الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Études

présenté par

Omari Isslam

&

Bekertou Redouane

pour l'obtention du diplôme de Master en Électronique option Système de Vision et Robotique

Thème

Conception et réalisation d'un système de détection de défauts sur faïence en temps réel par la vision artificielle

Proposé par : Mr NAMANE Abderahmane

Année Universitaire 2015-2016

Remerciements

Nous tenons à remercier en premier lieu Dieu le tout puissant qui nous a dotés de toute la force nécessaire à l'aboutissement de ce mémoire.

Nous tenons à remercier notre promoteur Mr A.NAMANE Qui nous a fait bénéficier de ces nombreuses et riches connaissances durant toute la durée de ce travail.

Nous remercions les membres du jury, qui nous honorent par la lecture de notre mémoire et de leur présence, le jour de notre soutenance.

Nous remercions toutes les personnes qui nous ont soutenus et aidés de près ou de loin à l'élaboration du présent mémoire.

Nous tenons à exprimer notre reconnaissance également à l'ensemble des enseignants de l'Université de Blida 1 sans lesquels nous ne serions pas arrivés là.

Dédicaces

Je dédie ce mémoire

A mes chers parents ma mère et mon père

Pour leur patience, leur amour, leur soutien et leurs encouragements.

A mes frères.

A mes très chères amis et notre promotion " Système de Vision et Robotique"

Ainsi qu'a tous mes amis d'étude

Sans oublier tout les professeurs que ce soit du primaire, du moyen, du secondaire ou de l'enseignement supérieur.

ملخص: يقترح العمل المعروض في هذه المذكرة حلا واضحا لكشف البلاط المميزة بالعلامة (ذات نوعية رديئة) من قبل عامل في مصنع باستخدام الرؤية الحاسوبية . نحن نقدم البرنامج الذي يكشف البلاط المميز و يرسل إشارة إلى وحدة اردوينو لتشغيلها عند مرور البلاط المميزة بالعلامة ويتحقق المشروع باستخدام لغة سي بلس بلس ، المكتبة مكتبة برمجية مفتوحة للرؤية الحاسوبية وحدة اردوينو.

كلمات المفاتيح: الكشف، البلاط المميزة بالعلامة، إشارة.

Résumé : Le travail présenté dans ce mémoire propose une solution de détection de la faïence marquée par un feutre de couleur bien déterminée (mauvaise qualité) réalisé par un opérateur dans une usine de fabrication de la faïence en utilisant la vision par ordinateur. Nous introduisons un programme qui suite à la détection du défaut envoie, un signal pour actionner le module Arduino .Le projet est réalisé en utilisant le langage C++, la bibliothèque OpenCV et module Arduino.

Mots clés : Détection, faïence marquée, signal.

Abstract: The work presented in this paper offers a solution for detecting earthenware marked by an indeterminate color felt (poor quality) directed by an operator in a pottery factory using computer vision. We introduce a program who detects the earthenware marked and sends a signal to the Arduino to activate .The project is implemented using the C ++ language, the OpenCV library and Arduino unit.

Keywords : detection, the distinctive tiles, signal.

Listes des acronymes et abréviation

PAO : Publication assistée par ordinateur

RVB : Rouge, Vert, Bleu

CMJN : Cyan, Magenta, Jaune

TSL : Teinte, Saturation, Luminance

β: élément structurant

ppp : points par pouce

OCR : Optical character recognition

CMOS : Complementary metal oxide semi-conductor

IDE : environnement de développement

OpenCV : Open Computer Vision

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

La vision est le sens qui nous fournit le plus d'informations sur le monde extérieur, elle nous permet de voir, de décrire, et de reconnaître les objets qui sont présents dans une scène.

L'être humain est capable de reconnaître les objets rapidement, avec une très grande précision, cela est réalisé grâce au système visuel humain qui est capable d'interpréter les informations sensorielles, c'est-à-dire les informations fournies par l'environnement au système visuel à un instant donnée. L'interprétation est réalisée grâce au cerveau, qui peut reconnaître et situer les objets, détecter le mouvement.

Avec la naissance des machines de calcul, des recherches scientifiques sont effectuées, en essayant de concevoir une machine qui peut remplacer le système de vision humain, pour arriver à des résultats similaires à la vision humaine.

La vision par ordinateur consiste à reproduire les résultats obtenus par la vision humaine sur un ordinateur en utilisant des moyens informatiques en remplaçant l'œil par une caméra et le cerveau par un ordinateur.

Ce projet utilise la détection des couleurs en utilisant le traitement d'image pour élaborer une solution au problème de qualité de faïence dans une chaine de fabrication du la faïence, de façon à faire détecter la couleur utilisé au marquage et arriver à changer cette couleur (mettre la couleur qu'on veut) et faire une signalisation lors de passage de faïence marquée.

L'objectif de ce projet de fin d’études est de développer une méthode de détection basée sur la vision par ordinateur pour détecter des faïences marquées par un opérateur. Le mémoire est organisé en trois chapitres.

Le premier chapitre présente des généralités sur le traitement d'images. Le deuxième chapitre présente les étapes de la chaine de détection et la signalisation lors le passage des faïences marquées, en détaillant notamment les algorithmes utilisés. Le troisième chapitre sera consacré à la partie application et résultats.

Chapitre 1 Généralités sur le traitement d'image

1.1 Introduction

Avec la parole, l'image constitue l'un des moyens les plus importants qu'utilise l'homme pour communiquer avec autrui. C'est un moyen de communication universelle dont la richesse du contenu permet aux êtres humains de tout âge et de toute culture de se comprendre.

C'est aussi le moyen le plus efficace pour communiquer, de ce fait, le traitement d'images est l'ensemble des méthodes et techniques opérant sur celles-ci, dans le but de rendre cette opération possible, plus simple, plus efficace et plus agréable, d'améliorer l'aspect visuel de l'image et d'en extraire des informations jugées pertinentes.

Dans ce chapitre nous présentons quelques notions de base du domaine de traitement d'image numérique tels que : la définition de l'image numérique, les types d'images, formats d'images, caractéristiques d'images, système de traitement d'images, filtrage, binarisation, segmentation et en fin quelques exemples concrets du traitement d'images.

1.2 Image numérique

Une image est la représentation d'un être ou d'une chose obtenue par exemple par la photographie, la vidéo ou l'utilisation d'un logiciel spécialisé. Elle est dite numérique lorsque sa sauvegarde est obtenue sous forme binaire. Donc image numérique fait appel à l'informatique. Chaque image numérique est constituée d'un nombre donné de lignes. Chaque ligne comporte un nombre de point donnés. L'ensemble constitue une matrice. Ces points sont dénommés pixel (de l'anglais Picture élément et noté souvent px). Chaque case de cette matrice contient des nombres caractéristiques à la couleur attribuée au pixel.

1.2.1 Classification des images numériques

On peut classifier les images numériques en deux catégories

a ***la représentation numérique***

Il existe deux types d'images numériques : les images matricielles et les images vectorielles. Une image matricielle est formée d'un tableau de points ou pixels. Plus la densité des points sont élevée, plus le nombre d'informations est grand et plus la résolution de l'image est élevée. Corrélativement la place occupée en mémoire et la durée de traitement seront d'autant plus grandes. Les images vues sur un écran de télévision ou une photographie sont des images matricielles. On obtient également des images matricielles à l'aide d'un appareil photo numérique, d'une caméra vidéo numérique ou d'un scanner. Dans une image vectorielle les données sont représentées par des formes géométriques simples qui sont décrites d'un point de vue mathématique. Ces images sont essentiellement utilisées pour réaliser des schémas ou des plans. Les logiciels de dessin industriel fonctionnent suivant ce principe ; les principaux logiciels de traitement de texte ou de PAO (publication assistée par ordinateur) proposent également de tels outils. Ces images présentent 2 avantages : elles occupent peu de place en mémoire et peuvent être redimensionnées sans perte d'information.

b la représentation des couleurs

- ***Le modèle RVB***

C’est le plus utilisé pour le maniement des images est l'espace colorimétrique rouge, vert, bleu (RVB ou RGB - red green blue). Cet espace est basé sur une synthèse additive des couleurs, c'est-à-dire que le mélange des trois composantes R, V, et B à leur valeur maximum donne du blanc, à l'instar de la lumière (voir figure1.1).

Le mélange de ces trois couleurs à des proportions diverses permet de reproduire à l'écran une part importante du spectre visible.

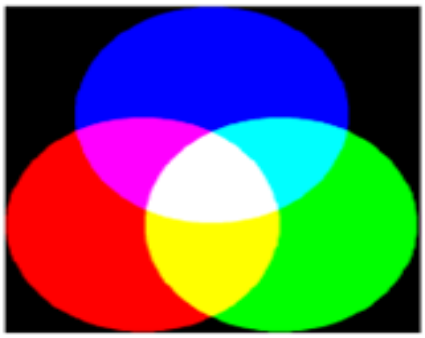


Figure 1.1. La synthèse additive.

Le système RGB peut être représenté sous la forme d’un cube (voir figure 1.2), les sommets du triangle représentent les couleurs primaires, Rouge (R), Vert(v) et Blue(B).

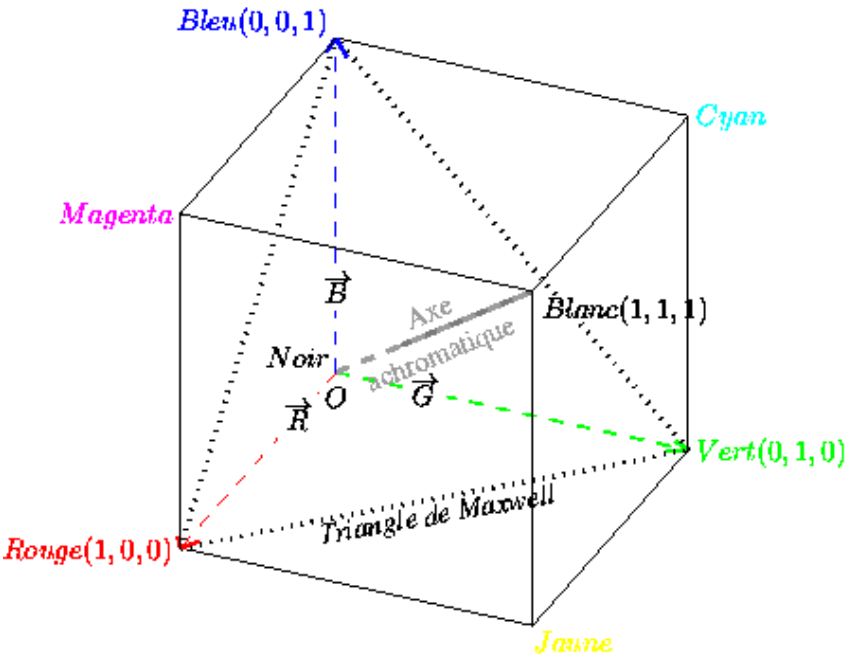


Figure 1.2. Le modèle de couleur RVB.

- ***Le modèle CMJN***

CMJN : Cyan, Magenta, Jaune, (Noir) ou CMYK (Cyan, Magenta, Yellow, Black)

Le modèle est basé sur le principe de la synthèse soustractive (voir figure1.3). Il est utilisé en imprimerie, en peinture. Le noir pur ne pouvant être obtenu par superposition des encres cyan, magenta et jaune, les imprimeurs rajoutent une 4ème encre noire « Impression en quadrichromie ».

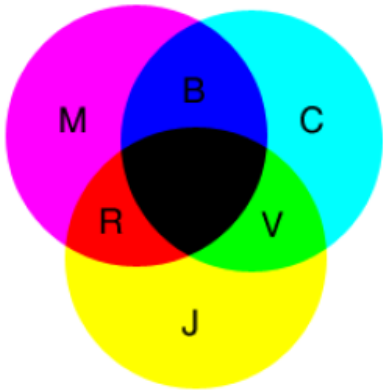


Figure 1.3. La synthèse soustractive.

- ***Le modèle TSL :***

Le modèle TSL (acronyme de Teinte, Saturation, Luminosité) est le plus intuitif de tous les modèles colorimétriques que nous étudierons. Il est basé sur le ressenti de la perception humaine d'où son nom de modèle perceptuel. Chaque critère de couleur est clairement séparé, ce qui en fait le modèle le plus pratique pour la retouche d'image ou l'ajustement des couleurs.

Les trois critères qui caractérisent le TSL (HSL en anglais pour Hue, Saturation et Lightness) sont la teinte, mesurée par un angle de 0 à 360° autour de la roue chromatique, la saturation, qui reflète bien la notion intuitive de coloration, car elle va des couleurs vives vers le gris. Et enfin, la luminosité, mesurée entre le noir (pas de lumière ou valeur 0) et le blanc (lumière maximum ou valeur 1) (voir figure 1.4).

La notion de couleur dans cet espace est plus intuitive des couleurs pour un artiste.

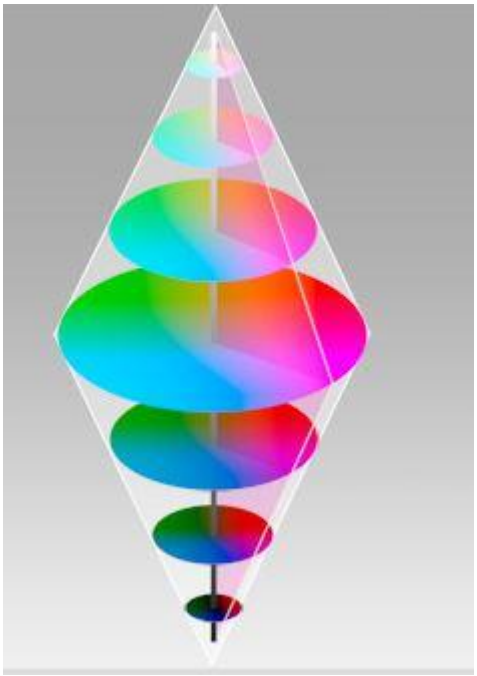


Figure 1.4. Le modèle TSL.

1.3 Les type d'images

1.3.1 Images en teintes (ou niveaux) de gris

On ne code ici plus que le niveau de l'intensité lumineuse, généralement sur un octet (256 valeurs). Par convention, la valeur zéro représente le noir (intensité lumineuse nulle) et la valeur 255 le blanc (intensité lumineuse maximale) comme l'indique la figure 1.5.

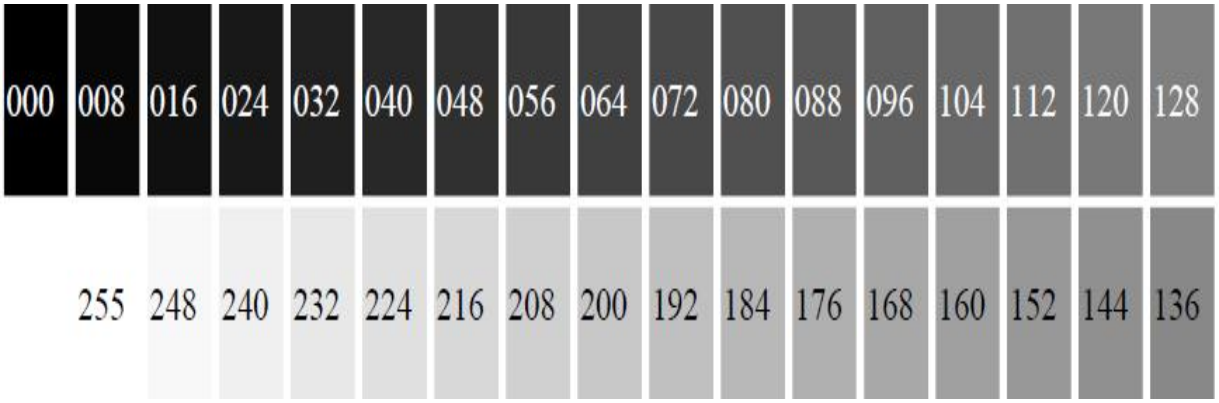


Figure 1.5. Valeurs des niveaux de gris et teintes correspondantes.

Ce procédé est fréquemment utilisé pour reproduire des photos en noir et blanc ou du texte dans certaines conditions (avec utilisation d'un filtre pour adoucir les contours afin d'obtenir des caractères plus lisses), comme l'indique figure 1.6.

Figure 1.6. Image à niveaux de gris.

1.3.2 Images à palettes

a Images en 256 couleurs (8 bits)

Est une image digitale dont chaque pixel peut gérer 8 bits d'information de couleurs cette image peut donc reproduire 2^8 soit 256 couleurs ou niveaux de gris. C'est le cas de format GIF. Les images sauvegardées dans ce format n'occupent pas beaucoup de place en mémoire ou sur disque mais n'offrent pas non plus une grande définition (qualité).

b Images 24 bits

Est qualifiée de '' true color '' (vrais couleurs ou photo-réaliste). Dans ce format, 24 bits sont assignés à chaque pixel, soit 8 bits de couleurs primaire R,G,B, ce qui représente une gamme de 2^{24} soit plus de 16.7 millions de couleurs. Tous les périphériques utilisés en infographie sont capables de supporter ce format (écran, scanner, APN, imprimante…).

1.4 Format d'image

Un format d'image est une représentation informatique de l'image, associée à des informations sur la façon dont l'image est codée et fournissant éventuellement des indications sur la manière de la décoder et de la manipuler. La plupart des formats sont composés d'un en-tête contenant des attributs (dimensions de l'image, type de codage, LUT, etc.), suivi des données (l'image proprement dite). La structuration des attributs et des données diffère pour chaque format d'image. De plus, les formats actuels intègrent souvent une zone de métadonnées (metadata en anglais) servant à préciser les informations concernant l'image comme : la date, l'heure et le lieu de la prise de vue, les caractéristiques physiques de la photographie (sensibilité ISO, vitesse d'obturation, usage du flash…) [5].

1.5 Caractéristiques d'une image numérique

L'image est un ensemble structuré d'informations caractérisé par les paramètres suivants:

1.5.1 Définition

C'est la quantité de pixels constituant une image numérique. On l'obtient en multipliant le nombre de pixels en Largeur (colonnes) par le nombre de pixels en Hauteur (lignes).
Elle s'exprime donc en PIXELS. On parle de définition d'un écran ou d'un capteur matriciel numérique (appareil photographique numérique APN). EX : Une image possédant 10 colonnes et 11 lignes aura une définition de 10 x11.

1.5.2 Résolution

C'est le nombre de points contenu dans une longueur donnée (en pouce). Elle est exprimée en **points par pouce** (PPP, en anglais: DPI pour Dots Per Inch). Un pouce mesure 2.54 cm, c'est une unité de mesure britannique. La résolution définit la netteté et la qualité d'une image. Plus la résolution est grande (c'est-à-dire plus il y a de pixels dans une longueur de 1 pouce), plus votre image est précise dans les détails. La résolution permet ainsi d'établir le rapport entre la définition en pixels d'une image et la dimension réelle de sa représentation sur un support physique (écran, papier...).

1.5.3 Bruit

A chaque étape de l'acquisition d'une scène, des perturbations vont dégrader la qualité de l'image. Ces perturbations sont regroupées sous le nom de « bruit d'image». Il existe différentes causes de bruit. On peut notamment citer le contexte d'acquisition (lumière, perturbation de capteurs), les étapes d'échantillonnage et de quantification qui apporte leur propre perturbation, mais aussi l'étape de transmission (perte ou corruption de données). On peut bien remarquer ce bruit notamment sur la transmission analogique de la télévision.

1.5.4 Histogrammes

Un histogramme est une courbe statistique indiquant la répartition des pixels selon leur valeur. L'histogramme est très utile pour contrôler l'exposition d'une image.

- A l'acquisition, il permet de contrôler et affiner les réglages de prise de vue.
- Pour le traitement, il permet de corriger ou modifier l'exposition de l'image, ainsi que l'échelle des couleurs.

Par exemple : améliorer le contraste, corriger une image sous-exposée, renforcer la composante rouge, corriger la non-linéarité du capteur....

- En utilisant judicieusement l'histogramme, on peut faire apparaître les détails et les nuances acquises par le capteur et présentes dans le fichier, mais non visibles à l'œil.

a Histogrammes des images en niveaux de gris

Il indique pour chaque valeur entre le noir (0) et le blanc (255), combien il y a de pixels de cette valeur dans l'image; En abscisse (axe x) : le niveau de gris (de 0 à 255); en ordonnée (axe y) : le nombre de pixels. Les pixels sombres apparaissent à gauche de l'histogramme, les pixels clairs à droite de l'histogramme et les pixels gris au centre de l'histogramme (voir figure1.7)

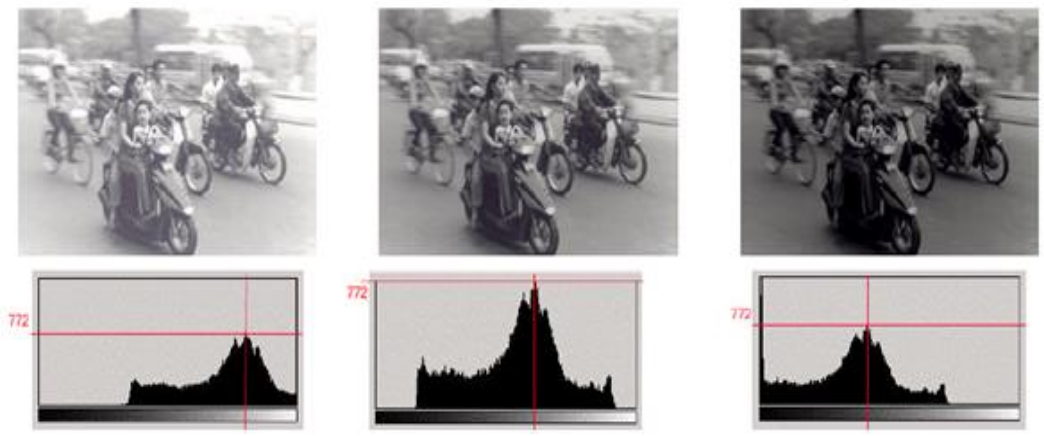


Figure 1.7. Exemple d'histogrammes pour une même image plus ou moins bien exposée.

b Histogramme des images couleurs

Pour les images couleurs, plusieurs histogrammes sont utilisés (voir figure1.8)

· L'histogramme des luminances

· les 3 histogrammes de chacune des composantes R,V,B.

Figure 1.8. Histogrammes des trois plans couleur.

1.5.5 Contours et textures

Les contours représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative. Les textures décrivent la structure de ceux-ci. L'extraction de contour consiste à identifier dans l'image les points qui séparent deux textures différentes.

1.5.6 Luminance

La luminance désigne le signal qui détermine les valeurs de contraste d'une image, du noir le plus profond jusqu'au blanc le plus pur. La couleur correspond à l'autre partie du signal, appelée chrominance. Cette grandeur photométrique fondamentale caractérise l'émission intrinsèque d'une source, indépendamment de sa géométrie. La luminance est le flux émis par unité de surface apparente et par unité d'angle solide. On montre que l'éclairement du plan image produit par un système

d'imagerie est proportionnel à la luminance de la source. En particulier, l'impression visuelle est proportionnelle à la luminance des objets car l'éclairement de la rétine de l'œil est proportionnel à la luminance.

1.5.7 Contraste

Le contraste d'une photo est lié à la notion de luminosité, instinctivement une image parait contrastée lorsqu'elle mêle des zones très lumineuses et des zones très sombres. Le contraste est une notion qui revient fréquemment lorsque l'on s'intéresse à la photographie: on juge le contraste d'une image, des logiciels permettent de le régler et même les capteurs et les objectifs ont un rôle à jouer. Ce billet aborde toutes ces notions.
Il existe deux approches pour calculer le contraste, la première consiste à regarder l'écart entre sa luminance minimale et sa luminance maximale. La formule est la suivante:

$$contraste = \frac{lum(max) - lum(min)}{lum(max) + lum(min)} \qquad (1.1)$$

Où lum(Max) désigne la luminance maximale de l'image et lum(Min) sa luminance minimale.
Remarque: la luminance désigne l'intensité lumineuse, je l'ai mis pour la formule mais je parlerai plus volontiers (et à tord) de luminosité.
Avec cette formule une photo est très contrastée si son histogramme est très étendu.

Une seconde approche, plus fine, s'intéresse au contraste local de tous les points de l'image et calcul la moyenne pour l'ensemble de l'image. On utilise la même formule qu'au dessus mais les Min et Max sont limités au voisinage de chaque point (par exemple dans un rayon de 2 pixels).
Avec cette méthode une photo est très contrastée si son histogramme est très étendu et qu'il y a beaucoup de zones claires et sombres, avec des transitions franches entre les deux.

La valeur même du contraste d'une photo n'est pas utilisée, mais le ressentit est bien réel.

La photo ci-dessous (figure 1.9) a un faible contraste, son histogramme est très étroit et la luminosité très homogène.

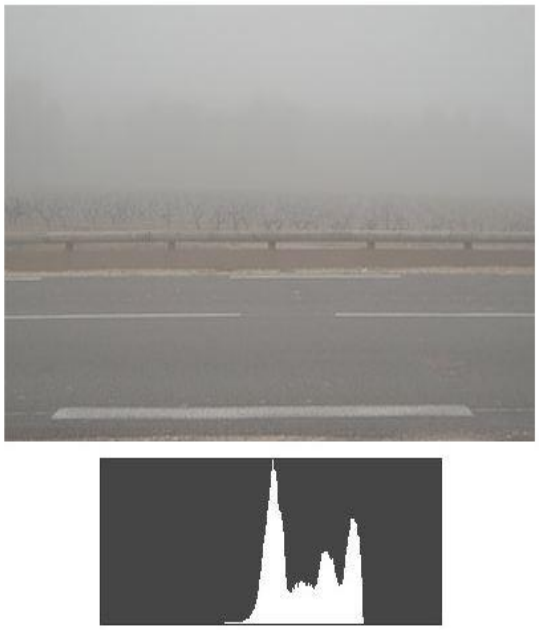


Figure 1.9. Image à faible contrasté**.**

La photo suivante (figure1.10) a un fort contraste, l'histogramme est très étendu et les zones clairs comme les zones sombres sont très présents. La sensation de contraste est renforcée par la présence de silhouettes très sombre sur fond très lumineux.

Figure 1.10. Image à fort contraste.

1.6 Système de traitement d'image

Dans le contexte de la vision artificielle, le traitement d'images se place après les étapes d'acquisition et de numérisation, et se compose de plusieurs étapes:

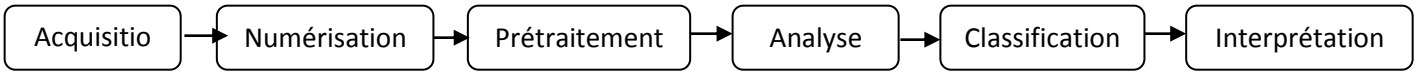


Figure 1.11. Système de vision artificielle.

1.6.1 Acquisition et numérisation

C'est le mécanisme qui permet l'obtention d'une image numérique (représentée par une matrice) à deux dimensions à partir d'une scène à trois dimensions, en passant par un système optique, l'image continue f(x,y) est approximée par des échantillons qui sont obtenus par discrétisation des coordonnés (x,y) (ce qu'on appelle l'échantillonnage) et la discrétisation des amplitudes de ces points là (c'est la quantification). La représentation obtenue ne peut être parfaite à cause du bruit introduit dans l'image lors de son acquisition [10].

1.6.2 Prétraitement

Le prétraitement Permet d'amélioré la qualité visuelle de l'image et d'éliminer le bruit par des techniques de modification de l'histogramme et le filtrage.

1.6.3 Analyse de l'image

Elle se manifeste par le partitionnement des diverses éléments de l'image en région connexes ayant même propriétés. Les régions peuvent être caractérisées par les pixels qu'il est composé il s'agit alors à la segmentation région ou bien elles peuvent être caractérisées par les pixels des frontières, il s'agit alors de la segmentation en contours.

1.7 Filtrage

Le filtrage est une opération qui a pour but d'extraire une information ou d'améliorer l'aspect de l'image, par exemple en éliminant un bruit (lignage, speckle des images radar, etc.), ou en améliorant les contours d'une image floue.

1.7.1 Filtres linéaires

Un filtre est une transformation mathématique (appelée *produit de convolution*) permettant, pour chaque pixel de la zone à laquelle il s'applique, de modifier sa valeur en fonction des valeurs des pixels avoisinants, affectées de coefficients. Parmi les filtres linéaires, 3 catégories de filtres sont envisagées qui se distinguent par des

propriétés dans le domaine fréquentiel; nous verrons donc plus loin la raison des appellations utilisées :

- **filtres "passe-bas"** : ils sont utilisés pour atténuer les détails de l'image qui "tranchent" nettement avec le reste de l'image : bords, caractéristiques particulières notamment. Le résultat de l'application d'un filtre "passe-bas" sera une image "lisse".
- **filtres "passe-haut"** : contrairement aux précédents, ils sont utilisés pour atténuer les caractéristiques "neutres" et mettre en évidence les détails qui "tranchent".

- **filtres "passe-bande"** : ils ont peu d'intérêt pour le rehaussement d'image, mais par contre ils sont utilisés dans le cas de la restauration d'image.

Le principe du filtrage linéaire est de remplacer le niveau d'un pixel par une combinaison linéaire des niveaux des pixels environnants. Cette combinaison linéaire est usuellement représentée par un masque.

a Filtres "passe-bas"

- ***Filtre moyenneur (lissage)***

C'est un filtre passe-bas c à d qu'il laisse passer les basses fréquences (figure 1.12)

•Lisse l'image (effet de flou)

• Réduit le bruit

•Réduit les détails

Filtre dont tous les coefficients sont égaux (chaque pixel est remplacé par la moyenne de ses voisins)

1/9	**1/9**	**1/9**
1/9	**1/9**	**1/9**
1/9	**1/9**	**1/9**

Ou **1/9***

1	**1**	**1**
1	**1**	**1**
1	**1**	**1**

3*3

Masque du filtre

Figure 1.12. Application du filtre moyenneur sur une image.

- ***Filtre gaussien***

Le filtre Gaussien est un filtre isotrope spécial avec des propriétés mathématiques bien précises. σ caractérise l'écart type soit la largeur du filtre : autrement dit la largeur du filtre en partant du point central est égal à 3σ arrondi à l'entier supérieur.

La réponse impulsionnelle :

$$h(x,y) = \left(\frac{1}{2\pi\sigma^2}\right) \times exp\left(\frac{x^2+y^2}{2\sigma^2}\right) \qquad (1.2)$$

L'effet de ce filtre sur l'image est assez similaire au filtre moyenneur, mais la moyenne est pondérée : les pixels près du centre ont un poids plus important que ceux que les autres. En général, un filtre Gaussien avec $\sigma<1$ est utilisé pour réduire le bruit, et si $\sigma>1$ c'est dans le but de flouter volontairement l'image. Il faut noter que plus σ est grand, plus la cloche Gaussienne est large et plus le flou appliqué à l'image sera marqué.

Voici une application du filtre sur une image au niveau de gris (figure 1.13).

Figure 1.13. Application du filtre gaussien sur l'image.

1.7.2 Filtres non linéaires

Les filtres linéaires ont des défauts. Ainsi, le filtre moyenneur tend à rendre floues les frontières de l'image traitée. Les filtres non linéaires peuvent éviter ces problèmes. Ainsi, ils peuvent homogénéiser l'image tout en préservant ses frontières.

a Filtre médian

Le filtre médian est un filtre numérique non linéaire, souvent utilisé pour la réduction de bruit. La réduction de bruit est une étape de prétraitement classique visant à améliorer les résultats de traitements futurs (détection de bords par exemple). La technique de filtre médian est largement utilisée en traitement d'images numériques car il permet sous certaines conditions de réduire le bruit tout en conservant les contours de l'image.

L'idée principale du filtre médian est de remplacer chaque entrée par la valeur médiane de son voisinage.

Par exemple, si on considère ces neufs pixels, dont une valeur aberrante (ici 111) :

5	**6**	**7**
6	**111**	**8**
7	**8**	**9**

Le filtre médian va considérer les valeurs du voisinage par valeurs croissantes :

5	**6**	**6**	**7**	**7**	**8**	**8**	**9**	**111**

Et prendre la valeur médiane, ici la valeur 7. La sortie du filtre donnera donc :

5	**6**	**7**
6	**7**	**8**
7	**8**	**9**

Figure 1.14. Principe du filtre médian.

Ce qui a permis de remplacer la valeur aberrante par une valeur "de consensus" entre les valeurs voisines. Plus généralement ce filtre permet d'éliminer les valeurs aberrantes sans se limiter à faire un calcul de moyenne qui aura tendance à contaminer les valeurs voisines avec cette valeur aberrante et flouter l'image.

Le filtre médian respecte le contraste de l'image (si on multiplie toutes les valeurs par une constante positive, l'ordonnancement des valeurs est inchangé) et la luminosité de l'image (ajouter une constante ne modifie pas l'ordonnancement non plus).

b Filtre maximum

Le filtre maximum est un filtre non linéaire utilisé pour supprimer le bruit de type "poivre" dans les images, il fonctionne sous le même principe que le filtre médian, sauf qu'il utilise la valeur maximale au lieu que la valeur médiane. Pour chaque pixel (i , j) :

$$\mathrm{fmax}(x, y) = \max(i, j) \in \mathrm{voisins}(x, y)\ f(i, j) \qquad (1.3)$$

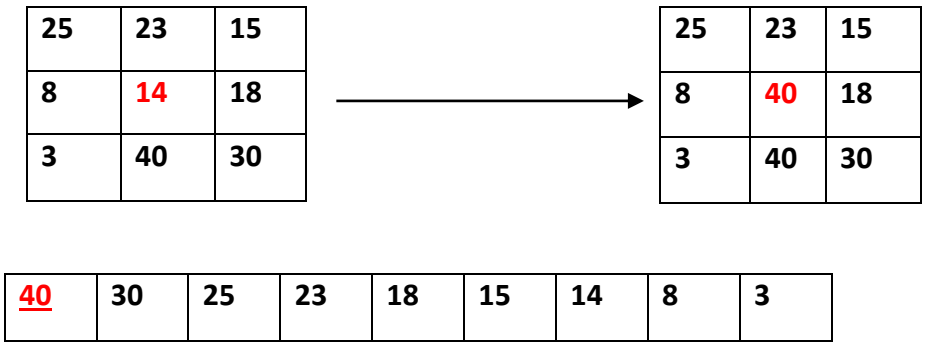


Figure 1.15. Principe du filtre maximum.

c Filtre minimum

Pour les images contenant le bruit de type 'Sel', c'est le filtre minimum qui est appliqué, comme le filtre médian et le filtre maximum, le filtre minimum est un filtre non linéaire qui fonctionne sous les mêmes principes que le filtre médian, sauf qu'il utilise la valeur minimale. Pour chaque pixel (i , j) :

$$fmin(x,y) = min(i,j) \in voisins(x,y)\ f\ (i,j) \qquad (1.4)$$

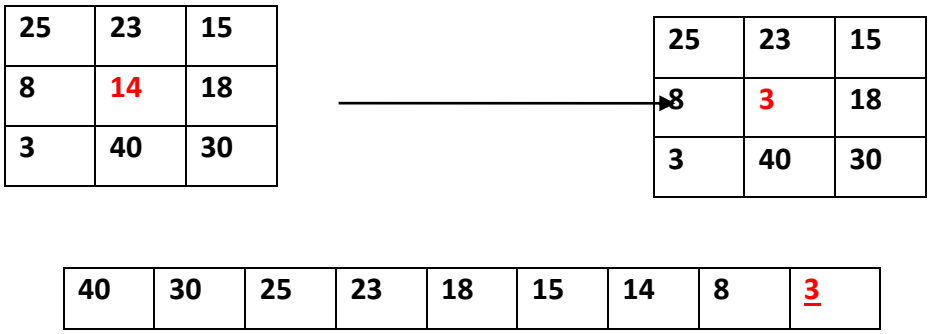


Figure 1.16. Principe du filtre minimum.

Médian, min et max sont appelés filtres de rang.

1.8 Binarisation

La binarisation d'image est la méthode la plus simple de la segmentation d'image. À partir d'une image à niveau de gris, le seuillage d'images peut être utilisé pour créer une image comportant uniquement deux valeurs, noir ou blanc (monochrome).

On remplace un à un les pixels d'une image par rapport à une valeur seuil fixée (par exemple 123). Ainsi, si un pixel à une valeur supérieure ou égale au seuil (par exemple 150), il prendra la valeur 255 (blanc), et si sa valeur est inférieure (par exemple 100), il prendra la valeur 0 (noir).

Figure 1.17. Exemple d'une binarisation.

Avec une image en couleur, on fera de même avec les trois composantes rouge, vert et bleu. Il y aura ainsi huit couleurs possibles pour chaque pixel : blanc, noir, rouge, vert, bleu, magenta, jaune et cyan.

Figure 1.18. Binarisation d'une image couleur.

1.8.1 Binarisation globale

Le principe du la binarisation globale est d'utiliser une valeur seuil pour tous les pixels à partir de laquelle on peut choisir à quelle classe le pixel appartient. La transformée peut s'écrire ainsi :

$$\forall i,j \in N \times M \; I(i,j) = \begin{cases} 255 & si \;\; \mathrm{f(i,j)} > \mathrm{S} \\ 0 & sinon \end{cases} \qquad (1.5)$$

- N ×M: Taille de l'image.
- I : l'image d'entrée.
-f : image binarisée.
- S : seuil de binarisation.

Cependant, cette technique pose beaucoup de problème. Le premier est de définir le seuil. La grande majorité des techniques utilisent l'histogramme des niveaux de gris pour choisir le seuil à appliquer.

Cependant, à cause du bruit, il n'est pas toujours facile de déterminer le seuil. Wu et al. Proposent alors de filtrer l'image avec un filtre passe-bas. Ils expliquent que le bruit sur le fond produit plus de hautes fréquences que le texte [7]. Cela permet bien souvent de séparer les deux modes de l'histogramme, et donc facilite le choix du seuil. Mais cette solution ne permet pas de trouver un seuil global (puisqu'il n'existe pas, figure 1.19), dans le cas d'une mauvaise illumination du document, ou dans le cas où le texte passerait de noir sur fond blanc à blanc sur fond noir. Pour pallier à ces problèmes, il fallut trouver des techniques permettant d'adapter localement le niveau du seuil.

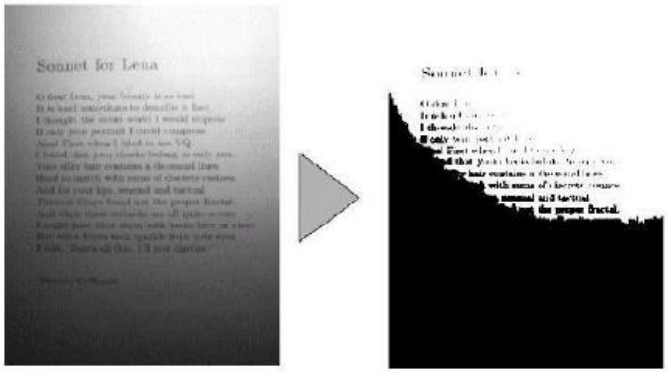


Figure 1.19. Problème de la binarisation globale.

1.8.2 Binarisation locale

Le premier à proposer une technique donnant de bons résultats fut Bernsen en 1986 [8], avec principe d'utiliser une étude localisée autour du pixel pour déterminer quel seuil utiliser. Pour réaliser cette méthode locale, les techniques utilisent une fenêtre d'étude centrée sur le pixel à étudier. Cette fenêtre peut avoir différentes tailles, souvent en fonction de la taille moyenne du texte dans le document.

$$s(i,j) = \big(max(i,j) + min(i,j)\big)/2 \qquad (1.6)$$

Avec :

- S(i, j) : seuil à appliquer pour le point i, j.

-Max(i, j) : valeur du niveau de gris maximal dans une fenêtre centré - en (i, j) de taille N×M.

- Min(i, j) : valeur du niveau de gris minimal dans une fenêtre centré en (i, j) de taille N ×M.

Cependant, ce filtre est très sensible au bruit du fond. À cause de la prise en compte du maximum et du minimum uniquement, dans le cas où la fenêtre est uniquement sur du fond, le bruit sera interprété comme objet, car le seuil sera bas.

La même année, Niblack proposa une méthode similaire sur le principe, mais prenant en compte d'autres paramètres [9].

1.9 Opérations morphologiques

La morphologie mathématique (MM) emploie les opérateurs Intersection, Union, Inclusion et Complément. L'image transformée est en général simplifiée par rapport à l'image origine, ce qui correspond donc à une perte d'information. Les principales caractéristiques de l'image originale sont cependant toujours présentes. Les mesures sont ensuite effectuées sur cette image simplifiée. Les opérations de morphologie mathématique s'appliquaient au début aux images binaires et aux images "monocanal" et avec les années 1970-1980 la MM a été généraliser pour les images à niveau de gris et a connu un extension des principaux concepts (dilatation, érosion, etc.), cette généralisation a abouti à de nouveaux opérateurs, tels que les gradients morphologiques et "la ligne de partage des eaux" (segmentation) [3]. Elles se font avec un élément structurant (β), caractérisé par sa dimension, sa forme et son centre. Chaque objet de l'image est comparé à (β) en déplaçant (β) en tout pixel de l'image.

Ainsi, dans le cas d'une image binaire, pour chaque position de (β), il peut être détecté si (β) est entièrement inclus dans une zone "pure" de 0 ou de 1, ou une zone qui contient des 0 et des 1. L'élément (β) définit le voisinage d'analyse du pixel traité. Ce voisinage peut être un rectangle, un cercle, un hexagone, etc. Parmi ces opérations morphologiques il y a la dilatation, l'érosion, l'ouverture et la fermeture mathématiques.

1.9.1 Dilatation

La dilatation est une opération qui « grandit » ou « épaissit » des objets dans une image binaire. D'un point de vue ensembliste, le résultat de la dilatation de l'ensemble A (Image) par l'ensemble β (élément structurant) est l'addition de Minkowski : c'est l'ensemble des points tels que lorsque β est centré sur un de ces points il y a une intersection non vide entre A et β [12] [4].

$$DilB(X) = \{Bp \setminus p\ X\} \quad (1.7)$$

1.9.2 Érosion

L'érosion est une opération qui « rétrécit » ou « amincit » les objets d'une image binaire. La définition mathématique de l'érosion est similaire à celle de la dilatation.
Le principe de l'érosion est le même que la dilatation, mais de façon inversée. En effet, pour que le pixel prenne la couleur de l'objet, ses pixels connexes doivent tous êtres compris (entièrement inclus) dans l'élément structurant (β) [12].

$$ErosB(X) = \{p \setminus Bp\ X\} \quad (1.8)$$

1.9.3 Ouverture

L'ouverture morphologique de A par B, est simplement l'érosion de A par N, suivie d'une dilatation du résultat par B.

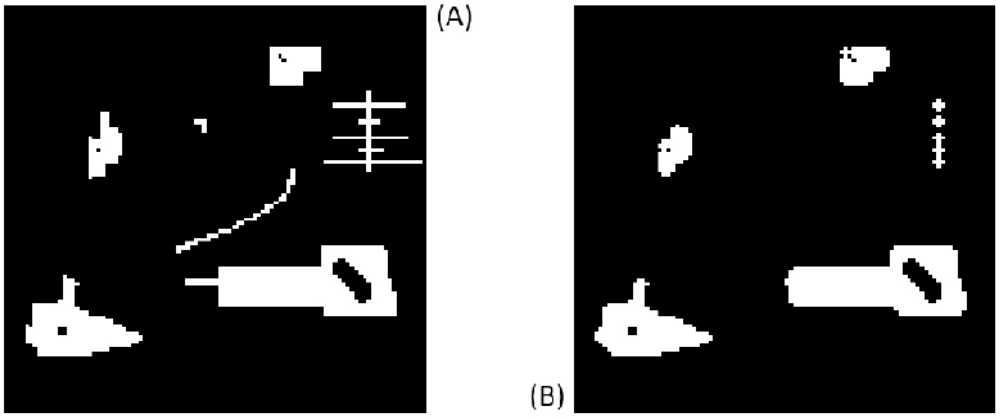


Figure 1.20: (A) Image originale ; (B) Résultat d'une ouverture.

On remarque que les petites structures ainsi que les détails fins disparaissent de l'image.
Cette technique permet ainsi d'éliminer des petits morceaux de contours bruités dans une image de contours [12].

1.9.4 Fermeture

La fermeture morphologique de A par B, correspond à une dilatation suivie d'une érosion, cette technique permet de remplir les trous [12].

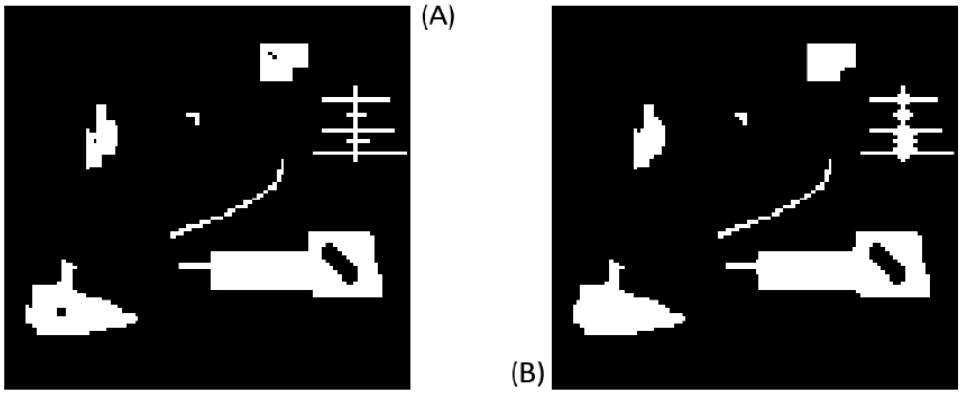


Figure 1.21: (A) Image originale ; (B) Résultat d'une fermeture.

1.10 Segmentation

La segmentation permet de distinguer les objets d'intérêt d'une image en isolant du fond ou des autres structures. Après la segmentation d'une image, chaque pixel se voit attribuer une région, et chaque région correspond à une partie sémantique de l'image.

Les discontinuités entre les régions correspondent aux contours des objets, aux incertitudes près. Les approches de segmentation peuvent se diviser en deux grandes classes :

- Approches de détection de discontinuité (contours ou encore frontières).
- Approches de détection de similarité (régions).

Remarquons qu'il existe aussi d'autres approches moins utilisées telles que les contours actifs, les templates, les modèles pyramidaux et les modèles mixtes. [6]

-Méthodes de détection de discontinuité

Les approches contours cherchent à extraire les contours présents dans l'image, en se basant sur l'étude des changements abrupts de la fonction de luminance ou sur la discontinuité des propriétés des ensembles.

-Méthodes de détection de similarité

Les approches régions cherchent quand à elles, à détecter les zones de l'image présentant des caractéristiques d'homogénéité et vérifiant un critère d'homogénéité.

Parmi les approches régions, on distingue les méthodes de classification, la croissance de régions et la division-fusion.

Les deux approches sont complémentaires et aucune n'a prouvé sa supériorité par rapport à l'autre, chacune ayant des avantages et ses domaines d'application.

1.11 Applications concrètes du traitement d'images

L'image est impliquée dans de nombreux domaines. On peut les différencier selon ce qu'on a au départ, et ce qu'on veut. En graphique par ordinateur (Computer Graphics), on construit un modèle et on le projette pour obtenir une image. En vision par ordinateur (Computer Vision), on analyse l'image pour avoir des informations.

Concernant les grandes applications actuelles, on peut en citer quatre :

- Reconstruction d'un objet d'après plusieurs images.
- Reconnaissance des formes, comparaison d'objets, OCR (reconnaissance de caractères). Lorsqu'on fait une comparaison, comme les modèles peuvent être à des tailles différentes sur des images de couleurs différentes, on a besoin d'une représentation plus indépendant, comme des statistiques.
- Calcul de mouvements entre des séries d'images.
- Trouver une image dans une base de données comme le net.

	Exemples	Domaine d'application
Traitement Des Images	Reconnaissance des caractères, du manuscrit	Saisie de texte, bureautique, tri postal, compression télécopie, chèque
	Reconnaissance des signatures	Banques, commerces
	Reconnaissance des empreintes digitales, des visages	Banques, commerces, police
	Analyse de radiographies, échographies, reconnaissance chromosomes, comptage globules	Contrôles systématiques de santé
	Détection de défauts circuits, pièces métalliques, manufacturées	Contrôle de qualité industrielle
	Identification d'objets	Tri d'objets industriels, surveillance Militaire
	Localisation d'objets	Guidage de robots industriels, guidage de missiles
	Analyse d'images de satellite	Météorologie, agriculture, ressources terrestres, surveillance militaire
	Analyse de photos aériennes	Agriculture surveillance militaire
	Analyse d'échos radar	Poursuite de cibles, pilotage missiles

Tableaux 1.1. Tableau des applications concrètes du traitement d'images.

1.12 Module arduino

1.12.1 Introduction

Les cartes Arduino sont conçues pour réaliser des prototypes et des maquettes de cartes électroniques pour l'informatique embarquée.

Ces cartes permettent un accès simple et peu couteux à l'informatique embarquée. De plus, elles sont entièrement libres de droit, autant sur l'aspect du code source (Open Source) que sur l'aspect matériel (Open Hardware). Ainsi, il est possible de refaire sa propre carte Arduino dans le but de l'améliorer ou d'enlever des fonctionnalités inutiles au projet.

Le langage Arduino se distingue des langages utilisés dans l'industrie de l'informatique embarquée de par sa simplicité. En effet, beaucoup de librairies et de fonctionnalités de base occulte certains aspects de la programmation de logiciel embarquée afin de gagner en simplicité. Cela en fait un langage parfait pour réaliser des prototypes ou des petites applications dans le cadre de hobby. Les possibilités des cartes Arduino sont énormes, un grand nombre d'application ont déjà été réalisée et testées par bon nombre d'internautes. On retrouve par exemple diverse forme de robot (voir figure 22), des stations météo (voir figure 23).

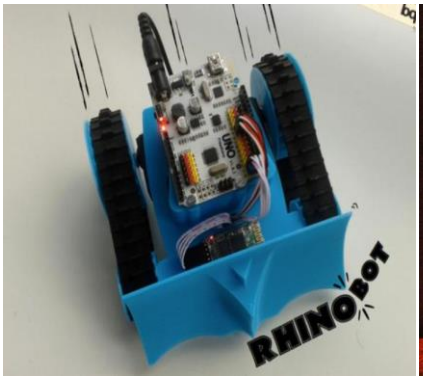


Figure1.22. Robot.

Figure 1.23. Système météo.

1.12.2 Présentation de la carte

a Qu'est ce qu'un microcontrôleur

Les cartes Arduino font partie de la famille des **microcontrôleurs**. Un microcontrôleur est une petite unité de calcul accompagné de mémoire, de ports d'entrée/sortie et de périphériques permettant d'interagir avec son environnement. Parmi les périphériques, ont recense généralement des Timers, des convertisseurs analogique-numérique, des liaisons Séries, etc. On peut comparer un micro contrôleurs à un ordinateur classique, mais système d'exploitation et avec une puissance de calcul considérablement plus faible. Les microcontrôleurs sont inévitables dans les domaines de l'informatique embarquée, de l'automatique et de l'informatique industrielle. Ils permettent de réduire le nombre de composant et de simplifier la création de cartes électroniques logiques.

b Caractéristiques techniques de la carte Arduino Due

La carte Arduino Due est la première carte Arduino basée sur une architecture ARM 32bit. Usuellement, les cartes Arduino sont plutôt basées sur une architecture ATMega 328.

La carte Arduino Due accueille un microcontrôleur Atmel SAM3X8E ARM Cortex-M3. Elle possède 54 entrées/sorties numériques (dont 12 peuvent être utilisées pour une sortie PWM), 12 entrées analogiques, 4 ports série UART, une horloge à 84 MHz, une connexion compatible USB OTG, 2 DAC (digital to analog), 2 TWI, une prise jack, un header SPI, un header JTAG et enfin un bouton de reset et un bouton d'effacement (voir figure 1.24).

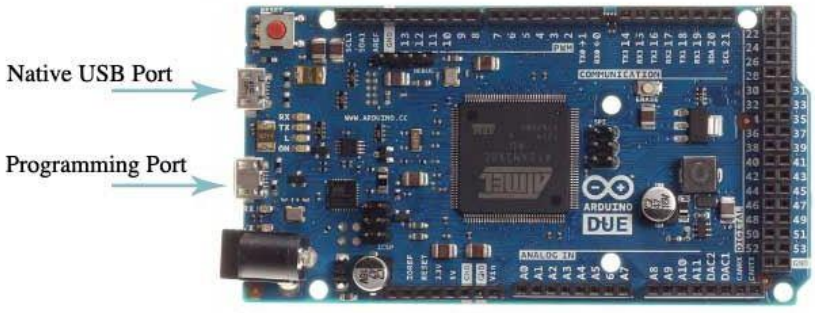


Figure 1.24. Arduino Due.

1.12.3 Présentation du logiciel

a IDE Arduino

Un **IDE** (environnement de développement) libre et gratuit est distribué sur le site d'Arduino (compatible Windows, Linux et Mac.

D'autres alternatives existent pour développer pour Arduino (extensions pour CodeBlocks, Visual Studio, Eclipse, XCode, etc.). L'interface de l'IDE Arduino est plutôt simple (voir figure1.25), il offre une interface minimale et épurée pour développer un programme sur les cartes Arduino. Il est doté d'un éditeur de code avec **coloration syntaxique** [1] et d'une **barre d'outils rapide** [2]. Ce sont les deux éléments les plus importants de l'interface, c'est ceux que l'on utilise le plus souvent. On retrouve aussi une barre de menus [3] plus classique qui est utilisé pour accéder aux fonctions avancées de l'IDE. Enfin, une **console** [4] affichant les résultats de la compilation du code source, des opérations sur la carte.

Figure 1.25. IDE(environnement de développement).

b Langage Arduino

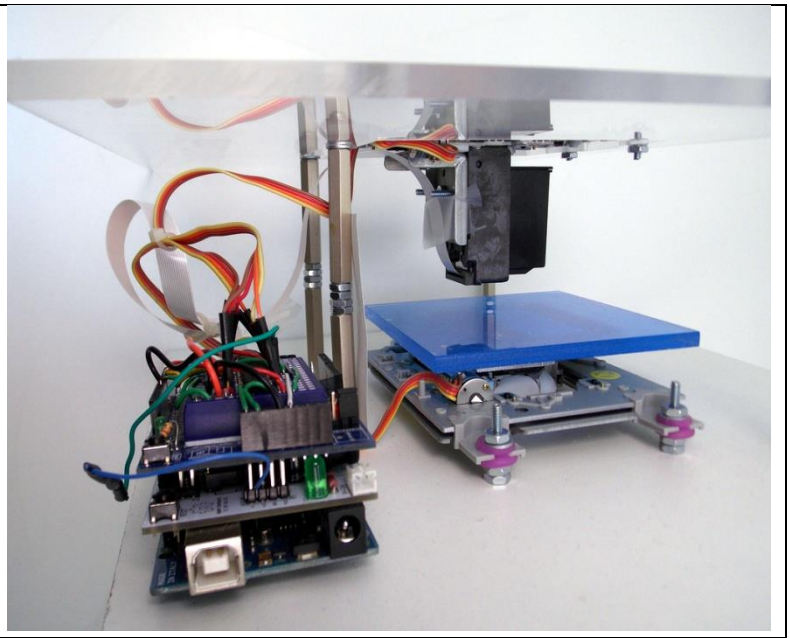
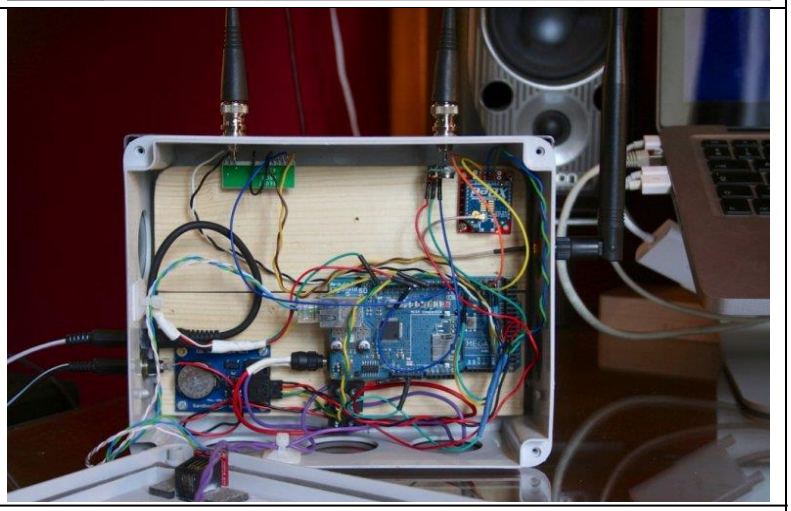
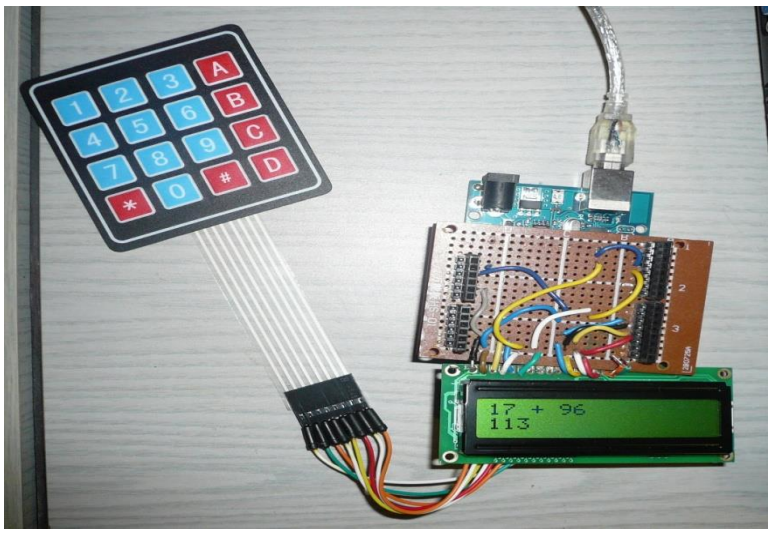
Le langage Arduino est inspiré de plusieurs langages. On retrouve notamment des similarités avec le C, le C++, le Java . Le langage impose une structure particulière typique de l'informatique embarquée. La fonction ***setup*** contiendra toutes les

opérations nécessaires à la configuration de la carte (directions des entrées sorties, débits de communications série, etc.). La fonction ***loop*** elle, est exécutée en boucle après l'exécution de la fonction *setup*. Elle continuera de boucler tant que la carte n'est pas mise hors tension, redémarrée (par le bouton ***reset***). Cette boucle est absolument nécessaire sur les microcontrôleurs étant donné qu'il n'on pas de système d'exploitation. En effet, si l'on omettait cette boucle, à la fin du code produit, il sera impossible de reprendre la main sur la carte Arduino qui exécuterait alors du code aléatoire.

1.12.4 Applications sur Arduino

L'arduino est utilisé dans de nombreux domaine tel que : Robotique, domotique, modélisme, télécommunications, arts & spectacles, plus globalement informatique embarquée...

Module Arduino	Un drone	

Module Arduino	Une imprimante 3D	
	Une box domotique	
	Calculatrice	

Tableaux 1.2. Tableau des applications sur la plaquette Arduino.

1.13 Conclusion

Nous avons présentés dans ce chapitre quelques notions sur la vision par ordinateur et sur le module Arduino ainsi que quelques techniques de traitement d'images. Dans le chapitre suivant nous allons détailler la chaine de la détection de la couleur que nous avons utilisé dans notre projet avec le système d'alerte.

Chapitre 2 Détection de couleur et Arduino

2.1 Introduction

La couleur est quelquefois utile dans certaines applications. Notre projet est de détecter une marque de couleur spécifique marquée par un opérateur sur un produit (faïences). Il s'agit d'une application industrielle dans le but de jouer le rôle d'un capteur électrique pour détecter une couleur, cette tache est composée principalement de plusieurs modules, notamment, le prétraitement, la détection de la couleur et la décision, comme l'indique la figure 2.1.

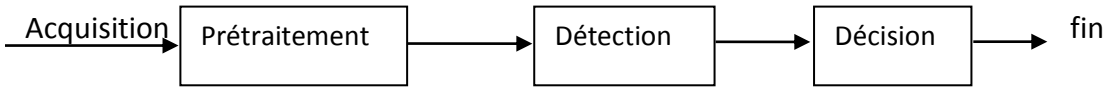


Figure 2.1. Schéma synoptique de la méthode proposée.

La détection de couleur en temps réel est une tâche très compliquée grâce à la présence des différents types de source de bruits tel que : l'éclairage ; les capteurs ; les objectifs …..

Avec la réalisation de tous les modules cites ci-dessus proposée dans cette tâche nous arrivons a la concrétisation d'une application industrielle.

Ce chapitre présente une méthode de détection de couleur tout en utilisant le module du traitement d'image, la figure 2.2 présente les étapes de traitement appliqué sur l'image acquise par la caméra numérique.

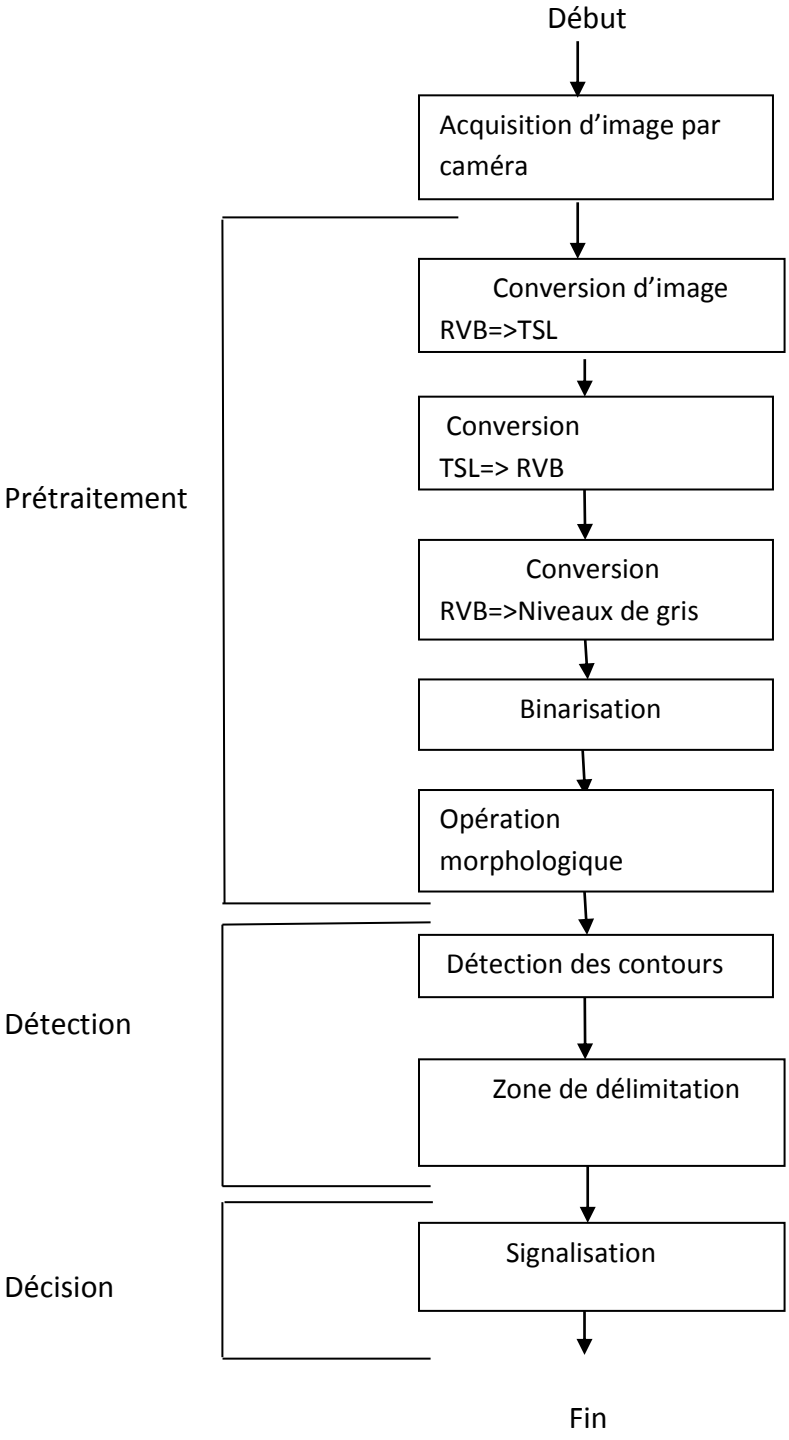


Figure 2.2.Chaine de vision.

2.2 Acquisition d'images

La camera utilisée à l'acquisition est de marque Logitech C910 de type CMOS. La résolution à été fixée à 960x720 pixels, et la vitesse à 30 images par second. L'image acquise est couleur et codée sous 24 bits (voir figure 2.3).

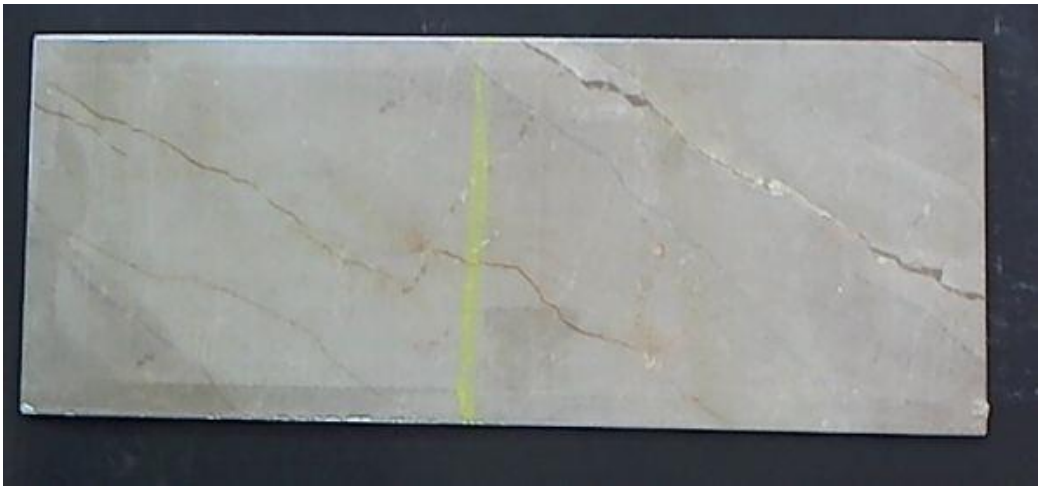


***Figure 2.3*.**Image acquise par la camera Logitech c910.

2.3 Prétraitement

La présence de différents types de bruits affecte directement l'image acquise par la camera, cette dégradation est due aux bruits tels que : bruit d'éclairage, bruits des capteurs est loin d'être parfaite.

Le rôle du prétraitement repose sur l'amélioration de la qualité visuelle de l'image tout en diminuant les types des bruits existant pour avoir une image résultante de meilleure qualité et prêt à la détection de la couleur.

2.3.1 Conversion d'image couleur(RVB) en espace TSL

Le modèle TSL (acronyme de Teinte, Saturation, Luminosité) (voir figure 2.4) est le plus intuitif de tous les modèles colorimétriques. Il est basé sur le ressenti de la perception humaine d'où son nom de modèle perceptuel. Chaque critère de couleur est clairement séparé, ce qui en fait le modèle le plus pratique pour la retouche d'image ou l'ajustement des couleurs.

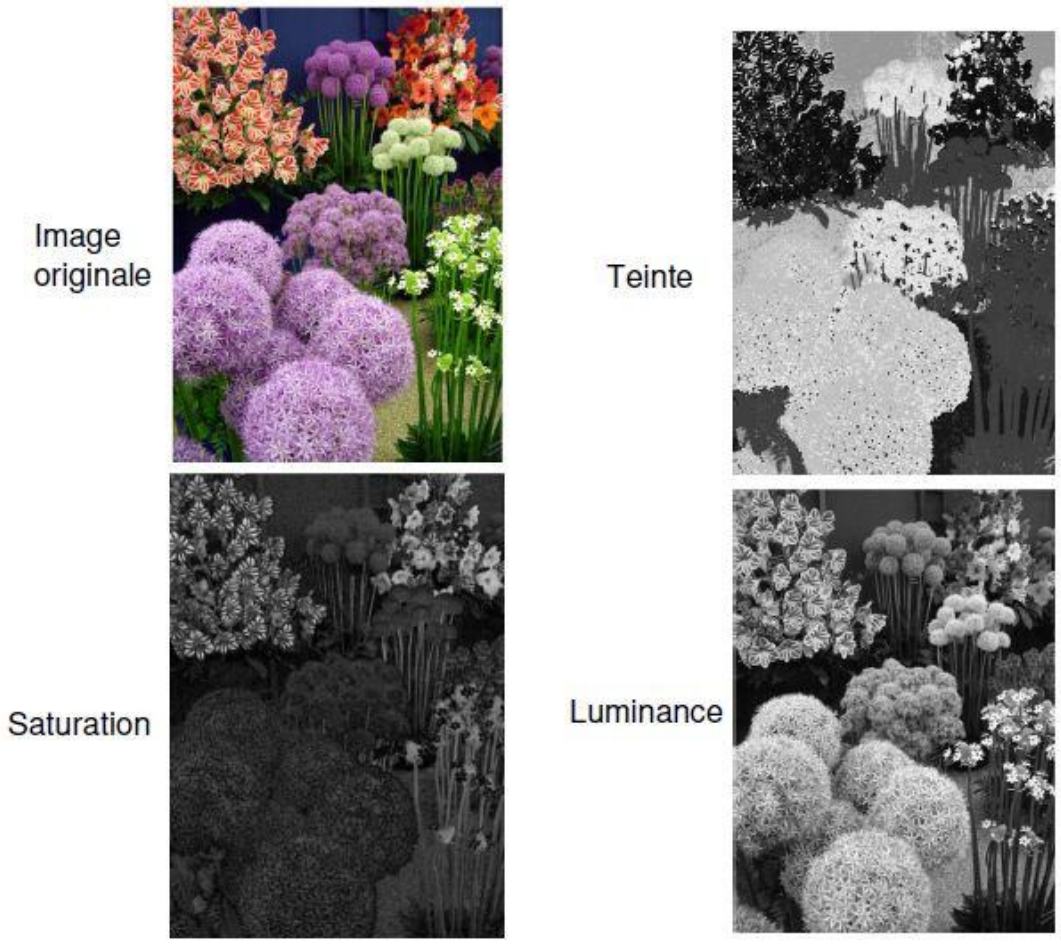


Figure 2.4. Les plans d'espace TSL.

Le modèle de couleurs TSL est utilisé pour la manipulation de la teinte et de la saturation (voir figure 2.5), car il permet de modifier directement ces valeurs (contrairement au modèle RVB).

Figure 2.5. Modification de la teinte de la voiture grâce à l'espace TSL.

Le passage entre l'espace RVB vers l'espace TSL se fait à partir d'une opération mathématique, Comme cela est représenté ci-dessous :

Pour faire les calculs il faut définir de nouvelles variables: M et m sont les max et min des composantes RVB

M=max(R,V,B)

m=min(R,V,B)

C=M-m

- Si M=R => $T = 60 \times \left(\frac{\mathrm{V}-\mathrm{B}}{\mathrm{C}}\right)$ (2.1)
- Si M=V => $T = 60 \times \left(\left(\frac{\mathrm{B}-\mathrm{R}}{\mathrm{C}}\right) + 2\right)$ (2.2)
- Si M=B => $T = 60 \times \left(\left(\frac{\mathrm{R}-\mathrm{v}}{\mathrm{C}}\right) + 4\right)$ (2.3)

$$L = \frac{M+m}{2} \quad (2.4)$$

$$\text{Si } L \geq 0.5 \Rightarrow S = \frac{M-m}{2-2L} \quad ; \quad \text{Si } L \leq 0.5 \Rightarrow S = \frac{M-m}{2L} \quad (2.5)$$

Avec

T : la teinte.

L : la luminance.

S : saturation.

2.3.2 Conversion d'image en espace TSL vers une image (RVB)

Pour continue la suite de traitement nous somme obligé de faire retour vers l'espace RVB.

Ce passage fait a l'aide d'opération mathématique suivant :

Pour faire les calculs il faut définir de nouvelles variables: C, m et X sont des variables uniquement utilisées pour le calcul.

- C = (1-|2L-1) (2.6)
- X =(1-|(H/60) mod2-1) (2.7)
- m = L-$\frac{C}{2}$ (2.8)

Si : $0 \leq H \leq 60$: (R', V', B')= (C, X, 0)

Si : $60 \leq H \leq 120$: (R', V', B')= (X, C, 0)

Si $120 \leq H \leq 180$: (R', V', B')= (0, C, X)

Si $180 \leq H \leq 240$: (R', V', B')= (0, X, C)

Si $240 \leq H \leq 300$: (R', V', B')= (X, 0, C)

Si $300 \leq H \leq 360$: (R', V', B')= (C, 0, X)

(R, V, B) = (R'+m, V'+m, B'+m)

Avec (H : la teinte, L : luminance, S : saturation)

On peut voir la saturation comme un écart entre la couleur considérée et une couleur neutre (un niveau de gris), les gris ont donc une saturation nulle, pour être très saturée une couleur doit avoir une composante RVB très grande et une autre très petite. La luminosité représente la "distance" entre la couleur considérée et le noir, pour être très lumineuse une couleur doit avoir une composante très élevée.

2.3.3 Conversion d’image couleur vers image à niveaux de gris

On passe de l'image couleur à trois plans à une image à niveau de gris (voir figure 2.6) en une image ayant un seul plan pour diminuer les calculs car l'information voulue (les contours) se présente aussi dans l'image à niveau de gris, et de ce fait, il n'est pas nécessaire de traiter les trois plans couleurs pour l'extraire. La conversion ce fait par calcul d’une moyenne pondérée car elle est mieux adapté à la perception humaine qui est plus sensibles au vert que les autres couleurs en utilisant la formule: Image gris = 0.30 R + 0.59 G + 0.11 B.

Avec

R : la valeur de rouge.

G : la valeur de vert.

B : la valeur de bleu.

Figure 2.6. Image à niveaux de gris.

2.3.5 Binarisation de l'image à niveaux de gris

Cette méthode permet d'obtenir une image noir et blanc où les pixels des objets désirés sont en blanc et le reste en noir. Dans cette technique, le seuil est une valeur qui nous permet de départager les pixels selon la relation suivante :

Pixel < Valeur du Seuil alors Pixel = 0

Pixel >= Valeur du Seuil alors Pixel = 255

Finalement l'image binarisé (voir figure 2.7).

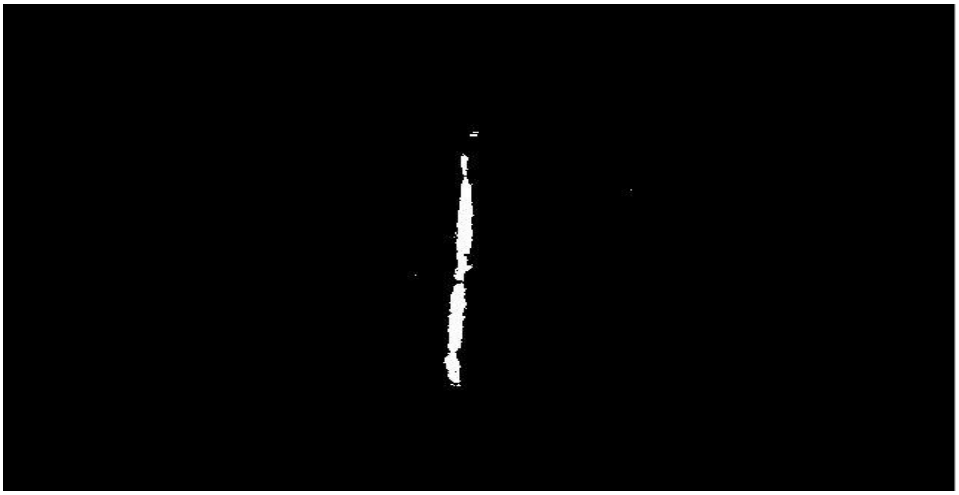


Figure 2.7. Image binarisée.

2.3.6 Opération morphologique

La composition d'une dilatation morphologique avec l'érosion par le même élément structurant ne produit pas, en général, l'identité, mais deux autres opérateurs morphologiques.

a Ouverture mathématique

L'ouverture de l'ensemble X par l'élément structurant B est définie par :

$$OuvB(X) = \{Bp \setminus Bp\ X\} \qquad (2.11)$$

L'ouverture a pour effet de supprimer les parties des objets plus petites que l'élément structurant (voir figure 2.8), et de lisser les contours en supprimant les petites excroissances (trop fines pour pouvoir contenir l'élément structurant. Elle ne réduit pas systématiquement toutes les structures comme le fait l'érosion. La figure 2.12 illustre ces effets.

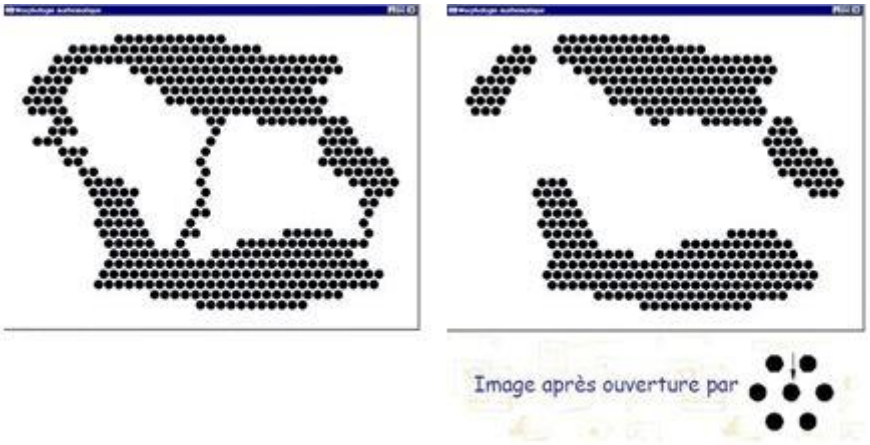


Figure 2.8. Application de l'ouverture.

Exemple d'ouverture binaire (de gauche à droite : image initiale, ouverture par un disque de taille 3*3, différence : en blanc, les parties supprimées par l'ouverture).

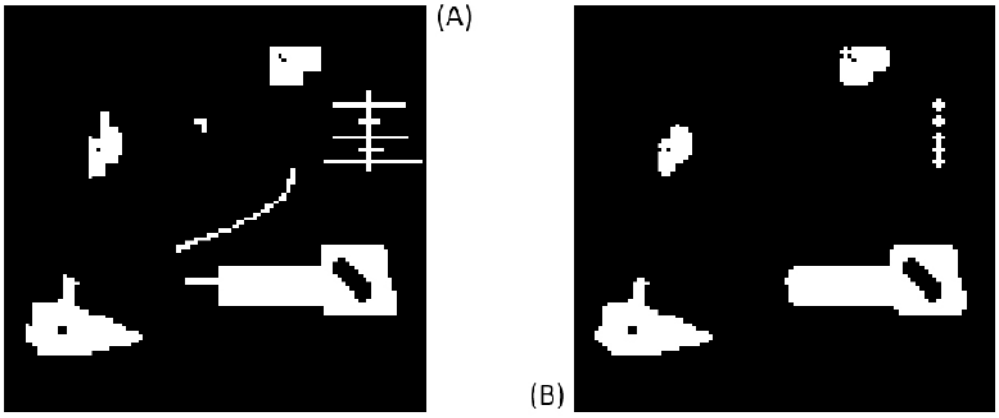


Figure 2.9.Suppression des parties inferieur au l'élément structurant.

Nous voyons clairement sur l'image binarisé (figure2.10) qu'il existe des parasites donc nous utilisons l'opération d'ouverture pour l'éliminer.

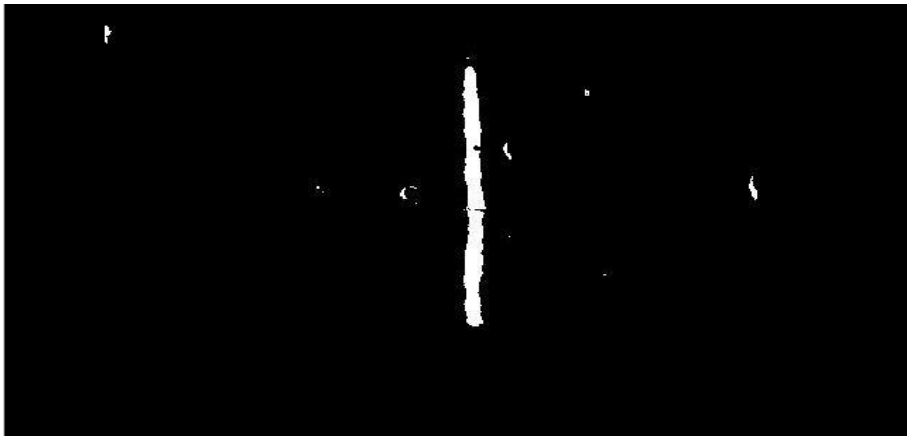


Figure 2.10 .Image binaire.

b Fermeture mathématique

La fermeture de l'ensemble X par l'élément structurant B définie par :

$$FermB(X) = [\ \{Bvp \setminus Bvp\ Xc\}\ ]c \qquad (2.12)$$

La fermeture a pour effet de boucher les trous des objets qui sont plus petits que l'élément structurant (voir figure 2.11).

Elle lisse les contours des objets en rajoutant des points dans les concavités étroites (dans lesquelles ne peut pas se glisser l'élément structurant). La figure 2.12 illustre ces effets.

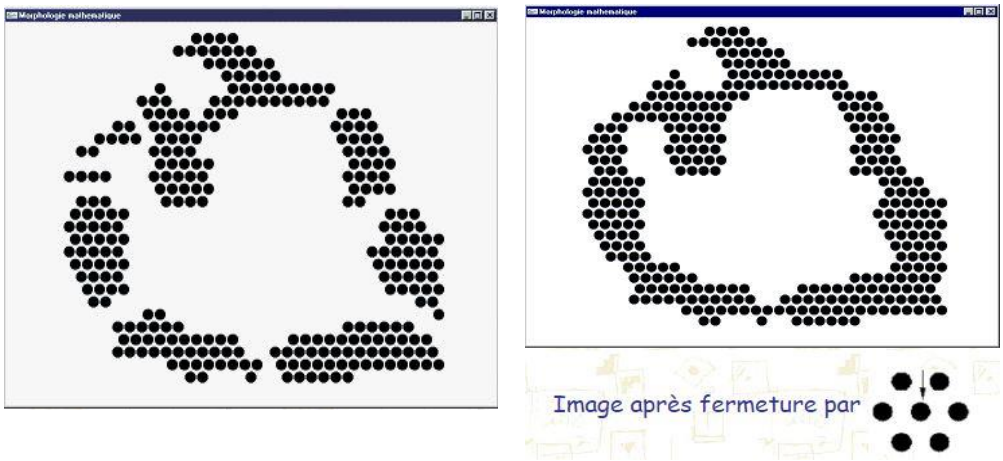


Figure 2.11.Fermeture de l'image par l'élément structurant spécifique.

Exemple de fermeture binaire (de gauche a` droite : image initiale, fermeture par un disque de taille 3*3, différence : en blanc, les parties rajoutées par la fermeture).

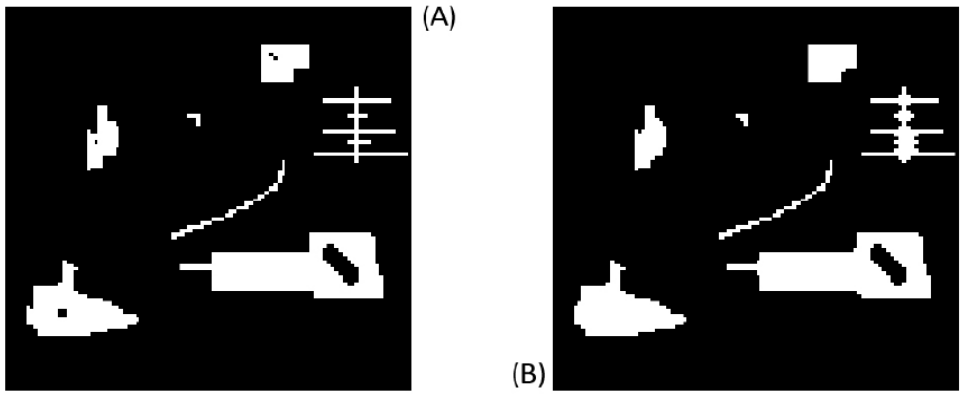


Figure 2.12. Remplissage des trous.

Nous voyons clairement sur l'image binarisé (figure2.13) qu'il existe des trous au niveau de trait qui nous voulons le détecter donc nous utilisons l'opération de fermeture pour les remplir.

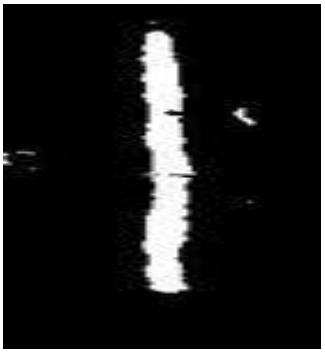


Figure 2.13. Les trous existent dans l'image.

2.4 Détection

2.4.1 Détection des contours

Comme nous l'avons vu dans le module précédent, les filtres morphologiques nous permettons d'isoler l'image binaire en comblant des trous ou d'éliminer des branches qui sont considère comme parasites. En ayant le contour d'objet, nous pouvons ressortir certains paramètres du contour grâce à quelques fonctions de la bibliothèque d'Open CV.

En effet, ce sont les pixels situés sur la bordure d'un objet qui nous permettent de déterminer la forme de cet objet. Ce module se penche donc sur les techniques disponibles dans Open CV pour extraire les caractéristiques des objets.

Nous utiliserons la fonction *findContours* pour récupérer les caractéristiques des objets présents dans l'image.

a Localisation des contours

La fonction *« findcountour »* permet de trouver les contours en remplissant un vecteur de point qui correspond aux points qui sont définies sur le contour extérieur de l'objet (voir figure 2.14).

Dans le module précédent, nous avons appliqué une ouverture suivie par une fermeture pour ressortir les pixels faisant partie du contour d'un objet. On peut ensuite, avec la fonction « findContours » trouver les points sur ce contour. Cette

séquence de point est essentielle dans la détection de couleur. Nous allons donc appliquer cette fonction aux objets de façon à déterminer leur forme.

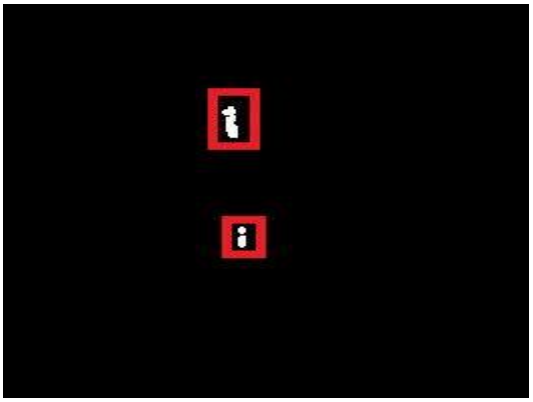


Figure 2.14. Image contient deux objets.

b zone de délimitation

Après la détection des contours d'une zone colorée, nous allons générer une zone de délimitation, ce dernier est la façon la plus compacte pour représenter et localiser des composants dans une image. Elle est représentée par un rectangle vertical de taille minimale (voir figure 2.15) et contient complètement la forme. Comparer la hauteur et la largeur du rectangle donne une indication sur l'orientation verticale ou horizontale de l'objet.

La fonction CV qui réalise cette opération est : *Bounding Rect.*

Figure 2.15.localisation des contours par des rectangles rouges.

2.5 Module arduino

2.5.1 Définition

C'est une plate-forme open-source d'électronique programmée qui est basée sur une simple carte à microcontrôleur (de la famille AVR), et un logiciel, véritable environnement de développement intégré, pour écrire, compiler et transférer le programme vers la carte à microcontrôleur.

2.5.2 port série

En informatique, un port série est une interface de communication série à travers laquelle les transferts d'information dans ou hors d'un bit à la fois (contrairement à un port parallèle).

Les ordinateurs modernes sans ports série peuvent nécessiter des convertisseurs série-USB pour permettre la compatibilité avec RS 232 périphériques série. Les ports série sont encore utilisés dans des applications telles que des systèmes d'automatisation industrielle, des instruments scientifiques, systèmes de point de vente et certains produits industriels et de consommation. Ordinateurs de serveur peuvent utiliser un port série comme une console de commande pour le diagnostic. Équipement de réseau (tels que les routeurs et les commutateurs) utilisent souvent la console série pour la configuration. Les ports série sont encore utilisés dans ces domaines car ils sont simples, pas cher et leurs fonctions de la console sont très normalisées et généralisée.

2.5.3 Signalisation

Nous avons utilisé le port série d'arduino pour démarrer le programme de signalisation sur l'environnement d'arduino lorsque on détecte la couleur sur la faïence marquée.

2.6 Conclusion

Dans ce chapitre nous avons présenté les différentes étapes de la chaine de détection proposé, qui consiste premièrement à faire une détection de la couleur marquée, puis deuxièmement d'extraire la couleur et la localisée, et finalement de faire appel au arduino pour un signal d'alerte.

Chapitre 3 Implémentation et résultat

Ce chapitre présente tout d'abord la partie logicielle et matérielle utilisée pour le développement de ce projet, ensuite la partie test et résultats expérimentaux.

3.1 Environnement de développement

3.1.1 Environnement matériel

Afin de mener à bien ce projet, il a été mis à notre disposition un ensemble de matériels dont les caractéristiques sont les suivantes :

Un ordinateur Dell avec les caractéristiques suivantes :

- Processeur : Intel(R) Core(TM) i5-4210U @ 1.70GHz
- RAM : 8,00 Go
- Carte graphique : AMD Radeon HD 8670M
- Système: Windows 7.

La Camera : Nous avons utilisé la camera Logitech C910.

- Carte Arduino Due.

3.1.2 Environnement logiciel

- **Logiciel de développement :** Qt creator C++, Arduino soft 1.6-6.
- **Bibliothèque graphique :** Open CV (Open Computer Vision)

3.2 Qt Creator C++

3.2.1 Définition

Qt Creator est un environnement de développement intégré (IDE) multiplate-forme faisant partie du framework Qt. Il est donc orienté pour la programmation en C++. Il intègre directement dans l'interface un débogueur, un outil de création d'interfaces graphiques, des outils pour la publication de code sur Git et Mercuriale ainsi que la documentation Qt. L'éditeur de texte intégré permet l'autocomplétion ainsi que la coloration syntaxique.

Qt Creator utilise sous Linux le compilateur gcc. Il peut utiliser MinGW ou le compilateur de Visual Studio sous Windows. Qt est initialement développé par la société Trolltech, qui fut par la suite rachetée par Nokia. Le développement de Qt a commencé en 1991 et il a été dès le début utilisé par KDE, un des principaux environnements de bureau sous Linux.

Qt et aussi une bibliothèque graphique qui comporte beaucoup de module Gui (Graphical user interface) bien qu'il soit possible de développer en C++ avec Qt en utilisant d'autre IDE (comme Code::Blocks) il est recommandé d'utiliser l'IDE Qt Creator. Il est particulièrement optimisé pour développer avec Qt. En effet, c'est un programme tout-en-un qui comprend entre autres :

- un IDE pour développer en C++, optimisé pour compiler des projets utilisant Qt (pas de configuration fastidieuse) ;
- un éditeur de fenêtres, qui permet de dessiner facilement le contenu des interfaces à la souris.
- une documentation indispensable pour tout savoir sur Qt.

Voici à quoi ressemble Qt Creator quand on le lance pour la première fois (voir figure 3.1).

Figure 3.1.Qt creator à l'ouverture.

C'est un outil très propre et très bien fait. Avant que Qt Creator n'apparaisse, il fallait réaliser des configurations parfois complexes pour compiler des projets utilisant C++ et des bibliothèques de Gui.

3.2.2 Qt porte série (SerialPort)

Port série (QtSerialPort en anglais) est un module Qt qui fournit un support pour les ports série tels que ceux qui suivent la norme RS-232. Modules de port série pour Qt ont existé pendant un certain temps, remontant au moins aussi loin que QExtSerialPort sous Qt la version 2, mais aucun d'entre eux ont été officiellement partie de Qt. QtSerialPort provenaient de la bibliothèque tiers QSerialDevice et a ensuite été fait partie du projet Qt. Il devient officiellement partie de Qt à partir de la version 5.1.0.

Le module fournit deux classes principales: QSerialPort et QSerialPortInfo. Pour utiliser le module avec Qt 5 ajouter cette ligne à votre fichier de projet qmake: QT + = serialport Vous pouvez ensuite inclure les fichiers d'en-tête <QSerialPort> ou <QSerialPortInfo>. Il est tout à fait une bonne documentation pour les classes.

3.3 Arduino soft 1.6.6

L'environnement de programmation Arduino (IDE en anglais) est une application écrite en Java inspirée du langage Processing.
L'IDE permet d'écrire, de modifier un programme et de le convertir en une série d'instructions compréhensibles pour la carte.

Le langage de programmation utilisé est le C++, compilé avec avr-g++ , et lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties. La mise en place de ce langage standard rend aisé le développement de programmes sur les plates-formes Arduino (voir figure 3.2), à toute personne maîtrisant le C ou le C++.

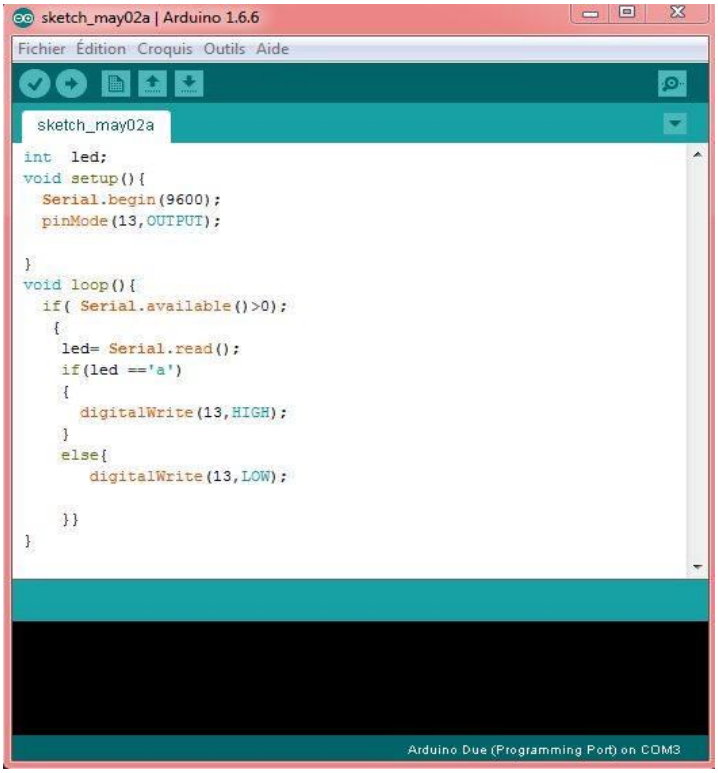


Figure 3.2.plateforme Arduino.

3.4 La librairie OpenCv

3.4.1 Présentation d'OpenCV

Un des objectifs buts d'OpenCV est d'aider les utilisateurs à construire rapidement des applications sophistiquées de vision à l'aide de simples opérations de vision par ordinateur.

OpenCV (Open Source Computer Vision) est une bibliothèque proposant un ensemble de plus de 2500 algorithmes de vision par ordinateur, elle a était écrite en C et C++, puis des interfaces ont été développées pour Python, Ruby, Matlab et autres langages. Elle est distribuée sous une licence BSD (libre) pour les plates-formes Windows, GNU/Linux, Android et MacOS. Initialement écrite en C il y a 10 ans par des chercheurs de la société Intel, OpenCV est aujourd'hui développée, maintenue, documentée et utilisée par une communauté de plus de 40 000 membres actifs. C'est la bibliothèque de référence pour la vision par ordinateur, aussi bien dans le monde de la recherche que celui de l'industrie.

En résumé elle est:

- Une librairie open source de traitement et analyse d'images et vidéos avec des interfaces pour les principaux langages de programmation C, C++, Java, Python ...
- Optimisée pour les applications temps réelles.
- Fournit une API bas et haut niveau.
- Utilisé aussi bien dans les laboratoires de recherche que dans l'industrie.

3.4.2 Fonctions

- Manipulation d'images (chargement, sauvegarde, copie, conversion...).
- Manipulation et acquisition de vidéos.
- Manipulations de matrices et algèbre linéaire.
- Structure de données utilitaires variées (listes, files, ensembles, graphes...).
- Traitement d'images (filtrage, détections de discontinuités, morphologie mathématique...)
- Analyse d'images (composantes connexes, ajustement de primitives, transformée de distance...)

- Vision (calibration de caméra, stéréovision, recherche d'association...)
- Reconnaissance de forme.
- Interface graphique (affichage d'images, de vidéos, gestion des évènements...).

3.5 Interfaces de l'application

3.5.1 Interface principale

Cette interface graphique représente le bouton principale de démarrage (voir figure 3.3) de cette application qui nous avons développé.

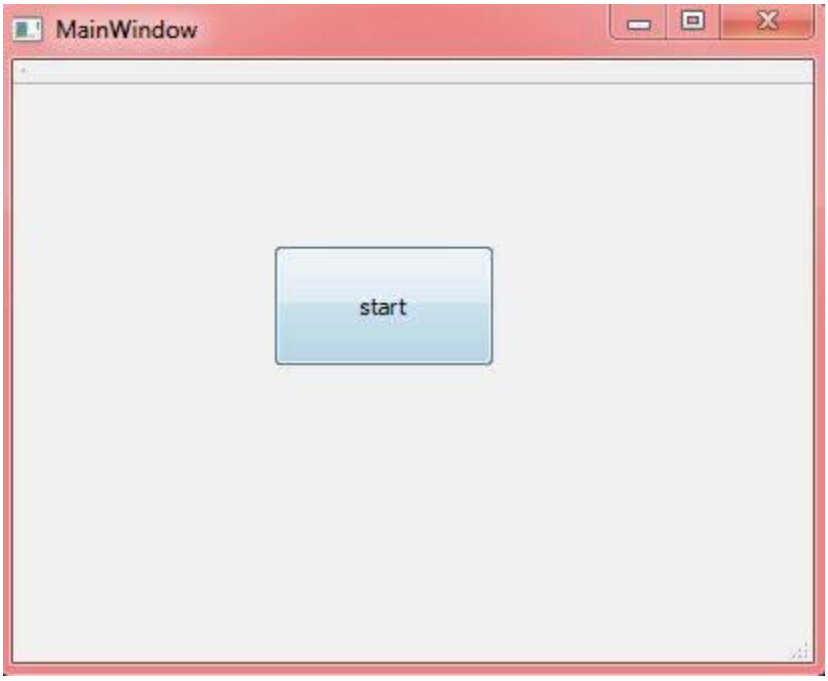


Figure 3.3.bouton de démarrage (start).

3.6 Organigramme de la méthode proposé

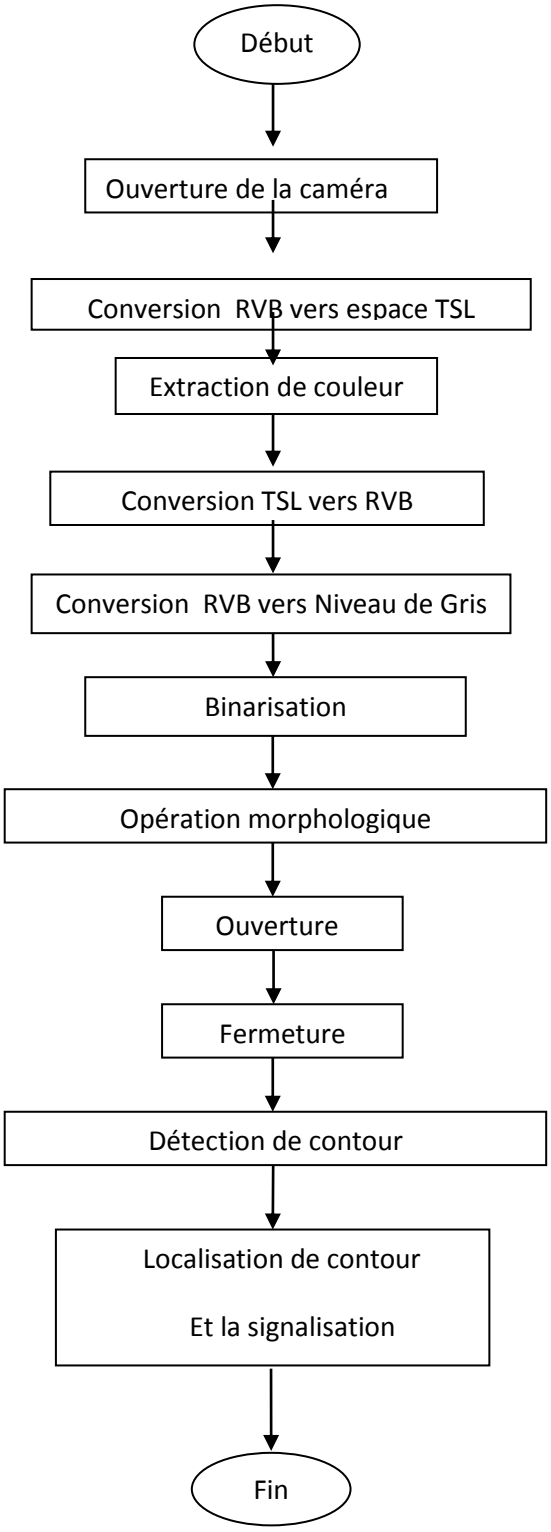


Figure 3.4. Organigramme de la méthode proposé.

3.7 Détection de marquage sur faïences

Le travail que nous avons développé dans la cadre de ce projet, consiste à détecter et identifier le marquage sur les faïences de mauvaise qualité, marqué par un opérateur.

3.7.1 Ouverture de la camera

Dans cette étape nous allons fait appel à la camera (voir figure3.5) grâce à la bibliothèque opencv 'imgproc', pour faire le traitement sur les images de faïences qui ont été acquises en temps réel.

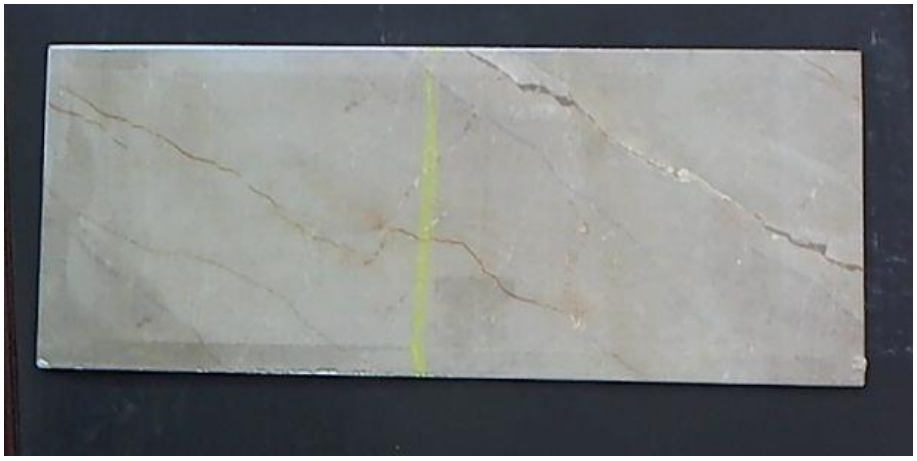


Figure 3.5.Image acquise par camera.

3.7.2 Conversion format RVB vers format TSL

Dans cette phase l'image de faïence couleur est convertie en une image sous format TSL, dans le but d'extraire la couleur du feutre utilisé pour marquer la faïence de mauvaise qualité. Le résultat de la conversion est représenté par la figure 3.6.

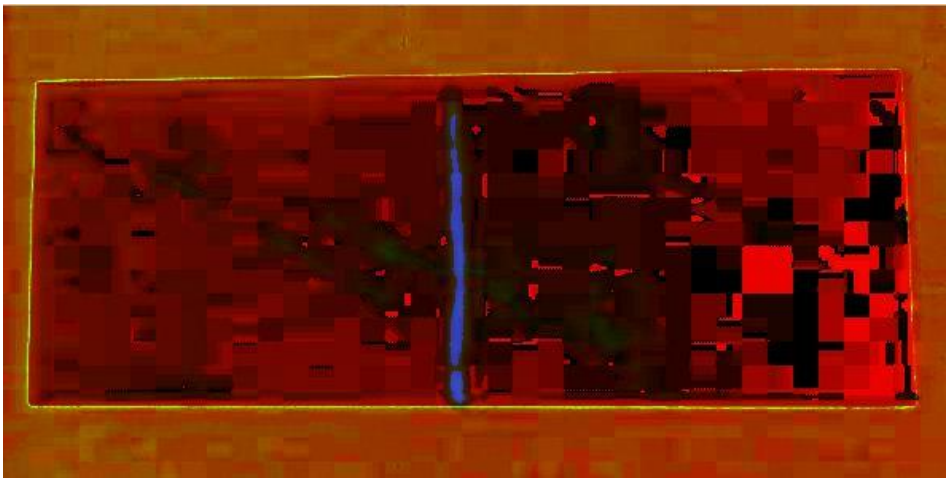


Figure 3.6. Affichage de l'image dans l'espace TSL.

3.7.3 Extraction de couleur de marquage

Dans cette partie nous allons jouer sur les trois composants (T S L), dans le but de faire isolé la couleur de feutre utilisé pour le marquage et éliminer toutes les autres couleurs qui existent sur l'image, après nous allons faire la conversion inverse (espace TSL vers espace RVB) pour continuer la suite de traitement. Le résultat est représenté sur la figure 3.7.

Figure 3.7.Image en espace RVB.

3.7.4 Conversion RVB en niveau de gris

Dans cette étape l'image de faïence couleur est convertie en une image à niveau de gris, dans le but de réduire les calculs et ne garder que la partie monochrome de l'image couleur de la marque. Le résultat de la conversion est représenté par la figure 3.8.

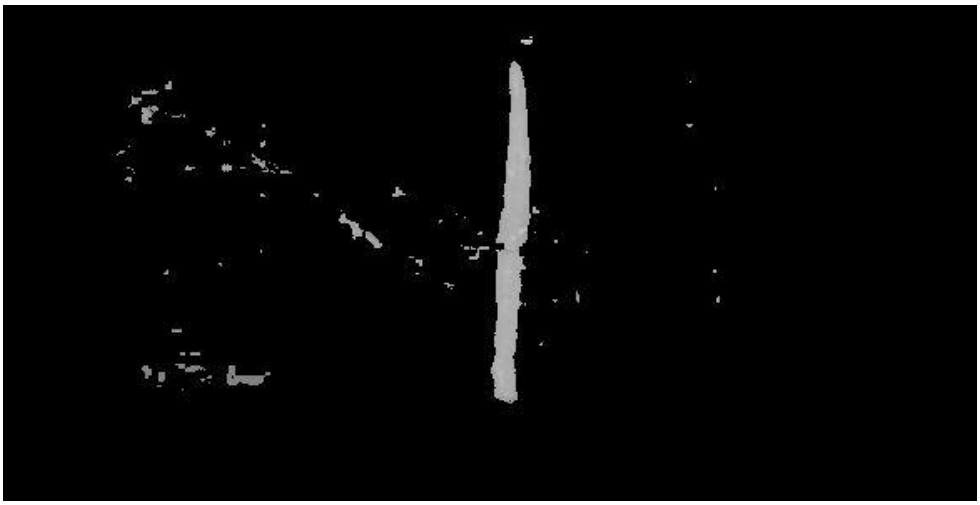


Figure 3.8. Image à niveau de gris.

3.7.5 Binarisation

Dans cette étape nous avons fait une binarisation (voir figure3.9) sur l'image obtenue, dans le but est d'appliqué des opérations morphologique sur l'image résultant.

Figure 3.9.Image binaire.

3.7.6 Opération morphologique

a Ouverture

Nous avons appliqué une ouverture avec un élément structurant de taille 5*5 (voir figure3.10) sur l'image binarisée. Dans le but de supprimer les taches (iles) qui apparaissent dans l'image.

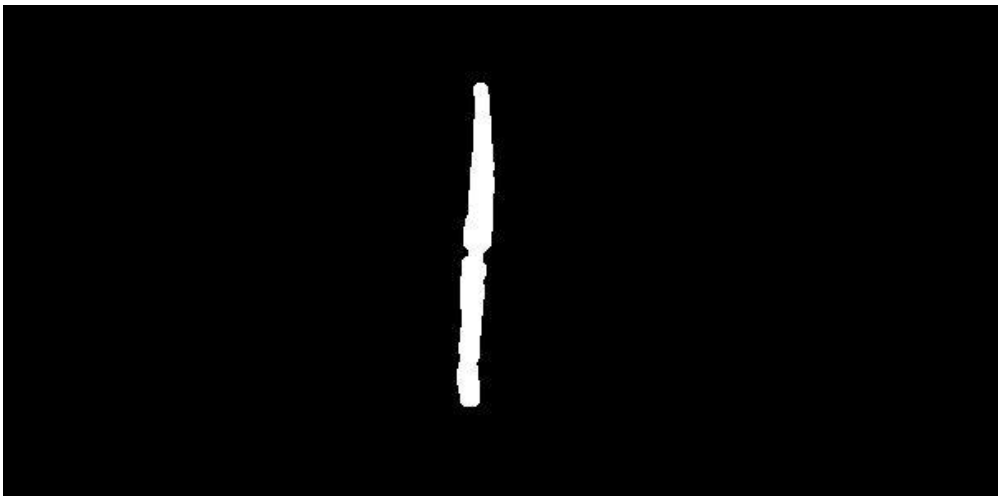


Figure 3.10.Résultat de l'ouverture.

b Fermeture

L'application de fermeture avec un élément structurant de taille 5*5, a pour but de faire remplir les trous (lacs) qui se trouvent sur la zone d'intérêt. Le résultat de la fermeture est représenté sur la figure 3.11

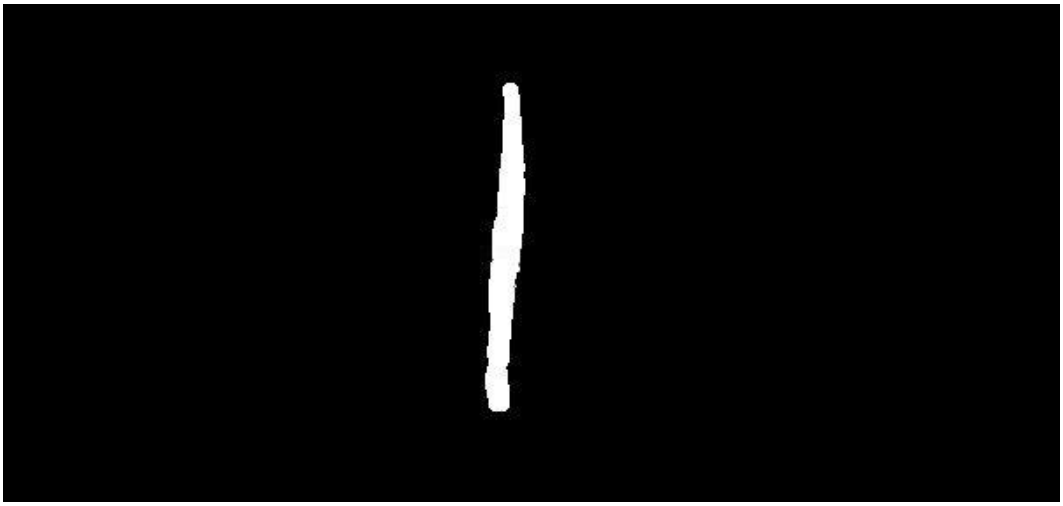


Figure 3.11.Résultat de la fermeture.

3.7.7 Détection des contours

L'utilisation de la fonction '*FindContours'* nous permettons de détecté les zones de transitions sur l'image obtenue.

a Zone de délimitation

Après la détection des contours et grâce a l'utilisation de la fonction *'BoundingRect',* nous arrivons à délimiter les contours que nous avons obtenus par un rectangle de couleur rouge sur l'image originale, le résultat est présenter par la figure3.12.

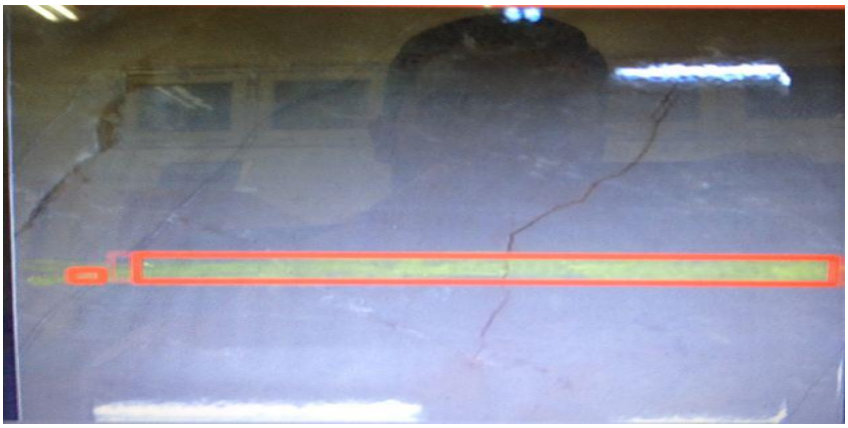


Figure 3.12.localisation de la zone d'intérêt.

b Signalisation (LED)

Lorsque nous détectons les contours (la couleur marquée), l'environnement QT fait appel au module arduino selon la porte serie (SerialPort), pour allumer la LED et le message sonore (buzzer) (voir figure3.13).

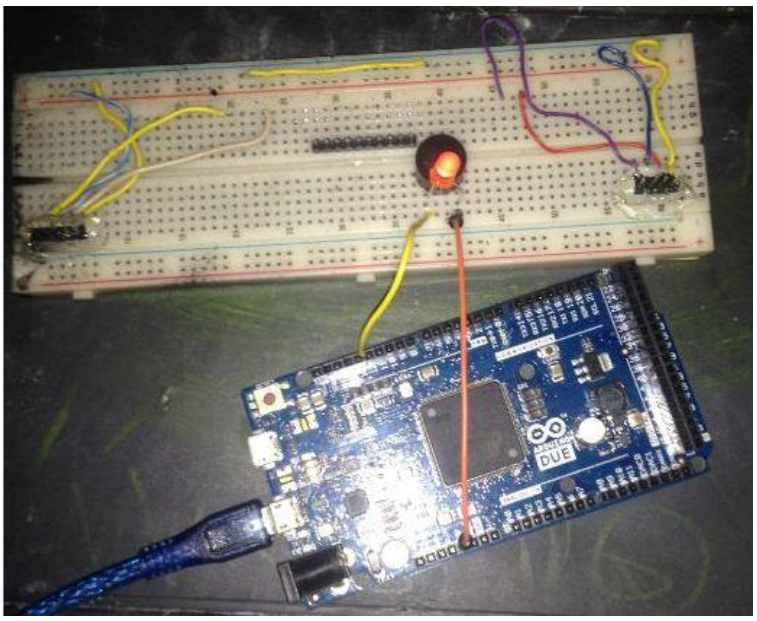


Figure 3.13.Allumage de la LED.

3.8 Résultats expérimentaux

L'application a été testée en temps réel sur différents modèles de faïences (voir figure 3.14). Il était important de maintenir les mêmes conditions d'éclairage et la distance de la caméra par rapport à la faïence afin de mieux conclure les résultats obtenus, nous sommes arrivés à un taux de réussite de 100%, avec un temps moyenne de traitement de 6ms.

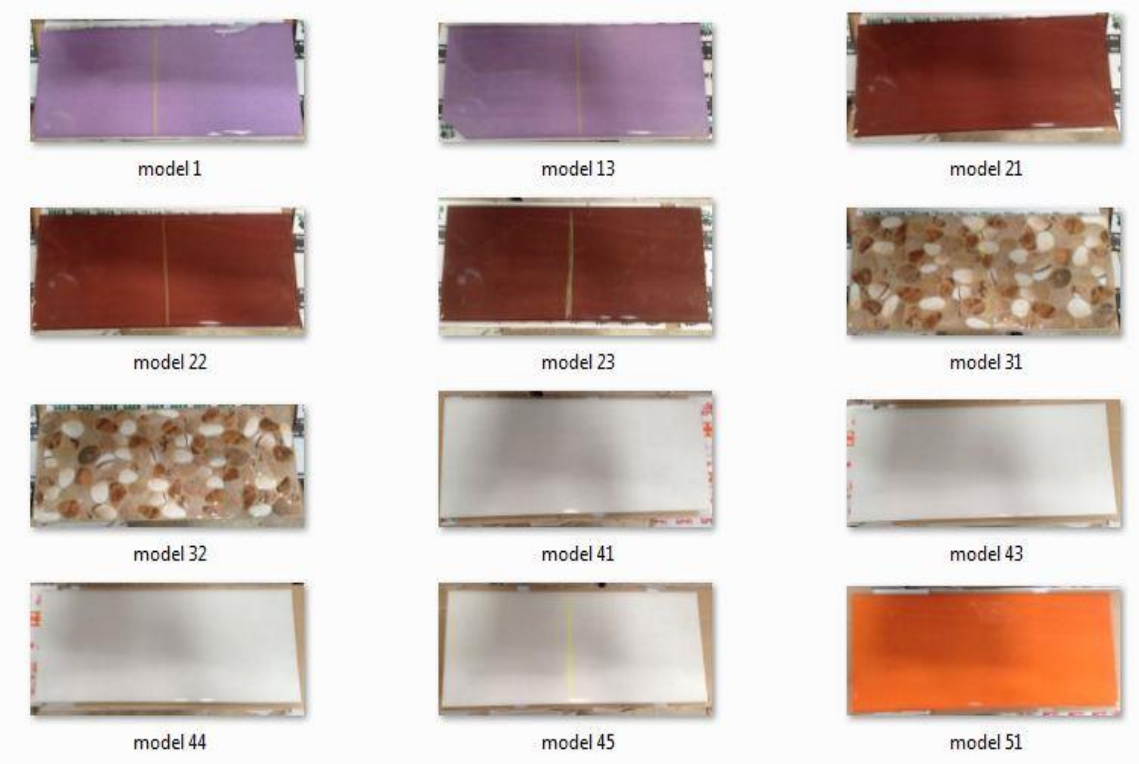


Figure3.14.Différent modèle de la faïence.

3.9 Conclusion

Le passage de la théorie à l'application afin d'obtenir des résultats satisfaisants n'est pas toujours évident. Ce passage est très délicat et plus important. Après plusieurs tests nous avons réussi à réaliser le rôle du capteur utilisé en usine avec la signalisation lors du passage d'un produit marqué (mauvaise qualité) devant la caméra. Les résultats expérimentaux montrent bien que notre algorithme donne d'excellents résultats. Cela montre bien que cette chaine de détection et prometteuse est peut faire l'objet d'une application industrielle.

Conclusion générale

Nous avons présenté dans ce mémoire une méthode de détection des faïences marquées basée sur la vision par ordinateur. Nous avons présenté en premier lieu des généralités sur le traitement d'images et le module Arduino (due) et détailler quelques outils du traitement que nous avons utilisé dans nos programmes. En deuxième lieu, nous avons présenté la méthode de la détection de la couleur marquée sur la faïence avec toutes les étapes proposées en détaillant les algorithmes utilisés. En troisième lieu nous avons présenté l'application créée avec l'implémentation et les résultats de tests.

Pour réaliser un tel travail la première étape était de faire passer des faïences en-dessous de la caméra afin de faire les tests. L'environnement étant très complexe, nous avons fait face au problème de luminosité et éclairage, notamment les ombres et la réflexion de la lumière sur la faïence, aussi la distance par apport a la caméra, la focalisation.

Nous avons réussi à créer un algorithme qui détecte la couleur marquée sur la faïence et la localiser, afin d'alerter l'opérateur à l'aide du module ardunio qui envoie un signal à la led pour s'allumer.

Au cours de ce projet et à travers le travail fait, nous avons pu acquérir beaucoup de connaissances. Premièrement, nous avons utilisé les outils appris au cours de notre formation, et enrichi nos connaissances dans le domaine du traitement d'image en apprenant de nouveaux outils (OpenCV, Qt creator) et le module arduino. Nous avons beaucoup appris à utiliser le langage de programmation IDE (logiciel

d’arduino) et C++ ainsi que le langage MATLAB R2009b dans la phase de création du Prototype.

Nous proposons comme continuité à ce travail et ce dans le cadre d’un projet de fin d’études d'ajouter un bras a l’aide d’un servo-moteur pour évacuer les faïences marquées sur un deuxième convoyeur. Nous estimons que notre module pourra être exploité dans des applications industrielles.

Annexes

- **La carte Arduino Due**

La carte Arduino Due est la première carte Arduino basée sur une architecture ARM 32bit. Usuellement, les cartes Arduino sont plutôt basées sur une architecture ATMega 328.
La carte Arduino Due accueille un microcontrôleur Atmel SAM3X8E ARM Cortex-M3. Elle possède 54 entrées/sorties numériques (dont 12 peuvent être utilisées pour une sortie PWM), 12 entrées analogiques, 4 ports série UART, une horloge à 84 MHz, une connexion compatible USB OTG, 2 DAC (digital to analog), 2 TWI, une prise jack, un header SPI, un header JTAG et enfin un bouton de reset et un bouton d'effacement.

a Caractéristiques techniques de la carte Arduino Due

Les principales caractéristiques de la carte Arduino Due sont les suivantes :

- Microcontrôleur AT91SAM3X8E
- Tension de fonctionnement 3,3 V
- Tension d'alimentation (recommandée) 7- 12 V
- Tension d'alimentation (limites) 6 - 20V
- Nombre d'entrées/sorties : 54 (dont 12 pouvant générer un signal PWM)
- Nombre de ports "Analogique/Numérique" : 12
- Nombre de ports "Analogique" :2 (DAC)
- Courant maximal par E/S : 130 mA
- Courant pour broches 3.3 V : 800 mA
- Courant pour broches 5 V : 800 mA
- Mémoire Flash : 512 KB
- SRAM : 96 KB (2 banques 64 KB et 32 KB)
- Vitesse d'horloge : 84 MHz

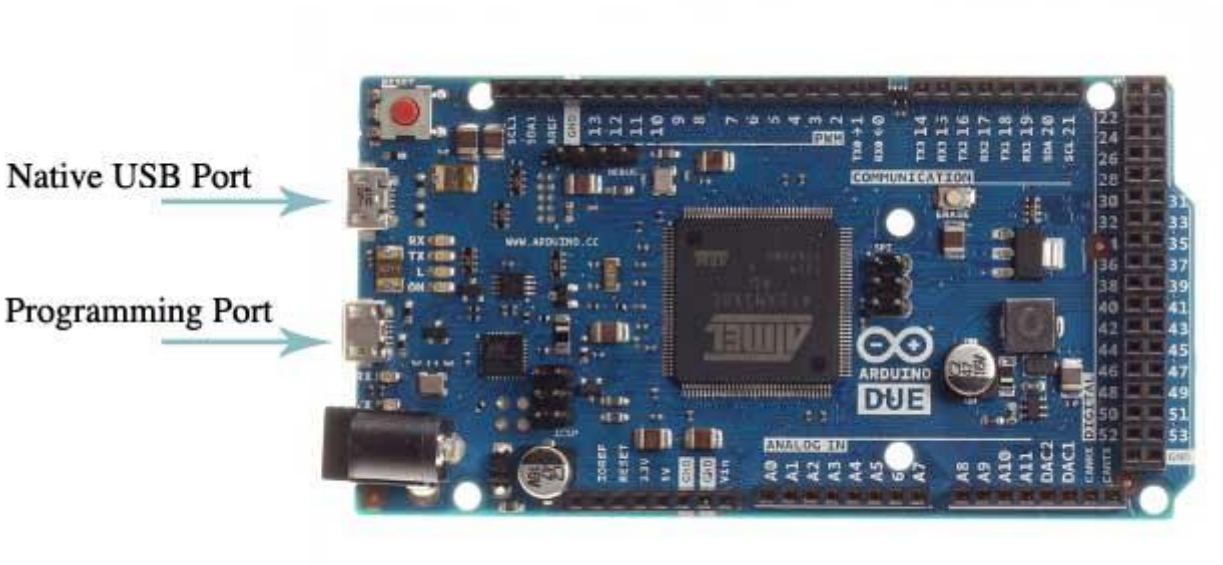
b Avantage du microcontrôleur ARM CORTEX-M3

Le Cortex-M3 propose plus de performance que les habituels microntrôleurs 8bits que l'on trouve sur les autres cartes Arduino. En particulier, l'ARM Cortex-M3 de la carte Arduino Due permet :

- Un noyau 32 bits qui autorise des opérations sur 4 octets dans un seul cycle d'horloge.
- Une vitesse d'horloge de la CPU de 84Mhz
- 96 Ko de SRAM
- 512 Ko de mémoire Flash pour le code
- Un contrôleur DMA qui peut soulager la CPU lors des tâches intensives en mémoire.

c Programmer l'Arduino Due

L'environnement de programmation open-source pour Arduino peut être téléchargé gratuitement (pour Mac OS X, Windows, et Linux). Des tutoriels sont disponibles sur le site officiel d'Arduino. Ces tutoriels vous permettent d'écrire votre premier programme à l'aide des cartes Arduino.



Comme le présente l'image ci-dessus, la carte Arduino Due possède deux connecteurs USB : le programming port et le native USB Port. Les deux sont utilisables pour injecter votre programme dans la carte, cependant, il est recommandé d'utiliser le programming port qui fonctionne même si la MCU a planté auparavant. Reportez vous à la documentation officielle pour en savoir plus.

Bibliographie

[1] Livret Arduino en français par Jean-Noël Montagné, Centre de Ressources Art Sensitif, novembre 2006, sous licence CC ,

[2] https://www.arduino.cc/en/Main/Software

[3] Appendix A: The 'Centre de Morphologie Mathématique', an overview" by Jean Serra, in (Serra *et al.* (Eds.) 1994), pgs. 369-374.

[4] Jean Serra, Ecole des Mines de Paris, 2000.

[5] B. Gugger, des photos : première approche de Photofiltre lexique_images.odt - TICE CRDP Bourgogne, avril 2006.

[6] M. Khouadjia, H. Khanfouf, and S. Meshoul. Une approche adaptative pour la segmentation d'images : Implémentation sur la plate-forme multi-agents netlogo.

[7] V.Wu and R. Manmatha. Document image clean-up and binarization. Proceedings ofIS&T/SPIE Symposium on Electronic Imaging, 3305 :263–273, 1998.

[8] J. Bernsen. Dynamic thresholding of grey-level images. In Proc. Eighth Int 'l Conf.on Pattern Recognition, pages 1251–1255, 1986.

[9] W. Niblack. An introduction to digital image processing. Prentice Hall, July 1986.

[10] Hadjila Feth Allah et Bouabdallah Réda, Reconnaissance des visages en utilisant les réseaux de neurones. Mémoire d'ingénieur. Université de Tlemcen. 2003.

[11] http://dept-info.labri.fr/~vialard/Traitement/cours/cours2.pdf

[12] Yoann Sculo, Introduction au traitement d'images Détection de contours et segmentation. Université de technologie de Troyes.2009

[13] http://www.profil-couleur.com/ec/102-espace-couleur-tsl.php

[14] http://www.la-photo-en-faits.com/2013/05/RVB-CMJN-TSL-conversion-definition.html

[15] Laboratoire de Traitement d'images et Reconnaissance de Formes, LTIRF/INPG, 46, avenue Félix-Viallet, 38031 GRENOBLE CEDEX .

[16] https://dpt-info.u-strasbg.fr/~cronse/TIDOC/MM/deof.html

[17] http://opencvexamples.blogspot.com/2013/09/find-contour.html

[18] https://fr.wikipedia.org/wiki/RS-232