Supervised Research

Antoine Ratouchniak
Supervisor : Enric Meinhardt-Llopis

April-July 2022

Remerciements

Je remercie Enric Meinhardt-Llopis qui m'a accompagné durant ce TER et pour le temps qu'il m'a accordé. Ce TER m'a conforté dans mon souhait d'orientation vers la recherche en mathématiques et ses applications dans la vision.

1 Introduction

Kernels play an important role in image processing and computer vision. They are ubiquitous in our daily lifes, and have many applications such as feature extraction or photomontage. In this work we will focus on Land's kernel [1] and its interesting properties. To do so, we will first study the kernel mathematically in particular in the sense of distributions [5] which will allow us to implement it in the Fourier domain. We will also introduce some properties of image editing like rotation, zoom and translation. Then, we will review some applications of the kernel such that its Laplacian allows to solve Poisson's equation [2], used in image processing for editing, segmentation, noise reduction or reconstruction... One other application is the Multiscale Retinex which aims to reconstruct a more illuminated image from a given image. It can also work with Land's kernel as Land proposed to use a singular radial kernel $\frac{C}{x^2+y^2}$ where C is a constant [3]. Finally, we will try to implement a U-Net [6] capable of applying Land's filter in less time. Indeed, as we are working with images, we will be able to use the GPU which performs well and will most likely be faster than using the traditional Fast Fourier Transform. This could lead to real-time editing for 4K videos by solving Poisson's equation [2]. Nevertheless, applying Land's filter with the U-Net will be a difficult problem, this is why we will try to deconstruct it as a sum of Gaussians [7] and [8].

2 Definitions and properties

2.1 Definitions

Definition 1. *We define Riesz-Shepard-Land's kernel [1]*

$$k\colon \mathbb{R}^2 \to \mathbb{R} \tag{1}$$

$$(x,y) \mapsto \frac{1}{\sqrt{x^2+y^2}} \tag{2}$$

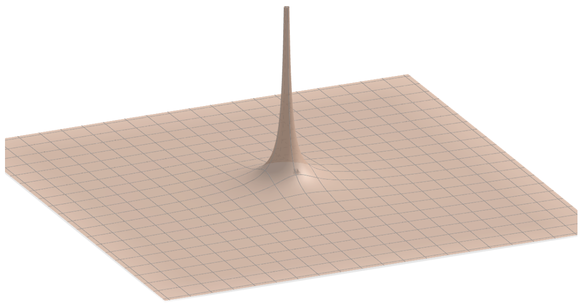


Figure 1: Graph of Land's Kernel

Proposition 1. *The function $k(x,y)$ is $C^\infty(\mathbb{R}^2 \setminus (0,0))$ and locally integrable on $\mathbb{R}^2$.*

Proof. The function k is $C^\infty(\mathbb{R}^2 \setminus (0,0))$ as a composition of C^∞ functions but not defined on $(0,0)$. Let $B_{(0,d)} = \{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq d\}$ be a closed ball centred at the origin of $\mathbb{R}^2$. By getting in terms of polar coordinates and setting $(x,y) = (rcos(\theta), rsin(\theta))$, we get
$\int_{B_{(0,d)}} \frac{1}{\sqrt{x^2+y^2}} dxdy = \int_0^{2\pi} \int_0^d \frac{1}{\sqrt{r^2}} r dr d\theta = 2\pi d < \infty.$ □

Property 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuous, bounded by $M \in \mathbb{R}$ and $g \in L^1_{loc}(\mathbb{R}^n)$ then we can define the convolution product $f * g$ which is also continuous.*

Proof. $\|f * g\|_1 \leq \|f\|_1 \|g\|_1 \leq M \|g\|_1$ □

Definition 2. *The function $f : \mathbb{R}^n \to \mathbb{R}$ is* σ-homogeneous *if $\forall x \in \mathbb{R}^n$, $\forall \lambda \in \mathbb{R}$ $f(\lambda x) = \lambda^\sigma f(x)$.*

Proposition 2. *The function $k(x,y)$ defined in the equation above (2) is positive (-1)-homogenous.*

Proof. $k(\lambda(x,y)) = \frac{1}{\sqrt{\lambda^2(x^2+y^2)}} = |\lambda|^{-1} \frac{1}{x^2+y^2} = |\lambda|^{-1} k(x,y)$. □

We can also look at what happens to the Fourier Transform of a σ-homogeneous function.

Definition 3. *Let $f \in L^1(\mathbb{R}^n)$ and $\boldsymbol{x} \in \mathbb{R}^n$. We can define the* Fourier Transform *of f as $\hat{f}$, $\forall \boldsymbol{\xi} \in \mathbb{R}^n$*

$$\hat{f}(\boldsymbol{\xi}) = \frac{1}{(2\pi)^n} \int_{\mathbb{R}^n} f(x) e^{-i\boldsymbol{\xi} \cdot \boldsymbol{x}} d\boldsymbol{x} \tag{3}$$

Property 2. *Let $f \in L^1(\mathbb{R}^n)$ be a σ-homogeneous function. Then, $\hat{f}$ exists and is $(-\sigma - n)$-homogeneous.*

Proof. Let $\boldsymbol{x}$, $\boldsymbol{\xi} \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$. Then, $\hat{f}(\lambda \cdot \boldsymbol{\xi}) = \frac{1}{(2\pi)^n} \int_{\mathbb{R}^n} f(\boldsymbol{x}) e^{-i\lambda \boldsymbol{x}^T \cdot \boldsymbol{\xi}} d\boldsymbol{x}$ setting $u = \lambda \boldsymbol{x}^T$, by using the σ-homogeneity we have $\frac{1}{(2\pi)^n} \int_{\mathbb{R}^n} \frac{1}{\lambda^n} f(\frac{u}{\lambda}) e^{-iu \cdot \boldsymbol{\xi}} du = \lambda^{-\sigma - n} \hat{f}(\boldsymbol{\xi})$. □

Property 3. $\Delta k(x,y) = k(x,y)^3$

Proof. $\frac{\partial}{\partial x}k(x,y) = \frac{-x}{(x^2+y^2)^{3/2}}$ $\frac{\partial}{\partial y}k(x,y) = \frac{-y}{(x^2+y^2)^{3/2}}$

$$\frac{\partial^2}{\partial x^2}k(x,y) = \frac{-(x^2+y^2)^{3/2}+3x^2\sqrt{x^2+y^2}}{(x^2+y^2)^3} = \frac{3x^2}{(x^2+y^2)^{5/2}} - \frac{1}{(x^2+y^2)^{3/2}}$$

$$\frac{\partial^2}{\partial y^2}k(x,y) = \frac{-(x^2+y^2)^{3/2}+3y^2\sqrt{x^2+y^2}}{(x^2+y^2)^3} = \frac{3y^2}{(x^2+y^2)^{5/2}} - \frac{1}{(x^2+y^2)^{3/2}}$$

Hence $\Delta k(x,y) = \frac{\partial^2}{\partial x^2}k(x,y) + \frac{\partial^2}{\partial y^2}k(x,y) = \frac{3(x^2+y^2)}{(x^2+y^2)^{5/2}} - \frac{2}{(x+y)^{3/2}} = \frac{1}{(x+y)^{3/2}} = k(x,y)^3$. □

2.2 Zooms and rotations

Now we could wonder what would happen if we tried to rotate our image. In this subsection, we consider a 2D square of size m, $f \in L^1([-m;m]^2)$ and $R \in SO(\mathbb{R}^2)$, a rotation matrix. A rotation on an image is defined as follow, $f' = R \circ f$ where f' is the rotated image. If $\theta \in [0, \pi[$, the resulting new size is $2 \cdot m \cdot (cos(\theta) + sin(\theta))$. Indeed, as the rotated image remains a square, we have that its size is always $|m(cos(\theta) + sin(\theta)) - (-m)(cos(\theta) + sin(\theta))|$ by periodicity.

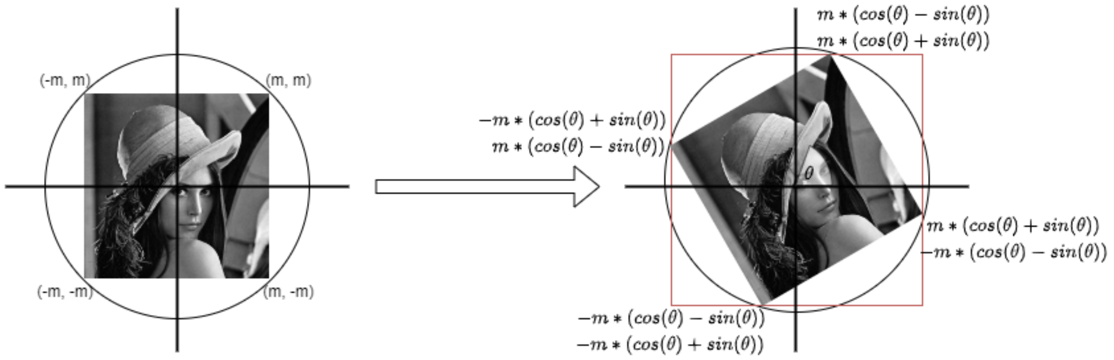

(-m, m)
(m, m)
(-m, -m)
(m, -m)
$m * (cos(\theta) - sin(\theta))$
$m * (cos(\theta) + sin(\theta))$
$-m * (cos(\theta) + sin(\theta))$
$m * (cos(\theta) - sin(\theta))$
θ
$m * (cos(\theta) + sin(\theta))$
$-m * (cos(\theta) - sin(\theta))$
$-m * (cos(\theta) - sin(\theta))$
$-m * (cos(\theta) + sin(\theta))$

Figure 2: Rotation of an image

Property 4. *Let $f \in L^1(\mathbb{R}^2)$, R a rotation matrix. Then $\widehat{f \circ R} = \hat{f} \circ R$.*

Proof. Let $\boldsymbol{x} = (x,y) \in \mathbb{R}^2, \boldsymbol{u} = (u,v) \in \mathbb{R}^2$ and $\boldsymbol{\xi} = (\xi, \eta) \in \mathbb{R}^2$. $\widehat{f \circ R}(\boldsymbol{\xi}) = \frac{1}{(2\pi)^2}\int_{\mathbb{R}^2} f(R \cdot \boldsymbol{x}) \cdot e^{-i\boldsymbol{x}^T\boldsymbol{\xi}}dxdy$, setting $\boldsymbol{u} = R \cdot \boldsymbol{x}$, we have $\boldsymbol{x} = R^{-1} \cdot \boldsymbol{u} = R^T \cdot \boldsymbol{u}$ since $R^{-1} = R^T$. We can compute the Jacobian matrix $J_{\boldsymbol{u}}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial}{\partial u}(ucos(\theta) + vsin(\theta)) & \frac{\partial}{\partial v}(ucos(\theta) + vsin(\theta)) \\ \frac{\partial}{\partial u}(vcos(\theta) - usin(\theta)) & \frac{\partial}{\partial v}(vcos(\theta) - usin(\theta)) \end{pmatrix} = R^T$.

Then, $\widehat{f \circ R}(\boldsymbol{\xi}) = \frac{1}{(2\pi)^2}\int_{\mathbb{R}^2} f(\boldsymbol{u}) \cdot e^{-i(R^T \cdot \boldsymbol{u})^T\boldsymbol{\xi}}|R|dudv = \frac{1}{(2\pi)^2}\int_{\mathbb{R}^2} f(\boldsymbol{u}) \cdot e^{-i\boldsymbol{u}^T \cdot R\boldsymbol{\xi}}dudv = \hat{f}(R \cdot \boldsymbol{\xi})$ because $|R| = |R^T| = 1$ and R is a bijection.

We remind $R = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix}$, its inverse $R^{-1} = \frac{1}{cos(\theta)^2 + sin(\theta)^2}\begin{pmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{pmatrix} = R^T$. □

We can also want to zoom-in or zoom-out on the image.

Property 5. *Let* $Z_t(f)(\boldsymbol{x}) = f(\boldsymbol{x}/t)$ *be the zoom operator. Then,* $\widehat{Z_t(f)(\boldsymbol{x})}(\xi,\eta) = t^2\hat{f}(t(\xi,\eta)) = t^2 Z_{t^{-1}}(\hat{f})(\boldsymbol{x})(t(\xi,\eta))$*. In the case of a homogeneous function, we have* $\widehat{Z_t(f)(\boldsymbol{x})}(\xi,\eta) = t^2 t^{-\sigma-2}\hat{f}(\xi,\eta) = t^{-\sigma}\hat{f}(\xi,\eta)$ *thanks to Property 2.*

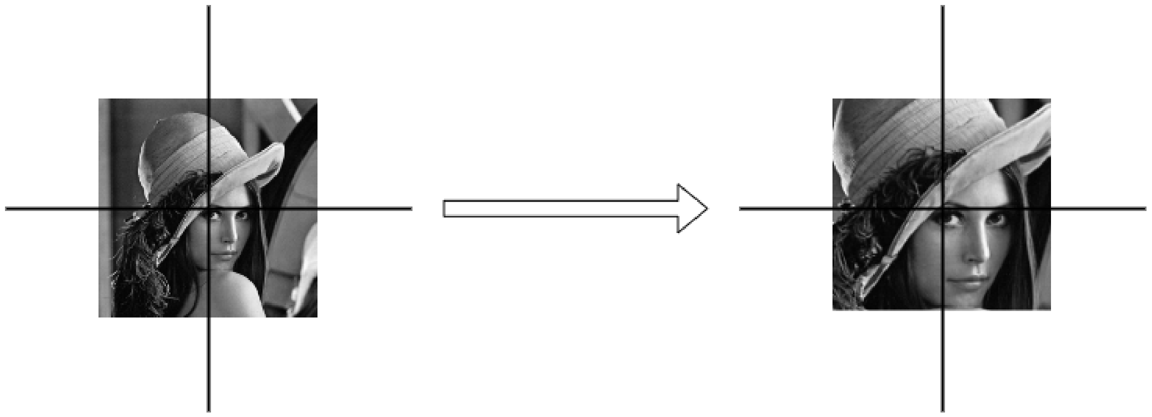


Figure 3: Zoom of an image

Finally, we could also want to zoom-in or out on a certain area of the image. One way would be to translate the image so that the area of the image we want to zoom-in or out is centred and zoom then. We recall the shift operator $\tau_a f(x) = f(x-a)$ and its Fourier Transform $\widehat{\tau_a f(\boldsymbol{x})}(\xi,\eta) = e^{-it(\xi+\eta)}\hat{f}(\xi,\eta)$.

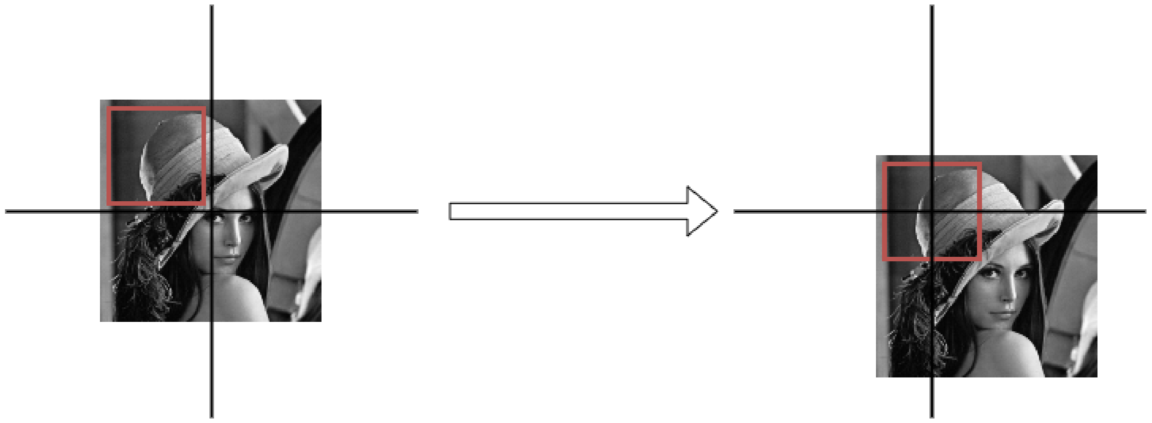


Figure 4: Translation of an image

2.3 Radial function and distribution

Definition 4. *Let $f : \mathbb{R}^n \to \mathbb{R}$, $n > 1$, f is a* radial function *if $\exists \varphi : \mathbb{R} \to \mathbb{R}, \forall \boldsymbol{x} \in \mathbb{R}^n$ such that $f(\boldsymbol{x}) = \varphi(\|\boldsymbol{x}\|)$ which means that the function is defined by the distance to the origin. Equivalently, f is invariant to all rotations : $f \circ R = f$, $\forall R \in SO(\mathbb{R}^n)$.*

Proposition 3. *Land's kernel $k(x, y)$ defined in the equation above (2) is radial.*

Proof. $k(x, y) = \frac{1}{\sqrt{x^2+y^2}} = \frac{1}{\|(x,y)\|_2} = \varphi(\|(x, y)\|_2)$ with $\varphi(\|\boldsymbol{x}\|) = \frac{1}{\|\boldsymbol{x}\|}$. □

Property 6. *Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a C^2 function, the Laplacian operator Δf is invariant by rotation.*

Proof. In two dimensions, performing a rotation is equivalent to obtaining the new coordinates $\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$ which gives $\begin{cases} u &= x\ cos(\theta) - y\ sin(\theta) \\ v &= x\ sin(\theta) + y\ cos(\theta) \end{cases}$. We first compute the first order derivative $\frac{\partial}{\partial x} f(u, v) + \frac{\partial}{\partial y} f(u, v) = \frac{\partial u}{\partial x}\frac{\partial f}{\partial u} + \frac{\partial v}{\partial x}\frac{\partial f}{\partial v} + \frac{\partial u}{\partial y}\frac{\partial f}{\partial u} + \frac{\partial v}{\partial y}\frac{\partial f}{\partial v} = \frac{\partial f}{\partial u} cos(\theta) + \frac{\partial f}{\partial v} sin(\theta) - \frac{\partial f}{\partial u} sin(\theta) + \frac{\partial f}{\partial v} cos(\theta)$ by using the chain rule. Repeating the same treatment by noticing $\frac{\partial^2}{\partial x^2} f(u, v) + \frac{\partial^2}{\partial y^2} f(u, v) = \frac{\partial}{\partial x}\frac{\partial f}{\partial x} + \frac{\partial}{\partial y}\frac{\partial f}{\partial y}$, we get

$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}(\frac{\partial f}{\partial u} cos(\theta) + \frac{\partial f}{\partial v} sin(\theta)) = \frac{\partial f}{\partial x}(\frac{\partial}{\partial u} cos(\theta) + \frac{\partial}{\partial v} sin(\theta)) = (\frac{\partial f}{\partial u} cos(\theta) + \frac{\partial f}{\partial v} sin(\theta))(\frac{\partial}{\partial u} cos(\theta) + \frac{\partial}{\partial v} sin(\theta))$

$\frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}(\frac{\partial f}{\partial v} cos(\theta) - \frac{\partial f}{\partial u} sin(\theta)) = \frac{\partial f}{\partial y}(\frac{\partial}{\partial v} cos(\theta) - \frac{\partial}{\partial u} sin(\theta)) = (\frac{\partial f}{\partial v} cos(\theta) - \frac{\partial f}{\partial u} sin(\theta))(\frac{\partial}{\partial v} cos(\theta) - \frac{\partial}{\partial u} sin(\theta))$

Finally, expanding everything we obtain $\frac{\partial^2}{\partial x^2} f(u, v) + \frac{\partial^2}{\partial y^2} f(u, v) = \frac{\partial f}{\partial u}(cos(\theta)^2 + sin(\theta)^2) + 2\frac{\partial^2 f}{\partial u \partial v} cos(\theta) sin(\theta) - 2\frac{\partial^2 f}{\partial u \partial v} cos(\theta) sin(\theta) + \frac{\partial^2 f}{\partial^2 v}(cos(\theta)^2 + sin(\theta)^2) = \frac{\partial^2}{\partial u^2} f(u, v) + \frac{\partial^2}{\partial v^2} f(u, v)$. □

Property 7. *Let $f \in L^1(\mathbb{R}^n)$ be a radial function then its Fourier Transform $\hat{f}$ is well-defined and is also radial.*

Proof. By using Property 4 we have that for R, a rotation matrix acting on $\mathbb{R}^n$, $\widehat{f \circ R(\boldsymbol{\xi})} = \hat{f} \circ R(\xi) = \hat{f}$ since f is radial. □

Lemma 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ such that :*
- f is σ-homogeneous (1)
- f is radial (2)
Then $\exists C \in \mathbb{R}$ such that $f(\boldsymbol{x}) = C\|\boldsymbol{x}\|^\sigma$

Proof. Let $\mathcal{B} = (\vec{e_1}, \vec{e_2}, ..., \vec{e_n})$ be the canonical basis of $\mathbb{R}^n$. We have $\forall x \in \mathbb{R}^n$, $\exists R \in SO(\mathbb{R}^n)$ such that $x = \lambda R \cdot \vec{e_1}$, with $\lambda = \|x\|$. Consequently, $f(x) = f(R \cdot \lambda\vec{e_1}) \overset{(2)}{=} \varphi(\|R \cdot \lambda\vec{e_1}\|) = \varphi(\lambda\|\vec{e_1}\|) \overset{(1)}{=} \lambda^\sigma f(\vec{e_1}) = f(\vec{e_1})\|x\|^\sigma$ meaning that f is necessarily of the form $C\|x\|^\sigma$ with $C = f(\vec{e_1})$. □

We would like to apply all these properties to Land's kernel, however, it is not integrable. In order to do this, we will introduce tempered distributions.

Definition 5. Schwartz Space *on $\mathbb{R}$ is defined by $\mathcal{S}(\mathbb{R}) = \{f : \mathbb{R} \to \mathbb{C} \mid f \in C^\infty(\mathbb{R}), \forall p, q \geq 0, \sup_{x \in \mathbb{R}^n} |x^p f^{(q)}| < \infty\}$. It is the set of smooth, bounded and rapidly decreasing functions. Similarly, $S(\mathbb{R}^n)$ is defined using monomials of degrees p and partial derivatives of degrees q. Notice that the function of $S(\mathbb{R}^n)$ are rapidly decreasing towards 0.*

Remark : We also observe that Schwartz Space is closed by derivatives and by Fourier Transform since $\widehat{f^{(n)}} = (ik)^n \hat{f}$ and $\widehat{x^n f} = i^n \widehat{f^{(n)}}$.

Definition 6. Tempered distributions *[5] are the topological dual space of $\mathcal{S}(\mathbb{R}^n)$. In other words a* tempered distribution T *is a continuous linear map $T : \mathcal{S}(\mathbb{R}^n) \to \mathbb{C}$. The action of T on a Schwartz function φ is denoted $\langle T, \varphi\rangle$.*

The set of *tempered distributions* is denoted $\mathcal{S}'$. Then, $T_f \in S'(\mathbb{R}^n)$.

Example 1 : Let $a \in \mathbb{R}^n$. We define the Dirac mass, δ_a by $\langle \delta_a, \varphi\rangle = \varphi(a)$, $\forall \varphi \in \mathcal{S}(\mathbb{R}^n)$.
Example 2 : Let $f \in L^1_{loc}(\mathbb{R}^n)$, slowly increasing, then we define a distribution T_f by $\langle T_f, \varphi\rangle = \int_{\mathbb{R}^n} f(\boldsymbol{x})\varphi(\boldsymbol{x})d\boldsymbol{x}$ $\forall \varphi \in S(\mathbb{R}^n)$. This integral is always finite and it can be proved [5] that $f \mapsto T_f$ is injective. Therefore we can interpret locally integrable slowly increasing functions as tempered distributions.

Definition 7. *The zoom of a distribution is defined by $\langle Z_t(T), \varphi\rangle = \langle T, t^n Z_{t^{-1}}(\varphi)\rangle$.*

Definition 8. *The rotation of a distribution is defined by $\langle T \circ R, \varphi\rangle = \langle T, \varphi \circ R^{-1}\rangle$.*

Definition 9. *The derivative of a distribution is defined by $\langle \partial_{x_i} T, \varphi\rangle = \langle T, -\partial_{x_i}\varphi\rangle$. Notice that this definition makes sense since Schwartz Space is closed by derivatives.*

Definition 10. *The Fourier Transform of a tempered distribution $T \in \mathcal{S}'$ is the distribution $\hat{T}$ defined by $\langle \hat{T}, \varphi\rangle = \langle T, \hat{\varphi}\rangle$. Notice that this definition makes sense since Schwartz Space is closed by Fourier Transform.*

These definitions are compatible with the correspondence between functions and distributions. This is proved in the propositions below.

Definition 11. *Let f be a C^1 function, f is* slowly increasing *if $\forall a > 0, \lim\limits_{x\to+\infty} \frac{f(ax)}{f(x)} = 1$.*

Property 8. *Let $f \in L^1_{loc}(\mathbb{R}^n)$ be a slowly increasing and C^1 function, then $Z_t(T_f) = T_{Z_t(f)}$.*

Proof. These two distributions are well-defined. To prove that there are equal we have to check that they produce the same result over an arbitrary function $\langle Z_t(T_f), \varphi\rangle = \langle T_{Z_t(f)}, \varphi\rangle$ $\forall \varphi \in \mathcal{S}(\mathbb{R}^n)$.

$\langle Z_t(T_f), \varphi\rangle \overset{Definition\ 7}{=} \langle T_f, t^n Z_{t^{-1}}(\varphi)\rangle = \int_{\mathbb{R}^n} f(\boldsymbol{x}) t^n \varphi(t * \boldsymbol{x})d\boldsymbol{x}$, setting $\boldsymbol{u} = t * \boldsymbol{x}$, we get $\int_{\mathbb{R}^n} f(\frac{\boldsymbol{u}}{t})\varphi(\boldsymbol{u})d\boldsymbol{u} = \int_{\mathbb{R}^n} Z_t(f(\boldsymbol{u}))\varphi(\boldsymbol{u})d\boldsymbol{u} = \langle T_{Z_t(f)}, \varphi\rangle$. □

Property 9. *Let $f \in L^1_{loc}(\mathbb{R}^n)$ be a slowly increasing and C^1 function, then $T_f \circ R = T_{f\circ R}$.*

Proof. The proof is similar to Property 4 by setting $\boldsymbol{u} = R^{-1} \cdot \boldsymbol{x}$. □

Property 10. *Let $f \in L^1_{loc}(\mathbb{R}^n)$ be a slowly increasing and C^1 function, then $\partial_{x_i} T_f = T_{\partial x_i f}$.*

Proof. $\langle \partial_{x_i} T_f, \varphi\rangle \overset{Definition\ 9}{=} \langle T_f, -\partial_{x_i}\varphi\rangle = \int_{\mathbb{R}^n} f(\boldsymbol{x})(-\frac{\partial}{\partial x_i}\varphi(\boldsymbol{x}))d\boldsymbol{x}$ and by integration by parts we get $[\varphi(\boldsymbol{x})f(\boldsymbol{x})]^{+\infty}_{-\infty} + \int_{\mathbb{R}^n} \frac{\partial}{\partial x_i} f(\boldsymbol{x})\varphi(\boldsymbol{x})d\boldsymbol{x} = \langle T_{\partial_{x_i} f}, \varphi\rangle$ since φ is a rapidly decreasing function and f is a slowly increasing function we have $[\varphi(\boldsymbol{x})f(\boldsymbol{x})]^{+\infty}_{-\infty} = 0$. □

Property 11. *Let $f \in L^1_{loc}(\mathbb{R}^n)$ and $C^1(\mathbb{R}^n)$, then $\widehat{T_f}$ exists and $\widehat{T_f} = T_{\hat{f}}$.*

Proof. This is a consequence of Fubini's theorem, therefore we have $\langle \widehat{T_f}, \varphi \rangle \overset{Definition\ 10}{=} \langle T_f, \hat{\varphi} \rangle = \int_{\mathbb{R}^n} f(\boldsymbol{x})\hat{\varphi}(\boldsymbol{x})d\boldsymbol{x} = \int_{\mathbb{R}^n} \hat{f}(\boldsymbol{x})\varphi(\boldsymbol{x})d\boldsymbol{x} = \langle T_{\hat{f}}, \varphi \rangle$. □

Proposition 4. *Land's kernel $k(x, y)$ defined in the equation above (2) can be defined as a tempered distribution and we can compute its Fourier Transform.*

Proof. The function $k(x, y)$ is a decreasing function, locally integrable and $C^\infty(\mathbb{R}^2 \setminus (0, 0)$ so we can define it as a tempered distribution. □

Proposition 5. *The Fourier Transform of Land's kernel $\hat{k}$ defined in the equation above (2) is $\hat{k} = k$.*

Proof. Thanks to Property 7, as the Fourier Transform of a radial function is radial, $\hat{k}$ is radial. Moreover, $\hat{k}$ has the form $C\|(x, y)\|^\sigma$ by Lemma 1 where $C = k(1, 0) = 1$ and $\sigma = \frac{-1}{2}$. We have $\langle T_{\hat{k}}, \varphi \rangle = \langle T_k, \varphi \rangle$ if $0 \notin Supp(\varphi)$. Therefore, $\hat{k} = k\mathbb{1}_{(x,y)\neq(0,0)} + \delta_{(0,0)}$ where δ will not be defined here. □

2.4 Land's filter and its applications

Now, we can focus on the effects of applying this kernel to an image. We proved that $\hat{k} = k$, and thanks to the convolution theorem we also have that for $f,\ g \in L^1(\mathbb{R})$, $\widehat{f * g} = \hat{f} \cdot \hat{g}$. We remind that by Fubini's theorem $(\widehat{f * g)(x})(\xi) = \frac{1}{2\pi} \int_{\mathbb{R}^2} f(t)g(x - t)e^{-ix\xi}dtdx = \frac{1}{2\pi}(\int_{\mathbb{R}} f(t)e^{-it\xi}dt \cdot \int_{\mathbb{R}} g(x - t)e^{-i(x-t)\xi}dx) = \frac{1}{2\pi}(\int_{\mathbb{R}} f(t)e^{-it\xi}dt \cdot \int_{\mathbb{R}} g(x)e^{-ix\xi}dx) = \hat{f} \cdot \hat{g}$.

Finally, combining all these properties we get for an image $u \in L^1(\mathbb{R}^2)$, $\widehat{k * u} = \hat{k} \cdot \hat{u} = k \cdot \hat{u}$. We then have to compute the Discrete Fourier Transform of u and must not forget to center k before doing the pointwise product. We remind the 2D $DFT \in \mathbb{C}^{NxM}$ of an image u

$DFT_u(k, l) = \frac{1}{\mathrm{N \cdot M}} \sum_{x=0}^{N} \sum_{y=0}^{M} u(x, y)e^{\frac{-2\pi ixk}{\mathrm{N}}} e^{\frac{-2\pi iyl}{\mathrm{M}}}$ where N and M are the width and height of the image. The convolution theorem holds for the discrete case.

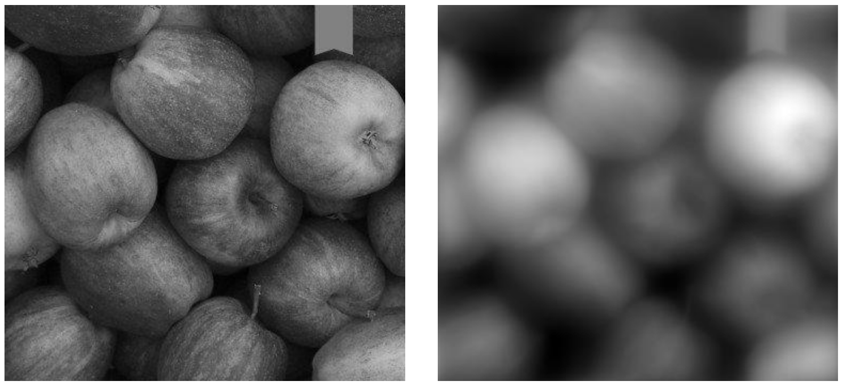

Original image Image convolved with Land's filter

Figure 5: Application of Land's filter

If the image is C^2, we could also take its Laplacian to recover the original image since we have $\widehat{\frac{\partial^2}{\partial\eta\partial\xi}u}(\xi,\eta) = (i\xi)\cdot(i\eta)\cdot\hat{u} = -\xi\eta\cdot\hat{u}$ and $\widehat{\Delta k} = -(\xi^2+\eta^2)\cdot\hat{k} = \widehat{k^3}$ thanks to Property 3.

$\widehat{\Delta u} = -(\xi^2+\eta^2)\cdot\hat{u}$ but knowing $\hat{k} = k$, we get $-k * \widehat{k * \Delta u} = -\widehat{k * k}\cdot\widehat{\Delta u} = \frac{1}{\xi^2+\eta^2}(\xi^2+\eta^2)\hat{u} = \hat{u}$

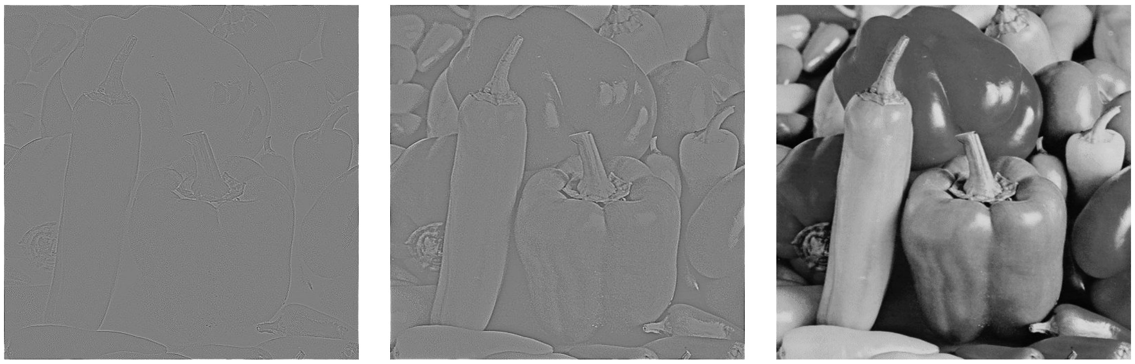


Figure 6: Retrieving the original image from the Laplacian

In practice, we use the Fast Fourier Transform to make the algorithm easier in complexity. The complexity of the FFT is $O(Nln(N))$ where N is the size of the image, height $\times$ width and thanks to the convolution theorem, the convolution in the Fourier domain has a $O(N)$ complexity. The final complexity to convolve the image with Land's filter (2) is therefore $O(Nln(N)+N) = O(Nln(N))$. *Remark : we can also notice that* $k * \Delta u = \Delta k * u$.

The second application we could do is Poisson image editing. We remind Poisson's equation

$$\begin{cases} \Delta u(x) = f(x) & \text{if} \quad x \in \Omega \\ u(x) = g(x) & \text{if} \quad x \in \partial\Omega \end{cases} \tag{4}$$

where u is unknown and f, $g \in L^2(\mathbb{R}^2)$ are known. We let Ω be a closed subset of $\mathbb{R}^2$ and $\partial\Omega$ the silhouette of this region. We now suppose we have a source image $S \in L^2(\mathbb{R}^2)$, C^2, and a target image $f \in L^2(\mathbb{R}^2)$ also C^2. We denote $\mathtt{v}$ the vector field of the region in the source image, i.e. the gradient $(\frac{\partial S}{\partial x}, \frac{\partial S}{\partial y})$. In the case of images, the derivative can be the forward finite difference i.e. for the horizontal coordinates $\frac{\partial S}{\partial x} = \lim\limits_{h\to 0} \frac{S(x+h,y)-S(x,y)}{h} \overset{h=1}{=} S(x+1,y) - S(x,y)$ and similarly for the horizontal coordinates.

The problem of blending two images [2] has been formulated by the following equation

$$\min_u \int_\Omega |\nabla u - \mathtt{v}|^2 \text{ such that } u|_{\partial\Omega} = f|_{\partial\Omega} \tag{5}$$

where $\nabla u = (\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y})$. The idea is to minimize color mismatch between the source and the target in L^2-norm. Finally, the colors on the boundaries have to match to avoid a discontinuity. This problem (5) has a unique C^2 solution which is given by Poisson's equation (4) where f is the Laplacian of the source image and g is the boundaries of the target image. We finally get $\Delta u = \text{div } \mathtt{v}$ over Ω and

$u|_{\partial\Omega} = f|_{\partial\Omega}$ where div $\mathtt{v} = \vec{\nabla}\cdot\mathtt{v} = \Delta S$. Land's filter can be used to solve Poisson's equation as we get back the original image from the Laplacian i.e. $\Delta^{-1}u = -k * k * u$. After computing the Laplacian, there are different methods to solve the equation $\Delta u = \Delta S$ such as the Jacobi method, Gauss-Seidel method or LU decomposition. Similarly for the Laplacian, we compute it thanks to the finite difference such as $\frac{\partial^2 u}{\partial x^2} = u'(x+1,y) - u'(x,y) = (u(x+2,y) - u(x+1,y)) - (u(x+1,y) - u(x,y)) = u(x+2,y) - 2u(x+1,y) + u(x,y)$. Similarly, we get $\frac{\partial^2 u}{\partial y^2} = u(x,y+2) - 2u(x,y+1) + u(x,y)$. Shifting both the horizontal and vertical second order derivatives to the left and top respectively, we get $\frac{\partial^2 u}{\partial x^2} = u(x+1,y) - 2u(x,y) + u(x-1,y)$ and $\frac{\partial^2 u}{\partial y^2} = u(x,y+1) - 2u(x,y) + u(x,y-1)$. Therefore, we obtain $\Delta u(x,y)e = u(x+1,y) + u(x-1,y) + u(x,y+1) + u(x,y-1) - 4u(x,y)$ and we can model the Laplacian as the filter $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$. This filter can be used to solve Poisson's equation to calculate Δu.

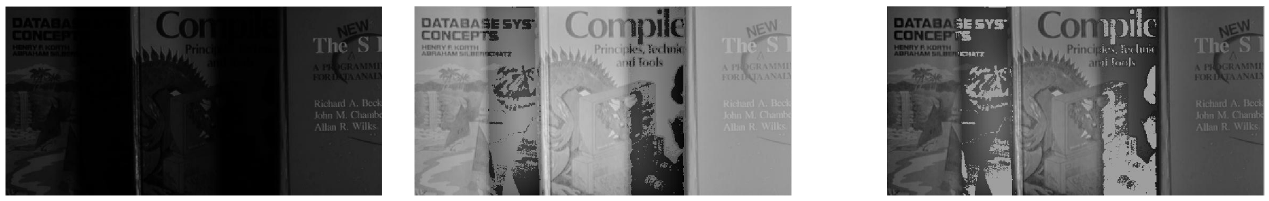
The final application is the Multiscale Retinex [3] which aims to enhance contrast or restore colors. The idea is to model how the human eye sees and perceive the variations of lightness in an image. In order to do this, Land developed a first algorithm that consisted in taking the average of the intensity calculated through a collection of paths. However, in a later work [3], he updated his method to avoid the paths and the random walks. This new method computes the ratio between the original image and the image convolved with the filter. We define the Retinex function by

$$R(x,y) = \frac{I(x,y)}{(F * I)(x,y)} \tag{6}$$

where I is the input image and F is a surrounding function. We can use Land's kernel defined in the equation above (2) as a surrounding function to compute the Retinex as Land proposed to use a singular radial kernel. Finally, we take the logarithm of $R(x,y)$ which gives

$$log(R(x,y)) = log(I(x,y)) - log((F * I)(x,y)) \tag{7}$$

This operation is not sufficient to get a good result, we can therefore restore the colors by giving back a part of their initial intensity [4]. There are different approaches to do so, the one we are going to use gives the final intensity $R'(x,y) = R(x,y) \cdot log(1 + C \cdot I(x,y))$ where $C \in \mathbb{R}$. Using Land's kernel has the advantage of getting rid of parameters. On the other hand, there is less flexibility compared to the Gaussian kernel with the standard deviation.

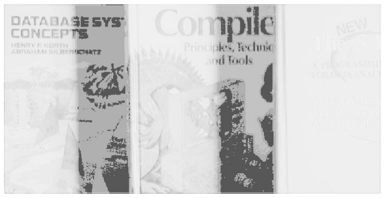


Input image

Single Scale Retinex

SSR with color restoration where $C = 1000$

Figure 7: Example of Single Scale Retinex with Land's kernel


Single Scale Retinex

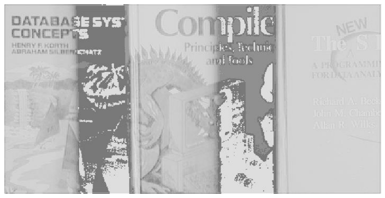

SSR with color restoration where $C = 1000$

Figure 8: Example of Single Scale Retinex with Gaussian kernel with $\sigma = 3$

As we can see, we obtain satisfactory results on a monochrome image with a Single Scale Retinex. The results seem to better with Land's kernel than the Gaussian kernel for this image.
Remark : the operations are performed on a single color channel image, this is a bit different if the image contains more channels.

3 U-Net

3.1 Architecture and training

3.1.1 Architecture

Doing the Fourier Transform can be computationally expensive, this is why we can try to apply Land's kernel defined in the equation above (2) thanks to a neural network. In this section we will work with images of size 512×512.

The U-Net [6] is a fully convolutional neural network used for image segmentation. The objective of this neural network is to, from an input image, obtain a new image with additional features. The name U-Net comes from the fact that it is U-shaped. It works similarly as an auto-encoder to compress and decompress the data.

During the contraction, the image is firstly convolved by two kernels of size 3x3, each followed by a ReLU activation to finish with a max pooling. This operation is repeated as many times as necessary to obtain an image of size 1x1 or 2x2. The image is then convolved twice again followed by a ReLU activation in the bottleneck. Finally, during the expansive path, unlike an auto-encoder in order not to lose all the information, a concatenation is performed with the image of the same size obtained during the contraction. We perform an up-sampling, which is a convolution transpose, then, the resulting image is concatenated with the previous image obtained during the contraction. In a similar way, we perform two 3x3 convolutions followed by a ReLU activation. This operation is repeated as many times as necessary to obtain an image of a certain size. Finally, a convolution of size 1x1 is performed.

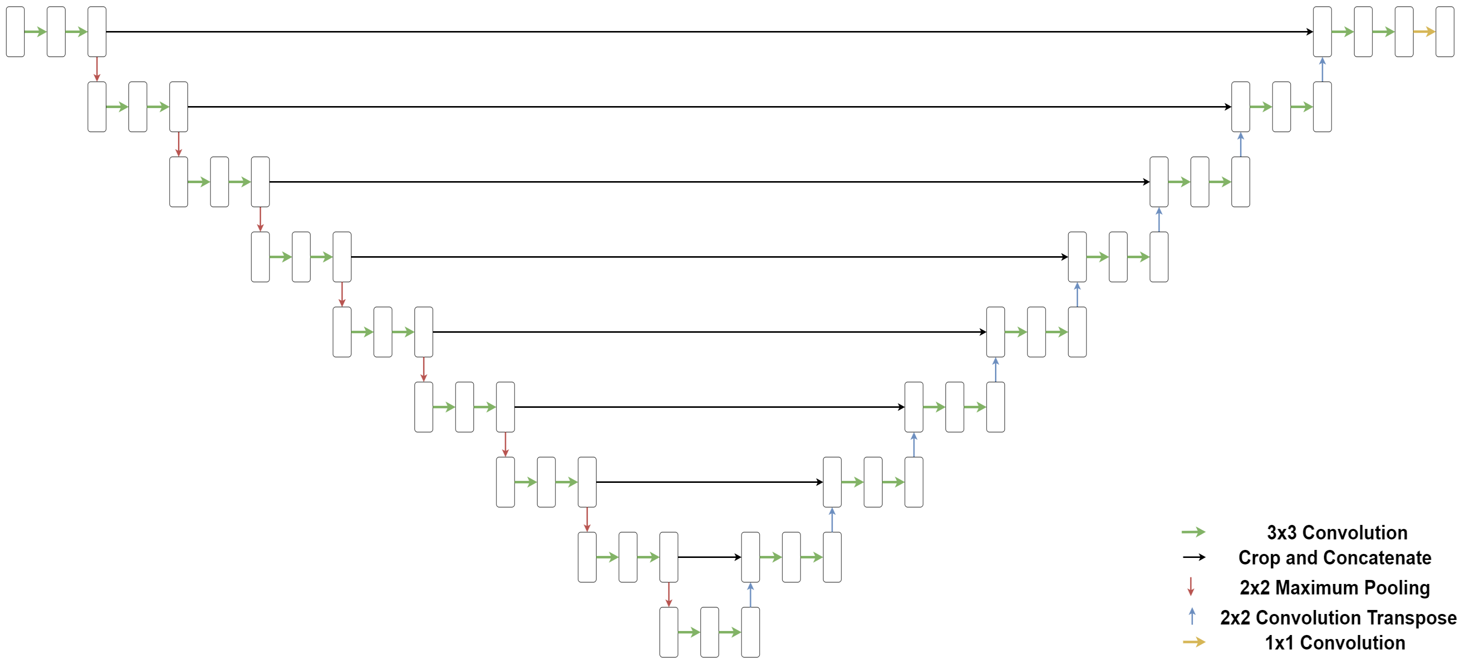


Figure 9: U-Net architecture

3.1.2 Training

Training the neural network to reproduce Land's kernel is a difficult problem. Indeed, since Land's kernel has a vertical asymptote at 0, we can obtain really large values. Normalizing the data would result in a loss of information. For example, an image with values in $[0, 100]$ would receive the same treatment as an image with values in $[0, 255]$, which is wrong. However, since Land's kernel is symmetric, we can approximate it with a sum of Gaussians [7] and [8].

We will train the neural network to perform Gaussian blurs. The architecture we are going to use is a bit different from the one in the article. The weights used for the contracting path will be the same while the weights used for the expansive path will be different. We are also using an average pooling instead of a maximum pooling, which makes more sense for a blur as we are considering all the values. Moreover, since we are not dealing with a classification problem but a regression problem, we will only use one 3x3 convolution without bias and without activation. We can evaluate how many parameters we are going to train. Each convolution in the contracting path contains one 3x3 convolutions, receiving 1 channel and giving 1 channel. The bottleneck contains one circular convolution as well. Finally, each convolution in the expansive path receives 2 channels, one from the contracting path and one from the up sampling. The up sampling contains 4 parameters. The final convolution is just a simple 1x1 convolution. Therefore, we have $1 \cdot 9 + 1 \cdot 9 + 9 \cdot 4 + 9 \cdot 2 \cdot 9 + 1 = 217$ parameters to train.
Remark : We call a convolution a 2D convolution.

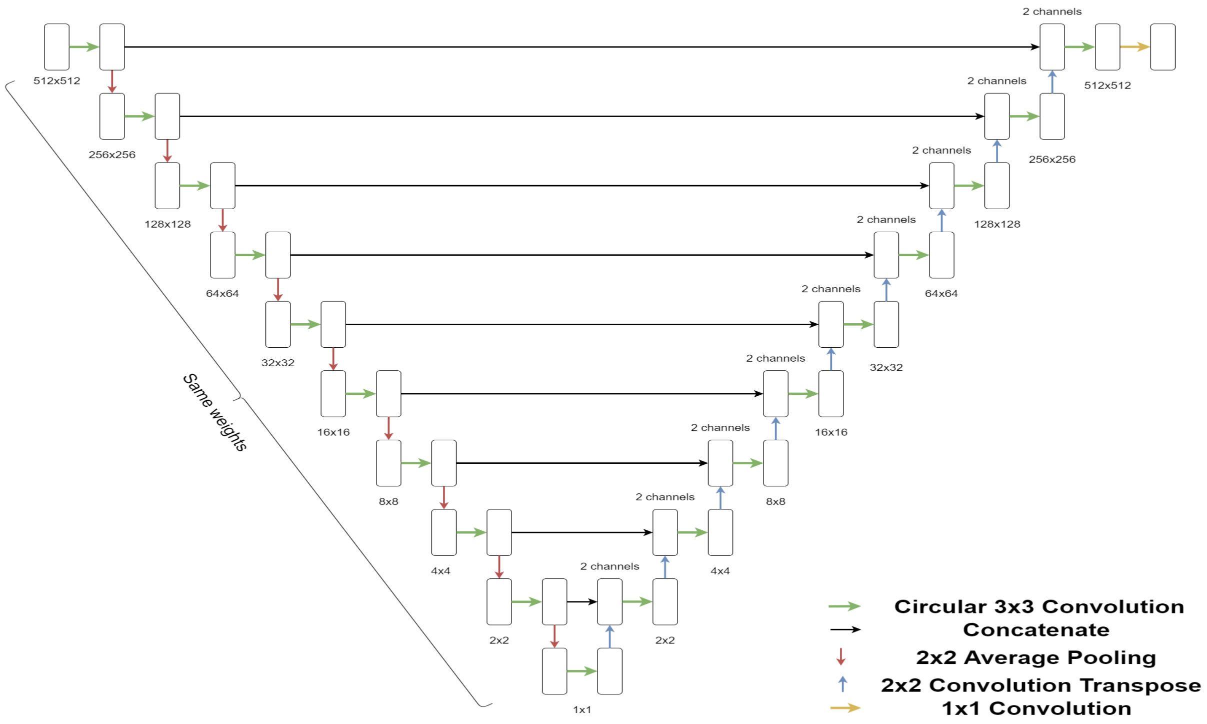


Figure 10: U-Net architecture for blurring

Besides that, we will use circular convolutions instead of zero-padding convolutions. The reason for this is that we consider our images as being periodic. Indeed, in the Fourier domain, convolutions are performed periodically on all values.

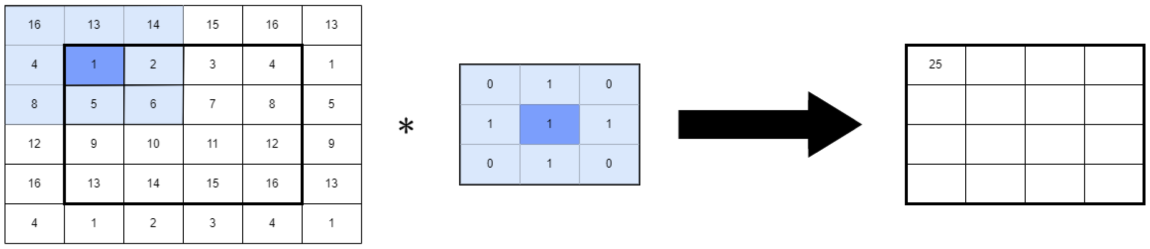


Figure 11: Circular convolution

Figure 12: Examples of training images

We train the U-Net to perform Gaussian blurs with $\sigma = 3$ and $\sigma = 5$ arbitrarily. We remind the Fourier Transform of the Gaussian G_σ in dimension 2 $G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$ is $\widehat{G_\sigma}(\xi,\eta) = e^{-\sigma^2\frac{\xi^2+\eta^2}{2}}$.

Indeed, we have $\widehat{G_\sigma}(\xi,\eta) = \int_{\mathbb{R}^2}\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}e^{-i\xi x - i\eta y}dxdy = \frac{1}{\sqrt{2\pi\sigma^2}}\int_{\mathbb{R}}e^{-\frac{x^2}{2\sigma^2}-i\xi x}dx\cdot\frac{1}{\sqrt{2\pi\sigma^2}}\int_{\mathbb{R}}e^{-\frac{y^2}{2\sigma^2}-i\eta y}dy = \frac{1}{\sqrt{2\pi\sigma^2}}\int_{\mathbb{R}}e^{-\frac{(x+i\xi\sigma^2)^2}{2\sigma^2}-\frac{\sigma^2\xi^2}{2}}dx\cdot\frac{1}{\sqrt{2\pi\sigma^2}}\int_{\mathbb{R}}e^{-\frac{(y+i\eta\sigma^2)^2}{2\sigma^2}-\frac{\sigma^2\eta^2}{2}}dy = e^{-\frac{\sigma^2\xi^2}{2}}\cdot e^{-\frac{\sigma^2\eta^2}{2}} = e^{-\sigma^2\frac{\xi^2+\eta^2}{2}}$. By using the convolution theorem 2.4, we can label our images for the training.

In order to determine whether the network blurs the image correctly, we use the Mean Squared Error loss (MSE). We are not facing a classification problem but a regression problem where every pixels has to be edited, the MSE loss is therefore appropriate. We use Adam optimizer with a variable learning rate depending on the training state.

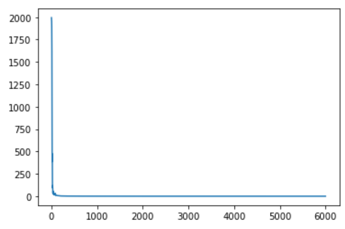


Figure 13: Loss

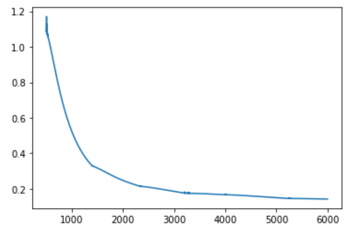


Figure 14: Loss after 500 epochs

Figure 15: Loss with $\sigma = 3$

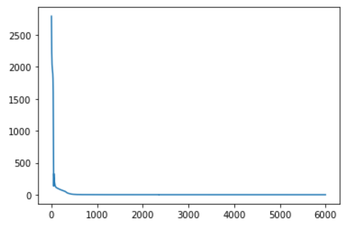


Figure 16: Loss

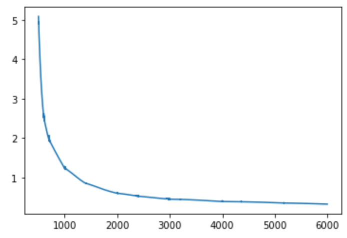
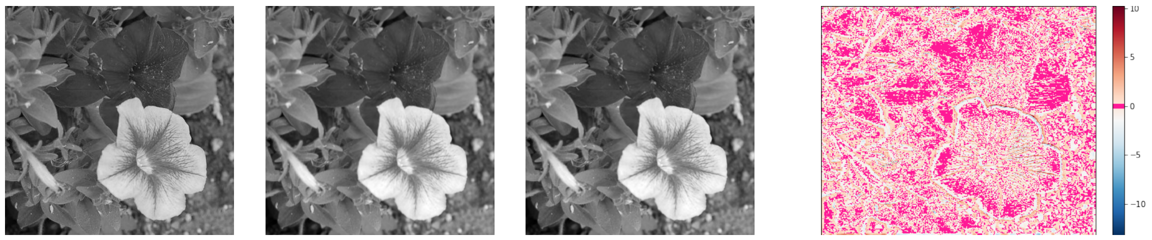


Figure 17: Loss after 500 epochs

Figure 18: Loss with $\sigma = 5$

In both cases, the U-Net fits well to the data. We can evaluate how well it blurs with $\sigma = 5$ by subtracting the image we should get from the image we get with the U-Net.



Input image Target Label Difference between label and target

Figure 19: Example of blurring with U-Net

The values in the difference figure 19 are bounded between -13 and 10 with a mean of -0.001, which means that the blur is done correctly. With more training, we could achieve better results. Finally, if we can manage to blur correctly with $\sigma = 5$, we can blur with a lower σ, which will be used to approximate Land's kernel (2).

Approximating Land's kernel (2) as a sum of N Gaussians can be formulated as an optimization problem. For example, we could solve $\underset{\sigma_1,\sigma_2,\ldots,\sigma_n}{\arg\min} \int_{[1;M]^2} |\sum_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x^2+y^2)}{2\sigma_i^2}} - k(x,y)|dxdy$ such that $\sigma_1, \sigma_2, ..., \sigma_N > 0$ and $M \in]1; +\infty[$ being half of the image size for instance. The reason we start the integral at 1 is due to the vertical asymptote in $(0,0)$ as we can define a Dirac in the discrete case. Using the distributivity of the convolution product, we could get for an image $u \in L^1(\mathbb{R}^2)$, $u * k \approx u * (\sum_{i=1}^{N} G(\sigma_i)) = \sum_{i=1}^{N} u * G(\sigma_i)$ with $G(\sigma_k)$ a Gaussian kernel of standard deviation σ_k, $k \in \{1, ..., N\}$.

3.2 Experiments

Our goal is to apply Land's filter (2) as fast as possible. We can imagine that we would like to perform it on a 4K video with the aim of solving Poisson's equation in real time. To compare the speed of the methods with the Fourier Transform and with the U-Net we will use a data set of 200 images to blur. We will also use the Gaussian blur of $\sigma = 5$.

Method	Treatment	Time (in seconds)
Fourier Method (CPU)	Gaussian blur	4.16052
	Land's filter	5.10455
U-Net (CPU)	Gaussian blur	3.27343
U-Net (GPU Google Colab)	Gaussian blur	0.73865

The CPU used is an Intel Core i7-8700. We see that the U-Net is faster than the Fourier method in both cases. However, we can imagine that we would need at least 3 Gaussians to approximate Land's kernel, which would result in a higher processing time. On the other hand, we clearly see that the GPU of Google Colab performs much better than the CPU, almost 5 times faster.

4 Conclusion

In this supervised research, we studied the properties of Land's kernel [1] and some of its possible applications. We have also mentioned some of the possible operations on images such as zooming, rotating or translating. The study in the sense of distributions of the kernel (2) allowed us to define its Fourier Transform in order to apply the convolution theorem since the function is homogeneous and the Fourier Transform of a radial function is radial. The first application we outlined is Poisson's equation for image editing. This equation is widely used and has many applications in image processing such as stitching two images, add reflectance to an image, enhance contrast, add an object in the background... The other application we presented is the Multiscale Retinex [3] that aims to increase the brightness of an image. In our case, we use the Single Scale Retinex as we are working with single channel images and used Land's kernel (2) instead of a Gaussian kernel. This has the advantage of getting rid of parameters and in our example, seems to give better results although we can not really have flexibility on our operations.

The next objective we had was to implement a neural network capable to perform Land's filter (2) to gain in speed. However, implementing a neural network to apply this filter seems to be a difficult problem due to the vertical asymptote in $(0,0)$. However, we noticed that by symmetry of the kernel, it was possible to approximate it by a sum of Gaussians [7] and [8]. The reason we chose a U-Net architecture [6] is that it is more consistent for larger σ. We have slightly modified the architecture so that the expansive path learns more than the contracting path. This makes sense since we only want the contracting path to give an approximation of the blur to the expansive path. To conclude, we compared speeds to perform a Gaussian blur of the two methods, using the convolution theorem with the Fourier Transform and the U-Net. We observed that the U-Net obtained better results, nevertheless, we would need to have at least 3 Gaussians to approximate Land's kernel (2) correctly, that would be slower that applying Land's kernel (2) directly with the convolution theorem. However, we clearly saw that the GPU on Google Colab achieved good performance to perform a single Gaussian blur. The idea of approximating Land's kernel (2) by Gaussian kernels is very encouraging.

There are still trails to explore, such as solving Poisson's equation [2] that could for instance edit a 4K video in real-time, finding enough σ's to approximate Land's kernel (2) as a sum of Gaussians [7] and [8] or perform Screen Space Ambient Occlusion [9], a technique used in computer graphics to simulate the ambient occlusion effect in real-time where Land's kernel (2) could be used as a surrounding function.

Image credits :
University of South California - Signal and Image Processing Institute

References

[1] Land, Edwin H. "An alternative technique for the computation of the designator in the retinex theory of color vision." Proceedings of the national academy of sciences 83.10 (1986): 3078-3080.

[2] Pérez, Patrick, Michel Gangnet, and Andrew Blake. "Poisson image editing." ACM SIGGRAPH 2003 Papers. 2003. 313-318.

[3] Land, Edwin H. "Recent advances in retinex theory." Central and peripheral mechanisms of colour vision (1985): 5-17.

[4] Jobson, Daniel J., Zia-ur Rahman, and Glenn A. Woodell. "A multiscale retinex for bridging the gap between color images and the human observation of scenes." IEEE Transactions on Image processing 6.7 (1997): 965-976.

[5] Schwartz, Laurent. "Théorie des distributions." (1966).

[6] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[7] Alecu, Teodor Iulian, Sviatoslav Voloshynovskiy, and Thierry Pun. "The gaussian transform." 2005 13th European Signal Processing Conference. IEEE, 2005.

[8] Lee, Wai Ha. Continuous and discrete properties of stochastic processes. Diss. University of Nottingham, 2010.

[9] Bavoil, Louis, and Miguel Sainz. "Screen space ambient occlusion." NVIDIA developer information: http://developers. nvidia. com 6.2 (2008).