

INPAINTING

Yanis CHEIKH
Elouan LeBizec



Sommaire

Présentation du cadre théorique, de l'algorithme Criminisi et des choix de conception.



Approche du projet

Organisation du code



Décryptage des modules, des fonctions critiques et de la logique de traitement.

Analyse des résultats obtenus et des limites rencontrées dans certains cas.



Résultats et limites

Ouverture et Conclusion

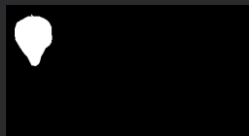


Synthèse du projet et pistes concrètes d'amélioration futures.

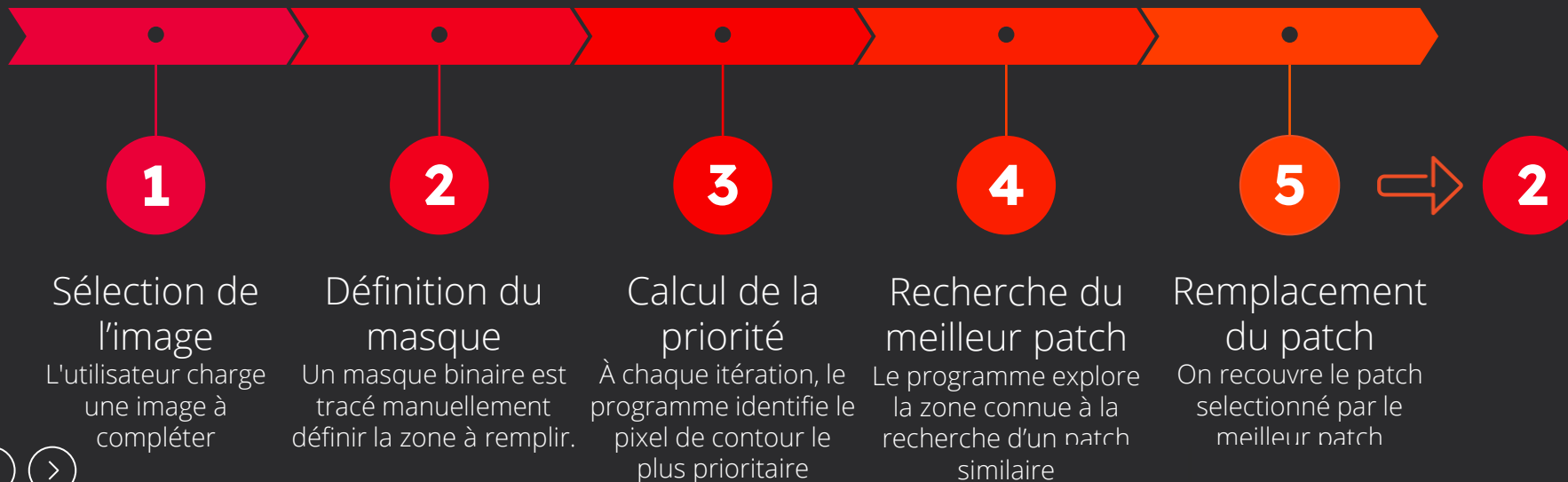
Approche du projet

Approche du projet

Présentation du cadre théorique, de l'algorithme Criminisi et des choix de conception

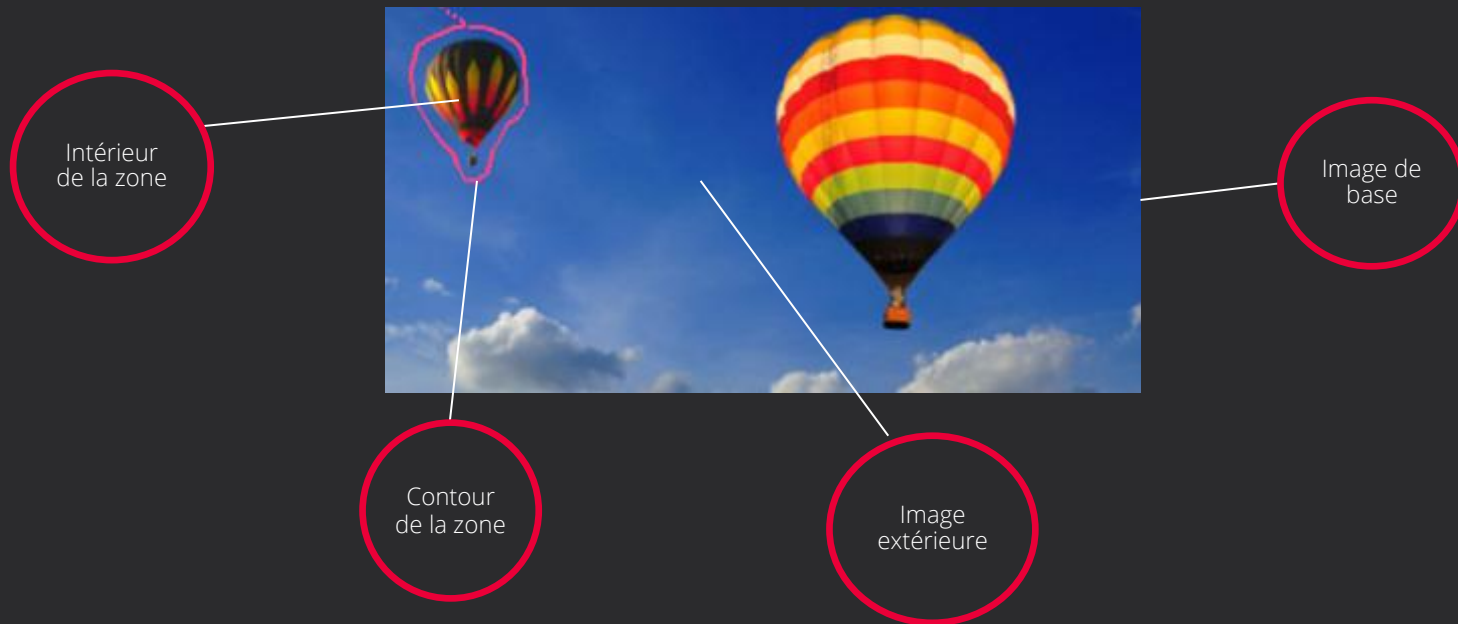


$$P(p) = C(p) * D(p)$$



Approche du projet

Définition des conditions du projet



Approche du projet

Calcul de la priorité



$$C(\mathbf{p}) = \frac{\sum_{\mathbf{q} \in \Psi_{\mathbf{p}} \cap (\mathcal{I} - \Omega)} C(\mathbf{q})}{|\Psi_{\mathbf{p}}|}$$

C(p)

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

D(p)

$$P(\mathbf{p}) = C(\mathbf{p}) * D(\mathbf{p})$$

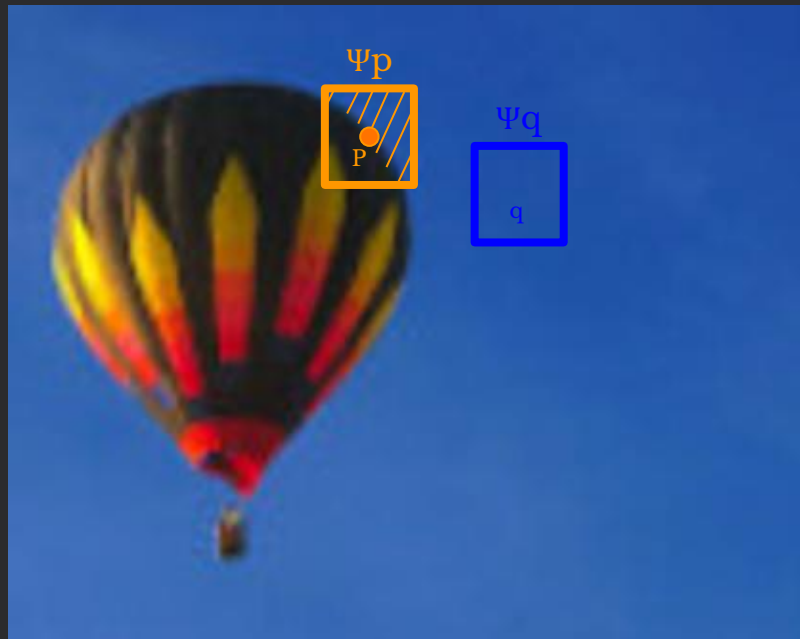
P(p)

Meilleur patch : Max P(p)



Approche du projet

Calcul du meilleur patch



On garde les pixels
sources du patch

Ψ_p

On prend un patch
candidat dans l'image

Ψ_q

On calcule la **similitudé**
entre les patches

d

Meilleur patch :
 $\min d(\Psi_p, \Psi_q)$.



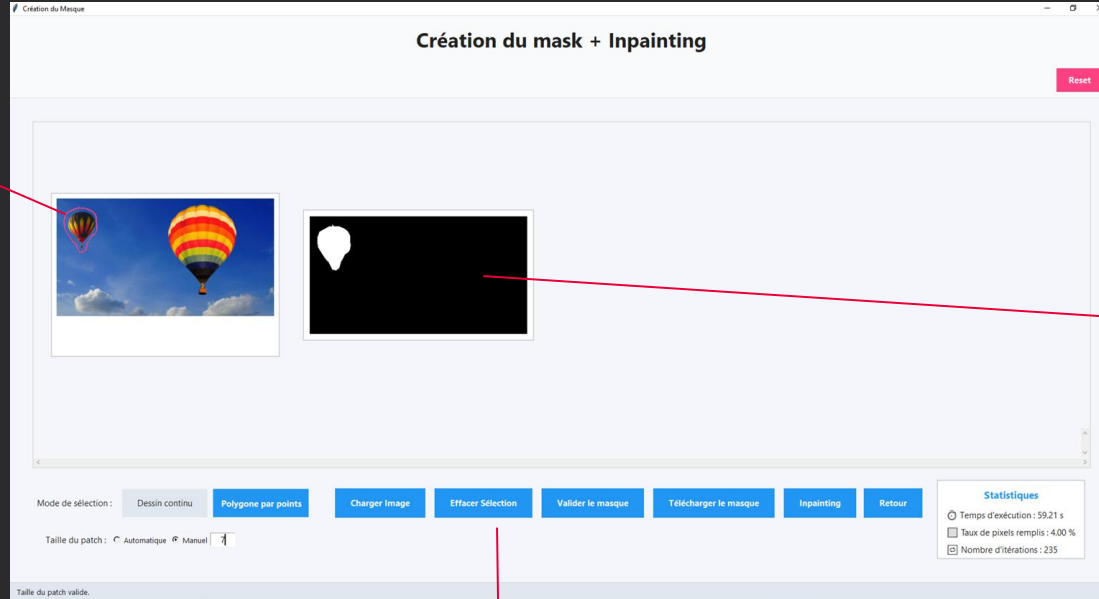
Démonstration

Organisation du code

Organisation du code

Centre de commande : le module Tkinter

Création
du mask



Mask

Commandes
utilisateurs

Algorithme du code de fonctionnement

Initialisation des variables

- Copie de l'image et du masque
- Carte de confiance



Début boucle

Tant que le mask n'est pas vide, on continue la boucle



Detection du front

get_fill_front()

La fonction permet de detecter le contour du mask (boundarymask en C)



Calcul des priorités

calcul_Cp() & calcul_Dp()

Pour chaque pixel du contour précédent on calcul sa priorité P(p).



Sélection du meilleur pixel

best_patch()

On prend le pixel p qui présente la plus grand valeur de P(p)



Algorithme du code de fonctionnement

Recherche du meilleur patch

trouver_meilleur_patch()

On parcourt l'image pour trouver le patch la plus "similaire" au patch source.



Copier le meilleur patch

copier_patch()

On parcourt l'image pour trouver le patch la plus "similaire" au patch source.



Mise à jour des variables

update_confiance()

On met à jours le mask et également la carte de la confiance



Retour au début de la boucle

Si le nouveau mask est vide, on peut sortir de la boucle



Fin de l'inpainting

Affichage de l'image finale.



Résultats et limites

Résultats et limites

Présentation des résultats avec divers paramètres



Image de référence



Résultat pour $N = 3$






Résultat net et
sans résidus



Temps long



Statistiques

-  Temps d'exécution : 435.90 s
-  Taux de pixels remplis : 3.32 %
-  Nombre d'itérations : 870

Résultats et limites

Présentation des résultats avec divers paramètres



Image de référence



Résultat pour $N = 5$



Résultat net et
sans résidus



Temps long

Statistiques

- 🕒 Temps d'exécution : 134.95 s
- ▒ Taux de pixels remplis : 3.32 %
- 📄 Nombre d'itérations : 351

Résultats et limites

Présentation des résultats avec divers paramètres



Image de référence



Résultat pour $N = 9$






Résultat net et avec 2
résidus



Temps
modéré

Statistiques

-  Temps d'exécution : 41.70 s
-  Taux de pixels remplis : 3.32 %
-  Nombre d'itérations : 123

Résultats et limites

Présentation des résultats avec divers paramètres



Image de référence



Résultat pour $N = 15$






Résultat net et avec
peu de résidus



Temps
modéré

Statistiques

-  Temps d'exécution : 13.02 s
-  Taux de pixels remplis : 3.32 %
-  Nombre d'itérations : 51

Résultats et limites

Présentation des résultats avec divers paramètres



Image de référence



Résultat pour $N = 21$



Résultat net et avec 10
résidus



Temps très
rapide

Statistiques

🕒 Temps d'exécution : 6.99 s

📊 Taux de pixels remplis : 3.32 %

📅 Nombre d'itérations : 27

Résultats et limites

Présentation des résultats avec divers paramètres



Image de référence



Résultat pour $N = 31$



Résultat pas net et
avec peu de résidus



Temps très
rapide

Statistiques

🕒 Temps d'exécution : 3.40 s

▒ Taux de pixels remplis : 3.32 %

📄 Nombre d'itérations : 13

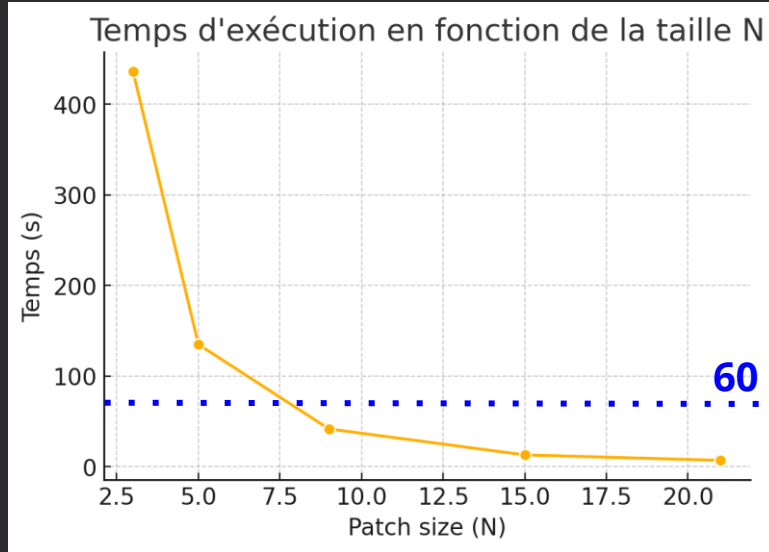
Résultats et limites

Présentation des résultats avec divers paramètres



Résultats et limites

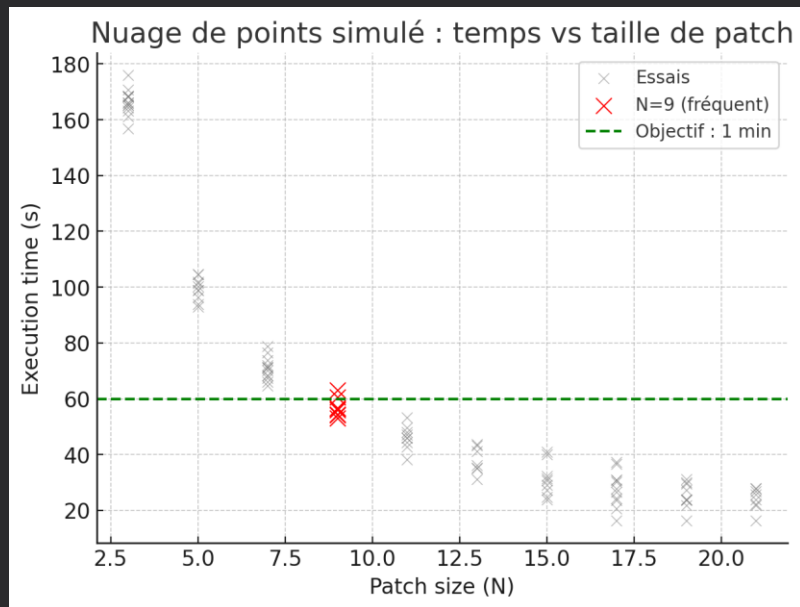
Présentation des résultats avec divers paramètres



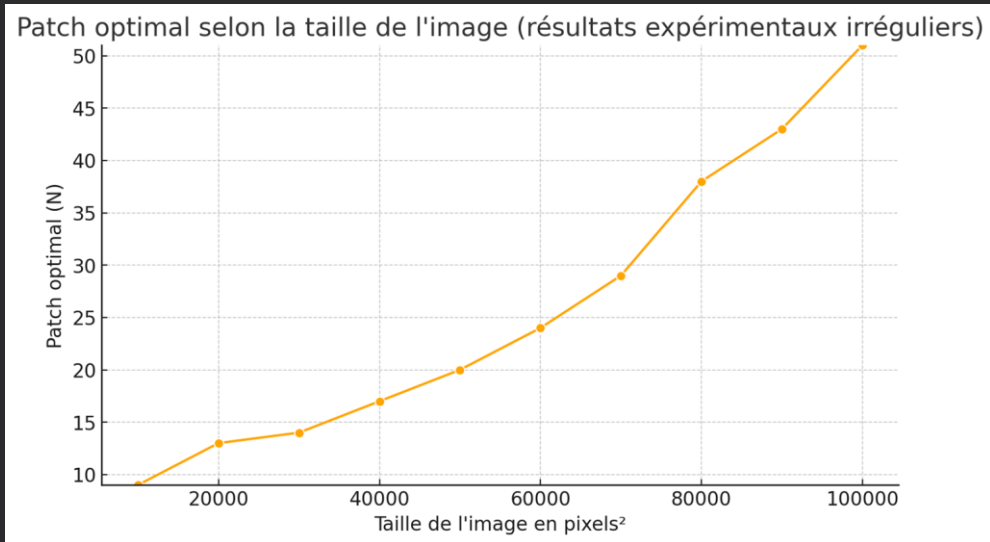
N=9 : taille idéale

Résultats et limites

Présentation des résultats avec divers paramètres



Mask de 4000 pixels²



10 tests sur différentes taille
de mask

Résultats et limites

Présentation des résultats avec divers paramètres



Image de référence



Taille du mask : 13 460 pixel²
N=13

Résultats et limites

Présentation des résultats avec divers paramètres



Image de référence

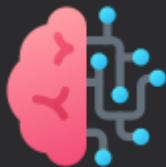


Taille du mask : 44 451 pixel²
N=19

Conclusion et ouverture

Ouverture et conclusion

Vers l'infini et au-delà



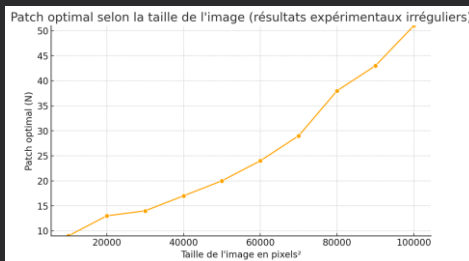
Trouver le meilleur patch à l'aide de l'IA



Adopter une taille de patch variable



S'adapter au contexte de l'image pour adopter une stratégie optimale



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

INPAINTING

Yanis CHEIKH
Elouan LeBizec

Merci pour
votre écoute