



جامعة هواري بومدين للعلوم والتكنولوجيا
Université des Sciences et de la Technologie
Houari Boumediene



Rapport du projet

Module : Vision

Intitulé :

Traitement d'Image et Implémentation d'un Jeu 'Brick Racing Game

Spécialité : Systèmes Informatiques Intelligents

Réalisé par :

- DJENANE NIHAD
- M'BAREK LYDIA
- TAIBI ABDESSETAR
- TALABOULMA ROMAÏSSA

Travail demandé par :

M. ABDAT ILYES

Promotion : 2023/2024

Table des matières

Introduction générale	1
1 Traitement d'Image et Détection d'Objets	2
1.1 Traitement d'Image	2
1.1.1 Mean Filter	2
1.1.2 Median filter	2
1.1.3 Convolution filter	2
1.1.4 Gauss filter	2
1.1.5 Laplacien filter	2
1.1.6 Morphological filters	2
1.1.7 Gradient filter	3
1.1.8 bilateral filter	3
1.2 Détection des objets	3
1.2.1 Principe	3
1.2.2 Pseudo Code	4
1.3 Réalisation de "Invisibility Cloak" et "Green Screen"	5
1.3.1 Green Screen	5
1.3.2 Invisibility Cloak	5
2 Réalisation de Brick Racing Game	6
2.1 Principe	6
2.2 Pseudo code	6
2.3 PRESENTATION DE L'INTERFACE DE L'APPLICATION :	7

Introduction générale

Imaginez une machine capable de "voir" et de comprendre le monde qui l'entoure en se basant uniquement sur des informations visuelles. C'est précisément ce que vise la vision par ordinateur. C'est comme donner des yeux à un ordinateur ! Tout cela repose sur des technologies qui aident la machine à traiter l'information numérique des images.

Dans le cadre de ce projet, la première partie sera dédiée au filtrage des images, des outils de traitement visant à améliorer la qualité visuelle en éliminant le bruit, en accentuant les contours et en mettant en évidence des caractéristiques spécifiques. Ces filtres contribuent ainsi à une analyse visuelle plus précise. De plus, nous aborderons la détection d'objets par couleur.

La deuxième partie, elle se concentrera sur l'application concrète de ces concepts. Nous avons développé un jeu interactif appelé "Brick Racing Game" pour illustrer cette application. Cette expérience immersive invite les joueurs à manœuvrer une voiture en utilisant la détection de couleur, évitant les obstacles tout en explorant des fonctionnalités telles qu'un système de score dynamique.

Chapitre 1

Traitement d'Image et Détection d'Objets

1.1 Traitement d'Image

1.1.1 Mean Filter

Le filtre de moyenne calcule la moyenne des pixels voisins pour chaque pixel de l'image, il est utilisé pour le lissage et la réduction du bruit, mais peut provoquer un flou.

1.1.2 Median filter

Le filtre médian est un filtre de lissage qui remplace chaque pixel par la médiane des pixels voisins, il est efficace pour réduire le bruit impulsif tout en préservant les contours.

1.1.3 Convolution filter

La convolution est l'opérateur de base du traitement linéaire des images. Pour calculer une convolution, on remplace la valeur de chaque pixel par la valeur du produit scalaire entre les valeurs du noyau de convolution et les valeurs du voisinage du pixel considéré, elle est appliquée à filtrer l'image pour atténuer le bruit présent, etc. . .

1.1.4 Gauss filter

Le filtre de gauss est un filtre linéaire, passe-bas, il effectue une convolution de l'image avec une gaussienne, Il est couramment utilisé pour le lissage dans une image.

1.1.5 Laplacien filter

c'est la dérivée seconde de la fonction de niveau de gris, le contour correspond au passage par zéro du laplacien, Il accentue les détails fins et les contours d'une image en détectant les variations de l'intensité lumineuse.

1.1.6 Morphological filters

Les filtres morphologiques sont basés sur deux opérations : l'érosion ; effectue un « et » logique entre les voisins d'un pixel et la dilatation ; effectue un « ou » logique. L'érosion diminue le contour de l'ordre d'un pixel et la dilatation augmente l'épaisseur d'un contour. On peut faire

une combinaison des deux filtres. Fermeture ; dilatation puis érosion, et on a Ouverture ; érosion puis dilatation. Ils sont particulièrement utiles dans le traitement d'images binaires.

1.1.7 Gradient filter

c'est la dérivée première de la fonction de niveau de gris, le contour correspond à la norme du gradient supérieure à un seuil donné. On a choisi le filtre le plus connu SOBEL. Il est utilisé pour détecter les changements d'intensité dans les directions horizontale et verticale, mettant en évidence les contours d'une image.

1.1.8 bilateral filter

Le filtre bilatéral est un filtre non linéaire qui est utilisé pour lisser les images tout en conservant les bords. Il fonctionne en calculant la valeur d'un pixel en fonction de ses voisins, en tenant compte à la fois de la proximité spatiale et de la similarité de couleur.

Les entrées du filtre bilatéral sont une image et un masque. L'image est la source de données sur laquelle le filtre agira. Le masque est une matrice qui définit la proximité spatiale et la similarité de couleur.

La sortie du filtre bilatéral est une nouvelle image qui est un lissage de l'image d'origine.

1.2 Détection des objets

1.2.1 Principe

Afin de détecter l'objet en fonction de sa couleur, notre approche commence par simplifier le traitement en convertissant l'image originale de l'espace RGB à l'espace HSV. Ensuite, un masque binaire est appliqué, attribuant la valeur 1 à tous les pixels dont la couleur se situe entre les valeurs "low" et "high", tandis que les autres pixels sont mis à zéro. Pour améliorer la qualité de la détection, des opérations de morphologie sont appliquées au masque binaire. Cela permet de régulariser la forme et les contours de l'objet détecté. Enfin, pour déterminer les contours de l'objet, nous utilisons le calcul de la magnitude du gradient. En parcourant les pixels du masque, la différence entre les valeurs de pixels voisins est évaluée, puis le gradient est calculé. Les pixels dont la magnitude du gradient dépasse un seuil défini sont considérés comme faisant partie des contours de l'objet détecté. Pour déterminer le centre de l'objet, nous prenons le point maximum et le point minimum dans la liste de points puis calculer le centre.

1.2.2 Pseudo Code

Input : image, low couleur , high couleur

Output: centre, points

Conversion de l'image de RGB a HSV ;

Applique une masque binaire ;

Appliquer des opérations de morphologie sur le masque ;

for *chaque pixel* (x, y) *dans* *mask* **do**

$a \leftarrow$ Calculer la différence entre le pixel $(x, y + 1)$ et le pixel $(x, y - 1)$;

$b \leftarrow$ Calculer la différence entre le pixel $(x + 1, y)$ et le pixel $(x - 1, y)$;

$\text{gradient_magnitude} \leftarrow \sqrt{a^2 + b^2}$;

if $\text{gradient_magnitude} \geq 0.5$ **then**

 | ajouter a la liste des points

end

end

centre \leftarrow centre des points ;

return centre, points

1.3 Réalisation de "Invisibility Cloak" et "Green Screen"

1.3.1 Green Screen

Principe : L'idée est d'ajouter un objet avec un fond vert à une image en supprimant le fond. L'algorithme parcourt chaque pixel du masque. Si le pixel correspond à un pixel de l'objet (masque = 1), le pixel de l'image de destination est remplacé par le pixel correspondant de l'image contenant l'objet. Sinon, le pixel de destination reste inchangé.

Pseudo-code :

Input : imageDestination, imageObjet, mask

Output: imageDestination avec l'objets

```
for chaque pixel (x,y) dans mask do
    if mask[x,y] = 1 then
        imageDestination[x,y] ← imageObjet[x,y]
    else le pixel de destination reste inchangé
    end
end
return imageDestination
```

1.3.2 Invisibility Cloak

Principe : L'idée derrière est de rendre un objet invisible dans une image. Le principe que nous avons adopté est de parcourir chaque pixel de l'image en vérifiant le masque associé. Lorsque le masque a une valeur de 1, nous avons appliqué une transformation au pixel correspondant en lui ajoutant un pas.

NB : Le pas est la différence moyenne entre les valeurs des pixels adjacents qui doivent être modifiés.

Par exemple, si nous avons l'image initiale suivante :

$$\begin{bmatrix} 18 & 16 & 60 & 10 & 23 & 43 & 70 \\ 39 & 38 & 85 & 29 & 10 & 39 & 58 \end{bmatrix}$$

et le masque correspondant :

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Nous calculons le pas comme suit : $\frac{70-60}{3} = 2, ..$

En appliquant la transformation, l'image modifiée devient :

$$\begin{bmatrix} 18 & 16 & 60 & 62 & 64 & 66 & 70 \\ 39 & 38 & 85 & 29 & 10 & 39 & 58 \end{bmatrix}$$

Ainsi, chaque pixel associé à une valeur de 1 dans le masque subit une modification en augmentant sa valeur par le pas calculé, tout en conservant les autres pixels inchangés.

Chapitre 2

Réalisation de Brick Racing Game

2.1 Principe

Dans cette partie, nous allons implémenter le jeu "Brick Racing Game", qui consiste à déplacer une voiture afin d'éviter les obstacles présents sur la route. Le déplacement de la voiture s'effectue à l'aide de la caméra en déplaçant une couleur spécifique. La voiture suivra le déplacement de cette couleur.

Pour réaliser le jeu, nous avons utilisé la fonction de détection d'objets que nous avons mise en place dans le chapitre 1 afin de localiser le centre de l'objet.

Ensuite, nous comparons le centre de l'objet avec le centre du cadre. Si le centre du cadre est supérieur, la voiture se déplacera vers la gauche. Si le centre du cadre est inférieur au centre de l'objet, la voiture se déplacera vers la droite.

2.2 Pseudo code

Input : image, low couleur , high couleu

Output: Voiture deplacer

centre \leftarrow centre de l'objet obtenue avec detecetion d'objets ;

centre de frame \leftarrow (largeur frame / 2, hauteur frame / 2) ;

if *centre*[0] \geq *centre de frame* [0] **then**

 | deplacer la voiture a gauche ;

end

if *centre*[0] \leq *centre de frame*[0] **then**

 | deplacer la voiture a droite ;

end

2.3 PRESENTATION DE L'INTERFACE DE L'APPLICATION :

L'interface est organisée de tel sorte à ce que l'utilisateur puisse l'utiliser très facilement. Elle se compose d'un seul être qui contient tous les opérations, les inputs et l'affichage de résultat . On peut trouver tous les filtres vu en tps plus d'autres filtres .on selection **Source image = image**

L'image avant et après seront affichés après le clique sur appliquer on remplissent tous les entres. Implémentation de détection objet par couleur avec tous les ameliorations ,Implémentation de Invisibility cloak et de Green screen on selection ,**Source image = camera**.

Et une fenêtre de jeu , où on peut utiliser les touches pour déplacer le jour ou utiliser la couleur , toutes les améliorations sont prises en compte .

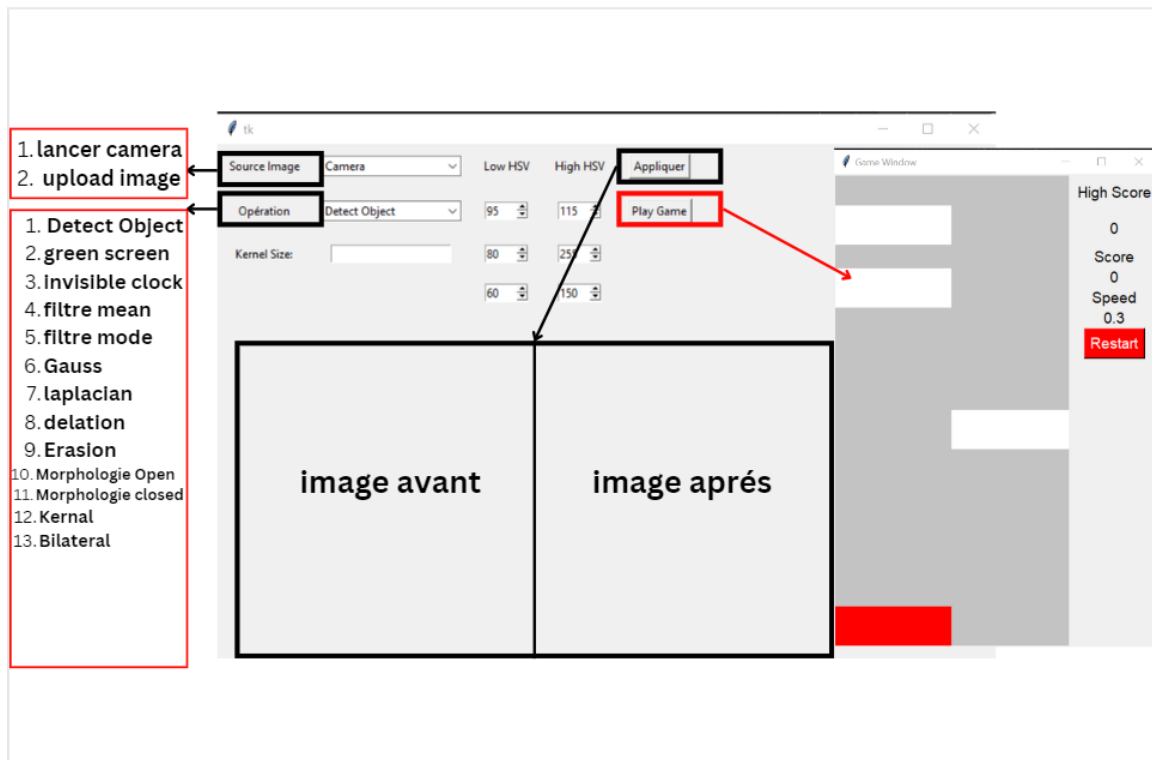


FIGURE 2.1 – Les interfaces de projet