A Stochastic Film Grain Model for Resolution-Independent Rendering

A Newson, Julie Delon, B Galerne

Volume xx (200y), Number z, pp. 1–15

A Stochastic Film Grain Model for Resolution-Independent Rendering

A. Newson, J. Delon and B. Galerne

Laboratoire MAP5, Université Paris Descartes and CNRS, Sorbonne Paris Cité, France

Abstract

The realistic synthesis and rendering of film grain is a crucial goal for many amateur and professional photographers and film-makers whose artistic works require the authentic feel of analog photography. The objective of this work is to propose an algorithm that reproduces the visual aspect of film grain texture on any digital image. Previous approaches to this problem either propose unrealistic models or simply blend scanned images of film grain with the digital image, in which case the result is inevitably limited by the quality and resolution of the initial scan. In this work, we introduce a stochastic model to approximate the physical reality of film grain, and propose a resolution-free rendering algorithm to simulate realistic film grain for any digital input image. By varying the parameters of this model, we can achieve a wide range of grain types. We demonstrate this by comparing our results with film grain examples from dedicated software, and show that our rendering results closely resemble these real film emulsions. In addition to realistic grain rendering, our resolution-free algorithm allows for any desired zoom factor, even down to the scale of the microscopic grains themselves.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—

1. Introduction

The digital revolution has changed the way we process, store and view images. An image unit, the pixel, is simply a quantized number representing in some manner light or color intensity, and it can be stored and transmitted in a unique and unambiguous fashion. This leads to many practical advantages of digital over "analog" photography (reproduceability, robustness etc.). However, when it comes to creating images of high artistic and visual quality, a great many amateur and professional photographers and film-makers prefer to use analog photography, in other words images produced with silver-halide based processes. In a recent interview [Lac15] Edward Lachman, the director of cinematography of the film "Carol", made the choice to use 16mm film in order to capture the feel of a specific period. To take an even more extreme example, the recognized director Quentin Tarantino declared in the 2014 Cannes film festival that "digital projection ... is the death of cinema as I know it" [Smi14]. Given the opinions of such prominent photographers and film-makers, it is not surprising that great efforts are made to recreate the "soul" of certain types of film emulsion.

Several factors contribute fundamentally to the feel of an analog film. Some of the main ones are image contrast, color palette and film grain. Film grain is the texture caused by the fact that analog photographs, mostly created by silver-halide type processes, are the result of many microscopic photo-sensitive particles reacting to light. These particles are called grains. Thus, on the microscopic scale, an analog image is binary : either a particle is present, in which case light is blocked, or it is not, in which case light is transmitted. Because humans can see with a limited resolution, what we perceive as an image is in reality a local average density of grains. The resulting visual aspect is often referred to as "graininess".

There are two main approaches to film grain synthesis. The first, which is commonly used in many commercial solutions such as DxO's FilmPack [DxO16], is to apply a stored example of film grain to the digital image. Advantages of this approach include speed and simplicity. However, the results will necessarily be *deterministic*, that is to say that if we apply the synthesis twice to the same image, we obtain the same output. This is clearly a considerable disadvantage for the synthesis of a random phenomenon, and will be particularly visible if the algorithm is applied to video sequences. Furthermore, this approach is completely reliant on the resolution and quality of the original scan. The second approach to film grain synthesis is to use a grain model. This is more frequently used in academic works [MRB06, Yan97, OLK09]. However, most of these models rely again on an example of scanned film grain for synthesis, which entails the same drawbacks as mentioned above. In the case where film grain is modelled as independent noise [MRB06, Yan97], with a variance which is dependent on the input image intensity, the grain texture is completely uncorrelated spatially, which gives a distinctly "digital" feel to the image. Indeed, this is one of the main criticisms of photographers and film-makers towards digital film grain synthesis methods. In Edward Lachman's interview he discusses this, stating that it is possible to

Full resolution | Zoom 4× | Zoom 8× | Zoom 20×

Figure 1: **Film grain rendering results with several zoom factors**. We propose a stochastic film grain model and a film grain rendering algorithm which can render film grain on a digital image at any chosen resolution.

"recreate grain digitally now, but it is pixel-fixated. It does not have this anthropomorphic quality in which the grain structure in each frame is changing". Thus, the spatial correlation of film grain is one of its defining features.

In light of the drawbacks of the previous methods, we propose a physically motivated film grain model and a synthesis algorithm to produce the grain texture for a given input image. To our knowledge, this is the first synthesis algorithm based on a physical model of the photographic process. This careful modeling leads to realistic and aesthetically pleasing results. In particular, the visual characteristics of our grain are dependent on the image grey-level. This is not the case in other approaches, which either scan or learn the grain at a fixed grey-level.

The contributions of this work are:

1. a resolution-free model of film grain based on the physical process of silver-halide photography;
2. a Monte-Carlo based algorithm to render a given image with film grain at a chosen resolution;
3. tunable parameters based on the physical characteristics of the film grain, such as the grain size and size distribution;

The C++ implementation of the proposed algorithm is freely available online [NGD17]

Our film grain model is continuous, and we produce a discrete image during the last step of the algorithm, once the photographic process has been imitated. This is a significant advantage, as it results in a resolution-free algorithm which can "zoom" indefinitely on the image, until the individual grains are visible, as illustrated in Figure 1. This is not possible for either the methods based on scanned examples of grain, as the resolution is fixed, or those which model film grain as independently distributed noise. In approaches using scanned film grain, zooming is either achieved by "stretching" the grain or tiling; in both cases the results are visually unsatisfactory. Some examples of these problems may be found in the supplementary material.

2. Background and the photographic process

The literature concerning film grain and film grain synthesis is clearly separated according to whether they belong to the analog world or the digital world. Those in the former category are concerned exclusively with identifying visual and statistical characteristics of film grain. The digital category looks at both removal and synthesis of film grain. From one point of view, film grain may be considered as a kind of noise, and thus should be removed. However, denoised images tend to be too smooth and not visually pleasing, and therefore an effort is also made to try and recreate the grain [OLK09]. Thus, many works of the digital category try to provide both possibilities. Unfortunately, to the best of our knowledge none of these methods take account of the analog literature. One of our goals is to produce a digital algorithm based on the modeling of the physical photographic process studied in the analog literature.

2.1. Previous work

The silver-halide photographic process has been extensively studied since the beginning of the twentieth century. Gurney and Mott [Gur38] proposed a comprehensive physical model of the process, which is widely accepted. Nutting [Nut13] was the first to study the statistical properties of the so-called "random dot model" for film grain, and proposed the Nutting formula which links the optical density of a film emulsion to the average number of grains present and to the size of the inspected region. An important quantity studied in the analog literature is *granularity* or root-mean-square (rms) granularity. This is experimentally measured using a microdensitometer on any given film emulsion after development, and corresponds to the standard deviation of the optical density of the emulsion. Another useful connected quantity is that of Selwyn granularity [Wer94], which is basically the granularity defined in such a fashion that it is independent of the aperture size. A good summary of these basic notions may be found in the paper of Bayer [Bay64] in the context of the random dot model. A particularly hot topic concerning film grain is that of grain "clumping". This corresponds to the perceived clustering of film grain. Much of the subsequent analog literature is concerned with proposing mathematical models which imitate this effect [CKT73, LTW72, TU83].

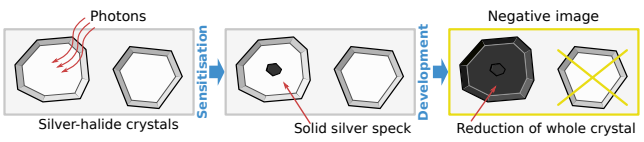
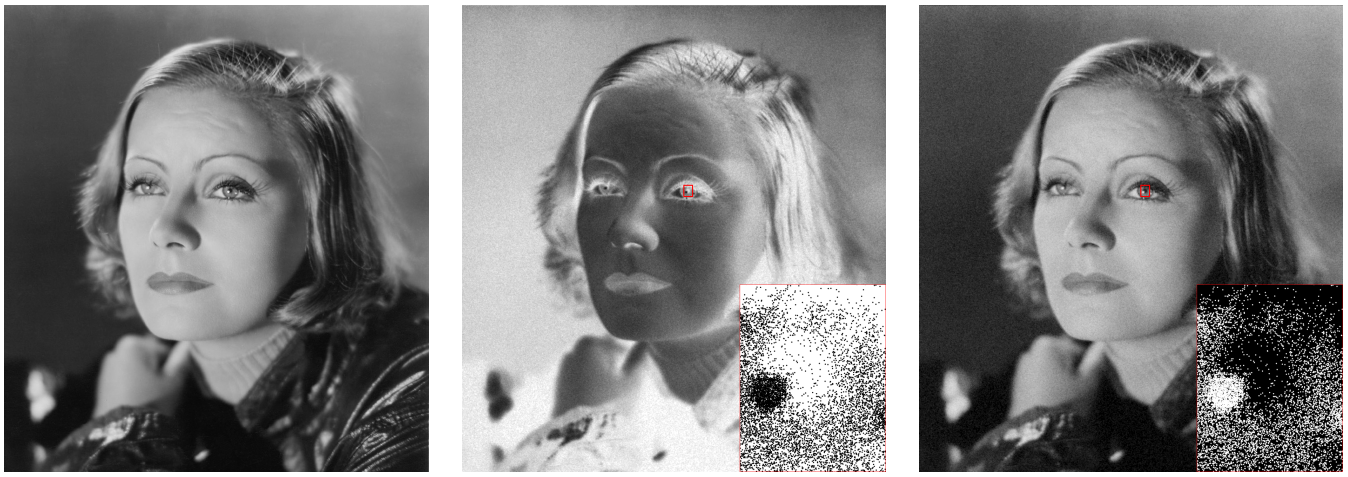


Figure 2: **Illustration of the physical photographic process.** The sensitization and development processes are shown here, with one crystal interacting with a photon and subsequently being developed, while the other crystal remains undeveloped and transparent.



Real-world image | Negative analog image | Positive analog image

Figure 3: **An illustration of the different steps of the photographic process**. The signal from the real-world image is recorded on the film negative with dark film grains on a light background. This is inverted in the positive image.

Many approaches use actual scanned examples of film grain for the purposes of synthesis. This appears to be the most popular approach in the industrial environment as well as in some academic approaches. Film grain synthesis products such as Dxo's "FilmPack" [DxO16] and Grubba Software's "TrueGrain" [Gru15] tools take this approach. More precisely, a single grain image is saved for each film type. In the same philosophy, Schallauer and Mörzinger [SM06] extract the grain pattern from real images of grain and synthesize a new grain image from these examples, which they then apply to the image in an additive fashion. Unfortunately, they do not go into detail as to how this synthesis is carried about. A similar approach is used in the Film Emulation feature of the G'MIC free software [G'M16] using the random phase texture algorithm [GGM11] to synthesize large grain textures from small stored samples. Bae et al. [BPD06] use the classical Heeger-Bergen texture synthesis [HB95] approach on a constant region in an example grainy image to produce film grain. However, they do not specify how this is applied to a given input image. Stephenson and Saunders [SS07] filter white noise in the Fourier domain. Yan et al. [Yan97] proposed an additive film grain model with signal-dependent noise. A drawback is that their approach supposes that film grain noise is spatially uncorrelated, which is clearly unrealistic for film grain noise. Oh et al. [OLK09] propose an auto-regressive model for film grain removal and synthesis. They point out that spatial correlation is crucial for producing realistic film grain. However, they consider that an input grainy image is available, and that the characteristics of the grain may be extracted. We also note that other works have looked at simulating various photographic processes [GK97, EWK*13], but these are not concerned with simulating physical film-grain.

A common drawback of the approaches of the digital literature is that no model based on the physical reality of film grain is proposed. Either a digital example of film grain, with fixed resolution, is considered to be available (which may not always be possible), or spatial correlation of the film grain texture is not considered. Furthermore, even if a good example of film grain is available, it is not obvious how to blend this grain with an input image. In this work, we propose a realistic film grain model based on physical considerations which requires no example for synthesis, and which does not need any such blending process.

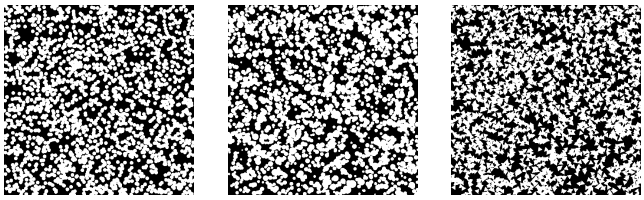
2.2. The photographic process

The photographic process is based on two steps : film grain sensitization and development. Sensitization takes place when silver halide crystals are exposed to light for a certain amount of time and made "developable" by the interaction with incoming photons. Development is the process which turns the sensitized crystals into solid grains of silver. These steps produce a negative image, which is then converted to a positive image with a second photographic process. Since the grain blocks light, the negative photographic image is in fact a binary function which is equal to 0 in the areas covered by the grains, and equal to 1 otherwise. The sensitization and development processes are illustrated in Figure 2.

From negative to positive The final positive image takes form on photographic (photosensitive) paper. To do this, light is shone through the negative film onto the photographic paper. During this step, the image is typically enlarged by a factor of between five and ten times. After the exposure of the photographic paper, a *positive* representation of the original image has been recorded. As a simplification, we shall consider that the photographic paper is a continuous recording material, even if the photographic paper can contain its own "grain". An illustration of the negative and positive images can be seen in Figure 3.

3. Review of the Boolean model

We wish to model the photographic process in order to produce a realistic image with film grain. From the previous section, we know that a photographic image is made up of microscopic grains of solid silver. The simplest way of modeling this is to consider that the



Constant radius Random radius Triangular grains

Figure 4: **Illustration of the Boolean model**. Three different Boolean models with balls of constant radius (left), balls of random radius following a log-normal distribution (middle), and randomly oriented triangles. Observe that the models display "groups" of grains, as in real film grain.

grains are convex sets which are uniformly distributed in the film emulsion. This model is implicitly used in much of the "analog" literature concerning film grain [Nut13, Sel35]. This model, with a few further hypotheses, corresponds very nicely to a well studied model from the stochastic geometry literature [CSKM13], known as the *Boolean* model.

3.1. The Boolean model

Let us first recall basic definition and properties regarding the homogeneous Boolean model in $\mathbb{R}^2$ which is the most natural and the most studied example of homogeneous random sets [CSKM13]. Let $\Phi = \{x_i,\ i \in \mathbb{N}\}$ represent a Poisson process on $\mathbb{R}^2$ with intensity λ. These x_i represent the centers of our grains. We also define a sequence of identically and independently distributed (i.i.d.) random compact sets in $\mathbb{R}^2$, $X_0, X_1, \ldots$, which will represent the grain shapes. The Boolean model is the random set Z defined as the union of all the shapes X_i placed at the locations x_i, that is,

$$Z = \bigcup_{i \in \mathbb{N}} (X_i + x_i).$$

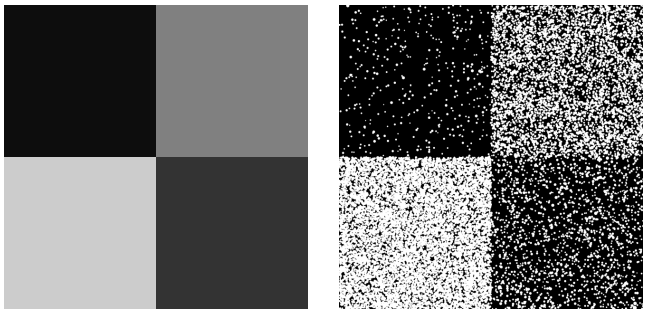
This is a particularly flexible model, as we can choose any sort of grain shape and size. In practice we shall use 2D *balls*, in which case $Z = \bigcup_{i \in \mathbb{N}} \mathcal{B}(x_i, r_i)$, where r_i is the (possibly random) radius of the i^{th} ball. We also present some experiments with other shapes in the supplementary material. Finally let us define the indicator function of the Boolean model Z as the function $\mathbb{1}_Z(y)$ that equals 1 if $y \in Z$ and 0 otherwise.

Let $A_i = \pi r_i^2$ stand for the area of the ball indexed by i. Given the Poisson assumption, the volume fraction of the Boolean model is given by

$$\mathbb{P}(\mathbb{1}_Z(y) = 1) = 1 - \exp(-\lambda \mathbb{E}[A_1]), \tag{1}$$

where $\mathbb{E}[A_1] = \pi \mathbb{E}[r_1^2]$ is the common mean area of the i.i.d. balls, and $y \in \mathbb{R}^2$. If the grains X_i are balls of constant radius r, we have $\mathbb{P}(\mathbb{1}_Z(y) = 1) = 1 - \exp(-\lambda \pi r^2)$. To summarize, the Boolean model consists of "white" balls on a "black" background, and we use this model to represent the physical reality of film grain.

Figure 4 shows examples of three different Boolean models. An important point to note is the well-known tendency of the model to produce the visual effect of clustering, or "clumping", which is crucial to producing realistic film grain. Now, in a film emulsion, there will be a varying density of developed grains, which means the Boolean model as we have presented it is not sufficient yet. In stochastic geometry, this varying density corresponds to the *inhomogeneous* Boolean model, which we present now.



Input pixels (close-up) Inhomogeneous Boolean model

Figure 5: **Illustration of inhomogeneous Boolean model.** The local intensity $\lambda(y)$ of the inhomogeneous Boolean model is chosen to respect the input pixel gray-levels.

3.2. The inhomogeneous Boolean model

As in the homogeneous case, the inhomogeneous Boolean model is built upon a sequence of random positions $\Phi = \{x_i, i \in \mathbb{N}\}$ given by a Poisson process. However, the intensity λ of the Poisson process Φ is no longer constant. It is given by a function $\lambda(y)$ which varies spatially with $y \in \mathbb{R}^2$. When the intensity function λ is bounded from above, one can interpret and simulate such an inhomogeneous Poisson process by thinning an homogeneous Poisson process having a large intensity [CSKM13] (which corresponds to a rejection method for Poisson processes). However, in what follows, we will only consider intensity functions λ that are piecewise constant and thus the corresponding Poisson point process can be easily simulated in a piecewise manner.

Now that we have given a brief description of the Boolean model, we proceed to see how it can be used to provide realistic film grain rendering for a given image.

4. Stochastic film grain model and rendering algorithm

As recalled in Section 2.2, at a microscopic level a positive analog photograph is a binary set of white grains in a black background. Given an input digital image u, we will define an inhomogeneous Boolean model such that, when seen at a distance, the corresponding random binary set represents the same image as u, but with the additional graininess which characterizes analog photography. In technical terms, we need to define a varying intensity function λ from an input image u. Also, we will not make any assumption on the grain radius distribution since, as results show, this is a meaningful parameter for tuning the visual aspect of the grain texture.

Let $u : \{0, \ldots, m-1\} \times \{0, \ldots, n-1\} \subset \mathbb{N}^2 \to [0, u_{max}]$ be the input image, of size $m \times n$. We start by normalizing the input image u to the interval $[0, 1)$ by defining $\tilde{u}(y) = \frac{u(y)}{u_{max} + \varepsilon}$, where ε is a small parameter, and u_{max} is the maximum possible gray-level value. We

restrict the image to $[0,1)$, as a Boolean model with $\mathbb{P}(\mathbb{1}_Z(y) = 1) = 1$ would require a degenerate infinite intensity λ.

4.1. Stochastic film grain model

Ideally, we would like to imitate the photographic process by choosing $\lambda(y)$ to reflect the physical concentration of grains in the emulsion. Unfortunately, for a given input image, we cannot necessarily know how the image was taken and therefore we do not have access to the average number of photons received.

Our solution to this problem is to define $\lambda(y)$ so that the *average area* covered by the balls of the Boolean model within a pixel location equals the input image gray-level. Consequently, the global contrast of the input image is correctly maintained. A significant advantage of this approach is that contrast changes can be handled independently from the rendering of film grain.

Therefore, using Equation (1), we set λ to be the piecewise constant function defined for all $y \in [0,m) \times [0,n)$ by

$$\lambda(y) = \frac{1}{\mathbb{E}[A_1]} \log\left(\frac{1}{1-\tilde{u}(\lfloor y \rfloor)}\right), \tag{2}$$

where $\lfloor y \rfloor$ are the coordinates of the pixel containing the point y. This provides us with the means to simulate the inhomogeneous Boolean model for any given input image. However, this model represents the positive image viewed at infinite resolution. In reality, we perceive images with finite resolution. Furthermore, we wish to produce an output digital image defined at a chosen resolution. This requires a filtering step, which we explain further now.

4.2. Filtering the positive image

As explained above we do not view the analog image with infinite resolution; indeed if we could, all such pictures would be binary ! In reality, we observe a filtered and sampled version that reveals the gray-levels of the image. To this end, we add a last step which imitates this effect by filtering and sampling $\mathbb{1}_Z$. This step is in fact essential for viewing a gray-level image and also in creating the "grainy" effect.

There are several origins of this filtering effect. One of these is the optical filtering which takes place during the transition from negative to positive. A second unavoidable filtering is that of the human visual system. To illustrate how this affects our model, let us denote a first filter with ψ acting on the negative image, and another filter ψ' acting on the positive image (which we perceive). We consider that $\int_{-\infty}^{\infty} \psi(y)dy = 1$, and similarly for ψ'. Thus, the resulting gray-level $v(y)$ perceived at position y is

$$v(y) = \left(\psi' * [1 - \psi * (1 - \mathbb{1}_Z)]\right)(y) = (\psi' * \psi * \mathbb{1}_Z)(y). \tag{3}$$

The upshot of this is that we can apply any sort of filter we like, independently, and a posteriori, to our Boolean model. This is particularly practical, as we can separate the creation of the Boolean model from the application of the filter. In our experiments, we simply apply a single Gaussian low-pass filter to represent the combined blurring steps from the negative image to the perceived image. To summarize, our continuous film grain model consists of a filtered indicator function of the inhomogeneous Boolean model.

Algorithm 1 Sampling of the inhomogeneous Boolean model from an input image.

Data: $u : \{0,1,\dots,m-1\} \times \{0,1,\dots,n-1\} \to [0,u_{max}]$ (input image)
Parameters:
$\mathcal{D}(\mu_r, \sigma_r^2)$: distribution of grain radii
Result:
x: List of grain centers
r: List of grain radii

Sample Boolean model within the whole image domain:
$x \leftarrow \emptyset$, $r \leftarrow \emptyset$
foreach $(i,j) \in \{0,\dots,m-1\} \times \{0,\dots,m-1\}$ **do**
  Convert gray-level to the interval $[0,1)$:
  $\tilde{u}(i,j) = \frac{u(i,j)}{u_{max}+\varepsilon}$
  Compute local value of intensity λ:
  $\lambda = \frac{1}{\pi(\mu_r^2+\sigma_r^2)} \log \frac{1}{(1-\tilde{u}(i,j))}$
  Draw the number of grains Q in the square $[i,i+1) \times [j,j+1)$:
  $Q \leftarrow \text{Poisson}(\lambda)$
  Sample $x_{i=1\dots Q}$ from $\mathcal{U}\left([i,i+1) \times [j,j+1)\right)$
  Sample grain radii $r_{i=1\dots Q} \sim \mathcal{D}(\mu_r, \sigma_r^2)$
  Add the new points to the list:
  $x \leftarrow x \cup x_{i=1\dots Q}$; $r \leftarrow r \cup r_{i=1\dots Q}$

4.3. Film grain rendering algorithm

Equation (3) gives us a theoretical model of the continuous photographic image $v(y)$ which we perceive. However, we also wish to render this image at any desired resolution, which is a non-trivial task. Accordingly, we now present our film grain rendering algorithm, which consists of the two following steps:

- sampling of the inhomogeneous Boolean model;
- evaluation of the filtered inhomogeneous Boolean model;

4.3.1. Realization/sampling of the inhomogeneous Boolean model

Firstly, we wish to produce a *realization* of the inhomogeneous Boolean model, in other words we wish to sample the centers and radii of the grains throughout the image.

Since the intensity function λ (2) of our inhomogeneous Boolean model is constant on each pixel (represented by unit squares $[i,i+1) \times [j,j+1)$), the Poisson process of the centers of our inhomogeneous model can be partitioned into the disjoint union of $m \times n$ Poisson processes having their points in their respective pixel square $[i,i+1) \times [j,j+1)$. Then, within a pixel square $[i,i+1) \times [j,j+1)$, the intensity is constant and given by (2) with $y = (i,j)$, and one can simulate the centers using the standard Poisson process simulation. This consists in drawing the number of points Q according to a Poisson distribution with parameter $\lambda(i,j)$ and then drawing Q grain centers x_i from a uniform distribution $\mathcal{U}([i,i+1) \times [j,j+1))$ and Q independent radii r_i from the radius distribution (in practice a constant distribution or a log-normal distribution). This method is described in Algorithm 1. Note that this pseudo-code describes our algorithm in the case where the grains

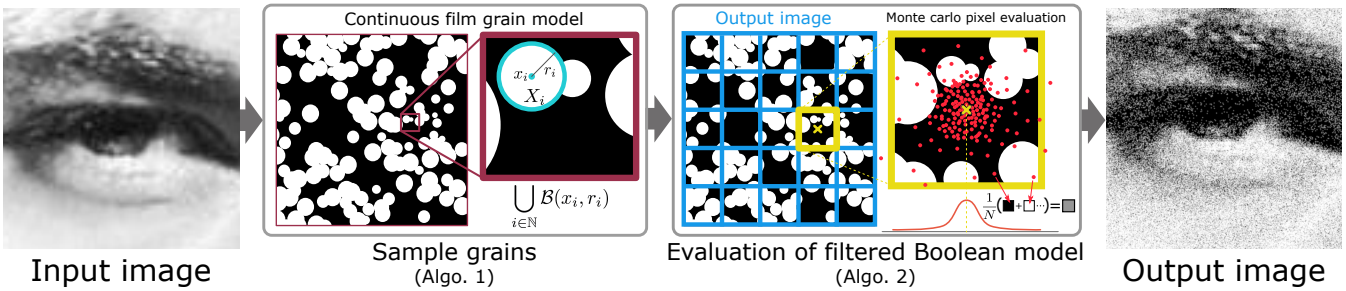


Figure 6: **Illustration of the steps of our film grain rendering algorithm**.

are balls of possibly random size, but it can be modified to include arbitrary shapes, as we shall describe in Section 4.4.

Algorithm 2 Evaluation of an inhomogeneous Boolean model with Monte Carlo simulation.

Data: $x_{i=1\dots Q}$, $r_{i=1\dots Q}$ sampled inhomogeneous Boolean model, with a total of Q grains
Parameters:
s : output zoom
σ : standard deviation of the Gaussian low-pass filter
N : number of iterations in the Monte Carlo method
Result: v : Rendered film grain image

Initialize the output image to 0:
$v = 0$
for $k = 1$ **to** N **do**
  Draw a random offset from a centered Gaussian distribution of variance σ^2:
  $\xi_k \leftarrow \mathcal{N}(0, \sigma^2 I_2)$
  for $\ell = 1$ **to** Q **do**
    $y = sx_\ell + \xi_k$
    foreach $(a,b) \in \{0,\dots,sm-1\} \times \{0,\dots,sn-1\}$ *s. t.* $\|y-(a,b)\|_2 \leq sr_\ell$ **do**
      $v(a,b) = v(a,b) + 1$
Average the contributions:
$v = \frac{1}{N} v$
return(v)

4.3.2. Evaluation of the inhomogeneous Boolean model

We now have a list of grain centers in a continuous space. The final step in our algorithm is to evaluate the filtered inhomogeneous Boolean model. Note that the output discretization grid need not necessarily be the same as that of the input image; our algorithm can freely zoom in or out on the model created in Section 4.3.1. This gives considerable flexibility to our algorithm.

Now, we cannot actually perform the continuous convolution described by (3) due to computational limitations. However, it is possible to approximate the integral required by the convolution using Monte Carlo simulation. We first define a scalar N representing the number of samples in the Monte Carlo simulation. For each output position y, we draw a list of offsets $\{\xi_i, i = 1\dots N\}$ whose row-column coordinates follow a Gaussian distribution $\mathcal{N}(y, \sigma^2)$. We produce the output pixel value using

$$v(y) = \frac{1}{N} \sum_{k=1}^{N} \mathbb{1}_Z(\xi_k). \tag{4}$$

As N increases, according to the law of large numbers :

$$\frac{1}{N} \sum_{k=1}^{N} \mathbb{1}_Z(\xi_k) \xrightarrow[N \to +\infty]{} \mathbb{E}(\mathbb{1}_Z(\xi_1)) = \int_{\mathbb{R}^2} \mathbb{1}_Z(t)\phi(y-t)dt, \tag{5}$$

where ϕ is the pdf of the Gaussian distribution $\mathcal{N}(0, \sigma^2 I_2)$, that is, the targeted Gaussian blur kernel. I_2 represents the identity matrix of size 2×2.

We denote with s the *zoom* factor of the output image, such that the dimensions of the latter is $sn \times sm$. Thus $\frac{1}{s}$ represents the output image grid discretization step, with respect to the unit square of the input. In simple terms, s represents the "zoom" of the output image resolution with respect to the input image. The pseudo-code for the Monte Carlo simulation is shown in Algorithm 2. We use the same set of random offsets for each pixel, which avoids the repeated use of random number generation, which can be slow.

4.4. Algorithmic details and parallelization

In Section 4.3, we presented the film grain rendering algorithm in two separate parts: the sampling of the inhomogeneous Boolean model (Algorithm 1), and the evaluation of the filtered model (Algorithm 2). A disadvantage of this method is that all the grain positions must be stored in memory and then processed. For example, if we suppose a grain radius $r = \frac{1}{40}$, with a high resolution image (2048×2048) with constant gray-level values of 128 everywhere, 35 GB of memory is needed to store the grain positions and radii, if the information is stored with single precision floating point.

We propose two algorithms which address this problem of storing the grain information. The first generates each grain once only, determines the effect of this grain on the output image, and then erases the grain information. We refer to this as the "grain-wise" approach (see Algorithm 3). Unfortunately, this algorithm is not well-adapted to parallelization on the GPU, due to excessive memory accesses. Therefore, we propose a second approach (Algorithm 4) which we refer to as the "pixel-wise" algorithm, which is suitable for GPU parallelization. We describe these approaches now.

Algorithm 3 The proposed "grain-wise" film grain rendering algorithm. The loop colored in blue is parallelized.

Data: $u : \{0,1,\dots m-1\} \times \{0,1,\dots,n-1\} \to [0, u_{max}]$: input image
Parameters:
$\mathcal{D}(\mu_r, \sigma_r^2)$: distribution of grain radii
s: output zoom
σ: standard deviation of the Gaussian low-pass filter
N: number of iterations in the Monte Carlo method
Result: v: Synthesized, film grain image

Set up N binary images of size ms × ns and draw N random offsets:
for $k = 1$ **to** N **do**
  $v_k = 0$
  $\xi_k \leftarrow \mathcal{N}(0, \sigma^2 I_2)$

foreach $(i,j) \in \{0,\dots,m-1\} \times \{0,\dots,n-1\}$ **do**
  $\tilde{u}(i,j) = \frac{u(i,j)}{u_{max}+\varepsilon}$
  $\lambda = \frac{1}{\pi(\mu_r^2+\sigma_r^2)} \log \frac{1}{(1-\tilde{u}(i,j))}$
  $Q \leftarrow \text{Poisson}(\lambda)$
  Sample $x_{i=1\dots N}$ from $\mathcal{U}([i,i+1) \times [j,j+1))$
  Sample grain radii $r_{i=1\dots Q} \sim \mathcal{D}(\mu_r, \sigma_r^2)$
  for $k = 1$ **to** N **do**
    for $\ell = 1$ **to** Q **do**
      $y = s x_\ell + \xi_k$
      foreach $(a,b) \in \{0,\dots,sm-1\} \times \{0,\dots,sn-1\}$ *s. t.* $\|y-(a,b)\|_2 \le s r_\ell$ **do**
        $v_k(a,b) = 1$

foreach $(i,j) \in \{0,\dots,sm-1\} \times \{0,\dots,sn-1\}$ **do**
  $v(i,j) = 0$
  for $k = 1$ **to** N **do**
    $v(i,j) = v(i,j) + v_k(i,j)$
  $v(i,j) = \frac{1}{N} v(i,j)$

return(v)

Algorithm 4 The proposed "pixel-wise" film grain rendering algorithm. The loop colored in blue is parallelized.

Data: $u : \{0,1,\dots m-1\} \times \{0,1,\dots,n-1\} \to [0, u_{max}]$: input image
Parameters:
$\mathcal{D}(\mu_r, \sigma_r^2)$: distribution of grain radii
r_m: maximum radius allowed
s: output zoom
σ: standard deviation of the Gaussian low-pass filter
N: number of iterations in the Monte Carlo method
Result: v: Image rendered with film grain

$\delta = \frac{1}{\lceil \frac{1}{r_m} \rceil}$
foreach $(i,j) \in \{0,\dots,sm-1\} \times \{0,\dots,sn-1\}$ **do**
  $v(i,j) = 0$
  for $k = 1$ **to** N **do**
    $\xi_k \leftarrow \mathcal{N}(0, \sigma^2 I_2)$
    $(i_g, j_g) = \frac{1}{s}((i,j) + \xi_k)$

    Get the list of cells which might contain the balls covering (i_g, j_g) *:*

    foreach $(i_\delta, j_\delta) \in \{\lfloor \frac{i_g - r_m}{\delta} \rfloor, \dots, \lfloor \frac{i_g + r_m}{\delta} \rfloor\} \times \{\lfloor \frac{j_g - r_m}{\delta} \rfloor, \dots, \lfloor \frac{j_g + r_m}{\delta} \rfloor\}$ **do**
      $\tilde{u} = \frac{u(\delta.i_\delta, \delta.j_\delta)}{u_{max}+\varepsilon}$
      $\lambda = \frac{1}{\pi(\mu_r^2+\sigma_r^2)} \log \frac{1}{(1-\tilde{u})}$
      $Q \leftarrow \text{Poisson}(\lambda)$

      for $\ell = 1$ **to** Q **do**
        $x \leftarrow \mathcal{U}([i_\delta, i_\delta + 1) \times [j_\delta, j_\delta + 1))$
        $y = (\delta.i_g, \delta.j_g) + \delta.s$
        $r = \min(\mathcal{D}(\mu_r, \sigma_r^2), r_m)$
        if $||y - \xi||_2 < r$ **then**
          $v(i,j) = v(i,j) + 1$
          Break: go to next Monte Carlo iteration
  $v(i,j) = \frac{1}{N} v(i,j)$
return(v)

4.4.1. Grain-wise algorithm

As previously mentioned, this approach samples the grains sequentially for each pixel, and evaluates the effect of each grain on each Monte Carlo iteration. Once a grain has been generated and processed, its information (position and radius) are erased from memory. Instead of saving the grain information, we store a sequence of N *binary* images v_k, $k \in \{1 \dots N\}$, with each image corresponding to the result of one Monte Carlo iteration. A Monte Carlo iteration consists in evaluating the Boolean model on the randomly shifted grids $\xi_k + \{0,\dots,sm-1\} \times \{0,\dots,sn-1\}$, with $\xi_k \sim \mathcal{N}(0, \sigma^2 I_2)$. Finally, the result of our algorithm is simply the average of all the temporary images v_k. An advantage of this approach is that the memory requirement is independent of the grain size, and only depends on the output image size and the number of Monte Carlo iterations N. This algorithm can be seen in Algorithm 3.

4.4.2. Pixel-wise algorithm

The second algorithm we present here, which we refer to as the "pixel-wise" approach, also avoids storing the grain information, but in quite a different manner from the grain-wise approach. The main reason for proposing another algorithm is that memory accesses should be limited when using the GPU. Therefore, we cannot save a large number of intermediate images required by the Monte Carlo simulation of Algorithm 3. Instead, we employ an on-the-fly Poisson process generation often used in the procedural noise literature [LLC*10]. This type of approach consists in using a grid partition of the space $\mathbb{R}^2$ and generating the Poisson process on-the-fly within each partition cell using a local pseudo-random number generator [Wor96, LLDD09]. Given a coordinate pair of a given partition cell, the pseudo-random number generator can generate the number of grains whose centers belong to this cell, as well

	Image size			
	256 × 256	512 × 512	1024 × 1024	2048 × 2048
Grain-wise, non-para.	168.815 s	676.502 s	2718.92 s	10843.0 s
Grain-wise, para. (CPU)	10.749 s	40.992 s	165.433 s	654.696 s
Pixel-wise, non-para.	9.207 s	36.567 s	147.687 s	584.496 s
Pixel-wise, para. (CPU)	0.732 s	2.430 s	9.499 s	37.786 s
Pixel-wise, para. (GPU)	0.137 s	0.429 s	1.275 s	4.534 s
# grains processed	3.58×10^6	14.3×10^6	52.3×10^6	229×10^6

Table 1: **Algorithm execution times**. In this Table, we show the execution times for our algorithms for different image sizes. We also note the total number of grains which need to be processed for each image. The images used are of increasing sizes, with a constant gray-level of 128. The grain radius is set to $r = 0.05$ pixels for all grains, and we use $N = 800$ Monte Carlo samples.

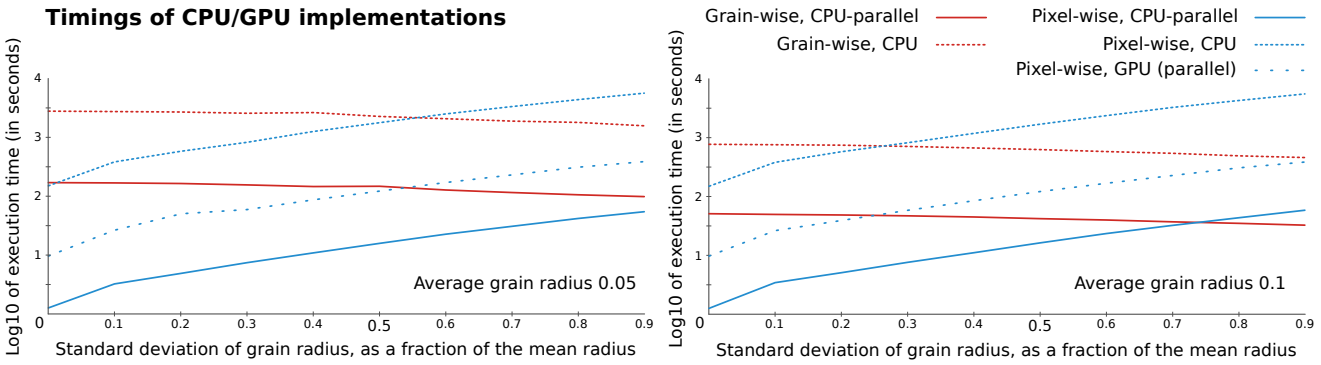


Figure 7: **Execution times of the proposed algorithms with increasing radius standard deviation.** The "grain-wise" approach is implemented with and without parallelization on the CPU, and the "pixel-wise" approach is implemented with and without parallelization on the CPU, and with parallelization on the GPU. The size of the image used is 1024x1024, with a constant grey-level of 128.

as the positions of these centers. The numbers given by the pseudo-random number generator are completely reproducible, so there is no need to store the information pertaining to the grains. Note that the cell size δ must be a fraction of the input image pixel size.

The main task is to evaluate $\mathbb{1}_Z(y)$ at $y \in \mathbb{R}^2$ for each Monte Carlo sample. This is equal to 1 if there is a point x_i such that $y \in \mathcal{B}(x_i, r)$. Therefore, one only needs to simulate the Poisson process in cells intersecting the ball of radius r centered at y to evaluate $\mathbb{1}_Z(y)$. Unfortunately, this approach is not valid when using random radii given by a distribution producing unbounded variates, such as the log-normal distribution. In this case, we specify a maximal value of the radii, r_m. Therefore, we need to check more cells in order to evaluate $\mathbb{1}_Z(y)$, and this number of cells obviously increases quadratically with a linearly increasing maximum radius. This approach is described in detail in the pseudo-code of Algorithm 4.

4.5. Performance comparisons

We have extensively tested both the grain-wise and pixel-wise algorithms in different situations in order to identify the speedups which are achieved. We have tested the following implementations:

- Grain-wise, no parallelization;
- Grain-wise, with parallelization (OpenMP) on a multi-core CPU;
- Pixel-wise, no parallelization;
- Pixel-wise, with parallelization (OpenMP) on a multi-core CPU;
- Pixel-wise, with parallelization on a GPU.

The machine used for these tests has four Intel Xeon 2.00 GHz processors, each with ten cores (for the purposes of parallelization on the CPU). The pixel-wise algorithm was implemented on a GPU in CUDA using an Nvidia Tesla T10 graphics card. This GPU implementation is based on the publicly available source code of [GLM16].

Table 1 shows the execution times for our algorithms. For these experiments, we rendered film grain on images of increasing sizes, whose gray-level values are equal to 128 everywhere. We set the grain radius to $r = 0.05$ pixel. We have shown the execution times of the parallelized versions of our code, and show our execution times with and without this acceleration. It can be seen that, for a fixed constant gray-level, the complexity of our algorithm is linear with respect to the number of pixels in the input image. We observe that it is possible to achieve interactive execution times with the GPU implementation of our algorithm in the fixed-radius case.

In Figure 7, we analyze the execution times of our algorithms when the grain radii are variable. As in the rest of the paper, the distribution of the radii is a log-normal distribution. The standard de-

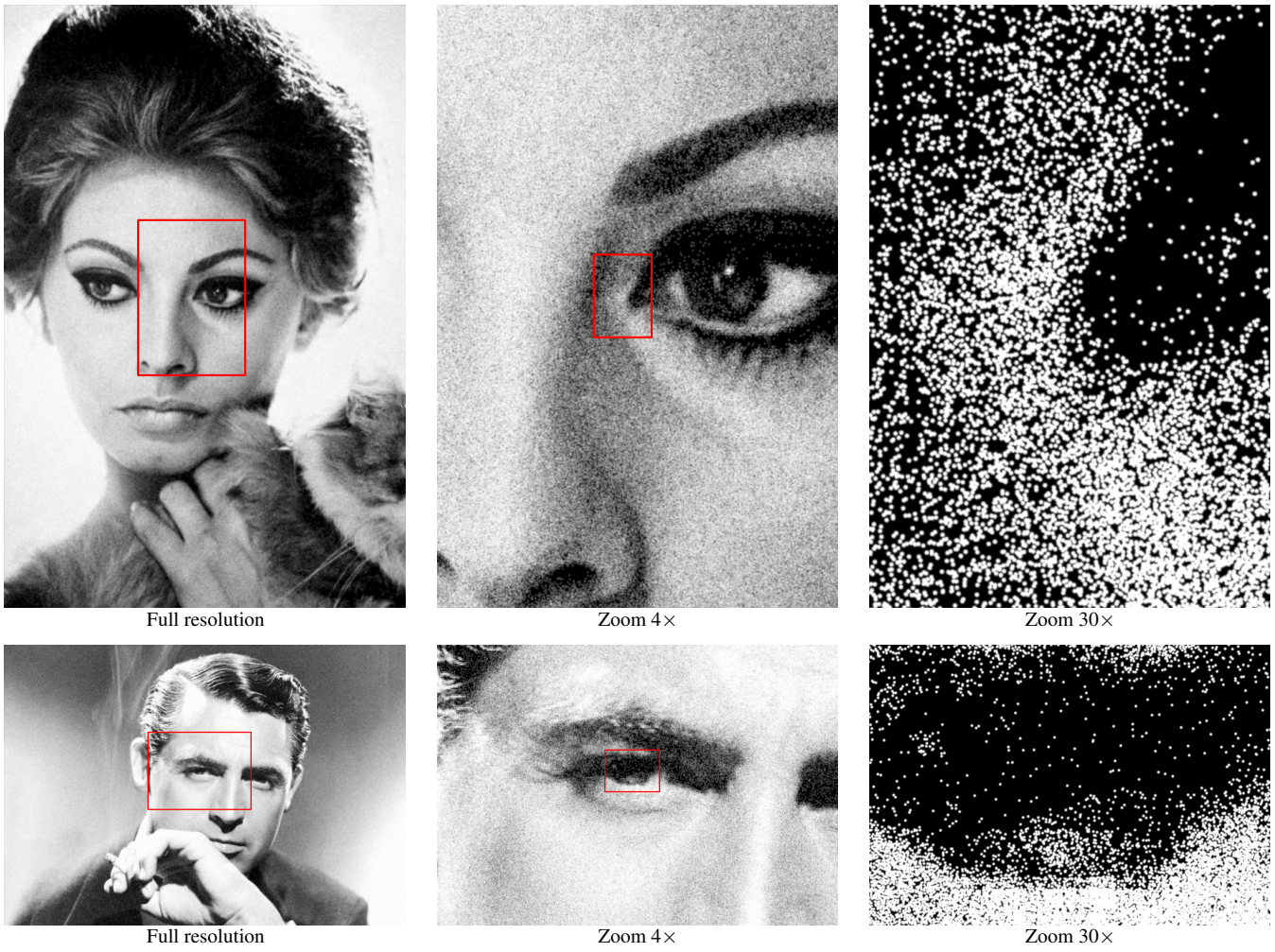


Figure 8: **Film grain rendering results on vintage images**. Our model and rendering algorithm allows us to create film grain at any desired resolution, which is not possible with any other grain synthesis approach. In these examples, we have used a constant grain radius $r = 0.1$.

viation of the grain radii are set to a certain fraction $a \in [0,1)$ of the average grain radius. Naturally, the fastest results are achieved with the pixel-wise algorithm on the GPU. This is several order of magnitudes faster than the grain-based approach without parallelization. Another observation is that, with equal processing power, the pixel-based approach is preferable to the grain-based one when the grain radii are fixed. However, it becomes slower as the standard deviation of the radii increases, since each evaluation requires that more and more cells be visited. Therefore, these two algorithmic approaches both have strengths and weaknesses in different situations. Nevertheless, the conclusion is that the considerable processing power of the GPU leads to a faster pixel-wise algorithm, and so this implementation is preferable in all situations apart from when we use a very large standard deviation, which is rarely required.

An important parameter which has a great impact on the quality of the output is the number of iterations of the Monte Carlo approach N. The trade-off here is obviously execution time versus accuracy. From Equation 5, we know that the approach converges to the correct convolution. Therefore, we control the standard deviation of $\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_Z(\xi_i)$. This standard deviation is bounded by $\sqrt{\frac{1}{4N}}$. In our experiments, we set $N = 800$, giving a standard deviation of 1.77% around the average, which is an acceptable error. For faster results, the parameter N can be reduced to around 100, which gives an error of 5%. The visual quality with this parameter is still quite high, as shown in the supplementary material. For readers who wish to reproduce similar results to those shown in this work, the following parameters are advised : $\mu_r = 0.1$, $\sigma_r = 0$ or $\sigma_r = 0.02$ (for increased graininess), $\sigma = 0.8$ and $N = 800$. The parameter s should be set on a case-by-case basis, usually $s \in \{1,2,3,4\}$.

5. Results

In this section, we present the results of our film grain rendering method. Firstly, we illustrate the advantages of having a model for film grain rendering, in particular the ability to handle any given resolution. We illustrate the influence of each of our model's parameters on the visual output, and show that using these parameters, it is possible to approximate real examples of film grain emul-

sions. We also clearly demonstrate that independently distributed random variables are insufficient to produce realistic film grain. The C++ implementation of our algorithm is freely available online [NGD17].

5.1. Film grain rendering results

In Figures 1 and 8, we show results of our film grain rendering on some vintage images which will benefit artistically from added film grain. One of the main claims of this paper is that our algorithm is resolution-free. To demonstrate this we show the capacity of our algorithm to render film grain on any digital image and at any desired resolution. In these experiments, we have used a constant grain radius $r = 0.1$ pixels. We have shown an extreme closeup of the eyes of the subjects in the images, so that the individual grains can be seen. To the best of our knowledge this capacity to chose any resolution is not proposed by any other grain synthesis algorithm.

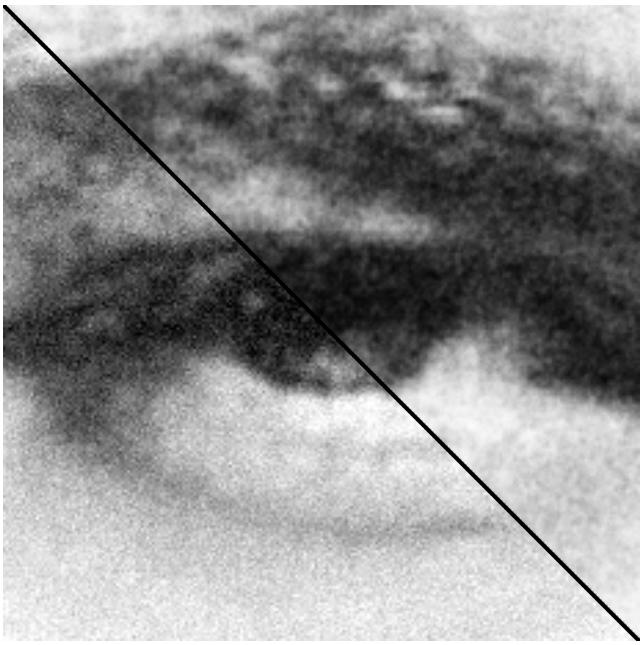
5.2. Variability of graininess

A key advantage of having a model is the possibility of tuning parameters to change the visual grain aspect. Therefore, an important question is what are the parameters in our model which will allow us to imitate different types of grain? Note that what we are interested in here is the subjective visual aspect of film grain "graininess" as opposed to "granularity", which is an objective measurement of the optical density of the developed film emulsion [Liv45].

The most important parameter which we can use to change the visual aspect of the film grain is the grain size. Since our algorithm is resolution-free we can provide accurate rendering for any grain size. Furthermore, the flexibility of our model means that we can take into account grain radii with any probability distribution (with finite variance). Let us recall that if the radii r_i are distributed according to a distribution with a mean μ_r and a variance σ_r^2 then $\mathbb{E}[A_1] = \pi(\mu_r^2 + \sigma_r^2)$ in Equation (2). In Figure 9, we show the effect of varying the grain radius. We choose a log-normal distribution, as indicated in the results reported in [Liv45]. The mean grain radius is increased, and the standard deviation is increased as a fraction of μ_r. The use of random grain sizes as opposed to fixed sizes changes the visual aspect of the grain considerably. The log-normal distribution means that very large grains have a non-negligible chance of appearing. This distribution is defined in the following manner. Suppose that a random variable X is normally distributed with mean μ and variance σ^2, then $Y = \exp(X)$ is distributed with a log-normal distribution. If we wish to specify the effective mean μ_r and standard deviation σ_r of the grain radii, then we specify the mean and standard deviation of the underlying Gaussian distribution as:

$$\mu = \log(\mu_r) - \frac{\mu_r^2 + \sigma_r^2}{2\mu_r^2} \quad ; \quad \sigma^2 = \log\left(\frac{\mu_r^2 + \sigma_r^2}{\mu_r^2}\right). \tag{6}$$

Another parameter which we can tune in order to change the visual result of our algorithm is the variance of the Gaussian filter ϕ used to represent the blurring processes due to the creation and perception of the positive image (see Section 4.2). Figure 10 shows the effect of this parameter on the film grain texture, for a fixed mean radius and standard deviation. The parameter can be tuned to produce more or less sharp grain. Finally, given the flexibility of our model, we can also use different grain shapes, such as triangles. The visual effect of different shapes is not very significant, so we choose to use balls in practice. Nevertheless, we show the results with different shapes in the supplementary material.



Comparison (bottom left: noise, above right: grain)

Figure 11: **Comparison of our film grain synthesis with signal dependent noise**. We show two closeups of images with our film grain synthesis approach vs. signal-dependent noise. The latter is clearly insufficient for realistic film grain synthesis.

5.3. Grain "dithering"

In the supplementary material to this work, we show a closeup comparing our film grain rendering with a compressed input image. In the input image, compression block artifacts are clearly visible. However, in the output with film grain, we have the subjective impression that the quality and resolution of the image are improved. This is linked to the well-known effect called dithering, where noise is added to a signal in order to avoid problems due to quantization. Thus, our grain rendering has the added advantage of giving a subjective impression of improved quality.

5.4. Film grain comparisons

We compare our results with those of independently distributed noise, those of a commercially available product based on scans of real film grain and finally to other previous academic work. In Figure 11, we show the comparison of our film grain rendering with additive independently distributed noise, that is to say where each pixel acts as a "grain" that has no spatial correlation with other pixels. In this experiment, we use Gaussian noise for which the variance is signal-dependent. We learned the variance from the result of our algorithm on a series of constant images of gray level values increasing from 0 to 255. We demonstrate that the covariance of

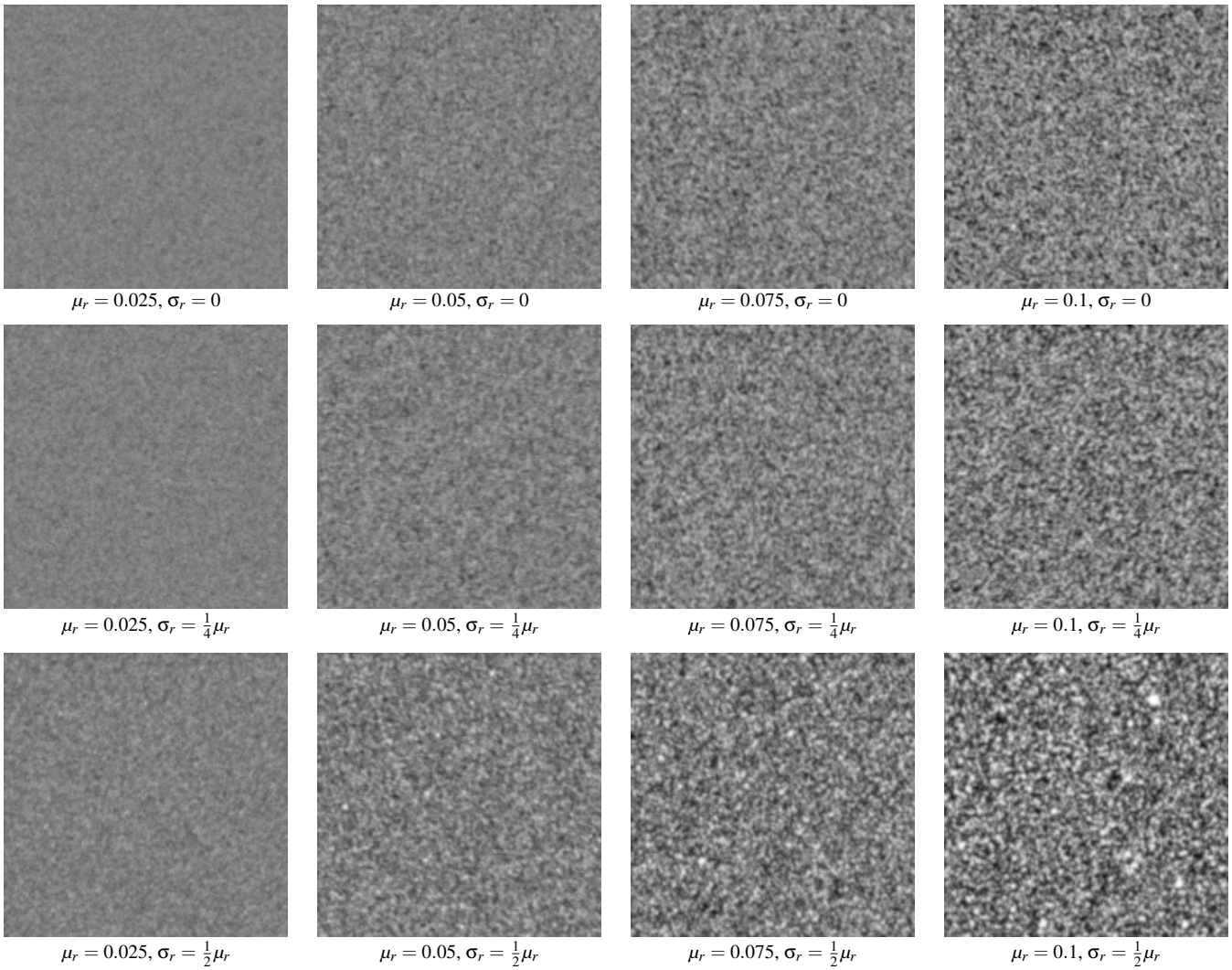


Figure 9: **Film grain texture with varying parameters.** In this Figure, we show the effect of varying grain size on the results of our grain synthesis. We vary the average size of the grains as well as the standard deviation of a log-normal grain distribution. It can be seen that using either constant or random grain sizes has a significant impact on the rendering results.

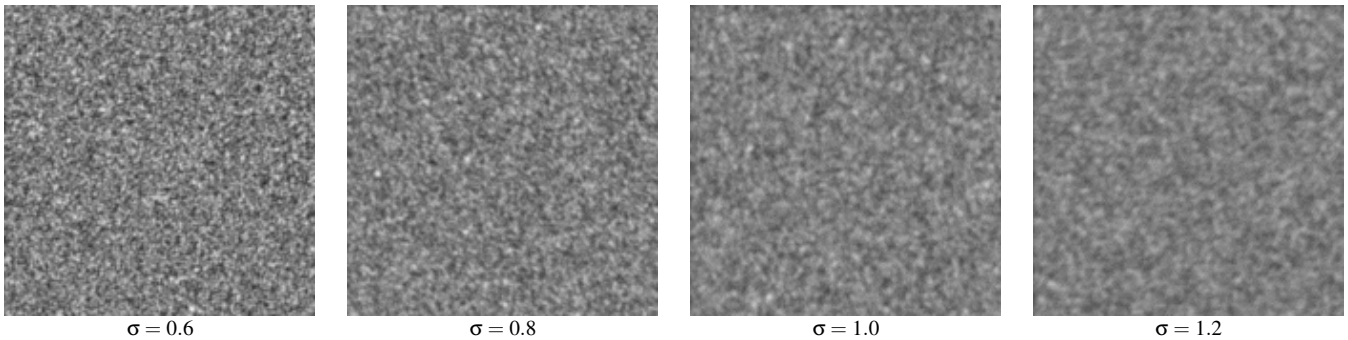
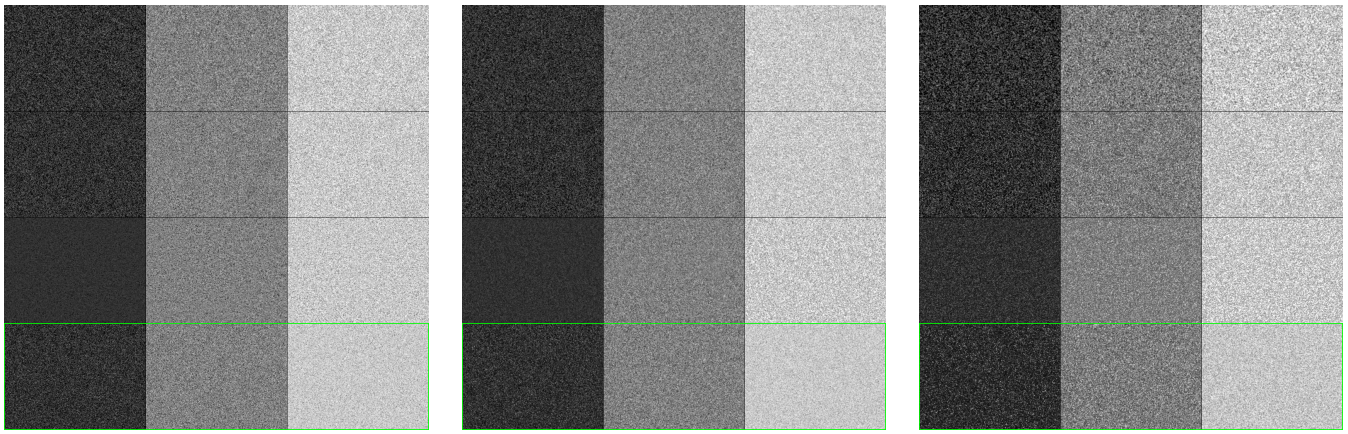


Figure 10: **Film grain texture with varying Gaussian blur parameter** σ. Increasing the standard deviation of the Gaussian filter results in a less pronounced graininess. In these experiments, $\mu_r = 0.05$ pixels and $\sigma_r = \frac{1}{2}r$.



Kodak T-Max 3200 Fuji Neopan 1600 Ilford Delta 3200

Figure 12: **Comparisons of film grain.** We have compared our work with three other approaches:. From top to bottom: that of Stephenson and Saunders [SS07], that of Bae et al. [BPD06], the result of the DxO grain tool [DxO16] and finally the proposed approach, outlined in green. From left to right, we show images with the following constant grey-levels: 50, 128, 200. The parameters used for our results are, from left to right, ($\mu_r = 0.06$, $\sigma_r = 0.024$, $s = 4$), ($\mu_r = 0.05$, $\sigma_r = 0.024$, $s = 4$) and ($\mu_r = 0.06$, $\sigma_r = 0$, $s = 4$).

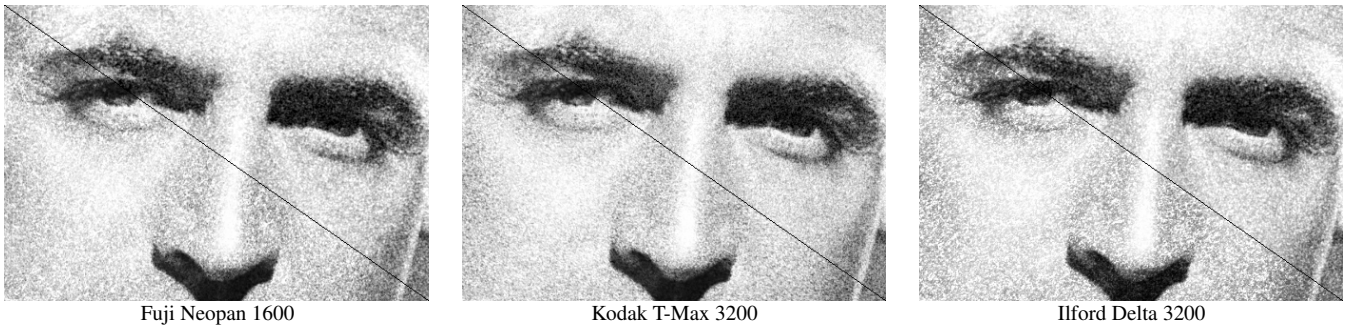
the film grain texture is one of its defining characteristics, and any model which lacks this covariance [Yan97] will not look realistic.

Figure 12 shows the results of film grain synthesis on constant images of increasing gray-levels with three other approaches: that of Stephenson and Saunders [SS07], that of Bae et al. [BPD06] and using the FilmPack software of Dxo [DxO16]. The gray-levels used are 50, 128 and 200. The first two algorithms are based on well-known approaches to texture synthesis. That of Stephenson and Saunders filters the spectrum of an input white noise, and the second employs the Heeger-Bergen [HB95] film grain synthesis algorithm. We used the implementation of Briand et al. [BVGR14] of the Heeger-Bergen approach. The FilmPack grain was used as an approximation of "ground truth" grain, since their algorithm is scan based. We have tuned the parameters of the other algorithms to ensure a similar visual aspect at an average grey-level of 128. In the case of the first two approaches [BPD06, SS07], neither of the papers specify how their texture is applied to an image, or either an unspecified multiplicative parameter is included to control the variance of the texture. Therefore, we set this parameter to a constant which ensured that the two methods had a similar visual aspect for the average grey-level 128. We observe that the visual grain aspect in the two first approaches is quite similar, and that their behavior with different average gray-levels is also consistent with one other. An extremely important point to notice is that the proposed approach shows very different behavior when confronted with dark or light backgrounds. The grain is much more striking and visible in dark areas than in light ones. This behavior is the result of our physical modeling of film grain, and is a strong argument in favor of our algorithm.

In Figure 13, we display approximations of our algorithm of several real film emulsion types, available with the FilmPack software of DxO [DxO16]. In these experiments, we have used an approach similar to that of DxO, in order to have meaningful comparisons. More precisely, we produce a grain image from an input image where the gray-level is equal to 128 everywhere. Then, we apply this texture additively to the input image, modulating the variance of the grain so that it attains a maximum value when the input image gray-level is 128, decreasing to zero at both gray-level extremities (0 and 255), which seems to be a similar procedure to that of FilmPack. We tune our parameters so that the result closely resembles each emulsion type. Interestingly, for the Kodak T-Max 3200 we find that constant grain sizes are most adequate, whereas for the Fuji Neopan 1600 the log-normal distribution is more visually accurate. This may reflect the fact that the size and shape of T-Max crystals are carefully controlled. These examples show that our model is realistic enough to approximate real film grain types, by tuning its physically meaningful parameters.

5.5. Color photography

Naturally, film grain is also present in color photography, and so we wish to synthesize this as well. Color photographic films are made of several layers of emulsions of silver halide crystals. Silver halide crystals are naturally sensitive to blue light. This sensitivity can be extended to other wavelengths via chemical treatment. Therefore, the top layer of the emulsion consists of normal silver halide, followed by a yellow filter. This is necessary since only a fraction of the blue light is actually absorbed by the grains, and most of it would get through otherwise. The next two layers consist of crystals which have been sensitized to green and red light. Note that these layers are also sensitive to blue light, which explains the need for the yellow filter. It is a good approximation to consider that the layers of color film interact independently with light, as each color is only absorbed by one layer. Thus, we add film grain to a color image by running our method independently on each color channel. Figure 14 shows an example of color film grain rendering with two different grain radii. Figure 15 shows a modern photo which has been rendered to give it a more vintage look.



Fuji Neopan 1600 | Kodak T-Max 3200 | Ilford Delta 3200

Figure 13: **Comparison with the DxO FilmPack software**. In this figure, we show three closeups of comparisons of our film grain rendering with different films types available in the FilmPack software of DxO. With this figure, we illustrate that our model is capable of producing film grain which closely resembles scanned images of grain. On each image, our result is shown on the upper right half, and the result of FilmPack is shown bottom left half. Please zoom on the electronic version of the paper for the best visual results.

$r = 0.05$ | $r = 0.1$

$r = 0.05$, close-up | $r = 0.1$, close-up

Figure 14: **Film grain on color images**.

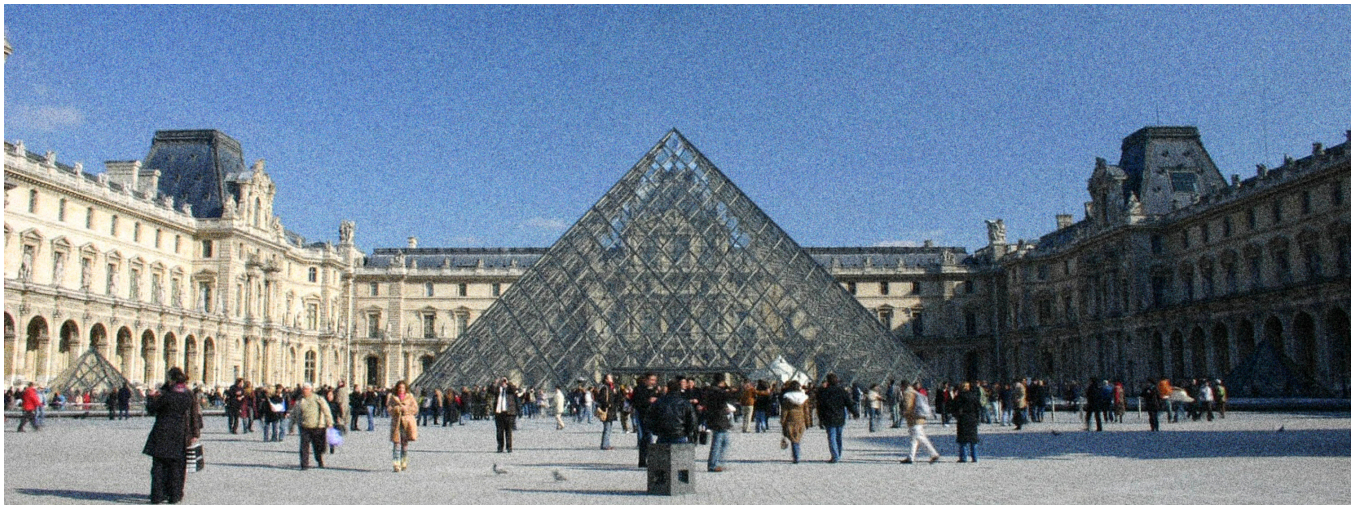


Figure 15: **Colour film grain on a modern image**. We show the result of our film grain rendering on a modern image, which shows that our film grain method can be used for personal artistic purposes to give photographs a certain look and feel.

6. Conclusion and future work

In this work, we have proposed a stochastic model based on the physical reality of silver-halide analog photography. Taking inspiration from the domain of stochastic geometry, our model is based on the *Boolean model* which has the advantage of being both extremely flexible and well understood. This approach allows us to integrate physically meaningful parameters in our model, such as grain size. We propose a film grain rendering algorithm which uses Monte Carlo simulation to produce a digital image with realistic film grain at any desired resolution.

This work opens up new research directions. In spite of our acceleration of the approach, the algorithm is still not real-time in many situations, which may not be practical in the case of processing films. One interesting direction would be to approximate the texture of our current model in a constant image with Gaussian textures, and apply this texture to the input image. Finally, our model is completely two dimensional, and does not take into account the 3D nature of the film emulsion. A more realistic model be a 2D projection of a 3D "*hard-core*" model [CSKM13] (a model where grains cannot overlap), leading to further theoretical and algorithmic challenges. A promising direction is to build upon the recent work on the rendering of granular materials by Meng et al. [MPH*15].

Acknowledgements

This work has been partially funded by the French Research Agengy (ANR) under grant no. ANR-14-CE27-001 (MIRIAM). The authors thank Dick Dickerson and Silvia Zawadzki for their invaluable expertise and the fruitful discussions about analog film. Many thanks also to Noura Faraj for her significant input and advice concerning the presentation of the algorithm and for the Figure 6, and to Henri Maître for his expertise on analog film.

References

[Bay64] BAYER B. E.: Relation Between Granularity and Density for a Random-Dot Model. *Journal of the Optical Society of America 54*, 12 (Dec. 1964), 1485+. 2

[BPD06] BAE S., PARIS S., DURAND F.: Two-scale tone management for photographic look. In *ACM Transactions on Graphics* (2006), vol. 25. 3, 12

[BVGR14] BRIAND T., VACHER J., GALERNE B., RABIN J.: The heeger & bergen pyramid based texture synthesis algorithm. *Image Processing On Line 4* (2014), 276–299. 12

[CKT73] CASTRO P. E., KEMPERMAN J. H. B., TRABKA E. A.: Alternating renewal model of photographic granularity. *Journal of the Optical Society of America 63*, 7 (July 1973), 820+. 2

[CSKM13] CHIU S. N., STOYAN D., KENDALL W. S., MECKE J.: *Stochastic geometry and its applications*, third ed. John Wiley & Sons, 2013. 4, 14

[DxO16] DXO: Dxo film pack 5, 2016. URL: http://www.dxo.com/us/photography/photo-software/dxo-filmpack. 1, 3, 12

[EWK*13] ECHEVARRIA J. I., WILENSKY G., KRISHNASWAMY A., KIM B., DIEGO G.: Computational simulation of alternative photographic processes. *Computer Graphics Forum 32*, 4 (2013), 7–16. 3

[GGM11] GALERNE B., GOUSSEAU Y., MOREL J.-M.: Random phase textures: Theory and synthesis. *IEEE Trans. Image Process. 20*, 1 (2011), 257 – 267. 3

[GK97] GEIGEL J., K. M. F.: A model for simulating the photographic development process on digital images. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques,* (1997), pp. 135–142. 3

[GLM16] GALERNE B., LECLAIRE A., MOISAN L.: *Texton Noise*. Tech. Rep. 2016-09, MAP5, 2016. URL: http://www.math-info.univ-paris5.fr/~bgalerne/texton_noise/. 8

[G'M16] G'MIC: Greyc's magic for image computing, 2016. URL: http://gmic.eu/. 3

[Gru15] GRUBBASOFTWARE: Truegrain, 2015. URL: http://grubbasoftware.com/. 3

[Gur38] GURNEY: The theory of the photolysis of silver bromide and the

photographic latent image. *Proceedings of the Royal Society of London 164* (1938), 151–167. 2

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), pp. 229–238. 3, 12

[Lac15] LACHMAN E.: Edward Lachman Shares His Secrets For Shooting Todd Haynes "Carol", Dec. 2015. 1

[Liv45] LIVINGSTON R.: The Theory of the Photographic Process. By C. E. Kenneth Mees. *J. Phys. Chem. 49*, 5 (May 1945), 509. 10

[LLC*10] LAGAE A., LEFEBVRE S., COOK R., DEROSE T., DRETTAKIS G., EBERT D., LEWIS J., PERLIN K., ZWICKER M.: A survey of procedural noise functions. *Computer Graphics Forum 29*, 8 (December 2010), 2579–2600. doi:10.1111/j.1467-8659.2010.01827.x. 7

[LLDD09] LAGAE A., LEFEBVRE S., DRETTAKIS G., DUTRÉ P.: Procedural noise using sparse Gabor convolution. *SIGGRAPH '09 28*, 3 (August 2009). doi:10.1145/1576246.1531360. 7

[LTW72] LAWTON W. H., TRABKA E. A., WILDER D. R.: Crowded Emulsions: Granularity Theory for Multilayers. *Journal of the Optical Society of America 62*, 5 (May 1972). 2

[MPH*15] MENG J., PAPAS M., HABEL R., DACHSBACHER C., MARSCHNER S., GROSS M., JAROSZ W.: Multi-scale modeling and rendering of granular materials. *ACM Trans. Graph. 34*, 4 (July 2015). 14

[MRB06] MOLDOVAN T. M., ROTH S., BLACK M. J.: Denoising Archival Films using a Learned Bayesian Model. In *Image Processing, 2006 IEEE International Conference on* (Oct. 2006), IEEE, pp. 2641–2644. 1

[NGD17] NEWSON A., GALERNE B., DELON J.:, 2017. URL: https://github.com/alasdairnewson/film_grain_rendering. 2, 10

[Nut13] NUTTING P. G.: On the absorption of light in heterogeneous media. *Philosophical Magazine 26*, 153 (Sept. 1913), 423–426. 2, 4

[OLK09] OH B. T., LEI S.-M., KUO C.-C.: Advanced Film Grain Noise Extraction and Synthesis for High-Definition Video Coding. *IEEE Transactions on Circuits and Systems for Video Technology 19*, 12 (Dec. 2009), 1717–1729. 1, 2, 3

[Sel35] SELWYN E. W. H.: A Theory of Graininess. *Photographic Journal 75* (1935), 571–580. 4

[SM06] SCHALLAUER P., MÖRZINGER R.: Film grain synthesis and its application to re-graining. vol. 6059, pp. 60590Z–60590Z–7. 3

[Smi14] SMITH N. M.: Quentin Tarantino Blasts Digital Projection at Cannes: 'It's the death of cinema.', May 2014. 1

[SS07] STEPHENSON I., SAUNDERS A.: Simulating film grain using the noise-power spectrum. In *Eurographics UK Theory and Practice of Computer Graphics* (2007). 3, 12

[TU83] TANAKA K., UCHIDA S.: Extended random-dot model. *Journal of the Optical Society of America 73*, 10 (Oct. 1983), 1312+. 2

[Wer94] WERNICKE G.: Silver-Halide Recording Materials for Holography and Their Processing. *Zeitschrift für Physikalische Chemie 187*, 2 (Jan. 1994), 322–323. 2

[Wor96] WORLEY S.: A cellular texture basis function. In *SIGGRAPH '96* (1996), ACM, pp. 291–294. doi:10.1145/237170.237267. 7

[Yan97] YAN J. C. K.: *Statistical methods for film grain noise removal and generation.* Master's thesis, University of Toronto, 1997. 1, 3, 12