Restoration of Non-rigidly Distorted Underwater Images using a Combination of Compressive Sensing and Local Polynomial Image Representations

Jerin Geo James
IIT Bombay
jeringeo@cse.iitb.ac.in

Pranay Agrawal
IIT Bombay
pranay.agr09@gmail.com

Ajit Rajwade
IIT Bombay
ajitvr@cse.iitb.ac.in

Abstract

Images of static scenes submerged beneath a wavy water surface exhibit severe non-rigid distortions. The physics of water flow suggests that water surfaces possess spatio-temporal smoothness and temporal periodicity. Hence they possess a sparse representation in the 3D discrete Fourier (DFT) basis. Motivated by this, we pose the task of restoration of such video sequences as a compressed sensing (CS) problem. We begin by tracking a few salient feature points across the frames of a video sequence of the submerged scene. Using these point trajectories, we show that the motion fields at all other (non-tracked) points can be effectively estimated using a typical CS solver. This by itself is a novel contribution in the field of non-rigid motion estimation. We show that this method outperforms state of the art algorithms for underwater image restoration. We further consider a simple optical flow algorithm based on local polynomial expansion of the image frames (PEOF). Surprisingly, we demonstrate that PEOF is more efficient and often outperforms all the state of the art methods in terms of numerical measures. Finally, we demonstrate that a two-stage approach consisting of the CS step followed by PEOF much more accurately preserves the image structure and improves the (visual as well as numerical) video quality as compared to just the PEOF stage.
The source code, datasets and supplemental material *can be accessed at [1], [3].*

1. Introduction

Underwater image analysis is a challenging and relatively less explored area of computer vision. In particular, if a scene submerged in water is imaged by a camera in air, the scene exhibits severe spatial distortions due to dynamic refraction at the wavy/dynamic water surface. Such distortions can interfere with higher level tasks such as object recognition, tracking, motion analysis or segmentation, which are required in applications such as coral reef monitoring, surveillance of shallow riverbeds to observe vegetation [32], or studying of visual perception in water-birds (see references in [5]). Applying the principle of reversibility of light, there also exist applications in submarines where a camera in water is observing scenes in the air [5].

Related Work: There exists a medium-sized body of literature on this topic. The earliest work, to our knowledge, is from [19] where *frame-to-frame* optical flow is estimated using a correlation-based method, and the underlying image is estimated from the centroid of the flow trajectory at each point. Such a method is expensive and error-prone due to ambiguities in optical flow (especially in case of large motion), and reflective or blur artifacts. The work in [29] infers a set of 'water bases' from synthetic underwater scenes generated from the wave equation, and then expresses deformation fields within small patches as linear combinations of the water bases. The work in [20] performs non-rigid registration of blurred versions of all frames in the sequence with an evolving template (initialized to be the mean image) followed by a robust PCA step [7]. Both these methods are expensive and prone to local minima in case of large motion, leaving behind some residual motion or geometric distortion. Along similar lines, [15] proposes a method to register all frames of a video sequence with a 'reference frame', chosen to be a frame with the least blur. The so-called 'lucky region approach' has been developed in [12], [9], [36] and [35]. In this, distortion-free patches which correspond to patches of the image formed due to a locally flat portion of the water surface, are identified and then stitched together using graph embedding. In [25], the restoration is performed based on the assumption that the water surface is a single unidirectional cyclic wave (UCW). The restoration process is framed as a blind deconvolution problem, the input to which is the average of the video frames (equivalent to a motion-blurred image due to long camera exposure). The UCW assumption will not hold in some scenarios, for example if the water waves are formed from a superposition of different constituent waves due to multiple independent disturbances. Recently, [18] presented a deep learning

framework to restore *single* distorted underwater images (as opposed to using video sequences). A neural network was trained on pairs of distortion-free and distorted images, to infer geometric distortion and also apply photometric correction. This method does not account for the extra information in the form of temporal redundancy, which is readily available in even short video sequences. To be effective, it also requires a large amount of training data. In contrast, our method is based on principles of basic physics/geometry. It also does not require training data as in [18], representative distortion-free templates (which are generally hard to acquire) to drive the technique as in [30, 31], multiple illumination sources as in [32] or multiple viewpoints as in [21].

Overview: In this paper, we present a novel method that exploits the inherent spatio-temporal redundancy of water waves. We note that the motion vector fields (MVFs), defined as the collection of displacement vectors at each point in every frame of the video sequence, have a sparse representation in the discrete Fourier basis. This emerges from the spatial smoothness of the flow, and its temporal periodicity as well as smoothness. We begin by tracking some N salient feature points across all T frames of the video, to yield point-trajectories (PTs) $\{p_i \triangleq \{(x_{it}, y_{it})\}_{t=1}^T\}_{i=1}^N$, and then convert these into displacement-trajectories (DTs) $\{d_i \triangleq \{(d_{x,it}, d_{y,it})\}_{t=1}^T\}_{i=1}^N$. Given these DTs, we use a compressed sensing (CS) method to infer the DTs at all other points in the image domain. The specific manner in which we have applied CS for motion estimation in this paper, is a novel contribution (see Sec. 2.3.4 for a comparison to other approaches for sparsity-based motion estimation). We also observe that our method largely reduces the nonrigid motion and outperforms the state of the art methods. Our second major contribution is the usage of an existing optical flow method based on local polynomial image representations [13], for this particular task. Despite its simplicity, we show that this method outperforms the state of the art. Lastly, we show that a two stage approach with CS followed by the optical flow method leads to better video stabilization as well as image structure preservation (visual and numerical) than the optical flow stage alone, at only slightly greater computational cost.

Organization: The main theory behind our method is explained in Sec. 2. The datasets and experiments are described in Sec. 3, followed by a discussion and conclusion in Sec. 4.

2. Theory

We first present a complete description of the various assumptions made in our restoration task, and compare them with those of other aforementioned methods.

2.1. Image Formation

We assume a static planar scene submerged at unknown depth h below a clear, shallow water surface, imaged by an orthographic camera in air whose optical axis is pointing downwards perpendicular to the scene. This assumption is valid in practice (as also seen from our results on real sequences in Section 3) and has also been made in existing work such as [29, 25, 19]. Let I_0 be the original image (size $N_x \times N_y$) as if it were formed in the absence of any wavy water surface. Then the distorted image I_d due to the wavy water surface is given as $I_d(x_0, y_0, t) = I_0(x_0 + d_x(x_0, y_0, t), y_0 + d_y(x_0, y_0, t))$, where $(d_x(x_0, y_0, t), d_y(x_0, y_0, t))$ is the displacement at point (x_0, y_0) (indexing into the *undistorted* image I_0) at time t. A precise relationship between $(d_x(x_0, y_0, t), d_y(x_0, y_0, t))$ and the derivatives of the dynamic height $z(x, y, t)$ of the water surface at the point of refraction, has been derived in previous work [19]. Here, our aim is to estimate $I_0(x_0, y_0)$ for all (x_0, y_0) given $\{I_d(:, :, t)\}_{t=1}^T$. We are assuming that the video frames are largely blur-free, though moderate deviations from this assumption do not affect our theory or results. We ignore effects such as reflection of light from the water surface (which were found to be absent or rare in real videos we gathered or those from [29]).

2.2. Water Surface and Motion Vector Field Model

In our work, we essentially require the wavy water surface to be a smooth signal in space and time, and also temporally periodic. The assumption of spatio-temporal smoothness is common in the literature on this topic, for example [29, 20], and excludes turbulent flows. We do not require the water surface to be explicitly modelled as a linear combination of sinusoidal waves, though our method works very well even for such cases. The motion vector at point (x_{i0}, y_{i0}) (of the underlying undistorted video) at time instant t is denoted as $d_i^{(t)} \triangleq (d_{xit}, d_{yit})$. The complete motion vector field (MVF) can be represented as two 3D signals d_x, d_y in $\mathbb{R}^{N_x \times N_y \times T}$, containing the X- and Y-components of the displacements at every pixel (x_{i0}, y_{i0}) and every time instant t. Due to the spatio-temporal smoothness (and thereby their band-limited nature), both d_x, d_y will admit a sparse (or compressible) decomposition in the Fourier space. For computational reasons, we use the *Discrete* Fourier Transform (DFT) basis. Given the innate interdependence between d_x, d_y (since they emerge from the same wavy water surface), we instead work with a complex-valued vector field $d \triangleq d_x + \iota d_y$ where $d \in \mathbb{C}^{N_x \times N_y \times T}$ and $\iota \triangleq \sqrt{-1}$. This is a natural way to exploit the interdependence. Moreover, if the video sequence is long enough so that the actual MVF is *temporally periodic*, then that fur-

ther contributes to the sparsity of the Fourier domain representation. This is because by definition, periodic signals are sparse in the Fourier domain, even more so if they are band-limited - which is a consequence of smoothness.

The assumption of sparsity of the MVF in the Fourier basis finds corroboration in the fluid mechanics literature. For example in [23, 14, 37], the rapid convergence of the Fourier series of different types of time-periodic velocity vector fields arising in fluid flow, has been demonstrated. Due to this, the water surface height $z(x, y, t)$ and hence the MVFs (which are related to derivatives of $z(x, y, t)$) will also admit sparse Fourier decomposition. In addition, in Sec. 3.3, we present an empirical verification of the Fourier sparsity of the MVF d from real underwater video sequences.

2.3. Method Overview

An overview of our method is summarized in a pseudo-code in Alg. 1. The detailed steps are described further.

---

input : Distorted video I_d
output: Restored image $\bar{I}_r$
1 Track N feature points to obtain point-trajectories $\{p_i\}_{i=1}^N$ as per Sec. 2.3.1.
2 Compute displacement trajectories $\{d_i\}_{i=1}^N$ as per Sec. 2.3.2.
3 Compute the motion vector field (MVF) d as defined in Sec. 2.2 from its measurements $\{d_i\}_{i=1}^N$ using the CS-based method from Sec. 2.3.3.
4 Perform motion correction from the computed MVF to obtain a restored video I_r.
5 Optionally, perform further motion correction on I_r using the PEOF technique from Sec. 2.3.5.
6 Compute mean or median frame of I_r to yield $\bar{I}_r$.

Algorithm 1: Algorithm to Restore Video

2.3.1 Feature point detection and tracking

We begin with a salient feature point detection and tracking algorithm yielding point-trajectories $\{p_i \triangleq \{(x_{it}, y_{it})\}_{t=1}^T\}_{i=1}^N$ for N salient feature points detected in the first frame. The coordinates (x_{it}, y_{it}) represent the position in frame t of the ith point whose coordinates in a distortion-free frame are denoted as (x_{i0}, y_{i0}), where the subscript '0' refers to an index in the undistorted image. Of course, (x_{i0}, y_{i0}) are unknown at the outset. Our salient feature point detection combines four algorithms: (1) difference of Gaussians (DoG) used by SURF[6], (2) the FAST method [24], (3) the popular Harris corner method, and (4) the BRISK technique [17]. Consider a union-set of salient points in the first frame, as detected by all these methods. All points in this set are tracked in subsequent frames using the well-known Kanade-Lucas-Tomasi (KLT) tracker [2].

We obtain excellent results with the standard KLT tracker because it inherently takes care of locally affine motion (a first approximation to non-rigid motion). In some cases, however, we encounter tracking errors. Such trajectories are weeded out and not used in later steps, if (1) they are considered invalid by the KLT tracker itself (which happens when the region around a salient feature point in a frame cannot be accurately expressed as an affine transformation of the corresponding region in a previous frame), or if (2) the center of trajectory (COT), as defined in Sec. 2.3.2, computed over the first $T/2$ and last $T/2$ frames differ by a threshold of more than 3 pixels.

We also trained a Siamese network following [27] to learn good feature descriptors. See `supplemental material` for further details about this. The Siamese network produced slightly better results than the KLT tracker on unseen synthetic and real data. However it did not perform as well as the KLT tracker if there was blur in the video frames. Hence we used the KLT tracker in all experiments. Examples of point tracking on real sequences are shown in the supplemental material folder 'CS_MotionReduction'.

2.3.2 Displacement computation

Following previous definitions of (x_{i0}, y_{i0}) and the point-trajectory p_i, we approximate $\hat{x}_{i0} \triangleq \sum_{t=1}^T x_{it}/T \approx x_{i0}; \hat{y}_{i0} \triangleq \sum_{t=1}^T y_{it}/T \approx y_{i0}$ (termed 'center of trajectory' or COT), although more robust 'means' such as the median can also be considered. This approximation is well justified by the assumption of the local symmetry of water motion, due to which the average surface normal (across time) at any point on the water surface is close to the vertical line [19]. Our experiments for synthetic and real sequences confirm that this is valid even for moderate $T \sim 50$ frames. The supplemental material includes an illustrative example. With this, our set of displacements for the ith salient feature point are given as $d_i \triangleq (d_{ix}, d_{iy}) \triangleq \{(x_{it} - \hat{x}_{i0}, y_{it} - \hat{y}_{i0})\}_{t=1}^T$. We term these as 'displacement-trajectories' (DTs).

2.3.3 MVF Estimation using CS

The DTs $\{d_i\}_{i=1}^N$ can be regarded as sparse samples (in the space-time domain) of the 3D MVF signal d. The signal d is sparse in the Fourier domain (see Sec. 2.2) and hence can be expressed as $\text{vec}(d) = F\theta$ where F is the 3D-DFT basis matrix and θ is a sparse vector of Fourier coefficients. If the DTs are concatenated to form a complex-valued 'measurement vector' e of NT elements, then we have the following model:

$$e = \Phi F \theta + \eta, \tag{1}$$

where Φ is a sampling matrix of size $NT \times N_x N_y T$ and η is a noise vector of NT elements indicating errors in the DTs obtained from the tracking algorithm. Note that

$\boldsymbol{\Phi}$ is the row-subsampled version of the identity matrix of size $N_x N_y T \times N_x N_y T$, and each row of $\boldsymbol{\Phi}$ is a one-hot vector which indicates whether or not the displacement at some pixel (x_{i0}, y_{i0}) (in the undistorted image) at some time frame t was included in the set $\{\boldsymbol{d_i}\}_{i=1}^{N}$ (and hence the measurement vector $\boldsymbol{e}$). The sensing matrix $\boldsymbol{\Phi}$ and representation matrix $\boldsymbol{F}$ are an ideal combination, because they are highly incoherent with each. This augurs well for the application of a CS algorithm for estimation of $\boldsymbol{\theta}$ (and thereby $\boldsymbol{d}$) from $\boldsymbol{e}, \boldsymbol{\Phi}$. This is because CS theory states that $\mathcal{O}(s \log(N_x N_y T) \mu(\boldsymbol{\Phi F}))$ measurements are sufficient for accurate reconstruction of the s-sparse vector $\boldsymbol{\theta}$ with very high probability [8]. Here $\mu(\boldsymbol{\Phi F})$ is the coherence between $\boldsymbol{\Phi}$ and $\boldsymbol{F}$ and is defined as $\mu(\boldsymbol{\Phi F}) \triangleq \max_{i,j,i \neq j} \frac{|\boldsymbol{\Phi^i F_j}|}{\|\boldsymbol{\Phi^i}\|_2 \|\boldsymbol{F_j}\|_2}$ where $\boldsymbol{\Phi^i}$ and $\boldsymbol{F_j}$ are the ith row of $\boldsymbol{\Phi}$ and j^{th} column of $\boldsymbol{F}$ respectively. Given the choice of $\boldsymbol{\Phi}, \boldsymbol{F}$ for our task, μ reaches its lower bound of 1, thereby reducing the number of samples required for reconstruction guarantees. To account for the noise in $\boldsymbol{e}$, we determine $\boldsymbol{d}$ using an estimator (popularly called the LASSO) which minimizes the following objective function:

$$J(\boldsymbol{\theta}) = \lambda \|\boldsymbol{\theta}\|_1 + \|\boldsymbol{e} - \boldsymbol{\Phi F \theta}\|^2. \qquad (2)$$

The regularization parameter λ can be chosen by cross-validation [34] from a set $\mathcal{S}$ of candidate values. That is, for every $\hat{\lambda} \in \mathcal{S}$, a candidate signal $\boldsymbol{d}_{\hat{\lambda}}$ is computed by the LASSO method using a set $\mathcal{T}_1$ of only (say) 90% of the measurements from $\boldsymbol{e}$. Following this, the value of $E(\boldsymbol{d}_{\hat{\lambda}}) \triangleq \sum_{i \in \mathcal{T}_2} |e_i - \boldsymbol{\Phi^i d}_{\hat{\lambda}}|^2$ is computed, where $\mathcal{T}_2$ is the set of the remaining measurements in $\boldsymbol{e}$. The value of $\hat{\lambda}$ that minimizes $E(\boldsymbol{d}_{\hat{\lambda}})$ can be selected. Following this, $\boldsymbol{d}$ is re-estimated by the LASSO method from *all* measurements in $\boldsymbol{e}$ and with the selected $\hat{\lambda}$ value.

2.3.4 Comments Regarding MVF Estimation using CS

Note that our method is very different from the bispectral approach in [36] which chooses 'lucky' (i.e. least distorted) patches, by comparing to a mean template. In that method, the Fourier transform is computed locally on small patches in the spatial domain for finding similarity with corresponding patches from a mean image. On the other hand, our Fourier decomposition is spatio-temporal and global. The idea of dense optical flow interpolation (not specific to underwater scenes) from a sparse set of feature point correspondences has been proposed in the so-called EpicFlow technique [22]. The interpolation uses non-parametric kernel regression or a locally affine method. However our method uses key properties (spatio-temporal smoothness and temporal periodicity) of water waves, and thus considers temporal aspects of the MVFs. This aspect is missing in EpicFlow. Nevertheless, we present comparisons to EpicFlow in Sec. 3.

The use of sparsity based techniques for dense flow field estimation is not entirely new and has been used earlier in [10, 26, 16]. However besides the usage of sparsity for underwater image restoration, there are key differences between our approach and the existing ones. (a) First, these papers use a sparse representation (*eg.*, wavelets [26], learned dictionaries [16] or low-rank and sparse models [10]) for optical flow in small patches unlike our method which is more global. (b) Second, they compute the optical flow only between two frames with a data fidelity term based on the brightness constancy equation (unlike our approach which uses displacement trajectories), they do not consider spatio-temporal patches, and do not account for *temporal redundancy*, which is a readily available and useful prior that our approach exploits.

2.3.5 Polynomial Image Expansions for Optical Flow

The classical optical flow method in [13] expresses small patches from the two images f_1 and f_2, between which the MVF has to be computed, as second-degree polynomials. This method can unduly smooth motion discontinuities as mentioned in [13], but it is well suited to our problem, due to the spatial smoothness of water waves. Consider the following:

$$f_1(\boldsymbol{x}) = \boldsymbol{x}^t \boldsymbol{A_1} \boldsymbol{x} + \boldsymbol{b_1}^t \boldsymbol{x} + \boldsymbol{c_1} \qquad (3)$$
$$f_2(\boldsymbol{x}) \approx f_1(\boldsymbol{x} - \boldsymbol{d}) = (\boldsymbol{x} - \boldsymbol{d})^t \boldsymbol{A_1}(\boldsymbol{x} - \boldsymbol{d}) + \boldsymbol{b_1}^t(\boldsymbol{x} - \boldsymbol{d}) + \boldsymbol{c_1}$$
$$\approx \boldsymbol{x}^t \boldsymbol{A_2} \boldsymbol{x} + \boldsymbol{b_2}^t \boldsymbol{x} + \boldsymbol{c_2},$$

where $\boldsymbol{d}$ is the 2D displacement vector at the point $\boldsymbol{x} \triangleq (x, y)^t$. Consider small patches in the two images respectively, centered around point $\boldsymbol{x}$. The polynomial coefficients $\boldsymbol{A_1}, \boldsymbol{A_2} \in \mathbb{R}^{2 \times 2}; \boldsymbol{b_1}, \boldsymbol{b_2}, \boldsymbol{c_1}, \boldsymbol{c_2} \in \mathbb{R}^{2 \times 1}$ can be determined by local regression. This process is repeated in sliding window fashion all through the image, and so these coefficients become functions of $\boldsymbol{x}$. Assuming a slowly changing MVF, the displacement $\boldsymbol{d}(\boldsymbol{x})$ can be computed in the following manner:

$$\boldsymbol{d}(\boldsymbol{x}) = \Big(\sum_{\tilde{\boldsymbol{x}} \in \mathcal{N}(\boldsymbol{x})} \boldsymbol{A}(\tilde{\boldsymbol{x}})^T \boldsymbol{A}(\tilde{\boldsymbol{x}}) \Big)^{-1} \sum_{\tilde{\boldsymbol{x}} \in \mathcal{N}(\boldsymbol{x})} \boldsymbol{A}(\tilde{\boldsymbol{x}})^T \boldsymbol{\Delta b}(\tilde{\boldsymbol{x}}),$$
$$(4)$$

where $\tilde{\boldsymbol{x}}$ is a point in a neighborhood $\mathcal{N}(\boldsymbol{x})$ around $\boldsymbol{x}$, $\boldsymbol{A}(\tilde{\boldsymbol{x}}) \triangleq (\boldsymbol{A_1}(\tilde{\boldsymbol{x}}) + \boldsymbol{A_2}(\tilde{\boldsymbol{x}}))/2$ and $\boldsymbol{\Delta b}(\tilde{\boldsymbol{x}}) \triangleq (\boldsymbol{b_1}(\tilde{\boldsymbol{x}}) - \boldsymbol{b_2}(\tilde{\boldsymbol{x}}))/2$. Further details about this method can be found in [13]. We term this method as polynomial expansion based optical flow (PEOF). Given $\boldsymbol{d}(\boldsymbol{x})$, the image f_1 is warped, followed by polynomial fitting in the warped version of f_1, and re-estimation of $\boldsymbol{d}(\boldsymbol{x})$. The procedure is repeated iteratively. In the present work, the PEOF method is used to find the MVF between each video frame and the mean image (the average of all video frames). The computed MVFs are

applied to each frame to obtain the restored video. These restored images are then averaged to yield a final restored image. As shown in Sec. 3, this method outperforms all state of the art methods in terms of image restoration quality as well as computation speed.

3. Experimental Results

In this section, we present an extensive suite of results on both synthetic and real video sequences. All image and video results are available in the `supplemental material`.

3.1. Description of Datasets

We created several synthetic 50 fps videos of size $\sim 512 \times 512 \times 101$ by simulating the refraction model from [19] on different images containing objects/text, for a scene depth of 25 cm below the water surface. The water surface was generated using superposition of $2 \leq K \leq 6$ sinusoidal waves with randomly chosen parameters. We henceforth refer to this dataset as **Synthetic**. We also gathered real video sequences (of size $\sim 700 \times 512 \times 101$ with a 50 fps camera) of laminated posters kept at the bottom of a water-tank, with waves generated by mechanical paddles. See `supplemental material` for details of the acquisition. Visual inspection revealed that blur was occasionally present in some frames. We henceforth refer to this dataset as **Real1**. For ground truth (I_0), we also acquired a single image of the same posters under still water with the same camera settings. We also demonstrate results on three text sequences (size $\sim 300 \times 250 \times 101$ at 125 fps) obtained from [29], for which ground truth was available. We henceforth refer to this dataset as **Real2**. Note that Real1 is a more challenging dataset than Real2 due to greater frame-to-frame motion - see Table 2 for the standard deviation values of the motion $\sigma_{motion} \triangleq \sqrt{\sum_{i=1}^{N} \sum_{t=1}^{T} ((x_{it} - \hat{x_{i0}})^2 + (y_{it} - \hat{y_{i0}})^2)/(NT - 1)}$, computed over salient point trajectories.

3.2. Key Parameters and Comparisons

We compared restoration results for several algorithms: (1) our CS-based method (CS) from Sec. 2.3.3; (2) our PEOF method from Sec. 2.3.5; (3) the CS-based method followed by the PEOF method (CS+PEOF); (4) the two-stage method in [20] consisting of spline-based registration followed by RPCA (SBR-RPCA) which is considered state of the art for underwater image restoration; (5) the method from [29] using learned water bases (LWB); and (6) the deep learning (DL) approach from [18]. For SBR-RPCA and LWB, we used code provided by the authors with default parameters. For DL, we used the pre-trained network and code provided by the authors, on each video frame separately and then computed the mean image. We per-

formed all computation and quality assessment with each video frame resized to 256×256 (after suitable cropping to maintain aspect-ratio), as required by their specific implementation. For CS, we used the well-known YALL1 (Basic) solver [4], which allows for ℓ_1-norm optimization of complex-valued signals. We observed better and faster results in practice by downsampling the DTs comprising e by a factor of 8 in X, Y directions (which is in tune with the bandlimited nature of water waves), followed by CS reconstruction and subsequent upsampling to obtain the final reconstructed MVF. For PEOF, we used the OpenCV implementation with a multi-scale pyramidal approach with 3 levels, a pyramid scale of 0.5 and 10 iterations (i.e. the default parameters). For quality assessment referring to ground truth, we used the following measures: (i) visual inspection of the restored video I_r as well as the mean-frame $\bar{I}_r$ of the restored video, (ii) RMSE computed as $\|\bar{I}_r - I_0\|_2 / \|I_0\|_2$ where I_0 is the image representing the undistorted static scene, (iii) normalized mutual information (NMI) between $\bar{I}_r$ and I_0, and (iv) SSIM [33] between $\bar{I}_r$ and I_0.

We also considered comparing PEOF to a very competitive optical flow algorithm: EpicFlow [22] (EF). For EF [22], we used the authors' code to estimate the deformation of all video frames w.r.t. the mean frame of the video. We then applied the deformations to each frame to yield the final image. Results comparing PEOF and EF are included in the `supplemental material`, and show that PEOF outperforms EF for underwater image restoration. Note that the EF method has so far *not* been applied for this task in the literature. We do not present results with the state of the art deep learning approaches for optical flow such as [11] or [28] here, for two reasons: (i) EF yielded superior results on our data compared to [11], and (ii) the results of PWC-net from [28] show only a small improvement over EpicFlow on some datasets such as Sintel.

We did not compare with the work in [25] because it relies on a unidirectional wave motion assumption (whereas we assume more general wave models), and due to unavailability of publicly released code. Also, we did not explicitly compare our results with the method in [15] for which publicly released code is unavailable. However, we observed that CS-PEOF outperformed the method of [15] on Real2 (compare 'Middle', 'Small' and 'Tiny' in Table 1 to 'Large', 'Medium' and 'Small' respectively in Table 1 of [15]).

3.3. Discussion of Results

The numerical results are presented in Table 1. The mean images of three real videos restored by various methods are presented in Figs. 3. The `supplemental material` contains results on 14 videos (mean images and restored videos) for all methods. From these results, it is clear that our methods (CS, PEOF and CS-PEOF) yield results sur-
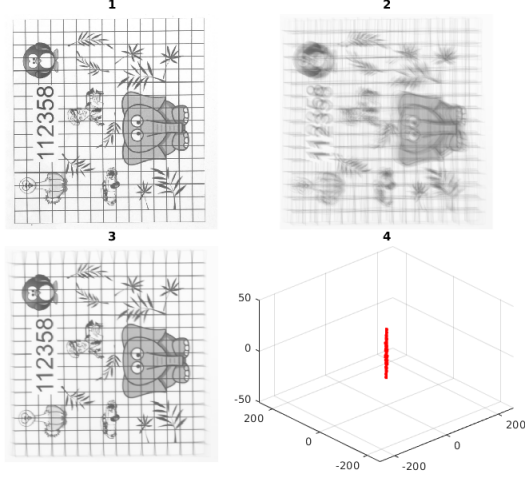
Figure 1. Verification of Fourier domain sparsity of MVF d estimated from a real sequence. Top row: original undistorted image acquired in still water (left), mean of distorted video sequence (right); Bottom row: mean of restored video sequence using CS (left), scatter plot of frequencies which account for 99% of the squared magnitude of the estimated MVF using CS (right).
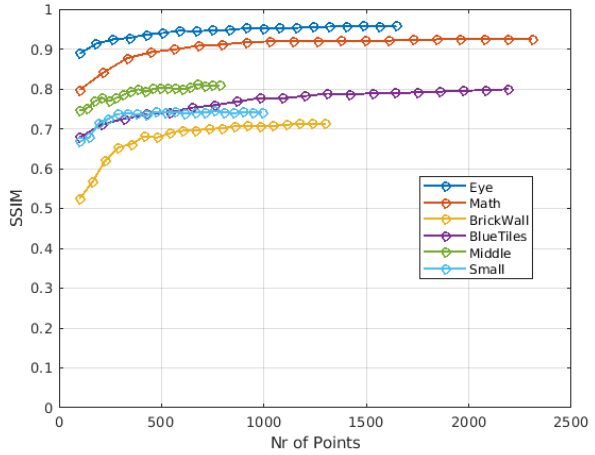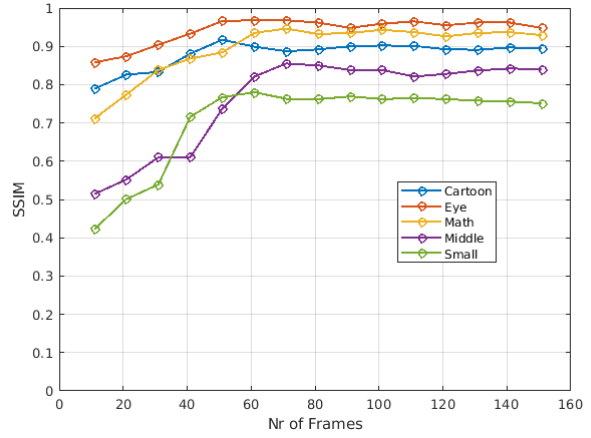

Figure 2. Effect of increase in number of frames T (top) and number of salient points N (bottom) on restoration performance for CS method. Results shown on 'Middle' and 'Small' from Real2, and a few sequences from Real1/Synthetic.

passing SBR-RPCA, LWB and DL on synthetic as well as real datasets, numerically and also in terms of visual quality. We also supplemented our method with a step involving RPCA [7] to remove sparse artifacts, which improved video stability but had very little impact over the quality of the mean image. Although PEOF produces superior numerical results to CS, we have observed that CS produces restored videos and mean images with superior visual quality as compared to PEOF - see Fig. 3 (grid lines on 'Elephant', words 'Imaging', 'Distortion' in 'Middle') as well as `supplemental material`. Additionally, we observed that SBR-RPCA, DL and LWB *did not preserve the image structure* as well as our method (see grid-lines in 'Elephant', the words 'Fluctuation' or 'Distortion' in 'Middle' and alphabets E, T, Z, large D in 'Eye' in Fig. 3. We do believe the DL method [18] may yield improved results if their network were trained to restore multiple frames together, as opposed to single frames individually (as done by their current algorithm) - which ignores the temporal aspect leading to loss in performance. All in all, our results show that exploiting spatio-temporal properties of water waves for this task is indeed useful.

Computational Time: The compute time for all methods (measured on a 2.6GHz Intel Xeon machine with 32GB RAM) are presented in Table 1. The DL method is the fastest, whereas our methods are much faster than SBR-RPCA and LWB. However for CS, the YALL1 solver uses GPU support, which is unavailable in the authors' code for SBR-PCA and LWB. We note that although cross-validation is an excellent way to pick the $\hat{\lambda}$ parameter in

Eqn. 2, we found that the optimal choice of this parameter did not change much across datasets. Also small changes in $\hat{\lambda}$ did not affect the performance much. Hence the time for cross-validation is not included in Table 1.

Verification of Fourier Sparsity: Here, we demonstrate the sparsity of the MVFs from real underwater sequences. This is shown in Fig. 1 for the 'Elephant' sequence (similar plots can be generated for all other sequences). We note that the actual MVF can only be *estimated*. However, we contend that the MVF estimated by our CS method is a good approximation to the actual one. This is evident when comparing the quality of the estimated mean image with the original undistorted image (gathered in still water). For further quantification, we also tracked the same N points (from the distorted video I_d) that were used for the CS algorithm in Sec. 2.3.3, in the restored video (I_r) produced by the CS step. This gave us new DTs $\{\hat{d}_i\}_{i=1}^N$.

Table 1. Comparison of various methods on synthetic and real video sequences w.r.t. compute time (h=hours,m=mins.,s=secs.), NMI, SSIM, RMSE. Lower RMSE, higher SSIM and NMI are better.

	CS				PEOF				CS+PEOF			
Dataset	**Time**	**NMI**	**SSIM**	**RMSE**	**Time**	**NMI**	**SSIM**	**RMSE**	**Time**	**NMI**	**SSIM**	**RMSE**
Real1												
Cartoon	0m 42s	1.227	0.902	0.065	0m 41s	1.216	0.913	0.062	1m 23s	1.255	**0.928**	0.057
Checker	1m 9s	1.206	0.884	0.104	0m 40s	1.196	0.89	0.105	1m 49s	1.22	**0.892**	0.104
Dices	1m 20s	1.172	0.937	0.067	0m 40s	1.139	0.905	0.075	2m 1s	1.188	**0.956**	0.059
Bricks	1m 1s	1.148	0.785	0.142	0m 34s	1.151	0.803	0.121	1m 36s	1.167	**0.843**	0.118
Elephant	0m 28s	1.128	0.801	0.141	0m 26s	1.102	0.763	0.152	0m 55s	1.132	**0.808**	0.143
Eye	1m 22s	1.266	0.961	0.052	0m 57s	1.26	0.975	0.042	2m 19s	1.303	**0.982**	0.037
Math	1m 19s	1.193	0.942	0.05	0m 37s	1.163	0.929	0.053	1m 56s	1.215	**0.961**	0.044
Synthetic												
BlueTiles	0m 28s	1.141	0.792	0.256	0m 23s	1.141	0.816	0.204	0m 52s	1.161	**0.871**	0.182
BrickWall	0m 23s	1.094	0.667	0.144	0m 24s	1.098	0.69	0.142	0m 47s	1.1	**0.703**	0.141
Vision	29s	1.181	0.938	0.09	23s	1.162	0.916	0.113	52s	1.211	**0.972**	0.066
HandWritten	0m 37s	1.123	0.878	0.081	0m 23s	1.131	0.907	0.077	1m 0s	1.156	**0.938**	0.075
Real2												
Middle	0m 13s	1.192	0.838	0.139	0m 7s	1.211	0.85	0.165	0m 20s	1.23	**0.914**	0.101
Small	0m 9s	1.169	0.763	0.164	0m 6s	1.182	0.772	0.206	0m 16s	1.195	**0.849**	0.133
Tiny	0m 11s	1.166	0.661	0.201	0m 7s	1.176	0.698	0.263	0m 19s	1.186	**0.745**	0.19

	SBR-RPCA[20]				LWB[29]				DL[18]			
Dataset	**Time**	**NMI**	**SSIM**	**RMSE**	**Time**	**NMI**	**SSIM**	**RMSE**	**Time**	**NMI**	**SSIM**	**RMSE**
Real1												
Cartoon	3h 2m	1.173	0.843	0.111	0h 54m	1.152	0.836	0.095	3s	1.203	0.803	0.162
Checker	4h 9m	1.158	0.791	0.239	1h 37m	1.105	0.66	0.322	3s	1.129	0.544	0.384
Dices	3h 58m	1.1	0.758	0.17	1h 26m	1.086	0.783	0.126	3s	1.085	0.637	0.242
Bricks	3h 43m	1.128	0.686	0.192	1h 24m	1.118	0.673	0.225	3s	1.058	0.49	0.422
Elephant	3h 7m	1.075	0.516	0.257	0h 59m	1.068	0.584	0.204	3s	1.075	0.378	0.347
Eye	4h 4m	1.179	0.913	0.104	1h 22m	1.155	0.903	0.089	3s	1.141	0.804	0.191
Math	4h 34m	1.1	0.841	0.102	3h 0m	1.067	0.766	0.1	3s	1.073	0.678	0.139
Synthetic												
BlueTiles	2h 5m	1.142	0.763	0.372	0h 55m	1.104	0.72	0.204	3s	1.091	0.365	1.067
BrickWall	2h 30m	1.093	0.666	0.158	1h 0m	1.066	0.481	0.19	3s	1.079	0.479	0.218
Vision	3h 4	1.115	0.739	0.216	36m	1.021	0.446	0.266	3s	1.095	0.599	0.215
HandWritten	0h 0m	1.112	0.851	0.12	0h 52m	1.073	0.678	0.147	3s	1.074	0.546	0.177
Real2												
Middle	1h 28m	1.189	0.782	0.204	0h 54m	1.163	0.761	0.194	3s	1.122	0.512	0.307
Small	1h 21m	1.153	0.741	0.181	0h 33m	1.151	0.688	0.198	3s	1.114	0.418	0.323
Tiny	1h 6m	1.161	0.657	0.395	0h 34m	1.167	0.654	0.238	3s	1.144	0.492	0.306

We computed a measure of the motion reduction given as $MR \triangleq \operatorname{median}_{i \in \{1,...,N\}} \frac{\|\hat{\boldsymbol{d}}_i - \boldsymbol{d}_i\|_2}{\|\boldsymbol{d}_i\|_2}$. We note that in most cases, we achieve more than 90% motion reduction by the CS step - see Table 2. We have included a few videos in the `supplemental material` for the visual comparison of the estimated MVF w.r.t. the ground truth MVF.

Effect of number of frames T**:** In the absence of attenuation, a large T helps improve the performance of our algorithm, due to better Fourier sparsity and better approximation of the COT. In practice, we observed on real datasets that just 100 frames were sufficient to yield good reconstruction results. Further increase in T had an insignificant impact on the result quality. A graph showing the effect of T on reconstruction from real sequences is shown in Fig. 2.
Effect of number of tracked points N**:** The number and accuracy of DTs affects the performance of the CS algo-

Dataset	N	MR	σ_{motion}
Cartoon (Real1)	1029	94.11%	7.42
Checker (Real1)	3149	85.25%	8.5
Dices (Real1)	2230	91.9%	7.75
Bricks (Real1)	1300	87.38%	7.42
Elephant (Real1)	3670	97.7%	7.34
Eye (Real1)	1647	81.66%	7.84
Math (Real1)	2309	96.12%	5.64
BlueTiles (Synth.)	2192	94.77%	5.71
BrickWall (Synth.)	3134	94.57%	8.68
Vision (Synth.)	5266	93.49%	6.77
HandWritten (Synth.)	3789	95.82%	4.33
Middle (Real2)	785	96.34%	5.65
Small (Real2)	993	97.26%	4.22
Tiny (Real2)	155	87.84%	5.03

Table 2. #salient points N, motion reduction MR, and σ_{motion} for different videos

rithm. The number of DTs varied across datasets, depending on the number of available salient feature points, but was always less than $0.03N_xN_y$. The slightly lower performance of CS on the 'Tiny' sequence for example (see Table 1) is due to the small number of available salient points, less than $0.002N_xN_y$ - see Table 2. A graph showing the positive impact of increase in the number of good quality tracks (upto a point, beyond which the performance saturates) is shown in Fig. 2. We note that we have ensured good quality of the trajectories for further stages of our algorithms, as mentioned in Sec. 2.3.1. We considered *global sparsity in this work, as opposed to sparsity of small spatial or spatio-temporal patches*, since there could exist many patches without any salient points.

4. Conclusion

We have presented two methods for correction of refractive deformations due to a wavy water surface, one based on a novel application of CS for interpolating MVFs starting from a small set of salient PTs (and their DTs), and the other based on polynomial image expansions. In both cases, we obtain results superior to the state of the art at low computational cost. Avenues for future work include (1) extending the CS algorithm to handle moving objects; (2) studying the effect of depth variation, perspective projection or wave attenuation on the results of our algorithms; and (3) exploring MVF sparsity in other bases instead of the DFT.
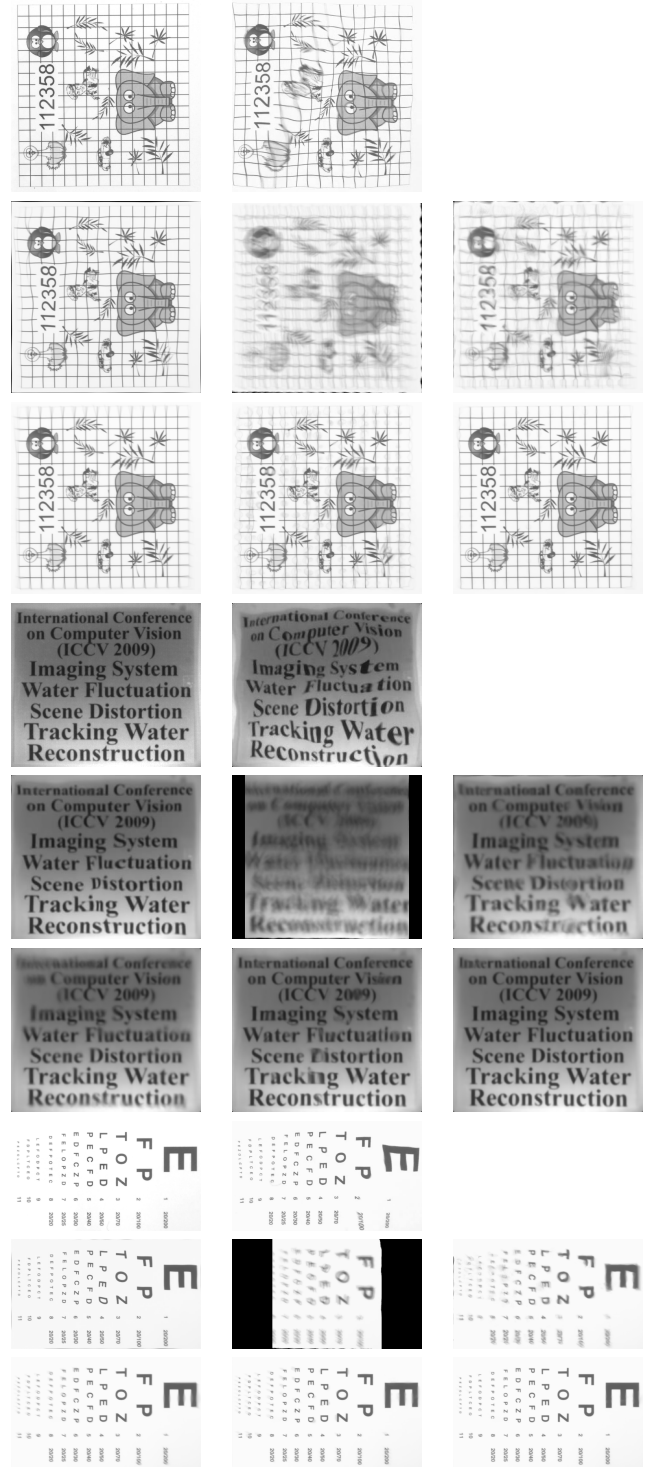
Figure 3. Left to right, top to bottom order in each of the 3 groups of images: ground truth, distorted sample frame; mean frame restored by SBR-RPCA [20], DL [18], LWB [29]; and by CS, PEOF, CS-PEOF. Zoom into pdf for better view. See `supplemental material` for more results. Notice geometric distortions in other methods unlike with our methods. The three groups are for 'Elephant' (Real1), 'Middle' (Real2) and 'Eye' (Real1)

References

[1] Github Repository. https://github.com/jeringeo/CompressiveFlows. 1

[2] MATLAB implementation for KLT tracker. https://in.mathworks.com/help/vision/ref/vision.pointtracker-system-object.html. 3

[3] Project Page. https://www.cse.iitb.ac.in/~ajitvr/publications.html. 1

[4] YALL1: Your algorithms for L1. http://yall1.blogs.rice.edu/. 5

[5] Marina Alterman, Yoav Y. Schechner, Pietro Perona, and Joseph Shamir. Detecting motion through dynamic refraction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):245–251, 2013. 1

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008. 3

[7] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, 2011. 1, 6

[8] Emmanuel J. Candès, Justin Romberg, and Terence Tao. Robust uncertainty prin- ciples: Exact signal reconstruction from highly incomplete. *IEEE Trans. on Information Theory*, 2006. 4

[9] Arturo Donate and Eraldo Ribeiro. Improved reconstruction of images distorted by water waves. In *Advances in Computer Graphics and Computer Vision*, 2007. 1

[10] Weisheng Dong, Guangming Shi, Xiaocheng Hu, and Yi Ma. Nonlocal sparse and low-rank regularization for optical flow estimation. *IEEE Transactions on Image Processing*, 23(10), 2014. 4

[11] Alexey Dosovitskiy et al. Flownet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015. 5

[12] Alexei A. Efros, Volkan Isler, Jianbo Shi, and Mirko Visontai. Seeing through water. In *NIPS*, pages 393–400, 2004. 1

[13] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370, 2003. 2, 4

[14] John D. Fenton and Michele Marie Rienecker. A fourier method for solving nonlinear water-wave problems: application to solitary-wave interactions. *Journal of Fluid Mechanics*, 118:411–443, 1982. 3

[15] Kalyan Kumar Halder, Manoranjan Paul, Murat Tahtali, Sreenatha G. Anavatti, and Manzur Murshed. Correction of geometrically distorted underwater images using shift map analysis. *J. Opt. Soc. Am. A*, 34(4):666–673, Apr 2017. 1, 5

[16] Kui Jia, Xiaogang Wang, and Xiaoou Tang. Optical flow estimation using learned sparse model. In *CVPR*, 2011. 4

[17] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *ICCV*, 2011. 3

[18] Zhengqin Li, Zak Murez, David Kriegman, Ravi Ramamoorthi, and Manmohan Chandraker. Learning to see through turbulent water. In *WACV*, pages 512–520, 2018. 1, 2, 5, 6, 7, 8

[19] Hiroshi Murase. Surface shape reconstruction of a nonrigid transport object using refraction and motion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(10):1045–1052, 1992. 1, 2, 3, 5

[20] Omar Oreifej, Guang Shu, Teresa Pace, and Mubarak Shah. A two-stage reconstruction approach for seeing through water. In *CVPR*, pages 1153–1160, 2011. 1, 2, 5, 7, 8

[21] Yiming Qian, Yinqiang Zheng, Minglun Gong, and Yee-Hong Yang. Simultaneous 3d reconstruction for water surface and underwater scene. In *ECCV*, pages 776–792, 2018. 2

[22] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015. 4, 5

[23] Moshe Rosenfeld. Utilization of fourier decomposition for analyzing time-periodic flows. *Computers and Fluids*, 24(4):349 – 368, 1995. 3

[24] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, 2010. 3

[25] Karthik Seemakurthy and Ambasamudram Narayanan Rajagopalan. Deskewing of underwater images. *IEEE Trans. Image Processing*, 24(3):1046–1059, 2015. 1, 2, 5

[26] Xiaohui Shen and Ying Wu. Sparsity model for robust optical flow estimation at motion discontinuities. In *CVPR*, 2010. 4

[27] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, pages 118–126, 2015. 3

[28] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 5

[29] Yuandong Tian and Srinivasa G. Narasimhan. Seeing through water: Image restoration using model-based tracking. In *ICCV*, pages 2303–2310, 2009. 1, 2, 5, 7, 8

[30] Yuandong Tian and Srinivasa G. Narasimhan. Globally optimal estimation of nonrigid image distortion. *International Journal of Computer Vision*, 98(3):279–302, 2012. 2

[31] Yuandong Tian and Srinivasa G. Narasimhan. Theory and practice of hierarchical data-driven descent for optimal deformation estimation. *International Journal of Computer Vision*, 115(1):44–67, 2015. 2

[32] D. G. Turlaev and Lev S. Dolin. On observing underwater objects through a wavy water surface: A new algorithm for image correction and laboratory experiment. *Izvestiya Atmosph. Ocean. Phys.*, 49(3):339345, 2013. 1, 2

[33] Zhou Wang and Alan C. Bovik. Mean squared error: Love it or leave it? A new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009. 5

[34] Rachel Ward. Compressed sensing with cross validation. *IEEE Trans. Inf. Theor.*, 55(12):5773–5782, 2009. 4

[35] Zhiying Wen, Donald Fraser, and Andrew Lambert. Bicoherence: a new lucky region technique in anisoplanatic image restoration. *Appl. Opt.*, 48(32):6111–6119, 2009. 1

[36] Zhiying Wen, Andrew Lambert, Donald Fraser, and Hong-
     dong Li. Bispectral analysis and recovery of images distorted
     by a moving water surface. *Appl. Opt.*, 49(33):6376–6384,
     2010. 1, 4

[37] Karen Willcox and Alexandre Megretski. *Real-Time PDE-
     Constrained Optimization*, chapter Fourier Model Reduction
     for Large-Scale Applications in Computational Fluid Dy-
     namics. SIAM, 2007. 3