Deep-Learning Course (05107-25501)

Final Project

Omri Berman - 305113458

March 2, 2021

Abstract

Motion blur is a common issue in our lifetime, as the usage of hand-held cameras with ranging qualities becomes increasingly popular. While many techniques attempt to apply a straight-forward deblurring of the captured scene, a novel notion struggles to reproduce a sharp video based on a single blurry frame. In this project, I inspect a fusion between the SOTA methods for video-from-image reproduction and a computational-imaging based technique that proved efficient for single-frame deblurring. I demonstrate the qualitative advantage of the proposed fusion compared to the SOTA blind-deblurring based solutions over a novel dataset with a wide variety of real-world scenes.

Contents

1 Introduction **3**

2 Related Work **3**
2.1 Blind Motion Deblurring . . . 3
2.1.1 Single Image Deblurring . . . 3
2.1.2 Video Deblurring . . . 4
2.1.3 Video from Motion . . . 4
2.2 Computational-Imaging Based Deblurring . . . 4

3 Data **5**

4 Methods **6**
4.1 Video-From-Motion Network . . . 7
4.1.1 Architecture . . . 7
4.1.2 Implementation . . . 8
4.2 Phase-Coding Acquisition Simulation . . . 9
4.3 Results Evaluation . . . 10
4.3.1 PSNR . . . 10
4.3.2 SSIM . . . 11

5 Experiments **11**
5.1 Baseline Generation . . . 12
5.2 Phase-Masked Data Contribution . . . 13
5.3 Final Results . . . 14
5.3.1 Quantitative . . . 14
5.3.2 Qualitative . . . 15

6 Conclusions **16**

7 Appendix **18**

1 Introduction

Motion blur is a phenomenon present in both images and videos acquired by cameras, in which some (or all) objects in the scene are being smeared along the spatial dimensions, due to motion. The motion that introduces the blur is either of the objects that appear in the scene, or the camera that captures the scene, at the time of the acquisition. The field of image/video motion de-blurring has become of great interest over the last years. One possible explanation for that is the increasingly growing role of smartphones as personal cameras. Smartphones are hand-held devices, hence are not fully static at the moment of the image acquisition. Combining that with the frequent use-case of capturing free-moving objects and dynamic scenes, motion-blur is becoming increasingly more present in real-world images.

In this project, I focused on the specific aspect of video-from-motion, which is the reconstruction of a sharp video based on a single blurry input frame. The idea behind this approach is that a blurry frame encapsulates a dynamic that occurred in the scene, which is intuitively described by a series of frames (video) rather than a single, static frame. Unlike the single frame deblurring or video deblurring, the field of video from motion is fairly novel and hence related works are rarely found in the literature. While posing a challenge of few references to rely on, it's also a great source of motivation, as new advances in this field are yet to be made.

My contribution is based on a fusion of the pioneer paper in the field of video-from-motion [3], and a computational-imaging based deblurring technique that was suggested by Elmalem et al. [1]. In addition, I built a dedicated training and validation datasets, based on the novel REDS [7] dataset, which was not yet available when both mentioned papers were published. Utilizing the computational-imaging technique and the more abundant dataset, I was able to qualitatively outperform [3] in the majority of test cases, as described in the *experiments* section.

2 Related Work

2.1 Blind Motion Deblurring

In the regular case of blurry image acquisition by regular cameras, the resulting image is the only input containing clues that can be used for the de-blurring procedure. In such case, there is no clue whatsoever regrading the blurring kernel that causes the motion blur, which is generally shift variant and different for each object in the scene. In that sense, the de-blurring algorithm is blind with respect to the blurring kernel, hence the name of this field.

2.1.1 Single Image Deblurring

The most popular aspect of motion deblurring is the single image deblurring, which is essentially the restoration of a single sharp frame based on a single blurry input frame. Several different methods, both classic-algorithms based and machine-learning based were proposed over the years to tackle this problem.

Several classic-computer-vision studies have been conducted in the past, aiming to remove blurs from photography by suggesting either a camera motion parametrization or a generic scene motion parametrization [2, 13, 5]. Some of those proposed algorithms used

energy-optimization methods to estimate the sharp image based on the blurry input image. More modern machine-learning-based approaches rely on large-scale data sets for model training. Pioneer algorithms tried to estimate the camera's blur kernel by obtaining the local motion field using neural networks [12]. More recent approaches use an end-to-end learning methods that produce the deblurred image directly from the blurred input, avoiding the explicit estimation of the blurring kernel. Nah et al. [6] presented an end-to-end multi-scale convolutional-neural network (CNN) that restores sharp images from blurry inputs, where the blur was caused by several sources. In contrast to many prior approaches, this approach makes no assumption or restriction regarding the blurring kernel model, and does not explicitly estimate the blurring kernel.

2.1.2 Video Deblurring

In contrast to Single-image deblurring, video deblurring aims at recovering multiple sharp frames based on multiple blurry input frames (not necessarily of the same amount). The key-idea in this field is that the temporal relation between the adjacent input frames can be exploited to provide better results. Su et al. [11] focused on camera-shakes originated blur, and used an end-to-end CNN based implementation to avoid the expensive pre-alignment phase of the blurry frames, while still ending up with satisfying de-blurred results. Zuckerman et al. [8] utilized the recurrence of spatio-temporal (ST) patches across the different video dimensions (by swapping the space and time dimensions). This approach uses the notion of self learning (i.e. learning based on the specific input video), and is used for both spatial and temporal super-resolution, in addition to motion deblurring.

2.1.3 Video from Motion

Video from motion is the notion of extracting video from a single image, i.e. extracting multiple sharp frames from a single blurry one. The motivation behind this approach is that blurry images are the product of a long camera exposure (compared to the scene dynamics), which could be though of as an accumulation of many sharp frames captured with a short exposure. This approach is further motivated by the artificial generation of blurry images for dataset generations, which is essentially done by averaging over several sharp adjacent frames. Jin et al. [3] were the pioneers of this field. They suggested an iterative DL-based video reconstruction algorithm, by which the middle frame of the output video is recovered first. After it's recovery, the recovered frame is used to reconstruct its adjacent 2 sharp frames, which are used for the reconstruction of their counterparts in the following iteration, and so on. Purohit et al. [9] suggested a two-stage training strategy with a recurrent architecture for extracting an ordered spatio-temporal motion representation from a blurred image in an unsupervised manner. The first stage includes training an auto-encoder, trained to encode and decode N sharp frames to reconstruct those input frames to the best of its ability. In the second stage, the decoder is frozen, the encoder is removed, and a blurred-image encoder (BIE) feeds the decoder instead. By doing so, the BIE is trained to encode an output similar to that of the previously used encoder, but based on a single blurry image as input (instead of the entire N sharp frames).

2.2 Computational-Imaging Based Deblurring

In contrast to the blind deblurring in which the camera acquisition procedure is treated as given and unchangeable, computational imaging tries to manipulate the acquisition in

order to obtain more information regarding the scene blurring. While this approach usually results in a low-quality intermediate image, this image is encoded with clues that, combined with the correct post-processing, could result in a better output than a regularly acquired image. Several different computational-imaging based approaches were inspected over the last years. Raskar et al. [10] suggested to open and close the cameras shutter during the exposure time of the acquisition according to a binary pseudo-random sequence. While the uniformly open shutter corresponds to a blurring kernel that is essentially a box filter (which has a very narrow frequency response), the fluttered-shutter technique is equivalent to a broad-band filter. Hence, it manages to capture details of high frequency, which are crucial for better motion deblurring. Levin et al. [**Levin˙SIGGRAPH2008**] tackled the issue of objects that move in varying velocities by dynamically moving the camera at the time of the acquisition following a parabolic trajectory. Based on their theory, the parabolic profile is optimal in the sense that for every element that moves in the scene, there is at least a single moment in which the element's speed and the sensor's speed are equal. This results in all objects having approximately the same point-spread function (PSF), making the deblurring easy to invert by deconvolution, without having to segment each object and estimate it's corresponding velocity. Elmalem et al. [1] introduced a method for generating a more informative intermediate encoding of the captured scene by dynamically changing the lens phase-coding during the image acquisition. Their method manages to capture both the direction and magnitude of the acquired trajectory by coloring the spatial blur of each element in the scene. They further trained a CNN fed with the intermediate encoding to provide a complete deblurring solution.

3 Data

As mentioned in the introduction, I constructed a dataset that is based on the novel **RE**alistic and **D**ynamic **S**cenes [7] dataset. The REDS dataset was introduced in 2019 by Nah et al. for video deblurring and super-resolution. The REDS dataset consists of 300 video sequences, each made of 100 sequential frames in 720×1280 resolution. All sequences were constructed based on manually recorded videos using GoPro HERO6 Black in 1080x1920 resolution at 120fps. To generate the input sequences, the raw videos were first upsampled to 1920fps, so that the averaging will exhibit smooth and realistic blurs without spikes and step artifacts. The eventual sequences are thus obtained by averaging in the signal-space and downscaling the spatial resolution to 720x1280 , and the temporal resolution to 24fps.

While the REDS could have been used almost "as is" for the purpose of single image deblurring, it wouldn't be fit for the task of video from motion, as each blurry input image needs to be paired with several sharp target images. Moreover, the reference paper I implemented used 7 consequent sharp frames to generate a single blurry image, whereas the REDS post-processed blurred images were produced based on 5 sharp frames only. Hence, instead of using the post-processed REDS dataset, I downloaded the raw images and manually generated the blurry images from their paired 7-sharp target sequences, following the same methodology suggested by Nah et al. An example for such blurry-sharp frames pair is depicted in figure 1.

In addition, I've written a code for automatically pulling the raw-images of the different sequences and organizing them in H5 files, as well as a compatible data-loader, for storage efficiency. Moreover, As some augmentations were needed to be applied to both the input

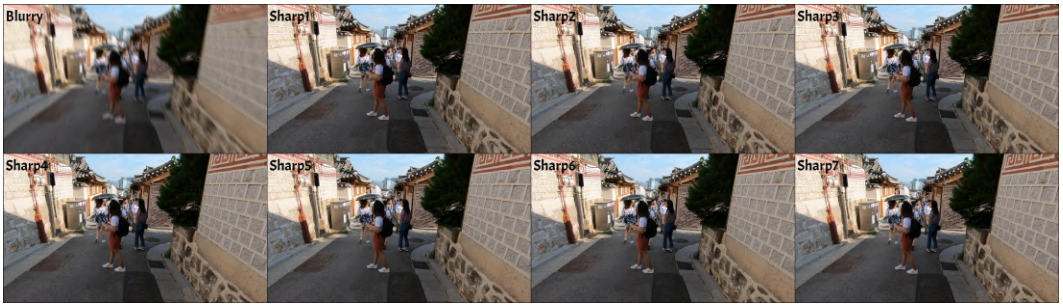

Figure (1) The manually generated dataset: the blurry image (upper left corner) was generated based on the average of the 7 raw-sharp frames.

and target frames (e.g cropping, rotation, etc), while others exclusively to the input (e.g additive noise) - the input and target frames were concatenated to a single tensor, jointly augmented by the common required augmentation, and then re-separated before applying the exclusive augmentations. An example for an augmented blurry input and sharp target frames appears in figure 2.

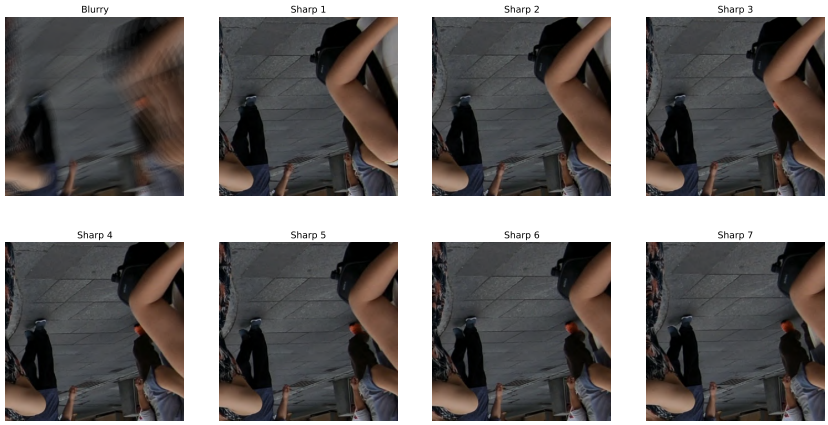

Figure (2) An exemplary augmented pair of blurry input and sharp target frames.

4 Methods

As mentioned in the introduction, my approach is to fuse a computational-imaging based deblurring technique into the evolving field of video from motion. The motivation behind this approach is that computational imaging techniques exhibited superior results for the case of a single frame deblurring, compared to blind-deblurring techniques. The ability to reconstruct a sharp frame has to do with the ability to estimate the dynamic motion that was captured. Hence, computational imaging could potentially generate superior videos

from motion based on single blurry frames.

4.1 Video-From-Motion Network

In order to fuse the presented fields, I first had to implement the backbone for the video-from motion generation. For that purpose, I chose to implement the pioneer paper in the field: *"Learning to Extract a Video Sequence from a Single Motion-Blurred Image"* by Jin et al. [3]. I chose to implement it because not only where there very few alternative papers (I found exactly 2 other relevant papers), it was the only one among them with partially-published open code (the networks models).

4.1.1 Architecture

In the original paper, 7 different networks were implemented and trained:

1. Middle-frame (#4) reconstruction network.
2. Middle-adjacent-frames (#3,#5) reconstruction network.
3. #2,#6 frames reconstruction networks.
4. #1,#7 frames reconstruction networks.

The networks were trained separately and sequentially, as the reconstructed frames of a trained network served as input to the following network in the enumeration (both for training and inference). The sharp frames reconstruction can be mathematically described using the following equations:

1. $\hat{x}_4 = \phi_4(y)$
2. $\hat{x}_3, \hat{x}_5 = \phi_3(\hat{x}_4, y),\ \phi_5(\hat{x}_3, \hat{x}_4, y)$
3. $\hat{x}_i, \hat{x}_{8-i} = \phi_i(\hat{x}_{i+1}, \hat{x}_{i+2}, y),\ \phi_{8-i}(\hat{x}_{7-i}, \hat{x}_{6-i}, y), i \in 1, 2$

where:

- y is the input blurry frame.
- $\hat{x}_i$ is the estimate for the i'th sharp frame.
- $\hat{\phi}_i$ is the network used to reconstruct the i'th sharp frame.

The middle-frame estimation network consists of 3 parts: feature extraction, feature refinement and feature fusion. First, the input blurry image is down-sampled by a factor of 4 in each color-channel, resulting in 16 different down-sampled images per color channel. The feature extraction is done by a 5x5 convolutional layer with 144 output channels. Then, as part of the feature refinements, 12 residual blocks are cascaded and used. To further increase the receptive field, dilated convolutions are applied to the middle 6 residual blocks. A general scheme of the network's architecture is presented in figure 3. A similar architecture was used for the non-middle frames estimation as well, with the main differences being the number of channels, the down-sampling factor, the feature extraction layers and the number of input images.

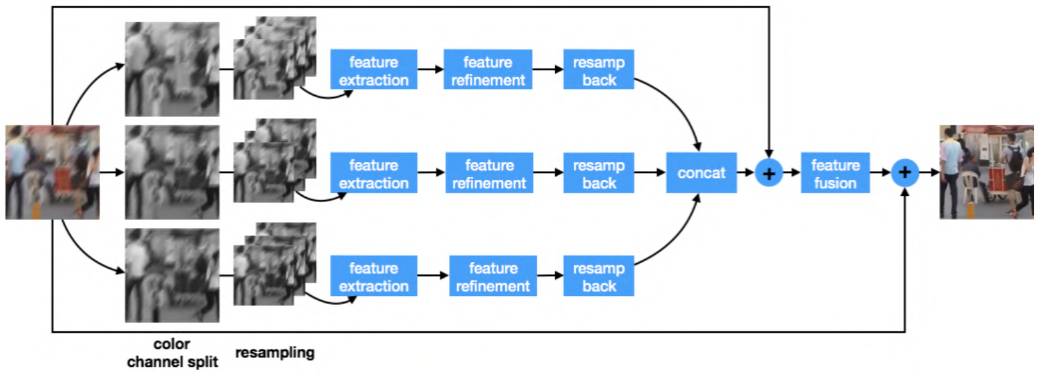

Figure (3) Jin et al.'s middle frame estimation network architecture.

4.1.2 Implementation

Due to the fact that the code for training was not available, and as training each network over the used dataset takes approximately 3 days, I decided to focus on implementing and performing the training for the middle-frame reconstruction network. In the experiments section, this trained network will be further combined with the other pre-trained non-middle frame estimation networks to estimate all 7 sharp frames. Moreover, as the hyper-parameters used for training the network were also unavailable, I took the liberty of exploring different possible configurations to achieve superior results. As part of the training routine, I also implemented the loss that was suggested in the paper:

$$\mathcal{L} = |x_m - \hat{x}_m|_2^2 + \lambda \cdot \mathcal{L}_{percept}(x_m, \hat{x}_m) \tag{1}$$

where:

- $|x_m - \hat{x}_m|_2^2$ is the standard L2 loss.
- $\mathcal{L}_{percept}(x_m, \hat{x}_m)$ is the perceptual loss based on Johnson et al.[4] paper. Specifically, *relu2_2* and *relu3_3* layers of the pre-trained vgg16 network were used as feature maps for the loss calculation.
- λ is a scaling factor that I chose to add to the implementation in order to test the effect of different weighting balance between the L2 and the perceptual loss over the eventual performance. In the paper, $\lambda = 1$.

In addition, and While I tried to remain faithful to the implementation described in the paper, I decided to make a few changes:

- In the paper, the authors downscaled the dataset images by 45% in both dimensions (prior to the augmentation), which is claimed to reduce the noise to some extent. However, when I applied this down-sampling to my dataset, it seemed to significantly compromise the quality of the data, as demonstrated in figure 4. Hence, I decided to keep the original images resolution.
- A batch size of 8 was used instead of 32. This choice was due to GPU resources limitations, as not enough memory was available for any batch size that is greater than 8.

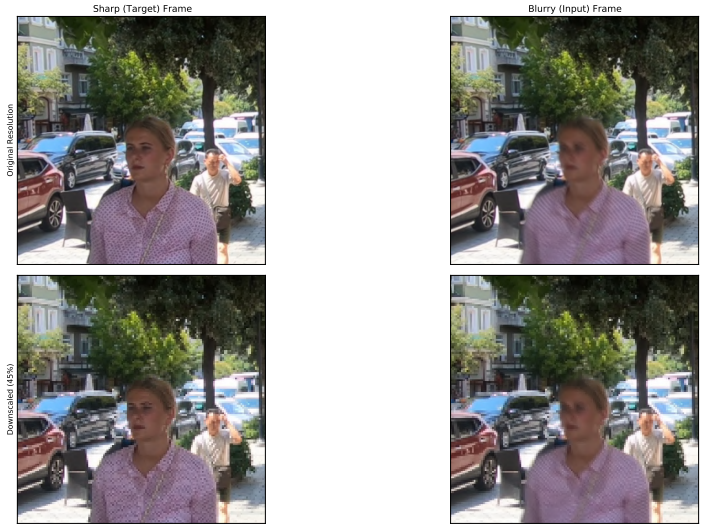

Figure (4) 45% down-scaling quality compromise: the left and right columns stand for the sharp (target) and blurry (input) frames correspondingly, whereas the upper and lower rows stand for the original resolution Vs the down-scaled versions.

4.2 Phase-Coding Acquisition Simulation

As discussed, a computational imaging technique was to be tested, combined with the video-from-motion architecture, to see whether superior results can be obtained. Specifically, I followed the approach of Elmalem et al. [1], that used a dynamically changing lens phase-coding during the image acquisition. The idea behind this approach is that the resulting image's chromatic aberrations imply both the direction and magnitude of the captured dynamics of the scene, as depicted in figure 5.

Figure (5) Motion blurred PSF simulation: conventional camera (left), gradual focus variation in a conventional camera (middle), and the proposed camera gradual focus variation with phase aperture coding (right).

Unfortunately, one can't simultaneously acquire a "regular" sharp-frame rate scene and it's blurry phase-coded equivalent, as the lens and acquisition characteristics are different. Therefore, a simulation must be used in order to obtain the blurry phase-coded equivalent based on the sharp regularly acquired frames. When applying the phase-coding technique, a blurry image is the integration of a continuous dynamic scene, with a continuously changing PSF, over the exposure time interval. To simulate it, multiple sharp adjacent frames can be convoluted with a discrete set of corresponding PSFs (matching the phase coding at the time of that frame's acquisition with respect to the blurry image's exposure time), and

summing the responses of all frames, to obtain an approximation of the described continuous integration. As described in the paper, the intermediate result of the procedure (both in real world and in simulation) is of inferior quality (compared to a regular blurry image). However, the phase-coding embeds cues in the intermediate result that could be taken advantage of by the video-from-motion network to eventually produce superior results. An example summarizing this procedure can be found in figure 6

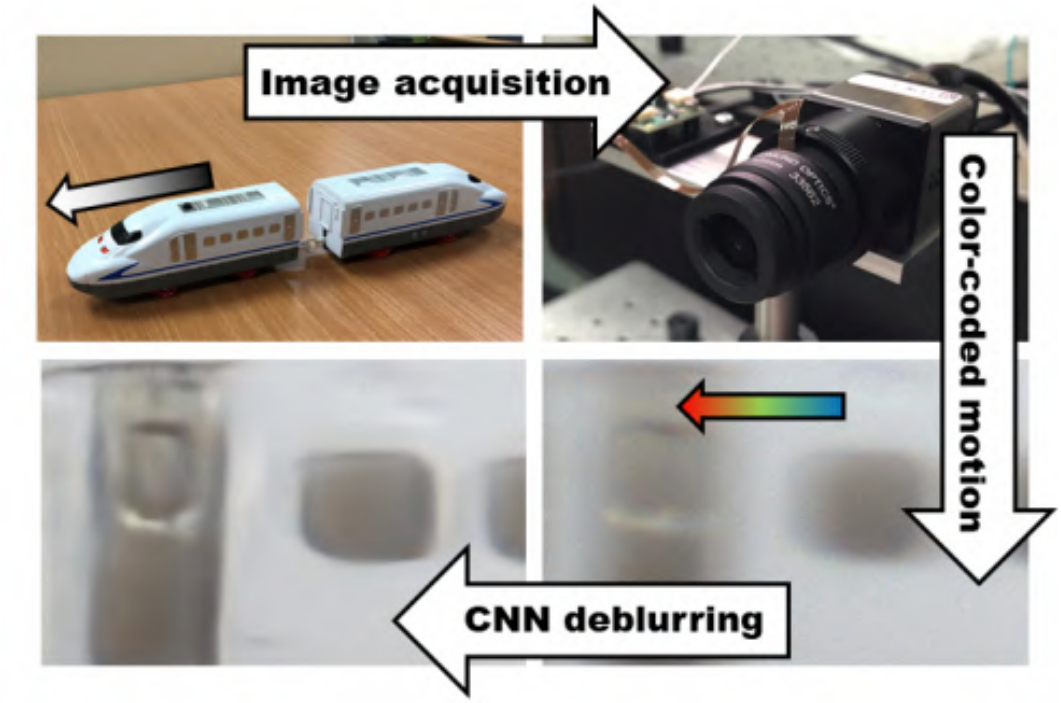

Figure (6) An example of the full processing pipe of the phase-mask based acquisition.

4.3 Results Evaluation

To make a fair comparison, the same architecture and set of hyper-parameters were used to train and test the video-from-motion network both with the original dataset and the phase-coded dataset. Other than a qualitative impression of the reconstructed videos, a quantitative comparison was also conducted using the following criteria:

4.3.1 PSNR

PSNR is the ratio between the maximum potential of the signal's power, and the power of the corrupting noise (or, in our case, restoration error). Mathematicly, PSNR can be formulated by:

$$PSNR = 10 \cdot log_{10}(\frac{max(I)^2}{MSE}) = 20 \cdot log_{10}(max(I)) - 10 \cdot log_{10}(MSE)$$

where:

- I is the target image. Typically, $max(I)$ is considered to be the maximum possible value an element in I can hold (e.g. 255 in case of UInt8 representation).
- MSE is the mean-squared error between the target image, and the reconstructed image.

Since $max(I)$ is typically constant, the PSNR is controlled by the MSE of the image restoration.

4.3.2 SSIM

SSIM is a method for evaluating the perceived quality of images, by comparing the tested image to a corresponding reference image. In the regular case, the reference is the original high-resolution image, and the tested image is the estimation of the original image based on it's distorted (in our case, blurred) version. In contrast to PSNR, which treats all pixels of the image independently, SSIM encapsulates the assumption of inter-dependence between pixels, which grows bigger the closer the pixels are to each-other in the spatial dimensions. These dependencies reflect structural attributes of the objects in the scene, which are ignored by the PSNR metric.
Given two pixel windows x,y of common size:

$$SSIM = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

where:

- μ_x is the mean of x
- μ_y is the mean of y
- σ_x^2 is the variance of x
- σ_y^2 is the variance of y
- σ_{xy} is the covariance of x and y
- $c_i = (k_i \cdot L)^2, i \in 1, 2$, where:
  - k_i are variables designed to provide numerical stabilization for the division in case of a weak denominator (typically, $k_1 = 0.01$, $k_2 = 0.03$)
  - L is the dynamic range of the pixel values (e.g 255 in case of UInt8 representation)

5 Experiments

For clarity reasons, I will follow the next terminology from this point forward:

- **Input**: The simulated blurry input.
- **Pre-trained**: The pre-trained network from Jin et al.'s paper, taken from Github.
- **Re-trained (Standard)**: After re-training with the standard-blur simulated dataset.
- **Re-trained (Masked)**: After re-training with the phase-masked simulated dataset.

5.1 Baseline Generation

As discussed in the Methods section, the first challenge was to establish a successful baseline training routine for the video-from-motion network, to reproduce the performance of the pre-trained network. As an initial step, I took the hyper-parameters from the last checkpoint of the pre-trained model, taken from the paper's Github project:

- Number of Epocs: 74
- Learning rate: 10^{-5}
- Optimizer: Adam
- Betas: (0.5, 0.999)

To gain some intuition regarding the required parameters tweaking, I ran a full validation epoch with the pre-trained network's weights, and compared it to my standard re-trained newtork's performance over the same validation set, as depicted in table 2.

Based on the comparison, 2 clear conclusions could be drawn:

1. The learning rate is too small (assuming it's stationary for the entire duration of the learning procedure).
2. The ratio between the average perceptual loss and the L2 loss is smaller than for the pre-trained network (173 compared to 230). This indicates that the λ weighting factor needs to be reduced such that the weight of the L2 loss in the total loss is amplified.

After some trial and error, I found that setting $\lambda = \frac{3}{4}$ and $lr = 10^{-4}$ (10 times the originally tested learning-rate) yields a satisfying result in terms of proximity to the pre-trained network's performance. The results of this optimized configuration on the validation set also appear in table 2. In addition to the improved results, the optimized training converges much quicker than the initially tested network (40 epochs Vs 70 epochs), which is equivalent to a 1.5 days difference. A convergence graph for the optimal training can be seen in figure 7. One can notice that the validation loss is uniformly smaller than the training loss. This happens due to batch normalization, which dynamically changes the estimate of the mean and variance during training, but uses the optimal estimations at evaluation time (validation), yielding better results.

Training Attributes	L2	Perceptual
Reference	0.0035	0.806
$\lambda = 1, lr = 10^-5$ (initial)	0.0066	1.144
$\lambda = \frac{3}{4}, lr = 10^-4$ (optimized)	0.0037	0.729

Table (1) Comparison of the average losses of the different experimental training parameters, versus the losses obtained by the pre-trained model taken from github.

To further validate the optimized training, an inference was done by both the pre-trained network and the optimized re-trained network to reconstruct the middle sharp frame using an exemplary (real) blurry input frame, as depicted in figure 8. Qualitative-wise, the differences between the solutions is clear: the pre-trained network results in sharper edges and higher contrast, which is an advantage when reconstructing textual information, e.g. the brand

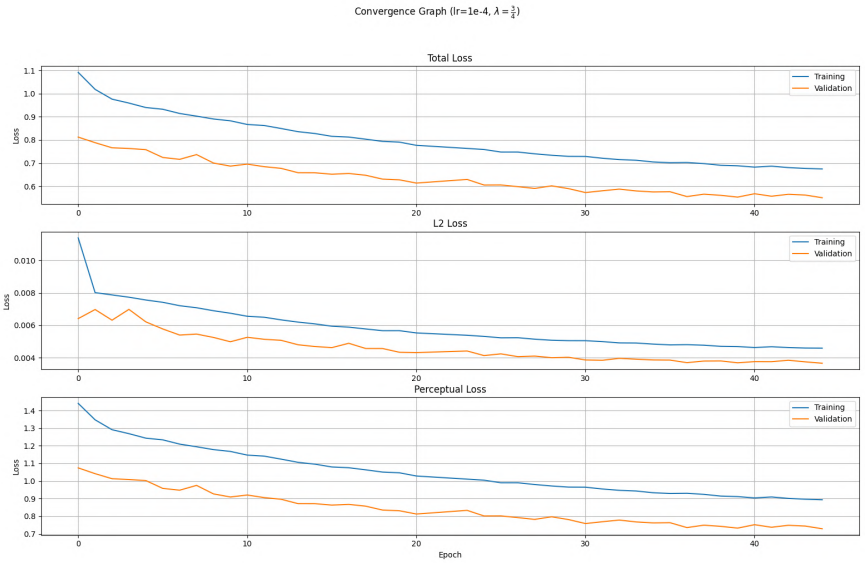

Figure (7) Convergence graphs for optimal training configurations

name on the ball in figure 8. Nevertheless, it appears that the high contrast is somewhat excessive, and sometimes results in artifacts, such as the spurious edge of the lower-finger holding the ball, which appears more natural in the optimally re-trained solution.

To conclude this part, while differing in performance from the pre-trained network, my standard re-trained solution provides satisfying results (at least qualitatively), and is even superior in some aspects.

A quantitative comparison, as well as a more thorough qualitative analysis, will be performed in the next subsection.

5.2 Phase-Masked Data Contribution

After having established a satisfactory reference solution and training procedure, it was time to put the spatio-temporal phase mask contribution to the test. The first step towards testing the phase-mask was to simulate it's impact on the acquired images. To do so, I used the same raw frames as in the standard blur-based training , but instead of simply averaging them, I depth-wise convoluted them with PSFs corresponding to the instantaneous phase masks that would have been used to acquire them, if they were truly acquired by a dynamically changing phase-mask. This procedure could be described using the formula:

$$y = g(\frac{1}{T}\int_0^T \tilde{x}(t) * m(t)dt) \approx g(\frac{1}{N}\sum_{i=1}^{N} x[i] * m[i])$$

where:

- y: Blurry frame
- g: Camera's CRF

(a) (b) (c)

Figure (8) Qualitative comparison between the pre-trained solution and my optimized re-trained network outputs, where (a) is the real-world blurry input image, (b) is the pre-trained network's middle frame estimation, and (c) is my standard re-trained network's estimation. Both networks exhibit different pros and cons.

- T: Exposure time (blurry frame).
- $\tilde{x}(t)$: Instantaneous irradiance.
- $x[i]$: Sharp frame (captured at $\frac{N}{T}$ FPS).
- $m[i]$: Phase mask PSF during the i'th frame acquisition.

Similarly to the standard-blur simulation, seven subsequent convoluted frames were used to generate a single blurry frame.

I order to make the final comparison "fair", I used the same training routine and set of hyper-parameters as the ones used to train the optimized network in the standard-blur case.

5.3 Final Results

5.3.1 Quantitative

To quantitatively compare and inspect the obtained solutions, I used the standard image quality metrics PSNR and SSIM, as described in the *methods* section. The average results obtained over the entire test-set appear in table 2.

Network	PSNR	SSIM
Pre-Trained	24.946	0.749
Re-trained (Standard)	24.602	0.72
Re-trained (Masked)	24.168	0.691

Table (2) Average PSNR and SSIM scores over the test-set.

Surprisingly, and while the obtained loss over the validation set seemed similar for all compared networks, the re-trained networks exhibit diminished PSNR and SSIM compared to the pre-trained network. While not expected and dis-satisfying, it is possible that the image quality metrics fail to capture some significant attributes of the obtained solutions. Moreover, as the deblurring problem is ill-posed, the notion of "target" used in these metrics is not well defined, further questioning the meaning of these results.

5.3.2 Qualitative

To qualitatively compare the different solutions, several inferences were performed using the different networks over random blurry images, taken from the test-sets. The inferences were inspected both from the middle-frame reconstruction aspect and from the full-video reconstruction aspect. After analysing several outputs, the following observations could be made:

- In most cases, in contrary to the quantitative results, both re-trained networks achieved superior deblurring results.
- In several cases, the phase-masked based blur re-training outperforms the standard based blur re-training. Furthermore, the superiority is not always uniform over the entire picture, but rather in specific regions of interest (ROIs). More concretely, it seems that the phase-masked based network is better at reconstructing facial features and keeping them temporally consistent along the video frames.

Examples 9,10,11,12 demonstrate the described observations in the middle frame reconstruction. The same observations also apply to full-video reconstructions, which are demonstrated in gif files available in the project's Github repository, under *Examples*. However, some inputs result in diminished output quality compared to the pre-trained network. In most cases, the failure manifests in delusional color-mosaic patterns, such as in examples 13,14. These failures could perhaps explain the diminished quantitative results, as these chromatic aberrations can result in significant differences between the target and the estimate, heavily penalizing the PSNR and SSIM metrics.

Full Image Cropped ROI

Figure (9) Facial features superiority: The phase-masked based network outperforms all others in reconstructing facial features, as clearly seen in the cropped ROI. Both the baby's face and that of the woman facing the camera are more consistent than in the other solutions.

Full Image Cropped ROI

Figure (10) Tiger stripes: the tiger stripes are much sharper and more temporally consistent in the masked-based trained case.

Full Image Cropped ROI

Figure (11) Carpet texture: the carpet texture appears more authentic in the masked-based network's output. Nonetheless, the grass texture is in fact better reconstructed by the standard-based network.

Full Image Cropped ROI

Figure (12) Bricks and walls: both bricks and walls edges appear sharper and clearer in the maked-based solution than in the others. The roof texture on the other hand is the most consistent for the standard-based network.

6 Conclusions

Based on the experiments, the application of the computational-imaging technique of temporal color-coding in the image acquisition is beneficial to the procedure of reconstructing a video from the intermediate result. While not manifested so in the quantitative results (and not uniformly qualitative-wise), there is no doubt that this technique holds great potential.

As the time and effort resources that could be dedicated to the project are limited, not

Full Image Cropped ROI

Figure (13) Color mosaic 1: An delusional color-mosaic pattern is produced by the masked network on the floor, near the white shoes. In addition, the masked network also fails to reproduce a sharp representation of the white shoe's lines.

Full Image Cropped ROI

Figure (14) Color mosaic 2: An delusional color-mosaic pattern is produced by the masked network on the back of the blue shirt.

all directions and ideas could be pursued. Nevertheless, here are some possible leads to follow for a follow-up activity:

1. Non-middle frame networks' re-training: as explained, and due to the lack of available code and lack of time, I devoted my efforts to re-training the middle-frame reconstruction network only. The pre-trained weights of the networks used for reconstructing the "outer" frames were taken directly from the original paper by Jin et al.[3], without any adaptation to the new dataset, in which the mask-phase contribution is manifested. Surely, re-training these networks using the cues coded by the mask phase in the newly designed dataset will produce even superior results to those displayed in this project.

2. Superior blur simulation: in the paper by Nah et al. [7], a special technique is used for generating the simulated blurry frames. Their technique is based on an up-sampling from 120FPS to 1920FPS, performed by a CNN (whose implementation was not available in the paper), prior to the averaging. Moreover, before the up-sampling phase, the writers used an estimate of the camera's CRF (also, not available in the paper), and it's inverse to reproduce the scenes in the signal space. The CRF was used again over the simulated blurry scene to re-transform the result to the image space. As these inputs were not available to me, I could not perform any of the described processes, and had to settle for simple averaging (or depth-wise convolution in the phase-mask case). Should these inputs (up-sampling CNN, camera's CRF) be obtained or reproduced, the re-training procedure could result in much better outputs.

3. Architecture: Again, due to time-constraints, I had no choice but to take the architecture from Jin et al.'s paper, as it was the only published architecture available for reconstructing a video from a single frame. While adequate for blind-deblurring, it might very well be that this architecture does not take full advantage of the color-coding that is introduced by the mask phase during the image acquisition. Further analysis and research could perhaps identify better architectures to take advantage of the phase-mask's contribution.

7 Appendix

All written code, as well instructions for training and inference and examples are available in the project's Github. As mentioned through the report, I manually implemented the following from scratch:

- Datasets and dataloaders.
- Training routine.
- Testing and Inferencing of sharp frames and videos.

For simplicity - all code that is outside the *Ref* directory (which is a sub-module that references Jin et al.'s repository) was written by me.

References

[1] Shay Elmalem, Raja Giryes, and E. Marom. "Motion Deblurring using Spatiotemporal Phase Aperture Coding". In: *ArXiv* abs/2002.07483 (2020).

[2] Ankit Gupta et al. "Single image deblurring using motion density functions". In: *European conference on computer vision*. Springer. 2010, pp. 171–184.

[3] 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, pp. 6334–6342.

[4] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *European Conference on Computer Vision*. 2016.

[5] Anat Levin. "Blind motion deblurring using image statistics". In: *Advances in Neural Information Processing Systems* 19 (2006), pp. 841–848.

[6] Seungjun Nah, T. Kim, and Kyoung Mu Lee. "Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 257–265.

[7] Seungjun Nah et al. "NTIRE 2019 Challenge on Video Deblurring: Methods and Results". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2019.

[8] Liad Pollak Zuckerman et al. "Across Scales Across Dimensions: Temporal Super-Resolution using Deep Internal Learning". In: *European Conference on Computer Vision (ECCV)*. Springer. 2020.

[9] Kuldeep Purohit, Anshul Shah, and A. N. Rajagopalan. "Bringing Alive Blurred Moments". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[10] Ramesh Raskar, Amit Agrawal, and Jack Tumblin. "Coded exposure photography: motion deblurring using fluttered shutter". In: *ACM SIGGRAPH 2006 Papers*. 2006, pp. 795–804.

[11] Shuochen Su et al. "Deep video deblurring for hand-held cameras". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1279–1288.

[12] Jian Sun et al. "Learning a convolutional neural network for non-uniform motion blur removal". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 769–777.

[13] Oliver Whyte et al. "Non-uniform Deblurring for Shaken Images". In: *International Journal of Computer Vision* 98 (2010), pp. 168–186.