



Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Électronique et d'Informatique
Département Informatique



Master Systèmes Informatiques Intelligents

Vision par ordinateur

Rapport de TP

BENRABIA Afeef

TABLE DES MATIERES

| | | |
|----|---|---|
| 1 | Introduction | 1 |
| 2 | Problématique et objectifs | 1 |
| 3 | Fonctionnalités..... | 1 |
| 4 | Outils d'implémentation | 2 |
| 5 | Présentation de l'interface | 2 |
| 6 | Dictionnaire des fonctions | 4 |
| 7 | Construction et synchronisation | 5 |
| 8 | Détection dynamique des couleurs | 5 |
| 9 | Localisation des couleurs sur le flux vidéo..... | 6 |
| 10 | Curseur et dessin sur le plan | 7 |
| 11 | Modification de la taille et couleur du trait de dessin | 8 |
| 12 | Enregistrement des captures et de vidéo | 8 |
| 13 | Génération d'exécutable..... | 8 |
| 14 | Conclusion | 9 |

1 INTRODUCTION

Le fonctionnement du corps humain est tellement fascinant que l'être humain lui-même essaye de le simuler, il tente de créer une intelligence artificielle qui servira ses intérêts et besoins divers. La vision par ordinateur est une branche de l'intelligence artificielle consacrée à l'étude et l'imitation du fonctionnement de l'œil, les méthodes développées reposent en principe sur l'extraction d'information à partir de l'image. L'imagerie trouve son application dans plusieurs domaines de la vie quotidienne tels que la robotique, l'industrie, la médecine, la télésurveillance, les jeux vidéo...etc.

Dans ce travail, on touche à un domaine d'application de la vision par ordinateur en utilisant nos connaissances basiques.

2 PROBLEMATIQUE ET OBJECTIFS

Le but du TP est de réaliser une application de dessin qui utilise les gestes de doigts. L'implémentation ne doit comprendre que les méthodes de base vues en séance de TP. Il est demandé de programmer le scénario suivant :

- L'utilisateur met deux couleurs distinctes sur deux doigts et lance sa caméra, l'application doit détecter et mémoriser les couleurs choisies.
- Une fois les couleurs mémorisées, l'application doit faire la détection sur le flux vidéo.
- Après la détection des couleurs sur le flux vidéo, on calcule la distance entre eux, si cette distance est inférieure à un seuil défini on dessine sur un fond blanc.

3 FONCTIONNALITES





L'application implémentée assure les fonctionnalités suivantes :

- Détecter dynamiquement les couleurs utilisées.
- Détecter les couleurs sur le flux vidéo et les mettre en évidence.
- Afficher un curseur sur le plan de dessin.
- Permettre de modifier la taille et la couleur du trait de dessin.
- Mettre en disposition une gomme dont la taille est modifiable.
- Afficher la taille et la couleur du curseur.
- Permettre de changer le fond du plan du dessin à une image.
- Permettre la sauvegarde du dessin.
- Permettre l'enregistrement d'une vidéo des étapes du dessin.

Dans ce qui suit, le principe de réalisation de ces fonctionnalités sera présenté.

4 OUTILS D'IMPLEMENTATION

Pour implémenter l'application, on utilise :

- Python 3 
- PyQt 5 
- OpenCV 3 
- NumPy 1 

Seules les méthodes de base de OpenCV ont été utilisées, il s'agit de : *VideoCapture*, *VideoWriter*, *VideoWriter_fourcc*, *line*, *circle*, *flip*, *waitKey*, *imread*, *resize*, *imwrite*, *cvtColor*.

NumPy est utilisé pour optimiser les boucles. En effet les boucles de parcours de matrice d'image prennent beaucoup de temps et ralentissent le flux vidéo se bloque , les fonctions utilisées sont : *full*, *array*, *where*, *amax*, *amin*, *ones*.

5 PRESENTATION DE L'INTERFACE

Au lancement de l'application, l'interface de Figure 3 s'affiche pour choisir un emplacement de stockage des captures et vidéo. Une fois validé, on y voit l'interface principale :

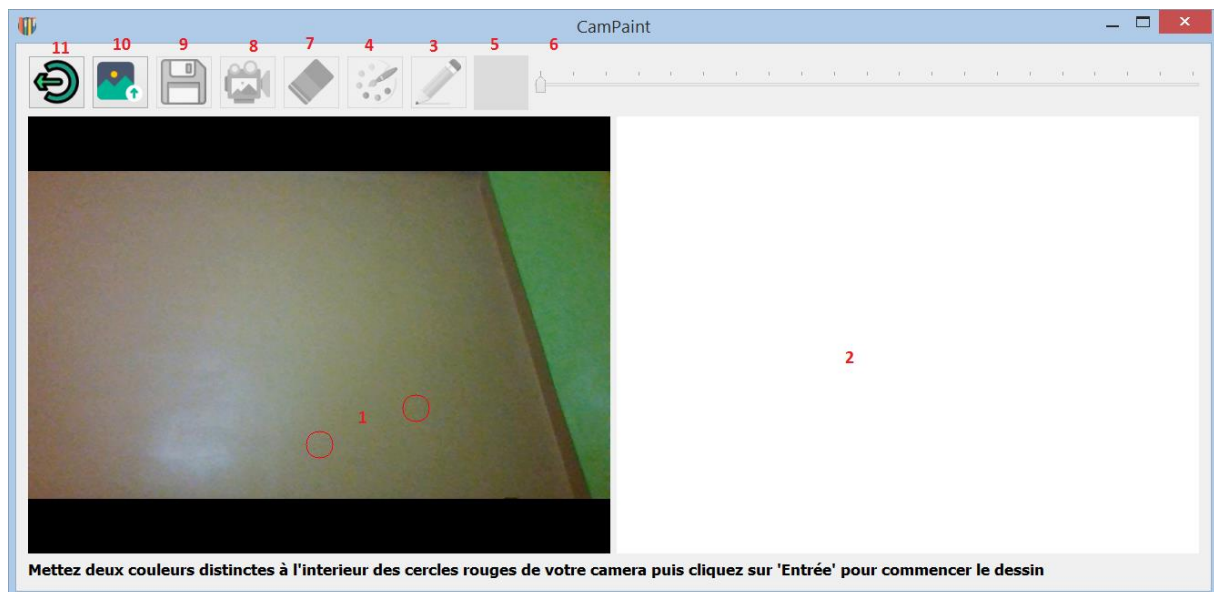


Figure 1: Interface principale

- 1- Le flux vidéo: on trouve deux cercles rouges, l'utilisateur doit positionner les couleurs à l'intérieur puis cliquer sur entrée.
- 2- Le plan de dessin.
- 3- Sélectionner/désélectionner le crayon.
- 4- Palette pour changer la couleur du crayon.
- 5- Afficher la taille et couleur crayon/gomme.
- 6- Modifier la taille du crayon/gomme.
- 7- Sélectionner/désélectionner la gomme.
- 8- Commencer/pauser l'enregistrement vidéo.
- 9- Enregistrer une capture du plan de dessin.
- 10- Télécharger une image pour le fond de dessin.
- 11- Terminer et quitter l'application.

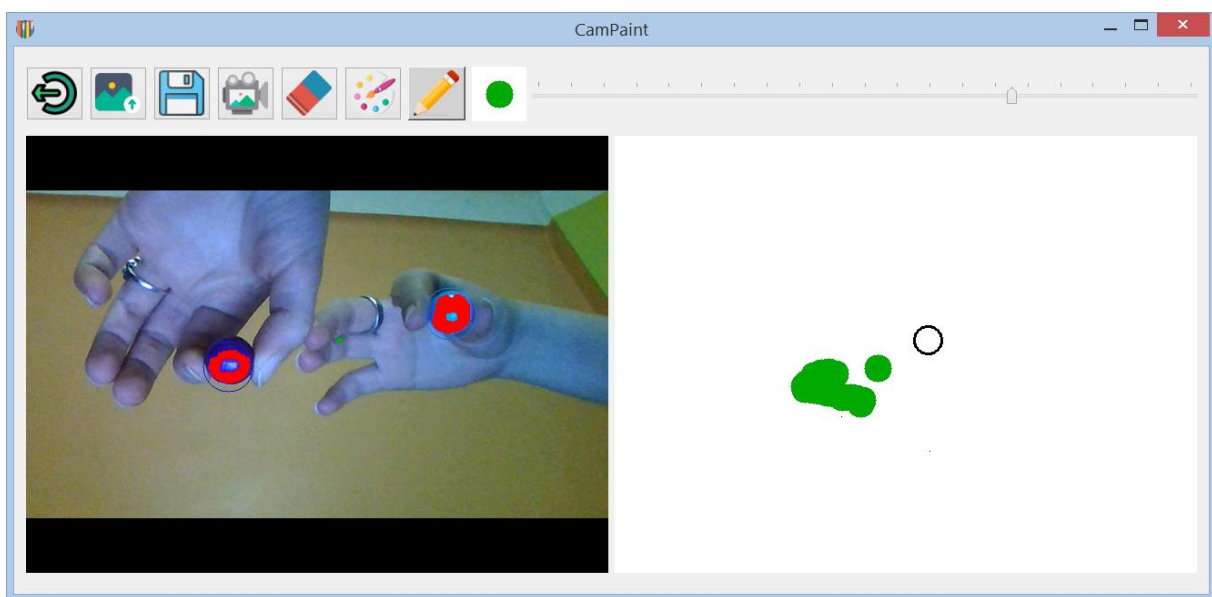


Figure 2: interface principale après détection des couleurs

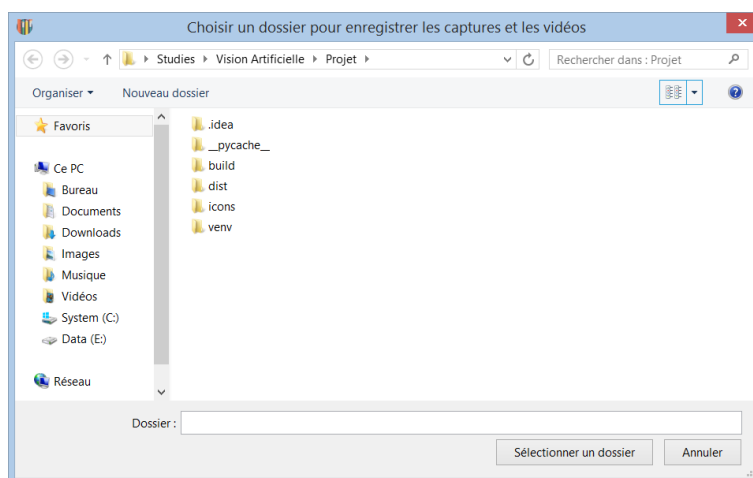


Figure 3: Interface choix d'emplacement de sauvegarde

6 DICTIONNAIRE DES FONCTIONS

Dans cette partie, on récapitule le rôle de chaque fonction, le pseudo-code des fonctions les plus importantes sera donné par la suite.

| fonction | rôle |
|----------------------------------|--|
| <code>__init__</code> | constructeur. |
| <code>Run</code> | fonction du chronomètre. |
| <code>KeyPressedEvent</code> | fonction prédéfinie qui se lance lorsqu'il y a une touche du clavier enfoncée. |
| <code>Process</code> | lance le dessin avec crayon/gomme. |
| <code>draw</code> | étant donnée une couleur, elle se charge de dessiner avec la couleur donnée ou d'afficher le curseur selon la distance entre les doigts. |
| <code>draw_circles</code> | affiche deux cercles rouges sur le flux vidéo pour le positionnement des doigts. |
| <code>detect_colors</code> | localise les couleurs sur le flux vidéo et les met en évidence. |
| <code>get_frame</code> | construit l'image de la caméra et du plan. |
| <code>display_frame</code> | affiche l'image de la caméra et du plan sur l'interface. |
| <code>setGUI</code> | constructeur d'interface. |
| <code>upload_btn_clicked</code> | lors d'un clic sur le bouton 10, chargez une image comme fond du plan de dessin. |
| <code>save_btn_clicked</code> | lors d'un clic sur le bouton 9, sauvegardez une capture du plan de dessin. |
| <code>video_btn_clicked</code> | lors d'un clic sur bouton 8, activer/ désactiver l'enregistrement de la vidéo. |
| <code>tools_enabled</code> | perd une valeur booléenne en entrée, elle active/désactive certaines composantes de l'interface. |
| <code>center</code> | affiche l'interface au centre de l'écran. |
| <code>closeEvent</code> | fonction prédéfinie lors de la terminaison de l'application, elle libère la camera |
| <code>create_btn</code> | création d'un bouton avec une icône. |
| <code>slider_changes</code> | appelée lorsque la valeur du slider change, elle met à jour la variable de taille du curseur et actualiser l'affichage du champ 5 de l'interface principale. |
| <code>palette_btn_clicked</code> | lors d'un clic sur le bouton 4 de palette, affiche une palette pour choisir une couleur puis change la variable de couleurs et actualiser l'affichage du champ 5 de l'interface principale |
| <code>pencil_btn_clicked</code> | lors d'un clic sur le bouton 3 de dessin. |
| <code>eraser_btn_clicked</code> | lors d'un clic sur le bouton 7 de gomme. |
| <code>esource_path</code> | recupère le chemin absolu d'une icône. |

7 CONSTRUCTION ET SYNCHRONISATION

Pour que l'application tourne proprement, il faut une certaine synchronisation entre le flux vidéo, le plan de dessin et l'interface utilisateur. *PyQt* met à disposition une classe pour servir ce besoin, il s'agit de la classe *QTimer*. C'est un chronomètre qui remplace une boucle *while*, et exécute une fonction passée en paramètre à chaque intervalle de temps. Le pseudo-code du constructeur et le suivant:

| Constructeur |
|--|
| initialisation des variables construction de l'interface si la caméra est ouverte afficher l'interface de l'application demander le chemin de sauvegarde lancer le chronomètre sinon afficher un message d'erreur |

Parmi les variables initialisées, on trouve des coordonnées et des états qui assurent le bon fonctionnement du programme. Le chronomètre exécute la fonction *run* dont le pseudo-code est :

| run. |
|---|
| si les couleurs ne sont pas encore définies construire l'image de camera et du plan dessiner les cercles de détection de couleurs sur l'image de camera afficher l'image de camera et du plan sinon construire l'image de camera et du plan localiser les couleurs sur l'image de la camera enregistrer une capture du plan pour la vidéo si demandée afficher l'image de camera et du plan |

8 DETECTION DYNAMIQUE DES COULEURS

Au lancement de l'application, deux cercles rouges se trouvent sur le flux vidéo, l'utilisateur doit positionner ses doigts à l'intérieur des cercles. Au clic sur entrée, on mémorise la couleur du pixel central de chaque cercle.

Cette fonctionnalité est liée à l'événement d'un clic sur une touche. Une fonction prédéfini en python (*keyPressEvent*) traite cet évènement, son pseudo-code et donné par :

| |
|---------------------------------|
| keyPressEvent(touche). |
|---------------------------------|

si la touche est 'entrée'
prendre la couleur du premier centre
prendre la couleur du second centre
changer la variable d'état de détection de couleur

9 LOCALISATION DES COULEURS SUR LE FLUX VIDEO

Le principe est de parcourir la matrice de l'image camera pixel par pixel et essayer de border la couleur par un rectangle. Donc, parcourir la matrice pour trouver les indices des pixels qui ont une couleur proche de la couleur mémorisée, puis prendre les valeurs max et min de x et de y, Figure 4 illustre le principe sur la couleur rouge. Ces quatre valeurs permettent de définir deux points (x_min, y_min) et (x_max, y_max) qui servent à créer le rectangle.

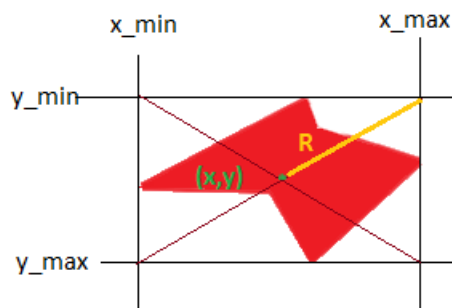


Figure 4: exemple de localisation de couleur

Pour calculer la distance entre la couleur d'un pixel et la couleur mémorisée, on utilise la norme euclidienne donnée par :

$$d(C1, C2) = \sqrt{(R1 - R2)^2 + (G1 - G2)^2 + (B1 - B2)^2}$$

Où :

- C : la couleur.
- R : le rouge.
- G : le vert.
- B : le Bleu.

Comme les boucles en python ne sont pas optimisées, les temps de parcours d'une matrice d'image ralenti le programme, de ce fait on utilise la fonction *where* de *NumPy* qui prend en paramètre une condition et retourne les indices des pixels qui la satisfont. Dans notre cas, la condition est que la distance entre les deux couleurs soit inférieure à un seuil défini.

Les pixels qui répondent à la condition seront affichés en rouge. On calculera aussi le centre (x, y) et le rayon R pour pouvoir encercler la couleur sur le flux vidéo (voir Figure 2: interface principale après détection des couleurs).

$$x = \frac{x_{min} + x_{max}}{2}, y = \frac{y_{min} + y_{max}}{2}$$

$$R = \frac{\sqrt{(x_{max} - x_{min})^2 + (y_{max} - y_{min})^2}}{2} \text{ (théorème de Pythagore)}$$

La fonction detect_colors assure cette tâche, son pseudo-code est comme suit :

detect_colors

pour chaque couleur mémorisée

 récupérer les indices des pixels dont la distance avec la couleur mémorisée est inférieure au seuil

 prendre les valeurs max et min de x et d'y

 calculer le centre et le rayon

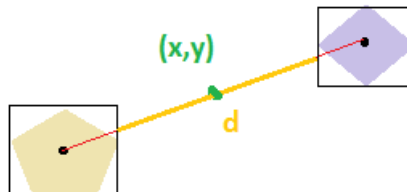
 dessiner un cercle autour des pixels sélectionnés

 colorer les pixels sélectionnés en rouge

10 CURSEUR ET DESSIN SUR LE PLAN

Le dessin sur le plan s'agit soit du trait d'un crayon, ou bien celui d'une gomme, la différence est que la couleur de la gomme est toujours blanche alors que celle du crayon dépend du choix de l'utilisateur. Cette étape vient après la localisation des couleurs, donc on dispose des centres et rayons de chacune.

On calcule le centre et la distance entre les bordures des deux couleurs. Comme le dessin se fait au milieu entre les deux couleurs, on calcule le centre.



Pour expliquer ces calculs, prenant le cas que présente l'image ci-dessus : deux couleurs distinctes C1 et C2, chacune son centre (x1, y1) (x2, y2) et son rayon R1 et R2, les formules sont données par :

$$x = \frac{x_1 + x_2}{2}, y = \frac{y_1 + y_2}{2}$$

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} - R1 - R2$$

Si la distance est inférieure à un seuil défini, on dessine le parcours du curseur sur le plan, sinon on affiche le curseur en déplacement (voir Figure 2: interface principale après détection des couleurs).

Pour l’affichage du curseur , on crée une copie du dernier état du plan, on dessine le curseur sur cette copie, la fonction d’affichage sur l’interface prendra en charge le soin d’afficher selon la distance, soit le plan original soit du plan dupliqué. Le principe décrit est assuré par la fonction *draw* qui est illustrée par le pseudocode suivant :

| |
|--|
| draw(couleur). récupérer le centre de chaque couleur mémorisée calculer le centre entre les deux afficher ce centre sur le flux vidéo calculer la distance si distance inférieure à seuil dessiner sur le plan original sinon dessiner sur le plan dupliqué |
|--|

11 MODIFICATION DE LA TAILLE ET COULEUR DU TRAIT DE DESSIN

Le changement de valeur du slider et le clic sur la palette modifient des variables globales de la couleur et de taille du trait, ces variables sont utilisées lors du dessin.

12 ENREGISTREMENT DES CAPTURES ET DE VIDEO

Le clic sur le bouton de capture permet de sauvegarder une image du plan de dessin. Alors que le clic sur le bouton de vidéo permet de changer l’état d’une variable qui utilise la fonction *record_video* pour enregistrer ou non la capture du plan sur le flux de sortie.

L’enregistrement des captures et de la vidéo se fait dans le chemin spécifié au lancement de l’application, les noms des fichiers sont du format de date aaaa-mm-jj-hh-mm.extension .

13 GENERATION D’EXECUTABLE

Pour la génération d’exécutable, on utilise PyInstaller . C’est un programme qui fige les programmes python en fichier exécutable avec une compression transparente, il fonctionne avec Python 2.7 et 3.4 - 3.7 et est multiplateforme.

On se positionne dans le dossier du projet et on exécute la commande :

```

pyinstaller projet.py      #nom du fichier python
--name CamPaint           #nom d’application
--icon icons\tools.ico    #icone d’application
--onefile                 # un seul fichier
--windowed                # sans consoles

```

`--add-data icons;icons` #inclure les ressources.

Une fois l'opération achevée, on trouve dans le chemin du projet, un dossier *dist* dans lequel se trouve notre exécutable.  CamPaint.exe

14 CONCLUSION

Ce TP nous a permis de mettre en pratique nos connaissances acquises durant les séances de TP de vision par ordinateur, et d'avoir une idée plus claire sur l'extraction d'information à partir des images. Nous avons pu mettre en œuvre une application de dessin qui permet de :

- Afficher le flux vidéo sur une interface graphique.
- Localiser des couleurs sur un flux vidéo.
- Offrir des options de dessin : changement de couleur et de taille et utilisation de gomme.
- Afficher un curseur en manipulant deux copies de la même image.
- Charger une image en fond pour le plan de dessin.
- Enregistrer des flux vidéo avec possibilité de pause.