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

Bang-Dang Pham1,* Phong Tran1,2,* Anh Tran1 Cuong Pham1,3 Rang Nguyen1 Minh Hoai1,4
1VinAI Research, Vietnam 2MBZUAI, UAE 3Posts & Telecommunications Inst. of Tech., Vietnam 4Stony 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:

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

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 $\tau$ 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 $k = 0, N$. 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.
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, \( \eta \) 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:

\[
L_{O1} = \sum_{k=0}^{2} \left( \| f_k(y) - f_{6-k}(y) \| - \| x_k - x_{6-k} \| \right) + \| f_k(y) + f_{6-k}(y) \| - \| x_k + x_{6-k} \|. \tag{3}
\]
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 \( x_k \) from \( y^i \) by optimizing:

![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).](image-url)
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 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:

$$R_{\text{hyp}}(f) = \frac{1}{M} \sum_{i=1}^{|\mathcal{N}/2|} \| \mathcal{H}(x_k^i, x_{N-k}^i) \|,$$

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:

$$L(f) = \frac{1}{M} \sum_{i=1}^M L_D(f(y^i)) + \alpha R_{\text{hyp}}(f),$$

where $L_D$ can be any blur2vid loss, such as regular $L_2$ loss [30] or order-invariant loss [5], and $\alpha$ 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-
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

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

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:

\[
p = \arg \min_{i} \text{PSNR} \left( C_{y_i^{fake}, y} \left( y_i^{fake} \right), y \right).
\]

The sharp-image sequence \( \mathcal{X} \) was taken as the set \( \{C_{y_0^{fake}, y}(\{x[p]\}), C_{y_1^{fake}, y}(\{x[p+1]\}), ..., C_{y_6^{fake}, y}(\{x[p+6]\})\} \). More details are given in the supplementary materials.
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).

<table>
<thead>
<tr>
<th>Model</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>7th</th>
</tr>
</thead>
<tbody>
<tr>
<td>[17]</td>
<td>26.99</td>
<td>27.99</td>
<td>29.45</td>
<td><strong>32.08</strong></td>
<td>29.55</td>
<td>28.06</td>
<td>27.04</td>
</tr>
<tr>
<td>[17] + HyperCUT</td>
<td><strong>28.29</strong></td>
<td><strong>29.20</strong></td>
<td><strong>30.43</strong></td>
<td><strong>32.08</strong></td>
<td><strong>30.53</strong></td>
<td><strong>29.22</strong></td>
<td><strong>28.25</strong></td>
</tr>
</tbody>
</table>

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

- **con**: measures the consistency 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.

![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.](image)

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 \( \mathcal{L}_{OJ} \) 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:

\[
pPSNR_k(x, x') = \max(pSNR(x'_k, x_k), pSNR(x'_k, x_{N-k}))
\]  

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 metrics, in which pSSIM and pLPIPS are defined similar to pPSNR. We denote these metrics as pPSNR, pSSIM, and pLPIPS, 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...
The results are PSNR score gradually increased and peaked at PSNR (dB) of seven generated sharp frames on the backbone on the RB2V-Street dataset, and computed the representative 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.

We ablated the Regularization weight for HyperCUT.

5.5. Ablation Studies

Regularization weight for HyperCUT. We ablated the weight parameter $\alpha$ to have a deeper understanding of its effect on the final performance. We experimented with different values for $\alpha$ from 0 to 0.3 with Purohit et al. [17] backbone on the RB2V-Street dataset, and computed the mean pPSNR score, denoted as pPSNR. The results are reported in Tab. 6. As can be seen, when $\alpha$ was increased, the 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 selected $\alpha = 0.2$ as the default setting for other experiments.

Table 4. Quantitative evaluation of the blurry image decomposition. $p_{PSNR}$↑, $p_{SSIM}$↑, and $p_{LPIPS}$↓ are used as evaluation metrics. For Zhihang et al. [30], we predict multiple motion guidance from the guidance predictor network. $P_\alpha$ 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Face</th>
<th>Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purohit et al. [17]</td>
<td>5.75</td>
<td>11.67</td>
</tr>
<tr>
<td>Purohit et al. [17] + HyperCUT</td>
<td>4.87</td>
<td>9.2</td>
</tr>
</tbody>
</table>

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

Applications of blurry2vid 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 blurry2vid 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.

<table>
<thead>
<tr>
<th>$n$</th>
<th>RB2V-Street</th>
<th>RB2V-Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>hit</td>
<td>con@2</td>
</tr>
<tr>
<td>1</td>
<td>95.5</td>
<td>97.7</td>
</tr>
<tr>
<td>16</td>
<td>98.4</td>
<td>98.2</td>
</tr>
<tr>
<td>64</td>
<td>99.1</td>
<td>98.4</td>
</tr>
<tr>
<td>128</td>
<td>98.7</td>
<td>98.5</td>
</tr>
<tr>
<td>256</td>
<td>97.5</td>
<td>97.7</td>
</tr>
</tbody>
</table>

Table 6. The pPSNR (dB) of seven generated frames on the RB2V-Street dataset when changing $\alpha$.

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.

<table>
<thead>
<tr>
<th>$n$</th>
<th>RB2V-Street</th>
<th>RB2V-Hand</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>hit</td>
<td>con@2</td>
</tr>
<tr>
<td>1</td>
<td>95.5</td>
<td>97.7</td>
</tr>
<tr>
<td>16</td>
<td>98.4</td>
<td>98.2</td>
</tr>
<tr>
<td>64</td>
<td>99.1</td>
<td>98.4</td>
</tr>
<tr>
<td>128</td>
<td>98.7</td>
<td>98.5</td>
</tr>
<tr>
<td>256</td>
<td>97.5</td>
<td>97.7</td>
</tr>
</tbody>
</table>

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

6. Conclusions

In this paper, we have proposed a method for the blurry2vid 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 blurry2vid model to generate only that specific solution, thereby enhancing its performance. The proposed regularization can be implemented with any existing blurry2vid 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


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