Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Informatique
Département IA et SD

Rapport de projet

Module : Vision par ordinateur

Bricks-Racing-Game

Travail présenté par :

- **BENKOUITEN Aymen————-191931046409**
- **BOUDJENAH Nassim————-191931089384**
- **KENAI Imad Eddine————-191932017671**
- **NOUGHI Tarek————————-191931041952**

2023/2024

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

La vision par ordinateur est un domaine fascinant et en constante évolution, offrant des possibilités novatrices dans de nombreux secteurs, de la reconnaissance visuelle à la création d'applications interactives. Ce rapport présente une exploration approfondie de la vision par ordinateur à travers la réalisation d'un projet dédié. L'objectif principal de ce projet était de mettre en pratique les connaissances acquises en implémentant des filtres d'image, en développant une fonction de détection d'objets par couleur, et en créant un jeu interactif baptisé "brick racing game".

Cette initiative ambitieuse visait à combiner la compréhension théorique des concepts fondamentaux de la vision par ordinateur avec une application concrète, illustrant ainsi les possibilités et les défis inhérents à ce domaine en constante expansion. Ce rapport détaille les étapes, les défis rencontrés, les méthodes employées et les résultats obtenus lors de la réalisation de ce projet, offrant ainsi un aperçu approfondi des différentes composantes de la vision par ordinateur mises en œuvre.

À travers cette démarche, l'objectif était non seulement de démontrer la maîtrise des concepts clés, mais aussi d'explorer des applications pratiques et ludiques, offrant ainsi un aperçu captivant des possibilités infinies offertes par la vision par ordinateur dans divers domaines.

Première partie

Réalisation de filtres

0.1 Description et rôle des filtres dans le traitement d'images :

Type	Filtre	Description	Rôle
Filtres 2D	Moyenne	Calcul de la moyenne des valeurs des pixels dans un voisinage pour lisser l'image et réduire le bruit.	Atténuation du bruit et des variations locales, lissage de l'image.
Filtres 2D	Médian	Détermination de la valeur médiane des pixels dans un voisinage pour atténuer le bruit tout en préservant les détails.	Réduction significative du bruit sans affecter les contours nets.
Filtres 2D	Laplacien	Utilisation d'un noyau pour accentuer les variations rapides d'intensité dans une image, mettant en évidence les détails fins.	Détection des points saillants, mise en évidence des bords.
Filtres 2D	Gaussien	Application d'un filtre basé sur une distribution gaussienne pour réduire le bruit tout en préservant les contours.	Réduction du bruit tout en conservant les structures importantes.
Type morphologique	Érosion	Réduction de la taille des objets dans une image en érodant les pixels selon un motif structurant spécifié.	Retrait ou réduction des objets et des structures, segmentation des régions.
Type morphologique	Dilatation	Augmentation de la taille des objets dans une image en dilatant les pixels selon un motif structurant spécifié.	Renforcement ou augmentation des objets, remplissage des petits trous.
Type morphologique	Ouverture morphologique	Utilisation de l'opération morphologique d'ouverture pour éliminer le bruit en arrière-plan et séparer les objets connectés.	Élimination du bruit, séparation des objets connectés.

Type	Filtre	Description	Rôle
Type morphologique	Fermeture morphologique	Application de l'opération morphologique de fermeture pour combler les petits trous et connecter les objets brisés.	Comblement des trous, connexion des régions disjointes.
filtre proposé	Emboss	Création d'une apparence en relief en accentuant les variations locales d'intensité selon une direction spécifiée.	Création d'effets artistiques, mise en évidence des textures et des reliefs.
filtre proposé	Filtre Sobel	Détection des bords en calculant le gradient d'intensité dans les directions horizontale et verticale.	Mise en évidence précise des contours et des variations d'intensité dans l'image.

TABLE 1 – Description et rôle des filtres dans le traitement d'images.

0.2 Justification du choix des filtres Sobel et Emboss

Les filtres Sobel et Emboss ont été choisis pour leur utilité spécifique dans le domaine du traitement d'images, offrant des fonctionnalités distinctes et complémentaires.

- **Filtre Sobel :** Ce filtre est largement utilisé pour la détection précise des contours dans une image. En calculant le gradient d'intensité dans les directions horizontale et verticale, le filtre Sobel met en évidence les variations rapides d'intensité, permettant ainsi une détection fiable des bords et des contours.

- **Filtre Emboss :** L'embossage est une technique visant à créer un effet de relief dans une image. En accentuant les variations locales d'intensité selon une direction spécifique, ce filtre ajoute une apparence en trois dimensions, simulant un éclairage directionnel. Il est souvent utilisé pour créer des effets artistiques et pour mettre en évidence les textures et les reliefs dans une image.

En incluant ces filtres dans le traitement d'images, on élargit la gamme des opérations possibles, offrant ainsi des outils supplémentaires pour différentes applications, allant de la détection précise des contours à la création d'effets visuels spécifiques.

0.3 Explication des algorithmes pour "Invisibility Cloak" et "Green Screen"

Avant de procéder à la mise en œuvre des techniques de "Invisibility Cloak" et "Green Screen", nous avons utilisé un algorithme de détection des couleurs. Ce processus a suivi les étapes de base que nous avons abordées lors de travaux pratiques antérieurs (TP) juste qu'on a ajouter **des améliorations** qui sont :

- On a utilisé l'algorothme qui détecte le contour. [1]
- Réduction de la taille de l'image pour accélérer les calculs avant la détection des couleurs.
- Transformation des centres des objets détectés pour récupérer leurs coordonnées dans l'image originale.
- Insertion du contour de plus grande taille en tête de liste pour optimiser la gestion des contours.

0.3.1 Fonction remove_green_screen :

- **But :** Supprimer l'écran vert (Green Screen) d'une image.
- **Algorithme :**
  1. Conversion de l'image de l'espace couleur BGR (Rouge, Vert, Bleu) à l'espace couleur HSV.
  2. Création d'un masque pour la plage de couleurs verte définie dans l'espace HSV à l'aide de la fonction **Color.my_in_range**.
  3. Inversion du masque pour garder les zones non vertes à l'aide de la fonction **bitwise_not_custom**.
  4. Création d'un arrière-plan noir.
  5. Remplacement de la zone d'écran vert par l'arrière-plan noir dans l'image d'origine à l'aide de la fonction **bitwise_and_custom**.

0.3.2 Fonction invisibility_cloak_frame :

- **But :** Créer l'effet d'une cape d'invisibilité en remplaçant une couleur spécifique (généralement le rouge) par un arrière-plan.
- **Algorithme :**
  1. Conversion de l'image d'entrée de l'espace couleur BGR à l'espace couleur HSV.
  2. Création d'un masque pour la plage de couleurs spécifique (couleur de la cape) définie dans l'espace HSV à l'aide de la fonction **Color.my_in_range**.
  3. Inversion du masque pour garder les zones non correspondantes à la couleur de la cape à l'aide de la fonction **bitwise_not_custom**.

4. Remplacement de la couleur de la cape par l'arrière-plan statique (image de fond) à l'aide de la fonction **bitwise_and_custom**.

Ces algorithmes utilisent la segmentation de couleur pour isoler des zones spécifiques de l'image, ce qui permet de créer des effets visuels comme l'effacement de l'écran vert et l'effet de cape d'invisibilité **en utilisant les opérations bitwise personnalisées** pour manipuler les masques et les zones d'image correspondantes.

Deuxième partie

Réalisation de "Brick Racing Game"

0.4 Algorithme du "Brick Racing Game"

Nous avons implémenté deux versions distinctes du jeu "Brick Racing" :

Première version : Contrôle par la détection de couleur et clavier

Cette première version permet le contrôle de la voiture soit par la détection des couleurs, soit par l'intermédiaire du clavier.

Deuxième version : Amélioration avec le filtre de Kalman

La deuxième version, quant à elle, n'offre pas l'option de contrôler l'objet avec le clavier, mais intègre l'utilisation du filtre de Kalman pour une détection améliorée de la couleur. Notons que l'utilisation du filtre de Kalman n'interfère pas avec le contrôle par clavier.

0.4.1 Étapes du jeu "Brick Racing Game"

1. **Détection et récupération de la position de l'objet coloré** : L'algorithme détecte et récupère la position de l'objet coloré sur l'écran.
2. **Déplacement de la voiture selon la position de l'objet** : La voiture est déplacée en fonction de la position de l'objet coloré détecté.
3. **Génération d'obstacles en mouvement vers la voiture** : Des obstacles sont générés et descendent progressivement vers la voiture.
4. **Gestion des collisions** : Si une collision entre l'objet et la voiture est détectée, le jeu prend fin.

0.4.2 Améliorations apportées

- **Gestion du score et leaderboard** : Le score est déterminé par le nombre d'obstacles évités. De plus, un leaderboard persistant est sauvegardé en mémoire centrale, affichant les cinq meilleurs scores.
- **Génération d'obstacles et niveaux de difficulté** : L'algorithme "Poisson Disque Simple" est employé pour générer les obstacles de manière équilibrée sur l'écran. Cela évite les préférences de zones spécifiques et assure une distance minimale entre les obstacles.
- **Utilisation du Filtre de Kalman** : Le filtre de Kalman est utilisé pour prédire la nouvelle position de l'objet détecté, même en cas de non-détection de l'objet.

Cette section détaille l'algorithme du "Brick Racing Game" ainsi que les améliorations majeures apportées à celui-ci.

Conclusion Générale

En conclusion, ce projet en vision par ordinateur a représenté une plongée captivante dans les multiples facettes de ce domaine. À travers l'implémentation de filtres d'image, la création d'une fonction de détection d'objets par couleur, et le développement d'un jeu interactif "brick racing game", cette expérience a offert une opportunité précieuse d'appliquer les connaissances théoriques dans un contexte pratique et ludique.

Cette démarche a permis de constater la complexité mais aussi la richesse des possibilités offertes par la vision par ordinateur. Chaque étape a représenté un défi stimulant, que ce soit pour comprendre les nuances des filtres d'image, perfectionner la détection d'objets par couleur, ou créer une expérience de jeu interactive utilisant la caméra pour le déplacement.

Au-delà de l'apprentissage technique, ce projet a souligné l'importance de la créativité et de l'innovation dans le domaine de la vision par ordinateur. Il a également mis en lumière l'interconnexion de ce domaine avec d'autres disciplines, illustrant son potentiel dans des applications variées allant de la reconnaissance d'objets à la création de jeux interactifs.

En somme, ce projet a été une aventure stimulante et éclairante, offrant un aperçu concret des possibilités infinies et des défis passionnants que présente la vision par ordinateur. Il représente une étape importante dans l'exploration et la compréhension de ce domaine en constante évolution.

Troisième partie

Annexe

0.5 Dépendance du code :

Les composants essentiels nécessaires pour exécuter le code incluent Python avec ses bibliothèques standard telles que NumPy et la bibliothèque aléatoire. De plus, le cadre de développement Streamlit est utilisé pour créer une interface utilisateur interactive. L'utilisation d'OpenCV est également indispensable pour la lecture et l'affichage d'images.

Le filtre de Kalman, un outil d'estimation utilisé dans ce projet, a été implémenté en se basant sur le code fourni par le professeur.

En somme, ces éléments - Python, NumPy, la bibliothèque aléatoire, Streamlit et OpenCV - forment le socle technique nécessaire pour exécuter le code et appliquer les filtres de vision par ordinateur ainsi que le filtre de Kalman.

0.6 Manipulation de l'interface :

Tests des Filtres et du Jeu

Étape 1 : Choix du Filtre

Sélection de filtres via la fenêtre contextuelle "Filtres" pour choisir un filtre spécifique parmi une liste prédéfinie.

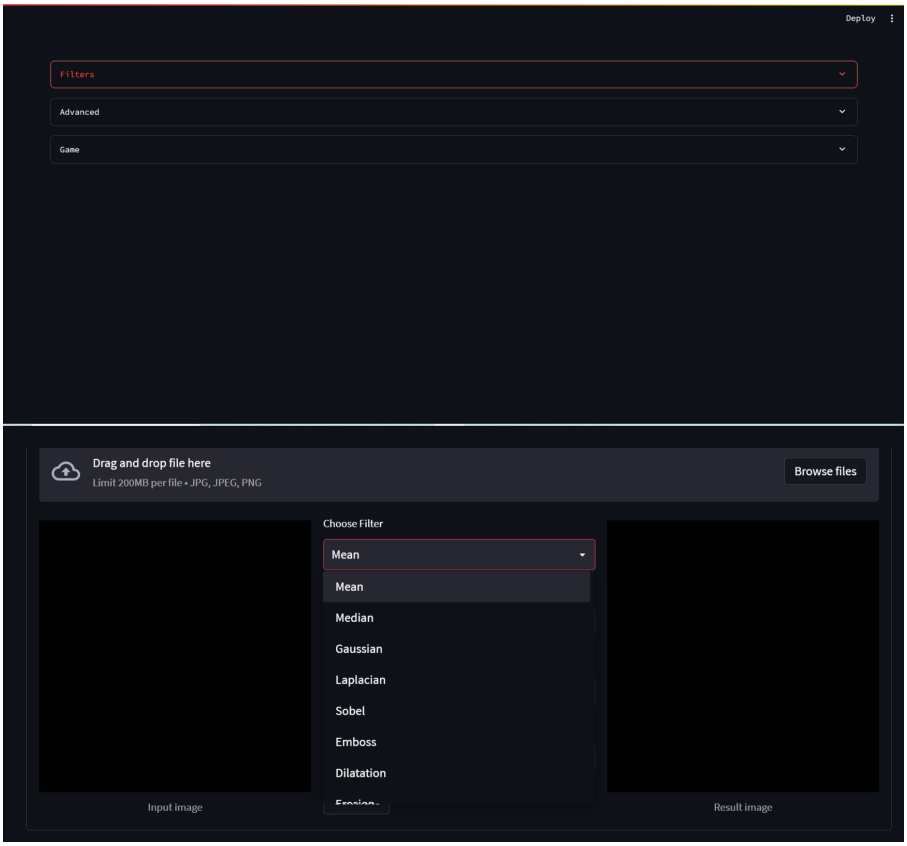

FIGURE 1 – Fenêtre de choix du filtre

Étape 2 : Test du Filtre de Délation

Évaluation des résultats après l'application du filtre de délitation, illustrant son effet sur l'image.

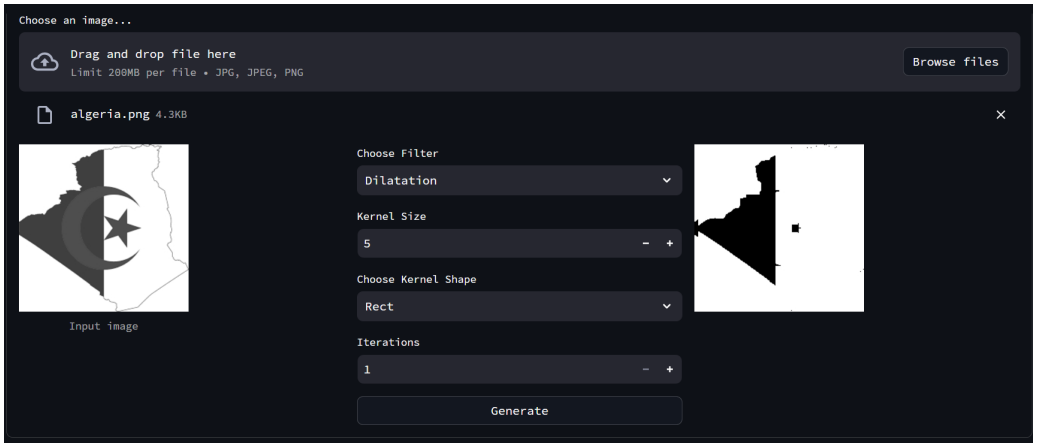

FIGURE 2 – Résultat du filtre de délitation

Étape 3 : Advenced - Effets Visuels

Utilisation de la fenêtre "Advenced" pour choisir parmi les effets visuels disponibles tels que "Invisibility Cloak", "Green Screen" ou "Color Detection".

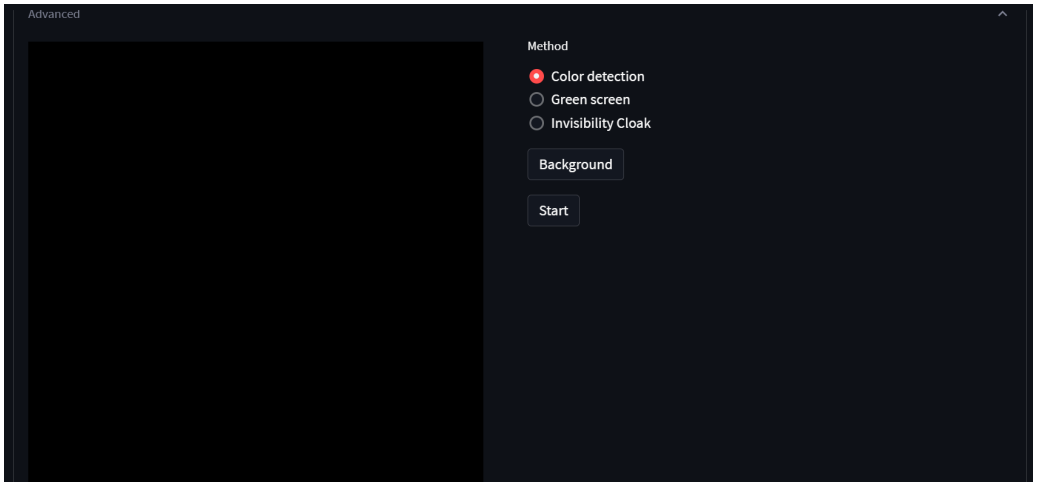

FIGURE 3 – Fenêtre de choix avancé - Effets Visuels

NB : Avant d'utiliser "Invisibility Cloak", il faut capturer le Background.

Étape 4 : Test de la "Color Detection"

Présentation du résultat obtenu suite à l'exécution de la fonction de détection de couleur sur l'image sélectionnée.

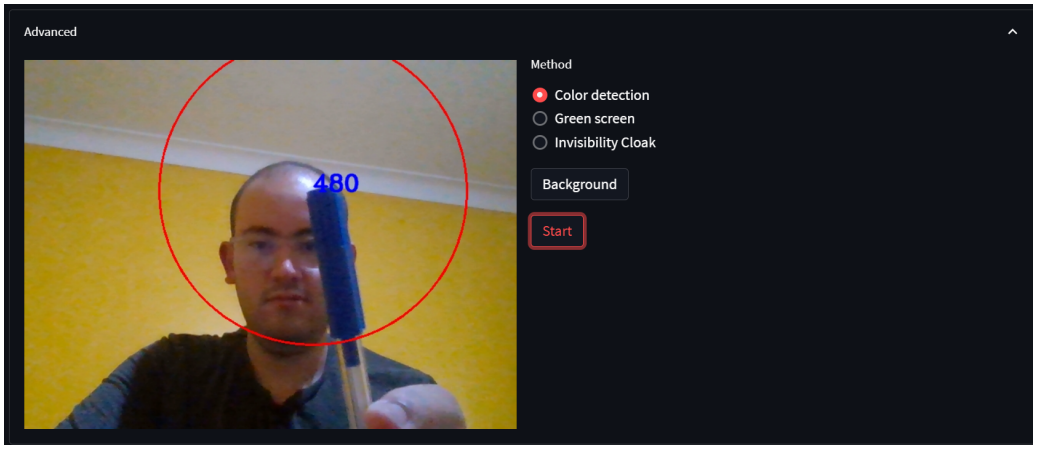

FIGURE 4 – Résultat de la "Color Detection"

Étape 5 : Test du Jeu

Capture d'écran montrant le jeu "Brick Racing Game" en cours d'exécution, illustrant les fonctionnalités et l'interface du jeu.

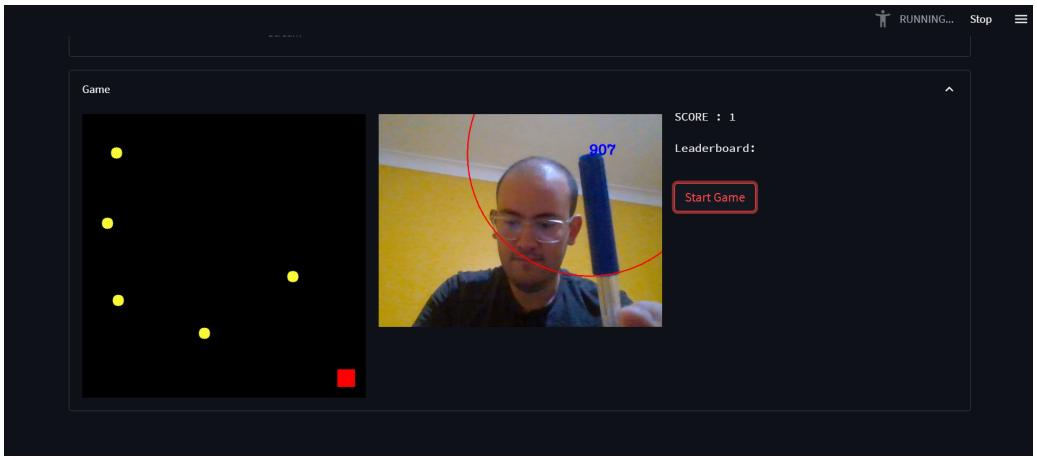

FIGURE 5 – Capture d'écran du jeu

Bibliographie

[1] Contour Algorithm. https://nevis.columbia.edu/~vgenty/public/suzuki_et_al.pdf.