Published in Image Processing On Line on YYYY–MM–DD.
ISSN 2105–1232 © YYYY IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.YYYY.XXXXXXXX

PREPRINT (DRAFT: with some missing images)

Atomic Models of Video Turbulence

Enric Meinhardt-Llopis

Abstract

We describe some algorithms to simulate turbulent videos of static objects. The effects are visually similar to those of taking a video of an object which is underwater or behind hot air. The proposed algorithms are not based on any physical models of turbulence but on the visual effect that the turbulence produces. The main application is to test methods for correcting the turbulence in a controlled setting.

1 Introduction

Turbulence is a well-known and widely studied problem in image processing [14]. This problem is treated by somewhat independent communities in astronomical imaging [13, 2], long-distance surveillance over a hot terrain [9, 11] and underwater imaging [5, 1, 7]. Typically, these studies begin by describing a model of image formation behind a physically realistic turbulent fluid, and then set up the reconstruction of the base (non-turbulent) image as an inverse problem. In this article we focus, not on the physical models, but on the visual effects produced by turbulence. Thus, we study turbulence as an arbitrary operation that takes an image obtained in ideal conditions and produces a distorted version of it.

On Section 2 we introduce five different models of turbulence in a continuous setting. On Section 3 we describe the algorithms needed to apply these models in discrete images. On Section 4 we show some images from real-world turbulent videos which can be explained by each of the proposed models.

2 Models of turbulent images

Let us describe five models of the formation of turbulent images. We are not interested in the physical processes that produce these images, but on the transformation that maps an image without turbulence to a sequence of images with turbulence. More specifically, we want to express these transformation as a combination of common image-processing operations, even if they do not correspond to any physical phenomenon.

In this section we consider gray-level images as functions $I : \mathbf{R}^2 \to \mathbf{R}$. Turbulence is modeled as an operation that takes the image I and produces a sequence of "turbulent" images $I_1, I_2, \ldots$. The idea is that each I_i is a distorted version of I.

2.1 Correlated additive noise

The simplest distortion that we can consider is additive noise:

$$I_i(\mathbf{x}) = I(\mathbf{x}) + u_i(\mathbf{x})$$

where $u_i(\mathbf{x})$ is a sequence of random functions whose expected value is point-wise zero. The random variables $u_i(\mathbf{x})$ are assumed to be identically distributed. In this setting, assuming that the variables have finite variance, the clean image I can be recovered from the turbulent sequence I_i by taking averages:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} I_i(\mathbf{x}) = I(\mathbf{x}) \qquad \forall \mathbf{x} \in \Omega.$$

This model includes white noise, where the values of $u_i(x)$ are assumed to be independent random variables when varying i or x. However, from the point of view of the simulation of turbulence, it is much more interesting to consider the case where these random variables are highly dependent. For example: $u_i(x)$ may be a smooth function that changes slowly as i advances. A real-world example of this kind of turbulence can be the caustics projected on underwater objects when the light of the sun traverses the turbulent surface of water.

2.2 Blur

A different model of turbulence is given by *blur*, i.e. the convolution with a positive kernel, which is a good model of the effects of turbulence for long-exposure images:

$$I_i(\mathbf{x}) = (k * I)(\mathbf{x}).$$

An appropriate kernel for atmospheric turbulence is the Fried kernel [4, 6]. A simpler kernel which is numerically similar is given by the multi-variate Laplace distribution [8]:

$$k_\beta(\mathbf{x}) = \frac{1}{2\pi\beta^2} \exp \frac{-|\mathbf{x}|}{\beta}$$

The parameter β controls the strength of the turbulence.

When the exact form of the kernel is known, the clean image I can be recovered from $k * I$ by a regularized deconvolution technique such as Wiener filtering. When there is an unknown parameter in the kernel (e.g., β), the value of this parameter can be chosen as the one which maximizes the total variation of the de-convolved images [10].

2.3 Geometric Deformation

Geometric deformation is modelled by the pull-back of the base image I by a sequence of random vector fields $\mathbf{u}_i$. Formally:

$$I_i(\mathbf{x}) = I(\mathbf{x} + \mathbf{u}_i(\mathbf{x}))$$

Here, as above, we suppose that the two components of $\mathbf{u}_i(\mathbf{x})$ are identically distributed random variables with zero mean. Notice that if we want these vector fields to be smooth, then these random variables can not be independent, because their values at nearby points must be similar.

The visual effect of these deformations is governed by three things:

1. The probability distribution $k(\mathbf{u})$ of the random variable $\mathbf{u}_i(\mathbf{x})$, for a fixed i and $\mathbf{x}$. A large variance of this random variable corresponds to large motions of the pixels, thus a large *spread* produced by the turbulence.

2. The amount of dependence between $\mathbf{u}_i(\mathbf{x})$ and $\mathbf{u}_i(\mathbf{y})$ when $\mathbf{x}$ and $\mathbf{y}$ are close. The larger this dependence, the smoother the vector fields, so that the *grain* of the turbulence is larger.

3. The amount of dependence between $\mathbf{u}_i(\mathbf{x})$ and $\mathbf{u}_j(\mathbf{x})$, when i and j are close. The smaller this dependence, the faster the vector fields are allowed to change, so that this controls the *speed* of the turbulence.

These three effects, spread, grain and speed, agree with visual features observed in videos of turbulence (see section 4 below). Notice that this interpretation is a simplification, as there are many more features that can be simulated. For example, some sequences show an apparent motion of fluid in a certain direction. This can be simulated with this model by a field $\mathbf{u}(\mathbf{x}, i)$ which is highly correlated in one spatio-temporal direction, and relatively independent in the others.

The average of the video frames:

$$I_*(\mathbf{x}) := \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} I(\mathbf{x} + \mathbf{u}_i(\mathbf{x}))$$

gives an estimation of the expected value of $I(\mathbf{x} + \mathbf{u}_i(\mathbf{x}))$ at each pixel $\mathbf{x}$. Notice that even if $\mathbb{E}[\mathbf{u}_i(\mathbf{x})] = \mathbf{0}$, this expected value need not be $I(\mathbf{x})$. The identity $\mathbb{E}[I(\mathbf{x} + \mathbf{u}_i(\mathbf{x}))] = I(\mathbf{x})$ holds only when I is an affine function. When I is convex, by Jensen's inequality we have $\mathbb{E}[I(\mathbf{x} + \mathbf{u}_i(\mathbf{x}))] \geq I(\mathbf{x})$. When I is concave the reverse inequality holds. In general, the expected value I_* is a smooth version of the image I, where local minima get a higher value and local maxima get a lower value.

The expected value of $I_i(\mathbf{x})$ can be effectively computed as

$$\mathbb{E}[I(\mathbf{x} + \mathbf{u}_i(\mathbf{x}))] = \int_{\mathbf{R}^2} I(\mathbf{x} + \mathbf{y}) k(\mathbf{y}) \mathrm{d}\mathbf{y}$$

which is precisely the convolution of the image I by the positive kernel k. This observation means that blur can be simulated as the temporal average of geometrical deformation, and the blur kernel is precisely the point-wise distribution of the deformation vector fields. This is indeed a realistic model of the formation of blur due to turbulence, because the temporal averaging corresponds to the integration performed by the camera captors during the exposure time.

2.4 Mass transportation

Mass transportation uses almost the same formula as geometric deformation, but enforcing local mass conservation:

$$\bar{I}_i(\mathbf{x}) = \det(\mathrm{Id} + D\mathbf{u}_i)\bar{I}(\mathbf{x} + \mathbf{u}_i(\mathbf{x}))$$

In this model, the color of a pixel changes after the deformation is applied. A possible physical interpretation of this model is an object composed of discrete luminous microscopic dots, and the turbulence changes the apparent position of these dots (but not their size or their brightness). Then, the apparent brightness of the object varies according to the point-wise variation in dot density. We do not know whether this effect can be described by a realistic model in terms of deformed paths of light rays. The interpretation in terms of luminous dots is defective because the dots would change in size if they were deformed. A possible toy model for the formation of this turbulence is given by a pinhole camera where the turbulence acts *after* the discrete light rays have passed through the pinhole.

In any case, this model can be used to simulate some observed effects of turbulence, where the turbulent flow is visible over large objects of constant color as local intensity changes. Geometric deformation alone can not explain this effect.

It is important take into account the *units* of intensity when applying the mass transportation model, for they should be linear in the number of photons received at a point. Typically, the

intensities encoded in images use a perceptually uniform scale which, by Weber's law, corresponds to the logarithm of the photon count. Thus, the correct mass transportation method shall use the following contrast change (modulus an arbitrary affine rescaling of I and $\bar{I}$):

$$I = \log_\lambda(\bar{I})$$

before applying the formula above. Here the base of the logarithm λ and the scaling factors are parameters of the method. The mass transportation operation then becomes:

$$I_i(\mathbf{x}) := \log_\lambda \left(\det(\mathrm{Id} + D\mathbf{u}_i)\right) + I(\mathbf{x} + \mathbf{u}_i(\mathbf{x})).$$

Notice that as $\lambda \to \infty$, the first term is less relevant and the transformation approximates a simple geometric deformation. That way, geometric deformation can be understood as a particular case of mass transportation.

The discussion above is based on the interpretation that the vector field $\mathbf{u}_i(\mathbf{x})$ represent the *displacement* of the points $\mathbf{x}$ between two frames. A different but common interpretation is to a assume a continuous video (e.g., use t as a continuous time index instead of a discrete index i). In that case, the vector field can be interpreted as the *velocity* of the points, and the expression $I(\mathbf{x} + \mathbf{u}) - I(\mathbf{x})$ is a first order finite difference approximation to $u \cdot \nabla I$. Under this alternative interpretation, the Jacobian $\det(\mathrm{Id} + D\mathbf{u}_i)$ is replaced by its first-order approximation $1 + \mathrm{div}(\mathbf{u})$.

2.5 Phase screens

The phase screen model acts in the frequency domain. Let $\vartheta_i(\boldsymbol{\xi})$ be a sequence of random functions with values in the interval $[0, 2\pi)$. Such a function is called a phase screen. The action of such a phase screen on the positive-valued image I is given by the following formula:

$$I_i(\mathbf{x}) = \left| \mathcal{F}\left(\mathcal{F}^{-1}\left(\sqrt{I}\right) e^{i\vartheta_i(\boldsymbol{\xi})}\right)\right|^2$$

Here $\mathcal{F}$ denotes the Fourier transform in 2D. Notice that if $\vartheta_i(\boldsymbol{\xi}) = 0$ then $I_i = I$. The distribution of $\vartheta_i(\boldsymbol{\xi})$ around 0 determines the visual effect of the phase screen.

This phase screen model is widely used in the astronomical community. It comes from a physical model of image formation in a telescope after light rays traverse a turbulent atmosphere. Without turbulence, the parallel light rays that reach the lens of the telescope arrive at a point of the captor in the same phase. The function $\vartheta_i(\boldsymbol{\xi})$ says how the phase of the light rays is changed, due to atmospheric turbulence, at each point in the lens. The resulting image in the captor is thus distorted.

Notice that, barring a quadratic contrast change, the phase screen is formally a time-varying "blur" where $k = \mathcal{F}(e^{i\vartheta})$ is a linear kernel of flat frequency response. Typically the function k is oscillating around a slightly off-zero center (thus, the images are visually translated by a constant, time-varying small vector). The main point of interest of this model is that the statistics of the phase screen distribution are well known according to models of atmospheric turbulence. Its weakest point is that it is a shift-invariant operator, so that it applies the same distortion to all the points in the image domain.

The particular form of the Fried kernel is derived by taking averages of phase screens with realistic statistics [3].

3 Simulation of the models

On the previous Section we defined several turbulence operators for images with a continuous domain. This simplifies the exposition, but in practice, we need to work with discrete-domain images $I : \mathbf{Z}^2 \to \mathbf{R}$. On this Section we explain how to adapt the operations to the discrete setting. First we introduce the basic building blocks of the algorithms, and afterwards we explain how to effectively simulate the proposed models.

3.1 Warp an image by a vector field

The pull-back, or warp, of a function $I : \mathbf{R}^2 \to \mathbf{R}$ by a vector field $\mathbf{x} \mapsto \mathbf{x} + \mathbf{u}(\mathbf{x})$ is defined as the function

$$\mathbf{u}^* I(\mathbf{x}) := I(\mathbf{x} + \mathbf{u}(\mathbf{x})).$$

To adapt this operation to the discrete setting, we must take into account that the domain of the functions I, $\mathbf{u}$ and $\mathbf{u}^* I$ is now $\mathbf{Z}^2$ instead of $\mathbf{R}^2$. If we denote by $\mathbf{i} = (i, j)$ an arbitrary point in $\mathbf{Z}^2$, what we really want to compute is

$$\mathbf{u}^* I(\mathbf{i}) := I(\mathbf{i} + \mathbf{u}(\mathbf{i})).$$

However, this computation is impossible because the point $\mathbf{i} + \mathbf{u}(\mathbf{i})$ need not belong to $\mathbf{Z}^2$, and the function I is now defined only over $\mathbf{Z}^2$. This is the problem of *irregular sampling*, which is a difficult, and well known, problem in image processing. As particular cases, it contains: zoom-in, which is difficult because it relies on an interpolation method; zoom-out, which is difficult because it relies on an smoothing method. Irregular sampling is more difficult because the same vector field may produce zoom-in and zoom-out at different parts of the image, or even a combination of both at different directions around the same point.

For our purposes, we forget momentarily about the difficulties of irregular sampling and use a simple interpolation model for evaluating $I(\mathbf{i} + \mathbf{u}(\mathbf{i}))$. Namely, we use *bicubic* interpolation, which is a piecewise third-degree polynomial that interpolates the values of I on a 4×4 neighborhood of the point $\lfloor \mathbf{i} + \mathbf{u}(\mathbf{i}) \rfloor$. This is not a good solution for the general problem of irregular sampling. However, since in our simulations the vector fields $\mathbf{u}$ will tend to be very smooth, they are locally approximately translations, with no zoom-in or out. Thus the artifacts of aliasing or blurring do not appear. In any case, if a greater precision is desired, we can always work with higher resolution images, which are then zoomed-out to the desired target resolution. This should eliminate any aliasing artifacts due to the irregular sampling.

In the case of mass transportation we have to multiply the warped image by the Jacobian of the transformation:

$$\mathbf{u}^\star I(\mathbf{i}) := |\mathrm{Id} + D\mathbf{u}| \;\; I(\mathbf{i} + \mathbf{u}(\mathbf{i})).$$

here the four partial derivatives in $D\mathbf{u}$ can be computed by centered finite differences around $\mathbf{i}$.

3.2 Invert a vector field

Let us suppose that a vector field $\mathbf{x} \mapsto \mathbf{x} + \mathbf{u}(\mathbf{x})$ maps the points of an image A to the corresponding points of an image B. The warping operation described above reconstructs the image A from the values of the image B, but not the other way round. If we want to map the values from A to B, a possibility is to invert the flow. That is, find a vector field $\mathbf{v}$ such that for every $\mathbf{x}$:

$$\mathbf{u}(\mathbf{x}) + \mathbf{v}(\mathbf{x} + \mathbf{u}(\mathbf{x})) = 0$$

or

$$\mathbf{v}(\mathbf{x}) + \mathbf{u}(\mathbf{x} + \mathbf{v}(\mathbf{x})) = 0$$

A reasonable approximation is $\mathbf{v}(\mathbf{x}) := -\mathbf{u}(\mathbf{x})$, which is good enough when $\mathbf{u}$ is very smooth. However, for higher precision we need to solve one of the equations above. Fixing $\mathbf{x}$, the second equation can be rewritten as a fixed point equation $\mathbf{F}(\mathbf{y}) = \mathbf{y}$ with

$$\mathbf{F}(\mathbf{y}) := -\mathbf{u}(\mathbf{x} + \mathbf{y})$$

And this function is Lipschitz of constant

$$L \leq |\mathrm{div}(\mathbf{u}(\mathbf{x}))|.$$

Thus, the recursive iteration of $\mathbf{F}$ converges to the desired $\mathbf{v}(\mathbf{x})$ at the points $\mathbf{x}$ such that the absolute value of the divergence of $\mathbf{u}$ is less than one. When the divergence is larger than one in absolute value, there may be none or more than one solutions. This is consistent with the intuition: when the divergence is too large, the vector field $\mathbf{u}$ leaves uncovered holes on the image B domain (in which case the inverse does not exist). When the divergence is too small, the vector field is not injective (in which case the inverse is multiple-valued).

3.3 Generate a random time-varying vector field

To generate a random function or vector field we need a random number generator. The random number generator produces uniform samples, but those can be converted to samples of any probability distribution (e.g., normal, Laplacian, exponential) by applying an appropriate function to the samples. The following list explains how to obtain samples of some common probability distributions, starting from uniform samples.

- If X_1 and X_2 are independent uniform random variables on $[0, 1]$, then the two variables

$$\begin{aligned} Y_1 &= \sqrt{-2\log(X_1)}\cos(2\pi X_2) \\ Y_2 &= \sqrt{-2\log(X_1)}\sin(2\pi X_2) \end{aligned}$$

  are independent normal variables of mean 0 and variance 1.

- If X_1 and X_2 are independent uniform on $[0, 1]$, then $Y = \log(X_1/X_2)$ is Laplace
- If X_1 and X_2 are independent normal then $Y = X_1/X_2$ is Cauchy
- If X is Laplace then $|X|$ is exponential
- If X_1, X_2, X_3, X_4 are independent normal then $Y = X_1X_2 - X_3X_4$ is Laplace [12].

The last relation is especially important for us since it will be used to produce smooth random fields with a pointwise Laplace distribution. The random samples obtained by these formulas at each point will be independent, and they are only by useful as white noise. To obtain interesting smooth random fields we can convolve the white noise by a positive Gaussian kernel of spatiotemporal correlation matrix Σ. The pseudocode below explains the method:

Algorithm: Generate a smooth random function

Input: desired dimensions (W, H, N) and a positive semidefinite matrix Σ

Output: an video I of correlated noise containing N frames of size $W \times H$

1. For each (i, j, t) in the image domain, set $n(i, j, t) := X$, where X is a random number generator
2. Compute the desired multivariate normal kernel:

$$k(i, j, t) := \frac{1}{\alpha}\exp\left(-(i, j, t)\Sigma\begin{pmatrix} i \\ j \\ t \end{pmatrix}\right)$$

3. Compute the convolution $I = k * n$.

This algorithm is crucial for most of the proposed simulators and it is important to understand it well. The first important point is that, when Σ is wide enough, the distribution of the random numbers X in step (1.) is irrelevant, as long as they have finite moments. Indeed, the convolution step computes a weighted sum of several samples of this distribution, and by the central limit theorem, the distribution of this sum is very close to a normal distribution. If the variables X were already normal, then the pixels of I are also normal random variables. This observation does not depend on the fact that the convolution kernel in step (2.) is normal distribution: it works for any positive convolution kernel.

The second important point is that the convolution in step (2.) can be computed efficiently, even for kernels with very large support, using the 3D FFT.

The third important observation is how the entries of the matrix Σ affect visually the result of the output functions. Let us use this notation:

$$\Sigma = \begin{pmatrix} a & b & p \\ b & c & q \\ p & q & r \end{pmatrix}$$

The simplest case is when Σ is a multiple of the identity $\Sigma = \lambda \mathrm{Id}$. When λ is close to 0, the kernel is very narrow and the resulting variables are almost independent, almost like in white noise (which corresponds to $\lambda = 0$). When λ grows to infinity, the kernel becomes almost a constant, and the resulting variables take the same value (which corresponds to the maximal possible dependency).

When $p = q = r = 0$, this means that there is no temporal correlation: even if the functions are smooth on a given frame, they are independent from frame to frame. If $a = c$ and $b = 0$, this means that the kernel is spatially isotropic so that the function has "isotropic" grain. If $b = 0$ and a is much larger than c, this means that the kernel is elongated horizontally, and so is the grain of the resulting random function. In general, the matrix Σ represents an ellipsoid (whose axes are given by its eigenvectors and eigenvalues), which determines the grain of the function. If this function is used to simulate a turbulence, this will be the grain of the turbulence.

A major problem with the method above is that it does not allow to control directly the distribution of the pixel values, which is always normal by the central limit theorem (or Cauchy, if the initial samples were Cauchy, or any other stable distribution). A trick to obtain other distributions consists in applying an appropriate transformation to the given normal samples so that the resulting samples have the desired target distribution. For example, if the variables X_1, X_2, X_3 and X_4 are independent normal variables, then the variable $Y = X_1X_2 - X_3X_4$ has the Laplace distribution. Thus, the following modified algorithm produces a random smooth function whose pixels have pointwise a Laplace distribution:

Algorithm: Generate a smooth random function with the Laplace distribution

Input: desired dimensions (W, H, N) and a positive semidefinite matrix Σ

Output: an video I of correlated noise containing N frames of size $W \times H$

1. For each (i, j, t) in the image domain and $s = 1, 2, 3, 4$, set $n_s(i, j, t) := X$, where X is a normal random number generator

2. Compute the desired multivariate normal kernel:

$$k(i, j, t) := \frac{1}{\alpha} \exp\left(-(i, j, t)\Sigma \begin{pmatrix} i \\ j \\ t \end{pmatrix}\right)$$

3. Compute the convolutions $I_s = k * n_s$ for $s = 1, 2, 3, 4$.

4. Compute the output image $I = n_1 n_2 - n_3 n_4$.

This method is not very flexible because it requires an ad-hoc and difficult to find formula for each target distribution.

3.4 Simulate deformation, transportation and caustics

We have explained above how to warp an image according to a vector field, with or without mass conservation. We have also explained how to generate random vector fields with various tunable statistical properties. In order to generate turbulent videos we only have to combine these operations.

The following pseudocode describes an algorithm to compute three turbulent videos of different kinds using the same random vector fields.

Algorithm: Compute turbulent versions of a clean image

Input: An image I, the desired number of frames N, a positive semidefinite matrix Σ controlling the "grain" and "speed" of the turbulence, a "strength" of the turbulence μ, and various internal normalization parameters such as λ.

Output: three videos I_i^{geom}, I_i^{mass} and I_i^{caust} of N frames with different turbulence effects.

1. For each (i, j, t) in the video domain, and $s = 1, 2$ set $u_s(i, j, t) = X$, where X is a normal random number generator
2. Compute the desired multivariate normal kernel:

$$k(i, j, t) := \frac{1}{\alpha} \exp\left(-(i, j, t)\Sigma \begin{pmatrix} i \\ j \\ t \end{pmatrix}\right)$$

3. Compute the convolutions $u = k * n_1$ and $v = k * n_2$, and put $\mathbf{u} = (\mu u, \mu v)$.
4. Compute $I_t^{geom}(\mathbf{i}) := I(\mathbf{i} + \mathbf{u}(\mathbf{i}, t))$
5. Compute $I_t^{mass}(\mathbf{i}) := \log_\lambda |\mathrm{Id} + D\mathbf{u}(\mathbf{i}, t)| \; + \; I_t^{geom}(\mathbf{i})$
6. Compute $I_t^{caust}(\mathbf{i}) := I(\mathbf{i}) + \mu \log \left| \frac{u(\mathbf{i},t)}{v(\mathbf{i},t)} \right|$

$I(\mathbf{x})$ $I_1^{geom}(\mathbf{x})$ $I_1^{mass}(\mathbf{x})$ $I_1^{caust}(\mathbf{x})$

Figure 1: An image I and three distorted versions by the same vector field.

4 Identification of the models in real-world turbulence

On Figures 2-6 we display some video frames taken with an underwater camera. These images show similar distortions as the ones that we have simulated above. It is difficult to cleanly separate the effects (e.g., a bright small spot that is deformed onto a dark larger spot can be explained both by mass transportation and by blur). However, it is clear that there are a few essentially different deformations, since some of the images are blurry (3), some of the images are sharp but deformed (4), and some of the images have non-uniform lighting (2).

In the case of sharp images, we can estimate the spectral profile of the blur kernel by comparing the spectrum of one sharp image (which is roughly invariant to geometric deformations), with the spectrum of the average of all video frames.

Figure 2: A real-world example of correlated additive noise due to turbulence. Notice that the shape of the objects does not change, but that there is a grainy irregular lighting, visible over the flat areas.

Figure 3: A real-world example of blur due to turbulence. The turbulence on this video is very fast, and this results in blur because the acquisition integrates many different images.

Figure 4: A real-world example of geometric deformation due to turbulence. Notice that the boundaries of objects are sharp and their color does not change.

Figure 5: A possible real-world example of mass transportation due to turbulence. Notice that some light and thin objects become darker when they are enlarged. This may be also explained as an effect of blur.

Figure 6: A possible real-world example of phase screen due to turbulence. A punctual object is duplicated near the center of the image. This effect can not be explained by the other models, but is a common occurrence in phase screen distortions.

References

[1] L. Dolin. *Theory of imaging through wavy sea surface*. Russian Academy of Sciences Institute of Applied Physics, 2006.

[2] D.L. Fried. Statistics of a geometric representation of wavefront distortion. *JOSA*, 55(11):1427–1431, 1965.

[3] D.L. Fried. Optical resolution through a randomly inhomogeneous medium for very long and very short exposures. *JOSA*, 56(10):1372–1379, 1966.

[4] D.L. Fried. Probability of getting a lucky short-exposure image through turbulence. *JOSA*, 68(12):1651–1657, 1978.

[5] G.D. Gilbert and R.C. Honey. Optical turbulence in the sea. In *Underwater photo-optical instrumentation applications*, pages 49–56. International Society for Optics and Photonics, 1971.

[6] J. Gilles and S. Osher. Fried deconvolution. Technical report, CAM Technical Report 11-62, UCLA Department of Mathematics, 2011.

[7] W. Hou, D. Gray, A. Weidemann, and R. Arnone. Comparison and validation of point spread models for imaging in natural waters. Technical report, DTIC Document, 2008.

[8] S. Kotz, T.J. Kozubowski, and K. Podgorski. *The laplace distribution and generalizations: A revisit with applications to communications, economics, engineering, and finance*. Number 183. Birkhauser, 2001.

[9] Y. Mao and J. Gilles. Non rigid geometric distortions correction-application to atmospheric turbulence stabilization. *Inverse Problems and Imaging*, 6(3):531–546, 2012.

[10] M. Micheli. The centroid method for imaging through turbulence. *arXiv preprint arXiv:1206.3925*, 2012.

[11] M. Micheli, Y. Lou, and A.L. Bertozzi. A dynamic texture model for imaging through turbulence. *UCLA CAM report*, pages 12–01, 2012.

[12] H. Nyquist, SO Rice, and J. Riordan. The distribution of random determinants. *Quarterly of Applied Mathematics*, 12(2):97–104, 1954.

[13] F. Roddier. The effects of atmospheric turbulence in optical astronomy. *Progress in optics.*, 19:281–376, 1981.

[14] M.C. Roggemann and B.M. Welsh. *Imaging through turbulence*. CRC, 1996.