DÉTECTION DE PLACES PARKING

PROJET : TRAITEMENT D'IMAGES AVANCÉS

03/2021 - P.BONNIN

Adrien CHAHINIAN, Lilian FAVRE GARCIA, Julie NEIGE

SOMMAIRE

PRÉSENTATION DU PROJET

Objectif : Détection de place de parking, détecte si la place est prise ou non

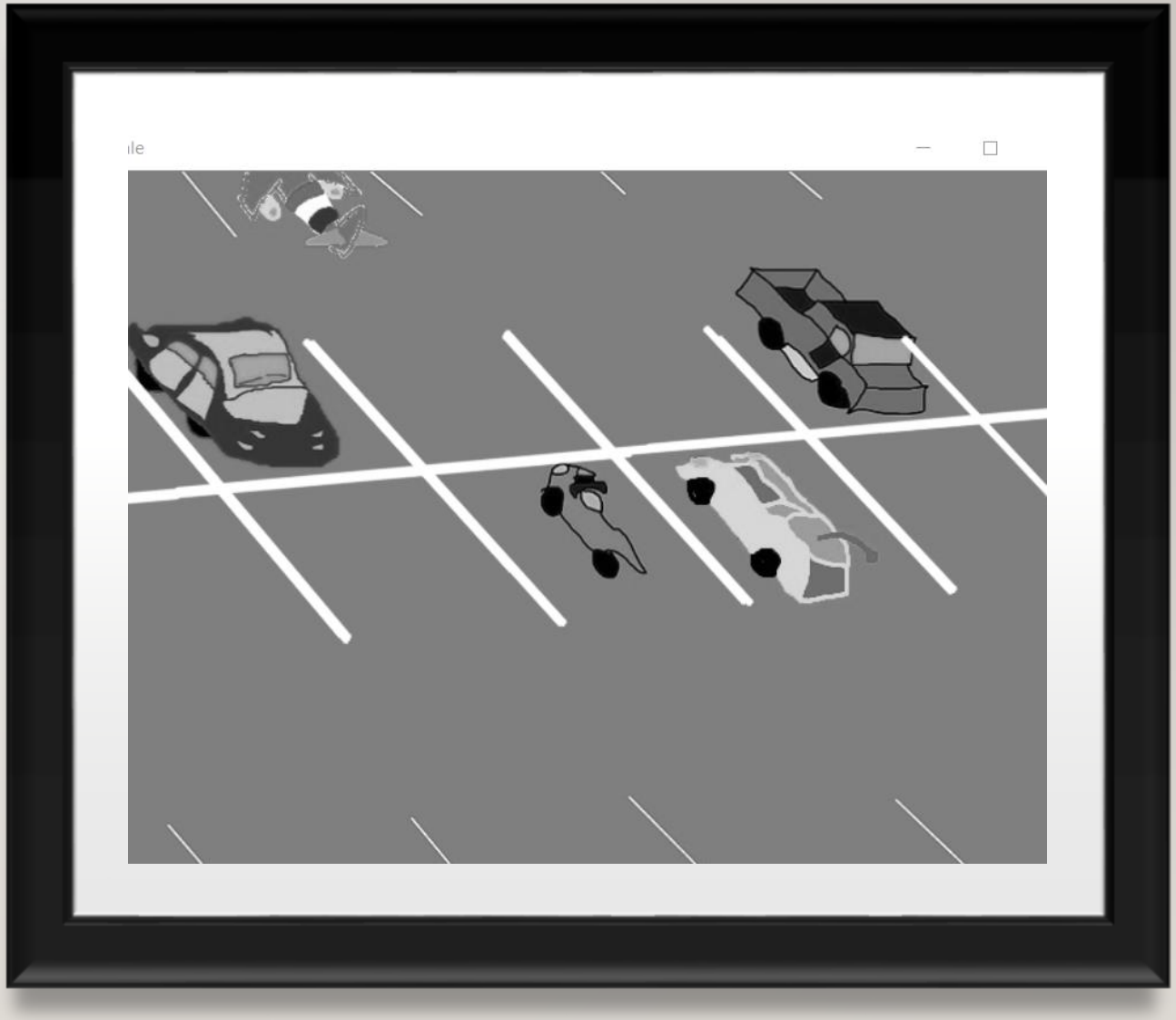
Environnement : C++ et Visual Studio

ETAPE 0

ON CHOISI UNE IMAGE A ANALYSER

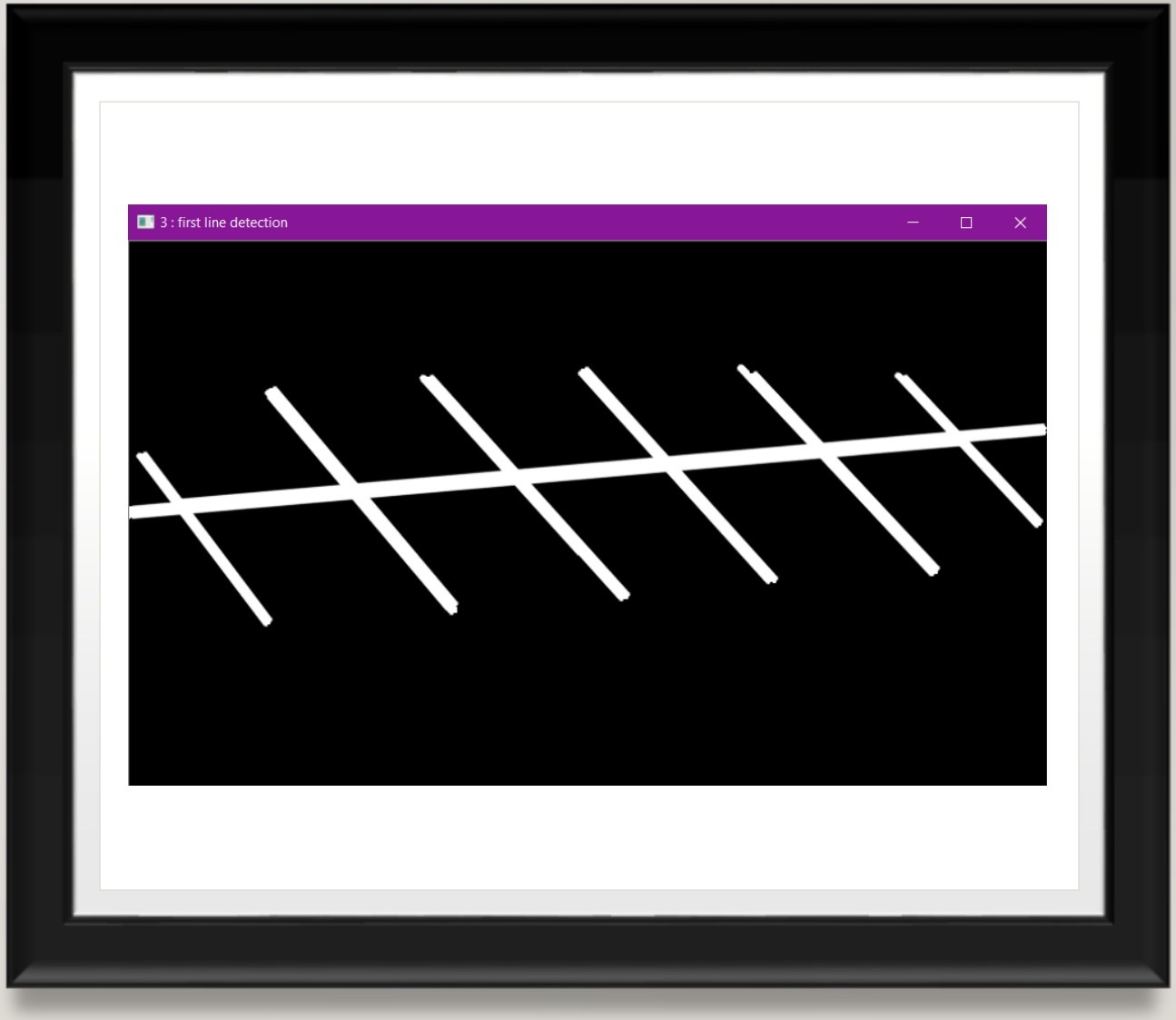
ETAPE 1

- On transforme l'image en niveaux de gris pour réaliser ensuite une détection de Canny.



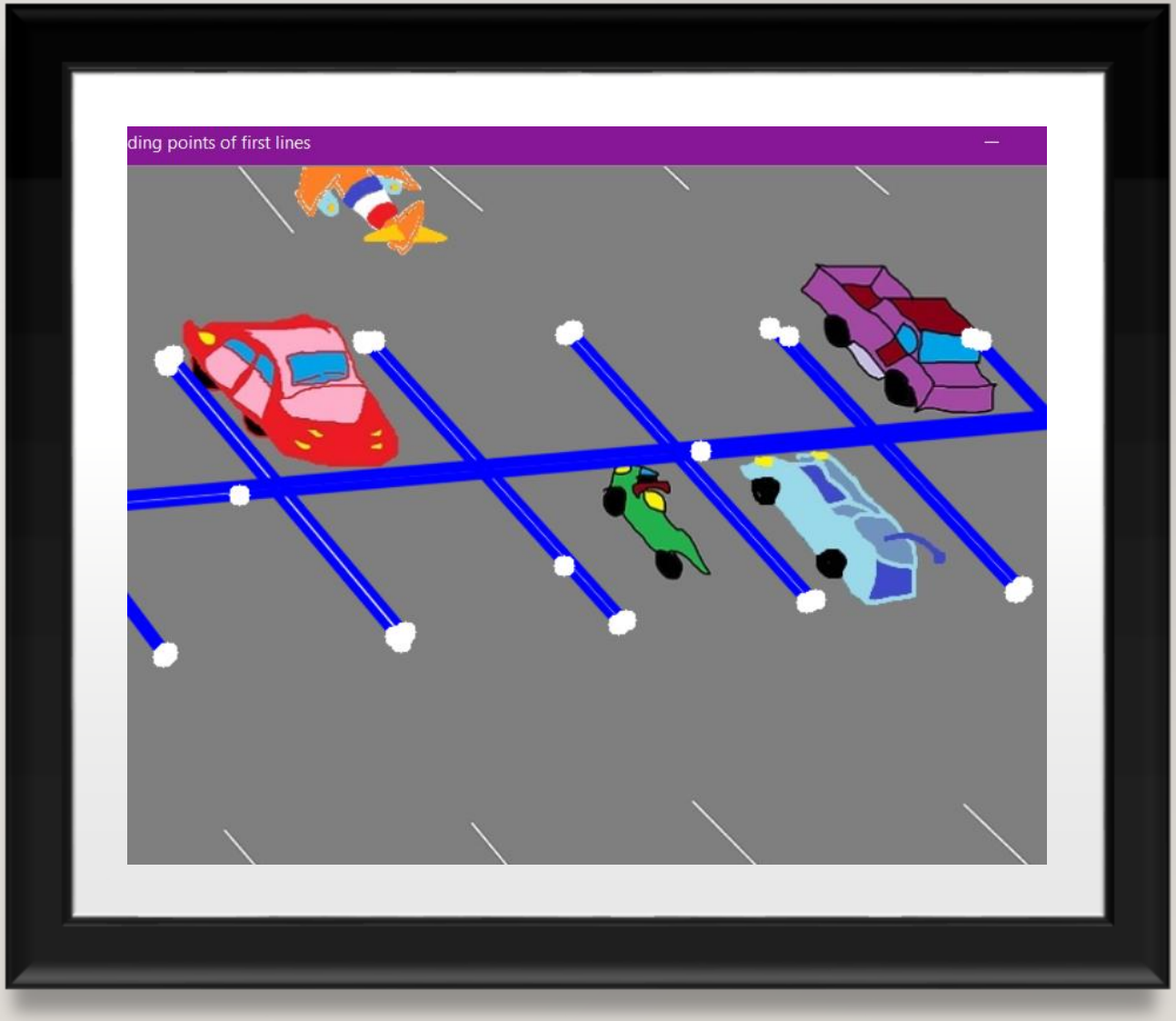
ETAPE 2

- Après les contours, s'ensuit une première détection de ligne grâce HoughP. Les lignes détectées sont donc multiples, redondantes voire pas réellement sur des lignes. De plus, Canny peut laisser de nombreux pixels blancs un peu partout.



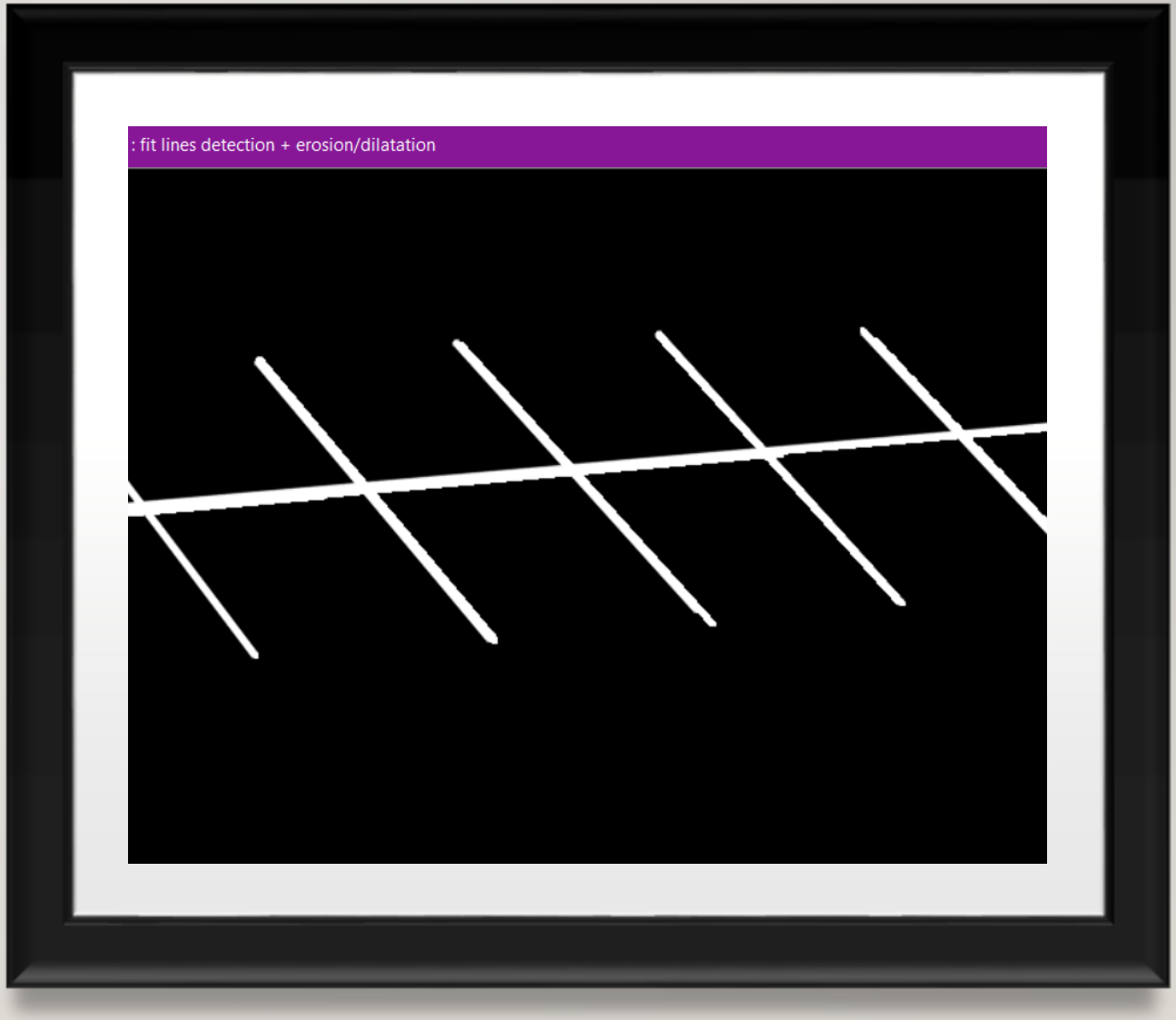
ETAPE 3

- On affiche les lignes précédentes sur l'image de Canny puis on réalise une ouverture pour supprimer de potentiels pixels blancs aléatoires et autre ligne détectée très fines (1px de large). On réaliser une deuxième détection de ligne afin de ne garder que des lignes "parfaites". Il y en a trop et potentiellement aléatoires, mais on réalise un tri à l'étape suivante



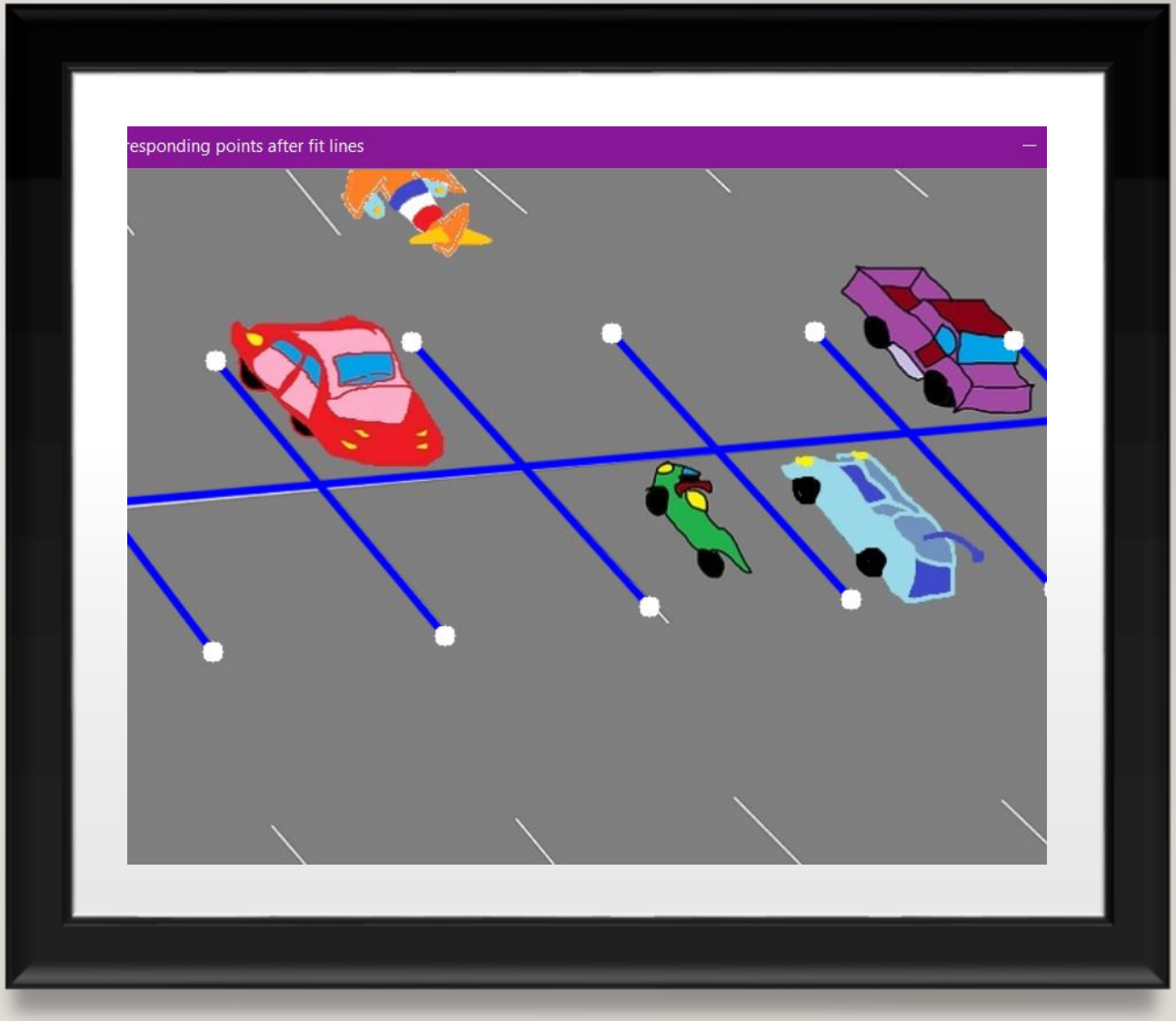
ETAPE 4

- On réalise donc ce qui est la partie la plus complexe du programme : comparer les lignes et les traiter afin d'en supprimer certaines, fusionner d'autres et garder les meilleures afin de ne garder qu'une vraie ligne détectée par ligne de peinture de parking



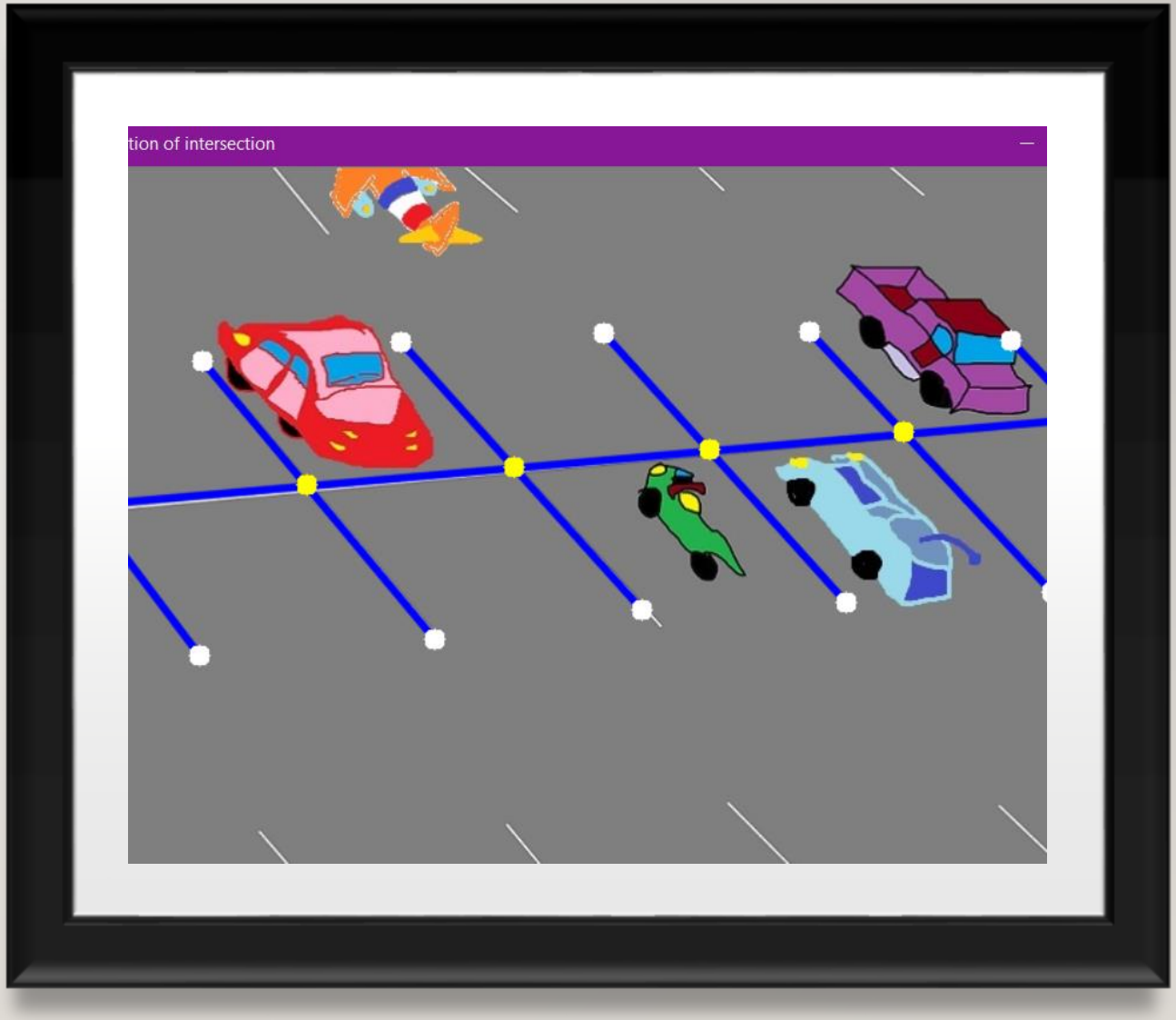
ETAPE 5

- A cette étape, on a donc normalement qu'une ligne détectée par vraie ligne. On récupère les points de fin et de début des dernières pour ensuite retrouver leurs intersections



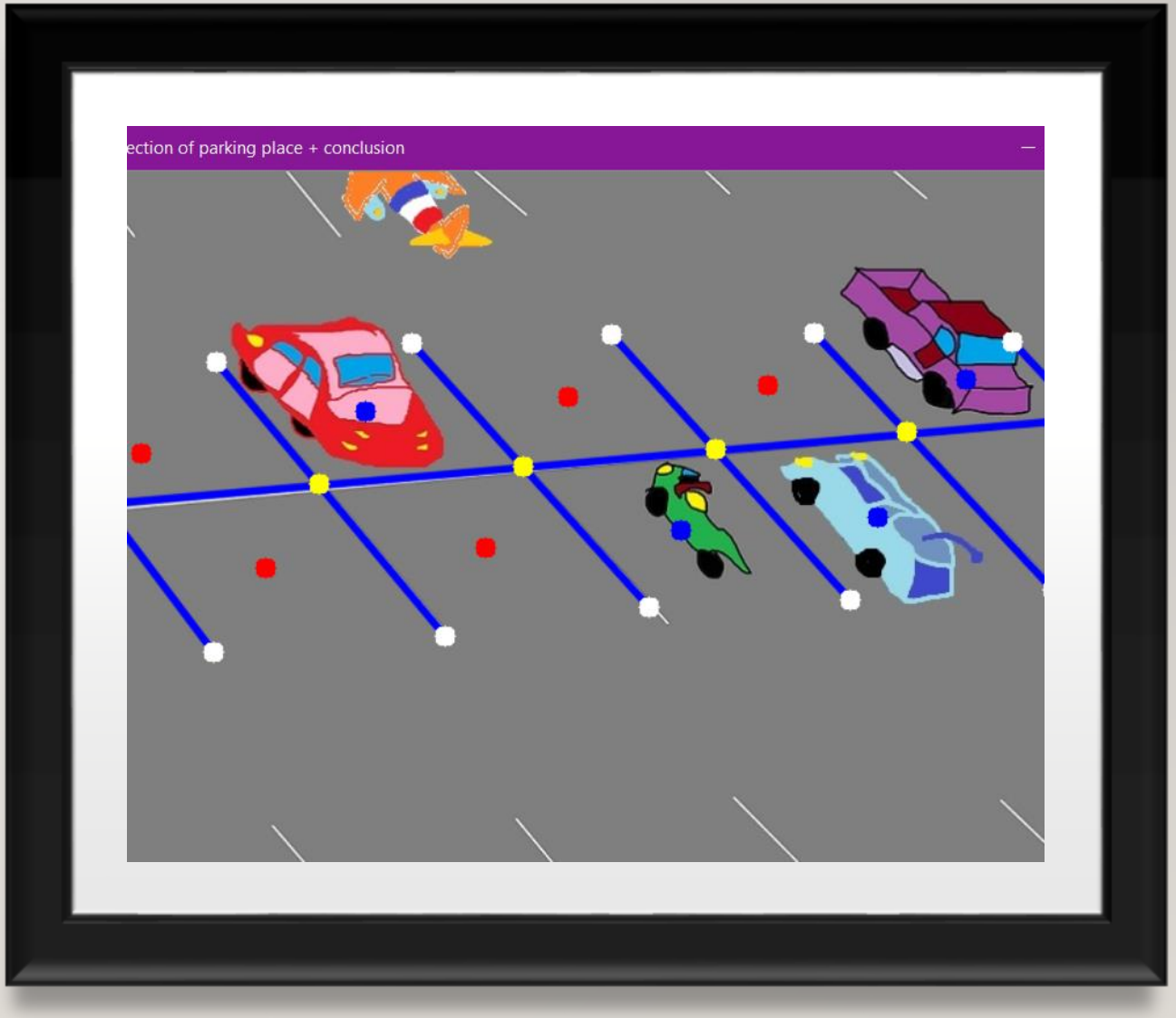
ETAPE 6

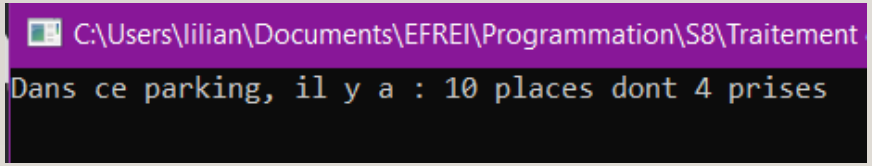
- Les intersections trouvées, il faut les trier. Visuellement il n'y a pas de différences mais dans le code, cela se résume à trier les points de gauche à droite pour avoir un tableau des points d'intersections dans l'ordre de la ligne



ETAPE 7

- Grâce aux 4 points de chaque place, on trouve le milieu. Puis on compare la couleur des pixels voisins au pixel du centre à la couleur de fond. Si c'est la même, il n'y a pas de voiture. Sinon, la place est prise.





LES DIFFICULTÉS RENCONTRÉES

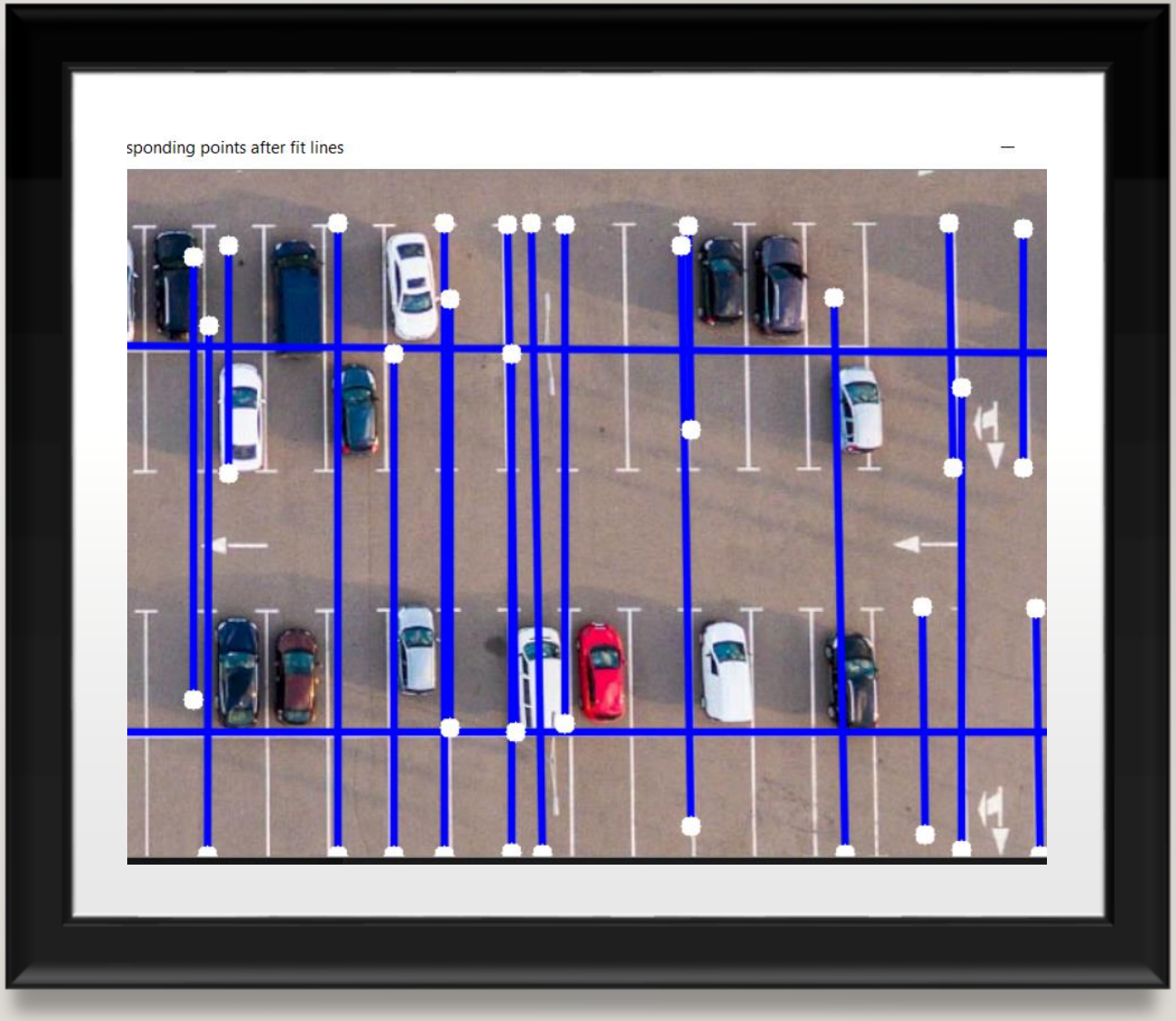
- Installation OpenCV:
  - Nous avons eu des difficultés au début
  - Passage à Visual Studio :

Gain de temps

Installation plus simple

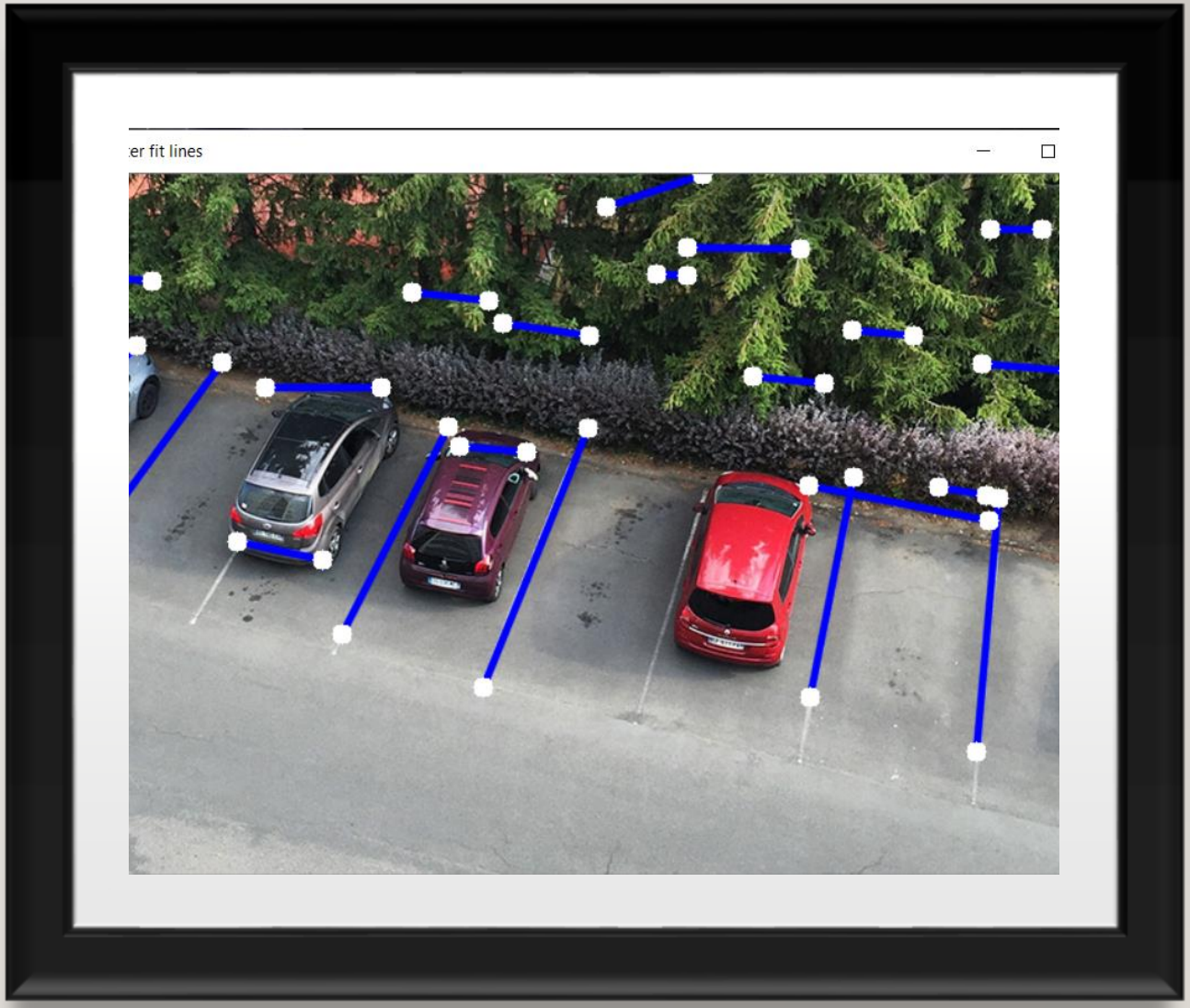
LES DIFFICULTÉS RENCONTRÉES

- Configuration différente :
  - Vue aérienne avec plusieurs ligne centrale
  - Des lignes parfois moins précises
  - Plus de lignes à traiter de tailles différentes
- La plus grande difficulté est de bien détecter les lignes. Il faut pour cela souvent changer les paramètres de HoughP d'OpenCV si on change d'image.



LES DIFFICULTÉS RENCONTRÉES

- Configuration différente :
  - Pas de ligne centrale, ce qui nous empêche de faire l'étape des points d'intersection et la suite.
  - Changement des paramètres de HoughLinesP pour avoir le résultat suivant. (ce qui ne fonctionne pas dans notre cas parfait)
  - HoughLinesP(FirstEdges,all_lines, 1, CV_PI / 180,100,60, 30);
- Effectuer un tri des lignes efficace après leur détection est la partie la plus complexe

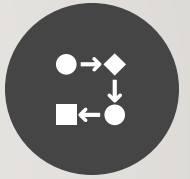


POUR ALLER PLUS LOIN

- Augmenter les configurations possibles:
  - Plusieurs lignes centrales
  - Aucune ligne centrale
  - Des lignes liées à seulement une place
- Améliorer notre traitement des lignes pour traiter des images réelles
- Améliorer la détection des lignes (soit trouver des coefficients de HoughP plus généraux, soit changer de méthode)

CONCLUSION

Découverte OpenCV



Réflexion Algorithmique

Réflexion mathématique

Voir Plus grand