Inpainting

Qu’est ce que l’inpainting ?

Supprimer des objets d’une photo et les remplacer par des arrière-plans visuellement plausibles.

Méthodes implémentées

Algorithme de criminisi

Fast digital inpainting

Fast digital inpainting

Objectifs :

- **Reconstruction rapide :** Proposer une méthode d'inpainting rapide et efficace.
- Préservation des contours : Assurer que les contours importants de l'image sont préservés pour un résultat visuellement cohérent.

Principe :

1. **Initialiser** la région à restaurer (Ω).
2. **Répéter plusieurs fois** (selon le nombre d'itérations) :

   Mélanger les valeurs (convolution) des pixels voisins pour remplir progressivement les zones manquantes, en évitant de modifier les pixels près des contours importants.

Récapitulatif :

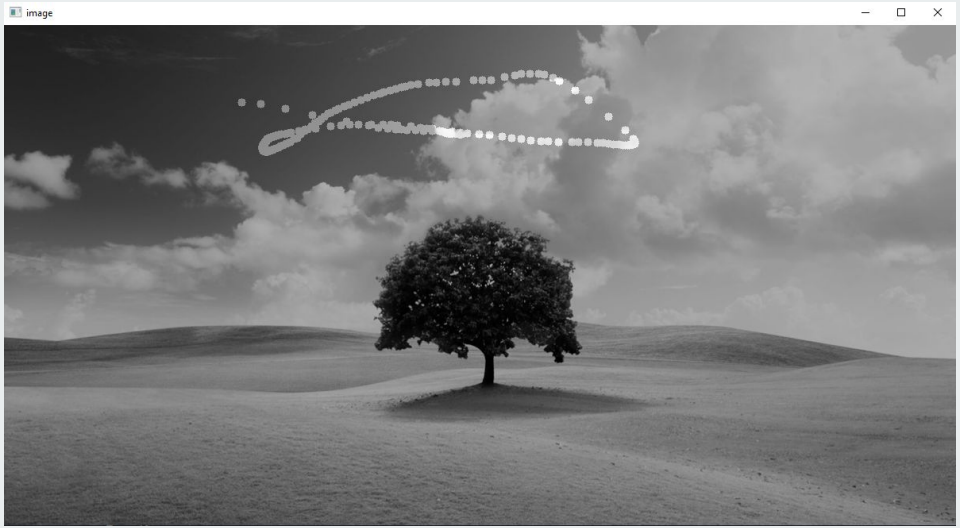
Etape 1 : choisir un masque adapté

Etape 2 : choisir nos paramètres adéquats (taille du filtre, type de filtre et nombre d'itérations)

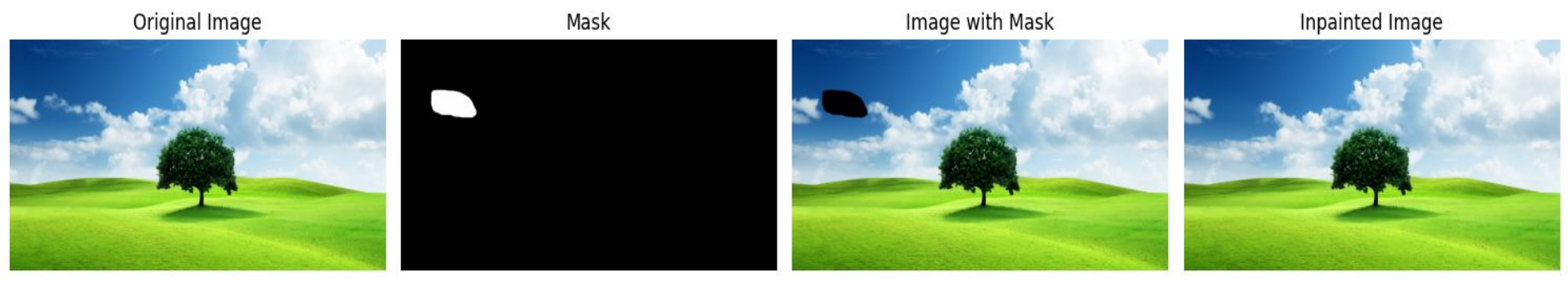
Etape 3 : Appliquer l'algorithme

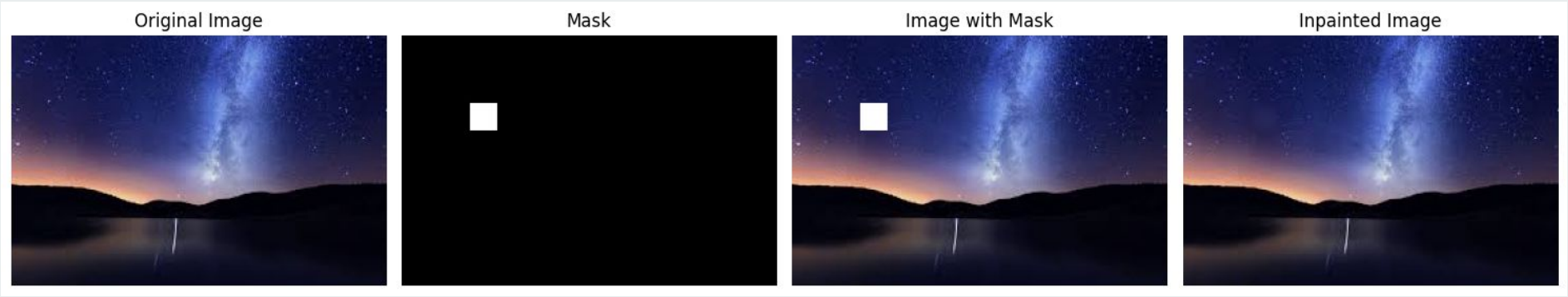
Comment créer son masque :

- Implémentation d'un méthode de détourage avec python (module open cv) :
- On y applique notamment une dilatation pour éviter les trous



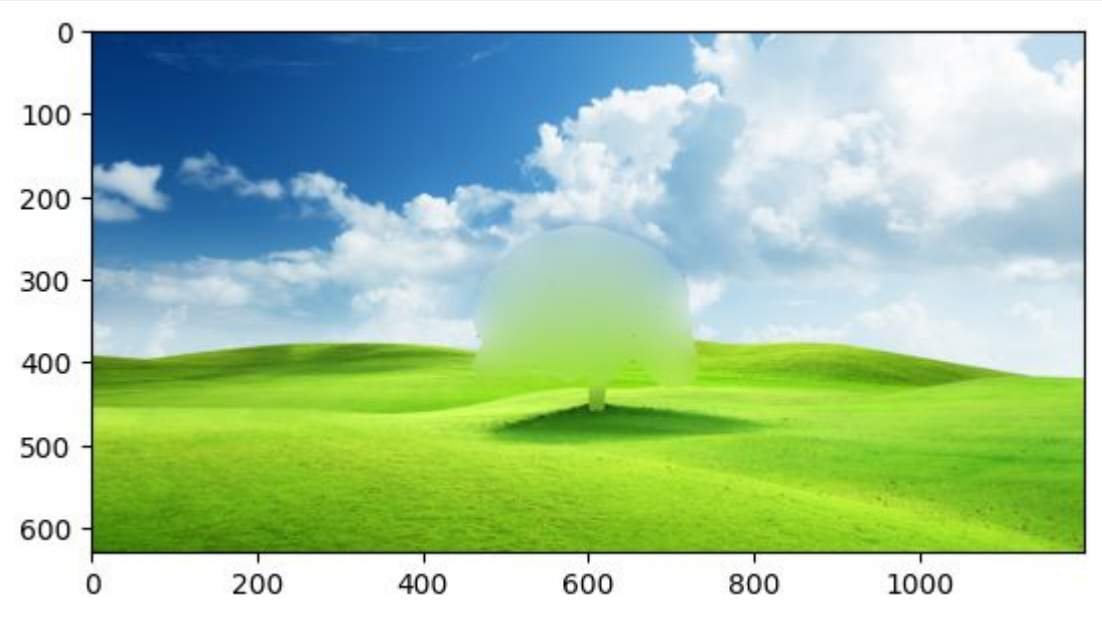
Exemple d’utilisation :



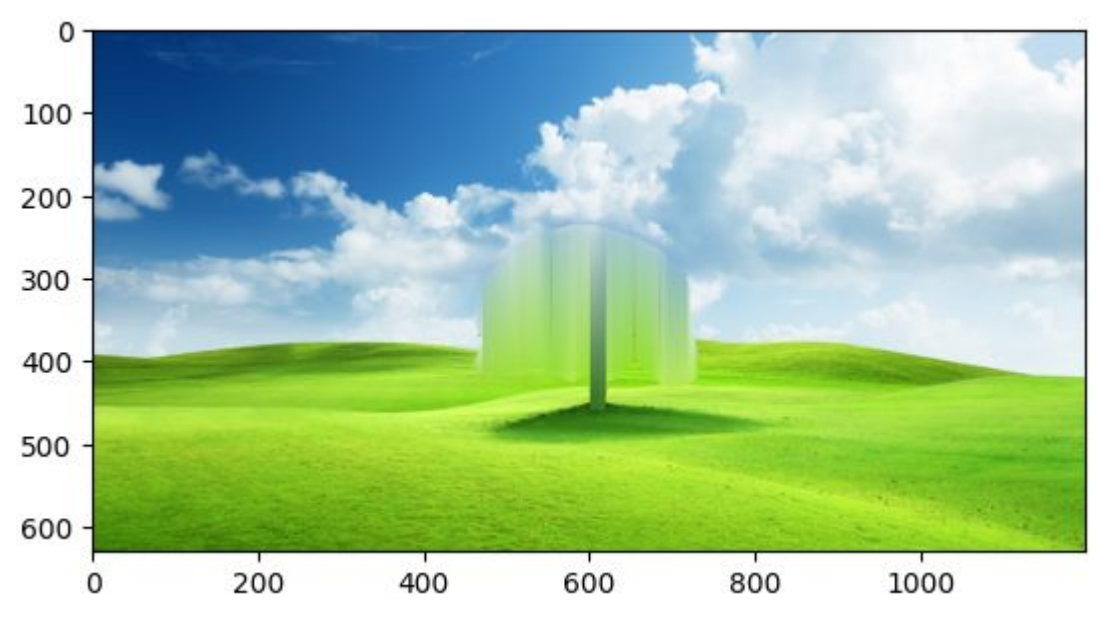

Original Image
Mask
Image with Mask
Inpainted Image

Et pour des éléments plus grands:

Résultats avec un filtre linéaire:



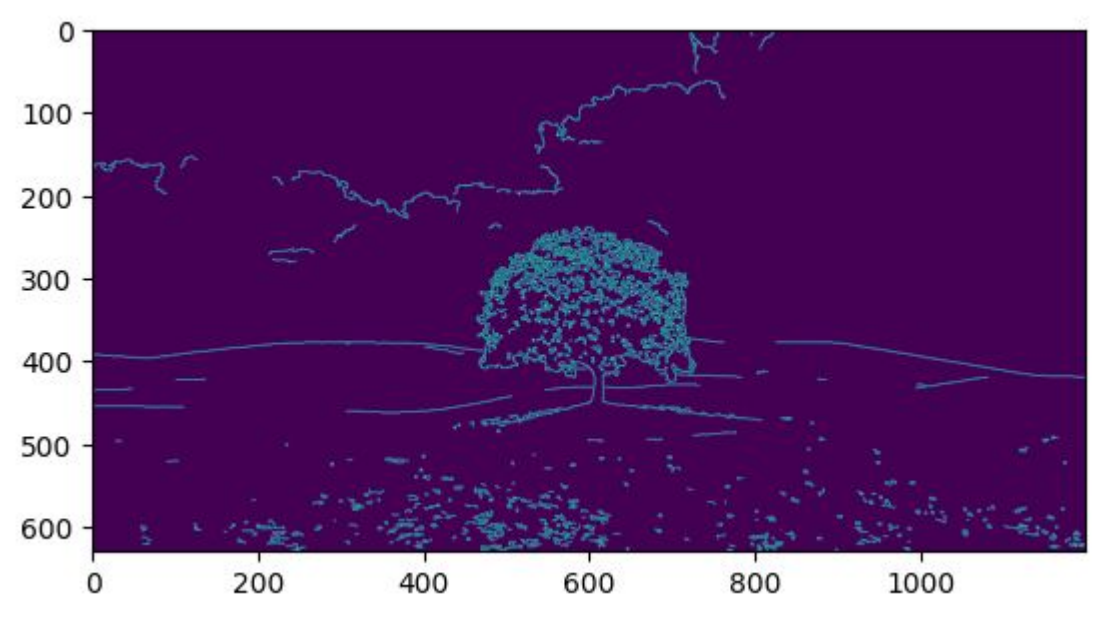
Résultats avec un filtre gaussien:



Limites de la méthode :

- Fonctionne avec de petits éléments
- Fonctionne mal avec des éléments situés sur les bords

Contours de Canny :



Algorithme de Criminisi

Antonio Criminisi
Microsoft Research

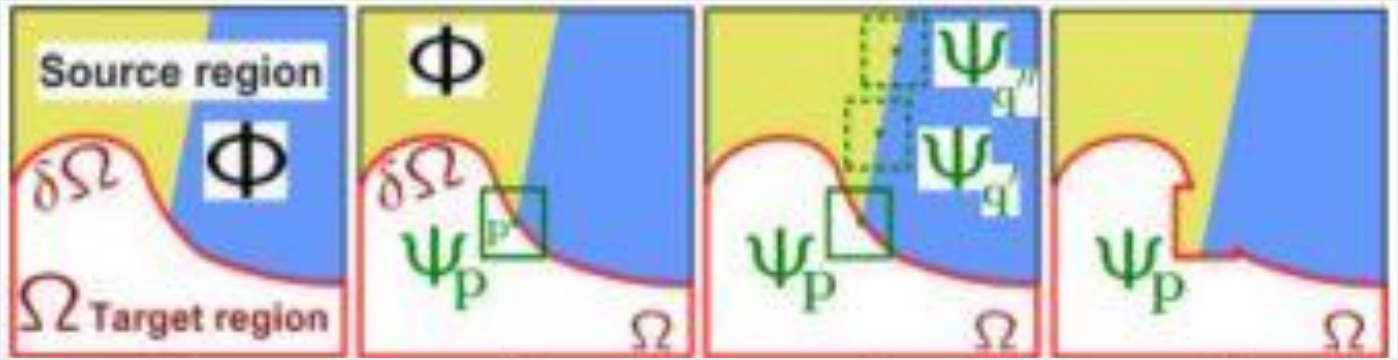
- L'approche géométrique :

les méthodes basées géométrie en utilisant des interpolations géométriques semi-locales.
Ces méthodes produisent des résultats intéressants sur la géométrie mais pas sur la texture.

- L'approche avec des patchs :

reconstruction de régions manquantes dans des images, par copier/coller de patchs d'images en calculant la similarité des patches.
Bonne synthèse de texture.
Assure pas une géométrie globale de la zone à remplir.

Explication de l'algorithme



Ω : Région cible qu'on souhaite enlever
δΩ : Contour de la région cible.
Φ : Région source
Ψp : Patch du pixel p **susceptible** de remplir la cible
Ψq : Patch source qu'on va copier dans Ψp

Quel pixel sur la frontière δΩ a la priorité la plus haute ?

Quel patch source à copier dans un patch cible spécifié ?

Détails de l'algorithme

- Extract the manually selected initial front $\delta\Omega^0$.
- Repeat until done:

1a. Identify the fill front $\delta\Omega^t$. If $\Omega^t = \emptyset$, exit.

1b. Compute priorities $P(\mathbf{p}) \quad \forall \mathbf{p} \in \delta\Omega^t$.

2a. Find the patch $\Psi_{\hat{\mathbf{p}}}$ with the maximum priority, *i.e.*, $\Psi_{\hat{\mathbf{p}}} \mid \hat{\mathbf{p}} = \arg\max_{\mathbf{p} \in \delta\Omega^t} P(\mathbf{p})$

2b. Find the exemplar $\Psi_{\hat{\mathbf{q}}} \in \Phi$ that minimizes $d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$.

2c. Copy image data from $\Psi_{\hat{\mathbf{q}}}$ to $\Psi_{\hat{\mathbf{p}}}$.

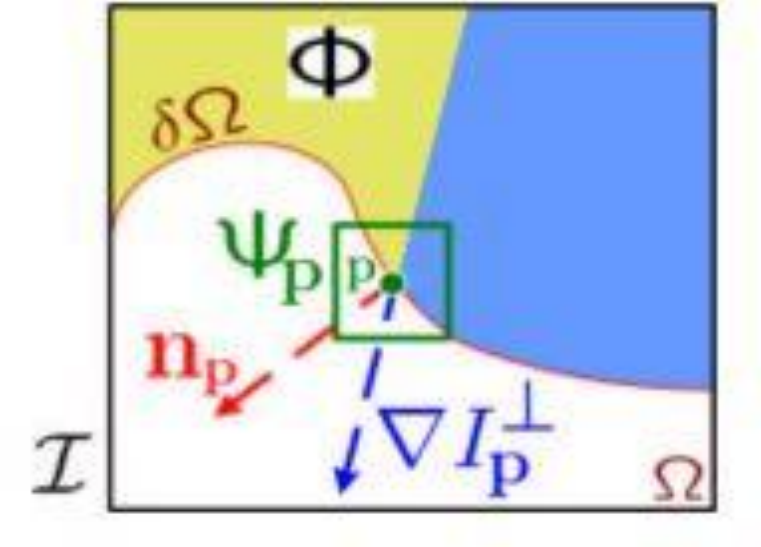
3. Update $C(\mathbf{p}) \quad \forall \mathbf{p} \mid \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega$

Priorité d'un pixel : P(p) = C(p)D(p)

C(p) terme de confiance = $\frac{\sum_{\mathbf{q} \in \psi_p \cap \Phi} C(q)}{|\psi_p|}$

avec initialement : $C(\mathrm{p}) = \begin{cases} 0, \forall \mathbf{p} \in \Omega \\ 1, \forall \mathbf{p} \in \Phi \end{cases}$

Terme de données



D(p) terme de données :

$$D(\mathbf{p}) = \frac{|\nabla I_{\mathrm{p}}^{\perp} \cdot \boldsymbol{n}_{\boldsymbol{p}}|}{\alpha}$$

np : la normale au contour δΩ. On effectue le produit vectoriel de la tangente ti = (xi+1-xi, yi+1 - yi, 0) et de (0,0,1) et on normalise.

α = 255 facteur de normalisation pour image 8 bits.

∇Ip : Isophote (orthogonale au gradient) on effectue le produit vectoriel du gradient de l'image avec (0,0,1)

Choix du meilleur patch source

Pour chaque patch candidat Ψq dans la région source , une mesure de similarité est calculée entre Ψp et Ψp. La mesure de similarité utilisée est la somme des différences au carré (SSD).

$$\mathrm{SSD}(\Psi_p, \Psi_q) = \sum_{(i,j)\in\Psi_p\cap\Phi} \left(I(\Psi_p(i,j)) - I(\Psi_q(i,j))\right)^2$$

I(Ψp(i,j)) : intensité du pixel dans le patch Ψp

Ψp∩Φ = l'ensemble des patchs sources Ψq

Pour chaque pixel p sur le front de remplissage, l'algorithme recherche le patch source Ψq dans la région source Φ qui minimise la mesure de similarité (SSD) :

$$\Psi_{\hat{q}} = \arg\min_{\Psi_q\in\Phi} \mathrm{SSD}(\Psi_p, \Psi_q)$$

Résultats

Le carré rouge représente le **patch cible** (ou le meilleur point à remplir) sur le front de remplissage. Ce patch est celui qui a été sélectionné pour être rempli en premier en fonction de la priorité calculée.

Le carré bleu représente le **meilleur patch source** trouvé dans la région source de l'image. Ce patch est celui qui minimise la distance (ou maximise la similarité) par rapport au patch cible.

Conclusion

Merci de votre attention

github : https://github.com/Lxvxo/Implementation-of-Inpainting-s-methods