Travaux réalisés

- Sélection de la zone à inpaint
  Le programme permet de sélectionner une zone à restaurer directement sur l'image à l'aide d'un polygone tracé à la souris, puis d'effectuer la reconstruction automatique des pixels manquants.

- Détection de la bordure de la zone à inpaint
  Nous avons utilisé la fonction cv2.morphologyEx(mask, cv2.MORPH_GRADIENT, ...) pour extraire précisément la bordure de la zone Ω (zone à remplir). Cette bordure est utilisée dans le calcul des priorités et dans le choix du point suivant à traiter.

- Gestion du calcul du gradient (isophote) malgré les valeurs inconnues
  Le calcul du gradient (filtre de Sobel) est adapté pour ne prendre en compte que les pixels connus.
  Concrètement, autour de chaque pixel de la bordure, l'algo regarde un voisinage (de rayon PATCH_RADIUS à choisir dans le code). Il n'utilise que les voisins dont les valeurs sont connues (mask == 0) pour estimer les dérivées partielles Ix et Iy. Cela permet d'obtenir un vecteur isophote cohérent même lorsque certaines valeurs de l'image sont encore manquantes.

- Calcul des priorités
  Les priorités des pixels de la bordure sont calculées selon le produit entre la confiance moyenne du patch et la valeur de l'isophote projetée sur la normale.
  Ce calcul repose sur les fonctions compute_isophote() et compute_normals() qui utilisent respectivement le gradient sur les pixels connus et le filtre de Sobel appliqué au masque.

- Stratégie de copie des patchs
  Lorsqu'un patch cible est partiellement inconnu (en bordure de la zone à inpaint), on ne remplace que la partie inconnue (mask == 255) du patch.
  Cependant, pour trouver le patch source le plus proche, on conserve une taille de patch constante (même si la zone à remplir est plus petite), afin d'assurer une comparaison fiable entre les régions de texture.

- Affichage interactif et visualisation du processus
  Une fenêtre affiche en temps réel la progression de l'algorithme.
  Les zones en rouge correspondent à la partie encore à inpaint, et le patch copié est surligné en vert.
  L'image est redimensionnée automatiquement pour un affichage plus clair.

Prochaines étapes

Au niveau du plan de travail, nous avons défini ces tâches par ordre chronologique et de priorité :

- Optimiser la recherche du meilleur patch source pour accélérer l’algorithme (utilisation possible d’une méthode de recherche approximative ou d’un espace de couleur réduit).

- Améliorer le calcul des priorités en testant différentes pondérations entre la confiance et le terme de données.

- Évaluer la qualité de la reconstruction sur plusieurs images tests et comparer visuellement avec l’inpainting OpenCV (cv2.inpaint).

- Ajouter une sauvegarde automatique du résultat final et éventuellement un mode non interactif (sélection via coordonnées). Cela permettrait d’ajouter une façon de retoucher la photo une fois l’algorithme fini (notamment pour itérer l’algorithme).

- Produire des exemples convaincants où l'algorithme a particulièrement bien fonctionné, et réussir à trouver quels facteurs font que l'algorithme fonctionne bien (ou mal) sur certaines images.