THE UNIVERSITY OF CHICAGO

UTILIZING BOTH PAST AND FUTURE: MULTI-FRAME MEMORY BASED NETWORK IN SOLVING PARTICLE IMAGE VELOCIMETRY

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCE
IN CANDIDACY FOR THE DEGREE OF
MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

BY

ZHUOKAI ZHAO

CHICAGO, ILLINOIS
NOVEMBER 4, 2021

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to first thank my thesis advisor, Professor Gordon Kindlmann, from the Department of Computer Science, for his dedicated, and continuing help throughout the entire process of writing this thesis. It would be impossible for this work to be finished without him.

I would also like to thank Professor Michael Maire, from the Department of Computer Science, for providing many inspirations on the network design and model performance analysis. Many of his questions during our discussions have shaped this work much more well-rounded and complete.

Last but not least, I would like to thank Professor William Irvine, from the Department of Physics, for providing guidance from the standpoint as a physicist. His insights on both physics and computer science truly help this work become interdisciplinary, and meaningful to both fields.

ABSTRACT

Research in experimental fluid dynamics requires methods for recovering velocity vector fields from within fluid flow experiments. A widely used method, Particle Image Velocimetry (PIV), analyzes video images of fluorescent particles moving in fluids. Various computational approaches have been applied to PIV, such as traditional cross-correlation, variational analysis, and most recently, machine learning (ML). We describe here a novel ML approach to PIV based on deep learning, with the goal of more accurately and efficiently estimating the dense 2D velocity field for each frame of a PIV video. Our approach is distinguished by how it flexibly uses multiple frames earlier and later in time, rather than only pairs of frames. We show that on a variety of images and flow fields, our deep learning PIV approach is competitive with other state-of-the-art methods. We also describe a software tool for synthesizing PIV images from known velocity fields for ML training, which will benefit future research on ML-based PIV.

CHAPTER 1

INTRODUCTION

With the development of machine learning, many research areas have been transformed from using classical methods which require a large amount of domain expertise and careful engineering, to learning systems that learn from vector representations of the raw data [16]. Such innovations have motivated novel research in many areas, ranging from face recognition in security systems [29], to product recommendations on e-commerce websites [5]. However, it has been less actively explored within scientific applications, such as Particle Image Velocimetry (PIV), which is a non-intrusive velocity measurement tool that extracts velocity fields from sequential particle images.

PIV is an important tool that physicists use to understand experimental (as opposed to computer-simulated) results in fluid dynamics. Complex flow phenomena, such as turbulence and vortex interactions, continue to present interesting research questions, which may be answered in part with improvements in the accuracy and efficiency of PIV algorithms. A typical PIV experiment consists of three steps. The first step introduces small particles that do not interfere with the flow motions into the flow medium. These particles are then illuminated by a laser sheet and captured by a camera, which generates successive particle images as time goes. The third step is to analyse these particle images and recover the underlying velocities from particle movements [24]. Our work here focuses on the computational aspects of the third step.

The contributions of this work can be summarized as follows.

1. We design a novel deep learning framework, named **Memory-PIVnet**, that utilizes a larger time window instead of being restricted to image pairs. The network uses a memory-based backbone, which consists of both Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to extract the features both spatially and temporally, and then use these features to predict the flow motions in a coarse-to-

fine fashion. Experiments show that the results of our model achieves state-of-the-art performance in solving PIV.

2. We develop a software tool that generates high-quality, long-sequence particle images from existing velocity fields, in contrast to existing work where image pairs is the only format. Besides providing training datasets for our model, it could be a very useful tool for future researchers to develop innovative PIV algorithms that does not only use image pairs.

The structure of this paper is as follows. Chapter 2 discusses previous work that has been dedicated to improve PIV. Chapter 3 explains in details how we designed our novel deep learning network, Memory-PIVnet, that uses multiple frames earlier and later in time, instead of only pairs of frames; Chapter 3 also introduces the software tool which generates synthetic particle images from existing velocity fields; Chapter 4 shows that our method achieves superior results when evaluated on synthetic datasets, compared to existing state-of-the-art PIV algorithms; Chapter 5 discusses ablation studies regarding several design choices that we made for our network, and how different designs impact our model's performance; Finally, Chapter 6 concludes by discussing various directions of future work.

CHAPTER 2

RELATED WORK

Although PIV originated in physics, it has long utilized techniques from image processing and computer vision. Methods in two main categories, cross-correlation (CC) [30] and variational optical flow [6], comprise a large portion of classical PIV solvers. In general, the CC-based methods calculate the cross-correlation between two interrogation windows from image pairs, and use the maximum correlation to compute the particle displacement vectors. These methods are efficient and easy to implement. However, they can only output velocity fields that have much lower resolution compared to the input images, and requires proper postprocessing to remove outliers [31], which makes it insufficient for today's studies that require a dense velocity field, at or near the resolution of the input image.

On the other hand, variational optical flow methods, studied intensively by the computer vision community, transform the problem into a optimization problem [7]. Their success depends on minimizing a well-chosen objective function. These approaches, unlike the CC-based methods, can provide dense field outputs. However, the process is usually computationally intensive. A more detailed comparison between cross-correlation and variational optical approaches can be found in [19].

Beyond the classical approaches, newer PIV methods have incorporated machine learning, based on the success deep learning has had with solving optical flow problems, which are similar to PIV. For example, Fischer et al. designed a network structure, FlowNet, which is capable of solving optical flow estimation as a supervised learning task [4]. FlowNet2, which is a cascaded version of the original FlowNet, was soon introduced to achieve higher accuracy with the cost of larger computation time [12]. Sun et al. later present PWC-Net that uses pyramidal processing, warping and cost volume to achieve a better performance while keeping the model 17 times smaller in size than the FlowNet2 [26]. Hui et al. introduced a sequence of work, namely LiteFlowNet [10], LiteFlowNet2 [9] and LiteFlowNet3 [8], which outperform FlowNet2 while keeping the computation time low. Besides these, other approaches which utilize spatial-channel attention maps [33] and recurrent neural networks [28] also provide comparative or better results. Although these methods have achieved state-of-the-art performance in estimating optical flow, they cannot be directly used to solve PIV and achieve comparable performance, because particle images have very different characteristics compared to typical video images. All PIV images contain a dense field of highly visually similar particles, which poses challenges for the optical flow methods mentioned above to efficiently find corresponding object pairs between adjacent images. PIV also seeks a higher level of spatial resolution on the output velocity field that is required of optical flow analysis. Nevertheless, the relative success of these approaches have encouraged further exploration of deep learning for PIV.

The first proof-of-concept work to combine PIV with deep learning was in 2017, in which Rabault et al. built a model with Convolutional Neural Networks (CNNs) and fully connected neural networks (FCNNs) for performing end-to-end PIV [23]. Although the proposed method at that time did not outperform state-of-the-art CC-based methods, it opened up a new way for others to continue the study. Soon after that, Lee et al. [17] developed a convolutional PIV, named PIV-DCNN, based on a cascaded network structure that has been developed originally for facial recognition [27]. PIV-DCNN takes two particle images as a paired input and outputs a velocity field with improved spatial resolution and comparable accuracy to classical PIV approaches. However, the computational efficiency remains a weakness. Cai et al. [3][2] proposed two models, one is based on the FlowNet [12] while the other is based on the LiteFlowNet [10]. Cai et al. modified the original network with more deconvolution layers, unbalanced weighting coefficients and a normalization step on each level's estimated flow to make them better suited for PIV. Their approaches achieved similar accuracy compared to state-of-the-art methods with better computational efficiency. However, as these approaches are modified versions of existing network structures developed originally for optical flow, we believe a novel deep learning framework designed from scratch for PIV could further improve the results.

In addition to the supervised learning approaches mentioned above, other deep learning techniques have also been lightly explored. Zhang et al. proposed an unsupervised learning method which is also based on the LiteFlowNet structure [34]. They developed a loss function consist of a photometric loss between an image pair, a consistency loss in bidirectional flow estimates and a spatial smoothness loss. This unsupervised approach fills the gap between the training set-ups and real-world scenarios that most supervised methods suffer, but is left behind in terms of flow estimation accuracy.

All the above existing approaches recover velocity vectors only from pairs of images adjacent in time. However, since turbulent fields have the nature of being rapidly changing at some areas while being relatively quiet at other areas, using an image pair as the only input for flow estimation poses problem for a uniform level of performance across the entire spatial domain. Therefore, our idea of solving PIV originates from finding the best way to utilize multiple frames of particle images instead of merely two, to get better results in low velocity areas.

One previous method has been designed to address this non-uniform performance concern as part of Cross-Correlation analysis. Wieneke et al. proposed an adaptive PIV method by varying the size of the interrogation windows based on flow gradients [32]. Their method demonstrates improvements in both accuracy and spatial resolution, which inspires us to explore further in the path of adaptive algorithms. However, instead of adopting a similar strategy that, for example, chooses the best two frames adaptively out of a larger time window, we utilize a memory-based network that extracts the features in both temporal and spatial domain, so that the network can learn what would be the most important features to use when estimating the velocity field. Such memory network, compared to the adaptive algorithms that choose the best two frames, enjoys the benefit of having pixel-level freedom.

Integrating memory units with neural networks has been a long-term development in the history of deep learning. Among many approaches, multigrid neural memory [11] is especially interesting as it is capable of learning and processing data on both spatial and time scales, while keeping high computational efficiency. It also promotes implicit internal attention, which works with our intuition that different time frames should be responsible for flow estimations on different areas within the field of view.

Multigrid neural memory utilizes convolutional long short-term memory (ConvLSTM) [25] to process the data in a time-dependent way, which was designed initially to learn the long-term dependencies. In our case, series of particle images can be either a long or short time-dependent dataset. However, using such ConvLSTM-based memory structure is superior to using memory structures that are only capable of short-term memory, because it provides the capability to explore various lengths of time windows in the future. Existing experiments [21] also shows that long-term memory delivers better performance when trained and tested with shorter series.

CHAPTER 3
METHODS AND RESULTS

As mentioned in Chapter 2, our novel network structure, Memory-PIVnet, utilizes a multigrid neural network backbone to extract features from multiple frames of particle images. By manually setting a time window, and re-initializing the internal memory stage every length of the time window, the network can be trained as if it is on the short series of images. On the other hand, if the internal memory stage is never re-initialized, it is then equivalent to training on long series of images.

Memory-PIVnet allows us to flexibly explore the role of varying time windows and the benefit of memory in realistic PIV settings. It is reasonable to assume that the particle image at one time $t = 0$ will have no bearing on the velocities at a far later time $t = 100$ (as compared to the characteristic time scale of the flow). This suggests using small time windows via repeated memory re-initialization. But it is also reasonable to expect that a network without such memory re-initialization will, with sufficient training, learn by itself that the features from $t = 0$ should not be used for estimation at $t = 100$. In this work, we use term *amnesia* to indicate the type of memory network where its internal memory get re-initialized after a certain number of frames; and use term *non-amnesia* to represent the type of memory network where its internal memory never gets re-initialized. The performance of these two training schemes are explored and discussed in Section 5.2.

The remainder of Chapter is structured as follows. Section 3.1 illustrates the network structure of our Memory-PIVnet. More specifically, Section 3.1.1 shows the memory-based backbone structure, and Section 3.1.2 discusses the flow estimation network, which uses the extracted features to estimate the dense velocity field of each particle image.

Section 3.2 shows how the data is prepared and generated for training our model. The process utilizes the software tool that we developed, as mentioned earlier in Chapter 1. We break down and discuss each of its components in this section.

Considering that particle images can be huge in size and the physical GPU memory is limited, to make the model work with all sizes of images on normal GPUs, a tiling technique is developed and illustrated in Section 3.3. The technique consists of two parts, dividing the full image into smaller patches before feeding into the model, and stitching together the velocity fields resulting from each patch as final output. Concerns regarding eliminating the discontinuities that exist in the boundaries between individual tile velocity field is discussed in Section 3.3.1 and 3.3.2.

3.1 PIV Estimation with Memory-PIVnet

In this section, we introduce our novel deep learning model, Memory-PIVnet, that is designed to improve the PIV performance via neural networks. A high-level visualization of the network architecture is shown in Figure 3.1. The model architecture could be split into two sections; a memory-based backbone (top), and a coarse-to-fine flow estimation network (bottom), which are discussed in Subsection 3.1.1 and 3.1.2 respectively.

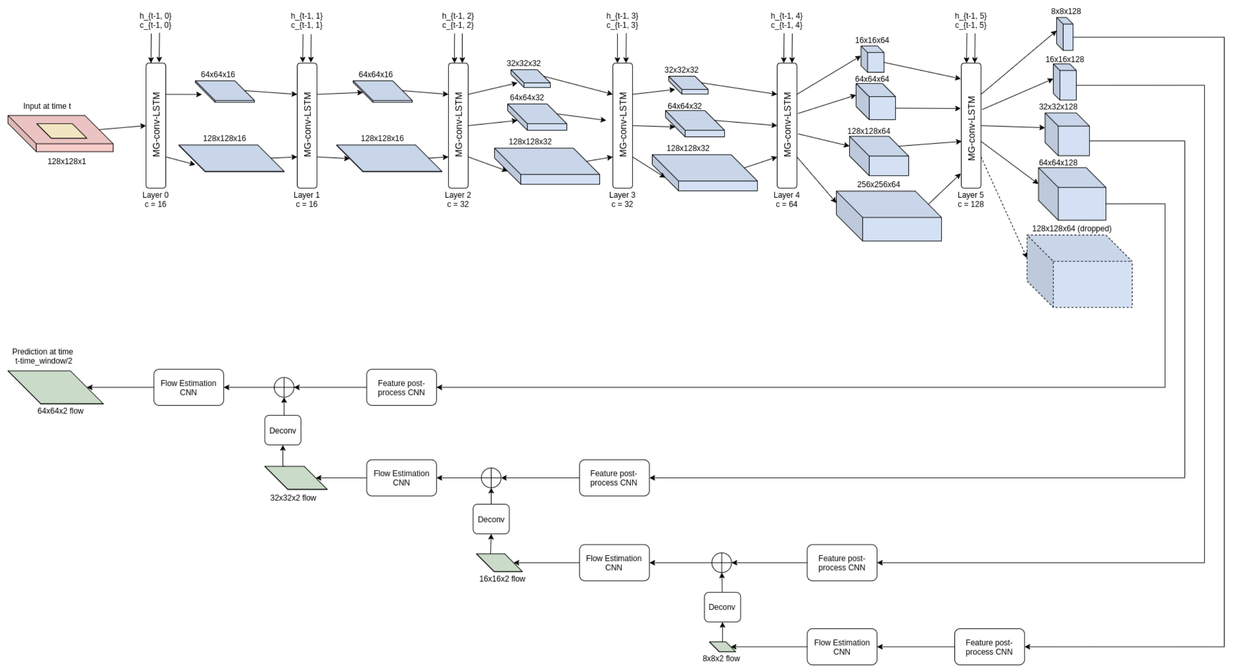


Figure 3.1: Memory-PIVnet network architecture; top section: memory-based backbone; bottom section: flow estimation network

3.1.1 Memory Network

Multigrid memory network consists of a series of multigrid memory layers, where each layer is crafted with multigrid convolutional layers [14] and convolutional long short-term memory (ConvLSTM) [25]. Such memory layers are abbreviated with name "MG-conv-LSTM" in our network structure diagram as shown in Figure 3.2. For detailed network structures inside the memory layer, we refer readers to [11] for more details.

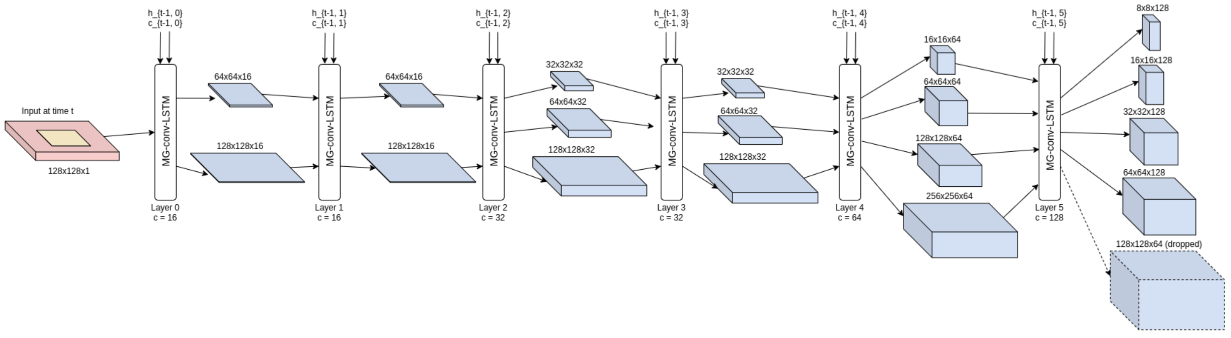


Figure 3.2: Memory network of Memory-PIVnet

Memory layers are able to process and mix data in both spatial and time domain. Each memory layer at time t and layer n takes previous time step, corresponding layer's outputs, $h_{t-1,n}$ and $c_{t-1,n}$, along with current time step, previous layer's outputs, $x_{t,n-1,k}$, where k is the level of inputs within one step (different spatial resolution inputs), to generate current time step, current-layer outputs, $x_{t,n,k}$. It determines, at each spatial resolution, what previous and current information to keep, just like the performance of normal LSTM, while blending features across all resolution levels. LSTM equips the network with ability to analyse and extract features from a chunk of data that spans across a larger time window, which serves the idea to design an adaptive PIV method that uses features extracted from different time steps to aid PIV estimation.

Suppose that we are estimating flow for frame x_{t_0}, using a time window with length T, where T is an odd number, the input data includes images $\{x_{t_s}, t_{t_s+1}, \ldots, x_{t_0}, \ldots, x_{t_e-1}, x_{t_e}\}$, where $t_s = t_0 - \lfloor \frac{T}{2} \rfloor$ and $t_e = t_0 + \lfloor \frac{T}{2} \rfloor$. The memory network processes this image block in a frame by frame fashion, and the final extracted feature is generated after processing the last frame x_{t_e}. The final features will then be used to estimate the flow for frame x_{t_0}. In other words, Memory-PIVnet performs flow estimation in a target-delay fashion, where the current estimation is for the image a few time steps ago. To support this training scheme, input data has been padded with $\lfloor \frac{T}{2} \rfloor$ frames to the front of the first frame and after the last frame. We choose to use the repeat of first frame for the front padding and the repeat of last frame for the end padding.

3.1.2 Flow estimation network

The flow estimation network predicts vector fields in an iterative coarse-to-fine manner, as shown in Figure 3.3. Extracted features with different resolutions, are first fed into a post-process CNN, which consists of one convolutional layer followed by a batch normalization [13] and a Leaky ReLU [20]. The post-process CNN helps further increase the number of channels that the extracted features have, which is cheaper than increasing the number of channels inside the memory network. Next, the outputs of the post-process CNN are fed into the flow estimation CNN, where its detailed structure is shown in Figure 3.4. For each level, except for the last (bottom) level, the estimated flow of the current level is up-sampled through a trainable deconvolution and padded with the next-level's (lower-level) outputs of its post-process CNN to generate the next-level estimated flow. As the extracted features coming from the memory network naturally have higher resolutions each level lower, they are designed to aid the flow estimation for a higher resolution, thus support the name of coarse-to-fine fashion.

3.2 Data generation

In this section, we demonstrate our data preparation pipeline that generates consecutive particle images from synthetic velocity fields in details. Synthetic velocity field is commonly

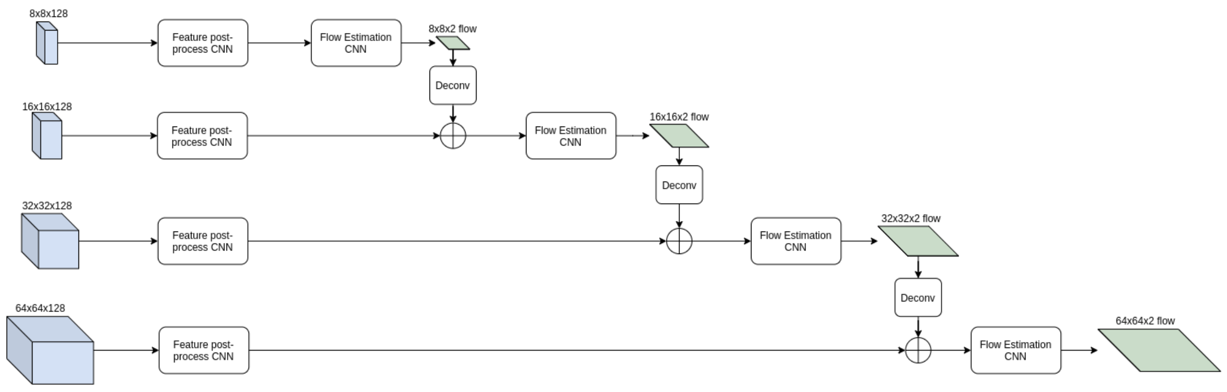


Figure 3.3: Flow estimation network of Memory-PIVnet

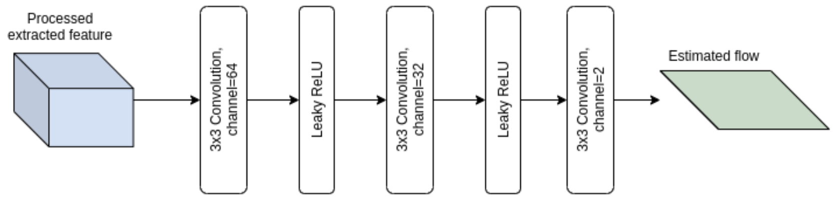

Processed extracted feature
3x3 Convolution, channel=64
Leaky ReLU
3x3 Convolution, channel=32
Leaky ReLU
3x3 Convolution, channel=2
Estimated flow

Figure 3.4: Flow estimation CNN detailed structure

simulated on a $x_{\text{range}} \times y_{\text{range}} \times z_{\text{range}}$ grid, where the velocities are stored every Δt_{data} seconds. Since velocity field can be stored at a different frequency than generating particle images, in our pipeline we use Δt_{image}, to distinguish this and represent the time gap between two consecutive particle images.

When introducing seeds into the synthetic flow fields, unlike existing PIV datasets [3] that consist of image pairs, extra strategies need to be applied to avoid the particles from gathering unevenly too much or exiting the field of view as time evolves for a longer period. Because these would make particle images contain too many blank or sparsely filled areas that are less useful for flow estimation. Here we propose a novel scheme where each seed's initial information is padded with a time step, which represents the initial time when this seed is introduced into the medium, in addition to the spatial locations. In other words, each point p is a four-vector $p = \{x, y, z, t\}$, where x, y and z are randomly initialized (drawn from a uniform distribution) from the spatial domain while t is from the time domain (length of the video). Each seed, although initialized at time t, is allowed travel in both directions of time until it gets out of the spatial or time range, whichever comes first.

Experimental particle images display randomness and noises on how particles are present on images. To mimic this, we generate synthetic images with the following cosmetic settings in mind.

- Particle radius

  The actual size of each particle differs from each other due to unavoidable manufacture imperfections. To account this, the radius of each particle in our synthetic particle images is drawn from a Gaussian distribution with $\mu = 1.5$ pixel, $\sigma = 0.5$ pixel, and truncated with finite support between 10^{-9} and 5.

- Particle luminosity

  Different seeds are illuminated differently as they are closer to or further away from the illuminated laser sheet. Seeds are assigned with different intensities based on their z positions compared to the z position of the illuminated slice, z_0. More specifically, we assign the peak luminosity using equation 3.1. We also set a spatial range in z direction as a visible bandwidth, which is set to be 60% of the total z range. Particles that are outside of this z range will not be shown in the particle images.

  $$\text{luminosity} = \exp \frac{-8 \times (z - z_0)^2}{L^2} \tag{3.1}$$

  where L represents the laser thickness and equals to 20 pixels by default

- Noise from physical process of acquisition

  A Gaussian blur with $\mu = 0$, $\sigma = r/2$, where r is the radius of the particle, is applied to each particle to simulate the noise existing in real world experiments.

Another place that our pipeline differs from the existing work is the measurement unit in point density. Prior work by Cai et al. uses points per pixel (PPP) to indicate the particle density, which is only meaningful when every frame has roughly the same number of particles [3]. This measurement works for image-pair datasets, where almost all points initialized in the first image will also be in the second image. Because the total time interval is very small, which makes it less likely that particles would drift out of the spatial ranges.

However, in our case, since the time interval is long, seeds might go out of the spatial range before reaching the end of the time interval. Therefore, we propose a new seed density measurement unit, points per frame (PPF), which equals to the total number of points N that are introduced throughout the entire time interval T divided by T, as shown in Equation 3.2. Notice that PPF does not indicate the expected number of particles that an image contains. Instead it provides a lower bound which means that at anytime, an image is expected to contain at least PPF number of particles.

$$\text{Points Per Frame (PPF)} = \frac{N}{T} \tag{3.2}$$

Throughout the experiments of this work, we used particle images with 10000, 50000, and 100000 total seed counts, which correspond to 39.68, 198.41, and 396.83 PPF respectively. Examples of the generated particle images are showed in Figure 3.5.

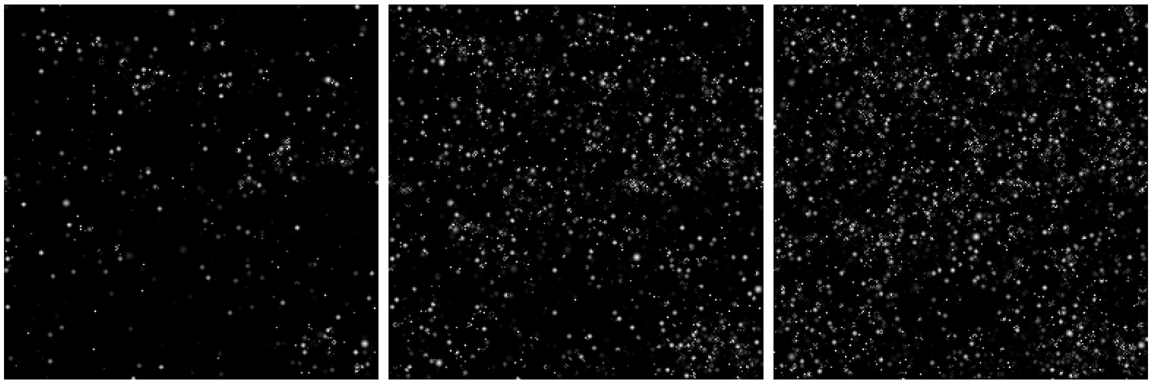


Figure 3.5: Particle images with point density = 39.68 (left), 198.41 (middle), and 396.83 (right) PPF respectively

3.3 Tiling and Boundary Elimination

Real-world particle images can be very large, sometimes as high-resolution as 4096×4096. Fitting these images without any tiling poses a huge challenge for current processing units, namely Graphics Processing Units (GPU). Although modern top-tier GPUs are having more video memory on a yearly basis, as well as both model and data parallelism are becoming easier thanks to machine learning libraries such as PyTorch [22], it is still beneficial, and often easier to develop a tiling strategy, that splits the original high resolution image into tiles, performs flow estimation on individual tiles, and stitches the results at the end. However, a huge downside of the tiling approach is that boundaries between tiles can produce discrepancies and hurt the continuity of the vector fields, which we call as boundary effect. To help eliminate such boundary effect, we propose two methods; margin padding and bi-linear interpolation blending. Margin padding utilizes the fact that memory layers blend data across different resolutions. It feeds the network with input tiles padded with surrounding information, while asking for estimation only on the center part of the input tiles. Bilinear interpolation, on the other hand, is a standard post-processing method that has been applied widely in image resembling.

3.3.1 Margin padding

By default, the size of the margin is set to match the size of the center area. In our experiments, original particle images have dimension 256×256, and are divided into 16 tiles per image, where each tile has size 64×64. The margins on each side have length 32, which bring the size of the padded tiles up to 128×128. Figure 3.1 shows the input-output relationship with margin padding; the input frame has spatial dimension being 128×128, while the resulting flow estimation has size 64×64, which matches the dimension of the center area of the input tile (center area is highlighted in yellow). The strategy, although making the training more time-consuming, is able to let the network get access to a broader area of the tiles when making flow predictions. Despite the largest-dimension features are dropped during flow estimation for the sake of computational efficiency, the neighboring information has been infused into lower-level features blocks. Because the memory layers blend higher spatial resolution information into lower level's, which is the key to why this margin padding strategy works. A visualization regarding the improvements that margin padding makes is showed in the middle of Figure 3.7.

3.3.2 Bilinear interpolation blending

Bilinear interpolation has been widely applied in image stitching and upsampling. It estimates a new pixel value by using the distance weighted average of the four nearest pixels. This idea is borrowed and slightly modified to eliminate the boundary effect as a post-processing aid to our final estimation results. As showed in Figure 3.6, suppose $ABCD$ is one tile, which consists of 4 sub-tiles 6, 7, 10 and 11, boundary effect happens at AB, AC, BD and CD. And there is no vector discontinuity inside the tile $ABCD$. Therefore, we divide the tile further into 4 sub-tiles, and perform bilinear interpolation blending on each sub-tile. We demonstrate our method based on sub-tile 6, but the same technique applies to all the sub-tiles.

To apply bilinear interpolation blending on each pixel of sub-tile 6, instead of using its geographical nearest pixels as used in the original version of bilinear interpolation, we obtain these "nearest" pixels through inference results from other tiles. Namely, tile $\{1, 2, 5, 6\}$, $\{2, 3, 6, 7\}$, $\{5, 6, 9, 10\}$ and $\{6, 7, 10, 11\}$, which are indicated in red, yellow, blue and green respectively in Figure 3.6. Inside sub-tile 6, the pixel closed to boundary AB should receive a blended value with higher weight from tile red and yellow, because no boundary effects happen inside tiles.

Therefore, let r and s be the horizontal and vertical axis as showed in Figure 3.6, the

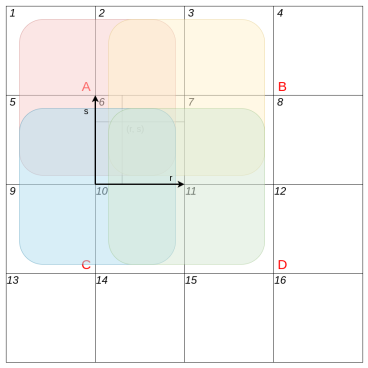


Figure 3.6: Bilinear interpolation blending in boundary removal

pixel value at location (r, s), where $r, s \in [0, 1]$ can be calculated as

$$I(r, s) = (1 - r)(1 - s)I_{\text{red}} + r(1 - s)I_{\text{yellow}} + (1 - r)sI_{\text{blue}} + rsI_{\text{green}} \tag{3.3}$$

where I_{red}, I_{yellow}, I_{blue} and I_{green} are the corresponding estimated pixel values from tile $\{1, 2, 5, 6\}$, $\{2, 3, 6, 7\}$, $\{5, 6, 9, 10\}$ and $\{6, 7, 10, 11\}$ at location (r, s). The resulting velocity fields are showed in the bottom of Figure 3.7.

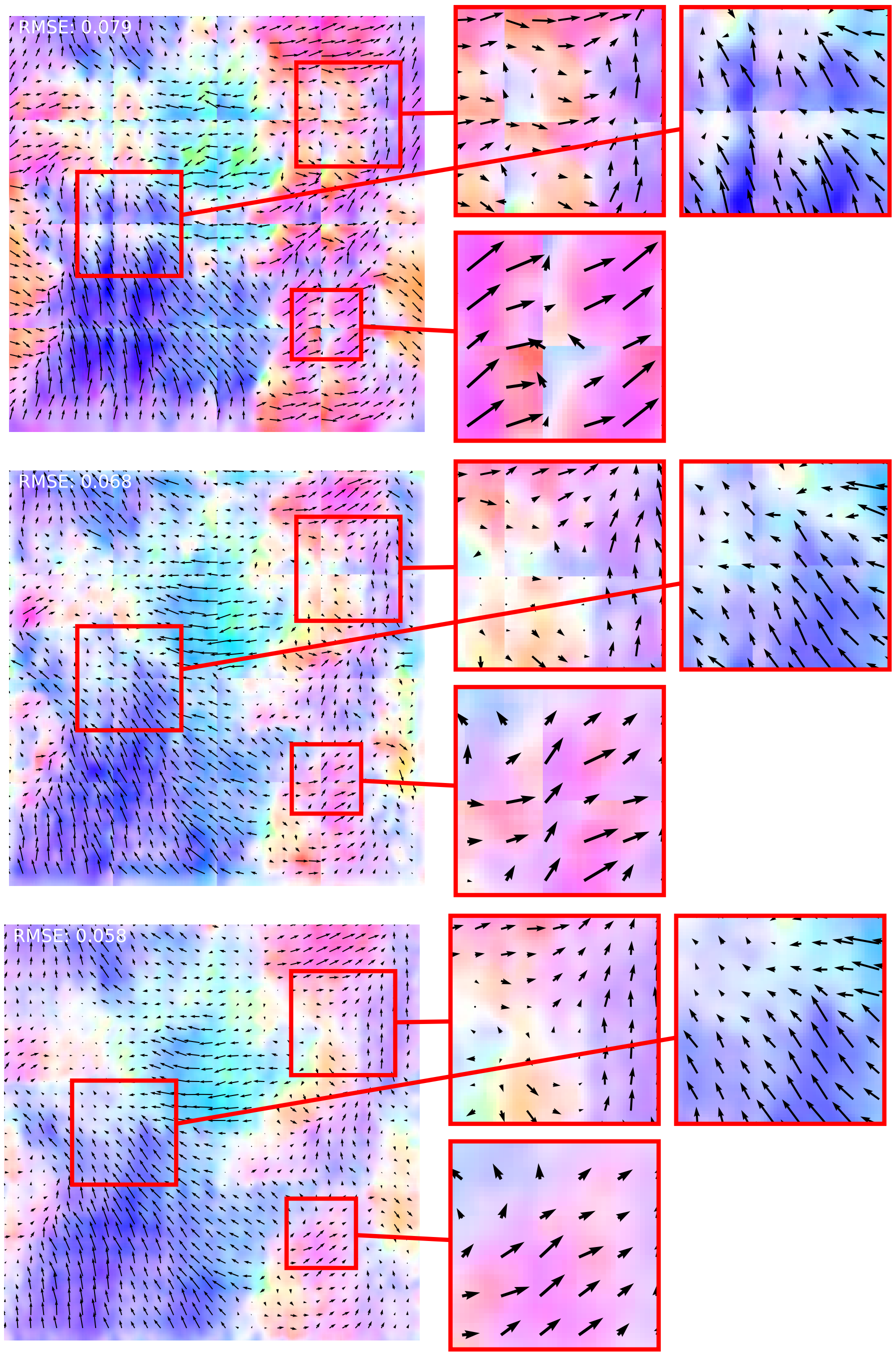


Figure 3.7: Boundary elimination; Top: boundary effects without margin padding or bilinear interpolation blending; Middle: boundaries are smoothed by margin padding; Bottom: boundaries are removed completely after both margin padding and bilinear interpolation blending.

CHAPTER 4
EVALUATION RESULTS ON SYNTHETIC DATASETS

We compare here our proposed Memory-PIVnet to recent learning-based approaches and to traditional variational optical flow. We use the Johns Hopkins Turbulence Database (JHTDB) [18] for our experiments. JHTDB provides numerous types of turbulence simulations using Direct Numerical Simulation (DNS), a state-of-the-art technique to simulate turbulent flow fields. We demonstrate the results on two distinct types of turbulence: forced isotropic and and forced magneto-hydrodynamic. The training and testing datasets consist of particle images and ground truths obtained from our software tool as discussed in Section 3.2.

4.1 Forced Isotropic Turbulence

This dataset records a simulation of forced isotropic turbulence on a 1024^3 grid with periodic boundary conditions, via numerical solution of incompressible Navier-Stokes equations [15]. Time integration of the viscous term is done analytically using integrating factor, where data are stored every 10 DNS steps, which equal to 0.002 seconds. In other words, $\Delta t_{\text{data}} = 0.002$, which results in a total of 5028 data frames for the entire simulation.

We divide the 1024^3 grid into $1024 \times 1024 \times 101$ chunks, down-sampled in x and y domain so that the dimension becomes $256 \times 256 \times 101$, and took the center 4 chunks. Particle images are generated every 20 data time-steps throughout the entire 5028 data frames, which means that adjacent particle images have a time-step $\Delta t_{\text{image}} = 0.04$ seconds. In total, 252 images are generated for each sequence.

When training the proposed Memory-PIVnet, if not otherwise mentioned, the time-window is set to 5 by default, which means that 5 images are padded along the time axis. The model predicts velocity field for the center frame. We use tiling to fit the model and data in normal consumer-level GPUs (ours with $\approx$ 10G of video memory), so each 256×256 image is divided into 16 64×64 tiles. Margin padding is applied as discussed in Section 3.3.1, where the neighboring pixels around the center 64×64 tile is padded to help eliminate the final boundary effect. The actual training and testing blocks then have dimension $128 \times 128 \times 5$. Among all the sequences, 3 sequences are used for training, and 1 is used for testing. In other words, 12096 blocks/samples are used for training while 4032 were used for testing. During testing, the tiles were stitched back from 16 64×64 tiles of predicted velocity fields to a single 256×256 velocity field. Then the result is blended as discussed in Section 3.3.2 to be the final result and used to calculate the error as well as shown in flow visualizations.

Consecutive particle images are converted to image pairs when training (if needed) and testing the existing pairwise-based methods, which include the previously mentioned PIV-LiteFlowNet-en [2], Un-LiteFlowNet [34], as well as the Horn-Schunck method (HS) [7]. The experiments include 3 sets of particle images that have been simulated with different seed densities. The average root mean square error (RMSE) of all the approaches are summarized in Table 4.1, while the individual RMSE of each testing sample is illustrated in Figure 4.1.

Methods / Density ρ	HS	PIV-LiteFlowNet-en	Un-LiteFlowNet	Memory-PIVnet
39.68	0.13044	0.10625	0.12408	**0.07298**
198.41	0.15877	0.10837	0.10145	**0.05635**
396.83	0.15779	0.10276	0.09675	**0.05314**

Table 4.1: Mean RMSE (unit: pixel) of different approaches on Forced Isotropic Turbulence particle images simulated with different seed densities (unit: ppf)

In addition, a set of visualizations of the ground truth velocity fields as well as the predicted fields are showed in Table 4.2. The 2D velocity fields are encoded in HSV colorwheel proposed by S. Baker et al. [1].

4.2 Forced Magneto-Hydrodynamic (MHD) Turbulence

This turbulence dataset is a numerical simulation of the incompressible MHD equations. Like the isotropic turbulence data in the previous section, it is simulated on a 1024^3 periodic

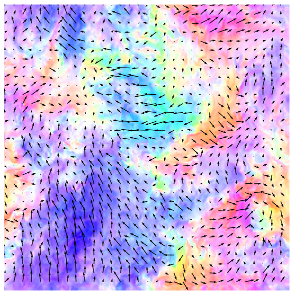
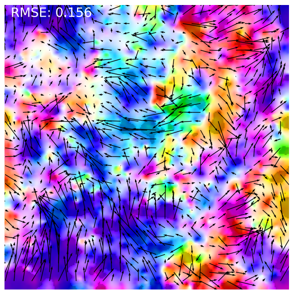
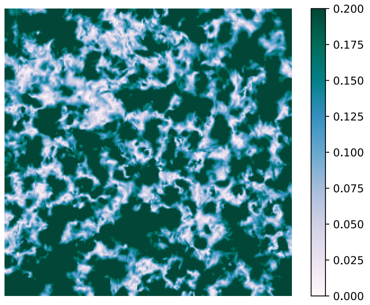
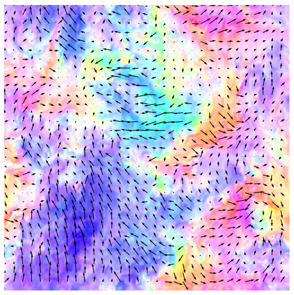
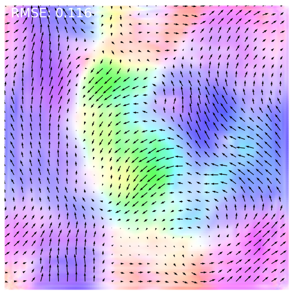
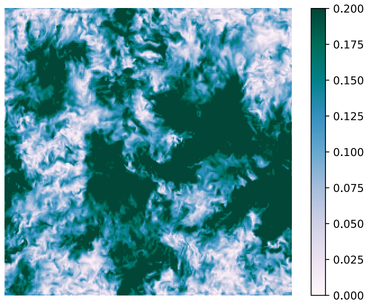
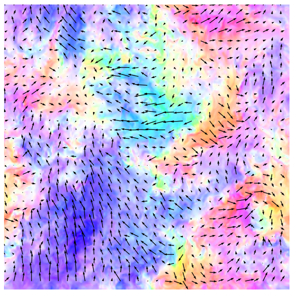
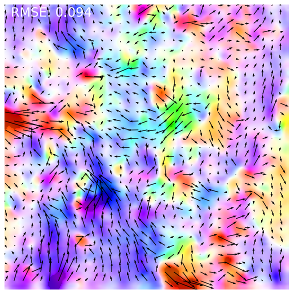
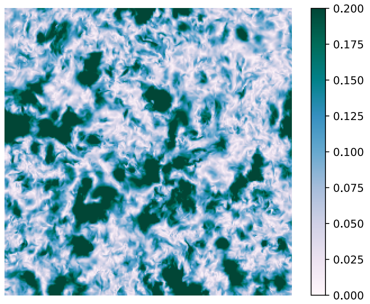
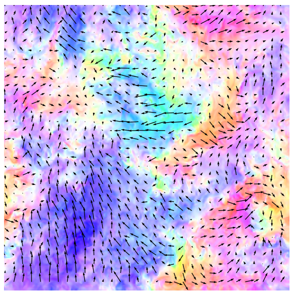
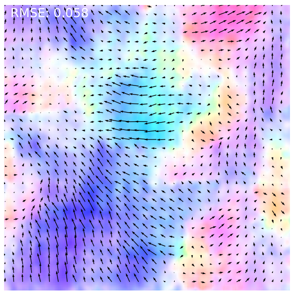
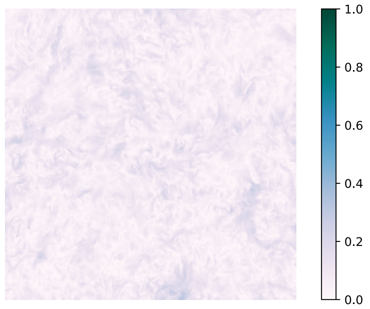
Methods	Ground truth	Prediction	Avg. Endpoint Error
The Horn-Schunck			
PIV-LiteFlowNet-en			
Un-LiteFlowNet			
Memory-PIVnet			

Table 4.2: Visualized ground truth, estimated velocity fields and average endpoint error (AEE) plots from The Horn-Schunck, PIV-LiteFlowNet-en, Un-LiteFlowNet, and Memory-PIVnet on Forced Isotropic turbulence dataset.

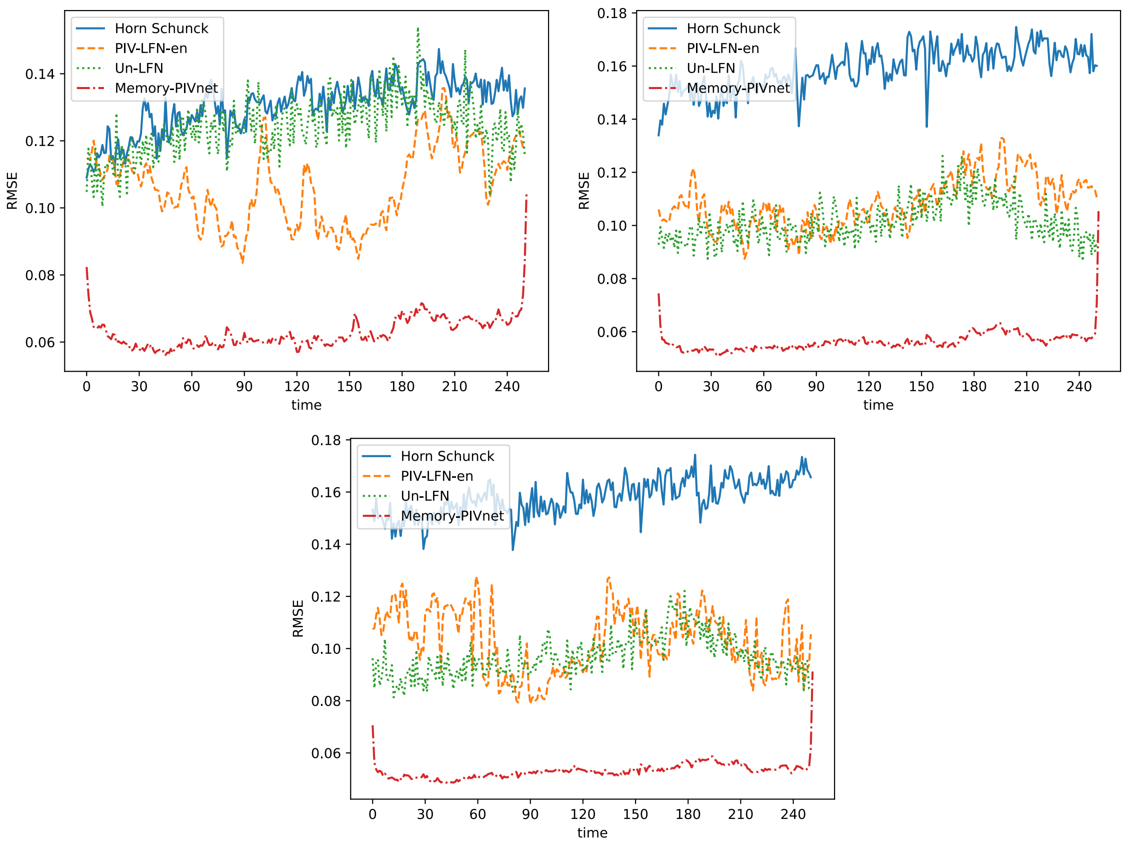


Figure 4.1: Individual RMSE (unit: pixel) of different approaches with each Forced Isotropic Turbulence particle image simulated with seed density $\rho = 39.68$(left), $\rho = 198.41$(middle) and $\rho = 396.83$(right) (unit: ppf)

grid, with the same corresponding spatial domain in the world space. The 2nd-order Adams-Bashforth, with viscous and resistive terms solved analytically by an integrating factor, is used to solve the time integration scheme. The simulation time-step (DNS step) is 0.00025 seconds, while data is stored every 10 DNS steps, i.e. $\Delta t_{\text{data}} = 0.0025$. 1024 time samples are saved for the entire simulation.

As before, the 1024^3 grid is divided into $1024 \times 1024 \times 101$ slices, and down-sampled to $256 \times 256 \times 101$. 9 of such slices are taken. Particle images are generated every 20 data time-steps throughout the 1024 data frames, which makes $\Delta t_{\text{image}} = 0.05$ seconds. 52 images are generated for each sequence. The length of the time window for Memory-PIVnet is again kept at 5, which is the same as the previous experiment. The tiling operation is also the same as before. Among all the sequences, 8 sequences are used for training, and 1 is used for testing, which makes 3328 blocks/samples for training while 416 for testing. During testing, the tiles were stitched together and blended the same way as discussed in Section 4.1 to produce the final result.

Image pairs are generated from the same underlying velocity fields to serve the purpose of training (if applicable) and testing current image pair based methods. Average and individual RMSE of all the approaches are illustrated in Table 4.3 and Figure 4.2 respectively.

Methods / Density ρ	HS	PIV-LiteFlowNet-en	Un-LiteFlowNet	Memory-PIVnet
39.68	0.07585	0.02527	0.03738	**0.02184**
198.41	0.07107	0.02390	0.03428	**0.02234**
396.83	0.07203	0.02412	0.03388	**0.02116**

Table 4.3: Mean RMSE (unit: pixel) of different approaches on Forced Magneto-Hydrodynamic (MHD) Turbulence particle images

4.3 Discussion

In previous sections, evaluations and accuracy comparisons between Memory-PIVnet and other methods are illustrated. However, our results support only relative, not absolute, comparisons between the accuracies of the methods, because the particle images contain less information than the underlying (downsampled) velocity field. Parts of the image sparsely populated by particles are not expected to contain sufficient information for any algorithm to predict its underlying velocity field, so even the perfect PIV solver could not achieve 100% accuracy. We believe there is thus a theoretical upper bound on the attainable accuracy of a PIV method relative to a turbulent vector field ground truth. This bound has not yet been estimated within the current scope of our work (see Section 6.2).

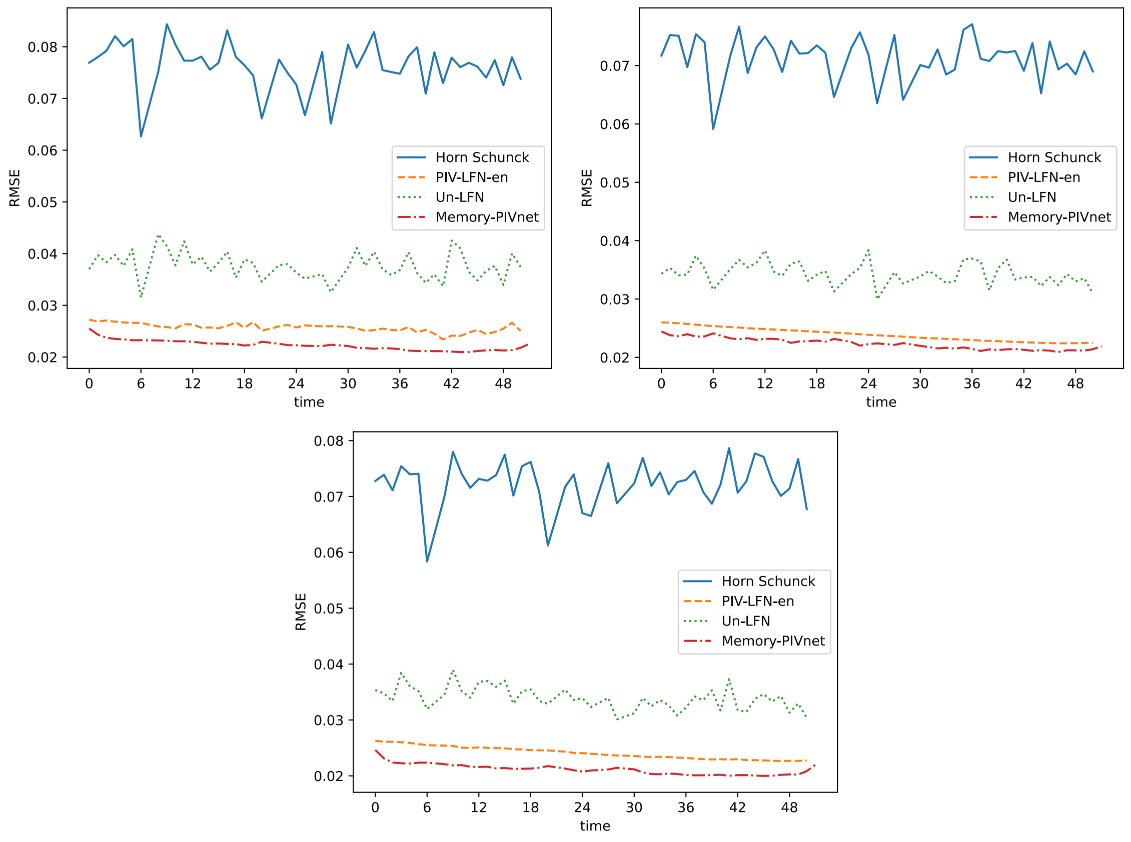


Figure 4.2: Individual RMSE (unit: pixel) of different approaches with each Forced Magneto-Hydrodynamic (MHD) Turbulence particle image simulated with seed density $\rho = 39.68$(left), $\rho = 198.41$(middle) and $\rho = 396.83$(right) (unit: ppf)

CHAPTER 5

ABLATION STUDY

The core idea of this work is to utilize a larger time window to improve the accuracy in solving PIV. To achieve this, a network structure consist of memory network and multi-level flow estimation network are proposed and tested. However, it remains unanswered that how would the network perform with different lengths of time windows? And does it matter or not to keep a persistent memory stage of the memory network during the entire image sequence? In other words, should the memory network be amnesia or non-amnesia? Therefore, in this chapter, we present an ablation studies that explore these questions.

5.1 Different Lengths of Time Windows

To experiment with different time windows, we use the particle images generated from the forced isotropic turbulence with simulation seed density $\rho = 39.68$, as used in Section 4.1. Time windows include $T = \{2, 3, 5, 7, 9\}$. Note that when $T = 2$, only a pair of images are used during training and testing. This contradicts with our idea that knowing both the past and future frames would be helpful. However, this setting puts an equal relationship on the data needed per inference between our method and other existing work.

Throughout this experiment, the network structure is kept the same as introduced in Section 3.1, with the only change being the length of the time window. Although smaller time windows result in a less complex model, which consumes less memory on GPU, and makes the tiling strategy less necessary, to keep the experiments an apple-to-apple comparison with as few difference as possible, the tiling and stitching scheme is preserved even if they are not strictly required.

We demonstrate the average RMSE results as well as the individual RMSE curve in Table 5.1 and Figure 5.1, respectively. The Memory-PIVnet is abbreviated as MPN and are labelled with the lengths of time intervals at the end. In addition to the various multi-frame

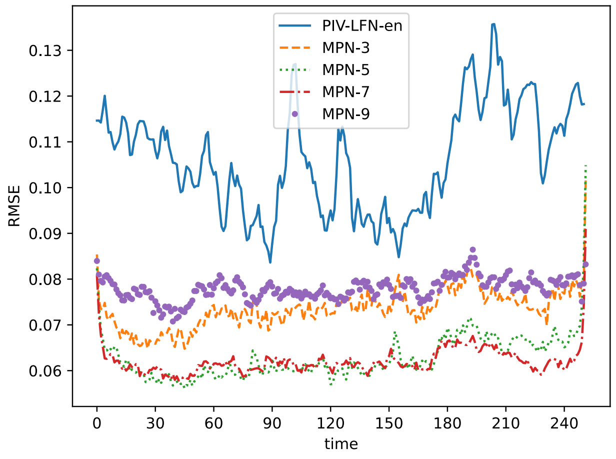


Figure 5.1: Bilinear interpolation blending in boundary removal

Memory-PIVnet, we also show the results from PIV-LiteFlowNet-en (abbreviated as PLFN-en) to serve as a baseline. A more detailed visualization that compares ground truths and predictions is showed in Table 5.2.

Methods / Density ρ	PLFN-en	MPN-3	MPN-5	MPN-7	MPN-9
39.68	0.10625	0.07349	**0.06283**	0.06206	0.07788

Table 5.1: Root mean squared error (unit: pixel) by PIV-LiteFlowNet-en (PLFN-en), Memory-PIVnet (MPN) and its variants with different time windows on forced isotropic turbulence dataset simulated with seed density $\rho = 39.68$ (unit: ppf)

From the results, we can conclude that $T = 5$ produces the best performance. When the time window is too small, not enough information has been provided to the network, which causes the estimated velocity field lack of fine details. On the other hand, when too many frames have been padded as input data, the network can be confused by potential misleading information contained in the further-away, less-related frames. It causes overly large velocity estimations in certain areas, which can be more clearly seen in the average endpoint error plots (third column in Table 5.2).

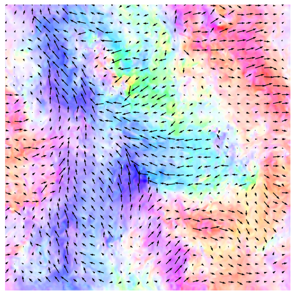
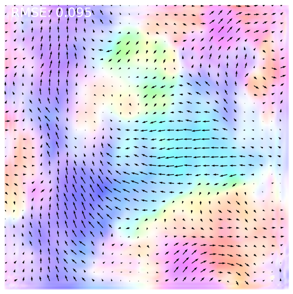
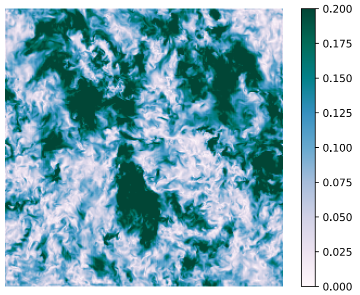
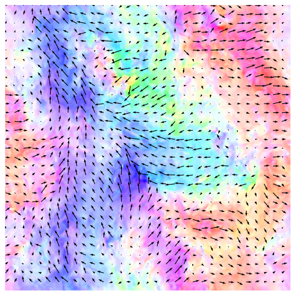
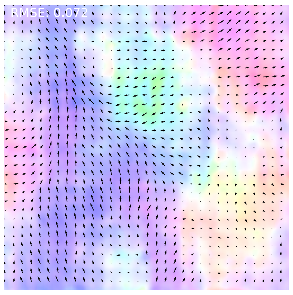
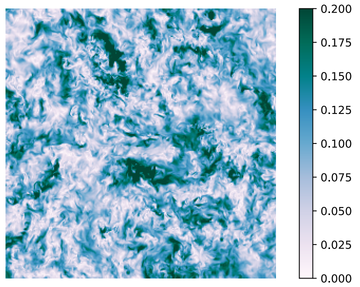
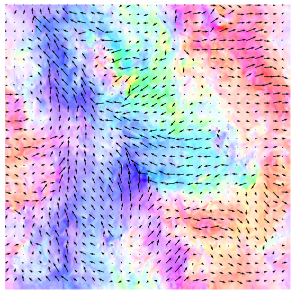
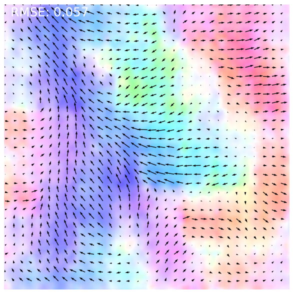
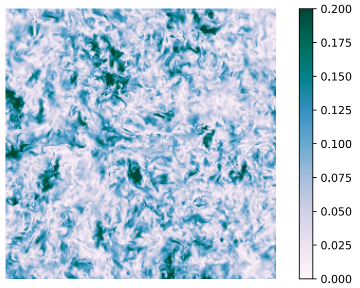
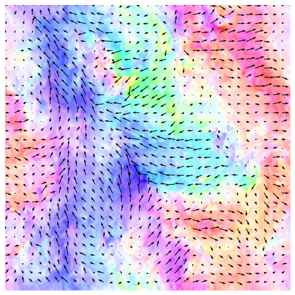
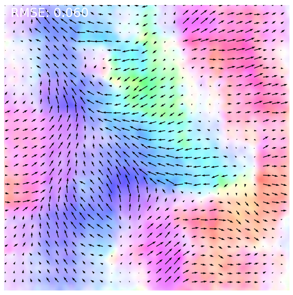
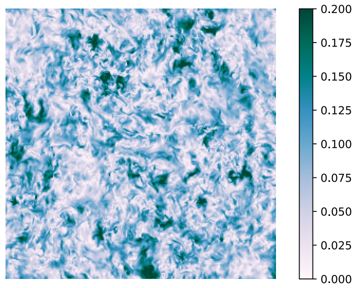
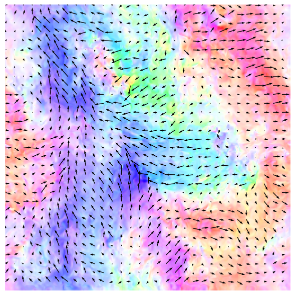
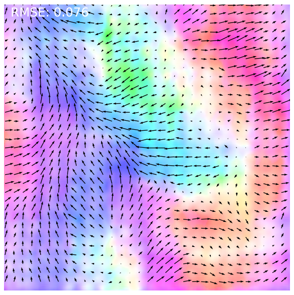
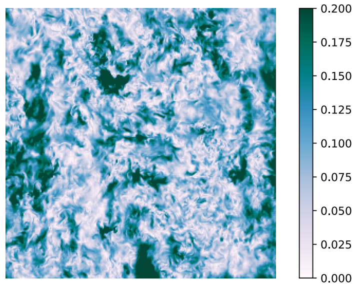
Methods	Ground truth	Prediction	Avg. Endpoint Error
PIV-LiteFlowNet-en			
Memory-PIVnet-3			
Memory-PIVnet-5			
Memory-PIVnet-7			
Memory-PIVnet-9			

Table 5.2: Ground truth and estimated velocity fields from PIV-LiteFlowNet-en (PLFN-en), Memory-PIVnet (MPN) and its variants with different time windows on forced isotropic turbulence simulated with seed density $\rho = 39.68$ (unit: ppf)

5.2 Amnesia and Non-amnesia Memory Network

When training with the memory network, it is a design choice to either initialize a new memory stage for each set of nearby frames, or keep the memory stage persistent across the entire sequence. More specifically, suppose we are using the forced isotropic turbulence dataset and training with a sequence of 252 particle images, with time window $T = 5$. After training with the first sample block that contains images $\{I_0, I_0, I_0, I_1, I_2\}$, where the subscripts indicate the images' indices in the sequence, the memory network can either initialize the new memory stage when processing the next sample, which contains images $\{I_0, I_0, I_1, I_2, I_3\}$, or pass the memory stage from processing I_2 to the next sample to replace the initialization. We call these two variants amnesia and non-amnesia respectively.

Ideally, if both variants produced the same level of accuracy, the non-amnesia version would be more attractive since it is more efficient during testing. In this section, we explore the two variants and compare their performance. Same as the previous section, we use the particle images generated from the forced isotropic turbulence with simulation seed density $\rho = 39.68$. To rule out the possibility that amnesia or non-amnesia might be related to the certain length of time windows, we compare the results for 3 sets of time windows. Average RMSE and individual RMSE curve are showed in Table 5.3 and Figure 5.2 respectively. Table 5.2 and 5.5 illustrates an example result from both approaches.

Methods / Time windows T	MPN-Amnesia	MPN-Non-Amnesia
3	**0.07349**	0.11640
5	**0.06283**	0.10384
7	**0.06206**	0.08804

Table 5.3: Root mean squared error (unit: pixel) by Memory-PIVnet-Amnesia (MPN-Amnesia) and Memory-PIVnet-Non-Amnesia (MPN-Non-Amnesi) with different time windows on forced isotropic turbulence dataset simulated with seed density $\rho = 39.68$ (unit: ppf)

The results demonstrate distinctive performance difference between the amnesia and non-

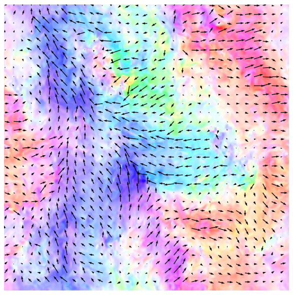
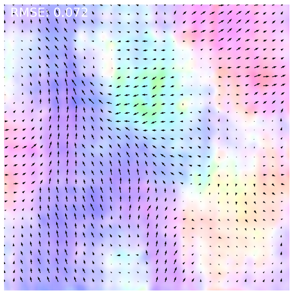
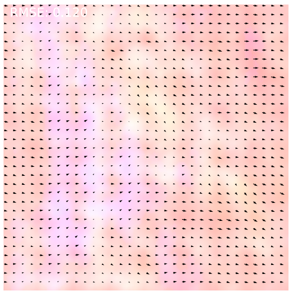
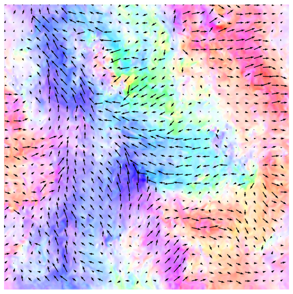
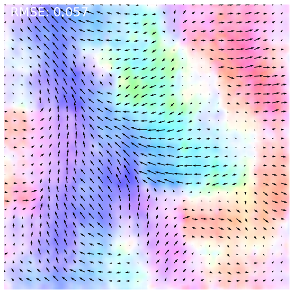
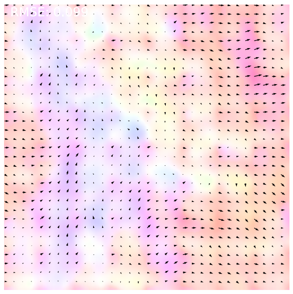
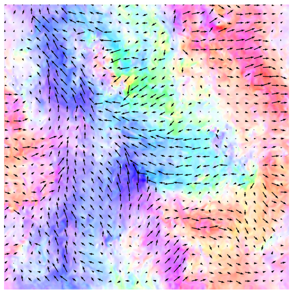
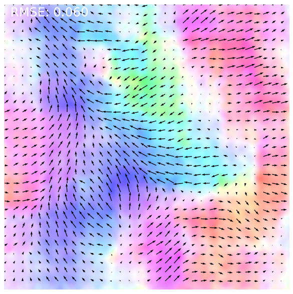
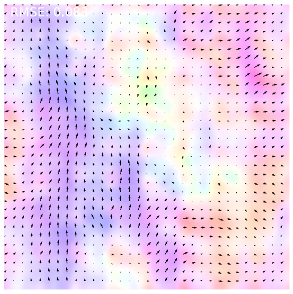
Time window	Ground truth	Amnesia	Non-Amnesia
$T = 3$			
$T = 5$			
$T = 7$			

Table 5.4: Ground truth and estimated velocity fields from the amnesia and non-amnesia version of Memory-PIVnet with different time windows on forced isotropic turbulence simulated with seed density $\rho = 39.68$ (unit: ppf)

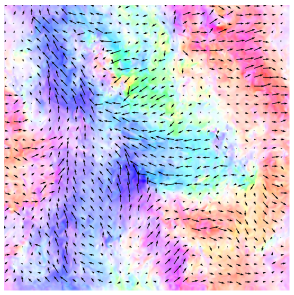
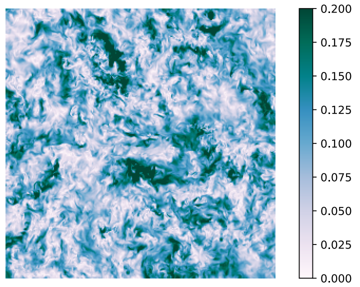
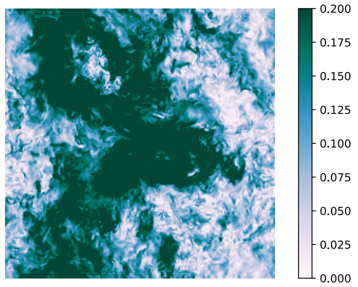
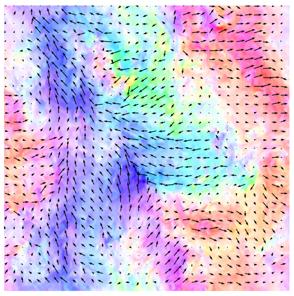
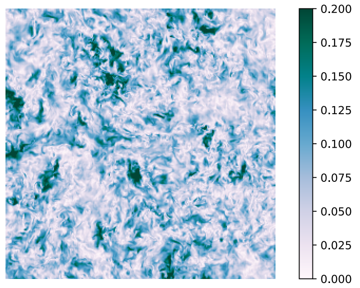
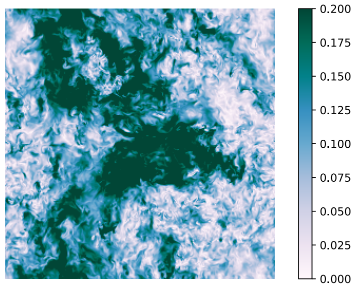
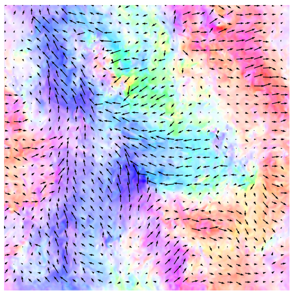
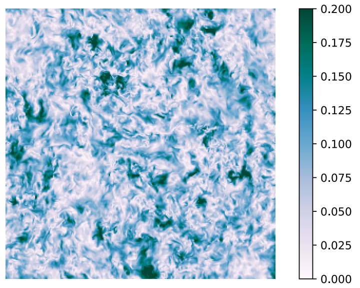
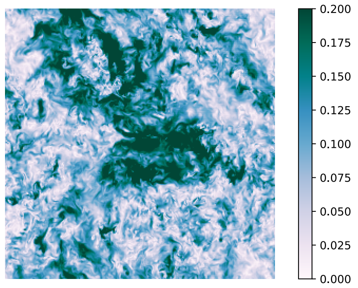
Time window	Ground truth	Amnesia	Non-Amnesia
$T = 3$			
$T = 5$			
$T = 7$			

Table 5.5: Ground truth and average endpoint error from the amnesia and non-amnesia version of Memory-PIVnet with different time windows on forced isotropic turbulence simulated with seed density $\rho = 39.68$ (unit: ppf)

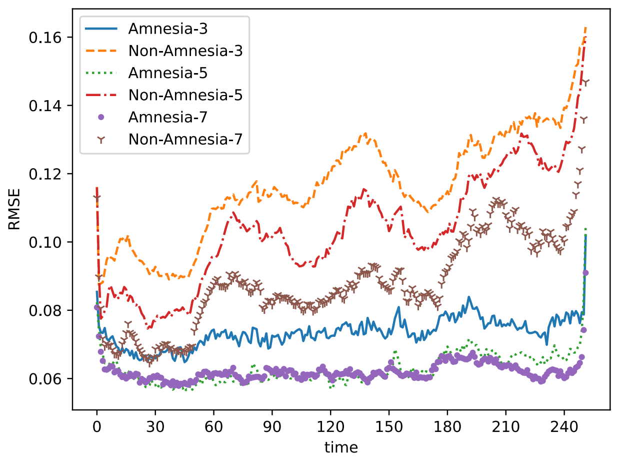


Figure 5.2: Bilinear interpolation blending in boundary removal

amnesia Memory-PIVnet across all time windows. And it is proficient to draw the conclusion that the amnesia version with re-initialized memory stage would be a better pick for solving PIV problems.

CHAPTER 6
CONCLUSIONS AND FUTURE WORKS

6.1 Conclusions

In this paper, we propose a new deep learning approach for solving particle velocimetry. Our method starts with the assumption that multiple consecutive frames of particle images can be more helpful than image pairs. A new network structure, which combines a memory network [11] and a multi-level flow estimation network is designed to implement and test this idea. The input of the network is a block of nearby particle images padded along time axis, and the output is a dense velocity field with displacement vectors at every pixel.

The proposed deep learning model is trained and evaluated with different synthetic turbulence datasets from JHTDB [18]. Experiment results show that our method, Memory-PIVnet, achieves the best performance in terms of RMSE and AEE among the current state-of-the-art PIV solutions. A number of ablation studies, which explore how various lengths of time windows, persistent or non-persistent internal memory stage, affect the model performance have been conducted and discussed. We conclude that our Memory-PIVnet performs the best with moderate length time windows ($T = 5$) and non-persistent memory stage.

Applying machine learning techniques on solving PIV is still at an early stage, with few existing work to prove that it is guaranteed to work much better than the traditional CC-based or optical flow methods. Therefore, the approach that this paper contains, which performs over 40% better than existing methods, provides a big leap in terms of showing that deep learning can be a promising PIV path. Memory-PIVnet is also, to our knowledge, the first network structure that is designed from the starting point of solving PIV, instead of building upon existing methods that more broadly work with general optical flow problems in computer vision.

6.2 Future Works

Although the proposed method shows good performance on synthetic turbulence dataset, it is undoubtedly a more complex model which requires longer training and inference time. It is less time-efficient than other methods, especially when the target turbulence field is less complex, which brings the potential performance gap smaller. Therefore, to work better on the real applications, a better balance between the accuracy and efficiency should be investigated.

When evaluating the model's performance, the current work deploys a scalar error comparison (RMSE) between our model and other methods. However, such comparison could be insufficient when evaluating other properties, such as equivariance properties, that scientific measurement tools like PIV should preserve. Therefore, another potential future work could be developing evaluation procedures that visually explore and display the success and failures of these properties. In addition, like mentioned earlier in Section 4.3, theoretical upper bound of the PIV performance, which is based on the information that particle images contain, could be another future work that help evaluate the model performance.

PIV traditionally ends with estimating the underlying velocity fields, from which other quantities of interest can be computed, including *vorticity*, an important quantity in turbulence studies. Predicting vorticity directly from particle images without the need to go through full velocity fields and their numerical differentiation could be very beneficial. Traditional methods do not have the ability to do so, but learning-based algorithms enjoy the privilege to achieve this goal by simply changing the ground truth to the vorticity. It could also potentially do a better job than predicting velocity fields, as one degree of freedom is dropped during training. Therefore, exploring the model's ability to predict vorticity directly, and comparing it with the vorticity results computed through velocity from traditional state-of-the-arts algorithms is another direction of future work.

BIBLIOGRAPHY

[1] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. 92(1):1–31.

[2] Shengze Cai, Jiaming Liang, Qi Gao, Chao Xu, and Runjie Wei. Particle image velocimetry based on a deep learning motion estimator. 69(6):3538–3554. Conference Name: IEEE Transactions on Instrumentation and Measurement.

[3] Shengze Cai, Shichao Zhou, Chao Xu, and Qi Gao. Dense motion estimation of particle images via a convolutional neural network. 60(4):73.

[4] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks.

[5] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering.

[6] Dominique Heitz, Etienne Mémin, and Christoph Schnörr. Variational fluid flow measurements from image sequences: synopsis and perspectives. 48(3):369–393.

[7] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. 17(1):185–203.

[8] Tak-Wai Hui and Chen Change Loy. LiteFlowNet3: Resolving Correspondence Ambiguity for More Accurate Optical Flow Estimation. pages 169–184, 2020.

[9] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A lightweight optical flow CNN - revisiting data fidelity and regularization.

[10] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. LiteFlowNet: A lightweight convolutional neural network for optical flow estimation.

[11] Tri Huynh, Michael Maire, and Matthew R. Walter. Multigrid neural memory. In *International Conference on Machine Learning (ICML)*, 2020.

[12] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift.

[14] Tsung-Wei Ke, Michael Maire, and Stella X. Yu. Multigrid neural architectures.

[15] Hwar C. Ku, Richard S. Hirsh, and Thomas D. Taylor. A pseudospectral method for solution of the three-dimensional incompressible navier-stokes equations. 70(2):439–462.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. 521(7553):436–444.

[17] Yong Lee, Hua Yang, and Zhouping Yin. PIV-DCNN: cascaded deep convolutional neural networks for particle image velocimetry. 58(12):171.

[18] Yi Li, Eric Perlman, Minping Wan, Yunke Yang, Charles Meneveau, Randal Burns, Shiyi Chen, Alexander Szalay, and Gregory Eyink. A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence. 9:N31.

[19] Tianshu Liu, Ali Merat, M. H. M. Makhmalbaf, Claudia Fajardo, and Parviz Merati. Comparison between optical flow and cross-correlation methods for extraction of velocity fields from particle images. 56(8):166.

[20] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

[21] Shahzad Muzaffar and Afshin Afshari. Short-term load forecasts using LSTM networks. 158:2922–2927.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[23] Jean Rabault, Jostein Kolaas, and Atle Jensen. Performing particle image velocimetry using artificial neural networks: a proof-of-concept. 28(12):125301.

[24] M. Raffel, C. Willert, S. Wereley, and J. Kompenhans. *Particle Image Velocimetry: A Practical Guide*. Springer, 2007.

[25] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting.

[26] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-net: CNNs for optical flow using pyramid, warping, and cost volume.

[27] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3476–3483. ISSN: 1063-6919.

[28] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow.

[29] Mei Wang and Weihong Deng. Deep face recognition: A survey.

[30] J Westerweel. Fundamentals of digital particle image velocimetry. 8(12):1379–1392.

[31] Jerry Westerweel and Fulvio Scarano. Universal outlier detection for PIV data. 39(1096).

[32] Bernhard Wieneke and Karsten Pfeiffer. Adaptive PIV with variable interrogation window size and shape. page 8.

[33] Xuezhi Xiang, Mingliang Zhai, Rongfang Zhang, Ning Lv, and Abdulmotaleb El Saddik. Optical flow estimation using spatial-channel combinational attention-based pyramid networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1272–1276. ISSN: 2381-8549.

[34] Mingrui Zhang and Matthew D. Piggott. Unsupervised learning of particle image velocimetry.