

ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES



INSTITUT
POLYTECHNIQUE
DE PARIS

PROJET DE PREMIÈRE ANNÉE 2025

REDIMENSIONNEMENT D'IMAGES PAR SEAM-CARVING

RAPPORT FINAL

CHENG LÉO
GARNAUD LÉO
HOSSAINY MALAK
SONDALI SOUFIANE

sous la direction de

M. MONASSE Pascal

IMAGINE/LIGM, École nationale des ponts et chaussées, Université Paris-Est, France

Année universitaire 2024-2025

May 12, 2025

Abstract

This project explores the Seam Carving algorithm as a content-aware image resizing method, offering an alternative to traditional resizing techniques such as scaling or cropping. These conventional methods often neglect image content, leading to visual distortions or loss of important information. Seam Carving addresses this issue by iteratively removing or inserting paths of low-energy pixels, thus preserving the image's salient features. We first implemented the algorithm in Python for grayscale and color images, then extended it to C++ for performance optimization. Several energy functions were tested and dynamic programming was used to determine optimal seams. In addition to resizing, we investigated applications such as object removal and image enlargement. We also worked on a user-friendly interface to make object selection possible directly on the image. Challenges included adapting the theoretical model to discrete digital images and managing computational constraints. Ultimately, Seam Carving demonstrated its potential for adaptive image resizing with visually coherent results, although limitations remain in terms of content preservation and processing time.

KEYWORDS Image - Resizing - Seam - Energy function - Pixels - Dynamic Programming.

Contents

1	Introduction.	3
1.1	Contexte et enjeux du redimensionnement d'images.	3
1.2	Problématique générale : limites des méthodes traditionnelles vs. besoins actuels.	3
1.3	Objectifs du projet et questions de recherche.	3
2	État de l'Art.	3
2.1	Méthodes traditionnelles.	3
2.1.1	Redimensionnement global.	3
2.1.2	Recadrage (<i>cropping</i>).	4
2.2	Seam Carving : Principes fondamentaux.	4
2.2.1	Notion de couture et de fonction d'énergie.	4
2.2.2	Algorithme de sélection optimale (programmation dynamique).	5
3	Implémentation et méthodologie.	5
3.1	Fonction d'énergie.	5
3.1.1	Norme L^1 du gradient de I	5
3.1.2	Autres fonctions d'énergie.	6
3.2	Choix techniques pour l'implémentation de l'algorithme.	7
3.2.1	Structures de données pour la manipulation d'images.	7
3.2.2	Méthodes d'implémentation.	8
4	Résultats et Expérimentations.	8
4.1	Redimensionnement des images.	8
4.1.1	Réduction d'images.	8
4.1.2	Agrandissement d'images.	8
4.1.3	Amplification d'images.	9
4.2	Applications avancées.	10
4.2.1	Reciblage (<i>Retargeting</i>).	10
4.2.2	Reconstruction de POISSON.	11
4.2.3	Suppression d'objets guidée par l'utilisateur.	12
4.2.4	Gestion des multi-images	13
4.3	Analyse des performances.	14
4.3.1	Temps de calcul : Méthode classique vs. méthode des pointeurs (cf. section 3.2.2).	14
4.3.2	Impact du choix de la fonction d'énergie (cf. section 3.1) sur le résultat de la réduction d'image.	14
5	Conclusion.	16
5.1	Défis rencontrés.	16
5.1.1	Implémentation technique.	16
5.1.2	Limites de la méthode.	16
5.2	Synthèse des contributions.	17
6	Annexe	18

1 Introduction.

1.1 Contexte et enjeux du redimensionnement d'images.

Avec l'essor du numérique, le traitement d'images s'impose comme un domaine central en informatique, que ce soit dans la photographie, la vidéo, le web ou les applications mobiles. L'un des problèmes les plus récurrents est celui du redimensionnement d'images, qui intervient lors de l'adaptation d'un contenu visuel à des supports aux formats variés, sans en altérer la lisibilité ni la qualité perçue. Dans ce contexte, les méthodes traditionnelles comme la mise à l'échelle uniforme ou le recadrage s'avèrent souvent insuffisantes, car elles peuvent altérer ou supprimer des éléments essentiels à la compréhension de l'image.

C'est pour répondre à ce besoin qu'est apparue la méthode du *Seam Carving*, introduite en 2007 par AVIDAN et SHAMIR dans [1]. Cette approche innovante repose sur la suppression ou l'insertion de chemins de faible énergie – appelés coutures – dans une image, permettant ainsi un redimensionnement structurellement conscient du contenu. Elle préserve les zones importantes tout en modifiant la taille de l'image, de manière souvent imperceptible à l'œil.

1.2 Problématique générale : limites des méthodes traditionnelles vs. besoins actuels.

Les méthodes classiques de redimensionnement ne prennent pas en compte la signification sémantique des zones de l'image : une réduction uniforme compresse tous les pixels sans distinction, tandis que le recadrage supprime arbitrairement des régions périphériques. Or, dans un contexte où les images sont utilisées à des fins de communication visuelle, ces pertes peuvent s'avérer problématiques : suppression d'un visage, déformation d'un objet, perte d'un point focal ...

Le *Seam Carving*, moyennant une fonction d'énergie, identifie les zones visuellement moins importantes, et oriente ainsi le redimensionnement pour qu'il évite les zones critiques. Cela répond aux exigences actuelles de flexibilité tout en respectant le contenu sémantique, ce qui en fait une méthode prometteuse tant pour la réduction que pour l'agrandissement ou la modification ciblée d'images.

1.3 Objectifs du projet et questions de recherche.

Dans le cadre de ce projet, nous nous sommes fixés plusieurs objectifs :

- Comprendre les fondements théoriques de la méthode du *Seam Carving*, en particulier le rôle de la fonction d'énergie et l'algorithme de programmation dynamique dans la recherche de coutures optimales.
- Implémenter une solution complète de redimensionnement basée sur cette méthode, en développant notre propre pipeline en C++.
- Évaluer les performances de notre solution en termes de qualité visuelle, robustesse et efficacité, notamment sur des images de complexité variable.
- Explorer des extensions possibles, telles que l'agrandissement, la suppression d'objets, ou l'adaptation à des images couleurs.

Ce projet adopte une démarche d'expérimentation approfondie, mêlant théorie algorithmique, implémentation pratique et analyse critique des résultats obtenus.

Dans l'ensemble de ce rapport, nous considérerons une image discrétisée de dimensions (n, m) en pixels (*ie.* un tableau) où n correspond au nombre de lignes (direction verticale) et m au nombre de colonnes (direction horizontale). La fonction I désigne l'intensité lumineuse (niveau de gris ou RGB) de l'image.

2 État de l'Art.

2.1 Méthodes traditionnelles.

2.1.1 Redimensionnement global.

Le redimensionnement global d'image consiste à appliquer une transformation d'échelle différente selon les dimensions de l'image d'origine. Autrement dit, on applique un facteur d'échelle α sur les lignes et un facteur β sur les colonnes, ce qui permet d'obtenir une image de taille $(\alpha n, \beta m)$. Cette opération nécessite une interpolation indépendante sur

les lignes et les colonnes, et peut être utilisée pour étirer, comprimer ou déformer l'image de manière non uniforme. Cette méthode est simple et rapide, mais elle présente deux limites majeures :

- Elle ne tient pas compte du contenu visuel : les objets importants peuvent être déformés.
- Elle introduit souvent du flou et perd des détails fins dans les zones critiques.

2.1.2 Recadrage (*cropping*).

Le recadrage consiste à découper une sous-région rectangulaire de l'image. Si l'on cherche à réduire la largeur ou la hauteur, on choisit simplement une fenêtre de taille inférieure à l'image initiale.

Cette approche permet de conserver la résolution et les proportions locales, mais elle peut supprimer des éléments essentiels de la scène, notamment si les objets d'intérêt ne sont pas centrés.

- Elle ne tient pas compte du contenu visuel : les objets importants peuvent être déformés.
- Elle introduit souvent du flou et perd des détails fins dans les zones critiques.

La figure 1 illustre les effets du redimensionnement appliqué à l'image [4] à l'aide de méthodes classiques, telles que le recadrage et le redimensionnement global.



(a) Image initiale sur laquelle sera appliquée une réduction de 200 pixels dans la direction horizontale.



(b) Résultat de la réduction par recadrage.



(c) Résultat de la réduction par redimensionnement.

Figure 1: Comparaison des méthodes de réduction horizontale appliquées à l'image [4]. (a) L'image originale, sur laquelle une réduction de 200 pixels est envisagée. (b) Résultat d'un recadrage simple, consistant à supprimer directement une bande verticale. (c) Résultat d'un redimensionnement classique, qui conserve toute l'image mais induit une déformation globale.

2.2 Seam Carving : Principes fondamentaux.

2.2.1 Notion de couture et de fonction d'énergie.

Le *Seam Carving* constitue une méthode de redimensionnement d'images dite orientée contenu, dont le principe fondamental repose sur la suppression ou l'insertion de coutures (*seams*) déterminées à l'aide d'une fonction d'énergie évaluant l'importance visuelle locale de chaque pixel.

Définition 1 (Couture). Une couture est un chemin continu connecté de pixels traversant l'image verticalement ou horizontalement, et dont le retrait affecte le moins possible les zones visuellement importantes.

Définition 2. Une **couture verticale** est une suite de pixels définie par :

$$(s_i) := \{(i, c(i)) \mid 0 \leq i < n\}$$

avec une application $c : \llbracket 0, n-1 \rrbracket \rightarrow \llbracket 0, m-1 \rrbracket$ telle que :

$$|c(i) - c(i-1)| \leq 1 \quad \text{pour tout } i \in \llbracket 1, n-1 \rrbracket \quad (\text{Condition de continuité})$$

De manière analogue, une **couture horizontale** est une suite de pixels définie par :

$$(s_j) := \{(r(j), j) \mid 0 \leq j < m\}$$

où $r : \llbracket 0, m-1 \rrbracket \rightarrow \llbracket 0, n-1 \rrbracket$ est une application qui vérifie :

$$|r(j) - r(j-1)| \leq 1 \quad \text{pour tout } j \in \llbracket 1, m-1 \rrbracket \quad (\text{Condition de continuité})$$

Définition 3 (Fonction d'énergie). Une fonction d'énergie est une application :

$$E : \llbracket 0, n-1 \rrbracket \times \llbracket 0, m-1 \rrbracket \rightarrow \mathbb{R}^+$$

mesurant l'importance de chaque pixel dans l'image. Les fonctions d'énergie les plus utilisées sont la norme L^1 du gradient de l'intensité lumineuse I de l'image, la norme L^2 de ce gradient, ou encore l'entropie de l'image. L'objectif est alors de supprimer les coutures de plus faible énergie cumulée.

2.2.2 Algorithme de sélection optimale (programmation dynamique).

La recherche de la couture d'énergie minimale repose sur une approche de programmation dynamique.

Concernant le problème de la couture verticale optimale, le but est de trouver le chemin minimisant la quantité :

$$\sum_{i=0}^{n-1} E(i, c(i))$$

Pour cela, on construit une matrice de coût cumulé $M \in \mathbb{R}^{n \times m}$, dont la récurrence est donnée par :

$$\forall j \in \llbracket 0, m-1 \rrbracket, \begin{cases} M(0, j) = E(0, j) \\ M(i, j) = E(i, j) + \min [M(i-1, j-1), M(i-1, j), M(i-1, j+1)] \end{cases} \quad , \quad \forall i \in \llbracket 1, n-1 \rrbracket$$

avec la convention $M(i-1, -1) = M(i-1, m) = +\infty$.

Une fois la matrice M calculée, la couture optimale correspond à la trajectoire partant du pixel de plus faible énergie dans la dernière ligne et remontant en suivant, à chaque ligne, le minimum local parmi les trois voisins de la ligne du dessus (si ceux-ci existent).

L'algorithme 1 représente un pseudo-code de l'implémentation de la couture verticale optimale.

De même, un raisonnement similaire s'applique pour obtenir la couture horizontale optimale : l'algorithme 2 représente un pseudo-code de l'implémentation de la couture horizontale optimale.

3 Implémentation et méthodologie.

3.1 Fonction d'énergie.

3.1.1 Norme L^1 du gradient de I .

Comme indiqué dans la section 2.2.1, la fonction d'énergie permet de quantifier l'importance des pixels d'une image afin de guider le choix de ceux à supprimer. Dans le cadre de ce projet, la fonction d'énergie principale que nous avons utilisée est la norme L^1 du gradient de l'intensité lumineuse I , définie comme suit :

$$\forall i, j \in \llbracket 0, n-1 \rrbracket \times \llbracket 0, m-1 \rrbracket, \quad E_1(i, j) = \left| \frac{\partial I}{\partial x}(i, j) \right| + \left| \frac{\partial I}{\partial y}(i, j) \right|$$

Étant donné que l'image est, par nature, discrétisée en pixels, le gradient de l'intensité lumineuse I est implémenté en approchant les dérivées partielles par la différence centrée des intensités des deux voisins adjacents selon la direction concernée. Formellement :

$$\forall i, j \in \llbracket 0, n-1 \rrbracket \times \llbracket 0, m-1 \rrbracket, \quad \frac{\partial I}{\partial x}(i, j) \simeq \frac{I(i, j+1) - I(i, j-1)}{2} \quad \text{et} \quad \frac{\partial I}{\partial y}(i, j) \simeq \frac{I(i+1, j) - I(i-1, j)}{2}$$

lorsque le pixel (i, j) possède deux voisins adjacents dans la direction concernée.

Pour les pixels situés en bordure de l'image, qui ne possèdent qu'un seul voisin dans au moins une direction, la valeur de l'intensité lumineuse du voisin manquant est remplacée par celle du pixel lui-même.

Pour une image en couleur composée de trois canaux d'intensité (Rouge, Vert, Bleu), la fonction d'énergie globale est définie comme la somme des fonctions d'énergie calculées séparément sur chacun des canaux R, G et B.

3.1.2 Autres fonctions d'énergie.

À des fins de comparaison et d'analyse de performance (cf. section 4.3.2), nous avons également pris en considération trois fonctions d'énergie alternatives, dans le but d'évaluer leur comportement respectif lors du redimensionnement d'images.

Norme L^2 du gradient de I . Cette fonction d'énergie correspond à la norme euclidienne (ou norme L^2) du gradient de l'intensité lumineuse de l'image. Elle est définie, pour tout pixel (i, j) , par :

$$E_2(i, j) = \sqrt{\left(\frac{\partial I}{\partial x}(i, j)\right)^2 + \left(\frac{\partial I}{\partial y}(i, j)\right)^2}$$

À l'instar de la norme L^1 , cette fonction repose sur les mêmes approximations numériques pour le calcul des dérivées partielles, ainsi que sur le même traitement appliqué aux pixels situés en bordure de l'image.

Fonction d'énergie HoG. La fonction d'énergie HoG (*Histogram of Oriented Gradients*) est utilisée pour quantifier l'importance des pixels en fonction des orientations et des magnitudes des gradients dans une fenêtre locale. Pour chaque pixel (i, j) , un histogramme des orientations des gradients est construit autour de la position considérée, avec une taille de fenêtre définie (ici, 10×10 pixels). L'orientation des gradients est déterminée par l'argument du vecteur gradient du pixel.

Pour chaque pixel dans la fenêtre autour de (i, j) , l'orientation du gradient est calculée et utilisée pour déterminer l'indice du bin de l'histogramme. Chaque bin représente une plage d'angles. Dans notre cas, le nombre total de bins est fixé à 9, représentant les orientations discrétisées sur 360° . Ensuite, la norme euclidienne du gradient (appelée magnitude) est placée dans le bin correspondant à cette orientation, de manière à accumuler les magnitudes des pixels de la fenêtre dans l'histogramme.

La fonction d'énergie HoG pour un pixel donné (i, j) est définie ainsi comme le rapport entre la norme L^1 du gradient à ce pixel et la valeur maximale de l'histogramme des gradients dans la fenêtre. Formellement, l'énergie pour chaque pixel est donnée par :

$$E_{\text{HoG}}(i, j) = \frac{E_1(i, j)}{\max[\text{HoG}(I(i, j))]}$$

où $\text{HoG}(I(i, j))$ représente l'histogramme des gradients orientés calculé pour chaque pixel (i, j) dans la fenêtre autour de ce dernier. L'algorithme 3 représente un pseudocode relatif à l'implémentation de cette fonction d'énergie.

Pour les pixels situés en bordure de l'image et pour lesquels une fenêtre 10×10 ne peut être appliquée sans déborder des limites de l'image, nous sélectionnons, dans la mesure du possible, uniquement les pixels qui se trouvent à l'intérieur des dimensions de l'image, en excluant ceux situés en dehors de celle-ci.

Entropie du gradient de I . La fonction d'énergie d'entropie est utilisée pour mesurer l'incertitude ou le désordre dans la distribution des gradients dans une fenêtre locale autour de chaque pixel. À chaque pixel (i, j) , un histogramme des orientations de gradient est construit en utilisant une fenêtre de taille 10×10 pixels. Ensuite, l'orientation du gradient est utilisée pour déterminer l'indice du bin. Ainsi, les magnitudes sont accumulées dans les bins correspondants de l'histogramme, qui est ensuite normalisé pour que la somme des valeurs de tous les bins soit égale à 1.

L'entropie est ensuite calculée sur la base de l'histogramme normalisé des orientations de gradient, ce qui mesure la diversité ou la distribution des orientations dans la fenêtre locale. L'entropie H d'un histogramme est donnée par :

$$H(i, j) = - \sum_{k=0}^{N_b-1} p_k \log(p_k)$$

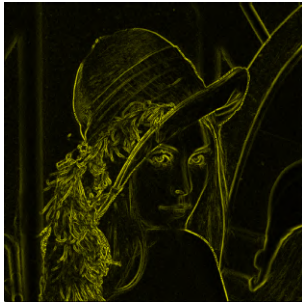
où p_k représente la probabilité d'occurrence du gradient dans le bin k , et N_b est le nombre total de bins de l'histogramme. Cette mesure est ajoutée à la norme L^1 du gradient $E_1(i, j)$ pour chaque pixel, ce qui aboutit à la fonction d'énergie totale définie par :

$$E_{\text{entropie}}(i, j) = E_1(i, j) + H(i, j)$$

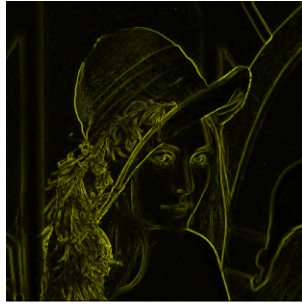
L'algorithme 4 représente un pseudocode relatif à l'implémentation de cette fonction d'énergie.

Le même traitement réservé aux pixels de bordure dans le calcul de l'énergie HoG est également appliqué lors du calcul de l'entropie pour ces pixels.

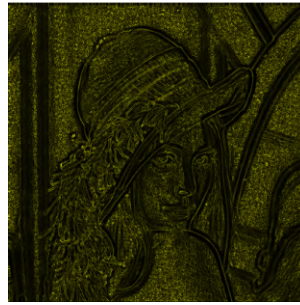
La figure 2 représente une comparaison visuelle des différentes fonctions d'énergie appliquées sur l'image [4].



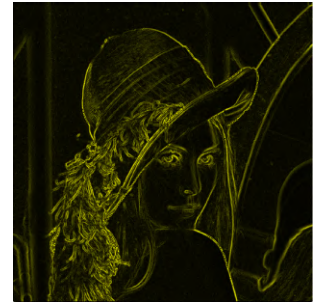
(a) Affichage de la fonction d'énergie E_1 sur l'image [4].



(b) Affichage de la fonction d'énergie E_2 sur l'image [4].



(c) Affichage de la fonction d'énergie E_{HoG} sur l'image [4].



(d) Affichage de la fonction d'énergie E_{entropie} sur l'image [4].

Figure 2: Comparaison visuelle des différentes fonctions d'énergie utilisées pour le redimensionnement de l'image [4].

3.2 Choix techniques pour l'implémentation de l'algorithme.

3.2.1 Structures de données pour la manipulation d'images.

On utilise essentiellement deux classes d'objets, les autres étant présents uniquement dans une des deux implémentations.

La première classe commune est la classe **Matrice/Energie** qui permet d'implémenter les matrices à 2 dimensions. Elle possède pour attribut un tableau pour stocker les valeurs et des entiers pour les dimensions. Il est possible d'accéder à chacune des valeurs de la matrice ainsi que ses dimensions. Seules les valeurs de coordonnées valides sont accessibles et modifiables. La valeur de coordonnées (i, j) est stockée à l'emplacement $i + w * j$ où w est la largeur de la matrice (on notera que $w = m$).

La seconde classe est la classe **Image** qui implémente une représentation des images. Comme la classe **Matrice**, elle possède ses dimensions en attribut mais le tableau est trois fois plus grand pour stocker les trois couleurs primaires: rouge, vert et bleu. Ainsi, on stocke les pixels dans le même ordre que les matrices, mais à chaque pixel correspond trois valeurs stockées successivement, respectivement les intensités rouge, vert puis bleu du pixel de coordonnées (i, j) . Cette seconde classe implémente en outre des méthodes utilisant les fonctions de la bibliothèque **Imagine++**, à savoir le chargement et l'affichage d'images.

3.2.2 Méthodes d'implémentation.

Méthode classique. Dans cette méthode, on utilise seulement les deux classes présentées ci-dessus. Ainsi, pour avoir accès au voisin de droite du pixel (i, j) dans la nouvelle image, on regarde si le pixel $(i + 1, j)$ fait partie ou non d'une couture. Pour cela, lors de l'actualisation du tableau d'énergie, on impose que $E < 0$ pour tout pixel appartenant à une couture. Si ce n'est pas le cas, alors le pixel $(i + 1, j)$ est le voisin de droite du pixel (i, j) et sinon, on recommence avec le pixel $(i + 2, j)$ et ainsi de suite, jusqu'à ce qu'on trouve un pixel qui n'appartient pas à une couture. On fait de même avec les autres voisins.

Méthode pointeurs. Dans la méthode pointeur, on définit une nouvelle classe de données additionnelle pour l'image réduite, et ainsi, accéder en temps constant aux pixels voisins afin de calculer le nouveau gradient du pixel.

Cette nouvelle classe s'appuie sur une brique élémentaire **Maillon** qui contient les coordonnées du pixel d'origine, des pointeurs vers ses nouveaux voisins et un booléen indiquant si ce **Maillon** fait partie de l'image réduite.

Ces différents **Maillons** sont stockés dans une matrice 2D que l'on nomme **MaillonTab** pour faciliter l'accès et la modification à un **Maillon** de coordonnées (i, j) . Ainsi, au moment de la mise à jour du tableau des énergies suite au retrait d'une couture, on met à jour les **Maillons** pour qu'ils pointent vers leurs nouveaux voisins, puis les nouveaux gradients d'énergies sont calculés en temps constant à l'aide de ces pointeurs.

4 Résultats et Expérimentations.

4.1 Redimensionnement des images.

Dans l'ensemble de la section 4.1, nous nous appuyons sur une image (figure 3) de dimensions initiales 411×279 pixels (largeur \times hauteur), représentant un fort [5]. Cette image servira de cas d'étude pour illustrer et analyser les différentes méthodes de redimensionnement abordées.



Figure 3: Image originale utilisée pour les expérimentations de redimensionnement - Fort [5].

4.1.1 Réduction d'images.

L'image de référence (Figure 3) est réduite de 200 pixels selon la direction horizontale. La figure 4 illustre le résultat obtenu à l'issue de l'algorithme. On observe que la méthode de *Seam Carving* s'avère efficace, parvenant à supprimer en priorité les zones de faible contraste, telles que le ciel ou le pré, tout en préservant visuellement les structures saillantes de l'image.

4.1.2 Agrandissement d'images.

L'agrandissement d'image peut être vu comme l'opération inverse de la réduction.



Figure 4: Résultat de la réduction horizontale de 200 pixels par l'algorithme de *Seam Carving*.

Pour cela, on commence par identifier les k premières coutures à supprimer, puis on les insère à nouveau pour agrandir l'image. Une première approche consiste à dupliquer ces coutures.

Cependant, une amélioration consiste à ne pas simplement les dupliquer : pour chaque couture sélectionnée, on insère une nouvelle couture à proximité, calculée comme la moyenne des pixels de la couture et de ses voisins. Cette méthode permet d'ajouter de l'information dans l'image de manière plus harmonieuse. La figure 5 illustre le résultat obtenu après un agrandissement de 200 pixels appliqué à l'image de référence dans la direction horizontale.



Figure 5: Résultat de l'agrandissement horizontal de 200 pixels par l'algorithme de *Seam Carving*.

4.1.3 Amplification d'images.

Lorsque la méthode *Seam Carving* est appliquée, les zones les moins informatives sont retirées. Ainsi, en redimensionnant l'image, on modifie les proportions des parties les plus saillantes de l'image.

Dans [1], les auteurs suggèrent d'agrandir d'abord l'image, puis de la réduire à sa taille d'origine à l'aide du *Seam Carving*. Dans notre expérimentation, nous avons suivi une approche inverse : nous avons d'abord réduit l'image, puis nous l'avons agrandie du même nombre de coutures. La figure 6 montre le résultat obtenu.



Figure 6: Résultat de l'amplification de l'image par l'algorithme de *Seam Carving*.

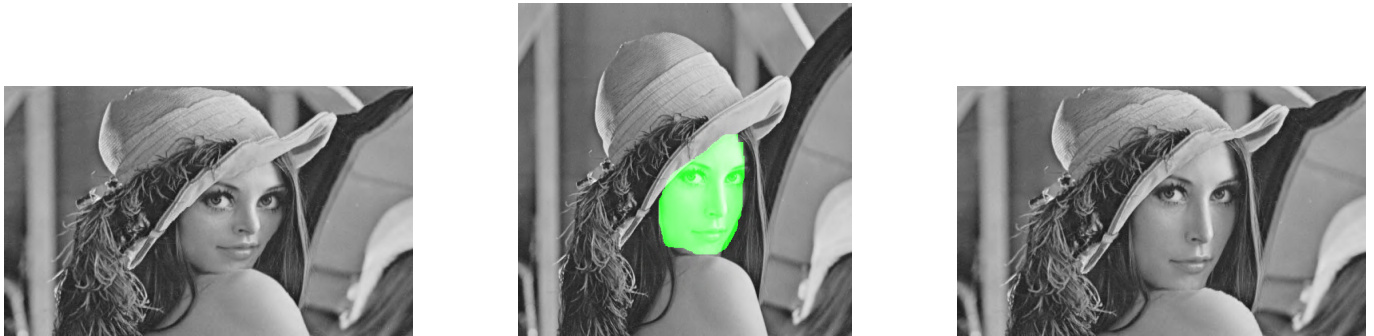
4.2 Applications avancées.

4.2.1 Reciblage (*Retargeting*).

La méthode du *Seam Carving* est une méthode qui permet de retirer la partie de l'image contenant le moins d'information. Cependant, il peut arriver que certaines parties importantes soient enlevées aussi car elles ne contiennent pas assez d'information ou ont un contraste trop faible, notamment les visages humains.

Il est alors possible, après sélection des parties importantes par l'utilisateur, de modifier la matrice d'énergie pour augmenter artificiellement le gradient de sorte que les coutures contournent ces parties à forts gradients. La figure 7 démontre l'intérêt d'un reciblage préalable dans le processus de réduction d'images contenant un visage humain [4], en particulier pour maintenir l'intégrité visuelle de cette région.

Toutefois, si l'utilisateur sélectionne une trop grande partie de l'image ou que cette partie traverse l'image dans toute sa longueur, alors cette partie sera tout de même affectée.



(a) Résultat de la réduction d'une image d'un visage humain [4], obtenue par application de l'algorithme *Seam Carving*.

(b) Reciblage appliqué spécifiquement à la région du visage selon les indications de l'utilisateur.

(c) Résultat de la réduction de l'image par la méthode *Seam Carving*, après reciblage sur la région du visage.

Figure 7: Comparaison des résultats de la réduction d'une image de visage humain [4] via la méthode *Seam Carving*, appliquée avec différentes approches de traitement. (a) Réduction effectuée uniquement par l'algorithme de réduction sur l'image d'origine. (b) Définition de la zone du visage à recibler durant la réduction de l'image. (c) Réduction réalisée après un reciblage appliqué de manière ciblée sur la zone du visage.

4.2.2 Reconstruction de Poisson.



(a) Image initiale sur laquelle sera appliquée une réduction de 100 pixels en largeur.

(b) Image réduite obtenue par les auteurs de l'article [1] en utilisant l'algorithme de *Seam Carving*.

(c) Réduction obtenue à l'issue de notre implémentation de l'algorithme de *Seam Carving*.

Figure 8: Illustration comparative des effets de différentes implémentations de l'algorithme de *Seam Carving* : (a) image d'origine, soumise à une réduction de 100 pixels sur l'axe horizontal ; (b) résultat obtenu par les auteurs de l'article ; (c) résultat de notre implémentation de l'algorithme.

La reconstruction de POISSON appliquée après réduction par *Seam Carving* devrait réduire les traces indésirables laissées par le redimensionnement de l'image. Cependant, notre implémentation de *Seam Carving* ne présente pas certains artefacts visuels observés dans l'implémentation de l'article [1]. La figure 8 met en évidence cette différence d'implémentation.

Toutefois, nous avons tout de même implémenté la reconstruction de POISSON afin de comparer son apport et son utilité.

La reconstruction de POISSON consiste à reconstituer l'image à partir de la donnée du gradient de l'image et de la valeur de certains pixels comme constante d'intégration. Ici, nous ne conservons que la valeur des pixels au bord de l'image réduite.

Le gradient utilisé peut être soit la moyenne des gradients issus de l'image originale, soit le gradient ayant la plus grande valeur absolue, les deux méthodes de calcul donnant des résultats similaires.

Nous avons choisi de réaliser la résolution de cette équation de POISSON à l'aide d'équations linéaires et la sur-relaxation de la méthode de GAUSS-SEIDEL, comme le décrit l'article [2].

Soit p un pixel de l'image après reconstruction de Poisson que l'on note Ω . On note N_p les pixels voisins de p dans l'image originale et N'_p les pixels voisins de p dans l'image réduite. On note f_p^* la valeur du pixel p dans l'image originale et f_p la valeur du pixel p dans l'image reconstruite. On observe que si p est un pixel au bord de l'image réduite, que l'on notera $\partial\Omega$, alors $f_p = f_p^*$.

On définit le gradient entre deux pixels p et q par :

$$\nu_{pq} = f_p - f_q \quad , \quad \forall q \in N_p \cup N'_p,$$

En discrétisant les équations de Poisson, on obtient les équations linéaires à résoudre :

$$\forall p \in \Omega \setminus \partial\Omega, \quad |N_p|f_p - \sum_{q \in N_p \setminus \partial\Omega} f_q = \sum_{q \in \partial\Omega} f_q^* + \sum_{q \in N_p} \nu_{pq}$$

Selon l'article [3], la résolution par sur-relaxation de la méthode de GAUSS-SEIDEL nécessite de considérer le système d'équations linéaires sous la forme d'une équation matricielle $AX = B$.

On introduit respectivement les matrices diagonale, triangulaire supérieure et triangulaire inférieure D, U, L telles que $A = D + L + U$. On choisit arbitrairement un vecteur X_0 , puis on définit par récurrence :

$$\forall k, \quad (D + \omega L)X_{k+1} = \omega B - (\omega U + (\omega - 1)D)X_k$$

La matrice A étant symétrique, définie positive, la suite (X_k) converge pour $\omega \in]0, 2[$ vers la solution de l'équation matricielle $AX = B$.

Etant donnée le manque de détails d'implémentation dans l'article [1], notre implémentation de la reconstruction de POISSON ne donne que très peu d'amélioration par rapport à l'image obtenue par *Seam Carving*. La figure 9 illustre le résultat de la réduction de l'image d'un cricquet [6], comparant les versions obtenues sans reconstruction et avec reconstruction par la méthode de POISSON.



(a) Résultat de réduction avec uniquement le *Seam Carving*.



(b) Résultat de réduction après la reconstruction de POISSON.

Figure 9: Comparaison des résultats de réduction d'une image de cricquet [6]. (a) Réduction effectuée uniquement par la méthode de *Seam Carving* ; (b) Réduction suivie d'une reconstruction par la méthode de POISSON. On observe que la reconstruction de POISSON n'apporte qu'une amélioration marginale par rapport au résultat initial du *Seam Carving*.

4.2.3 Suppression d'objets guidée par l'utilisateur.

De la même manière que le reciblage (cf. section 4.2.1), il est possible de modifier la matrice d'énergie pour, inversement, diminuer artificiellement le contraste. Ici, on a fait le choix de mettre un gradient négatif pour attirer les coutures sur cette partie de l'image.

Une fois l'objet supprimé, on agrandit l'image pour retrouver la taille d'origine. Le programme choisit de retirer des coutures horizontales ou des coutures verticales en fonction du plus petit diamètre. Ce choix n'est pas toujours optimal, mais c'est une heuristique satisfaisante et rapide. La figure 10a illustre le résultat obtenu après avoir retiré la silhouette humaine initialement présente dans l'image de référence (Figure 3).

Toutefois, il n'est pas toujours possible de satisfaire les demandes de l'utilisateur. Par exemple, si l'utilisateur souhaite enlever le trou d'un donut mais de garder le donut intact, la méthode *Seam Carving* modifiera nécessairement le donut. La figure 10b met en évidence un cas de figure où la tentative de suppression des trois fenêtres de la tour gauche du fort entraîne une altération significative de la structure globale. Cela souligne qu'il n'est pas toujours possible de satisfaire les contraintes visuelles souhaitées par l'utilisateur sans compromettre l'intégrité des objets environnants.



(a) Suppression de la silhouette humaine.



(b) Altération involontaire du reste du fort à cause d'une tentative de suppression des trois fenêtres de la tour gauche.

Figure 10: Résultats contrastés de la méthode de *Seam Carving* appliquée à la suppression d'objets. (a) Suppression d'une silhouette humaine, faiblement saillante, réalisée avec succès sans distorsion perceptible de la scène. (b) Tentative de suppression des trois fenêtres de la tour gauche du fort, entraînant une altération structurelle notable.

4.2.4 Gestion des multi-images

Le calcul d'une image de taille différente à partir d'une image originale par la méthode *Seam Carving* nécessite beaucoup de temps. La multi-image permet de stocker une image sous toutes les tailles pour s'adapter en temps réel au besoin de l'utilisateur sans avoir besoin de recalculer à chaque nouvelle taille.

Calcul d'une multi-image. Le calcul d'une multi-image consiste à numéroter chaque pixel dans le sens vertical et dans le sens horizontal afin de savoir à partir de quelle dimension en largeur ou en longueur ce pixel n'apparaîtra plus. Si de plus, l'utilisateur souhaite agrandir l'image, alors on agrandit l'image en appliquant le procédé décrit plus haut et on numérote les pixels de la même manière.

Première méthode. La première méthode consiste à calculer toutes les coutures de l'image dans un sens et de numéroter chaque pixel de la couture à laquelle il appartient. Ensuite, dans l'autre sens, on numérote par colonne ou ligne d'énergie de la plus faible à la plus élevée, afin de ne retirer qu'un seul pixel par couture dans l'autre sens. En effet, si l'on calcule les coutures dans les deux sens, il est possible qu'une couture dans un sens chevauche une autre couture dans l'autre sens sur plus d'un pixel, ce qui a pour conséquence, lorsqu'on retire ces deux coutures, de perdre la forme rectangulaire de l'image originale.

Seconde méthode. La seconde méthode reprend le procédé décrit dans l'annexe de l'article [1]. Pour cette méthode, on ne considère que des coutures qui sont connectées sur l'image d'origine, c'est-à-dire que l'on ne considère pas toutes les coutures qui pourraient apparaître suite à la suppression d'une couture et au recollement des deux parties de l'image. Pour des coutures qui sont connectées sur l'image d'origine, les seuls cas de chevauchement sur plus d'un pixel sont les chevauchements au niveau des parties diagonales. L'article [1] préconise de construire les coutures par morceaux de deux pixels, en considérant un problème d'affectation par paire de deux rangées et en appliquant la méthode hongroise pour résoudre ce problème.

Nous avons décidé d'utiliser plutôt un algorithme glouton pour construire ces morceaux de deux pixels, étant donné que la méthode *Seam Carving* est gloutonne. Pour simplifier la description de notre algorithme, on souhaite construire des coutures horizontales et on considère que la matrice des énergies a déjà été construite.

Soient deux colonnes i et $i + 1$ dans une matrice. La colonne $i + 1$ contient les énergies des pixels de cette colonne et la colonne i contient la somme des énergies des coutures construites jusqu'à la colonne i . Pour chaque couture ainsi construite, la seule possibilité de changer de ligne tout en gardant les autres coutures connectées sur l'image originale est d'intervertir deux coutures adjacentes en passant de la colonne i à la colonne $i + 1$. Ainsi, on pourra intervertir ces deux coutures si et seulement si :

- Ces deux coutures n'ont pas déjà été interverties avec d'autres coutures entre la colonne i et la colonne $i + 1$.
- Cette interversion permet de réduire l'énergie de la couture ayant la somme d'énergie la moins grande dans la colonne i .
- Cette interversion n'est pas interdite par la disposition des coutures dans l'autre sens, c'est-à-dire qu'il n'y a pas de coutures verticales qui chevaucheraient ce morceaux de deux pixels.

Pour satisfaire la deuxième condition, on traite les coutures dans l'ordre croissant.

Cette deuxième méthode permet de garder un équilibre dans les deux sens de coutures, alors que la première méthode est plus efficace pour les images très allongées.

Affichage d'une multi-image. La durée entre l'appel de la fonction avec une certaine dimension de l'image et l'affichage de l'image doit être minimale pour réduire les temps de latence et avoir une meilleure fluidité.

Les opérations restantes après le calcul de la multi-image sont les réductions de l'image dans les deux sens. Cependant, les matrices d'énergie ne sont calculées que pour une image de même taille. Il faut alors réduire la deuxième matrice des énergies comme l'image originale pour utiliser cette deuxième matrice lors de la deuxième réduction.

La figure 11 représente la manipulation d'une multi-image dans le cas d'une image d'une licorne [7].

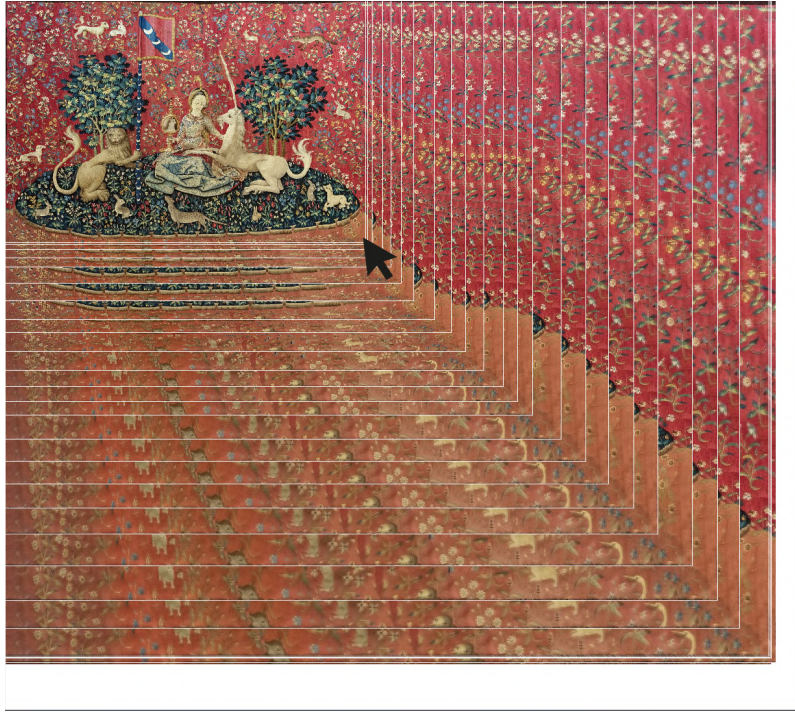


Figure 11: Affichage d'une multi-image.

4.3 Analyse des performances.

4.3.1 Temps de calcul : Méthode classique vs. méthode des pointeurs (cf. section 3.2.2).

Dans un premier temps, nous comparons les deux méthodes décrites dans la section 3.2.2, en nous restreignant au cas de la réduction d'image, et en utilisant la fonction d'énergie définie dans la section 3.1.1. La première méthode, fondée sur l'utilisation de pointeurs, présente une complexité en $O(kmn)$, où k désigne le nombre de coutures à supprimer. Cette méthode inclut un coût constant de l'ordre de $4mn$, correspondant aux opérations de transposition de l'image et de sa carte d'énergie, ainsi qu'à la réduction proprement dite de l'image. À titre d'exemple, pour $k = 200$, et avec l'image illustrée en figure 3, le temps d'exécution observé est d'environ 0,3 secondes.

En revanche, la méthode classique affiche une complexité en $O(k^2mn)$. Cette différence s'explique par le fait que l'accès aux voisins d'un pixel (i, j) ne s'effectue pas en temps constant, mais en $O(k)$, ce qui alourdit significativement le traitement. Pour le même cas d'étude ($k = 200$), cette méthode nécessite environ 1,2 seconde d'exécution.

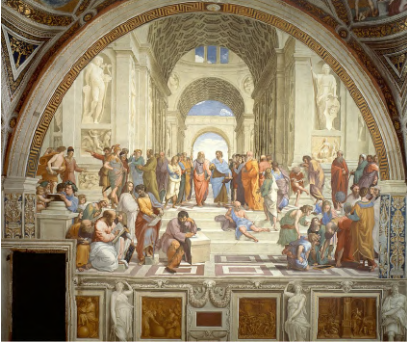
4.3.2 Impact du choix de la fonction d'énergie (cf. section 3.1) sur le résultat de la réduction d'image.

Dans cette section, nous comparons les performances des fonctions d'énergie introduites en section 3.1, appliquées à la réduction d'image.

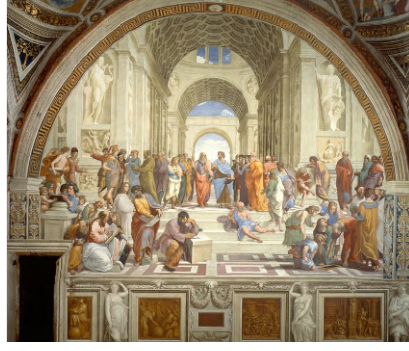
On observe tout d'abord que les fonctions E_1 et E_2 sont plus rapides à exécuter que les fonctions E_{HoG} et E_{entropie} . Cette différence s'explique par le fait que ces dernières nécessitent le calcul d'histogrammes, une opération plus coûteuse en temps de calcul.

Concernant la qualité des résultats, les quatre fonctions donnent des images visuellement proches lorsque le nombre de coutures supprimées est faible (environ 100). Cependant, lorsque ce nombre augmente, les différences deviennent plus perceptibles, même si les résultats produits par E_1 et E_2 restent très similaires entre eux.

La figure 12 illustre les écarts observés pour 100 coutures supprimées de l'image [8], en mettant l'accent sur le comportement spécifique de la fonction E_{entropie} . La figure 13 montre quant à elle une comparaison des quatre fonctions d'énergie pour une suppression de 200 coutures appliquée à l'image [4].



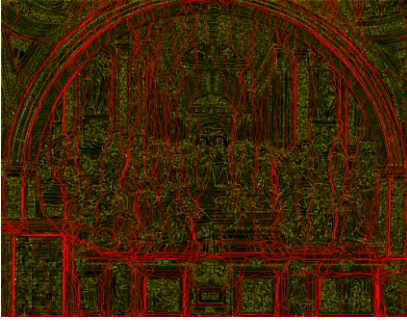
(a) Suppression de 100 coutures en utilisant la fonction d'énergie HoG.



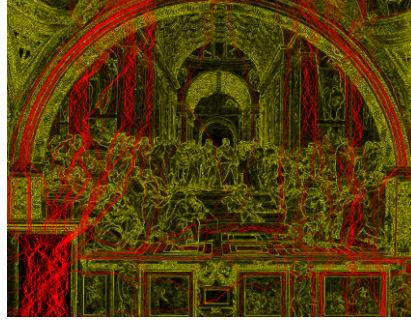
(b) Suppression de 100 coutures en utilisant la fonction d'énergie norme L^1 .



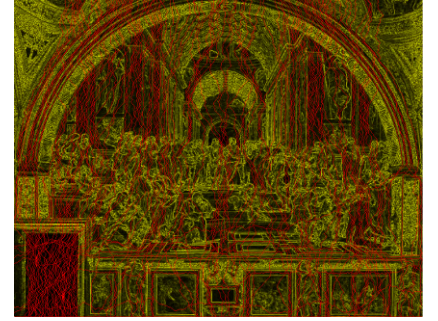
(c) Suppression de 100 coutures en utilisant la fonction d'énergie entropie.



(d) Affichage des coutures selon la fonction d'énergie HoG.



(e) Affichage des coutures selon la fonction d'énergie norme L^1 .



(f) Affichage des coutures selon la fonction d'énergie entropie.

Figure 12: Analyse comparative des effets de la suppression de 100 coutures selon la fonction d'énergie employée (E_{HoG} , E_1 , et E_{entropie}). On observe notamment des variations significatives au niveau du cadre noir en bas à gauche de l'image entre les cas (a), (b) et (c).



(a) Suppression de 200 coutures en utilisant la fonction d'énergie entropie.



(b) Suppression de 200 coutures en utilisant la fonction d'énergie HoG.



(c) Suppression de 200 coutures en utilisant la fonction d'énergie norme L^1 .



(d) Suppression de 200 coutures en utilisant la fonction d'énergie norme L^2 .

Figure 13: Comparaison des résultats obtenus par les quatre fonctions d'énergie pour la suppression de 200 coutures. On observe que le traitement de la région du visage varie selon la fonction utilisée, la fonction E_2 apparaissant ici comme la plus adaptée pour préserver les détails de l'image.

5 Conclusion.

5.1 Défis rencontrés.

5.1.1 Implémentation technique.

La principale difficulté rencontrée concerne l'équilibre entre qualité et performance. En C++, l'accès aux données image, la gestion dynamique de la mémoire et les calculs matriciels nécessitent une attention particulière pour éviter des ralentissements importants.

De plus, certaines opérations (recalculs de la matrice d'énergie, mise à jour de l'image après suppression de coutures) introduisent une complexité algorithmique non négligeable, qui pourrait être optimisée via des structures de données plus adaptées ou des traitements partiels.

5.1.2 Limites de la méthode.

Malgré ses qualités, le *Seam Carving* échoue dans certains cas :

- Sur des images contenant des motifs répétitifs ou des structures fines, les coutures peuvent fragmenter visuellement ces éléments.

La figure 14 présente une image comportant des motifs répétitifs issus d'un emballage [9], mettant en évidence l'échec de la méthode de *Seam Carving* dans la gestion des structures répétitives.



(a) Image originale d'un motif d'emballage répétitif, destinée à être amplifiée par *Seam Carving*.

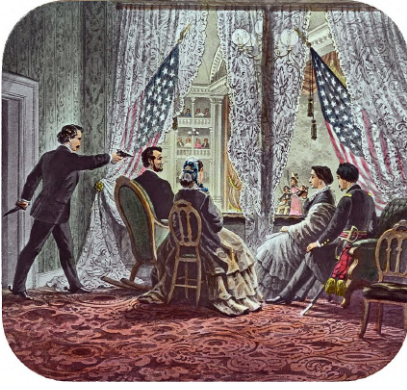


(b) Échec de l'amplification : apparition de zones floues et altération globale des détails de l'image

Figure 14: Illustration comparative des limites du redimensionnement par *Seam Carving* sur une image comportant des motifs répétitifs. (a) Image originale avant toute transformation. (b) Résultat après amplification.

- Si l'image est trop condensée, c'est-à-dire qu'elle ne comporte pas de zones visuellement moins importantes, alors toute stratégie de redimensionnement — telle que le *Seam Carving* — sera vouée à l'échec.

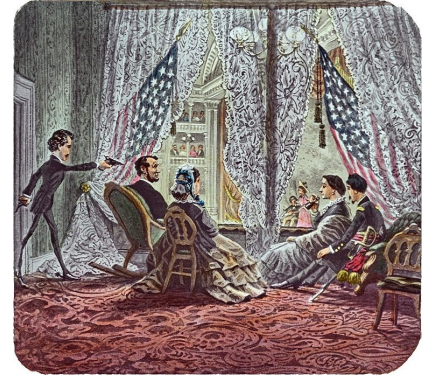
La figure 15 présente une image condensée [10] d'un tableau historique, illustrant l'échec de la méthode de *Seam Carving*, tant lors de l'agrandissement que de la réduction.



(a) Image initiale sur laquelle sera appliquée un agrandissement et une réduction.



(b) Échec de l'agrandissement et altérations visuelles marquées.



(c) Échec du redimensionnement par réduction : altérations dues à la suppression des coutures, affectant la structure de l'image.

Figure 15: Illustration comparative des limites du redimensionnement par *Seam Carving* sur une image trop condensée. (a) Image originale avant toute transformation. (b) Résultat après agrandissement. (c) Résultat après réduction.

Ces limites soulignent l'importance de mieux contextualiser l'énergie visuelle par rapport à la perception humaine.

5.2 Synthèse des contributions.

Ce projet a permis d'explorer et d'implémenter l'algorithme de *Seam Carving*, une méthode innovante de redimensionnement d'images qui préserve le contenu visuel important. Nous avons développé une solution complète en C++ pour le redimensionnement d'images en niveaux de gris et en couleurs, en utilisant différentes fonctions d'énergie pour identifier les zones les moins importantes. L'algorithme de programmation dynamique a été utilisé pour déterminer les coutures optimales à supprimer ou à insérer, permettant ainsi un redimensionnement structurellement conscient du contenu.

Nous avons également exploré des applications avancées telles que l'agrandissement d'images, l'amplification d'images, le recadrage, la reconstruction de POISSON, la suppression d'objets guidée par l'utilisateur et la gestion des multi-images. Ces extensions ont démontré la flexibilité et la robustesse de la méthode de *Seam Carving* pour diverses tâches de traitement d'images.

En termes de performances, nous avons comparé deux méthodes d'implémentation et évalué leur complexité algorithmique. La méthode utilisant des pointeurs s'est avérée plus efficace, avec une complexité en $O(kmn)$, tandis que la méthode classique affichait une complexité en $O(k^2mn)$. Ces résultats soulignent l'importance de choisir des structures de données et des algorithmes appropriés pour optimiser les performances.

Enfin, nous avons analysé les performances des différentes fonctions d'énergie et leur impact sur la qualité des images redimensionnées. Les fonctions E_1 et E_2 se sont révélées plus rapides à exécuter, tandis que les fonctions E_{HoG} et E_{entropie} ont montré des résultats visuellement proches pour un faible nombre de coutures supprimées, mais des différences plus perceptibles pour un nombre plus élevé de coutures. Ces observations mettent en lumière l'importance du choix de la fonction d'énergie pour obtenir des résultats de haute qualité.

6 Annexe

Pseudo-codes sources (extraits clés).

Cette section rassemble les différents pseudocodes présentés dans les diverses parties de ce rapport.

Algorithm 1: Recherche de la couture verticale optimale

Data: Image d'énergie E de taille $n \times m$
Result: Couture verticale optimale

```
1 for  $j \leftarrow 1$  to  $m$  do
2    $M(0, j) = E(0, j)$ 
3 for  $i \leftarrow 2$  to  $n$  do
4   for  $j \leftarrow 1$  to  $m$  do
5      $M(i, j) \leftarrow E(i, j) + \min [M(i-1, j-1), M(i-1, j), M(i-1, j+1)]$  (en respectant les bords)
6  $j_{\min} \leftarrow \arg \min_j M(n, j)$ 
7  $seam[n] \leftarrow j_{\min}$ 
8 for  $i \leftarrow n-1$  to 1 do
9    $j \leftarrow seam[i+1]$ 
10   $seam[i] \leftarrow j'$  tel que  $M(i, j')$  est minimal parmi
11     $M(i, j-1), M(i, j), M(i, j+1)$  (indices valides uniquement)
12 return  $seam$ 
```

Algorithm 2: Recherche de la couture horizontale optimale

Data: Image d'énergie E de taille $n \times m$
Result: Couture horizontale optimale

```
1 for  $j \leftarrow 1$  to  $m$  do
2    $M(i, 0) = E(i, 0)$ 
3 for  $j \leftarrow 2$  to  $m$  do
4   for  $i \leftarrow 1$  to  $n$  do
5      $M(i, j) \leftarrow E(i, j) + \min [M(i-1, j-1), M(i, j-1), M(i+1, j-1)]$  (en respectant les bords)
6  $i_{\min} \leftarrow \arg \min_i M(i, m)$ 
7  $seam[m] \leftarrow i_{\min}$ 
8 for  $j \leftarrow m-1$  to 1 do
9    $i \leftarrow seam[j+1]$ 
10   $seam[j] \leftarrow i'$  tel que  $M(i', j)$  est minimal parmi
11     $M(i-1, j), M(i, j), M(i+1, j)$  (indices valides uniquement)
12 return  $seam$ 
```

Algorithm 3: Calcul de l'énergie HoG (E_{HoG})

Input: Pixel (i, j) , Gradients ∇_x, ∇_y , Fonction L^1 , Taille de fenêtre $w = 10$, Nombre de bins $nb_bins = 9$

Output: Énergie $E_{HoG}(i, j)$

```
1 Function E_HoG( $i, j, \nabla_x, \nabla_y, L^1, w = 10, nb\_bins = 9$ ):
2   half_window  $\leftarrow \frac{w}{2}$ 
3   histogram  $\leftarrow [0.0] * nb\_bins$ 
4   total  $\leftarrow 0.0$ 
5   for  $dx = -half\_window$  to  $half\_window$  do
6     for  $dy = -half\_window$  to  $half\_window$  do
7        $px \leftarrow i + dx$ 
8        $py \leftarrow j + dy$ 
9       if  $px \geq 0$  and  $px < image\_width$  and  $py \geq 0$  and  $py < image\_height$  then
10         $gx \leftarrow \nabla_x(px, py)$ 
11         $gy \leftarrow \nabla_y(px, py)$ 
12         $angle \leftarrow orientation(gx, gy)$ 
13         $bin \leftarrow bin\_index(angle, nb\_bins)$ 
14         $mag \leftarrow norm(gx, gy)$ 
15        histogram[bin]  $\leftarrow$  histogram[bin] + mag
16   total  $\leftarrow sum(histogram)$ 
17   if total  $> 0.0$  then
18     for  $k = 0$  to  $nb\_bins - 1$  do
19       histogram[k]  $\leftarrow$  histogram[k] / total
20   max_bin_value  $\leftarrow max(histogram)$ 
21   if max_bin_value == 0.0 then
22      $E_{HoG}(i, j) \leftarrow 0.0$ 
23   else
24      $E_{HoG}(i, j) \leftarrow \frac{L^1(i, j)}{max\_bin\_value}$ 
25   return  $E_{HoG}(i, j)$ 
```

Algorithm 4: Calcul de l'énergie d'entropie

Input: Pixel (i, j) , Gradients ∇_x, ∇_y , Fonction L^1 , Taille de fenêtre $w = 10$, Nombre de bins $nb_bins = 9$

Output: Énergie $E_{entropie}(i, j)$

```
1 Function E_entropie( $i, j, \nabla_x, \nabla_y, L^1, w = 10, nb\_bins = 9$ ):
2   half_window  $\leftarrow \frac{w}{2}$ 
3   histogram  $\leftarrow [0.0] * nb\_bins$ 
4   total  $\leftarrow 0.0$ 
5   Line 5 ... Line 19 (cf. algorithm 3)
6   entropy  $\leftarrow 0.0$ 
7   for  $k = 0$  to  $nb\_bins - 1$  do
8     if histogram[k]  $> 0.0$  then
9       entropy  $\leftarrow$  entropy - histogram[k]  $\times \log(histogram[k])$ 
10   $E_{entropie}(i, j) \leftarrow L^1(i, j) + entropy$ 
11  return  $E_{entropie}(i, j)$ 
```

References

- [1] S. Avidan and A. Shamir, *Seam Carving for Content-Aware Image Resizing*, ACM Transactions on Graphics (TOG), vol. 26, no. 3, 2007.
- [2] P. Pérez, M. Gangnet, A. Blake *Poisson image editing*, ACM Transactions on Graphics (TOG), vol. 22, no. 3, 2003.

- [3] *Méthode de surrelaxation successive*. (2025, mars 16). Wikipédia, l'encyclopédie libre. Page consultée le 11 mai 2025 à partir de https://fr.wikipedia.org/wiki/M%C3%A9thode_de_surrelaxation_successive
- [4] Image Lena. Available: [https://en.wikipedia.org/wiki/File:Lenna_\(test_image\).png](https://en.wikipedia.org/wiki/File:Lenna_(test_image).png)
- [5] Image Tour. Available: https://en.wikipedia.org/wiki/File:Broadway_tower_edit.jpg
- [6] Image Criquet. Available: [https://commons.wikimedia.org/wiki/File:Choc%C3%B3_grasshopper_\(Opaon_varicolor\)_male.jpg](https://commons.wikimedia.org/wiki/File:Choc%C3%B3_grasshopper_(Opaon_varicolor)_male.jpg)
- [7] Image Licorne. Available: [https://commons.wikimedia.org/wiki/File:\(Toulouse\)_Le_Vue_\(La_Dame_%C3%A0_la_licorne\)_-_Mus%C3%A9_de_Cluny_Paris.jpg](https://commons.wikimedia.org/wiki/File:(Toulouse)_Le_Vue_(La_Dame_%C3%A0_la_licorne)_-_Mus%C3%A9_de_Cluny_Paris.jpg)
- [8] Image Académie. Available: https://commons.wikimedia.org/wiki/File:%22The_School_of_Athens%22_by_Raffaello_Sanzio_da_Urbino.jpg
- [9] Image Soupe. Available: <https://www.artchive.com/artwork/100-cans-andy-warhol-1962/#:~:text=%E2%80%9C100%20Cans%2C%E2%80%9D%20created%20by%20Andy%20Warhol%20in%201962%2C,the%20Albright-Knox%20Art%20Gallery%20in%20Buffalo%2C%20NY%2C%20US.>
- [10] Image Assassinat. Available: https://commons.wikimedia.org/wiki/File:Lincoln_assassination_slide_c1900_-_Restoration.jpg