RAPPORT | README

Projet Analyse d'image

28 AVRIL 2023

CORROLLER NATHAN MARC ANGELINE
L3 Informatique

Table des matières

Répartition des projets

Pour ce qui a été de la répartition des projets. Je (Nathan) me suis occupé de la partie analyse d'images et Angéline s'est occupée de la partie synthèse d'images. Ce readMe ne concernera uniquement que la partie synthèse d'image. Ci-dessous vous pourrez retrouver nos identifiants étudiants :

Corroller Nathan : 12208035

Marc Angéline : 12207337

Les Démarches utilisées

Recherche de technologie

Dans un premier temps, le but du projet était de créer un masque permettant d'isoler les pièces du fond orange. Je devais donc tout d'abord réaliser l'extraction des pièces à partir de la photo de base que vous pourrez retrouver en annexe.

Je me suis donc documenté sur internet sur les technologies d'extraction et sur les différents algorithmes déjà existants et je suis tombé sur la fonction d'open-cv « calcBackProject». En temps normal, cette fonction est utilisée pour extraire des objets, des personnes ou plus généralement des éléments qui se trouvent sur des fonds verts dans le domaine du cinéma.

L'espace de Couleur

Après avoir trouvé la fonction que j'allais utiliser, il a fallu que je trouve un espace de couleur dans lequel il était simple de faire la différence entre la couleur des pièces et la couleur du fond. Pour cette étape, il n'y a pas eu de recherche particulière, uniquement des essais à la suite. J'ai tout d'abord essayé de convertir mon image en niveau de gris, en HSV, en RGB, en LUV pour finalement arriver à un résultat plutôt satisfaisant, le LAB. (L'image de base étant en BGR)

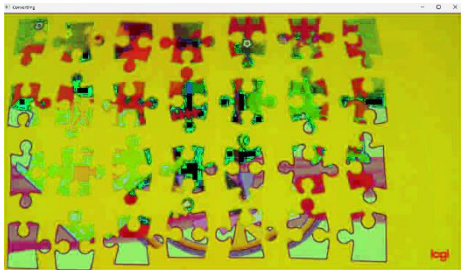


Figure 1 conversion BGR -> HSV

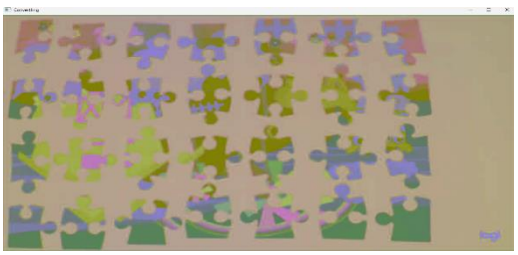


Figure 1 conversion BGR -> LAB

Figure 2 conversion BGR -> RGB

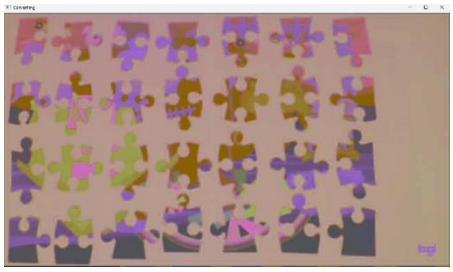


Figure 3 conversion BGR -> LUV

Le choix de l'espace de couleur est plutôt important pour cette fonction car celle-ci calcule l'histogramme de la région que l'on cherche à enlever (La ROI : region of interest). Pour nous, cette ROI était l'objet suivant :

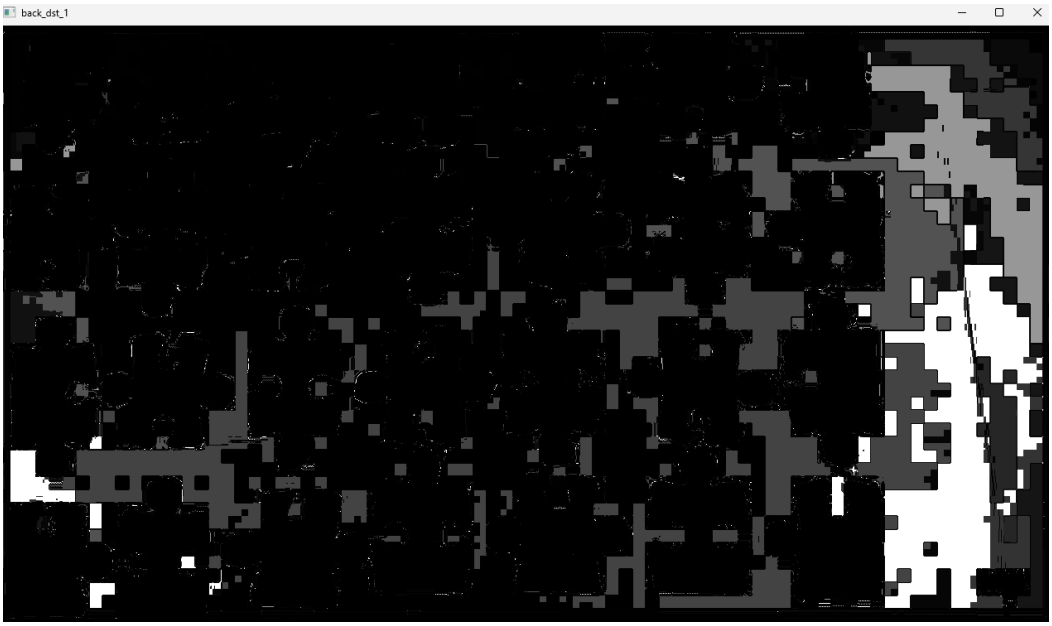
Figure 4 ROI en BGR

Figure 3 ROI en LAB

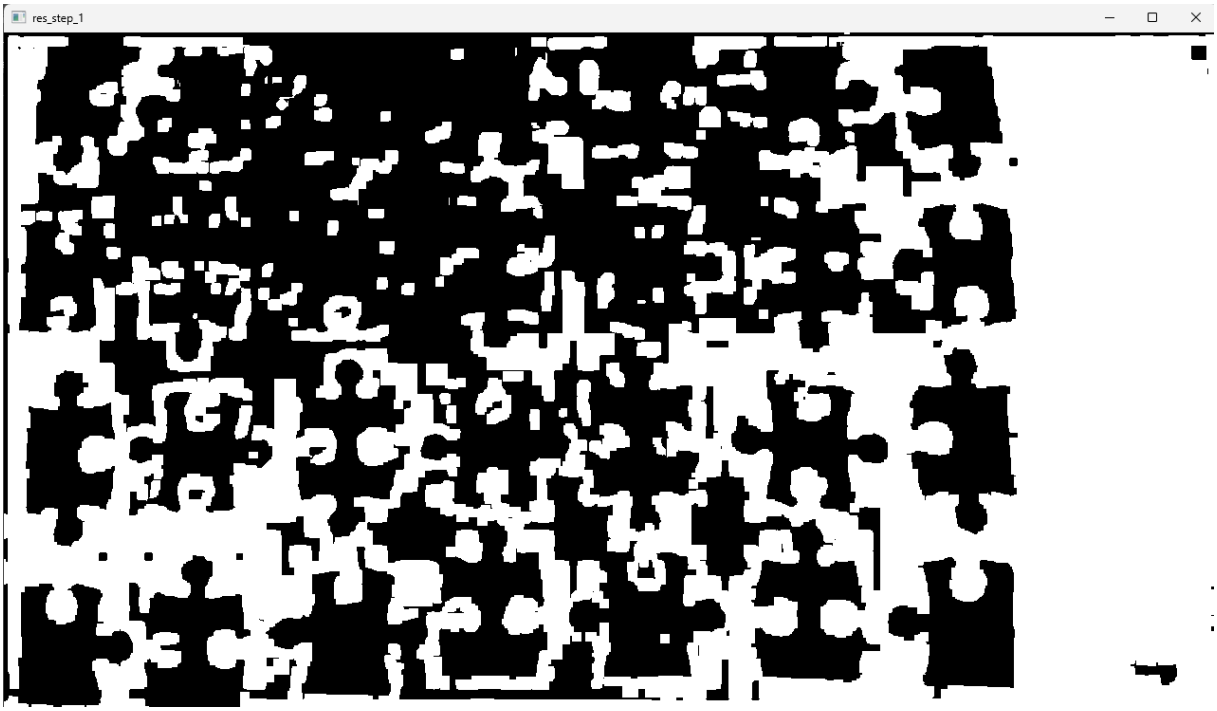
Cette region of interest est dans notre cas la partie à droite de l'image où il ne se trouve rien à part le fond. La fonction calcBackProject va donc prendre en paramètre l'histogramme normalisé de cet objet pour ensuite faire des calculs de probabilité et identifier sur une image, passée en paramètre, la probabilité que le pixel courant de l'image appartienne à la ROI.

Mise en place de la BackProjection et nettoyage

Le premier résultat obtenu est le suivant :

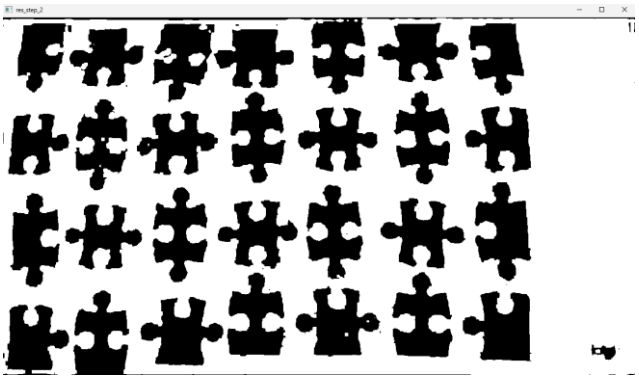


On peut remarquer que le résultat est inexploitable sous cette forme. Cependant, on aperçoit déjà quelques pièces de puzzle. Pour nettoyer et binariser cette image (car à la fin nous voulons uniquement du blanc et du noir pour ce qui est ou non une pièce de puzzle), nous passons d'abord un filtre appelé « boxFilter » qui va lisser un peu les pixels très marqués en faisant prendre à l'ancrage du filtre (le pixel au centre) la moyenne de ce qu'il se trouve dans le noyau. L'un des défauts de la BackProjection est de créer des pixels un peu trop marqués comme on peut le voir ci-dessus. Après avoir lissé l'image on va donc la binariser avec un seuillage à 20 et on va obtenir ce résultat :

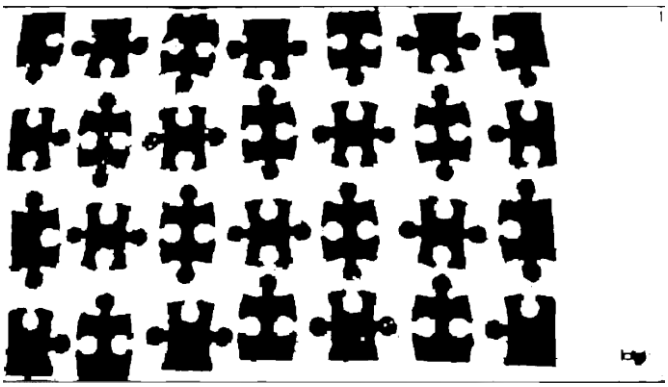


On peut voir que c'est tout de suite un peu plus propre et exploitable. Nous allons donc pour finaliser le masque, réitérer le processus quatre fois mais en changeant l'objet. Le nouvel objet à chaque itération sera le masque obtenu à la fin du seuillage précédent avec l'histogramme de l'image de base converti en LAB. Voici donc l'évolution du masque au fil des itérations de calcul d'histogramme, de calcBackProject, d'application du filtre et de seuillage.

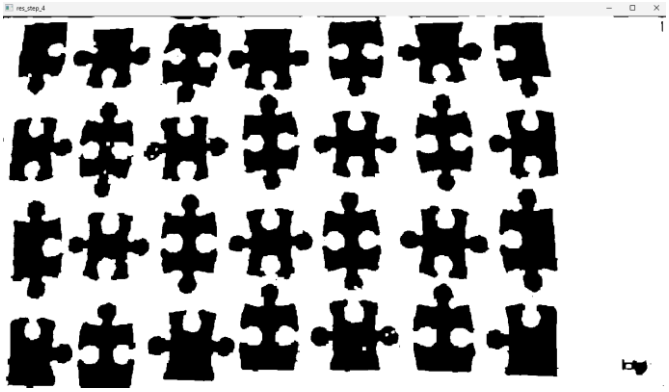
Résultat de la 2nd itération :



Résultat de la 3e itération :



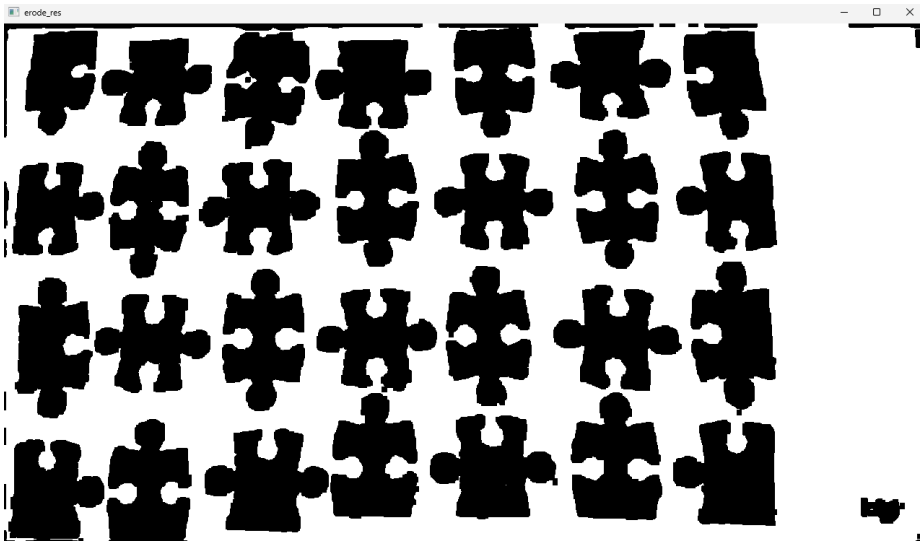
Résultat de la 4e itération :



Les deux dernières itérations étant uniquement pour des détails du masque.

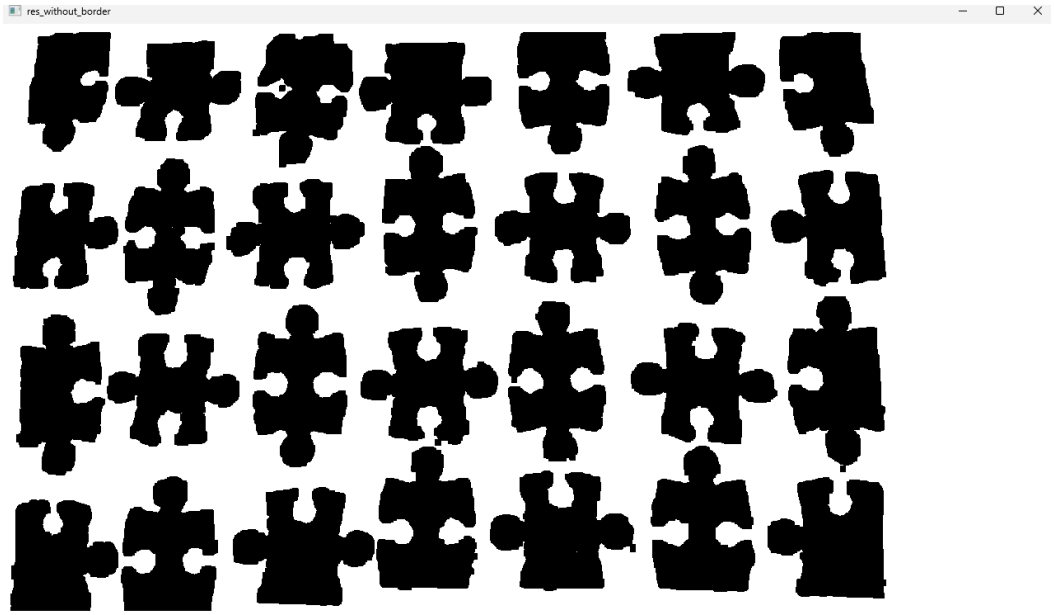
Une fois ce masque obtenu, on peut remarquer des trous dans certaines pièces. Pour les combler on va réaliser une érosion pour les combler. En temps normal nous aurions dû faire une dilatation mais dans notre cas les couleurs sont inversées (zone que l'on veut extraire, c'est-à-dire les pièces en noir et le fond en blanc).

Résultat après érosion :



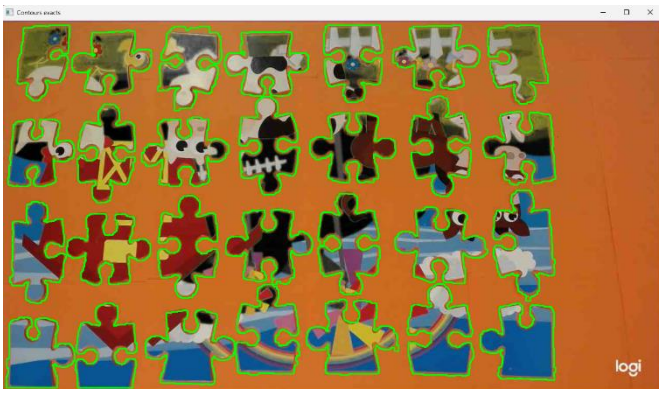
Notre masque est presque fini, un détail étant encore persistant. On retrouve encore des bordures noires sur le contour de l'image ainsi que le logo de l'image en bas alors que ce ne sont pas des pièces. On va donc tricher un peu et mettre quelques pixels à blanc sur tout le contour de l'image et mettre le logo de droite également en blanc pour n'avoir plus que les pièces de puzzle.

Masque final :



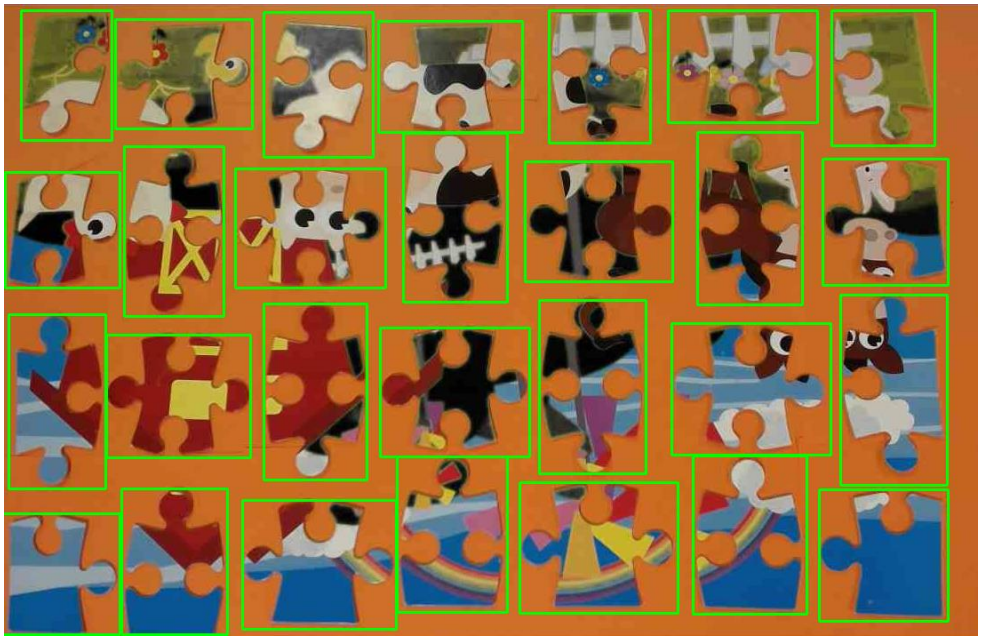
Recherche des contours

La recherche des contours se fait très simplement. Il suffit juste d'appeler la méthode « cv2.findContours » et de lui passer l'image en paramètre pour que la fonction retourne un tableau contenant pour chaque contour un tableau de point. Cependant lorsque j'ai regardé la taille du tableau obtenu pour savoir si elle correspondait bien au nombre de pièce (1 contour par pièce donc 28, j'avais un tableau de 37 contours) je me suis rendu compte que ce n'était pas bon. Des artefacts résiduels du masque devaient encore être présents. J'ai donc filtré tous les contours obtenus. Dans une nouvelle liste, j'ai ajouté chaque contour qui étaient composé de plus de 150 points. Les mauvais contours (ceux des artefacts) avaient entre 10 et 30 points dans leur contour et au minimum une pièce possède 170 points. J'ai donc mis un seuil de décision à 150 pour être large et être sûr que je n'avais après ça que des pièces de puzzle. Une fois cette manipulation réalisée, il a juste suffi d'appeler la méthode « cv2.drawContours » et de lui donner l'image de base en paramètre pour obtenir ce résultat :



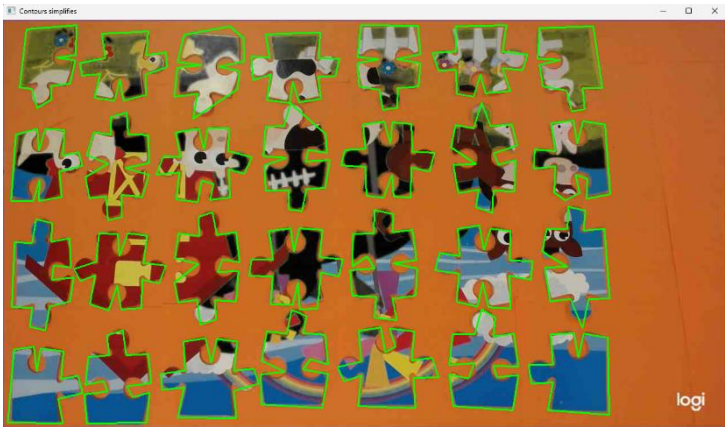
Isolation de chaque pièce

Nous possédons désormais des coordonnées qui définissent nos pièces de puzzle. Afin de les isoler les unes des autres nous allons rechercher dans chaque contour les cordonnées minimum en x et en y ainsi que les coordonnées maximums en x et en y. Pour ce faire, nous allons juste itérer sur les tableaux de points pour trouver les valeurs qui nous intéressent. Une fois ces valeurs obtenues, il ne nous reste plus qu'à dessiner des rectangles avec les max et min obtenus. Voici le résultat que j'ai obtenu :



Simplification des contours

Afin de par la suite déterminer quelle pièce est de quel type (un bord en bas, à gauche, à droite, une pièce commune, etc.), nous devons simplifier les contours que nous avons obtenus. Pour ceci, nous allons utiliser la méthode « cv2.arcLength » et la méthode « cv2.approxPolyDP ». Ces deux méthodes combinées vont nous permettre d'approximer nos contours en créant un polygone avec des segments plus ou moins grands et plus ou moins ressemblants aux contours. Ces deux paramètres varient en fonction d'un paramètre epsilon de la fonction « cv2. approxPolyDP ». En multipliant le résultat de « cv2.arcLength » par 0.018 (ce qui nous donne notre epsilon). On obtient ce résultat :



Justification des choix d'algorithmes

Les algorithmes qui ont été choisis dans ce projet ont été choisis après des recherches sur la documentation de la bibliothèque open-cv. Cette bibliothèque propose une bonne documentation et c'est ce qui m'a orienté, en plus de recherches sur StackOverflow ou encore YouTube, sur mes choix d'algorithmes.

Paramétrage d'algorithme

Tous les algorithmes utilisés, que ce soit la « calcBackProject », l'epsilon avec « cv2. approxPolyDP », la taille de noyaux des filtres et autres ont été dans un premier temps choisi en fonction de la documentation. Et dans un second temps en y allant un peu à l'aveugle et en essayant à la chaine différentes variantes, ces paramètres ont ainsi évolué pour donner le résultat que j'ai obtenu. Il n'y a donc pas de réelle stratégie mais uniquement des essais en continu qui ont fait varier le résultat de ce projet petit à petit pour donner les résultats que je vous ai présentés.

Méthodes utilisées sans succès

Les seules méthodes qui ont été utilisées sans succès sont les différentes conversions en différents espaces de couleurs (rgb, luv, niveau de gris …) ainsi que les différents filtres essayés avant d'arriver à trouver le box filter qui me permet de lisser les pixels après une backProjection (quelques filtres essayés : blur, gaussianBlur, medianBlur, …)

Annexes

Annexe 1 :

Figure 5: Image de départ