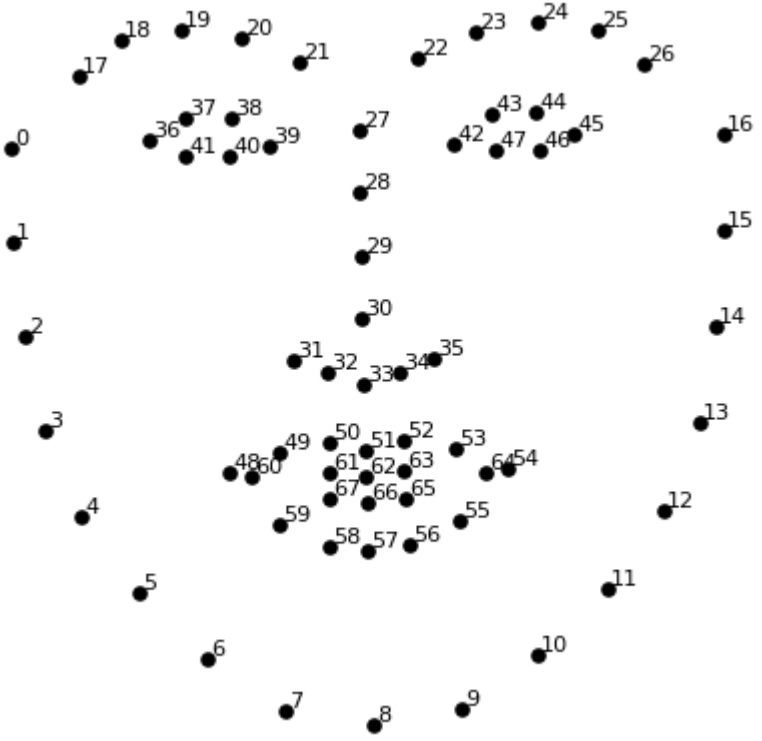
Facial landmarks

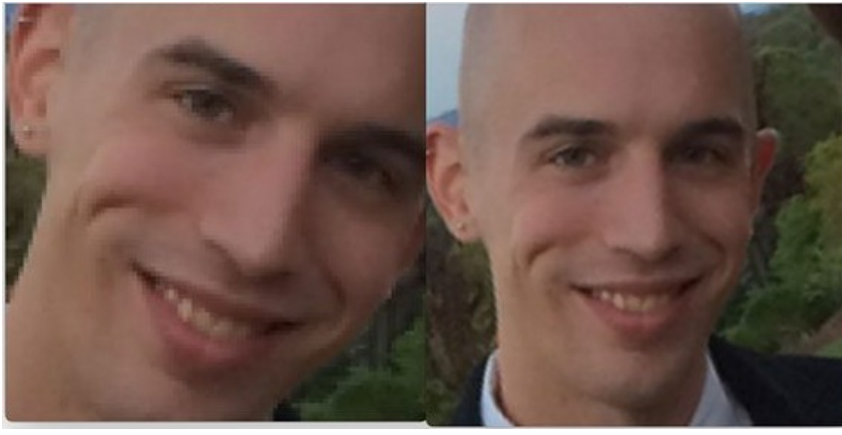


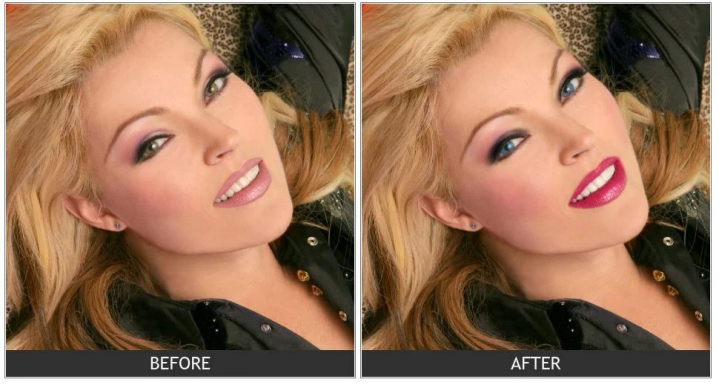
https://www.youtube.com/watch?v=9XXIJ1EQwMc

Usage

- Estimation de la position de la tête
- Morphing

- Maquillage virtuel

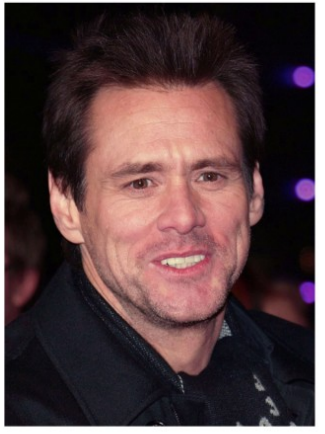


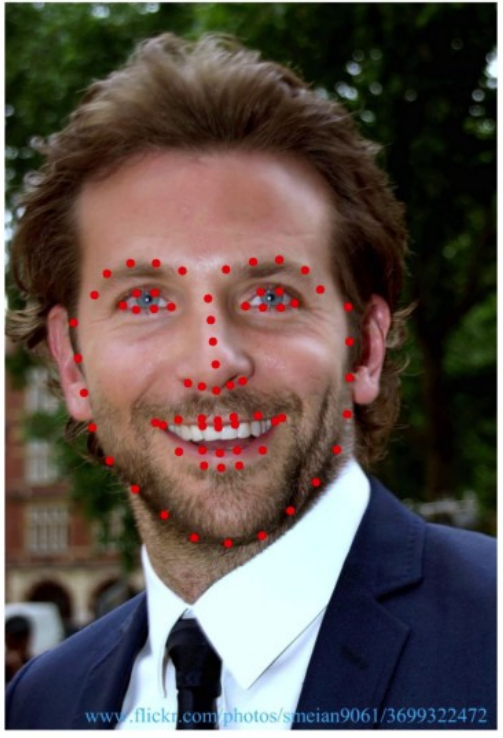


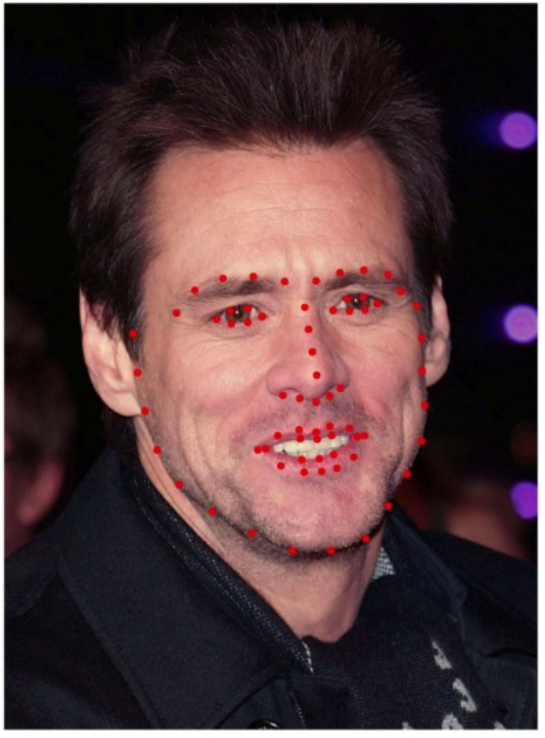
- Filtres….
- Substitution de visage

Source Image

Destination Image

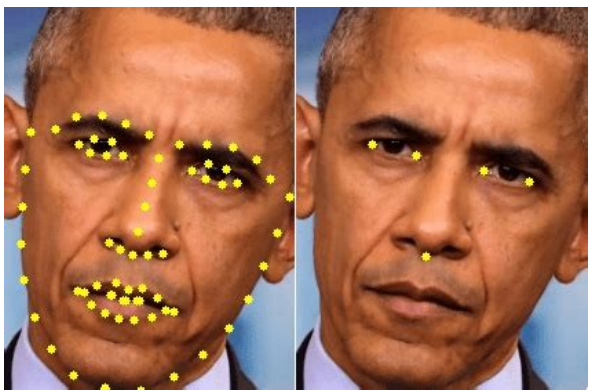


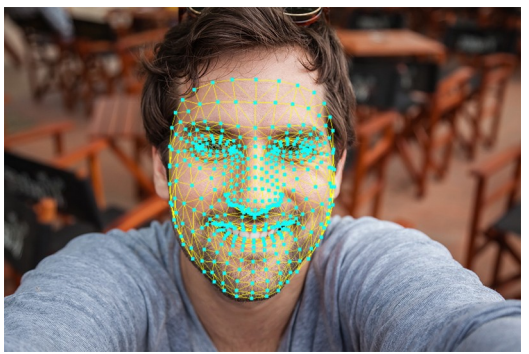

www.flickr.com/photos/smeian9061/3699322472



Facial landmarks ?

- De 5 à 468 points remarquables sur le visage.
- Le plus courant (nombreuses méthodes) : 68 en 2D
- Le mieux (mediapipe Google): 468 en 3D
- Le plus rapide : 5

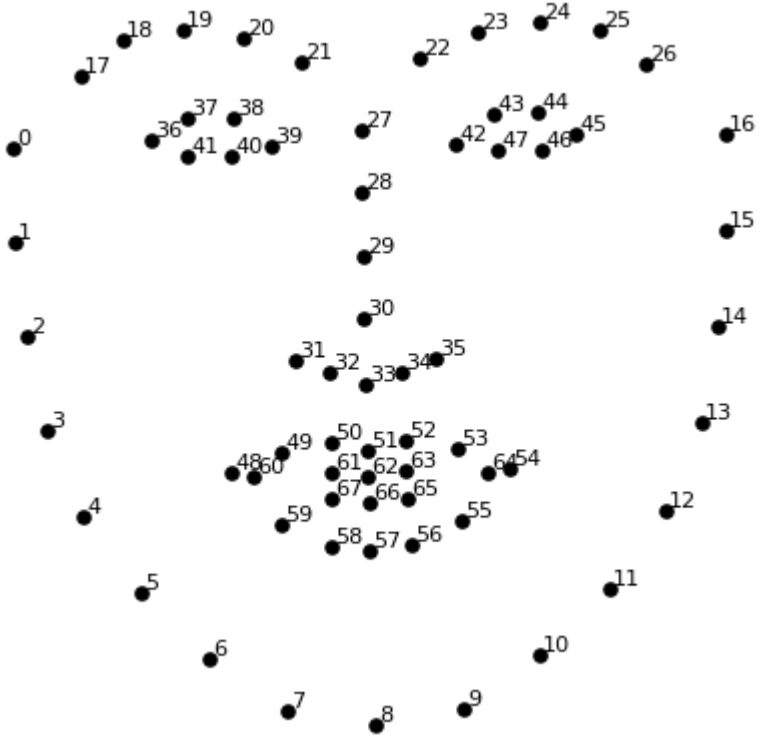


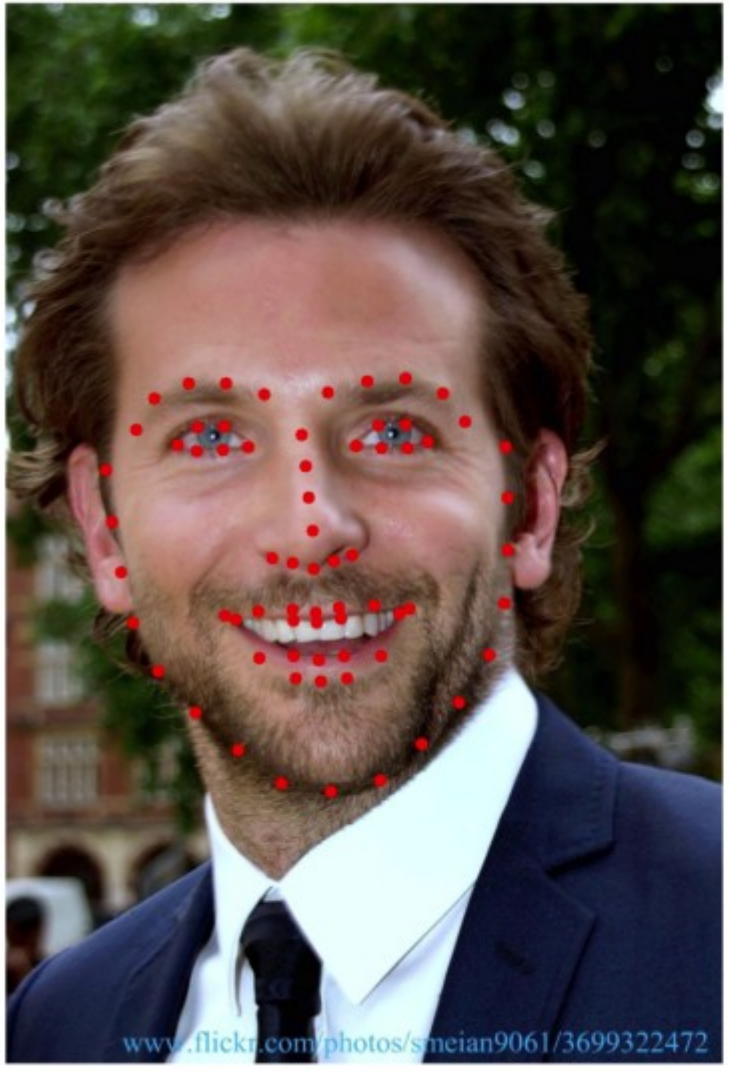


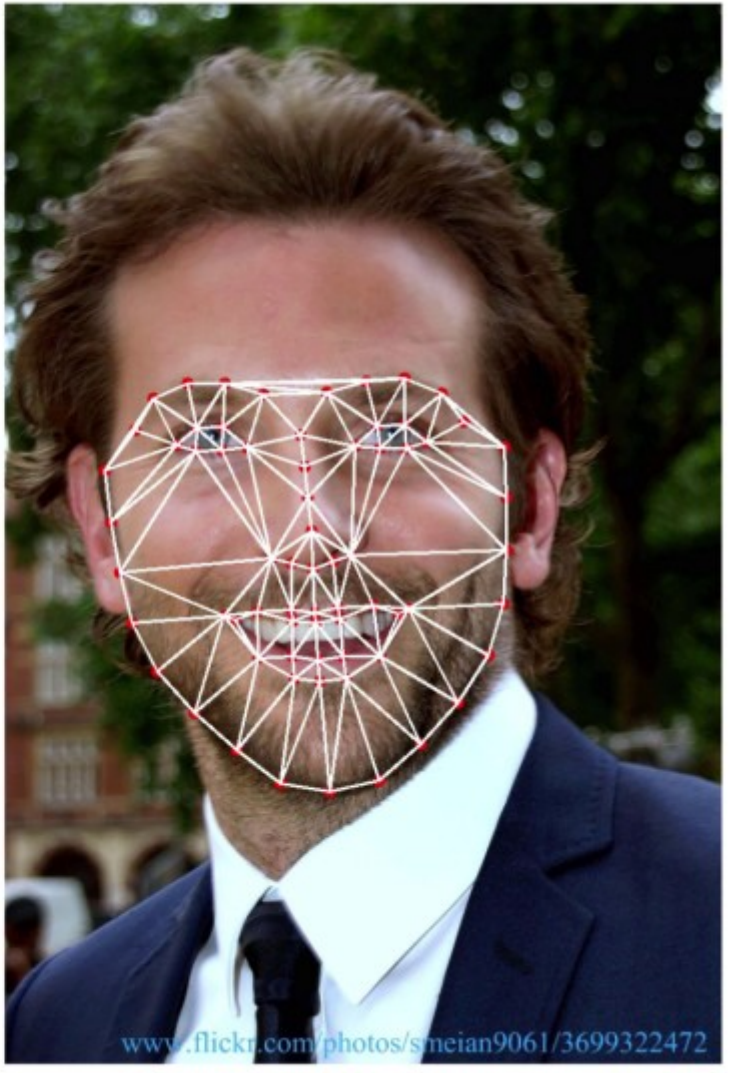
68 points

Numérotés toujours dans le même

- Mâchoire (d'une oreille à l'autre) :
- Sourcil gauche 17-21
- Sourcil droit : 22-26
- Nez : 27-30 | , 31-35)
- Oeil gauche : 36-41
- Oeil droit : 42-47
- Bouche : 48-67




www.flickr.com/photos/smeian9061/3699322472


www.flickr.com/photos/smeian9061/3699322472

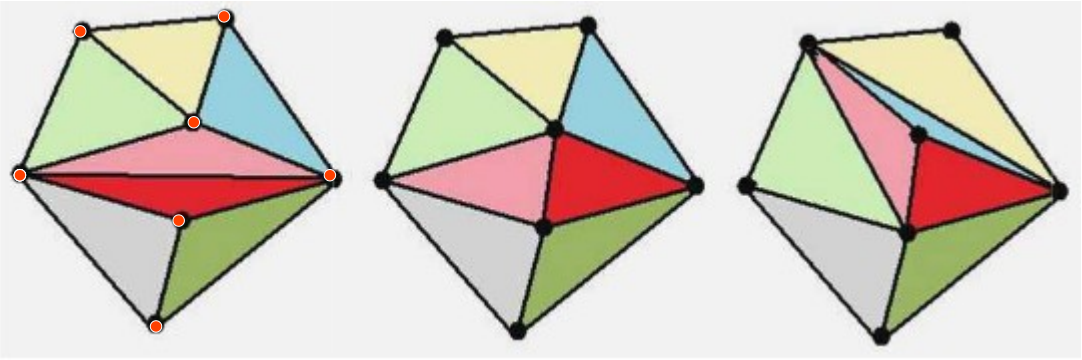
- Pour déformer l’image, et la plaquer sur une autre, il faut déformer des surfaces et pas des points 😱
- On pourrait utiliser des rectangles ou d’autres polygones mais le triangle est :
  - Plus simple (et utilisé par OpenGL, Direct3D…)
  - Très facile à manipuler, en particulier à déformer en changeant de perspective
  - Plus adapté à un visage !

Triangulation de Delaunay

- Triangulation de N points ?

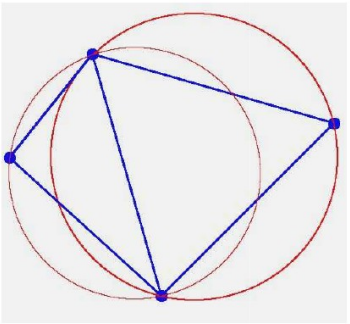
  Une triangulation est formée de triangles dont les sommets sont les points, et qui à eux tous constituent une partition de l'enveloppe convexe des N points.

- Unicité ?

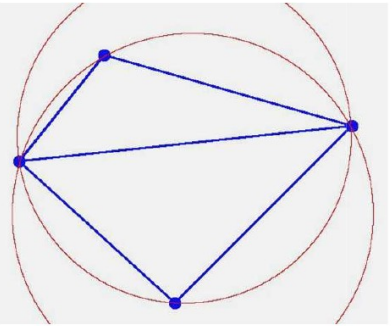


3 triangulations...

- « On appelle triangle de Delaunay un triangle qui a comme sommets trois des points, et tel que son cercle circonscrit n’ait en son intérieur strict aucun point ». 🤔

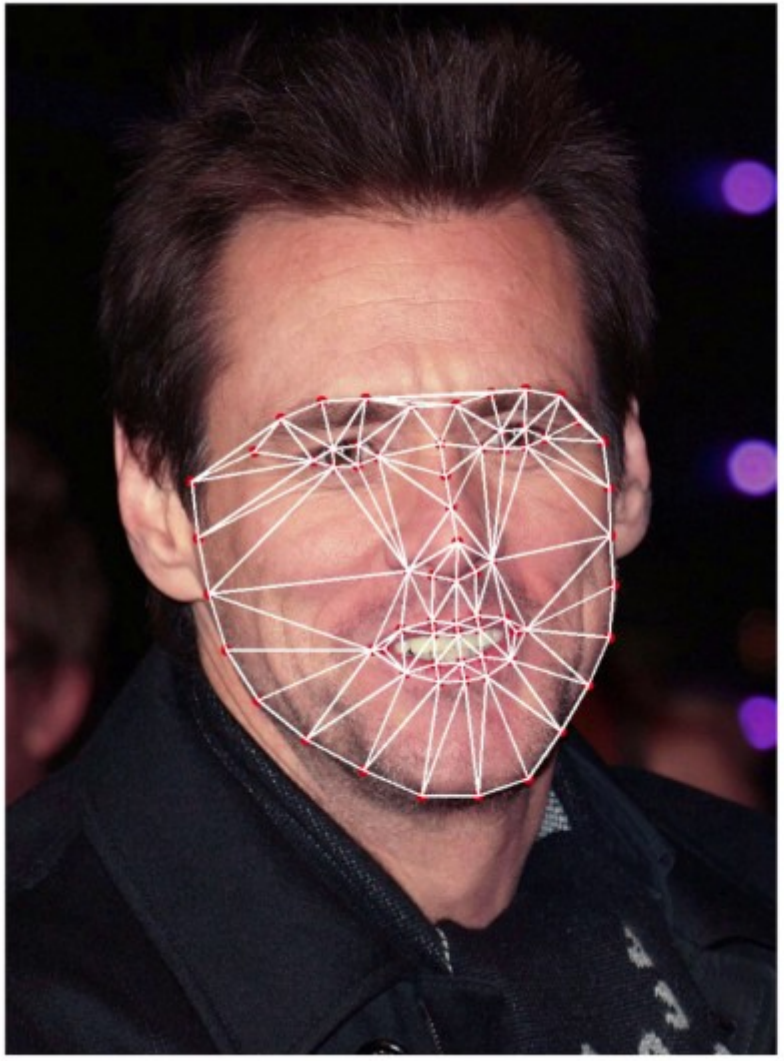


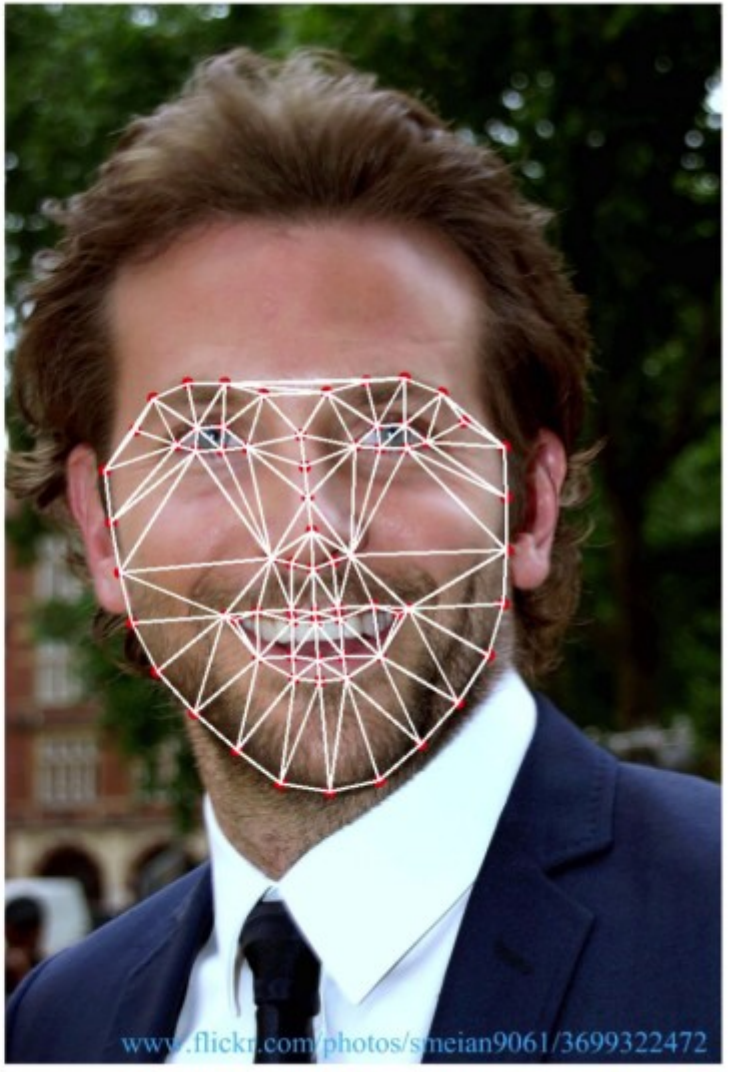
Delaunay

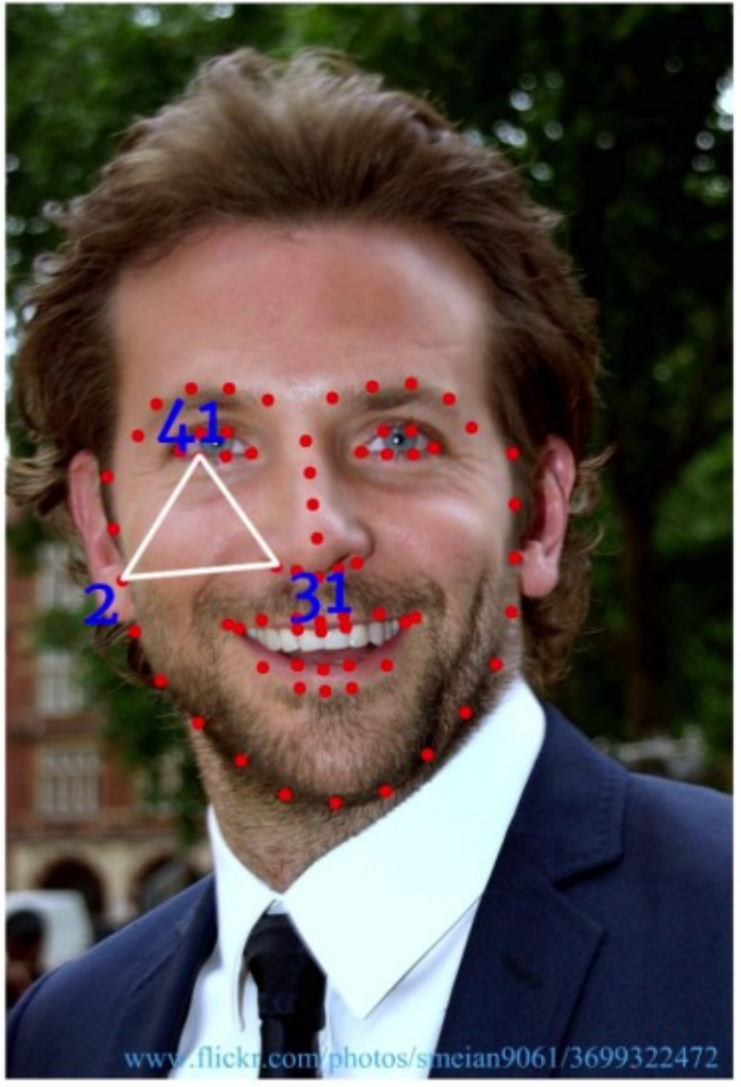


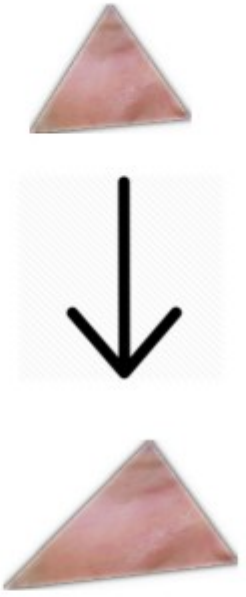
~~Delaunay~~

L’ensemble de tels triangles constitue une triangulation de Delaunay et elle est unique (sauf si elle n’existe pas : s’il y a uniquement des points alignés).
De plus, dans notre cas, elle est calculée une seule fois quel que soit le visage.



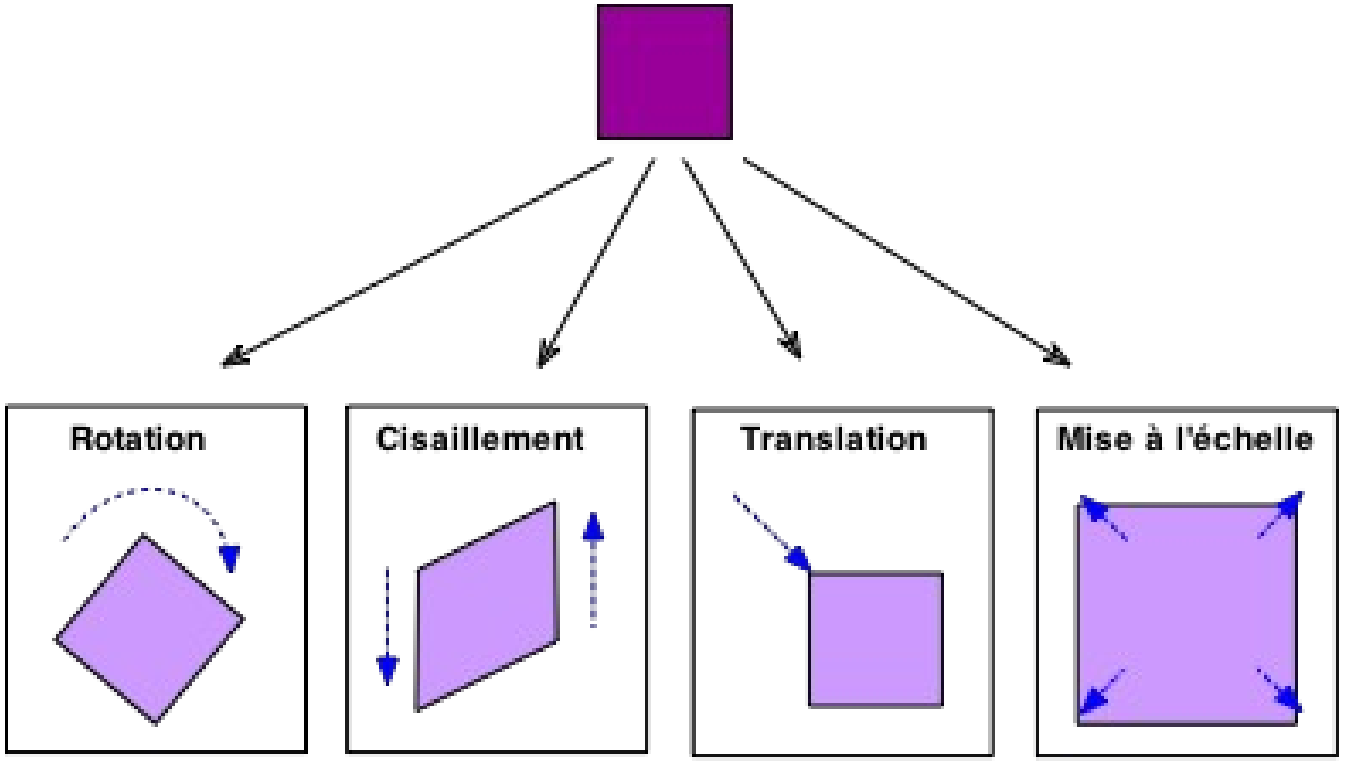

www.flickr.com/photos/smeian9061/3699322472


41
2
31
www.flickr.com/photos/smeian9061/3699322472

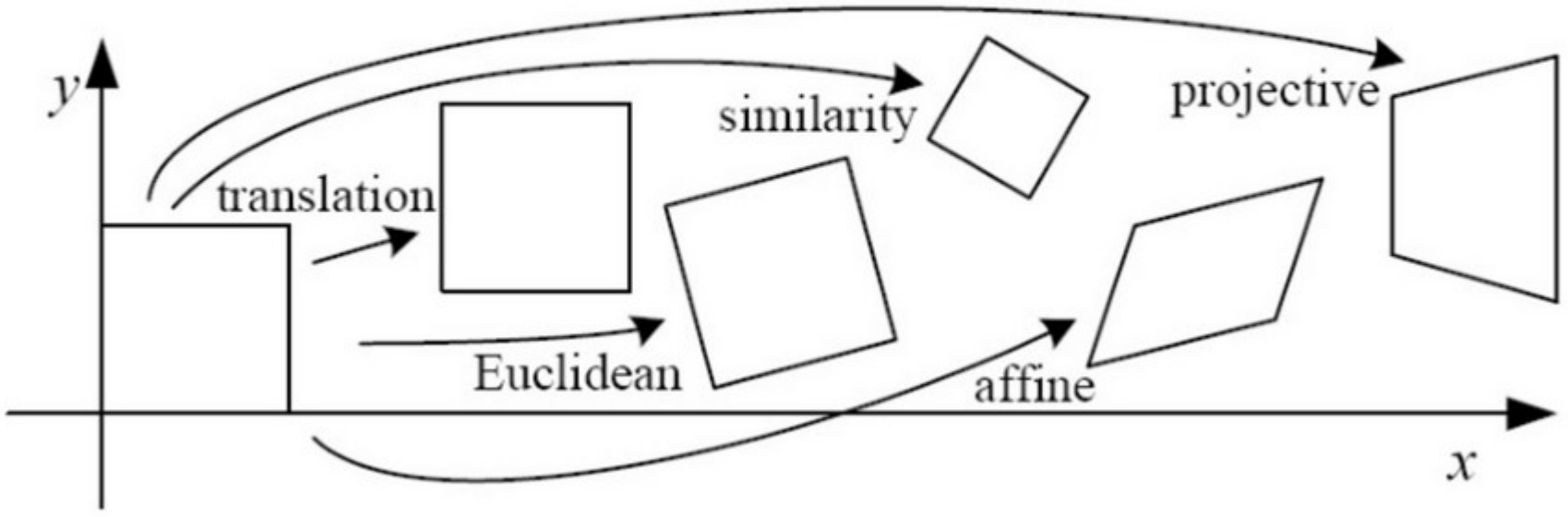


Déformation géométrique

- Classiquement on utilise une transformation **affine** comme approximation linéaire de la projection perspective (non linéaire).
- En coordonnées homogénes :
  - Translation
  - Rotation
  - Changement d’échelle
  - Cisaillement


Rotation
Cisaillement
Translation
Mise à l'échelle

Classification des transformations 2D



Coordonnées homogènes

- Représente des coordonnées 2D avec un vecteur 3D

$$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{Coordonnées homogènes}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \xrightarrow{\text{Point 2D}} \begin{bmatrix} x/w \\ y/w \end{bmatrix}$$

Transformations 2D par matrices 3x3

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Échelle

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & k_x & 0 \\ k_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Étirement
Cisaillement

Transformations affines

- combinaisons de:
  - Transformées linéaires
  - Translations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

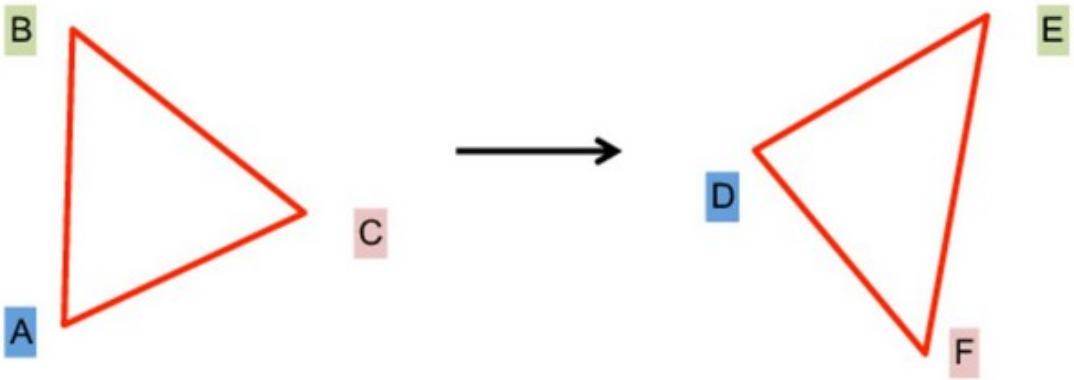
- Propriétés
  - L’origine n’est pas nécessairement préservée
  - Sont préservées: les lignes, lignes parallèles, rapports
  - Les compositions sont aussi des transformées affines (attention à l’ordre)
  - 6 degrés de libertés : 3 points 2D suffisent

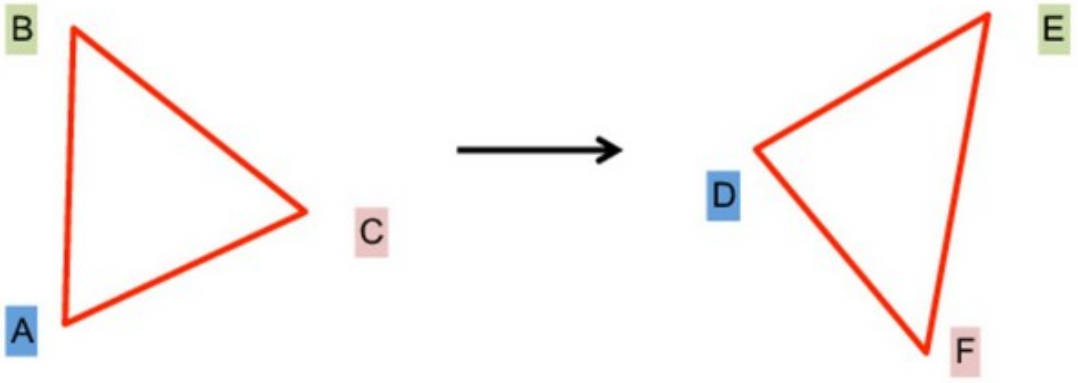
Estimation de la transformation

- A partir des coordonnées des sommets appariés de deux triangles.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$X' = TX$$





$$X' = TX$$

$$\begin{bmatrix} x_D & x_E & x_F \\ y_D & y_E & y_F \\ 1 & 1 & 1 \end{bmatrix} = T \begin{bmatrix} x_A & x_B & x_C \\ y_A & y_B & y_C \\ 1 & 1 & 1 \end{bmatrix}$$

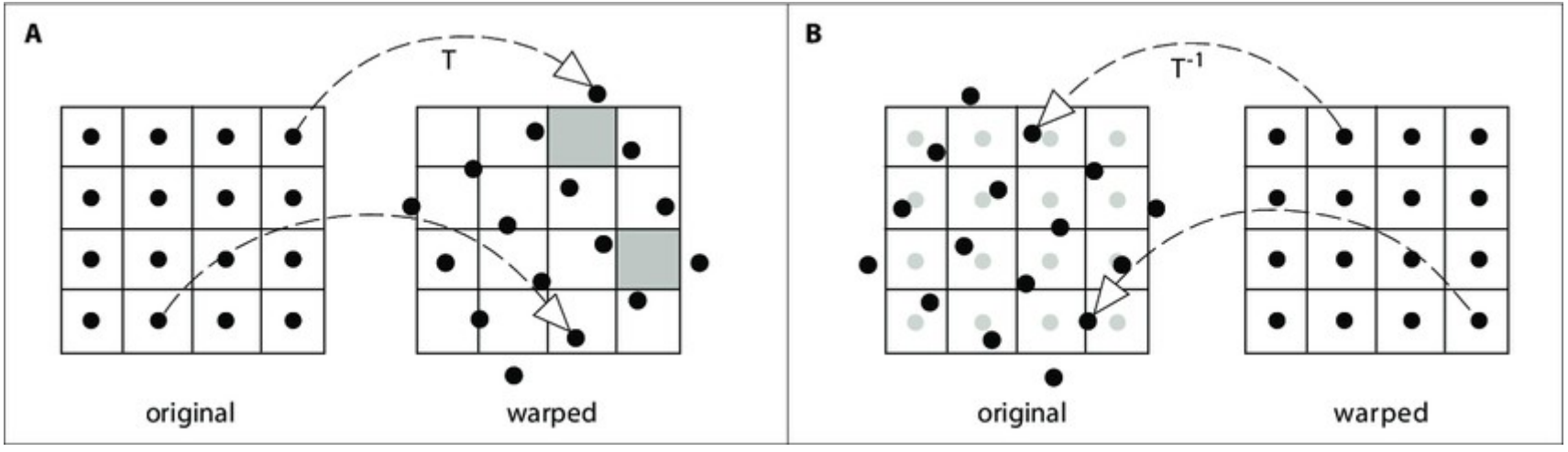
$$T = X'X^{-1}$$

$$T = \begin{bmatrix} y_B - y_C & x_B - x_C & x_B y_C - x_C y_B \\ y_{C-y_A} & x_A - x_C & y_A * x_C - x_A y_C \\ y_A - y_B & x_A + x_B & x_A y_B - y_A x_B \end{bmatrix} /|A|$$

$$|A| = x_A y_C - X_C y_B + y_A x_C + x_A y_B - x_A y_C$$

Texture warping (mapping)

- Pour obtenir l'image déformée on applique la transformation sur les pixels de l'image.

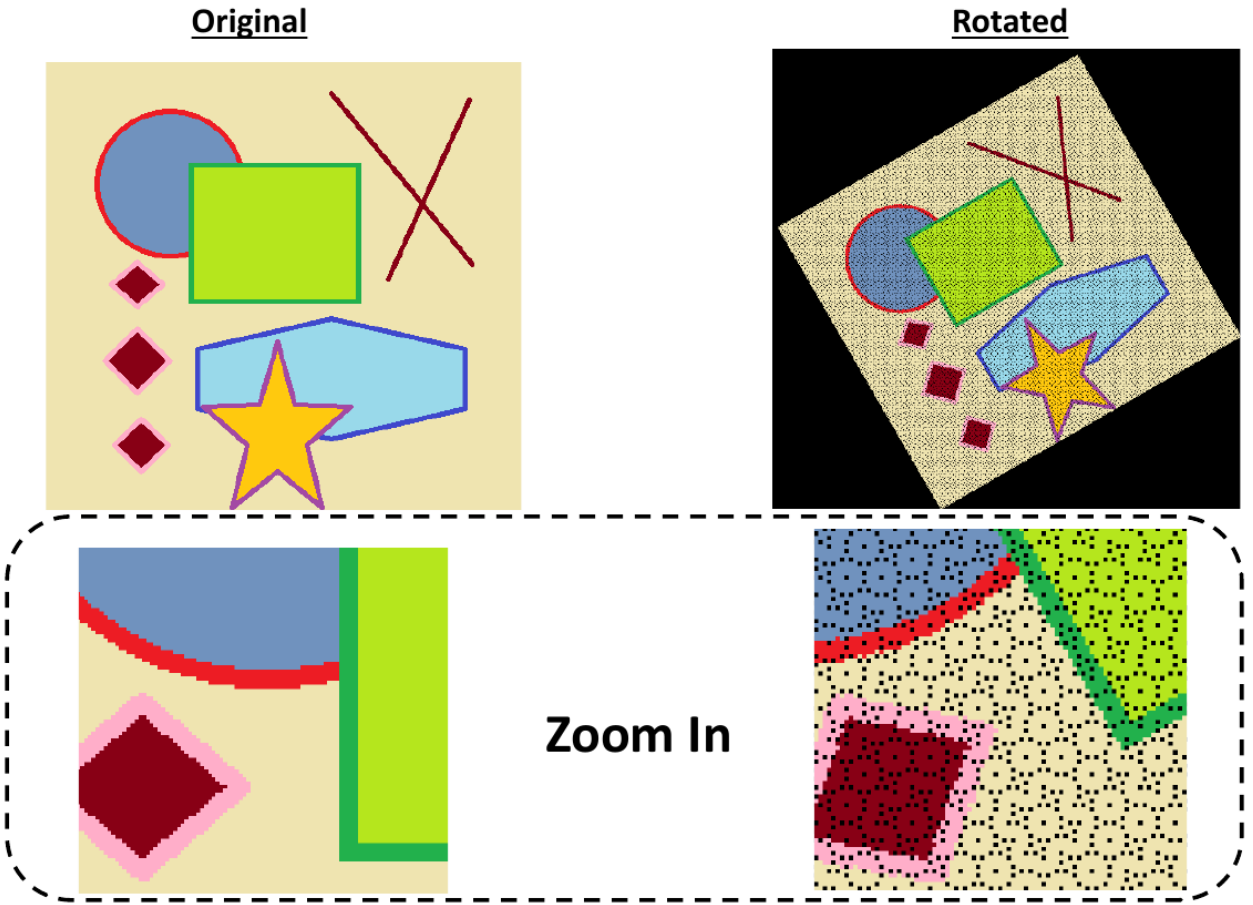


L. Schwarz

Transformation directe →
- trous difficiles à combler
- plusieurs pixels vers la même destination

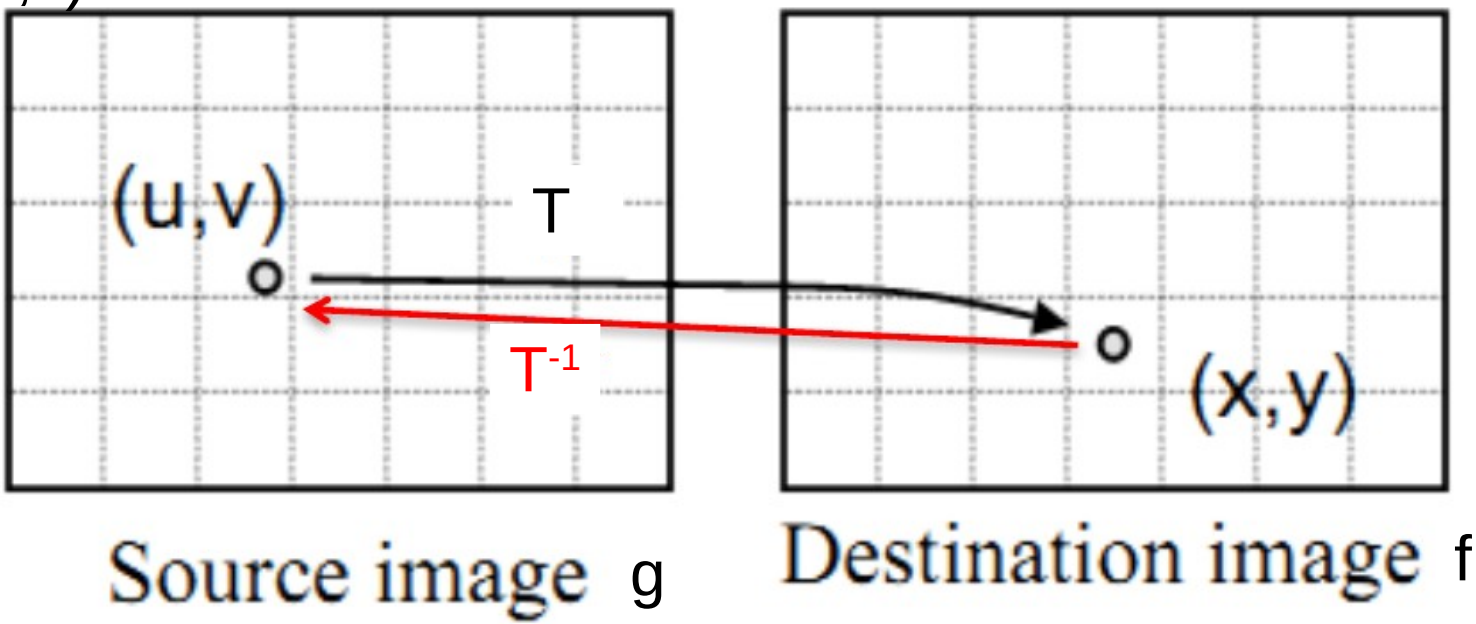
Transformation inverse →
- les coordonnées ne sont pas entières : interpolation des niveaux de gris (ou couleur)

Transformation directe



Transformation inverse

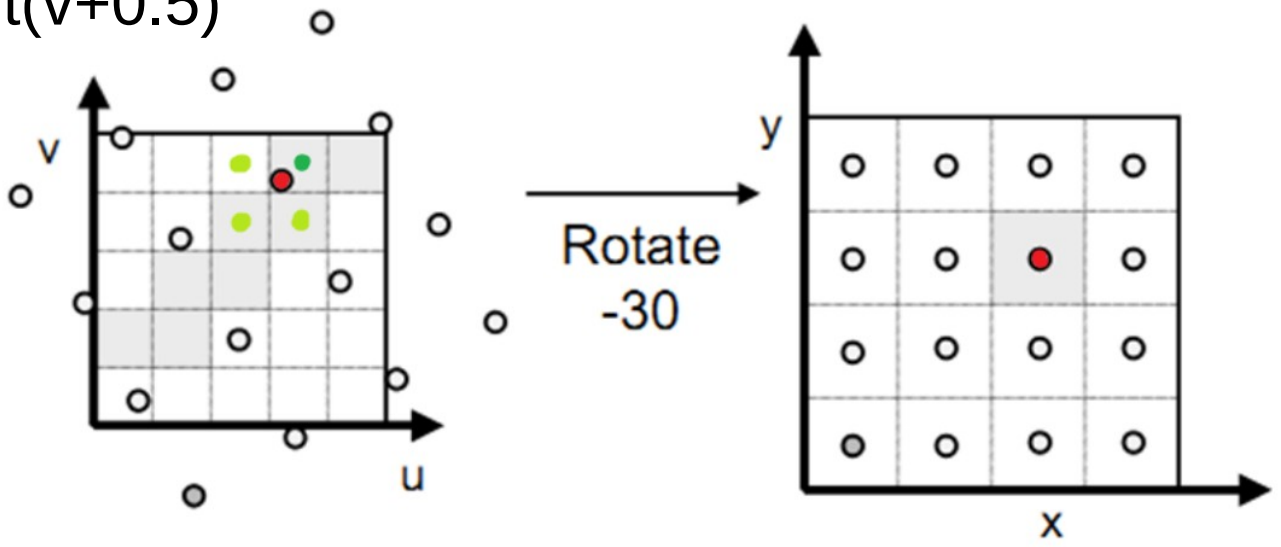
- Pour chaque pixel (x,y) de l'image destination f :
  - (u,v)=T^{-1}(x,y) # coordonnées dans l'image source
  - f(x,y)=g(u,v)



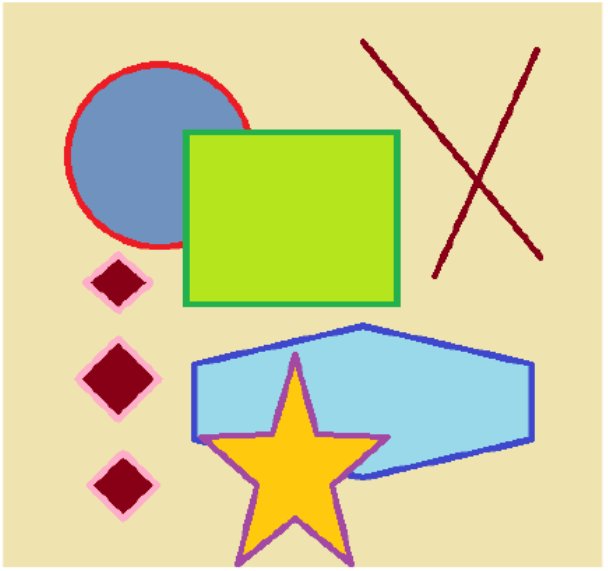
Interpolation ?

- Le plus simple : au plus proche voisin

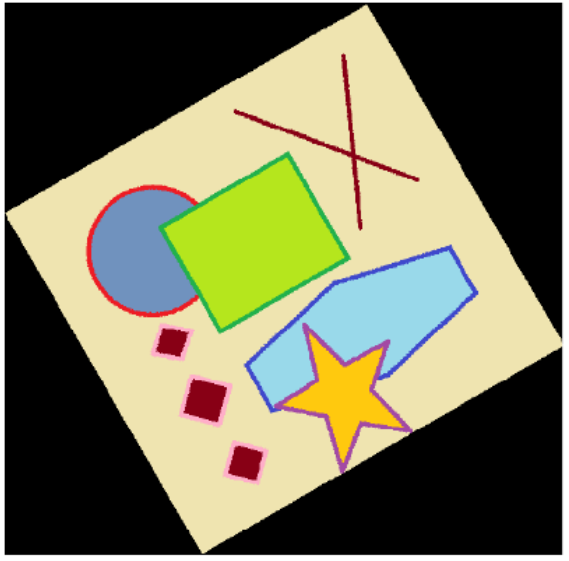
iu,iv=int(u+0.5),int(v+0.5)
f(x,y)=g(iu,iv)

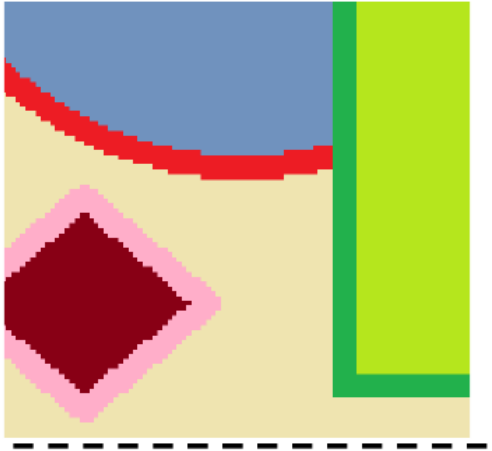


Original

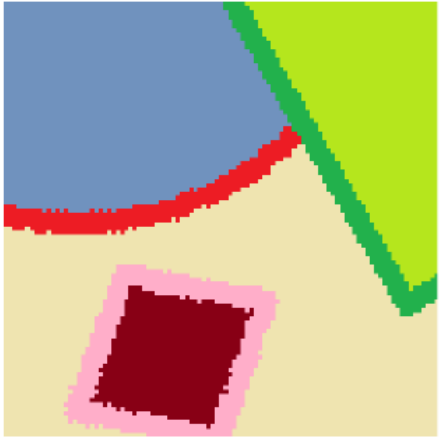


Rotated



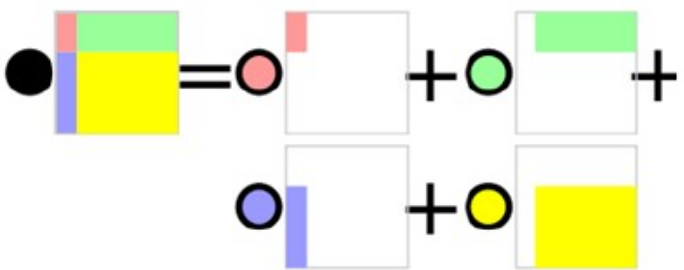


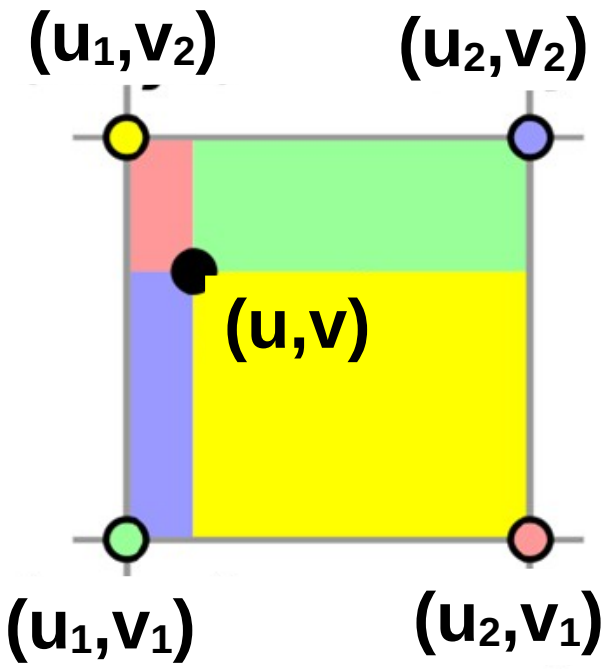
Zoom In



Interpolation bilinéaire

- Interpole la valeur des 4 pixels les plus proches
- Le poids de chaque pixel est (inversement) proportionel à sa distance au pixel « fictif »





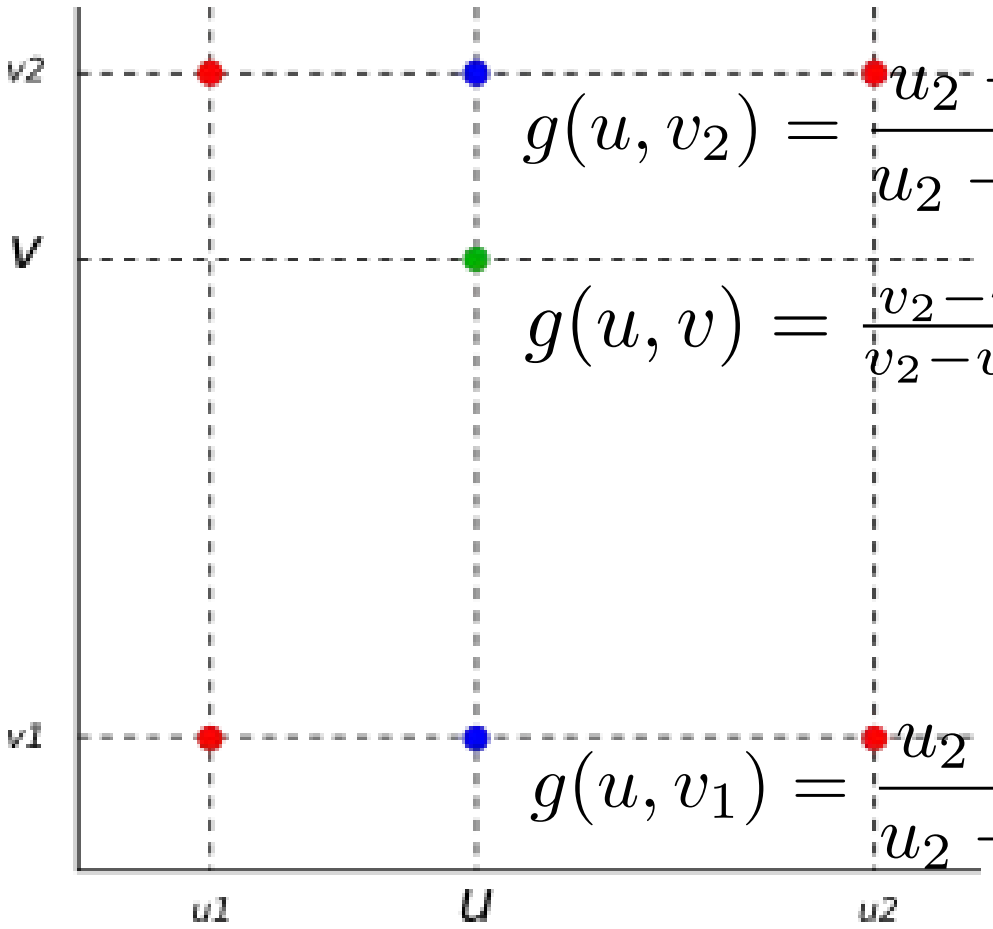
- Deux interpolations successives en u :

$$g(u, v_1) = \frac{u_2 - u}{u_2 - u_1} g(u_1, v_1) + \frac{u - u_1}{u_2 - u_1} g(u_2, v_1)),$$

$$g(u, v_2) = \frac{u_2 - u}{u_2 - u_1} g(u_1, v_2) + \frac{u - u_1}{u_2 - u_1} g(u_2, v_2)).$$

- Puis en v :

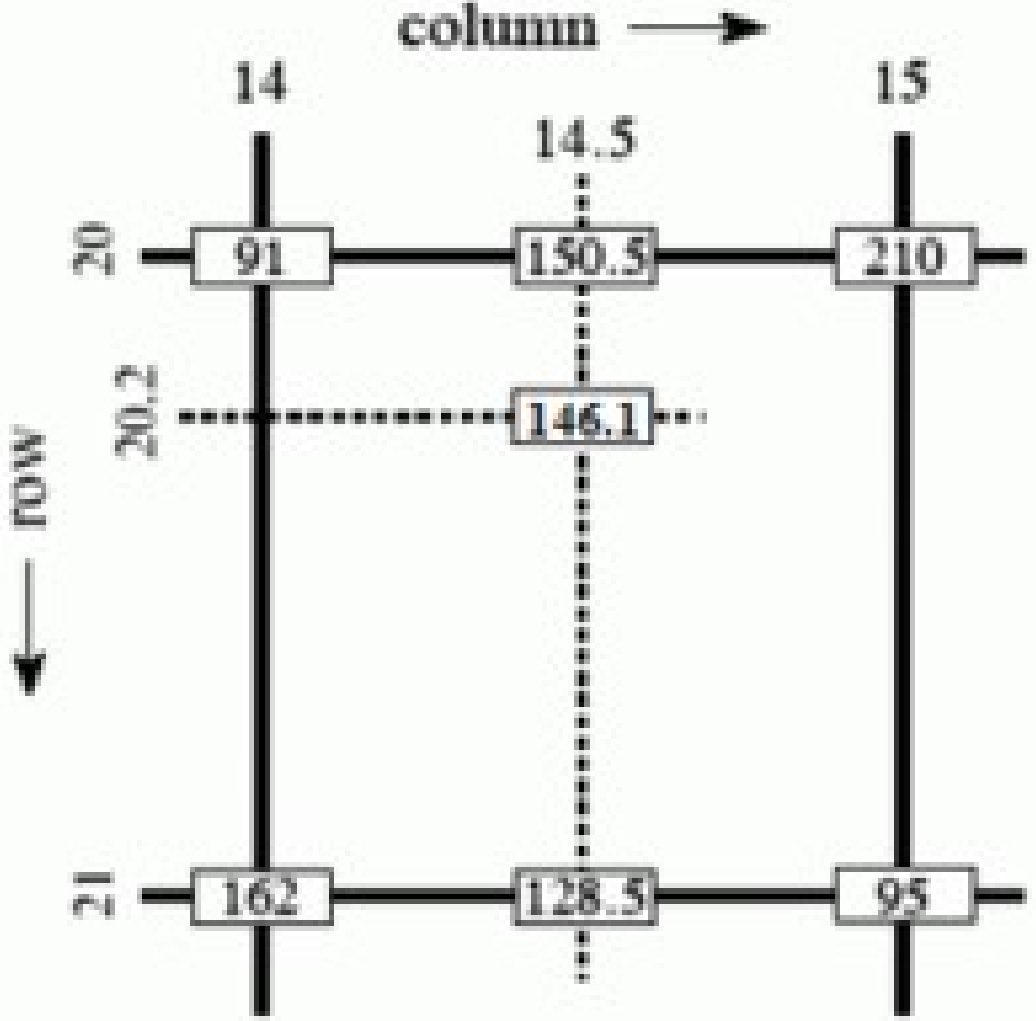
$$g(u, v) = \frac{v_2 - v}{v_2 - v_1} g\left(u, v_1\right) + \frac{v - v_1}{v_2 - v_1} g\left(u, v_2\right)$$



$$g(u, v_2) = \frac{u_2 - u}{u_2 - u_1} g(u_1, v_2) + \frac{u - u_1}{u_2 - u_1} g(u_2, v_2)$$

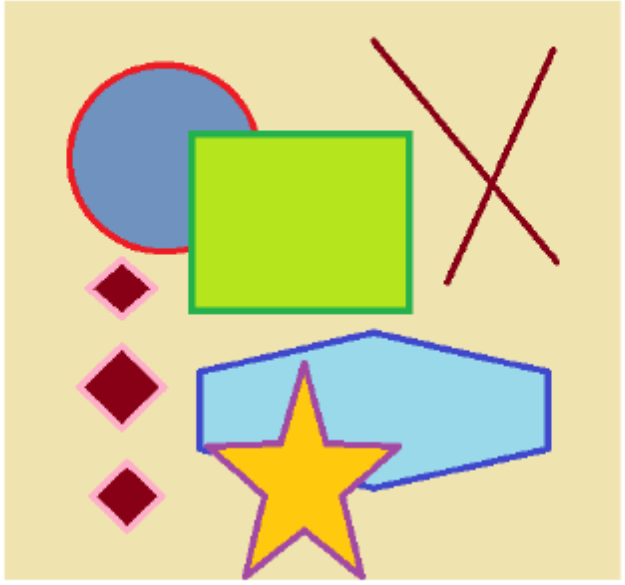
$$g(u, v) = \frac{v_2 - v}{v_2 - v_1} g(u, v_1) + \frac{v - v_1}{v_2 - v_1} g(u, v_2)$$

$$g(u, v_1) = \frac{u_2 - u}{u_2 - u_1} g(u_1, v_1) + \frac{u - u_1}{u_2 - u_1} g(u_2, v_1)$$

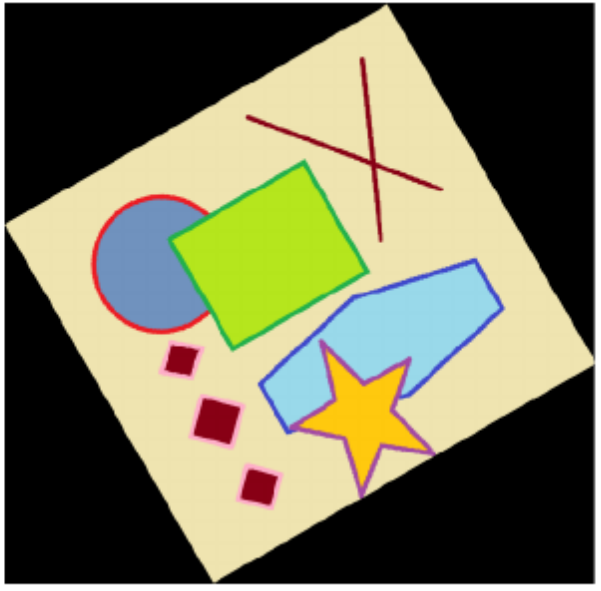


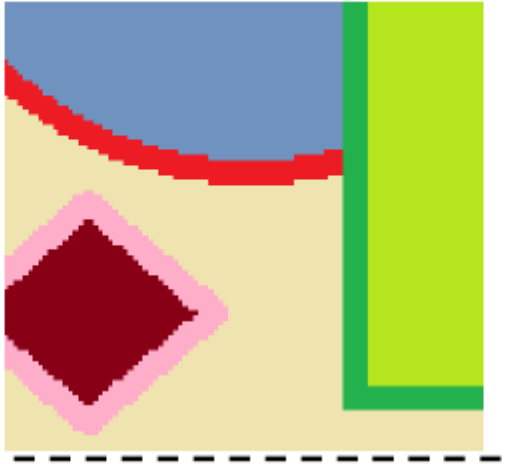
Wikipedia

Original

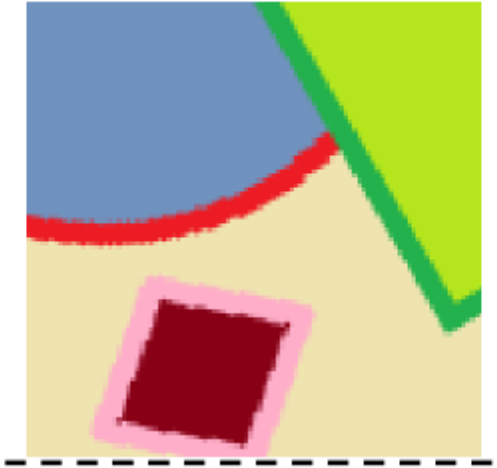


Rotated





Zoom In



- La transformation affine n’est qu’une approximation de la transformée réelle (perspective) mais on ne dispose en général que des coordonnées 2D

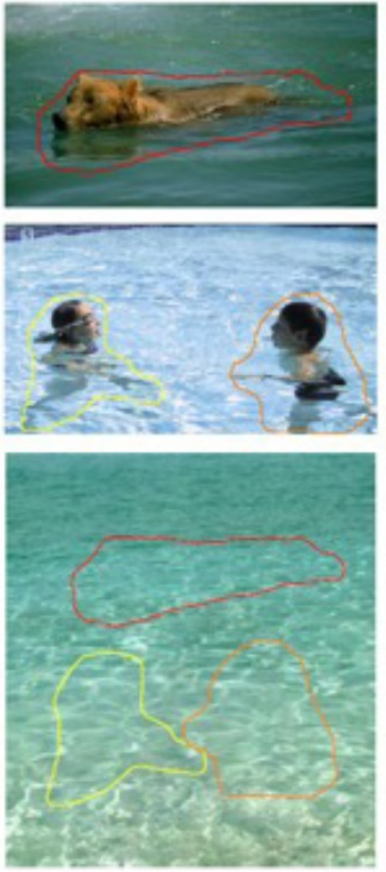
Après application de la transformation géométrique sur chaque triangle

Seamless cloning

- Article : Pérez, P., Gangnet, M., & Blake, A. (2003). Poisson image editing. In ACM SIGGRAPH 2003 Papers (pp. 313-318).
- Slides : d’aprés Frédo Durand -MIT – EECS

copié/collé entre images

- Problème...



sources/destinations

cloning

Rappels

Gradient (niveaux de gris, en couleur on travaille souvent plan par plan) $\nabla f = \left(\frac{\partial f}{dx}, \frac{\partial f}{dy} \right)$

- Pour une image, approché par :

(ou par convolution)

$$\frac{\partial f}{dx} \approx f(x+1, y) - f(x, y)$$

$$\frac{\partial f}{dy} \approx f(x, y+1) - f(x, y)$$

- **Laplacien**

$$\nabla^2(f) = \frac{\partial^2}{\partial x^2} f + \frac{\partial^2}{\partial y^2} f$$

$$\frac{\partial^2 f}{\partial x^2} \approx f(x+1, y) - 2f(x, y) + f(x-1, y)$$

$$\frac{\partial^2 f}{\partial y^2} \approx f(x, y+1) - 2f(x, y) + f(x, y-1)$$

On applique le schéma précédent

$$\frac{\partial^2 f}{\partial x^2} \approx (f(x+1, y) - f(x, y)) - (f(x, y) - f(x-1, y))$$

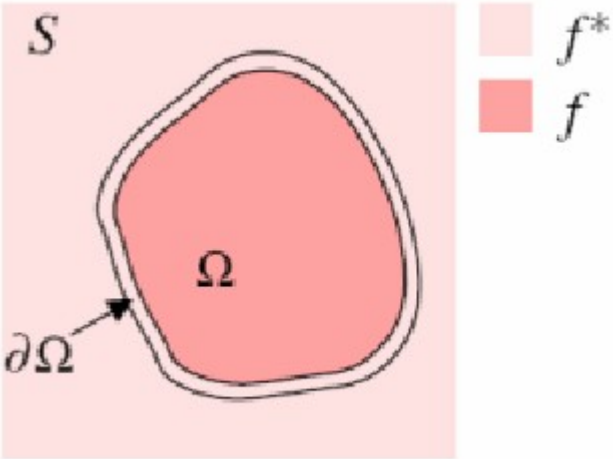
$$= f(x+1, y) - 2f(x, y) + f(x-1, y)$$

Remplissage...

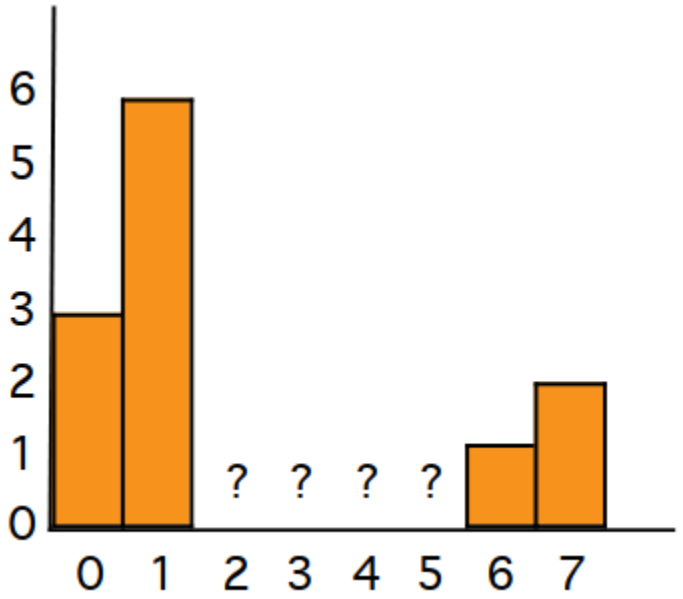
- On connaît l'extérieur (S) de l'objet (f*) et on veut combler l'intérieur (Ω) à partir de son bord (∂Ω)
- On cherche la solution f la plus lisse :

$$\min_f \iint_\Omega |\nabla f|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

- Minimisation de la norme du carré du gradient, condition au bord de Dirichlet (valeur au bord de la solution = valeur connue)
- Equation de Laplace (membrane)



Exemple 1D



Minimiser :

$Q = (f2 - f1)^2$

$+ (f3-f2)^2$

$+ (f4-f3)^2$

$+ (f5-f4)^2$

$+ (f6-f5)^2$

avec f1=6, f6=1

$$\frac{dQ}{df_2} = 2f_2 + 2f_2 - 2f_3 - 12 = 0$$

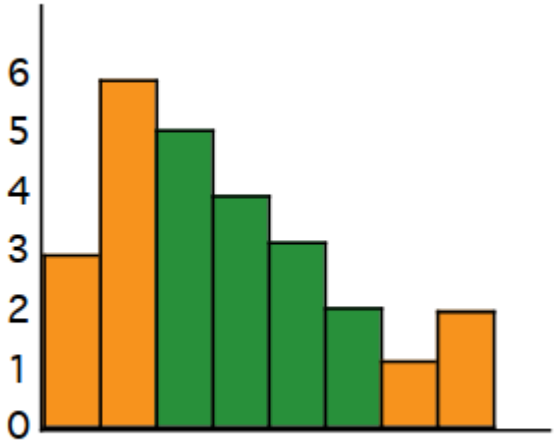
$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2f_3 - 2f_4 = 0$$

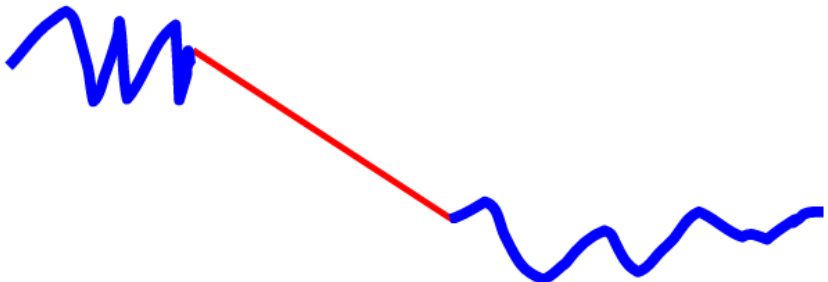
$$\frac{dQ}{df_4} = 2f_4 - 2f_3 + 2f_4 - 2f_5 = 0$$

$$\frac{dQ}{df_5} = 2f_5 - 2f_4 + 2f_5 - 2 = 0$$

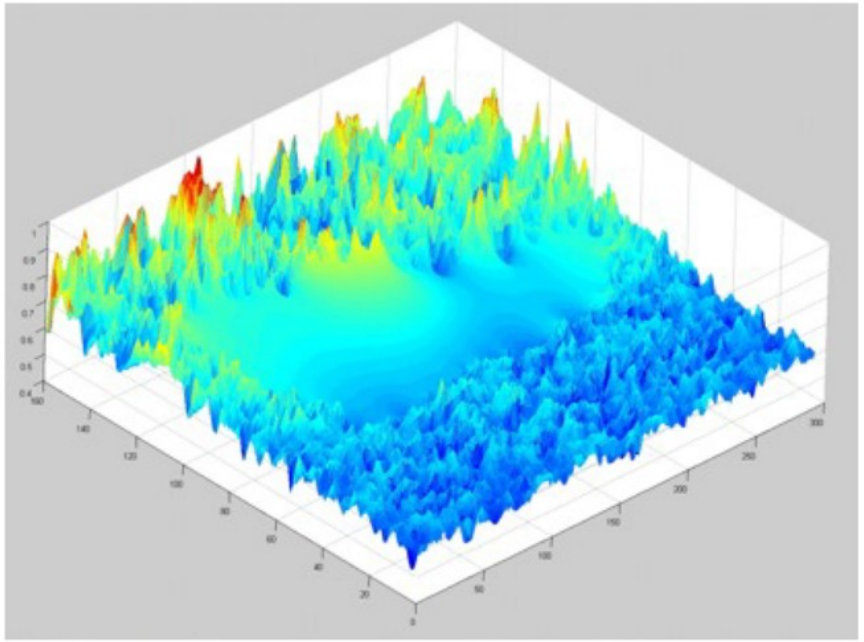
$$\begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 12 \\ 0 \\ 0 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 12 \\ 0 \\ 0 \\ 2 \end{pmatrix} \Longrightarrow \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \\ 3 \\ 2 \end{pmatrix}$$

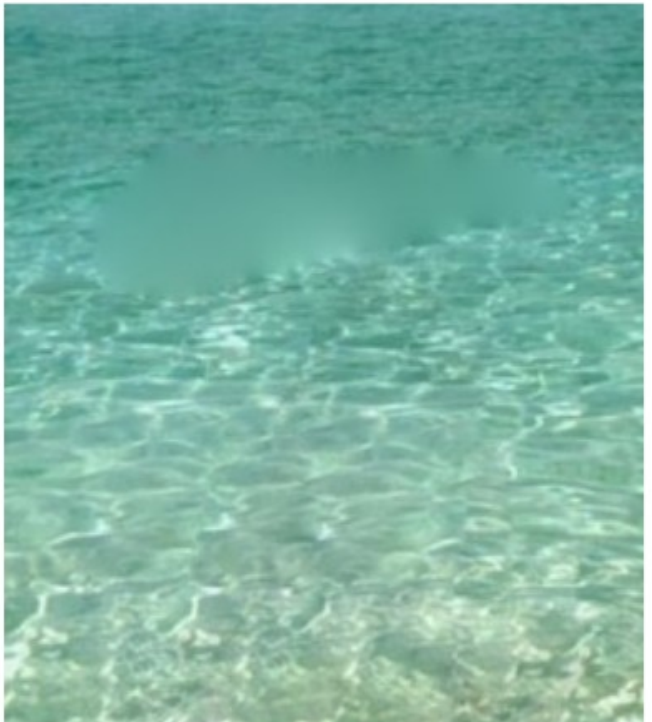




- C’est une interpolation linéaire !
- Obtenue en mimimisant la dérivée (gradient)
- En fait on annule la dérivée seconde
- Rappel : [1 -2 1] est un filtre convolutif pour calculer le Laplacien

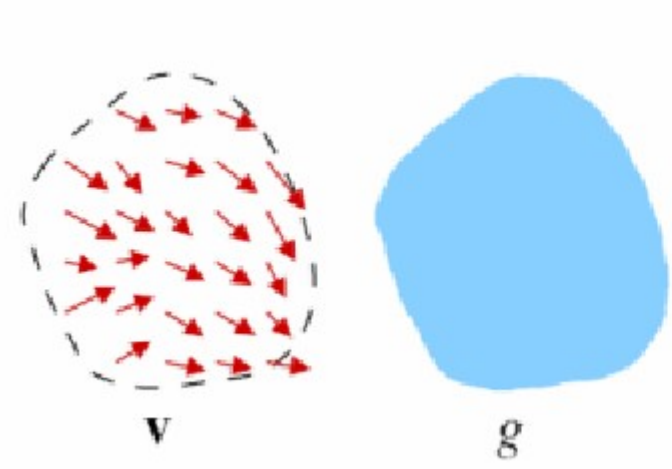


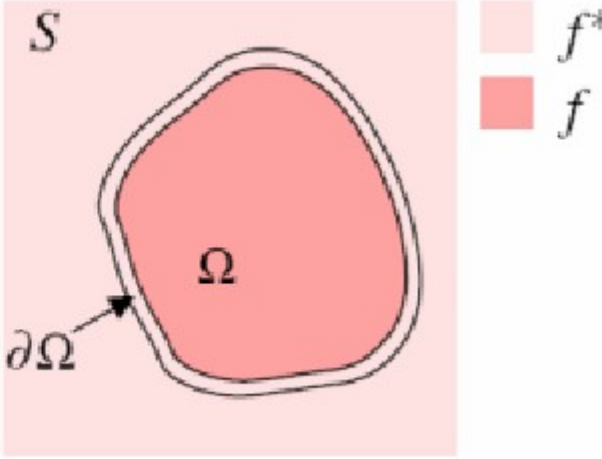
2D



Idée :

- Copier le gradient ($\boldsymbol{v}$) de l'image source (sélection de Ω) dans l'image destination S
- Rendre le nouveau gradient aussi proche que possible du gradient source en tenant compte des valeurs au bord ∂Ω





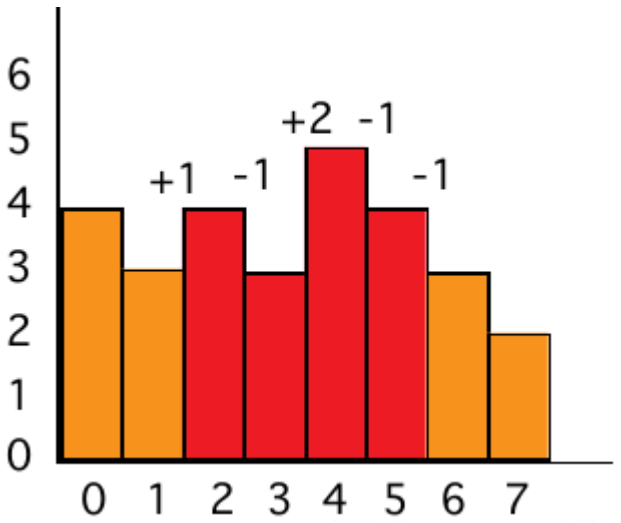
- Connaissant le gradient source (champ de vecteur $\mathbf{v}$), trouver f dans la zone g qui minimise:

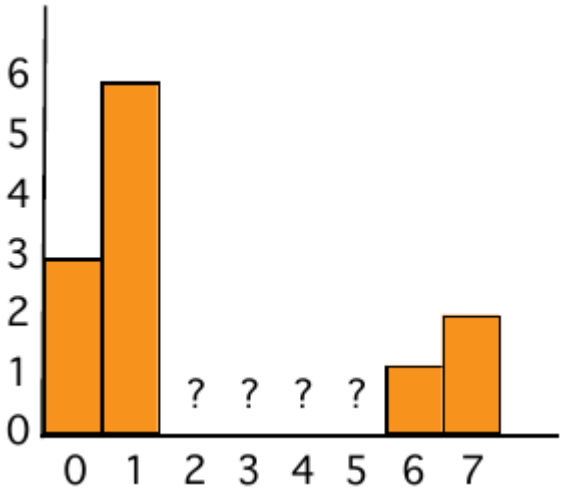
$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{ avec } \; f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

- C’est l’équation de Poisson
- D’où le nom de « Poisson editing »

Exemple 1D

- copier





- Minimiser

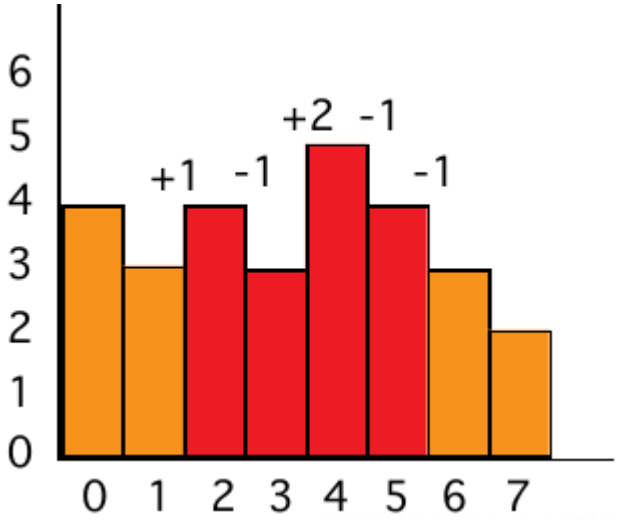
  $Q = [(f2-f1)-1]^2$

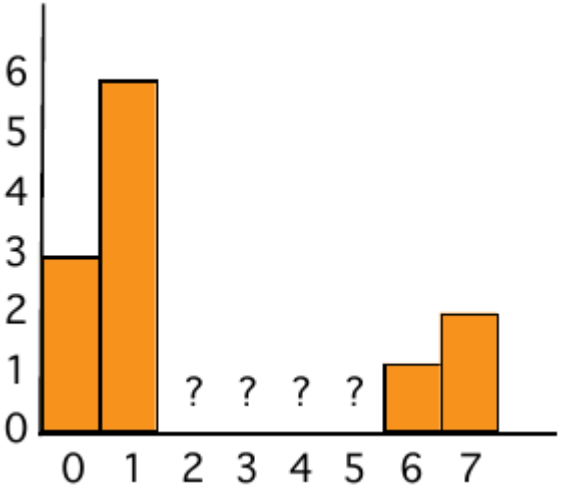
  $+ [(f3-f2)-(-1)]^2$

  $+ [(f4-f3)-2]^2$

  $+ [(f5-f4)-(-1)]^2$

  $+ [(f6-f5)-(-1)]^2$ avec $f1=6$ et $f6=1$





- Minimiser

  $Q= [(f2-f1)-1]^2$

  $+ [(f3-f2)-(-1)]^2$

  $+ [(f4-f3)-2]^2$

  $+ [(f5-f4)-(-1)]^2$

  $+ [(f6-f5)-(-1)]^2$

$f2^2+49-14f2$

$f3^2+f2^2+1-2f3f2+2f3-2f2$

$f4^2+f3^2+4-2f3f4-4f4+4f3$

$f5^2+f4^2+1-2f5f4+2f5-2f4$

$f5^2+4-4f5$

- Minimiser $Q= (f2^2+49-14f^2+ f3^2+f2^2+1-2f3f2 +2f3-2f2+ f4^2+f3^2+4-2f3f4 -4f4+4f3+ f5^2+f4^2+1-2f5f4 +2f5-2f4+ f5^2+4-4f5)$

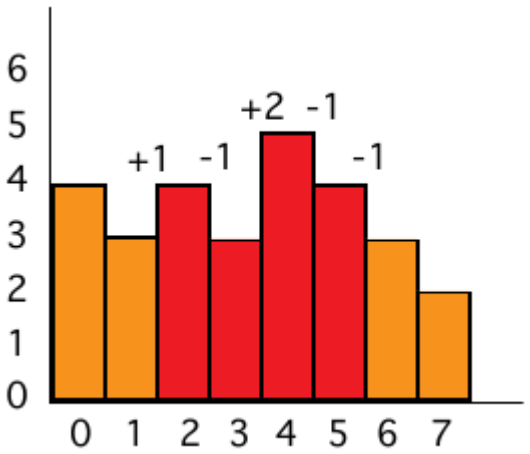
$$\frac{dQ}{df_2} = 2f_2 + 2f_2 - 2f_3 - 16 = 0$$
$$\frac{dQ}{df_3} = 2f_3 - 2f_2 + 2 + 2f_3 - 2f_4 + 4 = 0$$
$$\frac{dQ}{df_4} = 2f_4 - 2f_3 - 4 + 2f_4 - 2f_5 - 2 = 0$$
$$\frac{dQ}{df_5} = 2f_5 - 2f_4 + 2 + 2f_5 - 4 = 0$$

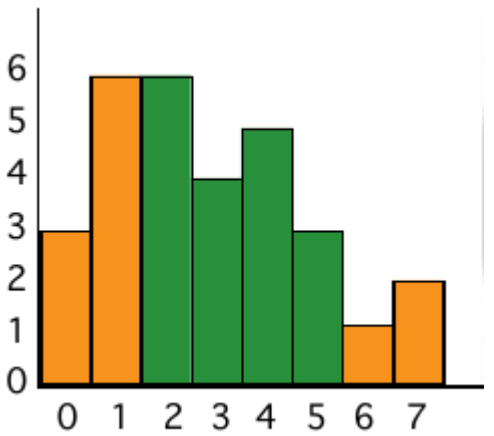
- Matrice creuse (des 0)
- Toujours la même ligne décalée
- Convolution (-2 4 -2)

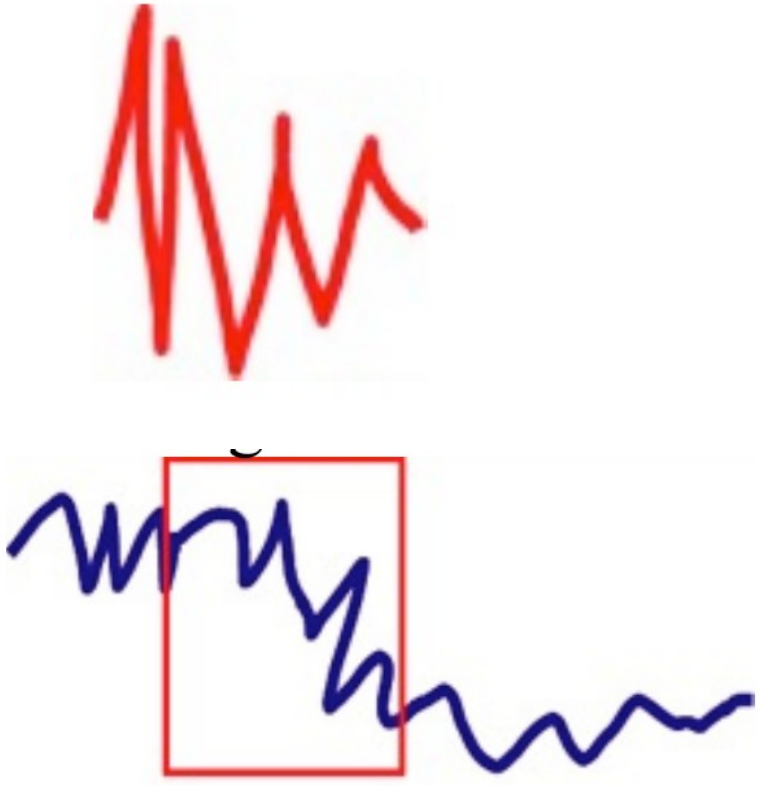
Bref : calcul très optimisable

$$\begin{pmatrix} 4 & -2 & 0 & 0 \\ -2 & 4 & -2 & 0 \\ 0 & -2 & 4 & -2 \\ 0 & 0 & -2 & 4 \end{pmatrix} \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 16 \\ -6 \\ 6 \\ 2 \end{pmatrix}$$

$$\begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \\ 5 \\ 3 \end{pmatrix}$$

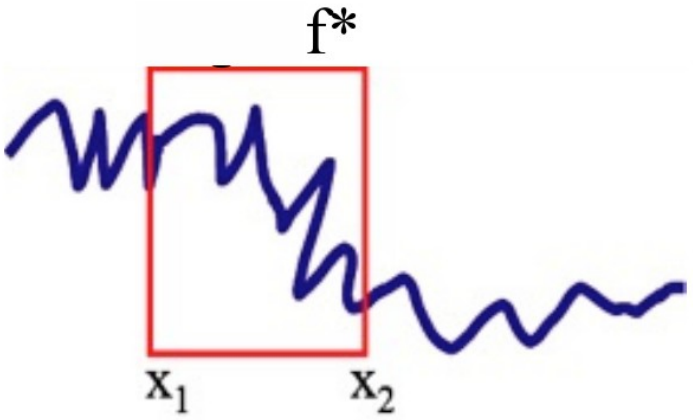


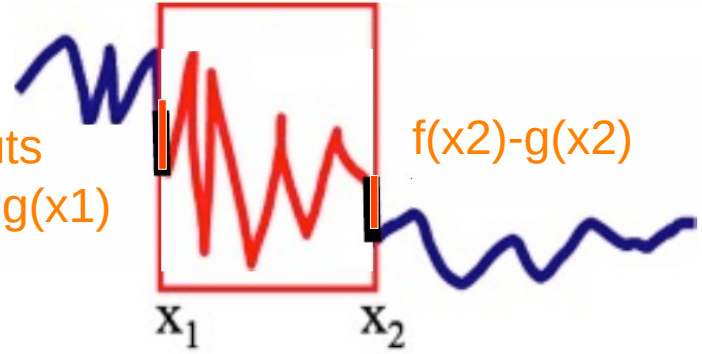




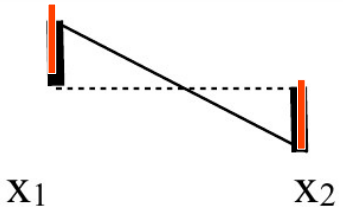
Copier

sur

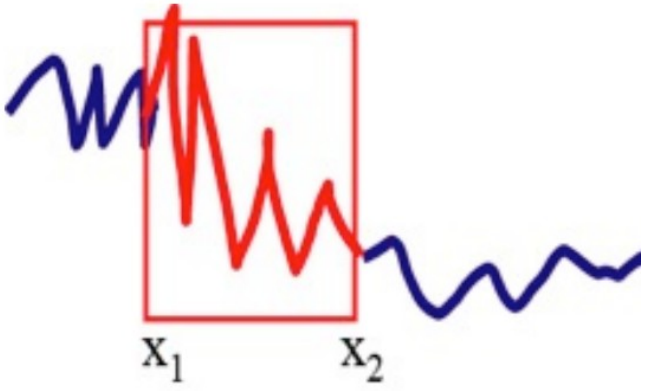




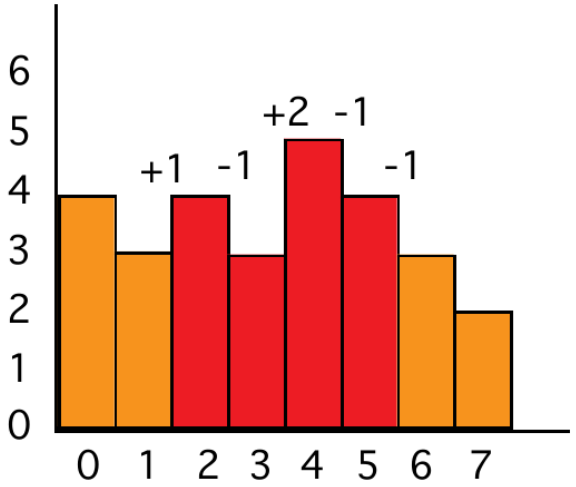
Interpolation linéaire (Laplace)



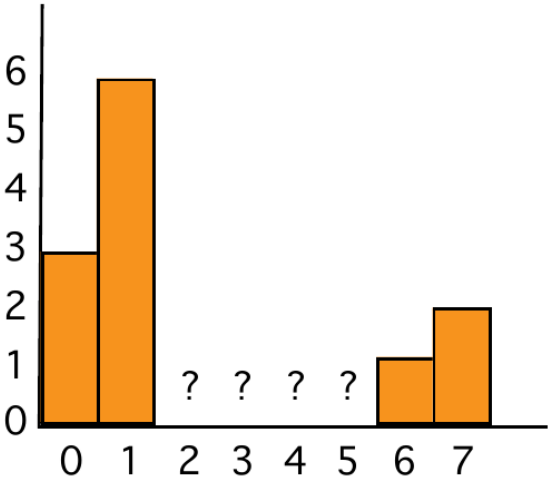
Interpolation linéaire + g



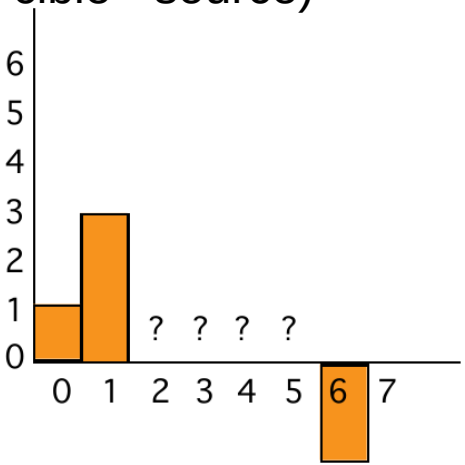
Copier



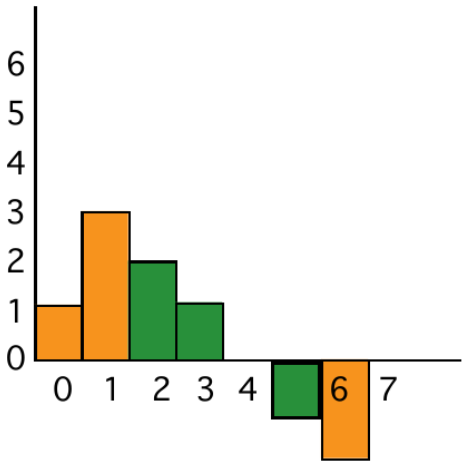
sur



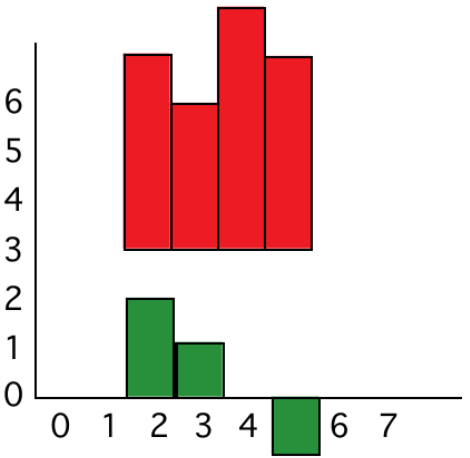
Calcul du saut (différence : cible - source)

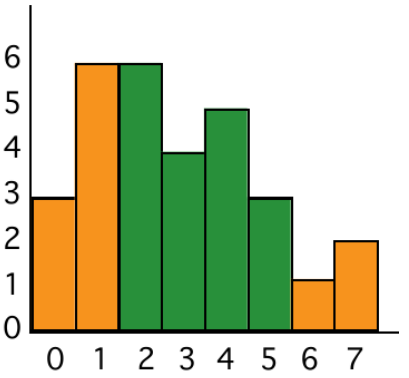


Interpolation Laplace



Addition

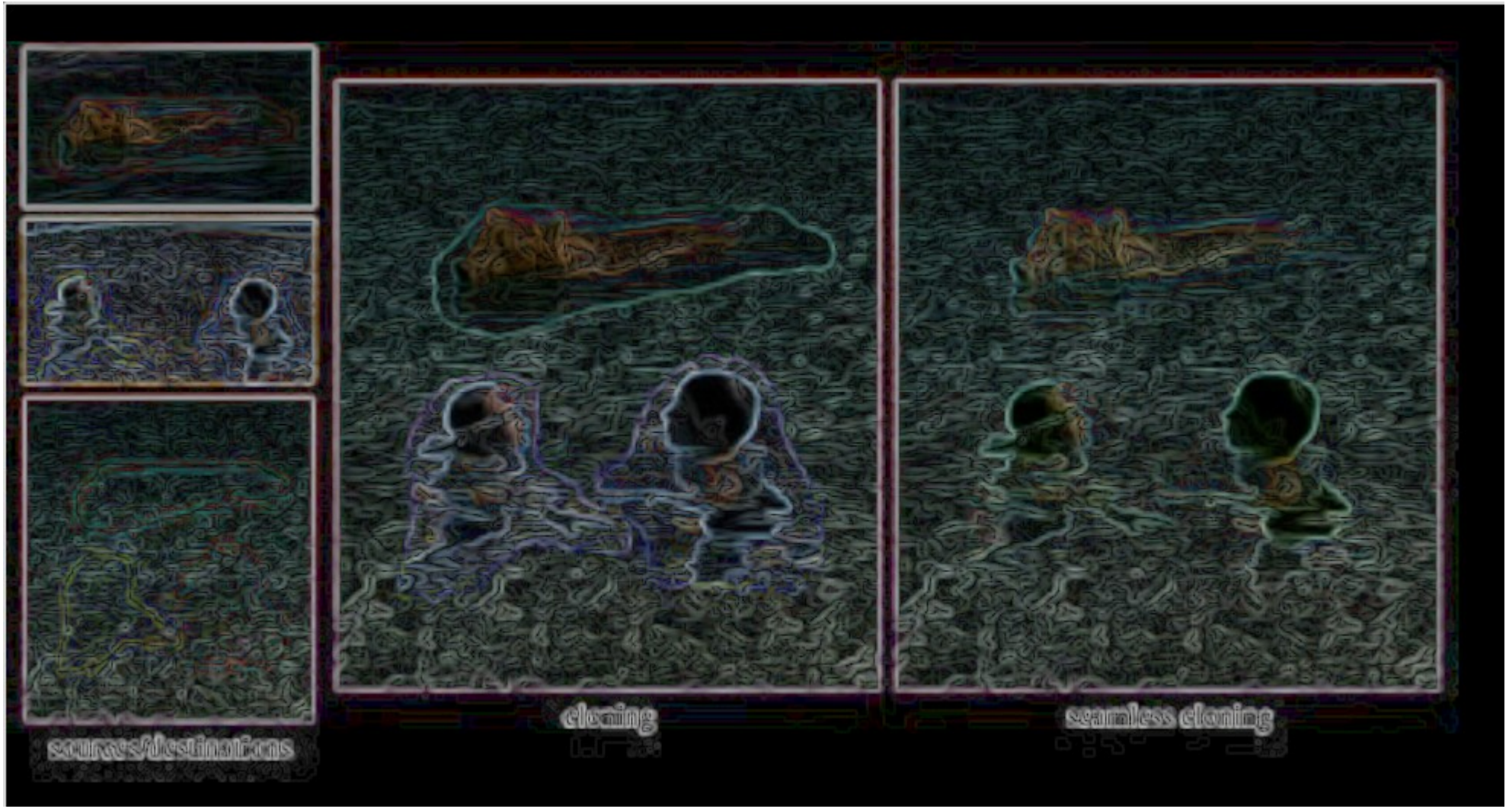




sources/destinations

cloning

seamless cloning


sources/destinations
cloning
seamless cloning