Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'électronique et d'informatique
Département d'informatique

Rapport de projet

Module : **Vision artificielle par ordinateur**

Master 1 SII

Réalisation d'une application de painting en utilisant une détection de couleurs

- Réalisé par :

  BENHADDAD Wissam

  BOURAHLA Yasser

29-11-2018

Table des matières

Table des figures

Liste des tableaux

CHAPITRE 1

INTRODUCTION

Objectifs et problématique

Talk about why we are gonna do this stuff.

Définitions

Some definitions of the concepts,techniques and algorithms that we will use

Une image sous différents angles

Talk about how an image can be represented(Color spaces, data structures) BRIEFLY

Notion de video

A video is a succession of images at some frame rate, lol

Opération sur les images

Notion de filtrage et convolutions

Filtre moyen

Filtre médiane

Érosion

Dilatation

NOT SURE IF WE'RE GONNA TALK ABOUT ALL OF THESE (BAH N3AMROU JEDDOU XD)

Conclusion

CHAPITRE 2

SOLUTION PROPOSÉES ET IMPLÉMENTATION

Outils utilisés

Environnement de travail

Langage

OpenCV

Qt framework

Schéma global du système

Le système se compose principalement de trois modules :

- D'abord le module de détection récupère l'image de la caméra pour y détecter les couleurs recherché et génère des informations sur le curseurs i.e sa position et son état.
- Le module de dessin reçoit les informations sur le curseur et les utilise pour générer l'image du dessin.
- Les deux modules précédents envoient leurs résultats à l'interface graphique pour l'affichage.

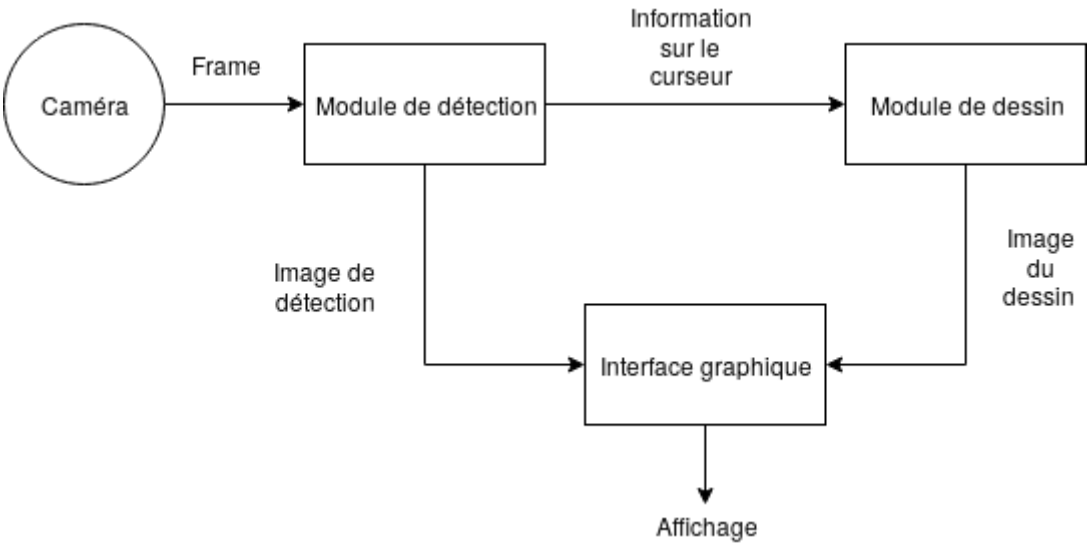

FIGURE 2.1 – Schéma du système

Détection de couleurs

La détection de couleurs se fait sur une image en HSV, ceci est dû au fait que HSV sépare la valeur de teinte de celle de la luminosité ce qui le rend plus robuste envers les changements de lumière.

Ainsi la détection d'une des deux couleur se fait en passant par tous les pixels de l'image et décider pour chaque pixel s'il a la couleur recherché ou non. La décision se fait en testant l'appartenance du pixel à un intervalle de détection. Pour chacune des valeur H,S,V du pixel, on vérifie si cette valeur appartient à un intervalle définit lors du choix de l'utilisateur de la couleur à détecter comme suit :

Soit la couleur choisie H1S1V1, et le pixel candidat HSV, les conditions que doit vérifier le pixel sont :

- $H1 - t \leq H \leq H1 + t$: l'intervalle de teinte dépend fortement de la couleur sélectionnée.
- $S1 * 0.6 \leq H \leq 255$: l'intervalle de saturation dépend légèrement de la couleur sélectionnée.
- $20 \leq H \leq 255$: On accepte un intervalle de luminosité large.

Après avoir sélectionner les pixels qui ont la couleur désirée, on calcule le centre de ces pixels. Ainsi la détection résultera en deux centres, un pour chaque couleur. La distance entre ces deux derniers détermine si le curseur est active. Si la distance est supérieur à un seuil donné le curseur est désactivé, il est active sinon. Quant à la position du curseur elle est représenté par le centre des deux points sus-cités.

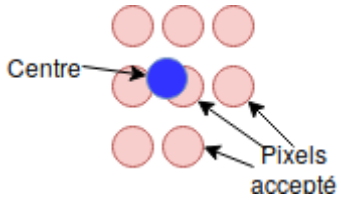

FIGURE 2.2 – Exemple centre de détection

Le problème qui se pose en se limitant à la détection par intervalle c'est qu'elle est extrêmement sensible au bruit. Un pixel aberrant jugé appartenir à l'intervalle va fausser la position du centre, et ainsi un suivi médiocre de la couleur.

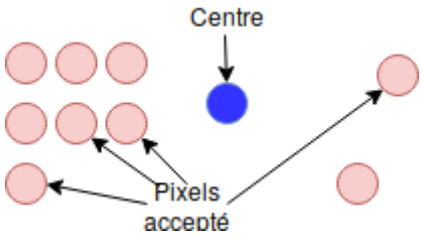

FIGURE 2.3 – Exemple centre de détection avec bruit

Élimination du bruit par regroupement

Pour remédier au problème rencontré. Nous avons opté à éliminer le bruit en regroupant les pixels proches qui sont acceptés. Avec cette méthode le bruit sera représenté par de petits groupes de pixels qu'on éliminera.
L'algorithme de regroupement fonctionne comme suit :

- Il prend en paramètre un seuil minimum pour jugé que deux pixels appartiennent au même groupe.
- Au début chaque pixel est dans un groupe à part.
- L'algorithme passe par plusieurs itérations de regroupement, dans une itération chaque groupe de pixel est fusionné avec un autre dont la distance est inférieur au seuil modifié* s'il existe, ainsi on regroupe les pixels proches entre eux dans un seul groupe.
- L'algorithme s'arrête quand les groupes ne changent pas après une itération, ceci implique que tous les groupes sont éloignés entre eux.

seuil modifié* : Comme le seuil donné en entré est défini pour regrouper deux pixels, il est donc inadapté pour le fusionnement de deux groupes contenant plus d'un pixel. Le seuil doit être donc ajusté en fonction de la taille des groupes. Nous avons choisi d'utiliser la formule suivante : $nouveauSeuil = (seuil/2) * (racine(tailleG1) + racine(tailleG2)$ avec $tailleG1$ et $tailleG2$ les tailles des groupes un et deux respectivement.
L'intuition derrière cette formule, c'est que la distance entre deux centres de carrés collés est égale à $1/2 * cote1 + 1/2 * cote2$, avec cote1 et cote2 les cotés des deux carrés.

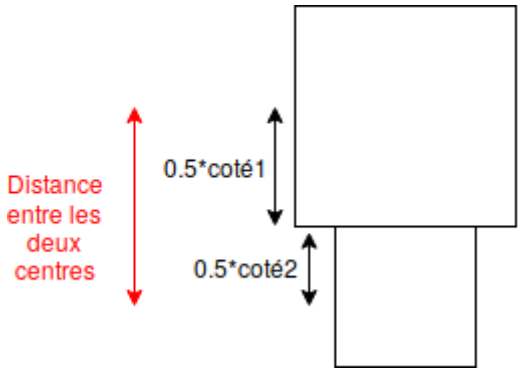

FIGURE 2.4 – Exemple distance entre deux carrés collés

Dans notre cas, la taille du coté d'un pixel est l'unité de mesure de distance. La distance entre deux pixels peut être écrite en fonction de la taille de son coté comme suit : $(distance/2)*(cotep+cotep)$ avec *cotep* la valeur d'un coté de pixel, comme cotep=1 (l'unité de mesure) la formule précédente se réduit en la valeur de distance.

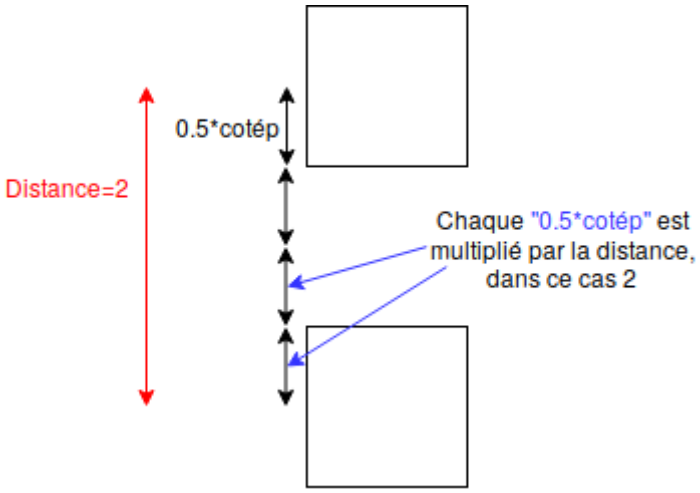

FIGURE 2.5 – Distance entre deux pixels représentant l'unité de mesure

Si on projette cela sur un groupe de pixels carré, la taille de son coté est égale à la racine du nombre de pixels, car le nombre de pixels représente la surface du carré, ainsi la formule devient : $(distance/2) * (racine(tailleG1) + racine(tailleG2)$.

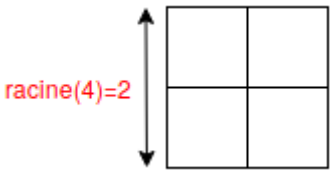

FIGURE 2.6 – Exemple de coté d'un groupe de pixels

Complexité La complexité de l'algorithme est de $O(n^2 * log(n))$, n étant le nombre de pixels acceptés. À chaque itération de l'algorithme pour chaque groupe on cherche un groupe avec lequel

on le fusionne, au pire cas on passe par tous les groupes, et donc une complexité quadratique par itération. On fait $log(n)$ itération au max, car un groupe qui n'est pas fusionné lors d'une itération sera enlevé dans les prochaines itérations, donc au pire cas le nombre de groupes qui résulte d'une itération est égale au nombre de groupe initial divisé sur deux, cas où on fusionne les groupes deux par deux, d'où la complexité logarithmique.
Il est à noter que l'opération de fusion se fait en $log(n)$ en utilisant une structure d'ensembles disjoints [1].

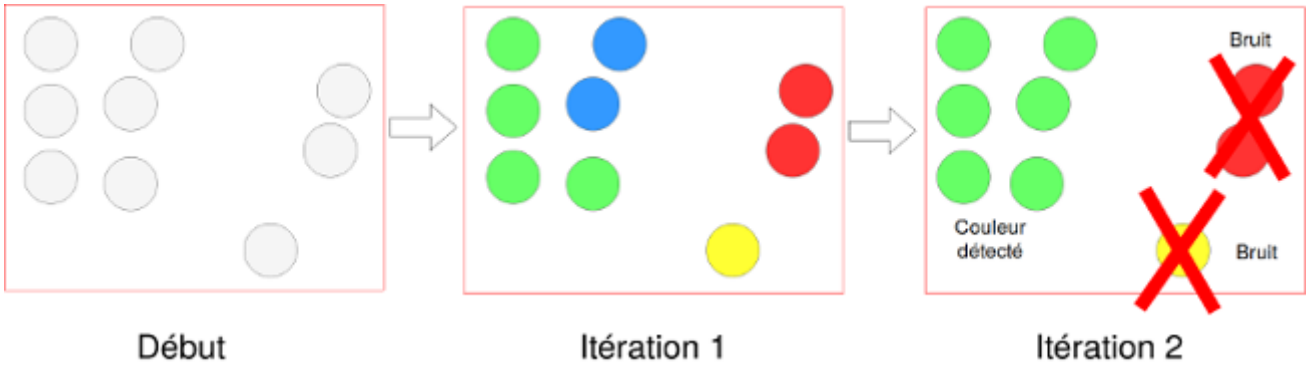

FIGURE 2.7 – Exemple déroulement de l'algorithme de regroupement

Inconvénients Le premier inconvénient est le temps de calcule de l'algorithme. Si le nombre de pixels détecté par la première partie est important, la complexité quadratique ne convient pas à une détection en temps réel.

Le deuxième inconvénient est le fait qu'on a besoin d'un seuillage pour faire le regroupement. Un seuil petit donne de bon résultat avec un grand intervalle de détection, il est robuste par rapport au bruit proche à l'objet qu'on suit. Par contre si l'intervalle de détection est petit, l'objet peut être divisé en deux groupes. Le contraire s'applique si le seuil est grand, un grand intervalle pourrait regrouper des pixels qui n'appartiennent pas à l'objet dans le même groupe que ce dernier.

Élimination du bruit par érosion et dilatation

Le principe de cette méthode est simple, on représente les pixels acceptés dans une image en niveau de gris, le pixel est blanc s'il est accepté et noir sinon. On applique une série d'érosions jusqu'à ce qu'on a que des pixels noir, et on garde la dernière image non nul. Dans cette dernière, les pixels non noir appartiennent probablement à l'objet qu'on veut détecter, avec l'hypothèse que l'arrière plan ne contient pas la couleur recherchée. La suite sera d'appliquer une série de dilatation tout en gardant en blanc que les pixels qui étaient blancs dans l'image d'origine, on s'arrête quand l'image ne change pas. Ainsi on est sur que les pixels en blanc après ces deux séries ont été détectés au préalable et qu'ils ont une forte chance d'appartenir à l'objet suivi.

1. plus d'information sur la structure d'ensembles disjoints : https ://en.wikipedia.org/wiki/Disjoint-set_data_structure

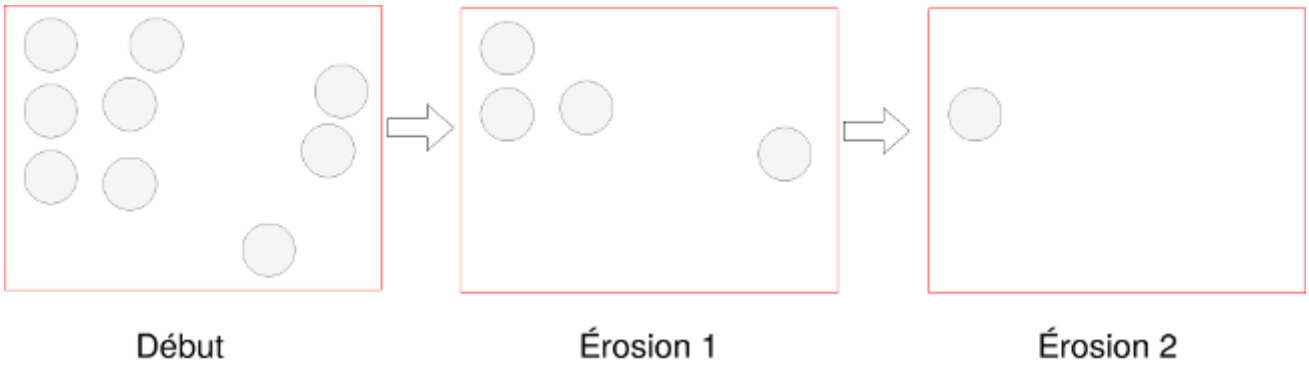

FIGURE 2.8 – Exemple application d'une série d'érosions

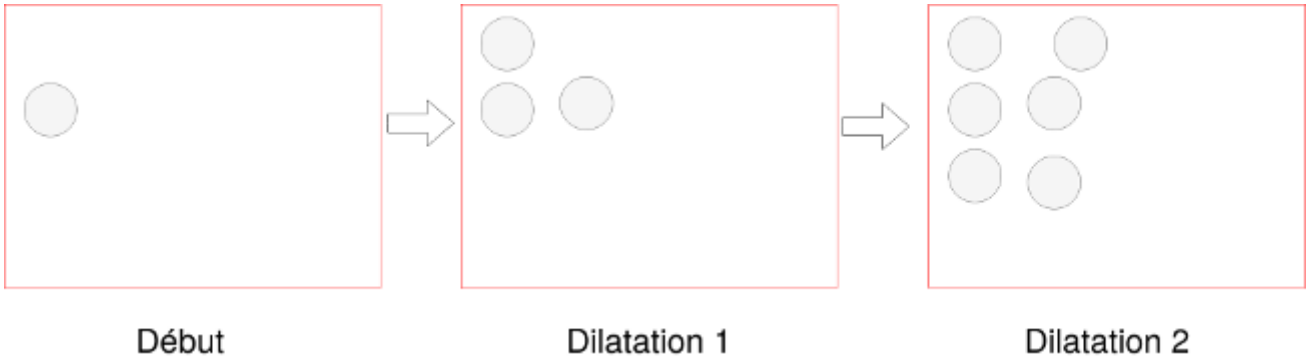

FIGURE 2.9 – Exemple application d'une série de dilatations

Conclusion

Kheliahli rak ta3ref xD

CHAPITRE 3

PRÉSENTATION DE L'APPLICATION

Dans ce chapitre nous allons détailler l'application qui utilise les méthodes présentées précédemment afin de permettre à un utilisateur de dessiner avec ses doigts.

Diagramme d'utilisation

L'application commence d'abord par donner la main à l'utilisateur pour choisir les couleurs à détecter. Ceci se fait en les mettant dans une zone définit par l'application afin qu'elle puisse récupérer les couleurs à suivre.

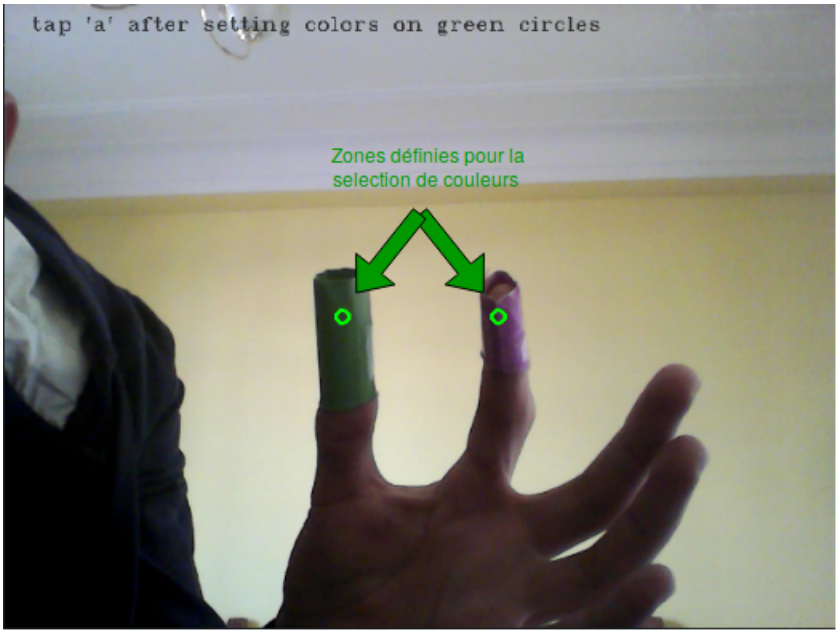

FIGURE 3.1 – L'étape initiation de couleurs à suivre

L'interface contient principalement 3 parties :

- Image récupéré par la caméra avec les centres des couleurs détectées.
- Espace de dessin.
- Options de l'application :
  - Changer la méthode de détection, i.e par regroupement ou par série d'érosions/dilatations.
  - Affichage des images de détections en niveau de gris.

Espace de dessin Cette espace est géré principalement par le module de dessin cité dans le chapitre précédent. Ce module utilise juste l'information sur le curseur, sa position et son état d'activation, pour déterminer l'action à prendre. L'espace de dessin est composé de deux parties :

- La partie paramétrage dans laquelle on peut changer les paramètres du dessin.
- La partie feuille de dessin, c'est dans cette partie que l'utilisateur va réaliser son dessin.

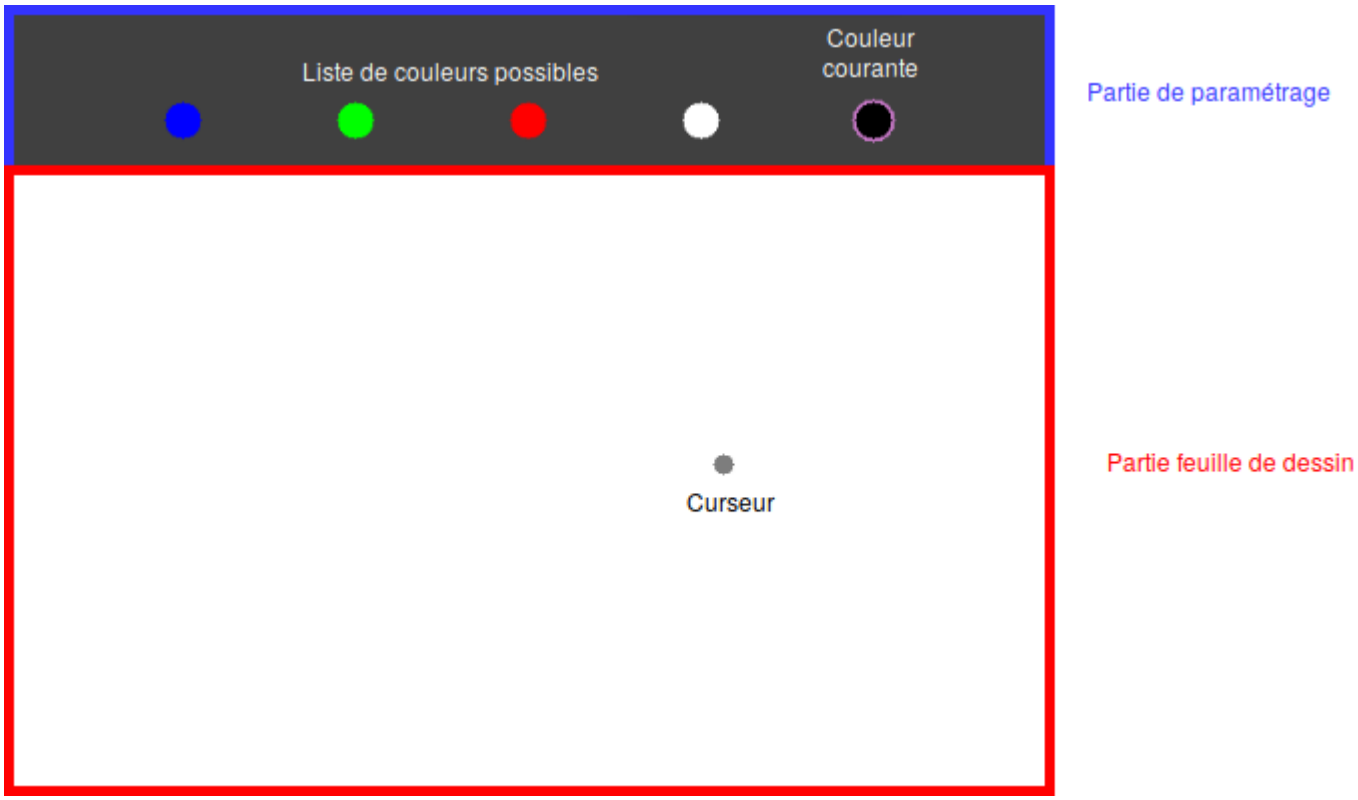

FIGURE 3.2 – L'espace de dessin

Il existe 3 types d'actions que le module de dessin peut prendre :

- Dessiner quand le curseur est active dans la partie feuille de dessin.
- Changer la couleur du curseur s'il détecte un clique sur une couleur de la partie paramétrage qui est différente de la couleur courante.
- Changer la taille du curseur s'il détecte un clique sur la couleur courante.

La dernière partie de l'interface graphique permet de visualiser comment les algorithmes de détection marche, par exemple on peut voir les groupes générés ainsi que leurs centres :

(a) A subfigure

(b) A subfigure

FIGURE 3.3 – A figure with two subfigures

Exemples d'utilisation

Limitations

CHAPITRE 4

CONCLUSION GÉNÉRALE