Published in Image Processing On Line on 2015–08–19.
Submitted on 2013–12–14, accepted on 2015–03–23.
ISSN 2105–1232 © 2015 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials, software, datasets and online demo at
https://doi.org/10.5201/ipol.2015.108

# The Flutter Shutter Code Calculator

Y. Tendero

Institut Mines-Télécom, Télécom ParisTech, CNRS - LTCI, Paris (France),
yohann.tendero@telecom-paristech.fr.

*Communicated by* Mauricio Delbracio *Demo edited by* Yohann Tendero

This IPOL article is related to a companion publication in the SIAM Journal on Imaging Sciences:
Y. Tendero and J.-M. Morel. "A Theory of Optimal Flutter Shutter For Probabilistic Velocity Models". *SIAM Journal on Imaging Sciences (submitted)*.

## Abstract

The goal of the flutter shutter is to make uniform motion blur invertible, by a "fluttering" shutter that opens and closes on a sequence of well chosen sub-intervals of the exposure time interval. In other words, the photon flux is modulated according to a well chosen sequence called flutter shutter code. This article provides a numerical method that computes optimal flutter shutter codes in terms of mean square error (MSE). We assume that the observed objects follow a known (or learned) random velocity distribution. In this paper, Gaussian and uniform velocity distributions are considered. Snapshots are also optimized taking the velocity distribution into account. For each velocity distribution, the gain of the optimal flutter shutter code with respect to the optimal snapshot in terms of MSE is computed. This symmetric optimization of the flutter shutter and of the snapshot allows to compare on an equal footing both solutions, i.e. camera designs. Optimal flutter shutter codes are demonstrated to improve substantially the MSE compared to classic (patented or not) codes. A numerical method that permits to perform a reverse engineering of any existing (patented or not) flutter shutter codes is also described and an implementation is given. In this case we give the underlying velocity distribution from which a given optimal flutter shutter code comes from. The combination of these two numerical methods furnishes a comprehensive study of the optimization of a flutter shutter that includes a forward and a backward numerical solution.

## Source Code

The C++ source code, version 2.0, is available from the article web page[1]. The documentation is included in the archive. Basic compilation and usage instructions are included in the README.txt file. The demo permits to compute optimal flutter shutter codes for (truncated) Gaussian and uniform probability velocity distribution. It computes the optimal snapshot, as well. In this case the demo provides the ideal exposure time, taking the velocity model into account. It also computes the gain in terms of MSE of the flutter shutter compared to the optimal snapshot. This comparison permits to decide the viability of the flutter shutter apparatus for any application.

[1]https://doi.org/10.5201/ipol.2015.108

## 1 Introduction

Digital cameras are devices that count the number of photons emitted by the observed landscape during a time span called exposure time. Due to the nature of photon emission the photon count is a Poisson random variable. This means that the more photons counted the least noisy the produced image is. If the scene being photographed moves during the exposition process, or if the scene is still but the camera moves, the resulting images are degraded by motion blur. As soon as the support of a uniform motion blur kernel exceeds two pixels the blur is not invertible[2]. Thus, when the camera and the landscape are in relative motion the aperture time of a classic camera must be reduced to guarantee an invertible motion blur kernel. A passive camera cannot artificially increase the photon count. Consequently, the image quality of a snapshot is limited. Obtaining longer exposure time without the effects of the motion blur can therefore be seen as one of the core problems of photography.

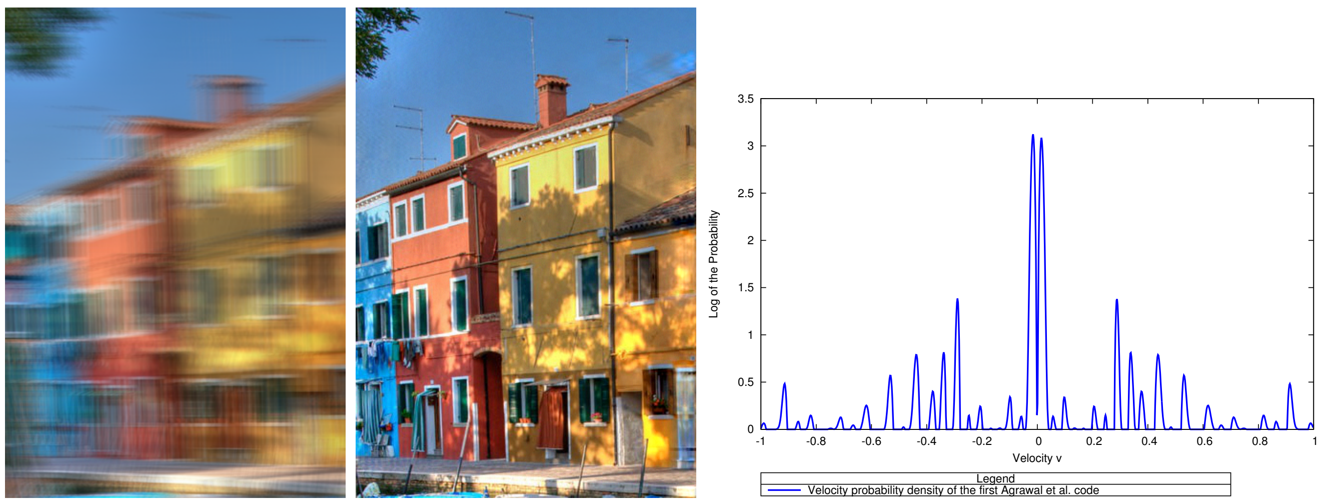


Figure 1: On the left panel: simulated observed (blurry and noisy) image using the Agrawal, Raskar et al. code published in [1]. The blur interval length is 52 pixels. Notice the stroboscopic effect of the flutter shutter apparatus. On the middle panel: the reconstructed image obtained by direct deconvolution. These images come from a peer-reviewed flutter shutter camera simulator [13]. On the right panel: the velocity probability density for which the Agrawal, Raskar et al. “near-optimal code” [11, p. 799] and patent application [12] is optimal. The $x$-axis is the motion (in signed pixels) and the $y$-axis represents the logarithm of the probability density ($\log(1 + \rho(v))$). The probability that $v = 0$ is overwhelming -recall that the $y$-axis is log scaled- and is small but nonzero for a (relatively) broad range of velocities.

A revolutionary alternative to classic photography was proposed in [1, 2, 3, 10, 12]. In [1, 2, 3, 10, 12] Agrawal, Raskar et al. attach to the camera a flutter shutter to get an invertible motion blur kernel. Numerically, the flutter shutter is described by a binary shutter sequence called flutter shutter code. This code gives the sub-interval of the exposure time where the photon flux is interrupted. The striking new fact is that if the flutter shutter code is well chosen, invertibility can be guaranteed for arbitrarily severe uniform motion blur as illustrated in Figure 1. As a byproduct, the exposure time can be as long as desired: many more photons are sensed by the camera. Viewed in that perspective, the flutter shutter looks like a magic solution that should equip all cameras. Yet, does that mean that one can decrease indefinitely the MSE by an increased exposure time, at no cost from

[2]The uniform motion model implies that the relative camera scene motion has a constant speed and follows a straight line.

the motion blur side? The answer is negative, as proved in [14] by Tendero et al. More precisely, given a landscape that moves in uniform translation at a known velocity $v$ the gain in terms of MSE with respect to an optimal snapshot cannot exceed a 1.17 factor. This 1.17 factor is significant but clearly not overwhelming.

In [15] Tendero et al. propose a mathematical framework that permits to optimize flutter shutter cameras beyond this bound in many realistic cases. This is possible provided the random velocity distribution of objects in the scene is known. Compared to [14] the setup is changed: the velocity is no more a known constant. Instead, in [15] the motion follows a random velocity distribution. Depending on this velocity distribution, the gain can be significant. Conversely, their theory permits to analyze a posteriori any existing flutter shutter strategy or code. A formula permits to decide if any existing code is optimal or not for some random velocity distribution. If the considered flutter shutter code is optimal, a formula permits to perform a reverse engineering of the code. Indeed, the formula reveals the probability density for which the given flutter shutter code is optimal.

This paper provides the implementation of the theory developed by Tendero et al. in [15]. Two velocity models are considered: truncated Gaussian and uniform distribution. For each, it computes the optimal codes, the optimal snapshot and compares the MSE of both solutions. In addition, this paper provides the implementation of the reverse engineering algorithm that, given a flutter code, computes the velocity distribution for which it is optimal. A glossary of notations is available in the Appendix (page 251).

## 2 The Flutter Shutter Formalism [14]

This section gives the whole flutter shutter formalism as it was developed by Tendero et al. in [14]. The exposition is self-contained.

### 2.1 Analog and Numerical Flutter Shutter Methods

Following the flutter shutter literature we assume that the relative camera scene motion is uniform. This implies that the relative camera-landscape motion can be associated with a one dimensional box kernel. The support of this kernel increases linearly with the exposure time $\Delta t$ and the velocity $v \in \mathbb{R}$ of the motion. If the exposure time is too long and the blur support exceeds two pixels, then the blur is no more invertible. In that case, the restoration process is an ill-posed problem [4]. The flutter shutter [1, 2, 3, 12, 10] (coded exposure) permits to ensure an invertible motion kernel for arbitrarily severe uniform motion blur. There are two different acquisition tools that implement a flutter shutter with a moving sensor (or landscape). The flutter shutter function can be implemented as an optical (temporally changing) filter. This filter controls the percentage of incoming photons allowed to travel to the sensor. The filtering function is generally assumed to be piecewise constant [1, 2, 3, 10, 12] with a flutter shutter code $(\alpha_k)_{k\in\{0,\dots,L-1\}}$, where $L$ is the length of the code. This setup, which corresponds to the initial technology of the inventors, is called analog flutter shutter.

A more flexible set up, the flutter shutter, is a mere temporal filter. In a nutshell, the camera takes a burst of $L$ images. The $k$-th elementary image is assigned a numerical gain $\alpha_k \in \mathbb{R}$. The observed image is obtained as the weighted sum of elementary images with weights $(\alpha_k)_{k\in\{0,\dots,L-1\}}$. According to [6, 9] an image sensor can have a duty ratio of nearly 100% (the duty ratio is the ratio of light integration time over readout, storage, reset times - that is the percentage of useful time). Thus, a sensor can integrate light without interruption. This means that the numerical flutter shutter, as it is described below, i.e. without “dead time” between two consecutive gains $\alpha_k$, is doable from a technological point of view. Notice that in both cases we have a flutter shutter code, but the formulas for the resulting image are not exactly the same, as illustrated in Table 1.

| Type of flutter shutter | Numerical flutter shutter | Analog flutter shutter |
|---|---|---|
| flutter shutter function $\alpha(t)$ | $\alpha(t) = \sum_{k=0}^{L-1} \alpha_k \mathbb{1}_{[k\Delta t,(k+1)\Delta t)}(t)$ (with $\alpha_k \in \mathbb{R}$ and $\Delta t > 0$) | $\alpha(t) = \sum_{k=0}^{L-1} \alpha_k \mathbb{1}_{[k\Delta t,(k+1)\Delta t)}(t)$ (with $\alpha_k \in [0,1]$ and $\Delta t > 0$) |
| Continuous flutter shutter gain function $\alpha(t)$ | $\alpha(t) \in L^2(\mathbb{R})$ | $\alpha(t) \in L^1(\mathbb{R})$, $\alpha(t) \in [0,1]$ |
| Observed samples $obs(n)$ | $obs(n) \sim \sum_{k=0}^{L-1} \alpha_k \mathcal{P}\left(\int_{k\Delta t}^{(k+1)\Delta t} u(n-vt)dt\right)$ | $obs(n) \sim \mathcal{P}\left(\frac{1}{v}(\alpha\left(\frac{\cdot}{v}\right) * u)(n)\right)$ |
| $\mathbb{E}\left(obs(n)\right)$ (observed) | $\left(\frac{1}{v}\alpha\left(\frac{\cdot}{v}\right) * u\right)(n)$ | $\frac{1}{v}(\alpha\left(\frac{\cdot}{v}\right) * u)(n)$ |
| $\mathrm{var}(obs(n))$ (observed) | $\left(\frac{1}{v}\alpha^2\left(\frac{\cdot}{v}\right) * u\right)(n)$ | $\frac{1}{v}(\alpha\left(\frac{\cdot}{v}\right) * u)(n)$ |
| Inverse filter $\hat{\gamma}(\xi)$ | $\frac{\mathbb{1}_{[-\pi,\pi]}(\xi)}{\hat{\alpha}(\xi v)}$ | $\frac{\mathbb{1}_{[-\pi,\pi]}(\xi)}{\hat{\alpha}(\xi v)}$ |
| $\mathbb{E}(\hat{u}_{est}(\xi))$ (deconvolved) | $\hat{u}(\xi)\mathbb{1}_{[-\pi,\pi]}(\xi)$ | $\hat{u}(\xi)\mathbb{1}_{[-\pi,\pi]}(\xi)$ |
| MSE (deconvolved) | $\frac{1}{2\pi}\int_{\mathbb{R}} \frac{\|\alpha\|^2_{L^2(\mathbb{R})}\|u\|_{L^1(\mathbb{R})}}{\lvert\hat{\alpha}(\xi v)\rvert^2}\mathbb{1}_{[-\pi,\pi]}(\xi)d\xi$ | $\frac{1}{2\pi}\int_{\mathbb{R}} \frac{\|\alpha\|_{L^1(\mathbb{R})}\|u\|_{L^1(\mathbb{R})}}{\lvert\hat{\alpha}(\xi v)\rvert^2}\mathbb{1}_{[-\pi,\pi]}(\xi)d\xi$ |

Table 1: This table summarizes the main formulas on numerical and analog flutter shutters. The first column describes the structure of the flutter shutter, the second describes the analog flutter shutter. The notation $\hat{f}$ denotes the standard Fourier transform on $\mathbb{R}$, and $\check{f}$ the inverse Fourier transform on $\mathbb{R}$. The notation $\mathcal{P}(\lambda)$ denotes a Poisson random variable with intensity $\lambda$. See also Section 2.1.

The whole flutter shutter study is usually performed as though the image were a one-dimensional signal, recorded on a line in the direction of the camera-landscape motion. Indeed, the motion blur is one-dimensional and the whole convolution and deconvolution model is applied on each line of the image. From the mathematical viewpoint, the flutter shutter reduces to the 1D convolution of a flutter shutter function $\alpha$ with the one dimensional stochastic observed landscape. The expected value at position $x$ of this stochastic landscape will be denoted by $u(x)$. In all statements, this ideal (noiseless) landscape $u$ is assumed to have finite energy: $u \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$ and $[-\pi, \pi]$ band limited (thanks to the combined camera and sensor frequency cut-off). Therefore, $u$ is well sampled at a unit rate.

The whole formalism of the flutter shutter is summarized in Table 1. Its first row indicates the kind of implementable flutter shutter function, depending on the flutter shutter type and with a discrete code. Formally, the flutter shutter function is

$$\alpha(t) = \sum_{k=0}^{L-1} \alpha_k \mathbb{1}_{[k\Delta t,(k+1)\Delta t)}(t), \tag{1}$$

where $\Delta t > 0$. From (1) we have

$$\hat{\alpha}(\xi) = \Delta t \mathrm{sinc}\left(\frac{\xi \Delta t}{2\pi}\right) \sum_{k=0}^{L-1} \alpha_k e^{-i\left(k+\frac{1}{2}\right)\xi\Delta t}. \tag{2}$$

Hereinafter $\hat{f}(\xi)$ denotes the classic continuous Fourier transform on $\mathbb{R}$ and $\check{f}$ the inverse Fourier transform on $\mathbb{R}$, see **(xx)**. (Hereinafter Latin numerals refer to formulae in the glossary, page 251.) For an analog flutter shutter we have $\alpha_k \in [0,1]$ while for a flutter shutter we have $\alpha_k \in \mathbb{R}$.

In the second row, for a sake of commodity in calculations, the flutter shutter formalism is extended to deal with time continuous as well as piecewise constant (coming from a code) flutter shutter functions. The flutter shutter function is $\alpha(t)$, meaning that the gain can change continuously with time. This extension is actually feasible (see [14] for the rigorous mathematical proof).

The third row of the table gives the exact formula of the observed samples. The notation $X \sim Y$ means that the random variables $X$ and $Y$ have the same law. The notation $\mathcal{P}(\lambda)$ denotes a Poisson random variable with intensity $\lambda$, see also **(vii)**. For the analog flutter shutter, the observed digital image at pixel $n$ is a Poisson noise with intensity $\frac{1}{v}\left(\alpha\left(\frac{\cdot}{v}\right) * u\right)(n)$. This parameter is nothing but the convolution (see **(x)** for the definition) of the landscape $u$ with the (rescaled) flutter shutter function $\alpha$. In other words, $obs(n)$ the sample at position $n \in \mathbb{Z}$, has law $obs(n) \sim \mathcal{P}\left(\frac{1}{v}\left(\alpha\left(\frac{\cdot}{v}\right) * u\right)(n)\right)$. For the flutter shutter, the formula is $obs(n) \sim \sum_{k=0}^{L-1} \alpha_k \mathcal{P}\left(\int_{k\Delta t}^{(k+1)\Delta t} u(n - vt)dt\right)$. It can only be stated with a discrete flutter shutter code. Indeed, we have a linear combination of weighted acquired images, and each one is a (Poisson) random variable. In both cases the observed samples $obs(n)$ are obtained for $n \in \mathbb{Z}$.

As explicit in the fourth row, the expected value of the (observed) image is

$$\mathbb{E}\left(obs(n)\right) = \left(\frac{1}{v}\alpha\left(\frac{\cdot}{v}\right) * u\right)(n),$$

for both kind of flutter shutter. It is nothing but the convolution of the landscape $u$ with the flutter shutter function $\alpha$. However, there is a significant difference in the fifth row. The variance of the observed value at a pixel $n$ is $\mathrm{var}(obs(n)) = \left(\frac{1}{v}\alpha^2\left(\frac{\cdot}{v}\right) * u\right)(n)$ for the flutter shutter. In other words, the variance of the observed value at a pixel $n$ depends on the square of the flutter shutter function $\alpha$ for the flutter shutter. The variance of the observed value at a pixel $n$ is $\mathrm{var}(obs(n)) = \left(\frac{1}{v}\alpha\left(\frac{\cdot}{v}\right) * u\right)(n)$. In other words, dependency is linear for the analog flutter shutter. The fourth and fifth rows are immediately obtained from the third by expectation and variance calculations.

The sixth row gives the inverse filter to be applied to the observed samples in order to deconvolve the observed image. The inverse filter is designed to give back, in expectation, the ideal landscape $u$. Thus, the inverse filter is

$$\hat{\gamma}(\xi) = \frac{\mathbb{1}_{[-\pi,\pi]}(\xi)}{\hat{\alpha}(\xi v)}, \tag{3}$$

for both kind of flutter shutter. Indeed, both kind of flutter shutter have the same expectation (see row four of Table 1). This inverse filter is nothing but the inverse of the Fourier transform of the convolution kernel, $\frac{1}{v}\left(\alpha\left(\frac{\cdot}{v}\right) * \mathrm{sinc}\right)(n)$. (See **(xxi)** for the definition of the sinc function.) The sinc function ensures that it is applied only on the $[-\pi,\pi]$ frequencies. Indeed, $u$ is assumed to be $[-\pi,\pi]$ band-limited.

The seventh row gives the expected value of the deconvolved image. We have $\mathbb{E}(\hat{u}_{est}(\xi)) = \hat{u}(\xi)\mathbb{1}_{[-\pi,\pi]}(\xi)$, for both kind of flutter shutter. We recall that the inverse filter was designed to give back the ideal landscape (in expectation).

The last row gives the main two formulas proved in [14], namely the MSE (or variance) of the restored signal. For the flutter shutter we have

$$\text{MSE}_{\text{numerical}}(\text{deconvolved}) = \frac{\|u\|_{L^1(\mathbb{R})}}{2\pi}\int_{\mathbb{R}} \frac{\|\alpha\|^2_{L^2(\mathbb{R})}}{|\hat{\alpha}(\xi v)|^2}\mathbb{1}_{[-\pi,\pi]}(\xi)d\xi. \tag{4}$$

For the analog flutter shutter we have

$$\text{MSE}_{\text{analog}}(\text{deconvolved}) = \frac{\|u\|_{L^1(\mathbb{R})}}{2\pi}\int_{\mathbb{R}} \frac{\|\alpha\|_{L^1(\mathbb{R})}}{|\hat{\alpha}(\xi v)|^2}\mathbb{1}_{[-\pi,\pi]}(\xi)d\xi. \tag{5}$$

Note that $u$ intervenes in the above formulas as a mere multiplication factor by the constant $\|u\|_{L^1(\mathbb{R})}$. Thus, optimizing a flutter shutter amounts to find flutter shutter functions $\alpha$ that minimize Equation (4) or (5), which are different for the analog and numerical flutter shutter.

With these formulas it is easily checked that if a flutter shutter function $0 \leqslant \alpha(t) \leqslant 1$ is implementable on both kinds of flutter shutters, the MSE of the analog flutter shutter is bigger than the MSE of the numerical flutter shutter. Indeed these conditions on $\alpha(t)$ imply $\alpha^2(t) \leqslant \alpha(t)$ and therefore $\|\alpha\|^2_{L^2(\mathbb{R})} \leqslant \|\alpha\|_{L^1(\mathbb{R})}$. Notice that the MSE of the flutter shutter does not change by changing $\alpha$ for $\lambda\alpha$ if $\lambda \neq 0$. This is not true for the analog flutter shutter, where for evident physical reasons, $0 \leqslant \alpha(t) \leqslant 1$ and (e.g.) $\frac{\alpha}{2}$ has a higher MSE than $\alpha$.

## 3 Optimal Flutter Shutter Codes Computation [15]

This section gives the formalism used to compute optimal codes as it is developed in [15] by Tendero et al. Nevertheless, the exposition is self-contained.

Let $\rho(v)$ be a probability density for the camera-landscape velocities $v$. Note that this is equivalent to assuming a probability density on motion blur supports measured in pixel(s). However, the discussion is easier when considering the velocities. We assume that $\rho(v) = 0$ for any $v$ such that $|v| > |v_{max}|$ and that the probability density $\rho(v)$ is known. As we have just seen in Section 2.1, minimizing the MSE amounts to minimizing $\int_{\mathbb{R}} \frac{\|\alpha\|^2_{L^2(\mathbb{R})}\|u\|_{L^1(\mathbb{R})}\mathbb{1}_{[-\pi,\pi]}(\xi)}{|\hat{\alpha}(\xi v)|^2}d\xi$ with respect to the flutter shutter function $\alpha \in L^2(\mathbb{R})$. Dropping the multiplicative constant $\frac{\|u\|_{L^1(\mathbb{R})}}{2\pi}$ and after a change of variable, this is equivalent to minimizing, for a fixed velocity $v$, the functional $E_v(\hat{\alpha}) = \|\hat{\alpha}\|^2_{L^2(\mathbb{R})}\int_{\mathbb{R}} \frac{\mathbb{1}_{[-\pi|v|,\pi|v|]}(\xi)d\xi}{|v|\,|\hat{\alpha}|^2(\xi)}$. Thus, taking the velocity distribution $\rho(v)$ into account, a flutter shutter function $\alpha \in L^2(\mathbb{R})$ leading to the lowest MSE is obtained by minimizing

$$E(\hat{\alpha}) = \int_{\mathbb{R}} E_v(\hat{\alpha})\rho(v)dv = \|\hat{\alpha}\|^2_{L^2(\mathbb{R})}\int_{\mathbb{R}}\frac{1}{|\hat{\alpha}(\xi)|^2}\left(\int_{\mathbb{R}}\frac{\rho(v)\mathbb{1}_{[-|v|\pi,|v|\pi]}(\xi)}{|v|}dv\right)d\xi, \tag{6}$$

where we used Fubini's theorem. Then, from (6), consider the function

$$w(\xi) := \int_{\mathbb{R}}\frac{\rho(v)\mathbb{1}_{[-|v|\pi,|v|\pi]}(\xi)}{|v|}dv = \int_{\mathbb{R}\setminus[\frac{-|\xi|}{\pi},\frac{|\xi|}{\pi}]}\frac{\rho(v)+\rho(-v)}{2|v|}dv, \tag{7}$$

that will be useful for the rest of this paper. From (7), we deduce that $w$ is even, that $w \geqslant 0$ and that $w(\xi) = 0$ for any $\xi \in \mathbb{R}$ such that $|\xi| \geqslant |v_{max}|\pi$. In addition, from (7), we deduce that

$$\int_{\mathbb{R}}|w(\xi)|d\xi = \int_{\mathbb{R}}2|v|\pi\frac{\rho(v)}{|v|}dv = 2\pi.$$

Thus, $w \in L^1(\mathbb{R})$ and is compactly supported, so that $\sqrt[4]{w} \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$. With the help of the function $w$ we can finally formulate the energy that minimizes (6) in a closed form.

**Definition 1** *Given a velocity probability density $\rho$ we call optimal flutter shutter gain function for $\rho$ any function $\alpha \in L(\mathbb{R})^2$ that minimizes*

$$E(\hat{\alpha}) = \|\hat{\alpha}\|^2_{L^2(\mathbb{R})} \int_{-\infty}^{\infty} \frac{w(\xi)}{|\hat{\alpha}|^2(\xi)} d\xi, \tag{8}$$

*where $w$ is linked to $\rho$ by (7).*

The energy (8) is invariant to arbitrary translations and scalings of $\alpha$, i.e. satisfies $E(C_1\hat{\alpha}(\cdot)e^{-iC_2\cdot}) = E(\hat{\alpha})$, for any constants $C_1 \in \mathbb{R}\backslash\{0\}$ and $C_2 \in \mathbb{R}$. Therefore, minimizers of (8) among functions $\hat{\alpha} \in L^2(\mathbb{R})$ are not unique, as soon as there is one. We have

**Theorem 1** ***(Optimal time-continuous flutter shutter gain functions)***
*Let $\rho$ be a probability density supported on $[-|v_{max}|, |v_{max}|]$ and consider $w$ obtained from $\rho$ by (7). A flutter shutter gain function $\alpha \in L^2(\mathbb{R})$ is optimal in terms of MSE (8) if and only if, for some $C > 0$, $\hat{\alpha}$ satisfies $|\hat{\alpha}| = C\sqrt[4]{w}$ on the support of $w$, and $\hat{\alpha} = 0$ outside the support of $w$.*

Given a code length $L$ we need to compute a flutter shutter gain function of the form (1) in order to deduce a feasible flutter shutter from the above solution.

**Theorem 2 (Optimal flutter shutter codes in terms of MSE)**
*Let $\rho$, $w$ as in Theorem 1 and $\Delta t$ be such that $|v_{max}|\Delta t \leqslant 1$. Consider a sequence $(\alpha_k)_k \in \ell^2(\mathbb{Z})$ and the $L^2(\mathbb{R})$ piecewise constant flutter shutter gain function uniquely associated with $(\alpha_k)_k$,*

$$\alpha(t) = \sum_{k\in\mathbb{Z}} \alpha_k \mathbb{1}_{[k\Delta t,(k+1)\Delta t)}(t), \tag{9}$$

*where $\Delta t > 0$. A sequence $(\alpha_k)_k \in \ell^2(\mathbb{Z})$ is optimal with respect to the MSE (8) if and only if $(\alpha_k)_k$ satisfies $\left|\sum_{k\in\mathbb{Z}} \alpha_k e^{-ik\xi}\right| = C\frac{\sqrt[4]{w(\frac{\xi}{\Delta}t)}}{\sqrt{\mathrm{sinc}(\frac{\xi}{2\pi})}}$ for some fixed $C > 0$ and for any $\xi \in [-\pi, \pi]$.*
*In addition, the real values $\alpha_k$, for $k \in \mathbb{Z}$, explicitly given by*

$$\alpha_k = \frac{1}{2\pi}\int_{-\pi|v_{max}|\Delta t}^{\pi|v_{max}|\Delta t} \frac{\sqrt[4]{w(\frac{\xi}{\Delta t})}\cos(ks)}{\sqrt{\mathrm{sinc}\left(\frac{\xi}{2\pi}\right)}} d\xi = \frac{1}{\pi}\int_{0}^{\pi|v_{max}|\Delta t} \frac{\sqrt[4]{w(\frac{\xi}{\Delta t})}\cos(ks)}{\sqrt{\mathrm{sinc}\left(\frac{\xi}{2\pi}\right)}} d\xi, \tag{10}$$

*define an optimal flutter shutter gain function $\alpha \in L^2(\mathbb{R})$ with respect to the energy (8), among all real-valued functions in $L^2(\mathbb{R})$ of the form (9).*

**Remark** 1 From (7) we have that $w(\xi) = \int_{\mathbb{R}\backslash[\frac{-|\xi|}{\pi},\frac{|\xi|}{\pi}]} \frac{\rho(v)+\rho(-v)}{2|v|} dv$. We deduce that optimal flutter shutter gain functions $\alpha \in L^2(\mathbb{R})$ only depend on the even part of $\rho$. Indeed, the choice of a positive direction for the velocities $v$ is arbitrary and does not change the MSE. Therefore, from Theorem 2 we deduce that optimal codes depend only on the motion magnitudes distribution.

The coefficients $\alpha_k$ of the optimal flutter shutter code are real. However, they are not necessarily positive (nor the ideal flutter shutter function $\widecheck{\sqrt[4]{w}}(t)$). This implies that, in general, the code cannot be implemented with an analog flutter shutter. Note that $w\left(\frac{\cdot}{\Delta t}\right)$ is supported on $[-\pi|v_{max}|\Delta t, \pi|v_{max}|\Delta t]$.

**Remark** 2 In the sequel we shall compute optimal codes for an uniform and a (truncated) Gaussian motion model. Note that to compute the $a_k$ defined in (10) we can drop any multiplicative constants. Indeed, the MSE of a flutter shutter defined in (4) is invariant by changing flutter shutter gain function $\alpha$ to $\lambda\alpha$ for any $\lambda \in \mathbb{R}\backslash\{0\}$. This means that one does not need to normalize $\rho(v)$ so that $\int_{\mathbb{R}} \rho(v)dv = 1$, and that one can normalize the code coefficients $\alpha_k$ so that (e.g.) $\int_{\mathbb{R}} \alpha(t)dt = 1$. In the sequel we shall consider a (truncated) Gaussian motion model of the form

$$\rho(v) \propto \mathbb{1}_{[-4\sigma_{\mathcal{N}},4\sigma_{\mathcal{N}}]} \exp\left(\frac{-v^2}{2\sigma_{\mathcal{N}}^2}\right), \tag{11}$$

where $\propto$ means proportional to, $\sigma_{\mathcal{N}} > 0$ and a uniform motion model of the form

$$\rho(v) = \frac{1}{2|v_{max}|}\mathbb{1}_{[-|v_{max}|,|v_{max}|]}(v), \tag{12}$$

where $v_{max} \neq 0$. When $\rho$ is given by (11), from (7) we have

$$w(\xi) = \int_0^{4\sigma_{\mathcal{N}}} \exp\left(\frac{-v^2}{2\sigma_{\mathcal{N}}^2}\right) \frac{\mathbb{1}_{[-|v|\pi,|v|\pi]}(\xi)}{|v|}dv. \tag{13}$$

When $\rho$ is given by (12), from (7) we have

$$w(\xi) = \int_0^{|v_{max}|} \frac{\mathbb{1}_{[-|v|\pi,|v|\pi]}(\xi)}{|v|}dv, \tag{14}$$

up to a multiplicative constant. The functions in (13) and (14) have singularities at $\xi = 0$. However, $w \in L^1(\mathbb{R})$ (see page 240). Therefore, in the sequel, to avoid numerical issues we propose to evaluate

$$\int_\varepsilon^{4\sigma_{\mathcal{N}}} \exp\left(\frac{-v^2}{2\sigma_{\mathcal{N}}^2}\right) \frac{\mathbb{1}_{[-|v|\pi,|v|\pi]}(\xi)}{|v|}dv \tag{15}$$

instead of (13) and

$$\int_\varepsilon^{|v_{max}|} \frac{\mathbb{1}_{[-|v|\pi,|v|\pi]}(\xi)}{|v|}dv \tag{16}$$

instead of (14). Hence, combining (10) and (15) we deduce that

$$\alpha_k = \int_0^{4\pi\sigma_{\mathcal{N}}\Delta t} \frac{\sqrt[4]{\int_\varepsilon^{4\sigma_{\mathcal{N}}} \exp\left(\frac{-v^2}{2\sigma_{\mathcal{N}}^2}\right) \frac{\mathbb{1}_{[-|v|\pi,|v|\pi]}\left(\frac{\xi}{\Delta t}\right)}{|v|}dv} \cos(k\xi)}{\sqrt{\operatorname{sinc}\left(\frac{\xi}{2\pi}\right)}}d\xi, \tag{17}$$

up to a multiplicative constant, when $\rho(v)$ is given by (11). When $\rho(v)$ is given by (12), combining (10) and (16) we obtain

$$\alpha_k = \int_0^{\pi|v_{max}|\Delta t} \frac{\sqrt[4]{\int_\varepsilon^{|v_{max}|} \frac{\mathbb{1}_{[-|v|\pi,|v|\pi]}\left(\frac{\xi}{\Delta t}\right)}{|v|}dv} \cos(k\xi)}{\sqrt{\operatorname{sinc}\left(\frac{\xi}{2\pi}\right)}}d\xi, \tag{18}$$

up to a multiplicative constant. We recall that the MSE (4) is invariant to non zero multiplicative constants. Therefore, the $a_k$ defined in (17) and (18) can be normalized so that, e.g. $\int \alpha(t)dt = 1$.

**Remark** 3 The same scheme is applied to snapshots. This is needed to compare on an equal footing the optimal flutter shutter and the optimal snapshot. The optimal snapshot minimizes (6) among flutter shutter functions of the form of $\alpha(t) = \mathbb{1}_{[0,T]}(t)$. Therefore, combining (2) and (6)-(7) we deduce that we need to minimize with respect to $T$ the following functional

$$\int_{\mathbb{R}} \int_{-\pi}^{\pi} \frac{1}{T\text{sinc}^2\left(\frac{\xi v T}{2\pi}\right)} d\xi \rho(v) dv. \tag{19}$$

Combining **(xxi)** and (19), assuming that $\rho$ is even, supported on $[-|v_{max}|, |v_{max}|]$ and dropping the multiplicative constant 4 we have

$$E(T) = \int_0^{|v_{max}|} \int_0^{\pi} \frac{1}{T\text{sinc}^2\left(\frac{\xi v T}{2\pi}\right)} \rho(v) d\xi dv. \tag{20}$$

Note that as soon as $\rho$ is even it is equivalent to minimize (19) or (20) and that $\rho$ is even for the two examples considered in this paper. In addition, snapshots are invertible as soon as $|v_{max}|T < 2$ [14]. Indeed, the energy $E(T)$ defined in Equation (20) is infinite for $T \geqslant \frac{2}{|v\_max|}$. It is also easy to see that $E(T) \xrightarrow{T\to 0} +\infty$. Therefore, the optimum $T^*$, if it exists, satisfies $T^* \in (0, \frac{2}{|v\_max|})$. In [15] it is proven that (19) is strictly convex. Since (19) is finite on $(0, \frac{2}{|v\_max|})$ we deduce that it admits exactly one minimizer. Hence, (20) also has exactly one minimizer. However, no analytical formula have been obtained that links the minimizer of $E$ and the velocity distribution $\rho$. Thus, in the sequel the minimizer is computed numerically for $T \in (\varepsilon, \frac{1.999}{|v_{max}|})$ where $\varepsilon > 0$.

**Gain evaluation** The gain in terms of RMSE depends on the velocity $v$. Thus, it is useful for the analysis to define $G(v)$ the gain at velocity $v$ of the optimal flutter shutter with respect to the optimal snapshot in terms of RMSE. From (4) the MSE of a snapshot with exposure time $T^*$ is

$$\text{MSE}_{\text{snapshot}}(v) = \frac{\|u\|_{L^1(\mathbb{R})}}{2\pi} \int_{-\pi}^{\pi} \frac{1}{T\text{sinc}^2\left(\frac{\xi v T}{2\pi}\right)} d\xi = \frac{\|u\|_{L^1(\mathbb{R})}}{\pi} \int_0^{\pi} \frac{1}{T\text{sinc}^2\left(\frac{\xi v T}{2\pi}\right)} d\xi. \tag{21}$$

Similarly, from (4) we deduce that

$$\text{MSE}_{\text{flutter}}(v) = \frac{\|u\|_{L^1(\mathbb{R})}}{\pi} \int_0^{\pi} \frac{\|\alpha\|^2_{L^2(\mathbb{R})}}{|\hat{\alpha}(\xi v)|^2} d\xi. \tag{22}$$

Therefore, from (21) and (22), $G(v)$ the gain at velocity $v$ of the optimal flutter shutter with respect to the optimal snapshot in terms of RMSE, is

$$G(v) = \sqrt{\frac{\int_0^{\pi} \frac{1}{T^*\text{sinc}^2\left(\frac{\xi v T^*}{2\pi}\right)} d\xi}{\int_0^{\pi} \frac{\|\alpha\|^2_{L^2(\mathbb{R})}}{|\hat{\alpha}(\xi v)|^2} d\xi}}, \tag{23}$$

where $v$ is in the support $[-|v_{max}|, |v_{max}|]$ of the velocity distribution $\rho$ , and $T^*$ is the exposure time of the optimal snapshot. The optimal exposure time of the snapshot $T^*$ is defined from Equation (20). Recall that a piecewise constant flutter shutter gain function $\alpha$ has the generic form of (1). Moreover, its time-step is $\Delta t$ and $\Delta t \neq T^*$ in general. The coefficients $\alpha_k$ of the flutter shutter code are explicitly given by (17) and (18). The average gain of the flutter shutter in terms of RMSE with respect to the optimal snapshot is defined by

$$\mu := \int_{\mathbb{R}} G(v)\rho(v)dv, \tag{24}$$

and the associated standard deviation is

$$\sigma := \sqrt{\int_{\mathbb{R}} \left| G(v) - \int_{\mathbb{R}} G(u)\rho(u)du \right|^2 \rho(v)dv}. \tag{25}$$

**Remark** 4 From (23) we deduce the gain is invariant by changing $\alpha$ to $\lambda\alpha$ for any $\lambda \in \mathbb{R}^*$. This means that we can without loss of generality normalize $\alpha$, $\widecheck{\sqrt[4]{w}}$ so that, e.g. $\int_{\mathbb{R}} \alpha(x)dx = 1$ and $\int_{\mathbb{R}} \widecheck{\sqrt[4]{w}}(x)dx = 1$. In addition, from (21) we deduce that $G(0) = \sqrt{\frac{\frac{\pi}{T^*}}{\frac{\pi\|\alpha\|^2_{L^2(\mathbb{R})}}{|\hat{\alpha}(0)|^2}}}$.

## 4 The Reverse Path: from Classic Codes to Their Underlying Motion [15]

By the formulae of the previous section we are now able to check if a flutter shutter gain function $\alpha \in L^2(\mathbb{R})$, is optimal for some velocity distribution, and to compute its underlying velocity distribution. Remark 1 implies that from a given optimal flutter shutter gain function $\alpha \in L^2(\mathbb{R})$ associated with some unknown probability density $\rho$, one can only recover the even part of $\rho$ namely $\frac{\rho(\cdot)+\rho(-\cdot)}{2}$. Indeed, the optimal flutter shutter gain functions $\alpha \in L^2(\mathbb{R})$ only depend on the even part of $\rho$. Consequently, throughout this section we shall assume that $\rho$ is even.

**Theorem 3** ***(An optimality test for flutter shutter codes and a formula for their underlying velocity distribution)***
*Assume that $\alpha \in L^2(\mathbb{R})$ has the form of (9) and that $|v_{max}|\Delta t \leqslant 1$. Then $\alpha$ is optimal in the sense of (8) only if $(0, \frac{\pi}{\Delta t}] \ni \xi \mapsto \frac{|\hat{\alpha}(\xi)|^4}{\left|\text{sinc}\left(\frac{\xi\Delta t}{2\pi}\right)\right|^2}$ is non-increasing. Moreover, if $(0, \frac{\pi}{\Delta t}] \ni \xi \mapsto \frac{|\hat{\alpha}(\xi)|^4}{\left|\text{sinc}\left(\frac{\xi\Delta t}{2\pi}\right)\right|^2}$ is non-increasing then*

$$\rho(v) = -\frac{v}{2}w'(\pi v) = -\frac{vC}{2}\left(\frac{|\hat{\alpha}(\xi)|^4}{\left|\text{sinc}\left(\frac{\xi\Delta t}{2\pi}\right)\right|^2}\right)'(\pi v), \text{ for } v \in [\frac{-1}{\Delta t}, \frac{1}{\Delta t}]\backslash\{0\}, \tag{26}$$

*where $C$ is a positive constant, $w$ is given by (7) and the derivatives in (26) are in the distribution sense, if $\rho$ is just in $L^1(\mathbb{R})$.*

Note that for any discrete $\alpha$ of the form $\alpha(t) = \sum_{k=0}^{L-1} \alpha_k \mathbb{1}_{[k\Delta t,(k+1)\Delta t)}(t)$ we have $\alpha \in L^2(\mathbb{R})$ and therefore Theorem 3 applies. Moreover, Theorem 3 gives a direct algorithm that computes $\rho$. This numerical method is detailed in Algorithm 2.

**Remark** 5 As we shall see in Section 6.2 most classic codes do not strictly satisfy the conditions of Theorem 3. Fortunately, for these codes the set where $|\hat{\alpha}|$ is increasing has small measure. In addition $|\hat{\alpha}|$ is small on this set. Thus, we can apply Algorithm 2 by modifying only slightly $\alpha$ (or $w$) by replacing (26) by

$$\rho(v) = -\frac{v}{2\pi}w'(\pi v)\mathbb{1}_{\{-vw'(\pi v)\geqslant 0\}}(v), \text{ for } v \neq 0, \tag{27}$$

and normalizing $\rho$ so that $\int \rho = 1$.

## 5 Algorithms

This section gives the numerical methods related to sections 3 and 4. Algorithm 1 implements the theory recalled in Section 3 that computes optimal flutter shutter codes and snapshots. Algorithm 2 implements the formulas of Section 4 that permit the reverse engineering of flutter shutter codes.

The goal of Algorithm 1 is to evaluate numerically the advantages of the flutter shutter method compared to the optimal snapshot. Therefore, one of the main questions is the evaluation of the trade-off: increase of the exposure time versus gain of the flutter shutter, in terms of RMSE, compared to an optimal snapshot. From Theorem 2 we deduce that optimal flutter shutter gain functions $\alpha^*$ are supported on $\mathbb{R}$. Thus, we propose to approximate $\alpha^*$ by a finitely supported function. Therefore, we propose 1) to truncate the sum in Theorem 2 and 2) to choose the code length $L$ and $\Delta t$ so that the total exposure time $L\Delta t$ of the flutter shutter is a given factor of the exposure time $c$ of the snapshot. Indeed, this allows us to evaluate the gain of the flutter shutter method keeping into account the increased exposure time. In a nutshell, this setup allows us to evaluate, numerically, the efficiency of the flutter shutter. The quality of the proposed approximation can be evaluated qualitatively in figures 2 and 4.

The input parameters of Algorithm 1 are: 1) a motion model $\rho$, 2) a code length $L$, 3) an exposure time factor $c$. Its outputs are text files. They contain the optimal flutter shutter code and the gain evaluation (see page 242). The Algorithm 1 consists of six steps. Step 1 computes the optimal snapshot as it is defined by Equation (19). Step 2 computes the time step $\Delta t$ of the flutter shutter function defined in Equation (9). Step 3 implements formulas (17) and (18) that give the flutter shutter code coefficients. Numerical details related to the integral computation are given in Algorithm 3. Step 4 consists in writing some useful results in a text file. Step 5 implements formula (23) and writes the result in a text file. Step 6 deals with formula (24) and, again, saves the produced result in a text file.

Recall that Algorithm 2 computes the velocity distribution for which a given flutter shutter code is optimal. The input consists in a flutter shutter code. It computes the flutter shutter function $\alpha$ defined in Equation (9), with a normalized $\Delta t := 1$. Its output is a text file that contains $\rho(v)$ the estimated velocity distribution. Algorithm 2 consists of five steps. Step 1 computes $\hat{\alpha}(\xi)$ (see (2)). Step 2 estimates the function $w$ defined in Equation (7). Step 3 estimates the derivative of $w$. Step 4 implements Theorem 3 with the variant given by (27), i.e. estimates the velocity distribution $\rho(v)$ for which the input code is optimal. Step 5 consists in a normalization and writing the result in a text file.

The next paragraph details the numerical methods used in step 3 of Algorithm 1.

**Numerical evaluation of an integral by the Simpson method** Let $f$ be the function to integrate on an interval $[a, b]$. Assume that $[a, b]$ is split in $n$ even subintervals. The Simpson method consists [5, p. 206] in the approximation

$$\int_a^b f(x)dx \approx \frac{h}{3}\left(f(x_0) + 2\sum_{j=1}^{\frac{n}{2}-1} f(x_{2j}) + 4\sum_{j=1}^{\frac{n}{2}} f(x_{2j-1}) + f(x_n)\right),$$

where $h = \frac{b-a}{n}$ and $x_j = a + jh$ for $j \in \{0, \dots, n-1\}$ (thus $x_0 = a$ and $x_n = b$). This formula leads to the pseudo code given in Algorithm 3.

```
Algorithm 1: Pseudo-code computing optimal flutter shutter codes.
input  : 1) velocity model: either $\rho(v) \propto \mathbb{1}_{[-4\sigma,4\sigma]} \exp\left(\frac{-v^2}{2\sigma_{\mathcal{N}}^2}\right)$ or $\rho(v) = \frac{1}{2|v_{max}|}\mathbb{1}_{[-|v_{max}|,|v_{max}|]}(v)$,
         where $\sigma_{\mathcal{N}} > 0$ and $v_{max}$ are parameters, 2) code length $L$, 3) exposure time factor $c$.
output: Text files containing: the code coefficients $\alpha_k$ for $k \in \{-\text{round}(\frac{L}{2}), \dots, \text{round}(\frac{L}{2}) - 1\}$,
         Fourier transform of $\alpha$ (see (2)), ideal Fourier transform $w$ (see (7)), gain $G(v)$
         (see (23)), average gain (see (24)) and associated standard deviation (see (25)).
   1. compute $T^*$ the exposure time of the optimal snapshot on average using (20), i.e.
         $$T^* = \text{argmin}_T \left( \int_0^{|v_{max}|} \int_0^{\pi} \frac{1}{T\text{sinc}^2\left(\frac{\xi v T}{2\pi}\right)} \rho(v) d\xi dv \right).$$
      Note that for both motion models $\rho$ is even and compactly supported. Therefore,
      Equation (20) is valid. The minimum is calculated by scanning on values of $T \in [\frac{0.02}{|v_{max}|}, \frac{1.999}{|v_{max}|}]$
      (see Remark 3 for a justification of these bounds) at a precision of $\frac{0.02}{|v_{max}|}$. The numerical
      evaluation of the integrals is detailed in Algorithm 3, the precision parameter is fixed at
      $\epsilon = 0.001$. Lastly, notice that it is not necessary to normalize $\rho$ to ensure that $\int_{\mathbb{R}} \rho(v)dv = 1$.
      Indeed, argmins do not depend on nonzero multiplicative factors. This step is implemented in
      *optimal_snapshot.cpp*;
   2. compute $\Delta t = \frac{cT^*}{L}$, the time step of the flutter shutter. This is done in *demo_fluttercode.cpp*
      line 150;
   3. compute the code coefficients $\alpha_k$ for $k \in \{-\text{round}(\frac{L}{2}), \dots, \text{round}(\frac{L}{2}) - 1\}$. We use (17) for the
      (truncated) Gaussian model and (18) for the uniform motion model. As discussed in
      Remark 2 we evaluate (15) for the (truncated) Gaussian model (with $\varepsilon = \frac{\sigma_{\mathcal{N}}}{1000}$) and (16) for
      the uniform motion model (with $\varepsilon = \frac{|v_{max}|}{1000}$). The numerical evaluation of the integral is
      detailed in Algorithm 3. The precision parameter is fixed at $\epsilon = 0.001$. Without loss of
      generality the code coefficients are normalized so that $\int \alpha(t)dt = 1$. See Remark 4 for the
      justification.
   4. compute and write the code coefficients $\alpha_k$ for $k \in \{-\text{round}(\frac{L}{2}), \dots, \text{round}(\frac{L}{2}) - 1\}$, the
      (modulus) of the Fourier transform of the flutter shutter function (see (2), this is
      implemented in *gain_evaluation.cpp* files) and the ideal Fourier transform $\sqrt[4]{w(\xi)}$ in their
      corresponding text files. Note that $\sqrt[4]{w(\xi)}$ is normalized, see Remark 4 for the justification.
      This step is implemented in *demo_fluttercode.cpp* lines 180-249;
   5. write the gain $G(v)$ defined in (23) in the appropriate text file. Integrals are evaluated by
      Riemann sums over 1000 points. This is implemented in *gain_evaluation.cpp*. Write the
      results to text files (see *demo_fluttercode.cpp* lines 250-324);
   6. compute the average gain $\mu$ (24) and the associated standard deviation $\sigma$ (25). Integrals are
      evaluated by Riemann sums over 1000 points. This is implemented in *gain_evaluation.cpp*.
      Write the results to text files (see *demo_fluttercode.cpp* lines 250-324).
```

**Algorithm 2**: Pseudo-code estimating the velocity distribution associated with a given code.

```
input  : a flutter shutter code $(\alpha_k)_{k\in\{0,\dots,L-1\}}$, flag for logarithmic scale.
output: underlying probability density $\rho$ for which the code is optimal
    1. compute $|\hat{\alpha}(\xi)|$ (see (2)). We recall that we normalize the time step, i.e. $\Delta t = 1$. This is
       implemented in routines.cpp;
    2. compute the function $w$ defined in (7) by $w(\xi) = \left(\frac{|\hat{\alpha}(\xi)|^4}{\left|\mathrm{sinc}\left(\frac{\xi\Delta t}{2\pi}\right)\right|^2}\right)(\xi)$ (see Theorem 3). This is
       implemented in routines.cpp;
    3. estimate $w'(\xi)$ by $w'(\xi) \approx \frac{w(\xi+\epsilon)-w(\xi)}{\epsilon}$ ($\epsilon = \frac{2}{1001}$ in the proposed implementation) using
       $w(\xi) = |\hat{\alpha}(\xi)|^4$. This is implemented in routines.cpp. Write it in a text file. This is
       implemented in demo_flutter_density.cpp;
    4. compute $\rho(v)$ using (27) for $v \in [-1,1]\backslash\{0\}$ (see Remark 5). Note that we can omit the $\frac{1}{2\pi}$
       multiplicative factor in (27), see step 5. This is implemented in demo_flutter_density.cpp;
    5. normalize so $\int_{\mathbb{R}} \rho(v)dv = 1$, write the result in text file. If logarithmic scale write
       $\log(1+\rho(\cdot))$ instead. This is implemented in demo_flutter_density.cpp;
```

**Algorithm 3**: Pseudo code: integral evaluation by the Simpson method (see, e.g. [5, p. 206]).

```
input  : $a$, $b$, $\epsilon > 0$ (precision), function $f$
output: $tn$ the numerical approximation of $\int_a^b f(x)dx$
initialization: $h \leftarrow \frac{b-a}{2}$;
$s1 \leftarrow f(a) + f(b)$;
$s2 \leftarrow 0$;
$s4 \leftarrow f(a+h)$;
$tn \leftarrow h\frac{s1+4s4}{3}$;
$ta = tn(1+2\epsilon)$;
$zh \leftarrow 2$;
1 while$|ta - tn| > \epsilon|tn|$ $ta \leftarrow tn$;
2 $zh \leftarrow 2zh$;
3 $h \leftarrow \frac{h}{2}$;
4 $s2 \leftarrow s2 + s4$;
5 $s4 \leftarrow 0$;
6 $j \leftarrow 1$;
7 while$j \leqslant zh$ $s4 \leftarrow s4 + f(a+jh)$;
8 $j \leftarrow j + 2$;
9 $tn \leftarrow h\frac{s1+2s2+4s4}{3}$.
```

## 6 Numerical Experiments

Section 6.1 gives the optimal flutter shutter codes and computes the gain of the optimal flutter shutter with respect to the optimal snapshot for two natural velocity distributions: a (truncated) Gaussian velocity model and an uniform velocity model. The Gaussian velocity model is explicitly given by (11). The corresponding numerical method is described in Algorithm 1. Section 6.2 gives the reverse engineering of classic flutter shutter codes in the literature. The corresponding numerical method is described in Algorithm 2.

## 6.1 Simulations on Optimized Codes

The goal of this section is to numerically explore two natural velocity distributions. For each velocity distribution, we give the corresponding optimal codes. For each velocity distribution, we compare their efficiency in terms of RMSE with respect to the optimal snapshot. This corresponds to the comparison on an equal footing of two alternative solutions: the optimal snapshot and the optimal flutter shutter.

Recall that the parameters of a flutter shutter are: 1) $L$ the length of the code $(\alpha_k)_{k\in\{0,\dots,L-1\}}$, 2) the velocity motion model $\rho(v)$ and 3) the time step $\Delta t$ of the flutter shutter function $\alpha$. The parameter for the optimal snapshot is only the velocity model $\rho(v)$. The optimal snapshot provides $T^*$ the optimal aperture time for a standard camera, i.e. without using a flutter shutter. In order to ease the comparison with the Agrawal et al. code [1, 3, 8, 10, 12], all experiments are made with $L = 52$. Indeed, in [1, 3, 8, 10, 12] the authors chose $L = 52$.

Two velocity motion models are considered and compared: truncated Gaussian (in Section 6.1.1), and uniform velocity distribution model (in Section 6.1.2). The time step of the flutter shutter is chosen such that $L\Delta t$ the total exposure time of the flutter shutter defined by the maximal support of the flutter shutter function $\alpha$ is an integer factor $c$ of $T^*$ the aperture time of the optimal snapshot. In other words, we have $L\Delta t = cT^*$ where $c \in \mathbb{N}^+$. This allows an easy comparison of the flutter shutter and of the snapshot. Indeed, the potential gain of the flutter shutter in terms of RMSE is $\sqrt{c}$ because it integrates on a time span $c$ times larger than the optimal snapshot.

The codes of figures 2 and 4 show the finitely supported and piecewise constant flutter shutter gain functions. These experiments permit to compare the two strategies using a finite exposure time which is mandatory for a practical solution. Recall that in [14] Tendero et al. prove that if the velocity $v_0$ is known, i.e, $\rho(v) = \delta_{v_0}(v)$, the optimal code comes from a zoomed sinc function. In addition, the optimal snapshot has an exposure time $T^*$ tuned so that $|v_0|T^* \approx 1.0909$ (see [14] for the mathematical proof). Moreover, the gain of the flutter shutter in terms of RMSE with respect to this optimal snapshot is of a 1.17 factor [14]. Hereafter the gains in terms of RMSE should also be compared with this 1.17 bound that corresponds to optimize the worst case scenario in a practical situation, i.e. the maximal velocity observed. Notice that *ceteris paribus* a scale change of the velocity model results in a scale change of the function $w(\xi)$ and in a zoom of the code. An equivalent result holds for the optimal snapshot.

### 6.1.1 Optimal Codes, Gaussian Velocity Model

This section provides the optimal codes for a truncated Gaussian $\mathcal{N}(0, \frac{1}{4})$ velocity motion model, i.e. from (11)

$$\rho(v) \propto \mathbb{1}_{[-1,1]} \exp\left(\frac{-v^2}{2(\frac{1}{4})^2}\right). \tag{28}$$

Optimal flutter shutter codes are explicitly depicted in figures 2(a) and 2(c). The Fourier transforms of the corresponding flutter shutter functions are given in figures 2(b) and 2(d). On those two graphs, the green curves show the actual function $\sqrt[4]{w(\xi)}$ which remains unchanged in the two experiments since it only depends on the velocity motion model (see Equation (7)). For comparison purpose, we also depict in red the finitely supported approximation of $\sqrt[4]{w(\xi)}$ by the flutter shutter code. In fact, the main change between the topmost plots of Figure 2 and the lower counterparts is the exposure time factor $c$. For the figures 2(a) and 2(b) we have $c = 5$ while for the figures 2(c) and 2(d) we have $c = 10$. Note that in these experiments we have set $52\Delta t = cT^*$, so that the discretization step is adapted to the value of $c$. Looking at the two values of $c$, one notices that the support of the flutter shutter function doubles between the top and bottom part of Figure 2. It is also clear that the approximation is slightly better for the larger exposure factor $c = 10$ shown at the bottom part of Figure 2. This is no surprise. Indeed, since $w(\xi)$ has compact support, the ideal time continuous flutter shutter function $\widecheck{\sqrt[4]{w(\xi)}}$ is supported on $\mathbb{R}$.

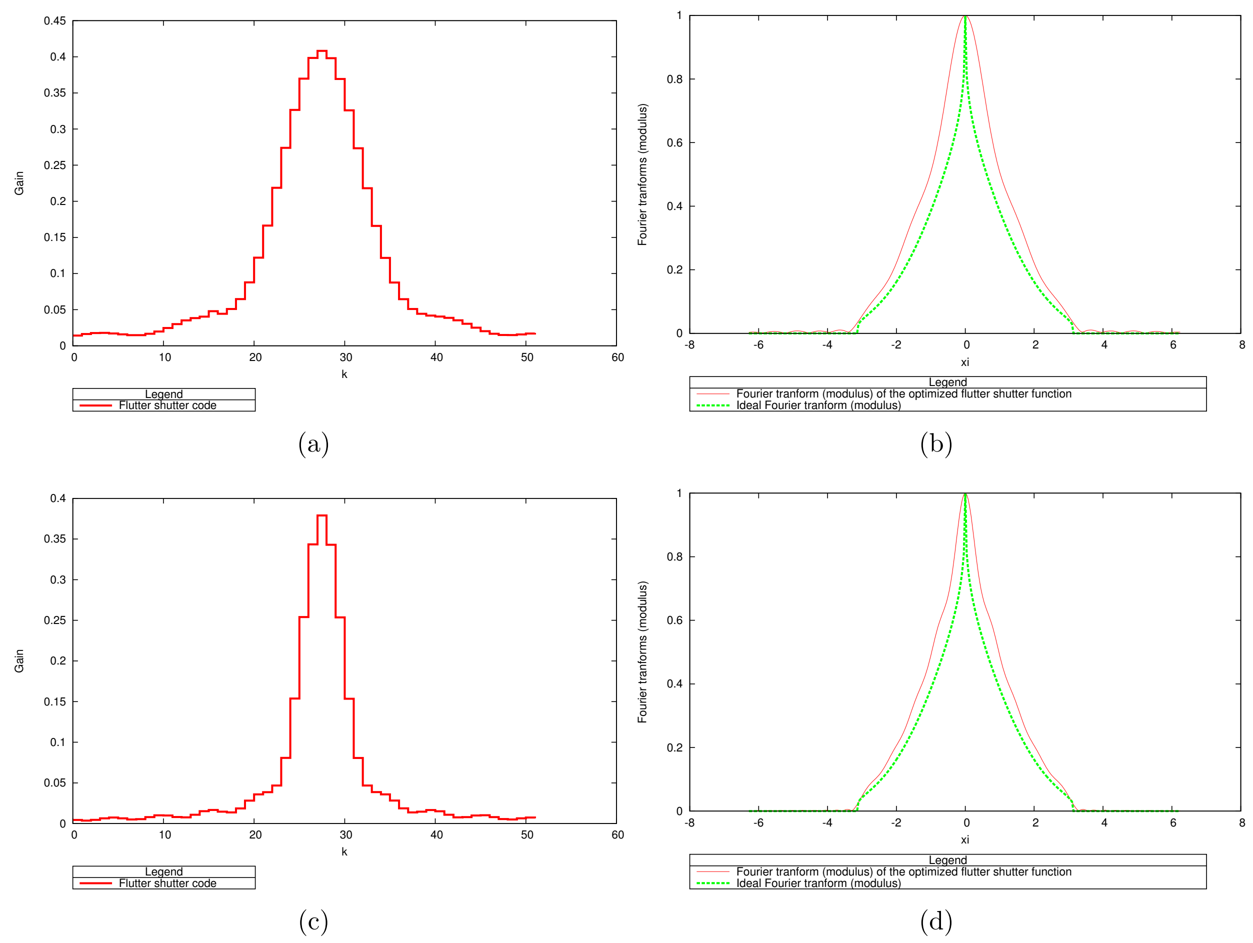


Figure 2: Codes obtained for a truncated Gaussian velocity density explicitly given in (28). On the left panel: top (respectively bottom): the flutter shutter code coefficients $\alpha_k$, using an exposure time 5 (respectively 10) times larger than for the optimal snapshot. On the right panel: (in red) the modulus of their corresponding Fourier transform, and the Fourier transform of the optimal time continuous flutter shutter function $\sqrt[4]{w(\xi)}$ defined in Equation (7) in green. The convergence is quite good, even for small exposure time factors.

Figure 3 provides the comparison with the optimal snapshot in terms of RMSE. For any velocity $v$ in the support of the velocity motion model, the red curves of Figure 3 show the gain $G(v)$ defined in Equation (23) as a function of the velocity $v$. The left panel of Figure 3 corresponds to an exposure factor $c = 5$, i.e. the flutter shutter integrates five times longer than the optimal snapshot. The right panel of Figure 3 corresponds to $c = 10$. Recall that the function $G(v)$ measures the gain of the flutter shutter compared to the optimal snapshot in terms of RMSE. The dotted blue curve provides the probability of the velocity according to the velocity distribution. Thus, we can check that the optimization permits to concentrate the gain in terms of RMSE on most probable velocities. For higher but less likely velocities $v$ the optimized flutter shutter performs worse than the snapshot, i.e. its RMSE is higher than the RMSE of the optimal snapshot. The green line shows the average of $G(v)$ taking the velocity motion model $\rho$ into account, as it is defined in Equation (24).

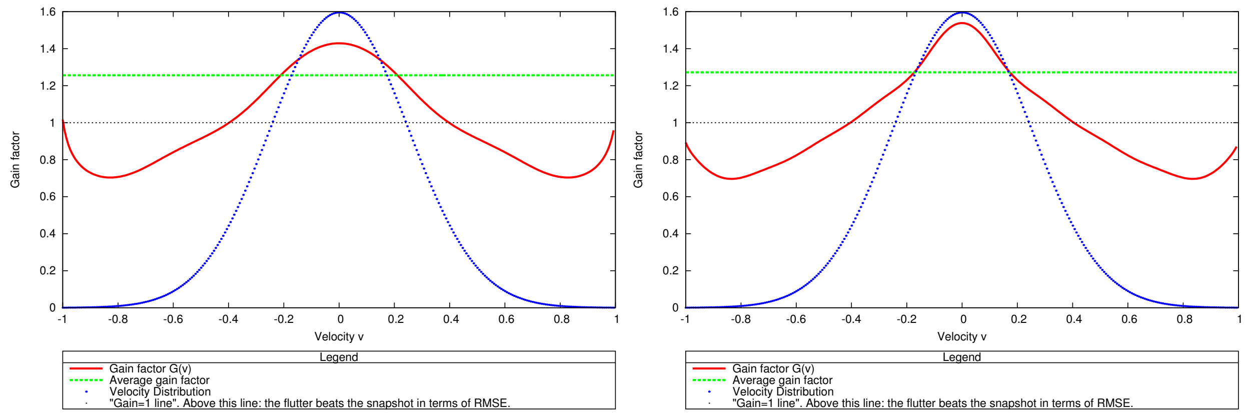


Figure 3: In red: the gain $G(v)$ in terms of RMSE (defined by Equation (23)) of the optimal flutter shutter code with respect to the optimal snapshot for the truncated Gaussian velocity distribution. On the left panel (respectively right panel) for exposure times factor of 5 (respectively 10). The dotted blue curve represents the probability density of the truncated Gaussian velocity distribution. The green curves show the average gain $\mu$ as it is defined in Equation (24). The optimization permits to concentrate the gain on most probable velocities, as expected. On average the gain is substantial compared to the bound of [14] that optimizes the maximal velocity (worst case).

Table 2 provides both the average gain defined by Equation (24) and its associated standard deviation defined by Equation (25). It permits to measure "how risky" the optimization is, i.e. how the gain will vary when one observes velocities according to the motion model explicitly given in (28). Notice that the asymptotic bound of [14] is beaten by approximately 50%.

| Exposure time factor $c$ | 5 | 10 |
|---|---|---|
| Code length $L$ | 52 | 52 |
| Average gain $\mu$ (24) | 1.2556 | 1.2722 |
| Standard deviation $\sigma$ (25) | 0.1706 | 0.1996 |

Table 2: Average gain of the optimized flutter shutter compared to the optimal snapshot, assuming a truncated Gaussian velocity distribution explicitly given in (28). As guessed from Figure 3 the gain is substantial and the increase is of approximately 50% compared to the asymptotic of [14].

### 6.1.2 Optimal Codes, Uniform Motion Model

This section provides the optimal codes for a $\mathcal{U}[-1,1]$ velocity motion model, i.e.

$$\rho(v) = \frac{1}{2}\mathbb{1}_{[-1,1]}(v). \tag{29}$$

The setup is exactly the same as in Section 6.1.1. Optimal flutter shutter codes are explicitly depicted in figures 4(a) and 4(c). The Fourier transforms of the corresponding flutter shutter functions are given in figures 4(b) and 4(d). For the figures 4(a) and 4(b) we have $c = 5$ while for the figures 4(c) and 4(d) we have $c = 10$, as in Section 6.1.1.

Figure 5 provides the comparison with the optimal snapshot in terms of RMSE. The dotted blue curve provides the probability of the velocity according to the velocity distribution. The green line shows the average of $G(v)$ (defined in Equation (23)) taking the velocity motion model $\rho$ into account, as it is defined in Equation (24).

Table 3 provides both the average gain defined by Equation (24) and its associated standard deviation defined by Equation (25), as in Section 6.1.1. The gain is negligible.

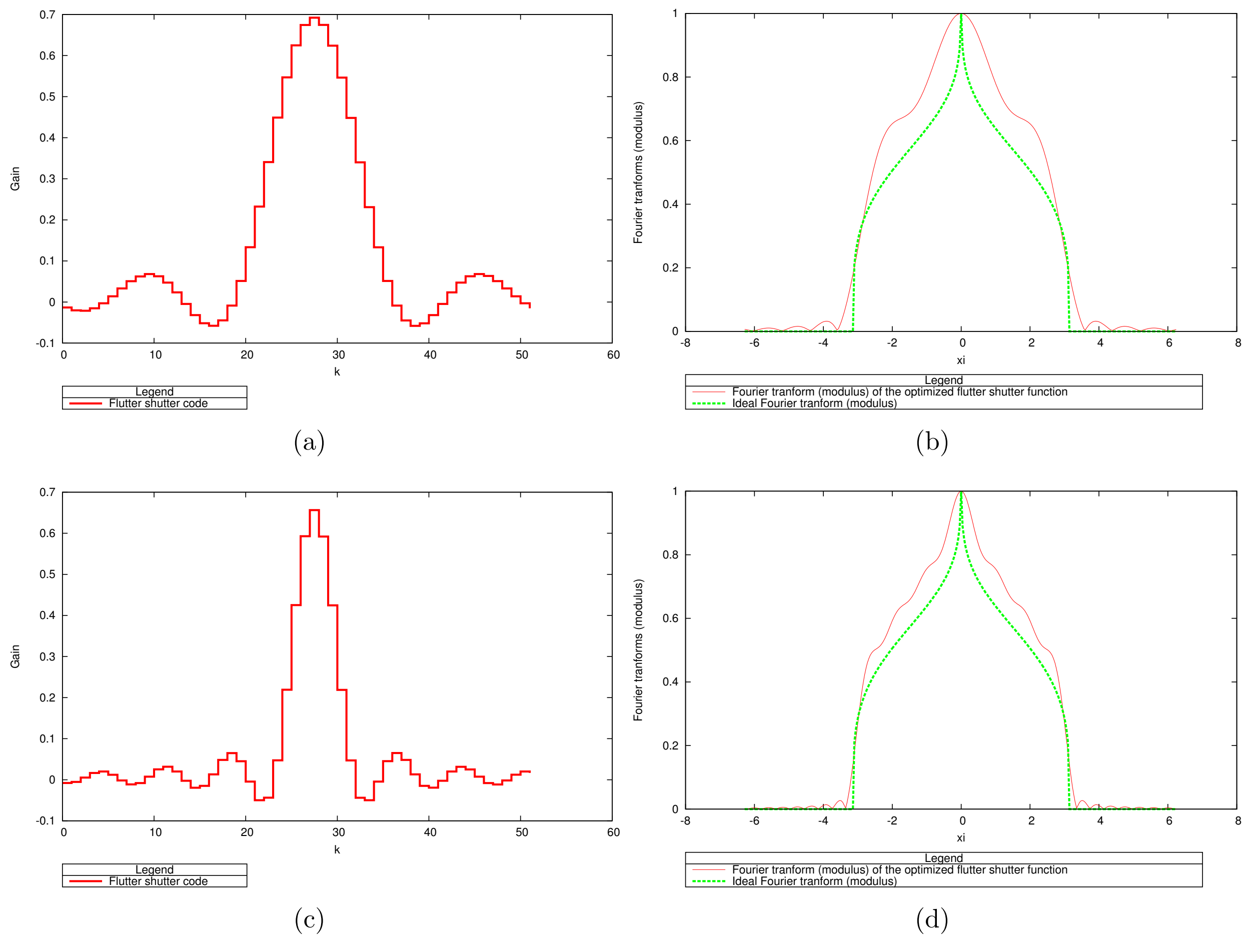


Figure 4: Codes $\alpha_k$ obtained assuming a uniform density for the velocities explicitly given in (29). On the left panel: top (respectively bottom panel): the flutter shutter code coefficients $\alpha_k$, using an exposure time 5 (respectively 10) times larger than for the optimal snapshot. On the right panel: (in red) the modulus of their corresponding Fourier transform, and the Fourier transform of the optimal time continuous flutter shutter function $\sqrt[4]{w(\xi)}$ defined in Equation (7) in green.

| Exposure time factor | 5 | 10 |
|---|---|---|
| Code length $L$ | 52 | 52 |
| Average gain $\mu$ (24) | 1.0702 | 1.0715 |
| Standard deviation $\sigma$ (25) | 0.0415 | 0.0531 |

Table 3: Average gain of the optimized flutter shutter compared to the snapshot, assuming a uniform density for the velocities explicitly given in (29). As could already be guessed from Figure 5, this gain is not significant.

## 6.2 A Reverse Engineering of Classic Flutter Shutter Codes

This section provides the underlying velocity distribution $\rho$ of classic flutter shutter codes of the literature. However, Algorithm 2 is applicable to *any* flutter shutter code. Algorithm 2 with the variant given by Equation (27) is used. We normalize $\Delta t$ in the definition of the flutter shutter gain function (1). This means that the velocities are expressed in pixel per $\Delta t$ s. (See also Remark 5.) Thus, the range of the $x$-axis of figures 6, 7 and 8 is $[-1, 1]$.

We proceed first to the reverse engineering of the Agrawal et al. flutter shutter code [11, p. 799] and patent application [12] on the panel of Figure 6(a). Note that the $y$-axis of 6(a) is log scaled.

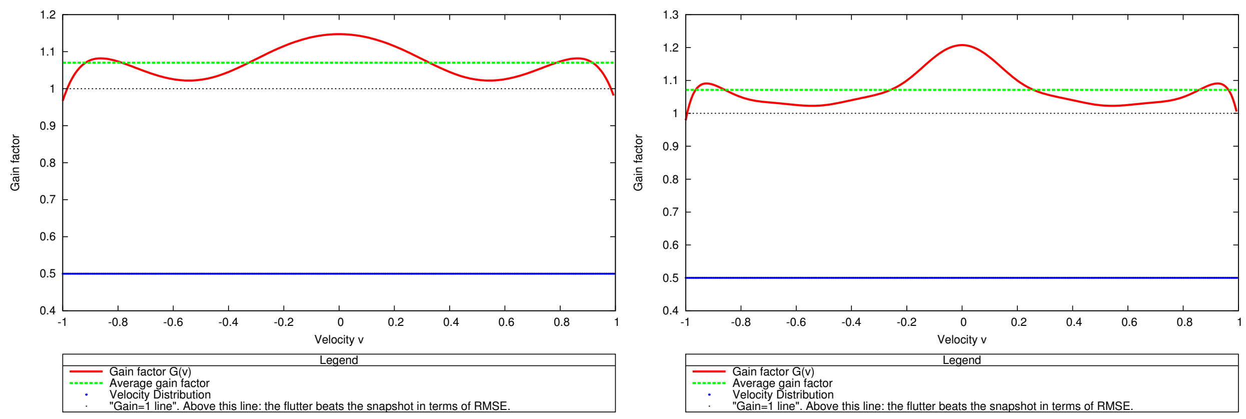


Figure 5: In red: the gain $G(v)$ in terms of RMSE (defined by Equation (23)) of the optimal flutter shutter code with respect to the optimal snapshot for the uniform velocity distribution explicitly given in (29). On the left panel (respectively right panel) for exposure times factor of 5 (respectively 10). The dotted blue curve represents the probability density $\rho(v) = \frac{1}{2}\mathbb{1}_{[-1,1]}(v)$ of the uniform velocity distribution. The green curves show the average gain $\mu$ as it is defined in Equation (24).

This distribution means that there is a high probability that the scene is still and that more or less uniformly distributed velocity motions occur on a certain interval of velocities. However, this is an unlikely model for a camera motion, due to the strange fluctuations of the velocity distribution. The velocity distribution of another flutter shutter code of Agrawal et al. [3, p. 7] is given on the right panel side of Figure 6(b). It is not more convincing than the previous one since it is just more concentrated on small velocities.

The reverse engineering of the Agrawal et al. flutter shutter codes published in [2, p. 2566] (respectively in [1, p. 5]) is shown in Figure 7(a) (respectively in Figure 7(b)).

Another example is the McCloskey code [7, p. 321], shown on the left panel of Figure 8(a). The same scheme can be applied to the "standardized" snapshot, i.e. $\alpha(t) = \mathbb{1}_{[0,1]}(t)$ to estimate the underlying probability density of a classic camera. This example is given in Figure 8(b) where we deduce that it is optimal for relatively broad intervals centered at approximately $|v| = 1$. Among the velocity densities of figures 6(a), 6(b), 7(a), 7(b), 8(a) or 8(b) the velocity distribution of the snapshot shown in Figure 8(b) is the most convincing one.

It is most probable that the velocity distribution of flutter shutter codes optimized using an arbitrary criterion, e.g. a fixed sum of the code coefficients will look like the ones depicted in figures 6(a), 6(b), 7(a), 7(b) or 8(a). Thus, one cannot expect them to have more realistic velocity distributions than the codes of these figures.

## Appendix: Main Notations and Formulae

**(i)** $t$ time variable

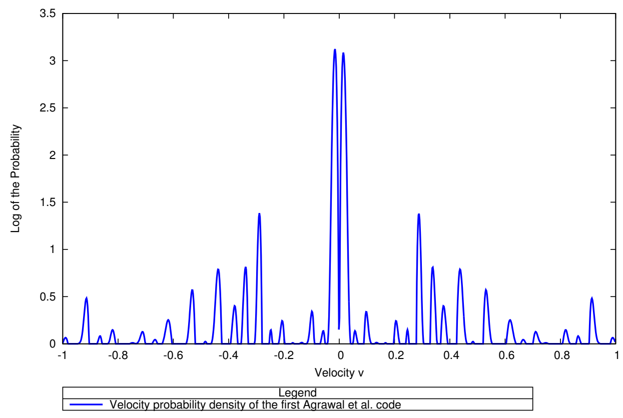
**(ii)** $\Delta t > 0$ length of a time interval

**(iii)** $x \in \mathbb{R}$ spatial variable

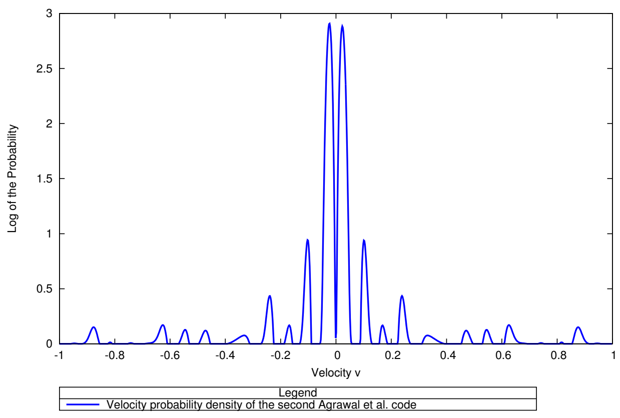
**(iv)** $\propto$ means "is proportional to"

**(v)** $X \sim Y$ means that the random variables $X$ and $Y$ have the same law

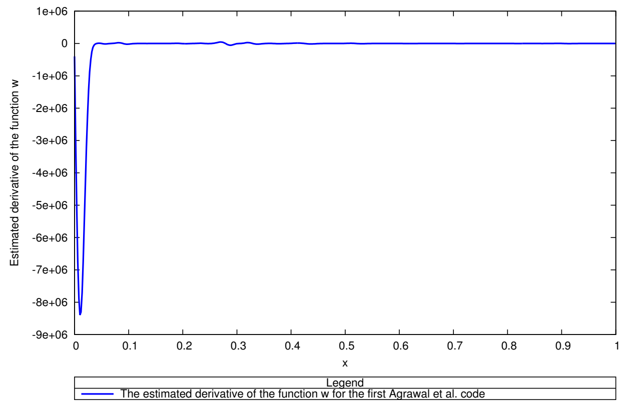
**(vi)** $\mathbb{P}(A)$ probability of an event $A$



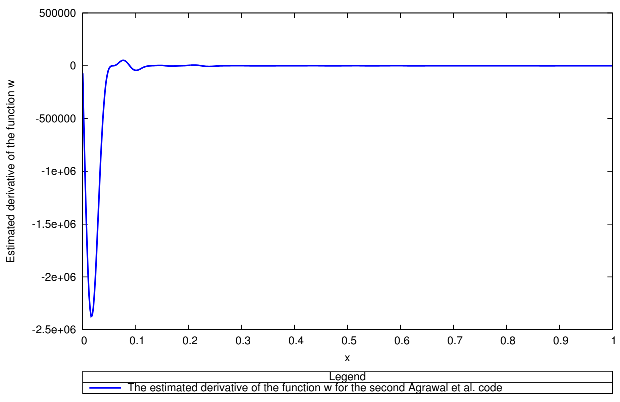
(a)



(b)



(c)



(d)

Figure 6: On figures 6(a) and 6(b): the velocity probability densities $\rho$ associated with Agrawal, Raskar et al. codes. $X$-axis: the velocity (in signed pixels per $\Delta t$)), $y$-axis: the logarithm of the velocity probability densities ($\log(1 + \rho(v))$). On the left panel: the code published in [11, p. 799] and patent application [12]. On the right panel: the code published in [3, p. 7]. It corresponds to an attempt to optimize both the MSE and the a posteriori velocity estimation. These velocity densities are unlikely models for a camera motion, due to the strange fluctuations of the velocity distributions. In addition, both codes have a very large probability concentrated around $v = 0$ (even when log scaled). On figures 6(c) and 6(d) the estimation of the $w'$ function. (Recall that $w$ is defined in (7).) $X$-axis: $x$, $y$-axis the estimated $w'(x)$ from the flutter shutter code coefficients. Recall that Theorem 3 relies on this estimation. As discussed in Remark 5 Algorithm 2 implements (27). Therefore, figures 6(c) and 6(d) permit to see where the $w$ increases. As announced the measure of the set where $w$ increases is very small.

**(vii)** $\mathcal{P}(\lambda)$ Poisson random variable with intensity $\lambda > 0$. Thus, if $X \sim \mathcal{P}(\lambda)$ we have $\mathbb{P}(X = k) = \frac{\lambda^k \exp(-\lambda)}{k!}$

**(viii)** $\mathbb{E}(X)$ expected value of a random variable $X$

**(ix)** $\mathrm{var}(X)$ variance of a random variable $X$

**(x)** Let $f, g \in L^1(\mathbb{R}) \cup L^2(\mathbb{R})$, then $f * g$ denotes convolution of two functions $(f*g)(x) = \int_{-\infty}^{+\infty} f(y)g(x-y)dy$

**(xi)** $u$ ideal (noiseless) observable landscape just before sampling. Assumption: $u \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$, $[-\pi, \pi]$ band-limited

**(xii)** $obs(n)$, $n \in \mathbb{Z}$ observation of the landscape at a pixel supported on $[n - \frac{1}{2}, n + \frac{1}{2}]$

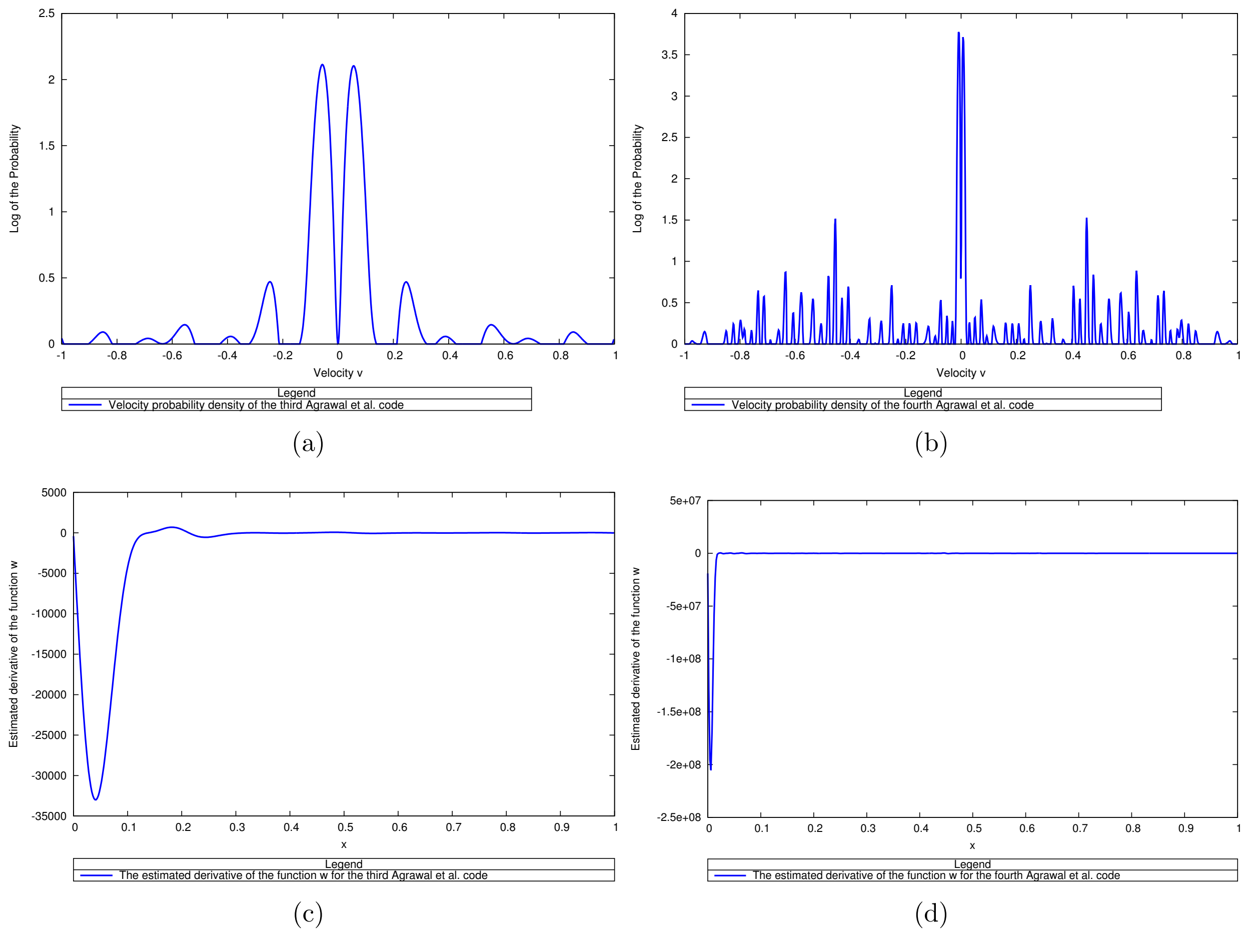


Figure 7: The velocity probability densities $\rho$ associated with Agrawal, Raskar et al. codes. $X$-axis: the velocity (in signed pixels per $\Delta t$)), $y$-axis: the logarithm of the velocity probability densities $(\log(1+\rho(v)))$. On the left panel: the code published in [2, p. 2566]. On the right panel: the code published in [1, p. 5]. On figures 7(c) and 7(d) the estimation of the $w'$ function. (Recall that $w$ is defined in (7).) $X$-axis: $x$, $y$-axis the estimated $w'(x)$ from the flutter shutter code coefficients. Recall that Theorem 3 relies on this estimation. As discussed in Remark 5 Algorithm 2 implements (27). Therefore, figures 7(c) and 7(d) permit to see where $w$ increases.

**(xiii)** $v$ relative velocity between the scene and the camera (unit: pixels per $\Delta t$)

**(xiv)** $\alpha(t)$ piecewise constant or time continuous gain control function for the analog flutter shutter and numerical flutter shutter methods

**(xv)** $\rho(v)$ probability distribution for the relative camera-scene velocities. Assumption: $\rho(v) = 0$ for any $v$ such that $|v| > |v_{max}|$

**(xvi)** $w(x) \geqslant 0$ weight function associated with the probability distribution $\rho$

**(xvii)** $\alpha^*$ optimal flutter shutter function

**(xviii)** $\|f\|_{L^1(\mathbb{R})} = \int |f(x)|dx$, $\|f\|_{L^2(\mathbb{R})} = \sqrt{\int |f(x)|^2 dx}$

**(xix)** $\mathbb{1}_{[a,b]}$ indicator function of an interval $[a,b]$

**(xx)** Let $f, g \in L^1(\mathbb{R}) \cup L^2(\mathbb{R})$, then

$$\mathcal{F}(f)(\xi) := \hat{f}(\xi) := \int_{-\infty}^{\infty} f(x)e^{-ix\xi}dx$$

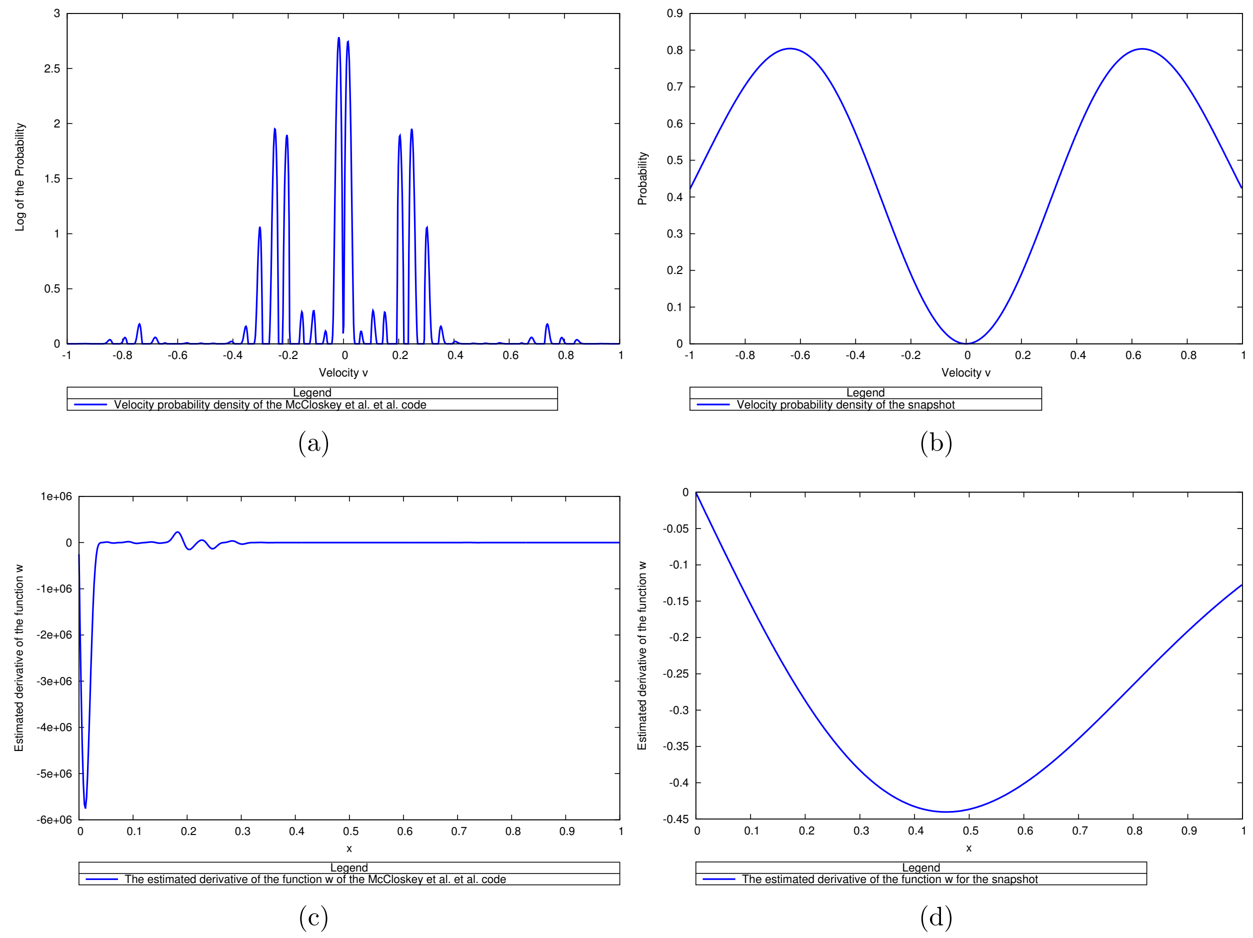


Figure 8: On the left panel: the velocity probability density $\rho$ associated with the McCloskey code [7, p. 321]. $X$-axis: the velocity (in signed pixels per $\Delta t$), $y$-axis: the logarithm of the distribution ($\log(1+\rho(v))$). It has a high probability of not moving and two small charges for two relatively small but non zero velocities. On the right panel: the probability density of velocities associated with a "standardized" snapshot with aperture function $\mathbb{1}_{[0,1]}$. On the $x$-axis: the velocity (in signed pixels per $\Delta t$). On the $y$-axis (not log scaled) the corresponding probability density. This snapshot is optimized a priori for objects moving at velocity $|v| \approx 1$. This bimodal density is quite natural for, e.g. a traffic surveillance camera. On figures 8(c) and 8(d) the estimation of the $w'$ function. (Recall that $w$ is defined in (7).) $X$-axis: $x$, $y$-axis the estimated $w'(x)$ from the flutter shutter code coefficients. Recall that Theorem 3 relies on this estimation. As discussed in Remark 5 Algorithm 2 implements (27). Therefore, figures 8(c) and 8(d) permit to see where $w$ increases.

$$\mathcal{F}^{-1}(\mathcal{F}(f))(x) := \widetilde{\mathcal{F}(f)}(x) = f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathcal{F}(f)(\xi)e^{ix\xi}d\xi$$

Moreover $\mathcal{F}(f * g)(\xi) = \mathcal{F}(f)(\xi)\mathcal{F}(g)(\xi)$ and (Plancherel)

$$\int_{-\infty}^{\infty} |f(x)|^2 dx = \|f\|^2_{L^2(\mathbb{R})} = \frac{1}{2\pi}\int_{-\infty}^{\infty} |\mathcal{F}(f)|^2(\xi)d\xi = \frac{1}{2\pi}\|\mathcal{F}(f)\|^2_{L^2(\mathbb{R})}$$

**(xxi)** $\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x} = \frac{1}{2\pi}\mathcal{F}(\mathbb{1}_{[-\pi,\pi]})(x) = \mathcal{F}^{-1}(\mathbb{1}_{[-\pi,\pi]})(x)$

## Acknowledgement

Yohann Tendero is happy to thank Jérôme Darbon for its careful reading of drafts. In addition, Yohann Tendero thanks the one who knocks and the Betches family. Yohann Tendero is partially supported on the grant ONR No: N000141410683. Part of this work was done at the UCLA Math. dept.

## Image Credits

Hervé Bry, Flickr CC-BY-NC-SA (http://www.flickr.com/photos/setaou/2162752903/.)

## References

[1] A. Agrawal and R. Raskar, *Resolving Objects at Higher Resolution from a Single Motion-blurred Image*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1–8. http://dx.doi.org/10.1109/CVPR.2007.383030.

[2] ——, *Optimal single image capture for motion deblurring*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 2560–2567. http://dx.doi.org/10.1109/CVPRW.2009.5206546.

[3] A. Agrawal and Y. Xu, *Coded exposure deblurring: Optimized codes for PSF estimation and invertibility*, in Proceddings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 2066–2073. http://dx.doi.org/10.1109/CVPRW.2009.5206685.

[4] G. Boracchi and A. Foi, *Uniform Motion Blur in Poissonian Noise: Blur/Noise Tradeoff*, IEEE Transactions on Image Processing, 20 (2011), pp. 592–598. http://dx.doi.org/10.1109/TIP.2010.2062196.

[5] R.L. Burden and J.D. Faires, *Numerical Analysis*, no. vol. 1 in Numerical Analysis, Brooks/Cole, 2001.

[6] H. Kozuka, *Image sensor*, Oct. 29 2002. US Patent 6,473,538.

[7] S. McCloskey, *Velocity-Dependent Shutter Sequences for Motion Deblurring*, in Proceedings of the Springer-Verlag European Conference on Computer Vision (ECCV), 2010, pp. 309–322. http://dx.doi.org/10.1007/978-3-642-15567-3_23.

[8] S. McCloskey, J. Jelinek, and K.W. Au, *Method and system for determining shutter fluttering sequence*, Apr. 9 2009. US Patent 12/421,296.

[9] K. Nakamura, H. Ohzu, and I. Ueno, *Image sensor in which reading and resetting are simultaneously performed*, Nov. 16 1993. US Patent 5,262,870.

[10] R. Raskar, *Method and apparatus for deblurring images*, July 13 2010. US Patent 7,756,407.

[11] R. Raskar, A. Agrawal, and J. Tumblin, *Coded exposure photography: motion deblurring using fluttered shutter*, ACM Transactions on Graphics (TOG), 25 (2006), pp. 795–804. http://dx.doi.org/10.1145/1141911.1141957.

[12] R. Raskar, J. Tumblin, and A. Agrawal, *Method for deblurring images using optimized temporal coding patterns*, Aug. 25 2009. US Patent 7,580,620.

[13] Y. Tendero, *The Flutter Shutter Camera Simulator*, Image Processing On Line, 2 (2012), pp. 225–242. http://dx.doi.org/10.5201/ipol.2012.t-fscs.

[14] Y. Tendero and J.-M. Morel, *The Flutter Shutter Paradox*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 813–847. http://dx.doi.org/10.1137/120880665.

[15] ——, *A Theory of Optimal Flutter Shutter For Probabilistic Velocity Models*, Submitted, UCLA CAM Report 13-80, (2014), p. 36.