

# Rapport de projet de département

## Redimensionnement intelligent d'images

Lucie Fayette ; Alexis Gadonneix ; Alice Gribonval ; Paul-Eloi Mangion ; Louis Reine

27 décembre 2020

### Table des matières

<b>1 Objectif du projet, présentation du SeamCarving pour le redimensionnement d'images</b>	<b>2</b>
1.1 SeamCarving pour la réduction d'image . . . . .	2
1.2 SeamCarving pour l agrandissement d'image . . . . .	5
<b>2 Implémentation de la méthode</b>	<b>5</b>
2.1 Réduire une image selon une direction . . . . .	5
2.2 Agrandir une image selon une direction . . . . .	7
2.3 Extension : supprimer ou protéger les régions d'une image . . . . .	7



(a) Image initiale



(b) Réduction linéaire



(c) Suppression de colonnes



(d) *SeamCarving*

FIGURE 1 – Réduction d'une image par différentes méthodes

L'omniprésence des contenus visuels sur les supports en ligne fait de leur adaptation à diverses tailles d'écran un enjeu capital du développement web. Il s'agit en effet d'utiliser une même image

mais en adaptant sa taille à l'espace d'affichage disponible, sans pour autant dénaturer son contenu. Par conséquent, des redimensionnements linéaires naïfs ne peuvent suffire. En retirant à l'image 1a des colonnes réparties linéairement sur la largeur de l'image, on obtient la Figure 1b. La Figure 1d a elle été obtenue en implémentant la méthode *SeamCarving* décrite dans [1].

## 1 Objectif du projet, présentation du SeamCarving pour le redimensionnement d'images

Le but de ce travail est d'apprendre à programmer en groupe autour d'un projet commun. Pour cela il nous a été proposé d'implémenter une méthode de redimensionnement intelligent d'images appelée *SeamCarving*. L'objectif est de modifier la taille d'une image, et en particulier son ratio, sans dénaturer ses éléments de plus grande importance, et en n'affectant par conséquent que ceux qui contiennent une information moindre.

### 1.1 SeamCarving pour la réduction d'image

Pour redimensionner des images sans les rogner, une méthode très simple est de supprimer des lignes ou des colonnes réparties linéairement sur la totalité de l'image. Cependant nous avons vu sur la Figure 1b que les résultats obtenus ne sont pas toujours satisfaisants. Le rapport des dimensions de l'image réduite n'a pas été conservé, et la réduction de la largeur de l'image ne tient pas compte des informations contenues dans l'image, ce qui fait que l'image nous apparaît clairement déformée.

Dans le cadre de la méthode *SeamCarving*, on suppose que les zones importantes d'une image sont celles qui présentent un fort gradient. Cette heuristique est assez intuitive, car les zones "homogènes" de l'image sont souvent des zones où il y a peu d'information.

Dès lors, on peut penser à retirer des lignes ou des colonnes réparties non plus linéairement sur l'image, mais là où le gradient est le plus faible. La Figure 1c a été obtenue en retirant de l'image les colonnes pour lesquelles la somme des gradients est la plus faible. Cependant ici encore, le résultat n'est pas optimal. La méthode *SeamCarving*, dont le résultat est illustré sur la Figure 1d permet d'améliorer significativement le résultat.

**Principe de la méthode** On cherche à supprimer les zones de l'image dans lesquelles l'information est la moins importante. Pour améliorer le résultat par rapport à une réduction classique d'image, et afin de limiter les artefacts, on ne supprime pas nécessairement des lignes ou des colonnes entières de l'image, mais des chemins de pixels horizontaux ou verticaux. Ainsi on pourra modifier significativement l'image en contournant ses zones importantes qui contiennent de l'information, et conserver un aspect naturel.

#### Définitions

- On appelle *chemin* (ou *seam* en anglais) vertical (resp. horizontal) *continu de pixels* d'abscisse  $x(i)$  (resp. d'ordonnée  $y(j)$ ) et d'ordonnée  $i$  (resp. d'abscisse  $j$ ) un ensemble

$$\{s_i^x\}_{i=1}^h = \{(x(i), i)\}_{i=1}^h$$

(resp.  $\{s_j^y\}_{j=1}^w = \{(y(j), j)\}_{j=1}^w$ ) de points tels que

$$\forall i, |x(i) - x(i-1)| \leq 1$$

(resp.  $\forall j, |y(j) - y(j - 1)| \leq 1$ ).

La Figure 2 illustre un chemin continu de pixels, par opposition à des chemins discontinus ou des ensembles de pixels qui ne sont pas des chemins.

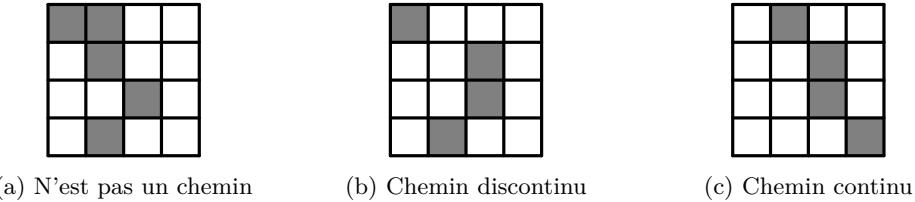


FIGURE 2 – Illustration de la définition d'un chemin vertical

- On appelle, par abus de langage, *gradient*  $V$  d'une image  $I$  le tableau pour lequel la valeur en  $(x, y)$  est la norme  $L^1$  du gradient, calculée de la façon suivante :

$$V(x, y) = \left| \frac{\partial I}{\partial x}(x, y) \right| + \left| \frac{\partial I}{\partial y}(x, y) \right| \quad (1)$$

Dans le cadre d'images, on approche en fait les dérivées continues par des différences finies. L'article [2] présente différentes normes pouvant s'avérer intéressantes dans la définition des contours dans une image.

- On appelle *poids* d'un chemin vertical (ou horizontal) la valeur de la somme des pixels du chemin dans l'image du gradient :

$$W(s^x) = \sum_{i=1}^h V(x(i), i)$$

La Figure 3 représente le gradient de l'image considérée. Les zones claires sont les zones dans lesquelles la norme du gradient est élevée, les zones sombres celles où la norme est faible.



FIGURE 3 – Image et son gradient

**Mise en place pratique** À titre d'exemple, nous présentons ici la réduction d'une image dans la direction horizontale. Une réduction dans la direction verticale peut être obtenue directement en appliquant ce qui suit à la transposée de l'image.

On cherche le chemin continu de pixels le "moins important" pour le retirer, et ainsi diminuer la largeur de l'image d'un pixel.

Pour construire ce chemin, on utilise le gradient  $V$  de l'image. On cherche donc le chemin vertical de poids minimal  $s_{min}$  parmi l'ensemble des chemins tel que :

$$W(s_{min}) = \min_{\forall s \in S} W(s)$$

Une fois ce chemin trouvé, on le retire de l'image, qui est donc effectivement réduite d'un pixel de large. La Figure 4b illustre le principe de la méthode.

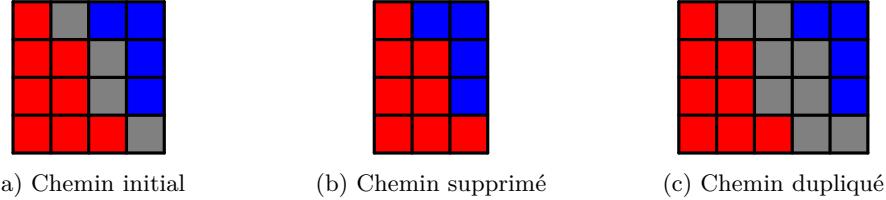


FIGURE 4 – Redimensionnement pour un chemin vertical

Pour réduire l'image de  $k$  pixels, il suffit de répéter les opérations précédentes  $k$  fois.

**Méthode de calcul du gradient** Nous avons comparé les résultats obtenus en utilisant le gradient défini par l'Équation 1, et le *gradient 1D*, calculé en tout point  $(x, y)$  en ne considérant la variation de l'image que dans la direction où l'on réduit l'image. Pour une réduction horizontale, on a alors  $V_h(x, y) = |\frac{\partial I}{\partial x}(x, y)|$ . La Figure 5 illustre la différence.



FIGURE 5 – Image réduite obtenue par différentes méthodes de calcul du gradient

L'image obtenue en utilisant le gradient *1D* est moins dénaturée que celle obtenue par le gradient *2D*. En particulier, le toit du bâtiment garde, dans le second cas, une forme cohérente qu'il n'a pas avec l'utilisation du gradient *2D*, et le lampadaire ne semble pas tordu. L'utilisation du gradient *1D* permet donc de limiter les artefacts, c'est celui que nous utiliserons par la suite.

## 1.2 SeamCarving pour l agrandissement d image

En utilisant les objets définis dans la partie précédente, il est possible d'adapter la méthode afin d'agrandir des images de manière agréable.

**Principe** Pour agrandir l'image il suffit de dupliquer les chemins de poids minimum. Ainsi on agrandira les zones relativement homogènes, tout en conservant sans les déformer les zones avec beaucoup de détails.

**Mise en oeuvre pratique** Il faut prendre garde à ne pas dupliquer toujours le même chemin de poids minimal, ce qui causerait une "colonne rayée" à l'emplacement dupliqué [1]. Pour éviter cet écueil, il ne faut pas que les chemins repassent au même endroit. Pour ce faire, on commence par déterminer les  $k$  premiers chemins qui seraient utilisés pour réduire l'image de  $k$  pixels en largeur, puis on les duplique.

## 2 Implémentation de la méthode

Le module Image de la bibliothèque Imagine [3] est utilisé pour manipuler et afficher facilement les images.

### 2.1 Réduire une image selon une direction

Pour réduire la largeur d'une image avec la méthode *SeamCarving*, la difficulté réside dans la recherche du chemin de poids minimal. Pour trouver un tel chemin, on applique une méthode de programmation dynamique, en trois étapes.

1. On construit une *matrice de poids* à partir de la carte des gradients comme illustré sur la Figure 6, en attribuant à chaque point de l'image un poids égal à la somme de son gradient et du minimum des poids de ses trois voisins du dessus.

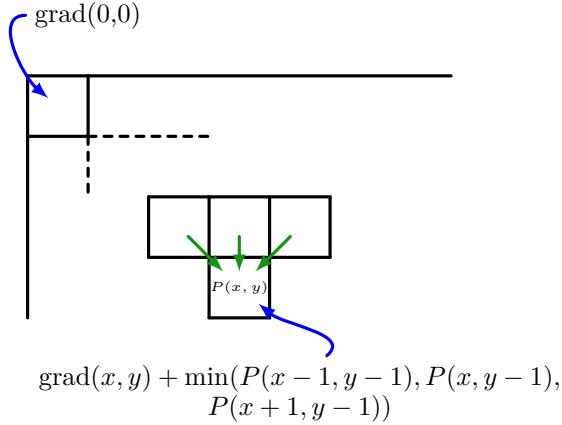


FIGURE 6 – Attribution des poids

2. On cherche le point de départ du chemin de poids minimal en cherchant le pixel de poids minimal sur la dernière ligne.

3. On trouve le chemin de poids minimal en partant de ce pixel et en remontant la matrice de poids de proche en proche comme sur la Figure 7.

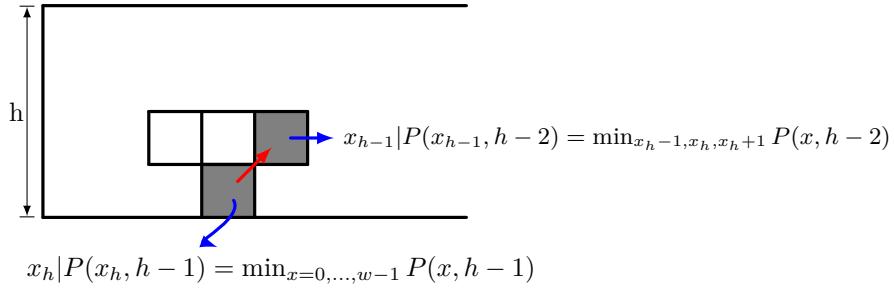


FIGURE 7 – Construction du chemin de poids minimal

La Figure 8 illustre la réduction horizontale d'une image par la méthode de *SeamCarving* en comparaison d'une méthode classique. La Figure 8b montre le chemin, ici horizontal, de poids minimal qui sera retiré pour réduire la hauteur de l'image. On constate que, conformément à notre heuristique, il évite les zones de l'image contenant une information importante.



FIGURE 8 – Réduction horizontale

## 2.2 Agrandir une image selon une direction

Nous traiterons ici l'exemple de l'élargissement qui n'est pas détaillé dans l'article [1].

Pour agrandir la largeur d'une image donnée de  $k$  pixels on procède de la façon suivante :

1. On rétrécit l'image initiale de  $k$  pixels en appliquant  $k$  fois l'algorithme de rétrécissement par *SeamCarving*.
2. On reconstruit l'image élargie à partir des chemins que l'on a retirés en les dupliquant.

La méthode de stockage des chemins est particulièrement délicate, étant donné qu'une fois retiré un chemin, les coordonnées des pixels dans l'image réduite sont différentes des coordonnées dans l'image de départ.

Pour chaque ligne de l'image, nous avons stocké dans une liste ordonnée les coordonnées des pixels retirés en utilisant la méthode suivante pour se replacer dans le système de coordonnées de l'image initiale :

```
Data : list x of size n, int c
Result : list x of size n+1
Initialisation;
s ← 0;
while s<n and c>=x[s] do
|   c ← c+1;
|   s ← s+1
end
Insert c between indexes s-1 and s;
```

**Algorithme 1 :** Détermination des coordonnées initiales d'un pixel sur une ligne

$x$  représente la liste ordonnée contenant les abscisses de sorte que  $x[s] = \bar{x}_s$ , et  $c$  est l'abscisse que l'on cherche à déterminer.

Il s'agit ensuite de construire l'image finale. Pour cela on copie l'image initiale dans une nouvelle image élargie de  $k$  colonnes vides à droite. Puis on construit par la droite pour chaque ligne  $i$  suivant la méthode de la Figure 9.

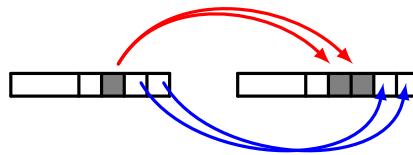


FIGURE 9 – Construction de l'image élargie

## 2.3 Extension : supprimer ou protéger les régions d'une image

Afin de préserver ou au contraire supprimer certaines régions d'une image, on applique un masque sur l'image : les zones à supprimer sont dotées d'un poids très négatif, de façon à être traversées par les chemins, et celles à préserver d'un poids très positif pour que les chemins n'y passent pas.

La Figure 10 montre que les troncs d'arbres sont réduits de façon assez importante s'ils ne sont pas protégés. La protection de l'image permet de résoudre ce problème de manière efficace.

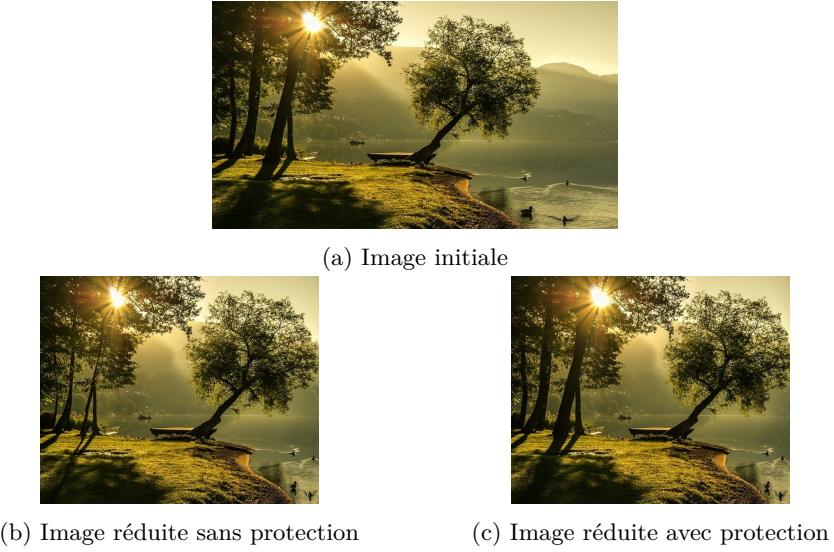


FIGURE 10 – Réduction d'image protégée

Pour effacer une zone d'une image sans en modifier les dimensions, on applique sur cette région des poids négatifs, puis on réécrit l'image, avant d'agrandir à nouveau l'image obtenue à la taille de l'image originale. Cette méthode permet ainsi de supprimer le vacancier dans la Figure 11.

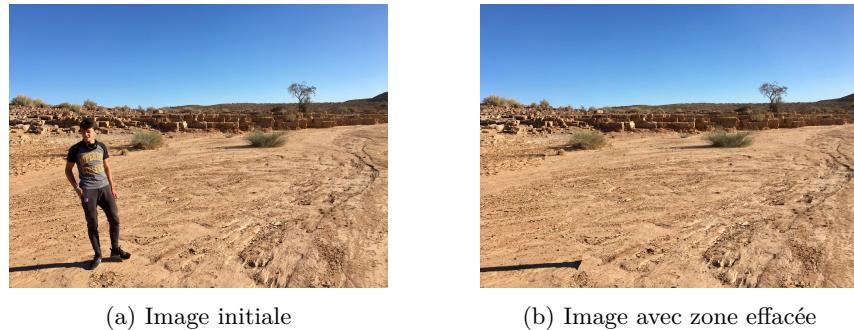


FIGURE 11 – Effacer une région de l'image

## Références

- [1] Ariel Shamir Shai Avidan. Seam carving for content-aware image resizing. *ACM Trans. On Graphics*, 2007.
- [2] Rachid Deriche David Tschumperlé. Restauration d'images vectorielles par edp. 2004.
- [3] Bibliothèque imagine++, version du 17 mars 2020. <http://imagine.enpc.fr/monasse/Imagine++/index.html>.