantitative evaluation of the blurry image decomposition**. $p\overline{\text{PSNR}}\uparrow$, $p\overline{\text{SSIM}}\uparrow$, and $p\overline{\text{LPIPS}}\downarrow$ are used as evaluation metrics. For Zhihang et al. [30], we predict multiple motion guidance from the guidance predictor network. $\mathcal{P}_\#$ denotes we evaluate # number of plausible decomposition results for each input, and choose the best. The results of Zhihang et al. [30] with the HyperCUT regularization represent the best performance calculated using either the forward or reverse outputs, following the original paper.

Deblur method	Face	Hand
Purohit *et al*. [17]	5.75	11.67
Purohit *et al*. [17] + HyperCUT	**4.87**	**9.2**

Table 5. Quantitative results for face and hand trajectory recovery from a single blurry image.

of each category and then test on the corresponding test set. The quantitative and qualitative results are given in Tab. 2 and the bottom left of Fig. 6, respectively. Again, our model shows better accuracy and reconstruction quality than [17].

Applications of $blur2vid$ **for faces and hands.** We tested the models on domain-specific datasets and measured the ability of recovering face and hand trajectories from a single blurry image. Given a blurry face image, we first run a $blur2vid$ model to obtain a sequence of sharp images, each of which would subsequently be fed into a facial landmark detection algorithm to detect 68 facial landmarks. To measure the quality of a recovered face trajectory, we calculated the Mean Squared Error (MSE) between the 68 facial landmarks detected on the recovered sharp image and the 68 facial landmarks detected on the ground truth sharp image. Similarly for hands, we detected the tip of the index finger in each recovered sharp image using the hand detection algorithm [27], and calculated its distance to the index finger detected on the ground truth sharp image. Quantitative results of face and hand trajectory recovery are given in Tab. 5. Compared to the baseline, the proposed model with HyperCUT regularization was more accurate, with reasonably small error for practical applications.

5.5. Ablation Studies

Regularization weight for HyperCUT. We ablated the weight parameter α to have a deeper understanding of its effect on the final performance. We experimented with different values for α from 0 to 0.3 with Purohit *et al*. [17] backbone on the RB2V-Street dataset, and computed the mean pPSNR score, denoted as $\overline{\text{pPSNR}}$. The results are reported in Tab. 6. As can be seen, when α was increased, the $\overline{\text{pPSNR}}$ score gradually increased and peaked at $\alpha{=}0.2$, confirming the positive contribution of the HyperCUT loss. When $\alpha{>}0.2$, the ordering information started to outweigh the order-invariant loss, decreasing the score. Hence, we se-

lected $\alpha = 0.2$ as the default setting for other experiments.

α	0	0.1	0.15	0.2	0.25	0.3
$\overline{\text{pPSNR}}$ (dB)	25.73	25.8	26.33	**26.95**	26.9	25.75

Table 6. The $\overline{\text{pPSNR}}$ (dB) of seven generated sharp frames on the RB2V-Street dataset when changing α.

Dimension of the hyperplane. We experimented with different settings for the number of dimensions of the hyperplane, and Tab. 7 shows the hit and consistency ratios of HyperCUT on the RB2V-Street and RB2V-Hand datasets. As can be seen, with $n \geq 16$, the accuracy of the order assigning is consistent with a small variance. In most of our experiments, we used $n = 128$ due to its best overall performance in terms of hits and cons ratios.

n	RB2V-Street			RB2V-Hand		
	hit	*con@2*	*con@3*	*hit*	*con@2*	*con@3*
1	95.5	97.7	94.5	95.0	96.3	92.5
16	98.4	98.2	96.1	96.8	96.6	92.8
64	**99.1**	98.4	96.4	97.4	96.0	93.5
128	98.7	**98.5**	**96.8**	**98.6**	**97.0**	**96.7**
256	97.5	97.7	95.2	96.9	96.8	93.3

Table 7. Ablation study for n, the dimension of the HyperCUT hyperplane.

6. Conclusions

In this paper, we have proposed a method for the $blur2vid$ task, effectively addressing the order-ambiguity issue with an innovative regularization called HyperCUT. The regularization assigns an order label to each potential solution and enforces the $blur2vid$ model to generate only that specific solution, thereby enhancing its performance. The proposed regularization can be implemented with any existing $blur2vid$ model for substantial improvements. Furthermore, we contributed a novel dataset for the development and evaluation of the image-to-video deblurring task. This dataset comprises real images from three distinct domains, namely street, face, and hand.

In this work, we focus on standard motion blur in normal capturing conditions with short exposure time, resulting in simple and consistent direction and velocity. Future research on adapting HyperCUT for handling complex movements and long exposure blur would be an interesting avenue for exploration.

References

[1] Tony F Chan and Chiu-Kwong Wong. Total variation blind deconvolution. *IEEE Transactions on Image Processing*, 7(3):370–375, 1998.

[2] Tianyu Ding, Luming Liang, Zhihui Zhu, and Ilya Zharkov. Cdfi: Compression-driven network design for frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[5] Meiguang Jin, Givi Meishvili, and Paolo Favaro. Learning to extract a video sequence from a single motion-blurred image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Dilip Krishnan and Rob Fergus. Fast image deconvolution using hyper-laplacian priors. *Advances in Neural Information Processing Systems*, 22:1033–1041, 2009.

[8] Dilip Krishnan, Terence Tay, and Rob Fergus. Blind deconvolution using a normalized sparsity measure. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2011.

[9] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiˇri´ Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[10] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *Proceedings of the International Conference on Computer Vision*, 2019.

[11] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++. In *Proceedings of the International Conference on Computer Vision*, 2021.

[12] Guangcan Liu, Shiyu Chang, and Yi Ma. Blind image deblurring using spectral properties of convolution operators. *IEEE Transactions on Image Processing*, 23(12):5047–5056, 2014.

[13] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *Proceedings of International Conference on Learning and Representation*, 2016.

[14] Seungjun Nah, Radu Timofte, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[15] Jinshan Pan, Deqing Sun, Hanspeter Pfister, and Ming-Hsuan Yang. Deblurring images via dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10):2315–2328, 2017.

[16] Liyuan Pan, Cedric Scheerlinck, Xin Yu, Richard Hartley, Miaomiao Liu, and Yuchao Dai. Bringing a blurry frame alive at high frame-rate with an event camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[17] Kuldeep Purohit, Anshul Shah, and AN Rajagopalan. Bringing alive blurred moments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[18] Jaesung Rim, Haeyun Lee, Jucheol Won, and Sunghyun Cho. Real-world blur dataset for learning and benchmarking deblurring algorithms. In *Proceedings of the European Conference on Computer Vision*. Springer, 2020.

[19] Ziyi Shen, Wenguan Wang, Xiankai Lu, Jianbing Shen, Haibin Ling, Tingfa Xu, and Ling Shao. Human-aware motion deblurring. In *Proceedings of the International Conference on Computer Vision*, 2019.

[20] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[21] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision*. Springer, 2020.

[22] Phong Tran, Anh Tuan Tran, Quynh Phung, and Minh Hoai. Explore image deblurring via encoded blur kernel space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[23] Fang Xu, Lei Yu, Bishan Wang, Wen Yang, Gui-Song Xia, Xu Jia, Zhendong Qiao, and Jianzhuang Liu. Motion deblurring with real events. In *Proceedings of the International Conference on Computer Vision*, 2021.

[24] Li Xu, Shicheng Zheng, and Jiaya Jia. Unnatural l0 sparse representation for natural image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[25] 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.

[26] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[27] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[28] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[29] Zhihang Zhong, Ye Gao, Yinqiang Zheng, and Bo Zheng. Efficient spatio-temporal recurrent neural network for video deblurring. In *Proceedings of the European Conference on Computer Vision*. Springer, 2020.

[30] Zhihang Zhong, Xiao Sun, Zhirong Wu, Yinqiang Zheng,

Stephen Lin, and Imari Sato. Animation from blur: Multi-modal blur decomposition with motion guidance. In *Proceedings of the European Conference on Computer Vision*. Springer, 2022.

[31] Zhihang Zhong, Yinqiang Zheng, and Imari Sato. Towards rolling shutter correction and deblurring in dynamic scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.