République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumédiène

Faculté d'Informatique
Département d'Intelligence Artificielle et Sciences des Données

Master 2 Systèmes Informatiques intelligents

Module : Vision par ordinateur

Rapport :
Projet vision par ordinateur

Réalisé par :
AMAZOUZ Sara Selma
BELKACEMI Meriem
LABRI Ahlem
MOULAI Mohamed Youcef

Année universitaire : 2023 / 2024

Table des matières

Table des figures

1 Filtres

1.1 Descriptions et rôles

Filtre	Description	Rôle
Moyen	Consiste à remplacer le pixel par la moyenne de son voisinage	Floutage et réduction de la grande variation des valeurs entre un pixel et un autre
Médiane	Consiste à trier les valeurs du voisinage d'un pixel, puis remplacer ce dernier par la médiane. Ce qui fait qu'il est plus lent que le moyen	- Réduction du bruit - Préservation des contours
Gaussien	Effectue une convolution(moyenne pondérée) avec un noyau qui a une courbe en forme de cloche. Plus le pixel est proche du centre du noyau, plus le poids assigné est élevé. Le noyau doit être normalisé en le divisant par la somme de ses poids.	- Floutage et lissage de l'image - Essaie de préserver les détails
Laplacien	Effectue une convolution avec un noyau qui approxime l'opérateur laplacien, qui calcule la seconde dérivée spatiale.	Met en avant les contours et les détails de l'image
Érosion	Consiste à convoluer une image A avec un noyau B. Celui-ci a un point d'ancrage défini, généralement le centre du noyau. Lorsque le noyau B est passé sur l'image, si *tous* les pixels chevauchée par B sont à 255 nous remplaçons le pixel de l'image dans la position du point d'ancrage par cette valeur maximale.	Rétrécis les contours et les tailles
Dilatation	Lorsque le noyau B est passé sur l'image, si *un* des pixels chevauchée par B est à 255 nous remplaçons le pixel de l'image dans la position du point d'ancrage par cette valeur maximale.	- Provoque la croissance des contours
Ouverture	Application d'une érosion de la dilatation d'une image.	- Lissage du bruit - Lissage des contours
Fermeture	Application de la dilatation de l'érosion d'une image.	Comblage du vide
Motion	Effectue une convolution avec un noyau qui contient des 1 sur la ligne du milieu, ce qui fera étirer les pixels horizontalement	- Floutage dans une direction donnée - Suppression des contours
Gaufrage	Convolution avec un noyau de gaufrage qui accentue les variations le long des contours.	- Met en avant les contours et les détails - Lissage du reste de l'image

TABLE 1.1 - Informations sur les filtres implémentés.

1.2 Justification du choix

Filtre	Motion Blur	Gaufrage(Emboss)
Justification	- Simulation d'une image en mouvement - Filtre utilisé dans la photographie - Peut être utilisé pour camoufler une partie de l'image (le visage par exemple) sans heurter l'image - Correction des images involontairement floues	- Simulation d'une image sculptée - Donne un aspect 3D à l'image

TABLE 1.2 - Justification du choix des filtres implémentés.

2 Fonction object color detection :

2.1 Convertir RGB en HSV :

La frame qui est en RGB sera convertie en HSV. Cette conversion implique la normalisation des valeurs RGB, puis le calcul de la teinte (H), de la saturation (S) et de la valeur (V). La frame HSV résultante sépare les informations d'intensité et de couleur plus efficacement que RVB.

2.2 Définir les plages de couleurs :

Les plages de couleurs sont établies dans l'espace HSV pour définir précisément la couleur à détecter.

2.3 Créer un masque de couleur :

Un masque est généré en utilisant la plage de couleurs définie. Ce masque agit comme un filtre, marquant les pixels dans la frame qui se trouve dans la plage de couleurs spécifiée. Le masque binaire obtenu a des pixels blancs représentant la présence de la couleur cible.

2.4 Créer un background noir :

Un fond noir est initialisé comme un tableau de zéros avec les mêmes dimensions que la frame d'origine. Ce background servira de canevas pour visualiser la couleur détectée.

2.5 Appliquer le masque au background :

Le masque de couleur est appliqué au fond noir. Cette étape isole essentiellement les pixels correspondant à la couleur détectée dans la toile noire, en les définissant en blanc tout en laissant le reste de la toile en noir.

3 Amélioration de la fonction "object color detection" :

3.1 Utilisation du filtre gaussien :

Le filtre gaussien peut être utilisé pour améliorer la détection des couleurs en réduisant le bruit dans l'image. Appliqué avant la détection des couleurs, il lisse l'image, réduisant le bruit à haute fréquence et assurant une transition plus cohérente entre les couleurs. Sa capacité à préserver les bords garantit le maintien des limites de couleur. En réduisant le bruit, le filtre gaussien améliore la précision et la fiabilité de la détection des couleurs.

3.2 Utilisation du filtre Médiane :

Le filtre médian peut être utilisé pour la détection de couleur en raison de ses capacités de réduction du bruit et de son approche unique pour préserver les bords. Il remplace la valeur de chaque pixel par la médiane dans son voisinage local, réduisant ainsi les valeurs aberrantes et le bruit aléatoire. Cette approche lisse la répartition des couleurs sur l'image. Il est important de noter que le filtre médian excelle dans la préservation des détails des bords, ce qui le rend avantageux pour les tâches de détection des couleurs où le maintien de limites de couleur distinctes est crucial.

3.3 Amélioration visuel du résultat de la detection de couleur :

Cette fonction crée un fond noir et applique un masque de couleur pour isoler la couleur souhaitée. La couleur détectée est ensuite superposée avec la couleur voulant etre detecté a la place du blanc sur le fond noir. De plus, le cadre d'origine est mis à jour avec des cercles représentant la couleur détectée. La fonction renvoie à la fois la frame originale modifiée et le résultat .

3.4 Subtracting background :

Background subtraction (soustraction d'arrière-plan) est une technique utilisée pour détecter les pixels d'une image qui diffèrent d'un background prédéfini. Cette technique est utilisée dans des cas où le but est de détecter des objets en mouvement ou des changements dans la frame.

Processus de la Background subtraction :

1. **Capturer le Background initial :** Le background est capturé au démarrage de la vidéo. Il représente la scène sans aucun objet ni changement au premier plan.
2. **Calculer la difference de frames :** Pour chaque frame de la vidéo, la différence entre l'image actuelle et le Background est calculée. Cela se fait pixel par pixel.
3. **Thresholding (Seuillage) :** L'étape de seuillage est appliquée à l'image soustraite. Les pixels avec une valeur de différence supérieure à un certain seuil sont considérés comme faisant partie de l'objet détecté, tandis que les pixels inférieurs au seuil sont traités comme faisant partie du Background.
4. **Opérations morphologiques :** Des opérations morphologiques, telles que la dilatation et l'érosion, sont appliquées à l'image de seuil. Ces opérations aident à nettoyer le masque binaire et à lisser les contours des objets détectés.

Comment fusionner la Background subtraction et la détection de couleur :

1. **Subtraction background :**

   La soustraction du fond d'une image vise à isoler les objets en mouvement. Cette opération génère une nouvelle image où le fond apparaît en noir, mettant en évidence les objets mobiles.

2. **Détection des couleurs :** elle est ensuite appliquée à l'image résultant de la soustraction du background. Le masque de couleur binaire est appliqué a la frame résultante. Cette opération ne conserve que les pixels dans le frame qui correspondent à la couleur détectée, alors que le reste devient noir.

4 La fonctionnalité Invisibility Cloak

L'invisibility cloak (cape d'invisibilité) en vision par ordinateur implique l'utilisation de techniques de traitement d'image pour détecter et masquer une couleur spécifique dans une vidéo en temps réel. En remplaçant cette couleur par un arrière-plan, l'effet visuel donne l'illusion que l'objet de cette couleur a disparu ou est devenu transparent dans le flux vidéo capturé.

Afin de réalisé cette fonctionnalité, nous avons procédé de la manière suivante :

1. **Initialisation :**
   — Capture d'un flux vidéo à partir de la caméra en utilisant OpenCV (cv2.VideoCapture()).
   — Acquisition d'une image de référence pour l'arrière-plan (capturée après quelques secondes du lancement de la vidéo).
2. **Conversion en HSV et création du masque :** en utilisant les fonctions de Object Color Detection.
3. **Application du masque :**
   — Utilisation du masque pour remplacer les pixels de la couleur spécifiée dans la frame actuelle par les pixels de l'image de référence (background) à l'aide de la ligne frame[np.where(mask == 255)] = background[np.where(mask == 255)].
4. **Affichage du flux vidéo modifié :** Affichage en temps réel du flux vidéo traité, où la couleur spécifiée est masquée par l'arrière-plan capturé, créant ainsi l'illusion de l'invisibilité de cette couleur dans le flux vidéo.

5 Green Screen

Le green screen ou l'écran vert en vision par ordinateur, est une technique largement utilisée en production vidéo. Cette technique détecte une couleur spécifique (généralement verte) dans une image ou une vidéo, la supprime et la remplace par un autre contenu visuel, tel qu'un arrière-plan alternatif ou un effet visuel. Dans le cadre de ce projet, nous nous intéressons au traitement des vidéos pour remplacer la couleur détéctée par un arrière plan alternatif.

Afin de réaliser cette fonctionnalité, nous avons procédé de la manière suivante :

1. **Détection de la Couleur :**
   Comme fait précédemment, en capturant le flux de vidéo à partir de la caméra, convertissant les couleurs en HSV, et créer un masque binaire pour isoler la couleur verte du frame.
2. **Chargement et redimensionnement de l'image de remplacement :**
   — Ajustement de la taille de l'image de remplacement pour qu'elle corresponde aux dimensions de notre frame.
3. **Remplacement de l'Arrière-plan :**
   — Remplacement des pixels correspondant à la couleur choisie et détectée par les pixels de l'image de remplacement.

Partie 2 : brick racing game

1 Description du jeux

La conception du jeu "Brick Racing Game" repose sur le défi de déplacer une voiture de gauche à droite pour éviter les obstacles. Ce déplacement peut être effectué de plusieurs manières, que ce soit par les touches fléchées du clavier, les touches WASD ou même par la détection de couleur.

2 Conception du jeux

Pour créer le jeu "Brick Racing Game" nous avons suivi ces étapes :

1. **Initialisation du Jeu :** Configuration initiale comprenant la position initiale de la voiture, le score de départ, et la vitesse initiale.
2. **Détection de la Couleur :** Utilisation d'une fonction de détection de couleur pour repérer la position de la couleur spécifiée dans la trame vidéo.
3. **Déplacement de la Voiture :** Déplacement de la voiture en fonction de la position identifiée par la fonction de détection de couleur, en utilisant les touches de déplacement (AD).
4. **Gestion des Obstacles :** Intégration d'obstacles tels que d'autres voitures sur la piste.
5. **Gestion du score :** Incrémentation du score à chaque évitement d'obstacle par la voiture.
6. **Gestion de la Vitesse :** Augmentation progressive de la vitesse au fil de l'avancement du jeu.
7. **Boucle du Jeu :** Boucle qui capture en continu la trame vidéo, met à jour la position de la voiture, gère les événements tels que la détection de couleur et le déplacement de la voiture, gère les obstacles, et affiche la trame mise à jour.
8. **Conditions de Fin :** Arrêt du jeu et affichage d'un écran de fin si la voiture entre en collision avec un obstacle, indiquant ainsi la fin de la partie.

3 Implementation

Dans notre implementation, nous avons realisé les fonctions de base suivantes :

— **initialize_game() :** Cette fonction initialise le jeu en configurant la fenêtre, la caméra, les éléments du jeu, et en définissant les seuils de détection de couleur. Elle crée une surface de route distinctive et positionne les voitures du joueur et des adversaires sur la piste.
— **detect_color() :** En utilisant la caméra, cette fonction repère une couleur spécifique dans la trame vidéo. Elle fait appel à des fonctions telles que **create_mask()** pour créer un masque binaire, **find_contour()** pour trouver les contours de l'objet en couleur, et **bounding_rect()** pour calculer le rectangle englobant le contour.
— **draw_score_speed() :**Dessine le texte du score et de la vitesse sur la fenêtre.

— **handle_input() :** Elle gère les entrées de l'utilisateur pour le mouvement de la voiture dans le jeu.
— **move_cars() :** Elle gère le déplacement de la voiture adverse dans le jeu et met à jour le score en fonction du temps écoulé.
— **check_collision() :** Vérifie s'il y a une collision entre la voiture du joueur et la voiture adverse.
— **display_game_over() :** Cette fonction déclenche potentiellement la fin du jeu en cas de collision.
— **main() :** Est la boucle principale du jeu.

Annexe

1 Bibliothèques requises

Pour exécuter le code présenté dans ce projet, les bibliothèques suivantes sont nécessaires. Assurez-vous de les installer avant d'exécuter le code :

- Numpy
- Math
- Time
- OpenCV
- Threading
- Tkinter
- PIL
- Functools
- Pygame

2 Interface de la partie 1

Cette interface est conçue pour offrir l'ensemble de fonctionnalités présentées dans le rapport via différents boutons. Voici une description pour un guide d'utilisation :

1. **Fenêtre Principale :**

   La fenêtre principale affiche le titre "Projet Vision" et propose plusieurs boutons pour accéder à des fonctionnalités spécifiques.

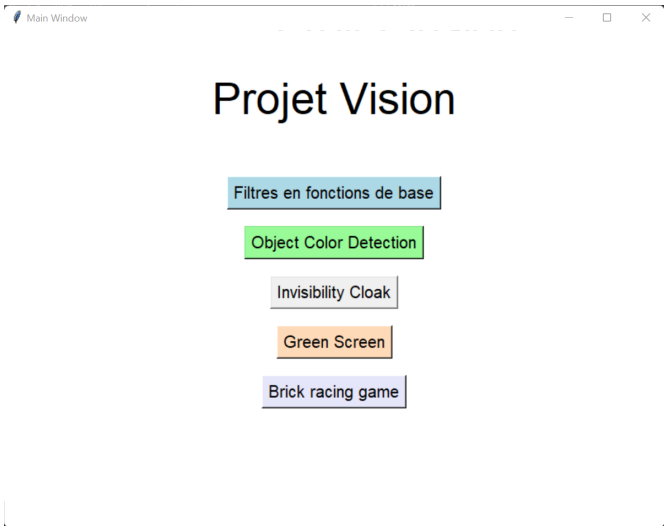

FIGURE 2.1 – Fenêtre Principale

2. **Boutons Principaux :**

— **Filtres en fonctions de base :** Cette fonctionnalité ouvre une fenêtre offrant des options pour appliquer différents filtres d'image de base tels que flou, binarisation, detection de contours, etc. Pour chaque filtre, nous devons saisir sur les champs qui se trouvent sur la meme ligne (s'ils existent) les paramètres dont le filtre a besoin en entrée tels que la taille du kernel ou le threshold. L'image sur laquelle nous allons appliquer ces filtres doit être téléchargée du PC via le bouton 'Upload Image'.

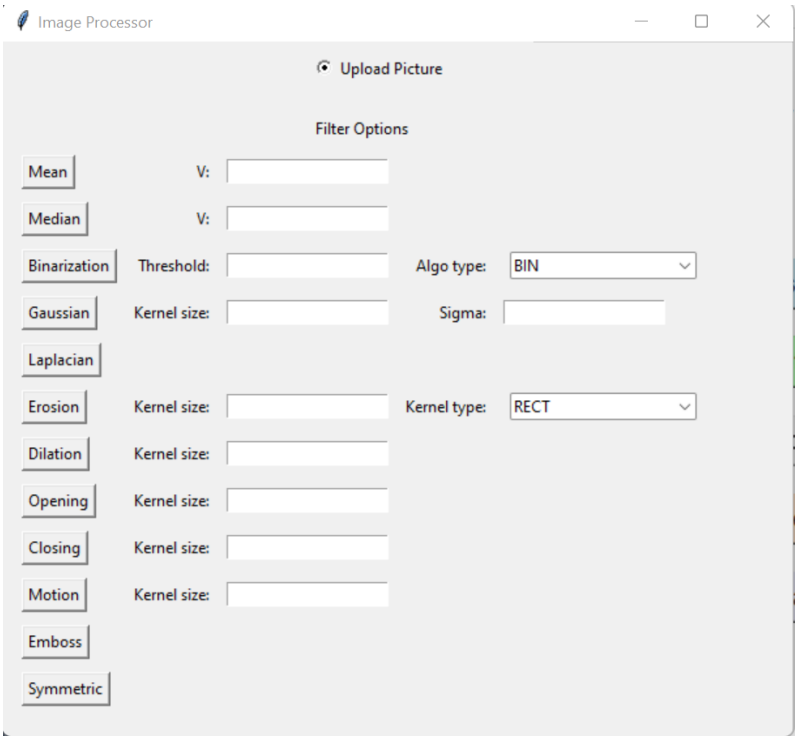

FIGURE 2.2 – Fenêtre de filtres

— **Object Color Detection :** Cette fonctionnalité ouvre une fenêtre permettant la détection de couleurs spécifiques dans une vidéo en temps réel via une webcam. Nous pouvons choisir la couleur que nous désirons détécter (vert, bleu ou jaune) et ensuite choisir le type de détéction que nous voulons effectuer, soit une détéction simple ou avec amélioration (choisir parmi les options affichées) et a la fin cliqué sur "start video" pour lancer la video.

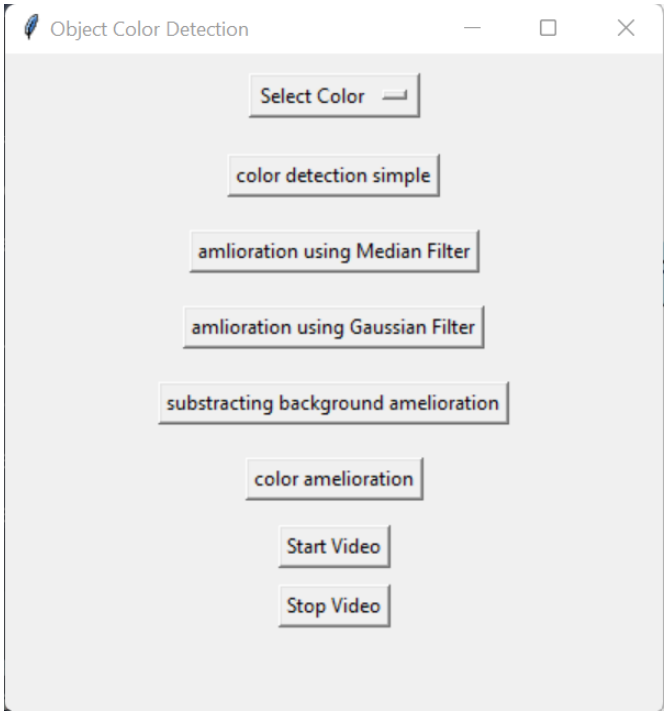

FIGURE 2.3 – Fenêtre de color de detection

— **Invisibility Cloak :** Il suffit d'appuyer sur le bouton pour lancer la caméra et la fonctionnalité sera appliquée automatiquement.
— **Green Screen :** En appuyant sur le bouton, une autre fenêtre sera ouverte pour nous permettre de télécharger une image de remplacement de notre choix. Ensuite, il suffit d'appuyer sur le bouton 'Start Green Screen' pour démarrer la caméra.

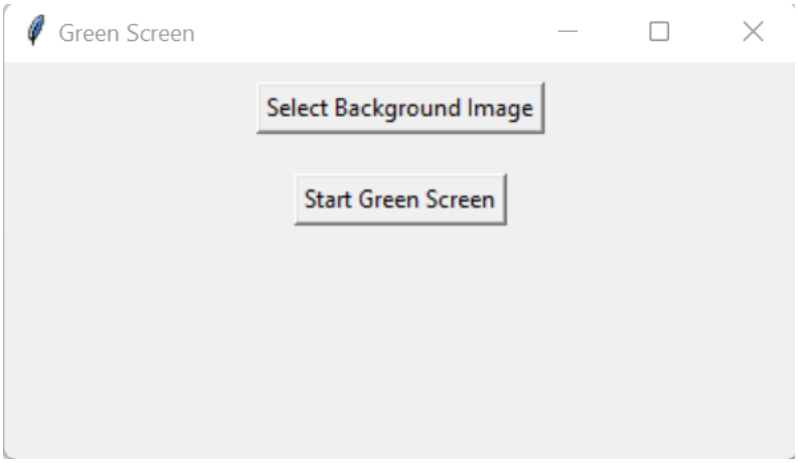

FIGURE 2.4 – Fenêtre de green screen

3 Interface brick racing game

L'interface affiche la caméra du PC en arrière-plan. Elle comprend une section appelée "route" que le véhicule ne peut pas dépasser.

Il y'a aussi le score et la vitesse actuels. L'utilisateur peut contrôler son déplacement à l'aide d'un objet de couleur rouge en utilisant les touches de déplacement vers la droite et vers la gauche ou les touches A et D.

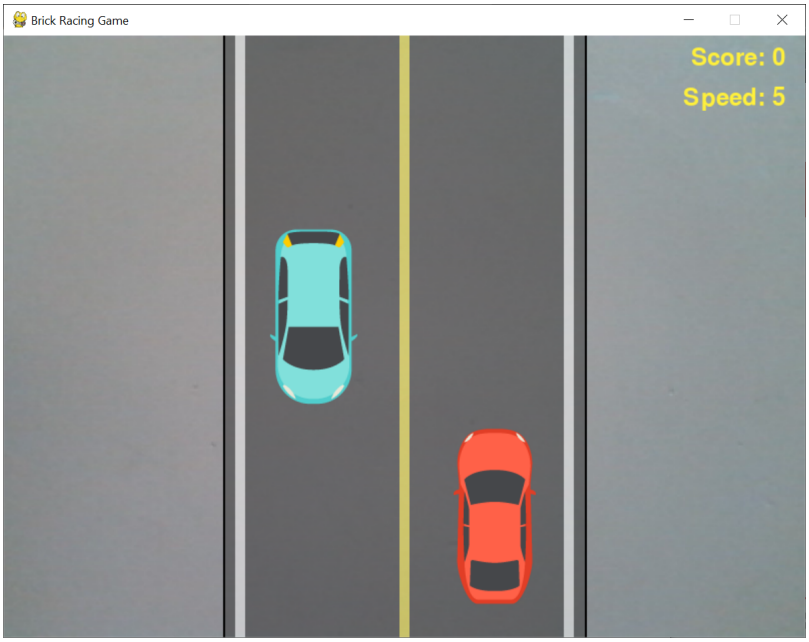

FIGURE 2.5 – Interface du jeux Brick racing

Si deux véhicules entrent en collision, le jeu se termine et l'écran affiche le message "Game Over".

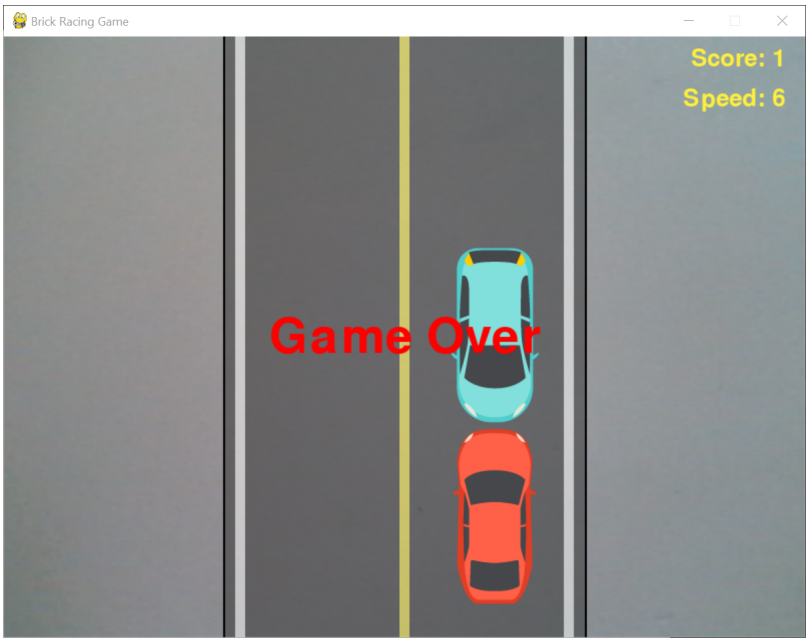

FIGURE 2.6 – Interface du jeux Brick racing- fin jeux