

Projet Inpainting - 4IM01

Yacine Khalil, Yassine Madani Alaoui
Encadré par : Saïd Ladjal

24-11-2024

Introduction

Ce projet a pour objectif de mettre en œuvre et d'évaluer l'algorithme présenté dans l'article [1], qui propose une méthode d'inpainting basée sur des exemples (exemplar-based inpainting) pour combler les zones inconnues d'une image en s'appuyant sur le contenu existant. Cette approche vise à remplacer de manière réaliste et convaincante les objets supprimés par un arrière-plan visuellement plausible, en intégrant les textures et structures environnantes de l'image.

L'algorithme utilise un ordre de remplissage spécifique et sélectionne des patchs qui permettent de propager efficacement les structures telles que les lignes et les contours d'objets. Ce choix de remplissage priorise les pixels pour lesquels les informations environnantes sont jugées les plus fiables. Ainsi, le résultat attendu après suppression d'un objet est une image dans laquelle la zone effacée est remplacée par un arrière-plan qui conserve l'aspect visuel de la région d'origine.

Dans ce rapport, nous décrivons les étapes de l'implémentation de cette méthode, les difficultés rencontrées et comment on a pu y remédier et présenter les résultats obtenus. En outre, nous analysons les performances de l'algorithme sur différents types d'images en discutant les points forts et les limitations observées sur des exemples bien choisis. Nous amènerons aussi des réflexions sur les éventuelles améliorations de l'algorithme pour mieux réagir face à certains cas où les limites de l'approche déterministe se font sentir.

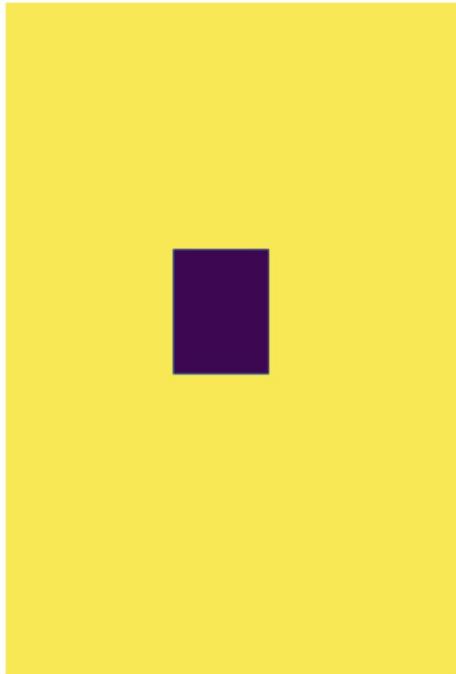
Implémentation de l'algorithme

1 – Manipulation des masques

Nous avons décidé de définir les masques de la manière suivante : les parties connues de l'image (ie : non masquées) seront mises à 1. La partie masquée de l'image est quant à elle mise à 0. La taille du masque est la même que celle de l'image car cela est plus pratique pour les calculs et les raisonnements ultérieurs. L'image et le masque sont deux entités différentes ; au cours de l'exécution de l'algorithme, l'image ainsi que le masque évoluent simultanément : l'image est remplie et le masque est réduit au fur et à mesure de l'avancement de l'algorithme.

1.1 – Masques rectangulaires

Au début, on a décidé de créer un masque rectangulaire en annulant les pixels qui se trouvent entre les lignes X_0 et X_1 et les colonnes Y_0 et Y_1 . En ce qui concerne le reste des pixels, ils sont mis à la valeur 1.



(a) Masque rectangulaire



(b) Masque appliqué à l'image

1.2 – Sélection manuelle du masque directement sur l'image

Pour une manipulation plus fluide, on a fait appel, grâce à la bibliothèque OpenCV, à une interface graphique qui nous permet de sélectionner la zone à masquer manuellement directement sur l'image tout en ayant la possibilité de changer la taille du curseur utilisé pour une sélection plus ou moins précise.



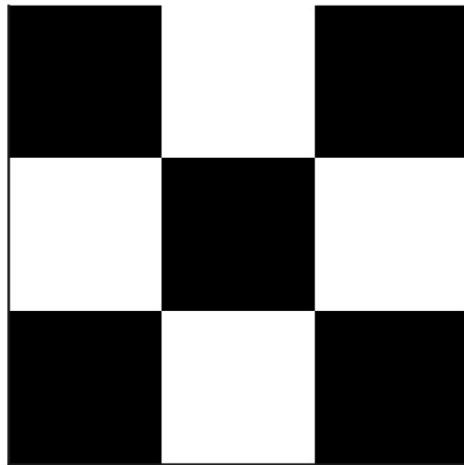
2 – Détection de bords

2.1 – Convolution avec un noyau bien choisi

Afin de déterminer les bords du masque, on a fait recours dans cette partie à un noyau de convolution qui nous permet de savoir si on se situe dans un point limite entre

les points nuls et ceux qui valent 1. Cette technique donnait les résultats escomptés ; détecter la frontière du masque.

Noyau de convolution



Noir = 0, Blanc = 1

2.2 – Utilisation de la morphologie mathématique

Après avoir étudié le cours de morphologie mathématique, on s'est rendu compte qu'on peut utiliser une simple érosion avec un disque de rayon 1 pour détecter les contours. En effet, le fait de soustraire l'érosion du masque au masque est équivalent à la convolution précédente. Cela nous a conduit finalement à adopter cette technique.



3 – Calcul des priorités

Dans cet algorithme, le calcul de la priorité joue un rôle clé pour déterminer l'ordre dans lequel les zones inconnues d'une image seront remplies. Il s'agit d'une métrique combinant plusieurs facteurs qui évaluent l'importance relative des différents points de la frontière du masque. L'objectif est de donner la priorité aux zones où les informations disponibles permettent une propagation cohérente des structures de l'image.

La priorité au niveau d'un point p de la frontière est exprimée par un terme $P(p)$, calculé comme le produit de deux composantes principales :

- Un terme de confiance $C(p)$ qui reflète la quantité d'information fiable dans la région entourant un point donné.
- Un terme de données $D(p)$ qui prend en compte la présence de structures linéaires, il est calculé en utilisant les gradients et les normales au niveau de la frontière.

Ces deux termes sont en compétition pour privilégier les zones structurées ou la complétion de texture, tout en garantissant que les informations déjà reconstruites sont solidement ancrées dans les nouvelles zones à remplir.

3.1 – Calcul du terme $C(p)$

Le terme de confiance évalue la quantité d'information fiable dans un patch à la frontière du masque. Il reflète la solidité des informations déjà reconstruites dans le voisinage immédiat du point en question.

Pour un patch Ψ_p centré sur un point p , le terme de confiance est défini comme suit :

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

où $C(q)$ représente la confiance du pixel q appartenant à l'intersection du patch Ψ_q avec la zone connue de l'image.

3.2 – Utilisation de l'algorithme avec ce seul terme

L'utilisation du terme de confiance seul fait que l'algorithme poursuit une trajectoire concentrique lors du remplissage ("Onion peel"). Cela donne des résultats acceptables dans certains cas particuliers (complétion de textures notamment), mais de mauvais résultats dans d'autres cas comme la propagation de structures nécessitant le terme $D(p)$ comme on va le voir juste après.

3.3 – Calcul du terme $D(p)$ et mise en évidence de son importance

Ce terme s'appuie sur les gradients pour détecter les isophotes qui doivent être propagés.

Pour un point p à la frontière du masque, le terme de données est défini comme suit :

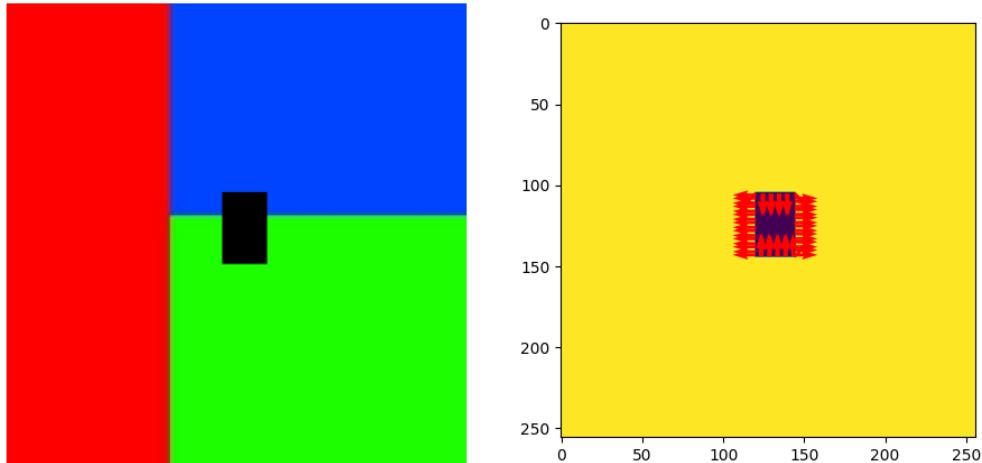
$$D(p) = \frac{|\nabla^\perp I_p \cdot \vec{n}_p|}{255}, \quad (2)$$

où ∇I_p est le gradient de l'intensité de l'image et \vec{n}_p est le vecteur normal à la frontière au point p .

3.3.1 - Calcul du vecteur normal

La méthode adoptée pour calculer les vecteurs normaux consiste à extraire des patchs locaux autour des points du contour du masque, puis à utiliser les gradients de ces patchs pour déterminer la direction normale à chaque point du contour. Les gradients $\frac{\partial I}{\partial x}$ et $\frac{\partial I}{\partial y}$ sont calculés à l'aide de *np.gradient*, qui mesure les variations d'intensité du masque dans les directions x et y. Au centre de chaque patch, ces gradients sont utilisés pour

former un vecteur perpendiculaire au contour qui est ensuite normalisé. Ce processus garantit que chaque vecteur normal est unitaire (de longueur 1) et orienté (localement) perpendiculairement au contour.



3.3.2 - Trois méthodes pour le calcul des gradients

a - Gradient basé sur le canal dominant

Cette fonction calcule le gradient d'une image aux points de la bordure en exploitant les trois canaux de couleur (Rouge, Vert, Bleu). Pour chaque point de la bordure :

- Elle extrait un patch centré sur le point et calcule les gradients (∇x et ∇y) pour chaque canal séparément.
- Elle calcule la norme du gradient pour chaque pixel dans le patch et pour chaque canal.
- Elle identifie le pixel dans le patch qui a la norme de gradient maximale, indépendamment du canal.
- Elle associe au point de la bordure les valeurs du gradient (∇x , ∇y) du pixel avec la norme maximale.

b - Gradient en niveaux de gris

Cette méthode simplifie le problème en convertissant l'image en niveaux de gris avant de calculer les gradients. Ensuite, pour chaque point de la bordure :

- Un patch est extrait.
- Les gradients (∇x , ∇y) sont calculés pour tous les pixels de ce patch.
- Les gradients sont moyennés pour fournir une estimation locale unique qui est attribuée au pixel de la bordure.

Cette méthode est rapide et efficace, mais elle perd les informations spécifiques aux canaux de couleur, ce qui limite sa précision dans des images où une couleur est dominante.

c - Gradient basé sur la moyenne des canaux

Ici, les gradients sont calculés séparément pour chaque canal (Rouge, Vert, Bleu) dans un patch centré autour d'un point de la bordure :

- Pour chaque canal, la moyenne des gradients est calculée à partir de tous les pixels du patch.

- Les moyennes des gradients des trois canaux sont ensuite combinées en effectuant leur moyenne arithmétique.
- Le résultat final est attribué au pixel de la bordure.

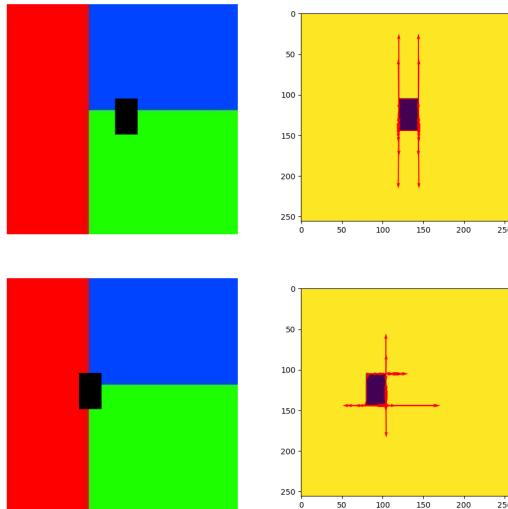


FIGURE 6 – Illustration pour la méthode retenue (moyenne)

3.4 – Calcul de la distance entre deux patchs

La comparaison entre deux patchs est une étape essentielle pour déterminer lequel correspond le mieux à la zone à remplir. Deux approches principales ont été utilisées pour calculer cette distance :



3.4.1 - Distance dans l'espace RGB : La fonction utilisée calcule la distance euclidienne pondérée entre les pixels des deux patchs, canal par canal (Rouge, Vert, Bleu). Les étapes sont les suivantes :

- Chaque patch est représenté par ses valeurs RGB.
- Les différences entre les valeurs des pixels des deux patchs sont calculées pixel par pixel pour chaque canal.
- Un masque binaire est appliqué afin d'ignorer les zones masquées dans le patch de référence.
- La distance finale est donnée par la somme des distances pour les trois canaux.

3.4.2 - Distance en niveaux de gris : Cette méthode simplifie le calcul en convertissant les patchs RGB en niveaux de gris. Elle suit les étapes suivantes :

- Les patchs sont convertis en niveaux de gris à l'aide d'une pondération standard basée sur la luminosité perçue.
- Une distance euclidienne est calculée entre les deux patchs en niveaux de gris.
- Les zones masquées sont pondérées par 0 pour ne pas influencer le résultat final.

À noter qu'on a opté pour la distance dans l'espace RGB car cela nous a semblé plus pertinent pour les images en couleurs. Nous avons aussi essayé de travailler dans l'espace CIELAB pour tester si cela était meilleur, mais ce changement n'a pas significativement amélioré les résultats.

Tests et résultats

1 – Résultats pour la complétion de textures :

1.1 – Exemple du tissu :

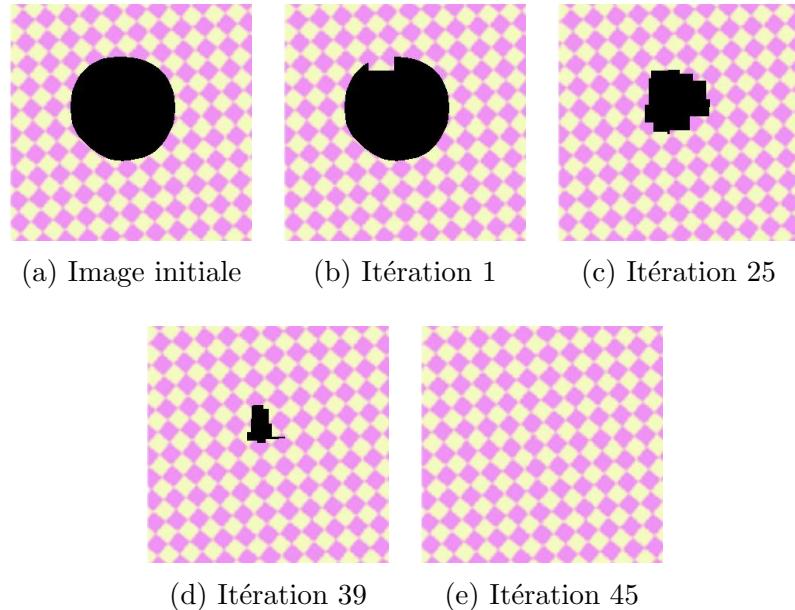
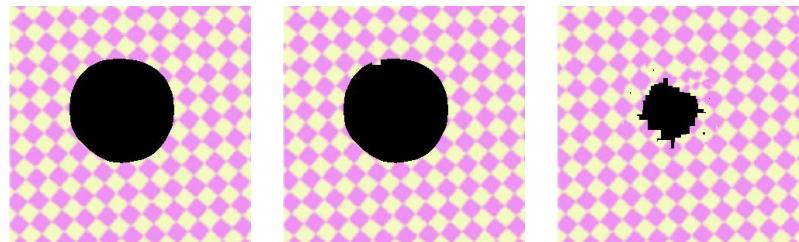
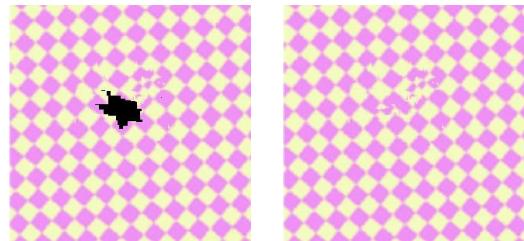


FIGURE 7 – Algorithme appliqué à l'image initiale avec un patch de taille 33.



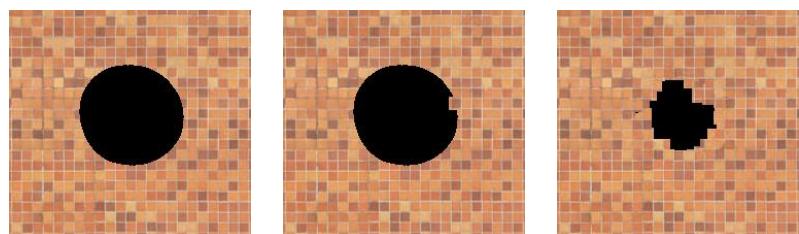
(a) Image initiale (b) Itération 1 (c) Itération 232



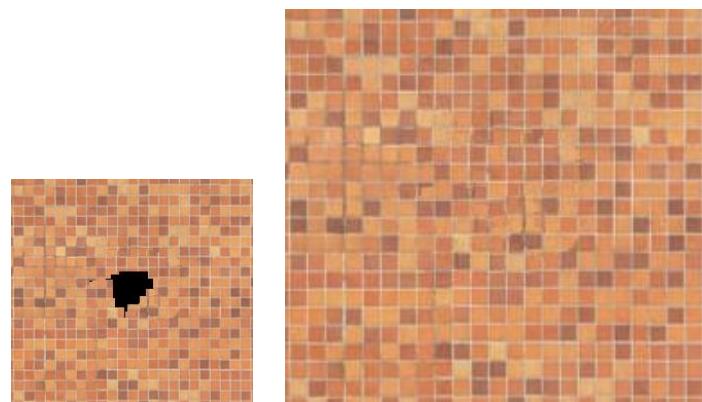
(d) Itération 300 (e) Itération 338

FIGURE 8 – Algorithme appliqué à l'image initiale avec un patch de taille 11.

1.2 – Exemple du carrelage :



(a) Image initiale (b) Itération 1 (c) Itération 74



(d) Itération 110 (e) Itération 127

FIGURE 9 – Algorithme appliqué à l'image initiale avec un patch de taille 13.

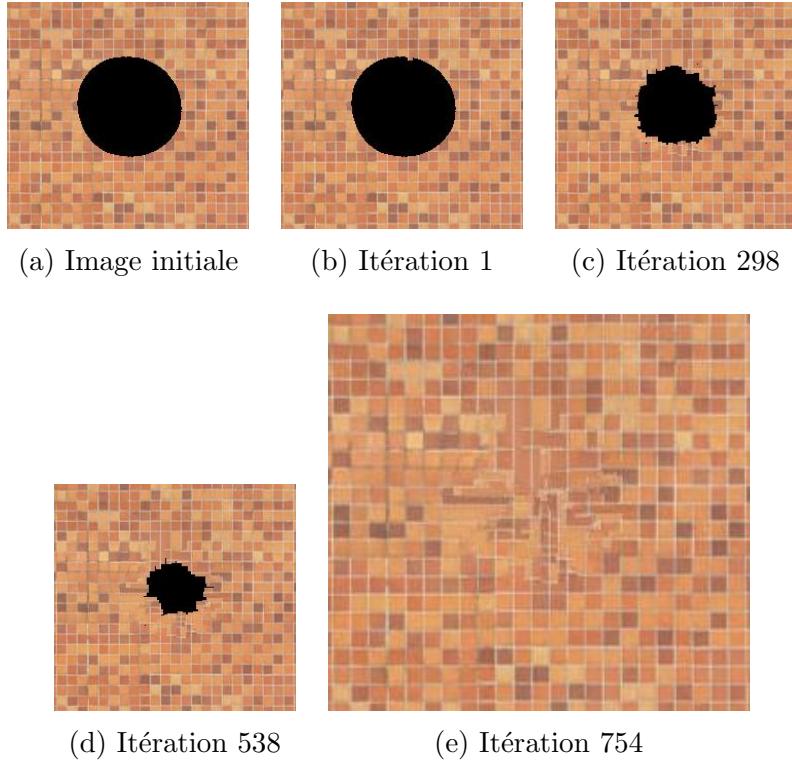


FIGURE 10 – Algorithme appliqué à l'image initiale avec un patch de taille 5.

L'algorithme offre des résultats raisonnables pour la complémentation de textures. Cependant, la taille du patch joue un rôle déterminant quant au bon remplissage. En effet, cette dernière doit être à peu près égale à la taille d'un motif élémentaire de la texture considérée.

2 – Résultats pour la propagation de structure :

2.1 – Exemple du poteau :

L'utilisation du terme de confiance seul fait que l'algorithme poursuit une trajectoire concentrique lors du remplissage. Cela donne de mauvais résultats lorsqu'on est amenés à propager des structures qui, intrinsèquement, nécessitent le terme où intervient le gradient).

Pour prouver cette idée, nous allons commencer par appliquer l'algorithme sur l'image d'un poteau sans terme de gradient :

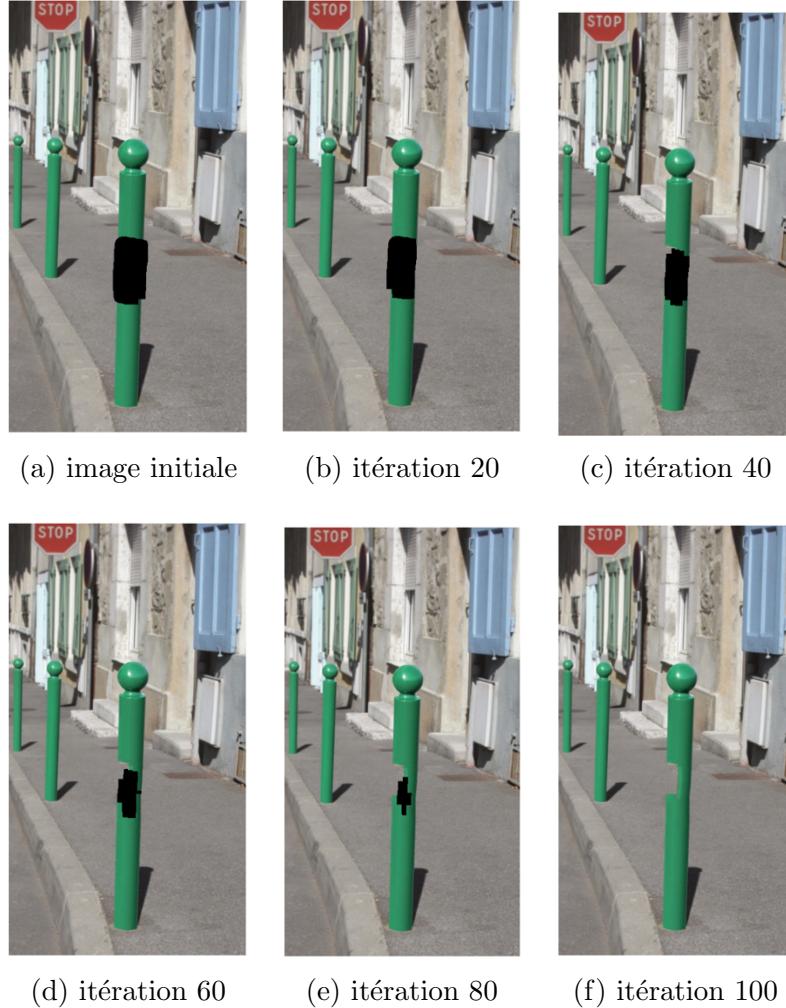
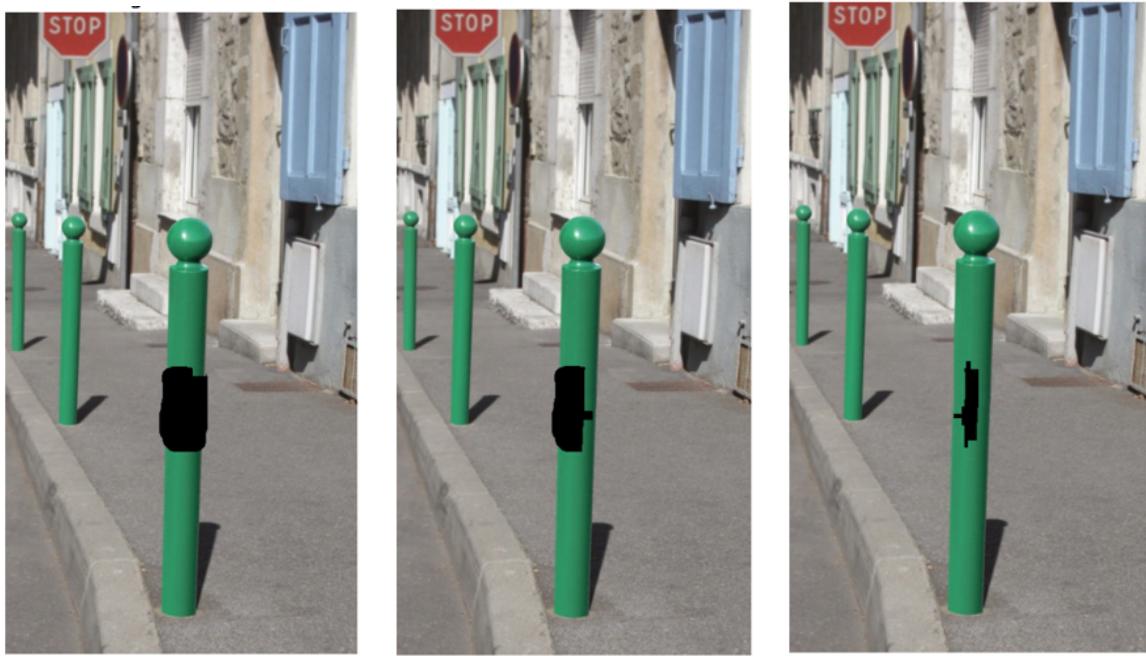


FIGURE 11 – Algorithme appliqué à l'image initiale avec un patch de taille 15 sans terme du gradient

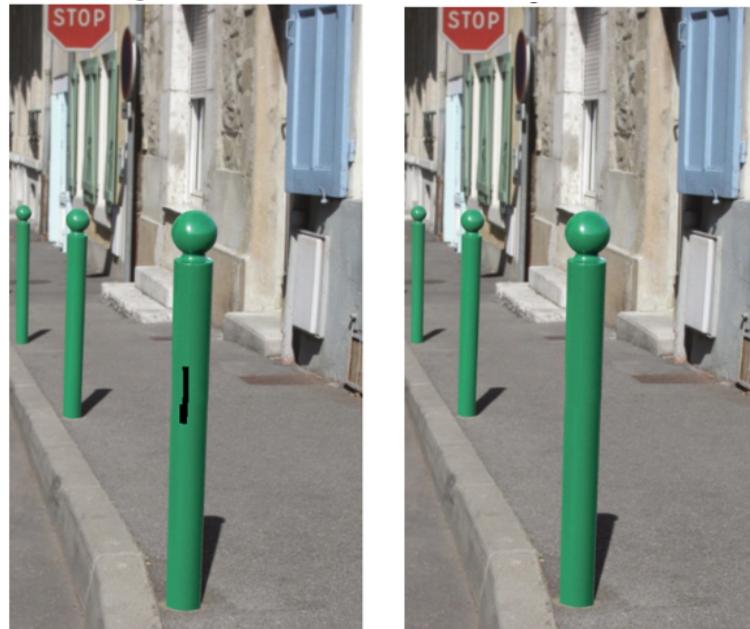
Nous remarquons que l'algorithme commet une erreur et détériore un bout du poteau. En revanche, en utilisant l'algorithme complet qui comporte le terme du gradient, l'ordre de remplissage des patchs change et commence d'abord par prolonger les deux lignes délimitant le poteau puis remplace à la fin la partie interne ce qui donne un résultat nettement meilleur :



(a) Image initiale

(b) Itération 20

(c) Itération 46



(d) Itération 61

(e) Itération 71

FIGURE 12 – Algorithme appliqu   l'image initiale avec un patch de taille 15

2.2 – Exemple du chat :

Pour l'image utilis   dans l'article, les r  sultats sont les suivants :



(a) image initiale

(b) itération 45

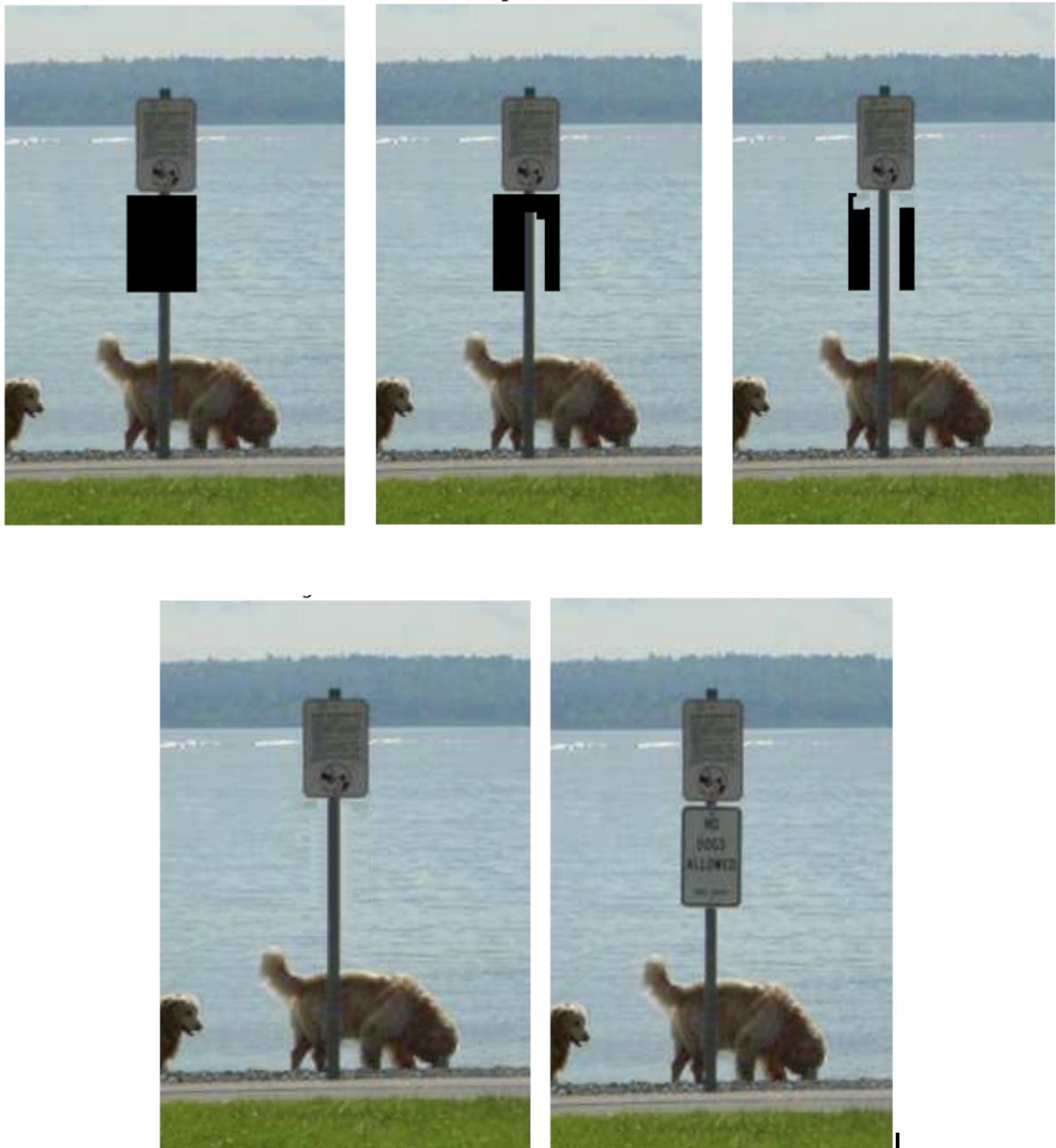
(c) itération 122



(d) itération 180 et résultat final

FIGURE 13 – Algorithme appliqué à l'image initiale avec un patche de taille 5

Les résultats de l'algorithme n'étaient pas très satisfaisants, car le terme de confiance empêchait le fait de privilégier suffisamment les points de continuité de la structure. Cela nous a donné l'idée d'appliquer cet algorithme sur la même image, en ne prenant en compte que le terme du gradient cette fois. Les résultats obtenus étaient très concluants :



(a) résultat final et image originale

FIGURE 14 – Algorithme appliqué à l'image initiale avec un patche de taille 5 sans terme de confiance

Les résultats de ce deuxième test nous ont permis de tirer la conclusion suivante : Lorsqu'il s'agit d'une image où l'inpainting concerne essentiellement une propagation de structure, il se peut que le terme du gradient seul suffit pour avoir de meilleurs résultats.

2.3 – Test des images de l'article et comparaison :

a - Sauteur à l'élastique : Pour l'image du plongeur les résultats sont :



(a) Image masquée



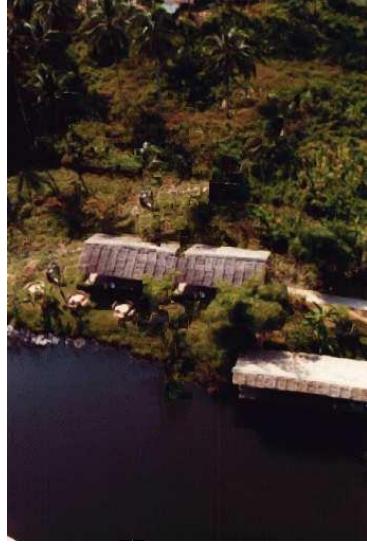
(b) Itération 70



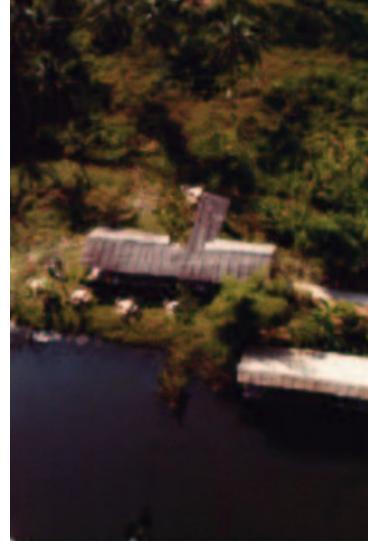
(c) Itération 120



(d) Itération 200



(e) Itération 295



(f) Résultat de l'article

FIGURE 15 – Algorithme appliqué à l'image masquée avec un patch de taille 17

L'algorithme a bien prolongé les deux contours parallèles de la maison en premier puis a continué le remplissage en alternant entre prolongation de structures et complétion de textures. Le résultat final est satisfaisant et proche de celui obtenu dans l'article.

b - île : Pour l'image de l'île les résultats étaient les suivants :

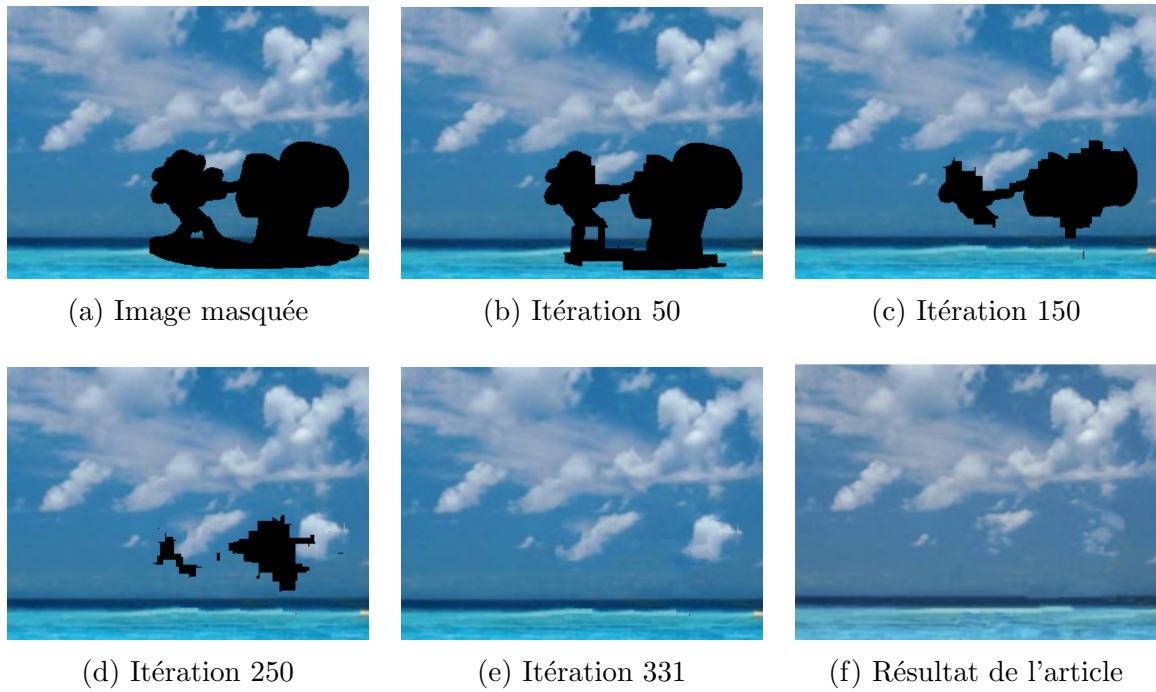
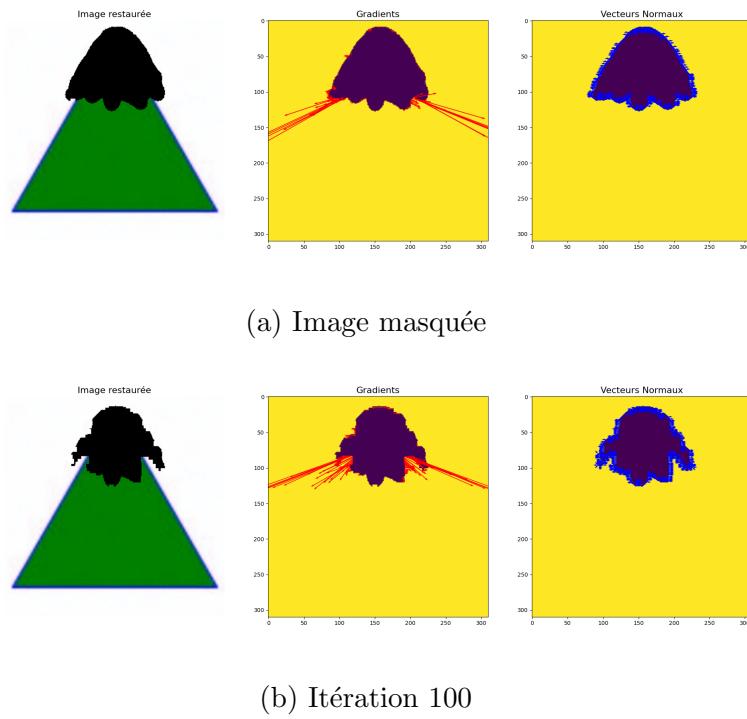
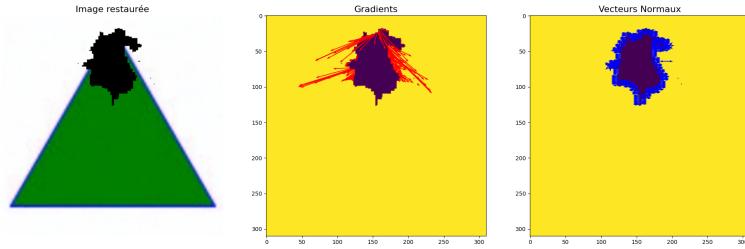


FIGURE 16 – Algorithme appliqué à l'image masquée avec un patch de taille 13

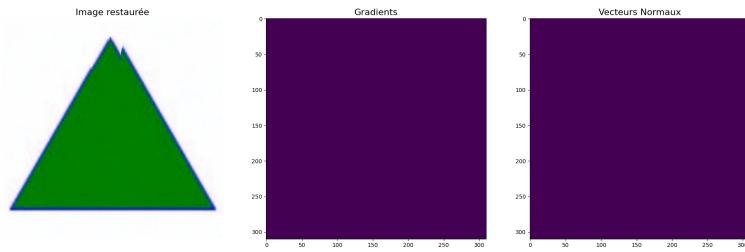
Le résultat obtenu est assez similaire à celui de l'article, l'information sur l'horizon a bien été restituée et l'algorithme a commencé par remplir cette zone là d'abord vu que c'est une structure linéaire où le gradient est important.

Triangle : Pour l'image du triangle les résultats sont :





(a) Itération 300



(b) Itération 565



(c) Résultat de
l'article

FIGURE 18 – Algorithme appliqué à l'image masquée avec un patch de taille 7

L'algorithme commence par remplir les deux arrêtes du triangle jusqu'à arriver au sommet puis termine par remplir l'intérieur du triangle. Il commet cependant une petite erreur au niveau de la pointe, erreur, qui, était prévisible vu que la forme du masque n'est pas symétrique d'une part, et que l'algorithme n'a aucune information sur le bout de la pointe d'autre part. Pour espérer obtenir un triangle parfait, il faudra laisser le bout de la pointe non masqué pour que l'algorithme puisse prolonger les deux arrêtes des deux cotés du bas vers le haut et du haut vers le bas.

2.4 – Autres tests :

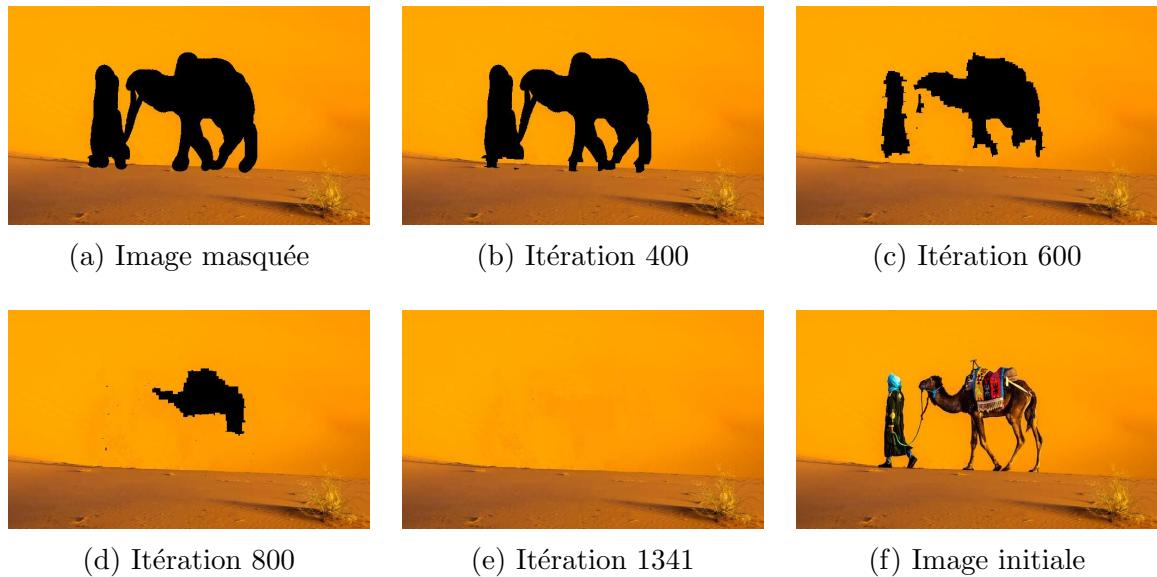


FIGURE 19 – Algorithme appliqué à l'image masquée avec un patch de taille 9

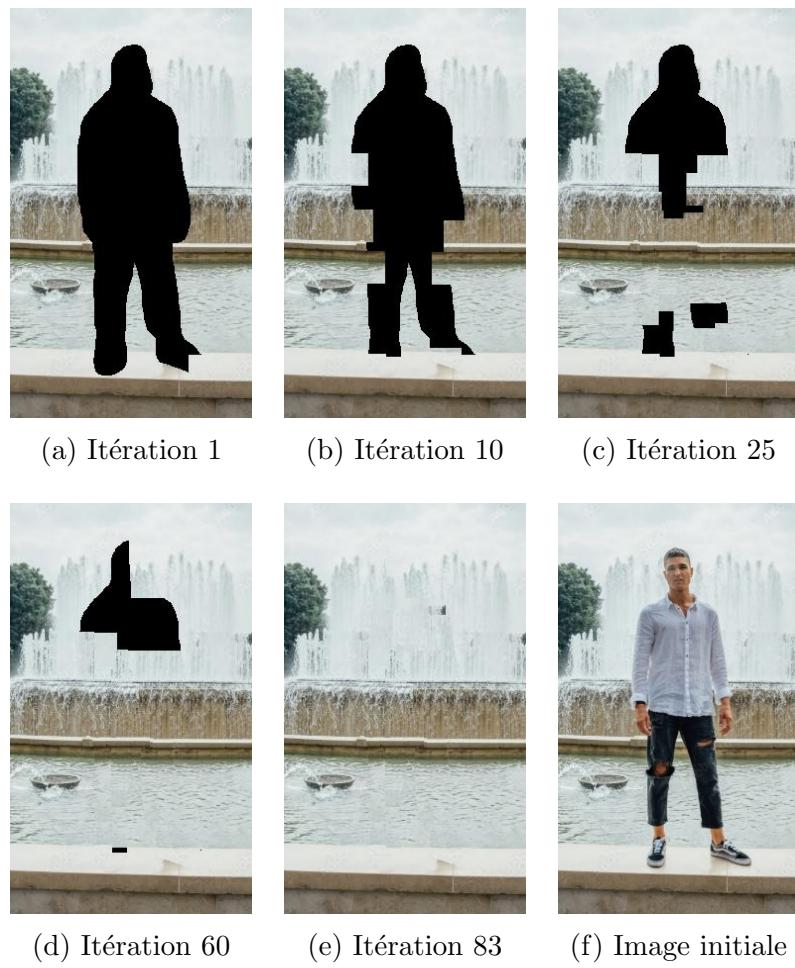


FIGURE 20 – Algorithme appliqué un patch de taille 33

Dans ces deux exemples on montre bien que l'algorithme favorise au début le remplissage des zones contenant des structures (donc où le gradient est fort). Le masque occupait dans ces deux cas des proportions importantes par rapport à la taille de l'image et pourtant les résultats sont corrects. Cependant, certains décalages au niveau de la marche sur laquelle était debout l'homme et celle de derrière sont survenus. Cela est difficilement corrigable car la manière avec laquelle notre algorithme remplit ne prend pas en compte les éléments constituant l'image dans leur globalité mais travaille localement patch par patch.

Limites et interprétations

1- Instabilité de la méthode :

Il suffit parfois de changer un tout petit peu le masque pour que le résultat soit totalement différent. En effet, dans certains cas où les parties voisines de l'objet à enlever ressemblent à cet objet, et qu'un objet similaire à ce dernier se trouve dans la photo, il pourrait le recréer. Tout commence par un patch qui est mal copié (le problème vient d'un patch dont la partie "aberrante" et occultée par le masque et qui n'entre donc pas en considération lors du calcul de distance et qui a été évoqué juste après dans le paragraphe "Améliorations possibles") cela cause l'algorithme à diverger petit à petit en accumulant les erreurs. Un exemple frappant nous a été donné par cette image, où à partir de l'omnbre de la première femme supprimée, sa voisine est recréée (à moitié). Il suffisait donc de réadapter un petit peu le masque pour éviter ce résultat.



FIGURE 21 – Résultat qui présente cette anomalie

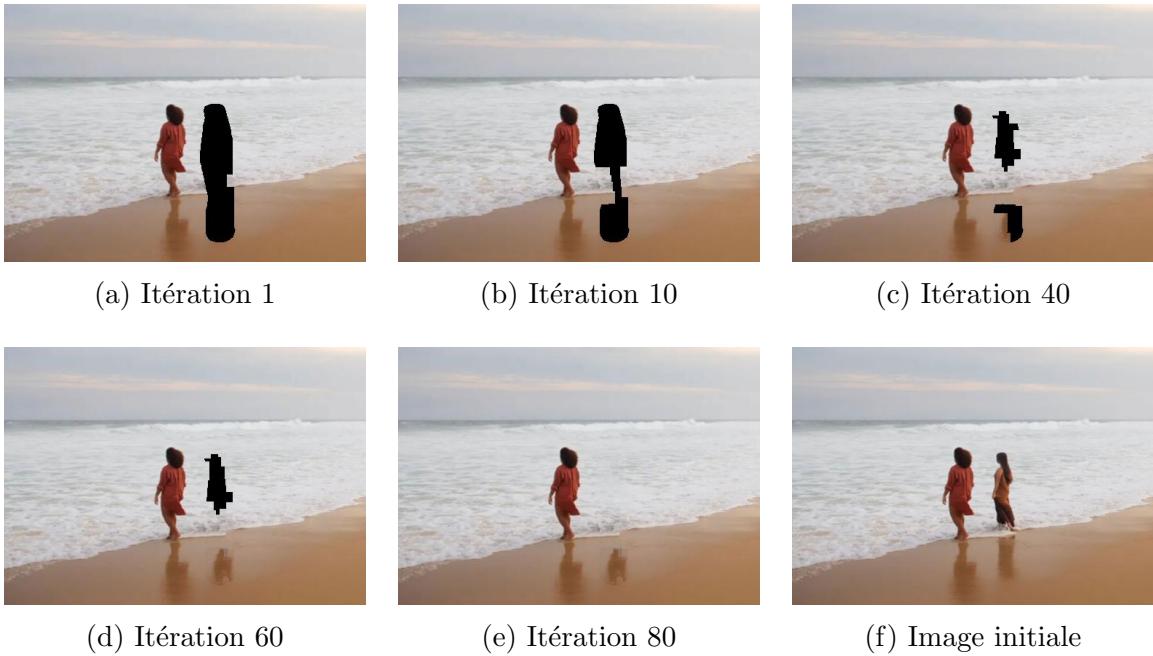


FIGURE 22 – Résultat après un léger changement du masque

2 - Limites d'un algorithme sans apprentissage :

Il se trouve des cas simples où notre algorithme donne des résultats très décevants, cela est intrinsèquement lié à l'absence d'approches basées sur l'apprentissage. Par exemple, si une image est détériorée au niveau de l'œil d'une personne. Et qu'on veut remplir cette zone. Notre algorithme ne saura jamais copier l'autre œil pour donner une apparence plausible d'une personne avec deux yeux.

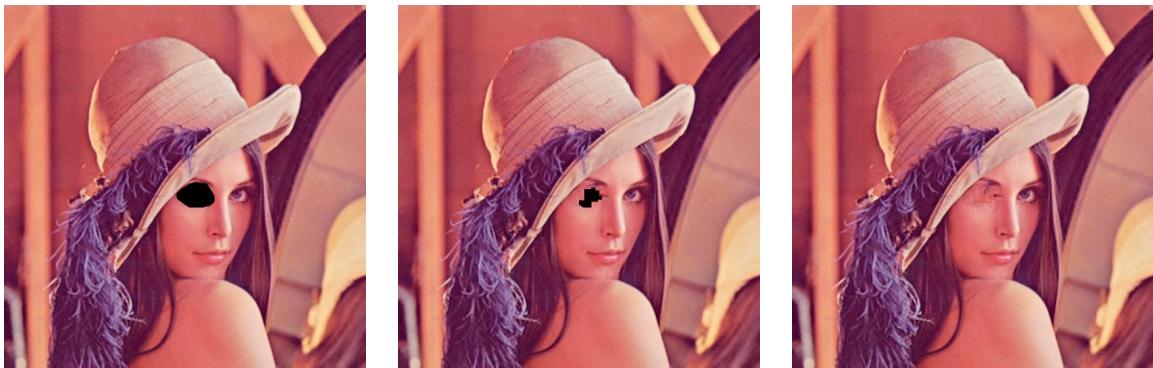
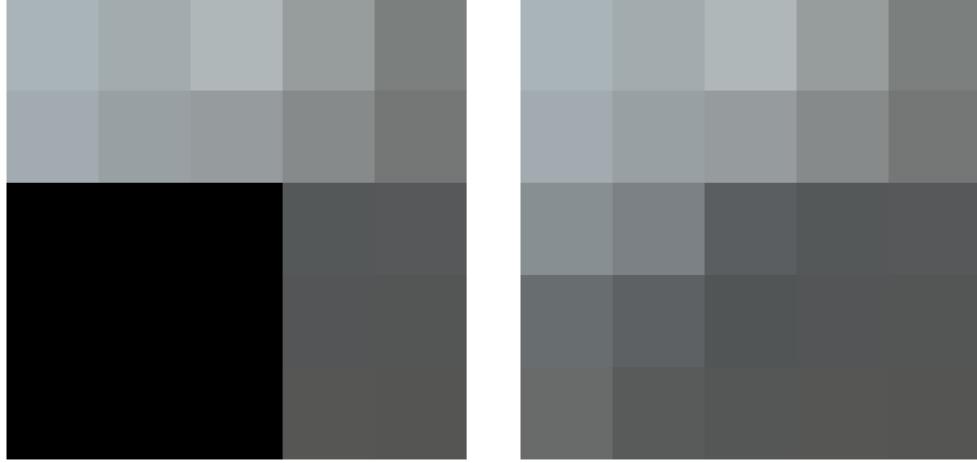


FIGURE 23 – Tentative d'inpainting de l'œil de Lena

Améliorations possibles

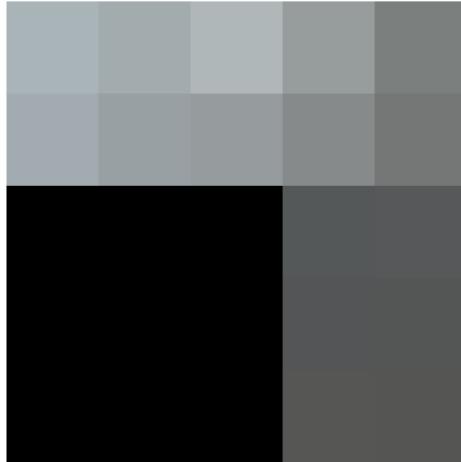
1 - Ajout d'une tolérance

En revenant aux résultats de la figure 7, on peut remarquer que les défauts ont commencé à apparaître lors de la 64 ième itération. En effet les patches mis en jeu sont les suivants :



(a) Patche prioritaire

(b) Patch de distance minimale



(c) Patch de distance minimale
masqué

On peut remarquer que lorsqu'on applique le masque au patch choisi on trouve le même patch initial, et donc évidemment la distance est minimale pour ce dernier. Or on peut constater que la zone masquée, et qui fait la différence entre les patchs, n'est pas prise en considération. Dans ce cas, la zone masquée n'avait pas la bonne couleur pour garantir la continuité de notre structure. C'est en effet une partie de la plaque du dessus qui a commencé à être copiée au lieu de prendre un patch où la partie masquée est une partie du ciel. Ce cas nous montre les limites de notre approche de comparaison.

En partant de l'hypothèse que plus le patch choisi est proche plus il a de chances d'être le patch idéal, on peut faire un compromis en termes de distance. On met en place une valeur de tolérance qui nous permet de dépasser la distance courante à condition que

le patch que l'on teste soit plus proche de celui sélectionné. Pour ce, nous avons défini une distance spatiale qui est simplement la distance euclidienne entre le centre du patch prioritaire et celui de test.

Ainsi, pour une tolérance de 5% en distance couleur le résultat est le suivant :

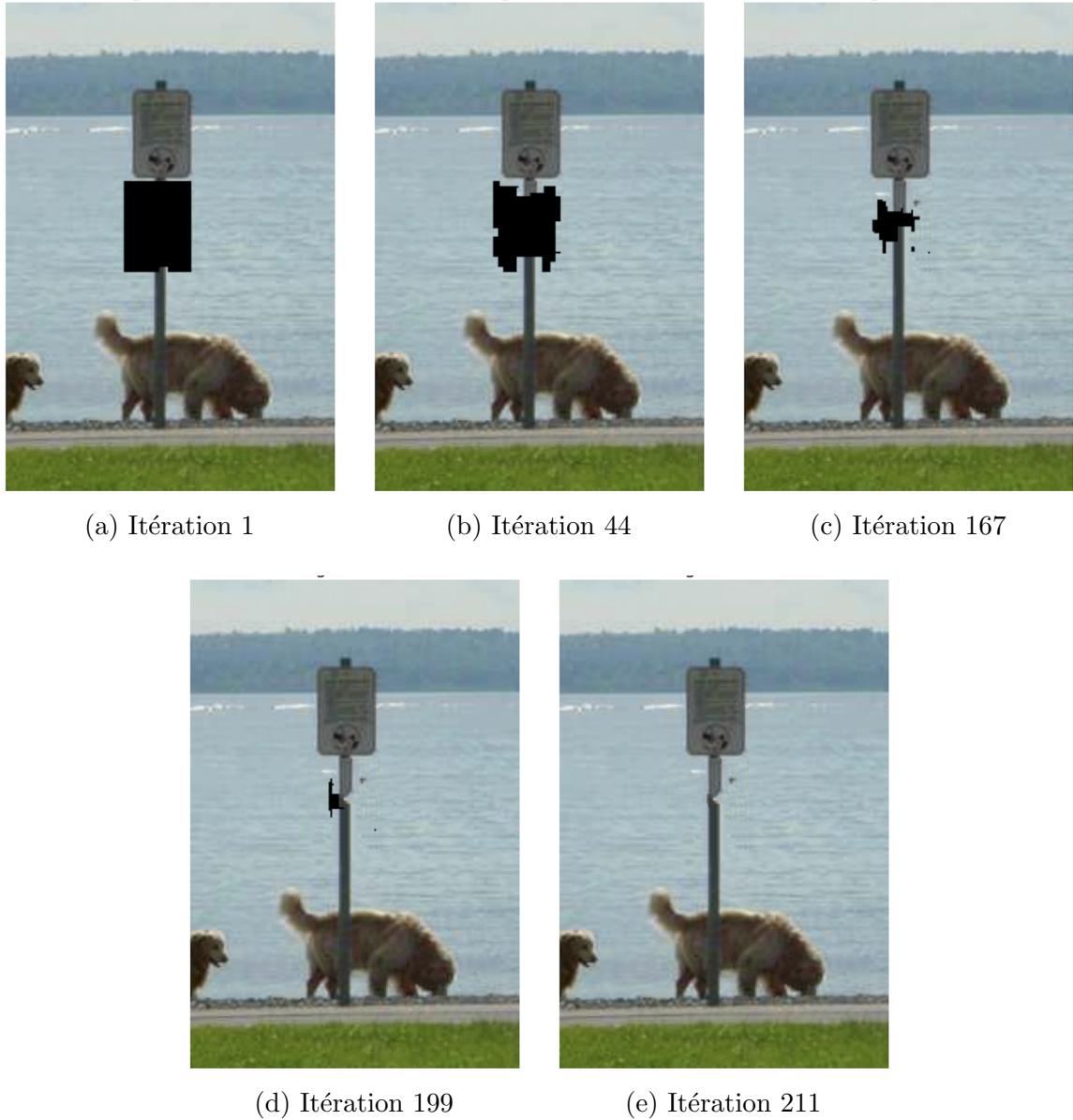


FIGURE 25 – Algorithme appliqué à l'image initiale avec un patch de taille 5 en faisant une marge de tolérance.

Conclusion

En conclusion, l'algorithme d'inpainting basé sur des exemples donne des résultats convaincants pour les images présentant des textures homogènes ou des structures bien définies. Cependant, ses limites deviennent évidentes dans des contextes plus complexes. Les améliorations suggérées pourraient significativement accroître sa robustesse et son

efficacité dans divers scénarios, bien qu'il reste impossible d'égaler les performances des approches d'apprentissage, qui représentent l'état de l'art dans ce domaine.

Références

- [1] A. Criminisi, P. Perez and K. Toyama, "Object removal by exemplar-based inpainting," 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., Madison, WI, USA, 2003, pp. II-II, doi : 10.1109/CVPR.2003.1211538.