Projet Inpainting (4IM01) : Rapport intermédiaire

Daniel Akbarinia & Abdennour Kerboua

October 14, 2024

1 Présentation du sujet

L'inpainting basé sur des exemples (exemplar-based) est une méthode de restauration d'images qui remplit les zones manquantes d'une image en réutilisant les parties similaires de l'image intacte, permettant de conserver les textures et motifs visuels pour un résultat plus réaliste. La technique proposée par l'article permet donc d'allier une solution de type diffusion donnant de l'importance à la reproduction des contours et formes géométriques de l'image et de reproduction sur base d'exemples. Les formes présentes autour de la zone à remplir influencent en effet l'ordre de remplissage des pixels.

2 Avancement

L'article décrivant l'implémentation proposant un processus détaillé par étape, nous avons décidé de les suivre. Voici un résumé de notre état d'avancement sur chacune de ces étapes.

1. **a.** Identification du front de remplissage (noté $\delta\Omega^t$, dans l'article) : on a implémenté deux méthodes. La première consiste à maintenir à jour une matrice indiquant pour chaque pixel son état de remplissage (`True` pour rempli, `False` pour en attente de remplissage). A partir de cette matrice, on décide pour chaque pixel, s'il appartient au front de remplissage par 8-connexité (un pixel $p_{i,j}$ non rempli appartient à $\delta\Omega^t$ si et seulement si $\exists k, l \in [i-1, i+1] \times [j-1, j+1]$, tel que $p_{k,l}$ est rempli). La seconde est une variante équivalente qui convolue le masque (valant 1 en les pixels remplis et 0 autre part) avec le filtre : $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$. En prenant l'intersection des pixels nuls dans l'image de départ et des pixels strictement positifs dans l'image résultante de la convolution, on obtient le front de remplissage.

b. Calcul des priorités : Ce calcul implique le calcul de deux facteurs : la confidence $C(p)$ et le terme *data* $D(p)$,

- **Confidence** : On maintient à jour une matrice `confidence` de même taille que l'image qui contient pour chaque pixel $p_{i,j}$ la valeur de la confidence de ce pixel. Elle est initialisée en affectant 0 aux pixels de la zone cible et 1 à ceux de la zone source. Au début de chaque itération du programme, la priorité des pixels du contour actuel est calculé suivant cette formule :

$$C(p) = \frac{\sum_{q \in \Psi_p \cap \bar{\Omega}} C(q)}{|\Psi_p|} \tag{1}$$

  Ce terme permet de favoriser le remplissage en cercle concentrique.

- **Coefficient *Data*** : Il s'obtient à partir de cette formule : $D(p) = \frac{|\overrightarrow{\nabla I}^{\perp} \cdot \overrightarrow{n_p}|}{\alpha}$. Avec,
  - $\overrightarrow{n_p}$: le vecteur normal au contour de remplissage au pixel p, on le calcule en prenant le gradient au point p du masque (valant 0 au pixel non-rempli et 1 ailleurs)
  - $\overrightarrow{\nabla I}^{\perp}$: l'orthogonal du vecteur gradient au point p. On explore actuellement plusieurs méthodes de calcul. La surface manquante étant noircie, un calcul usuel de différences au point p peut donner lieu à des valeurs aberrantes. Si l'accès à l'image d'origine est possible (dans le cas d'une suppression volontaire) le calcul du gradient peut se faire sur cette dernière. Sinon, nous avons implémenté :
    * un calcul moyennant le gradient sur les pixels de la zone remplie ne faisant pas intervenir les pixels de la zone noire (non-remplie)
    * un calcul prenant le maximum du gradient sur les pixels de la zone remplie ne faisant pas intervenir les pixels de la zone noire (non-remplie)

    Comme on utilise un filtre de Sobel pour calculer le gradient, pour éviter que les pixels noirs de la bordure influencent le gradient aux pixels voisins de la bordure, on remplace les pixels de la bordure par la moyenne des pixels remplis du patch.

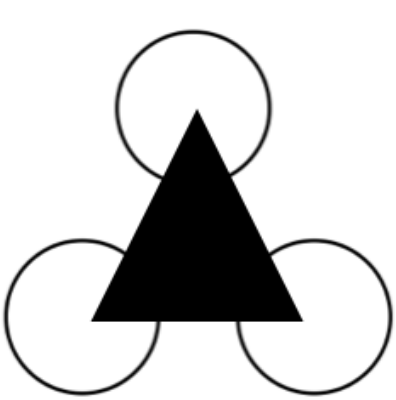
2. **a.** On obtient alors le terme de priorité pour chaque pixel du contour et on accède facilement au pixel de plus grande priorité. On sauvegarde alors le patch cible Ψ_p de taille fixée en début de programme (le plus souvent 9×9)

b. La fonction `distance` est implémentée de façon à renvoyer la SSD entre un patch Ψ_t et un patch source Ψ_s par des calculs élémentaires. On trouve alors le meilleur patch en parcourant tous les patchs de l'image et en retenant le patch Ψ_s tel que la valeur `distance`(Ψ_p, Ψ_s) est minimale. La recherche est l'étape la plus chronophage de l'algorithme. On expérimente alors des variantes limitant la recherche à un certain voisinage de la zone cible Ω.

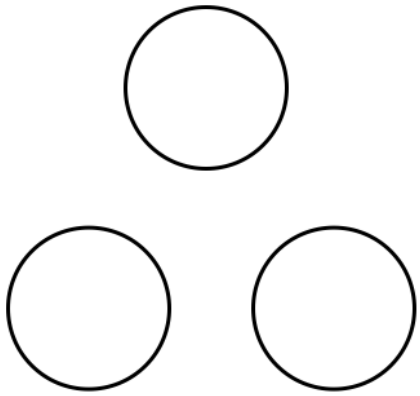
c. Etape triviale

3. On met à jour la matrice `priority` avec la formule: $C(\mathbf{q}) = C(\hat{\mathbf{p}}) \quad \forall \mathbf{q} \in \Psi_{\hat{p}} \cap \Omega$.

3 Résultats



(a) image masquée



(b) image remplie avec l'algorithme

Figure 1: Image formes

(a) Image de départ

(b) image remplie avec l'algorithme

Figure 2: Image d'un lion en couleur

(a) Image de départ (b) Image remplie avec l'algorithme

Figure 3: Image d'un bateau en couleur

(a) Image de départ (b) Image remplie avec l'algorithme

Figure 4: Image d'un saut en couleur

4 Conclusion et perspectives d'améliorations

En conclusion, à ce point de l'implémentation notre algorithme donne de très bons résultats sur des images en noir et blanc et des images de formes basiques. Cependant, le traitement des images couleur peut être encore amélioré pour éviter certaines irrégularités et artéfacts visibles (Figures (2), (3), (4)). Notre travail se concentrera sur les enjeux suivant :

- Amélioration du calcul du gradient (exploration d'autres méthodes de calcul)
- Amélioration du temps de calcul