

Correction des yeux rouges



par

AZAIZ Hana & MEJRI Abir
Master M2 Informatique
Encadré par : RIAZANOFF Serge

TABLE DE MATIERE :

INTRODUCTION :	2
I. PHENOMENE DES YEUX ROUGES :	3
II. PHASE DE LA DETECTION :	6
III. PHASE DE LA CORRECTION :	16
IV. ANALYSE DES RESULTATS EXPERIMENTAUX :	20
CONCLUSION:	21
BIBLIOGRAPHIE	22

INTRODUCTION :

Les yeux rouges sont un problème classique. C'est un effet qui s'observe en photographie dans des clichés pris au flash dans un environnement faiblement éclairé: les personnes photographiées ont les pupilles de leurs yeux totalement rouges sur la photo. L'effet s'observe aussi chez les animaux, la couleur n'étant alors pas nécessairement le rouge.

Par conséquent, on expliquera dans la première partie de ce document les raisons physiques de ce phénomène et la façon permettant d'éviter l'apparition des yeux rouges dans nos photos. Ensuite, on présentera les étapes essentielles de notre algorithme qui sert à faire une détection automatique des yeux rouges et de les corriger. Enfin, on présentera un bilan pour l'analyse des résultats obtenus.



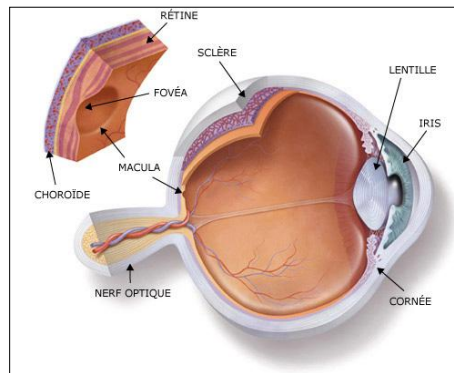
I. PHENOMENE DES YEUX ROUGES :

Les yeux rouges, quelle plaie n'est-ce pas? Cela peut gâcher une belle photo réussie. Le problème vient-il de l'appareil photo? Ou bien de l'être humain ? Est qu'on peut éviter ces yeux rouge dans nos images photographiques ? En effet, il ya plusieurs raisons responsables à ce phénomène.

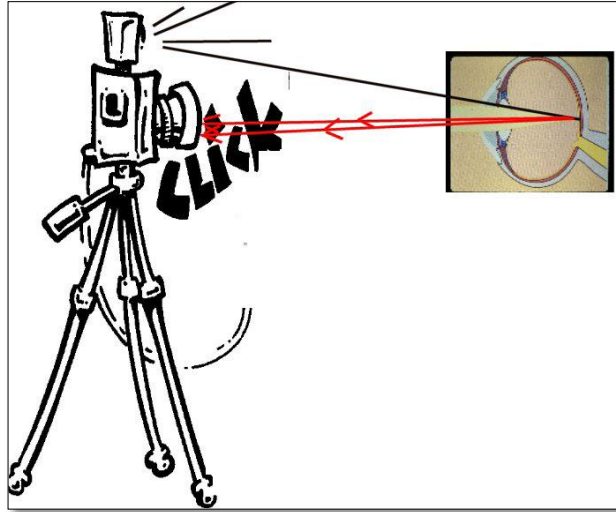
1. POURQUOI LES YEUX DEVIENNENT-ILS ROUGES ?

A. PUPILLE NON PRETE FACE AU FLASH:

Nos yeux sont composés de l'iris (la partie colorée qui nous donne des yeux bleus, verts, etc.) et de la pupille, plus sombre. Quand la lumière rentre par la pupille, les faisceaux lumineux vont se concentrer pour toucher un point précis de notre rétine: la macula. La macula est donc le point sur la rétine où se forme l'image de ce que l'on voit. Les problèmes de vue que l'on corrige par les lunettes sont causés par le fait que l'image se forme soit avant la macula, soit virtuellement après.



Dans les conditions d'obscurité, le petit point noir situé au centre de l'œil, la pupille, se dilate offrant une grande porte de passage à la lumière. D'autre part, pour avoir une photo dans ces conditions, on a besoin d'utiliser le flash. Ce dernier vient directement et subitement éclairer le fond de notre rétine, la macula, qui est rouge en raison des vaisseaux sanguins irriguant cette zone. La lumière blanche est reflétée par la macula et les vaisseaux sanguins, ce qui donne les tâches rouges et un air étrange aux gens sur les photos...Ainsi, la tâche rouge correspond en fait à l'image du fond de l'œil. La figure ci-dessous illustre bien ce phénomène :



B. MANQUE DE « MÉLANINE » :

La couleur des yeux est déterminée par l'abondance d'un colorant brun-noir, la mélanine, dans la partie antérieure de l'iris de l'œil (l'iris, c'est la partie colorée de notre œil). Les personnes aux yeux bleus ne possèdent pas de mélanine dans cette partie de l'iris mais elles en ont quand même un peu dans la partie plus profonde. L'œil apparaît alors bleu pour la même raison que l'eau profonde nous apparaît bleue. L'iris disperse la composante bleue de la lumière plus que les autres couleurs qui sont partiellement absorbées. L'iris semblera alors bleu puisque c'est surtout le bleu qui sera réfléchi. Le gène des yeux bleus empêche presque totalement la formation de mélanine à l'avant de l'iris alors que le gène des yeux bruns permet l'accumulation du pigment à cet endroit.

Les nouveau-nés ne possèdent pas de mélanine dans la partie antérieure de l'iris même s'ils sont porteurs du gène des yeux bruns. C'est pourquoi ils ont toujours les yeux bleus. Lorsque le gène des yeux bruns devient actif, le colorant se dépose et l'œil acquiert sa couleur brune ou noire.

Revenant à notre problématique, la faible quantité de mélanine permet la pénétration facile de la lumière dans la rétine, ou plus précisément sa pénétration à la macula. Ainsi, les enfants et les gens ayant les yeux clairs ont d'habitude des yeux rouges dans les images photographiques.



2. COMMENT EVITER LE PHENOMENE DES YEUX ROUGES ?

Alors, on est condamné à rater toutes ses photos nocturnes ? Non. Pour éviter les yeux rouges, il y a plusieurs possibilités :

- Orienter le flash directement vers les yeux, de face. De cette manière, pas de risque de photographier la rétine.
- Demander au sujet de votre photo de regarder brièvement quelque chose d'éblouissant (soleil, lumière vive, etc.) juste avant de le prendre en photo, cela actionnera la contraction de la pupille
- Utiliser un flash anti-yeux rouges. L'appareil envoie deux éclairs lumineux. Le premier est destiné à rétracter la pupille, diminuant alors la quantité de lumière qui pénétrera dans l'œil lors du second éclair.

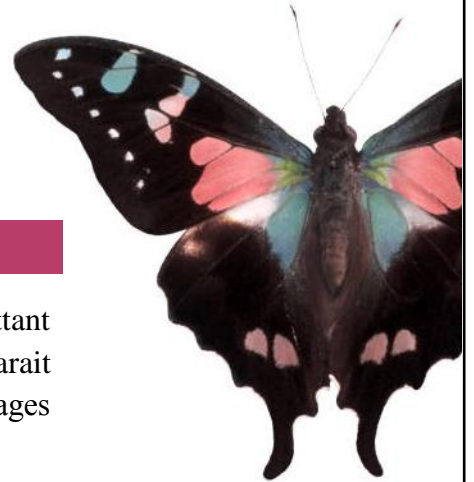
II. PHASE DE LA DETECTION :

Le but de ce projet est de développer procédé efficace permettant une correction automatique de « l'artéfact des yeux rouges » qui apparait souvent, comme on l'a déjà expliqué, dans certaines images photographiques.

Afin d'atteindre notre but, on a commencé par une première étape de détection des zones rouges ou rougeâtres dans les yeux. Ensuite, une fois ces parties sont bien déterminées à l'aide d'un certain nombre de traitements consécutifs, on leurs applique une formule de correction pour récupérer le plus que possible, la couleur originale des yeux

En fait, tout notre traitement a été effectué sur trois photos illustrant ce défaut avec un niveau croissant de difficulté :

- **Photo 1** : est le cas le plus classique avec une luminosité dans le rouge très marquée.



- **Photo 2 :** est un cas plus difficile avec un œil assez rouge à ne pas confondre avec le rouge des lunettes



- **Photo 3 :** est un cas difficile avec un rouge peu marqué.



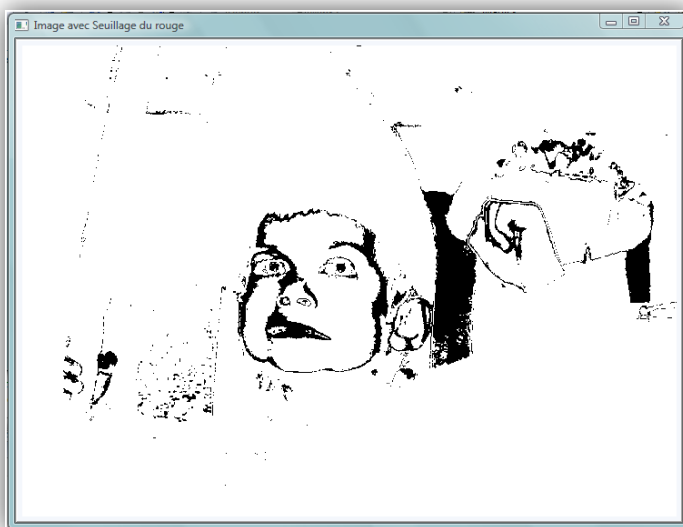
Cette première phase consiste à déterminer avec une bonne précision les zones modifiées des yeux par une certaine couleur rouge. Ce traitement est composé de plusieurs étapes consécutives appliquées aux images dans l'espace de couleur RGB:

1. DETECTION DE LA COULEUR ROUGE :

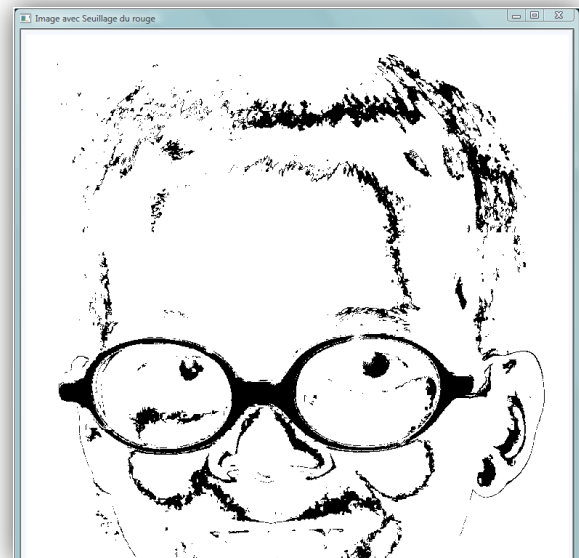
Le premier traitement qu'on applique à l'image est un seuillage pour détecter la couleur rouge. En fait, on a essayé, avec un seuil bien choisi, de trouver tous les pixels ayant une couleur qui remplit les conditions présentes dans la formule suivante :

```
// Condition remplie par un pixel rougeatre:
if(( (red>160) && (green<150) && (blue<150) && (red>green) && (red>blue) ) || ( (red>50)
    && (green<50) && (blue<50) && (red>green) && (red>blue) ) )
{
    // si le pixel est rouge alors il est dessiné en noir dans l'image binaire du
    masque:
    scalaire.val[0]= 0;
}
```

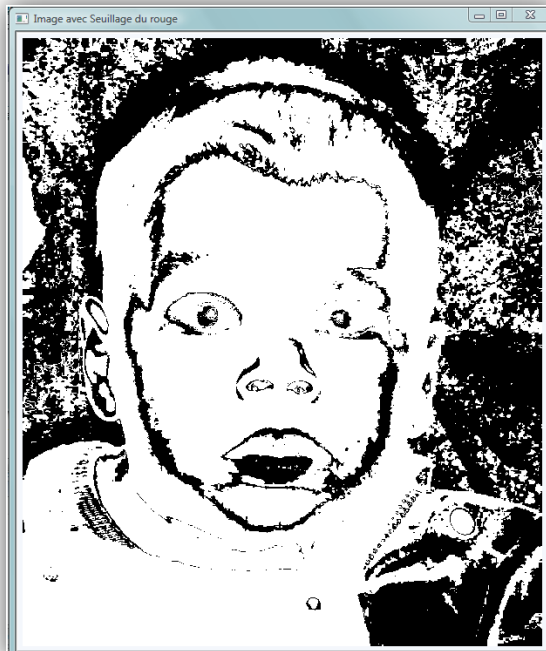
Cette première étape a été, alors, réalisée sur les trois images qu'on a décrit précédemment avec les quelles on a obtenu des résultats satisfaisants stockés dans des images binaires (les pixels rouges sont représentés par la couleur noire).



1. Résultat obtenu sur : photo1_pirate.jpg



3. Résultat obtenu sur : photo2_oeil_lunettes.jpg



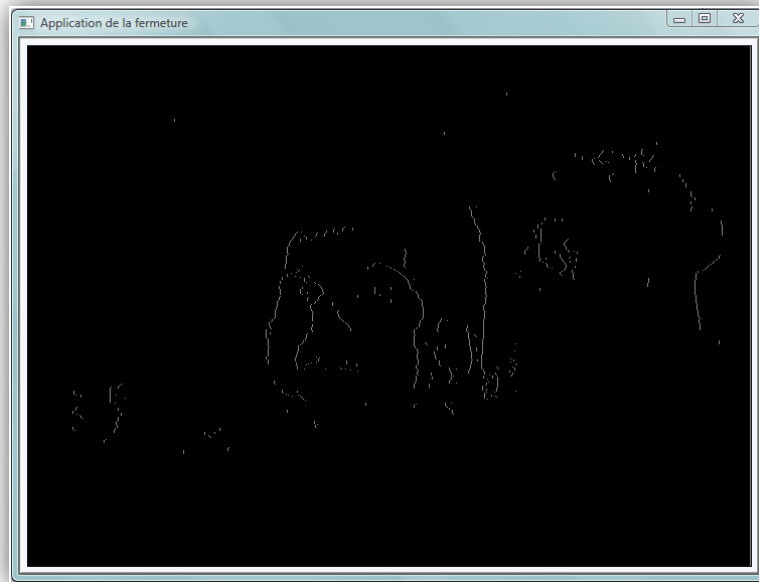
2. Résultat obtenu sur : photo3_rouge_leger.jpg

2. APPLICATION DE LA FERMETURE:

Etant donné que le masque du seuillage n'est pas toujours parfait et l'image originale peut aussi bien contenir des pixels rouges isolés, on a choisi d'appliquer une étape supplémentaire de fermeture qui se compose d'une dilatation de l'image résultante du seuillage précédent suivie d'une deuxième étape d'érosion comme suit :

```
//Dilatation de l'image:  
cvDilate(redMask, img2, NULL, 1);  
  
//Erosion de l'image de la dilatation:  
cvErode(img2, img3, NULL, 1);
```

Et voici le résultat de cette transformation sur la photo1:



Résultat obtenu sur : photo1_pirate.jpg

3. DIVISION EN ZONES CONNEXES :

Pour pouvoir distinguer les deux yeux rouges des autres objets, on a choisi de diviser l'image binaire qu'on a obtenue, en zones connexes de pixels rouges.

Il s'agit de regrouper chaque pixel rouge avec ceux de son voisinage et présentant les mêmes caractéristiques (ici on parle des pixels noirs dans l'image binaire). Pour ce, on récupère ces zones à l'aide d'un opérateur qui permet de leur dessiner les contours :

```
//Récupérer les contours des zones connexes de l'image suillée:
```

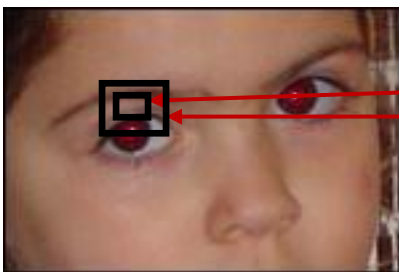
```
Int Nc = cvFindContours(img3,storage,&first_contour,sizeof(CvContour),CV_RETR_LIST);
```

Une fois on a obtenu ces zones de pixels rouges, on va procéder d'un certain nombre de filtrages consécutifs qui vont nous permettre enfin de ne récupérer que les deux contours des zones des deux yeux qu'on cherche à détecter.

A. Détection des pixels blancs et de la peau :

Les deux yeux rouges, qu'on cherche à détecter, représentent, en fait, deux zones de pixels rouges entourés d'un certain nombre de pixels blancs et d'autres appartenant à la peau humaine. On a essayé alors d'entourer chaque groupe de pixels rouges avec :

- ✓ Un premier rectangle : (de taille $w*h$) il présente le tout petit rectangle encadrant complètement la zone de « l'œil rouge ».
- ✓ Un deuxième rectangle : il est plus grand que le premier en largeur et en hauteur avec le coefficient $(w*h)/4$. Il permet de trouver le nombre de pixels blancs et ceux qui présentent des caractéristiques de la peau

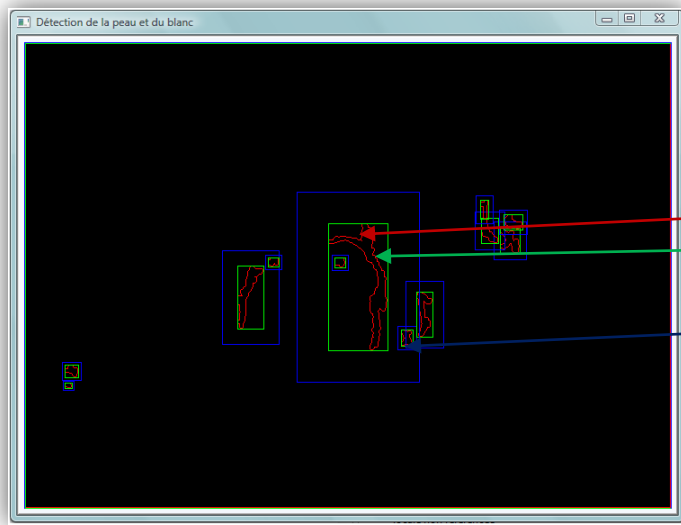


- Rectangle contenant la partie rouge
- Rectangle contenant les pixels blancs et de la peau

Donc, durant cette étape, on filtre l'image binaire en dessinant les deux rectangles décrits ci-dessus et ne laissant que ceux contenant un certain nombre seuil de pixels blancs et de peau :

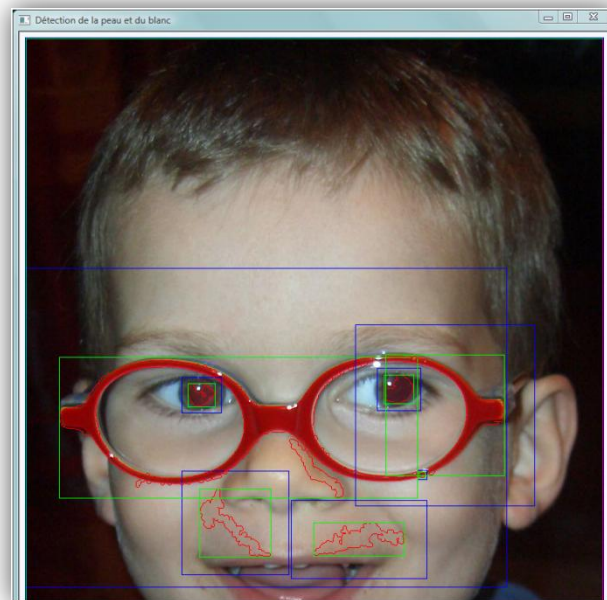
```
//définir le rectangle intérieur:  
  
rect = cvBoundingRect(c,1);  
  
//seuil de detection de l'oeil par skin:  
  
double seuilNonSkin = (9*(rect.height*rect.width))/4;  
  
double seuilBlanc = (5*(rect.height*rect.width))/4;
```

Ce premier filtrage nous a bien permis d'éliminer un bon nombre de zones obsolètes et ce résultat est bien identifié avec les photos qu'on a utilisées :



- Contour de la zone rouge
- Le premier rectangle délimitant la zone rouge
- Le deuxième rectangle

Résultat obtenu sur : photo1_pirate.jpg

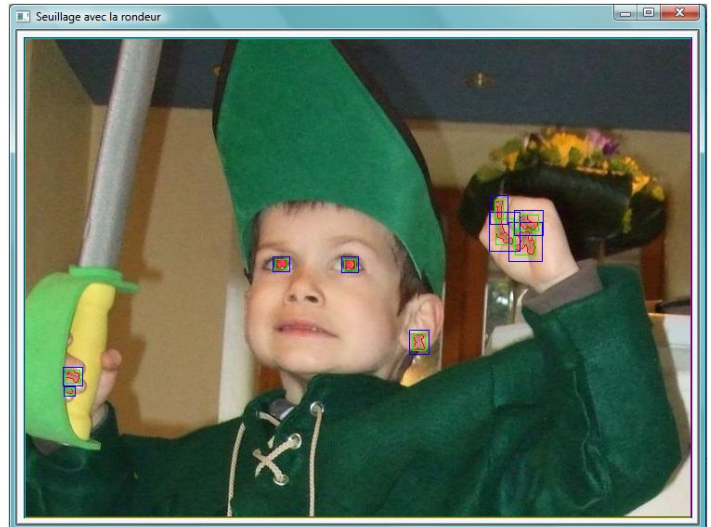


Résultat obtenu sur : photo2_oeil_lunettes.jpg

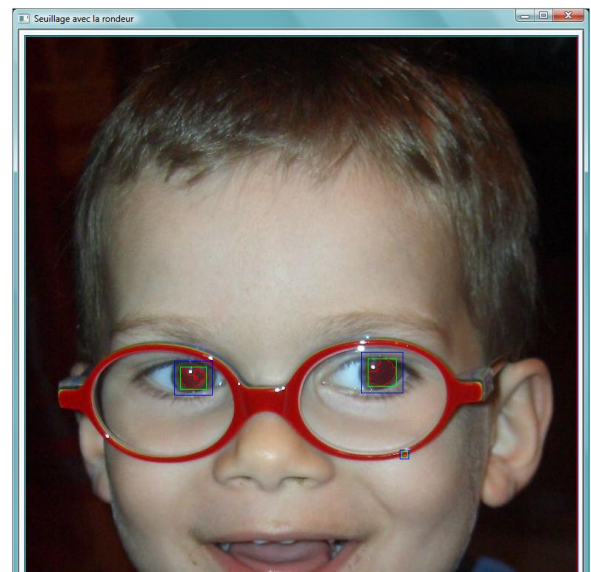
B. Filtrage par critère de rondeur :

Après le filtrage des zones par le critère de la couleur blanche et de celui de la peau, on peut procéder par l'élimination d'autres zones qui sont au dessous d'un certain seuil de rondeur. En fait, les deux yeux rouges sont bien ronds, donc ce critère a l'air d'être déterminant pour nos images.

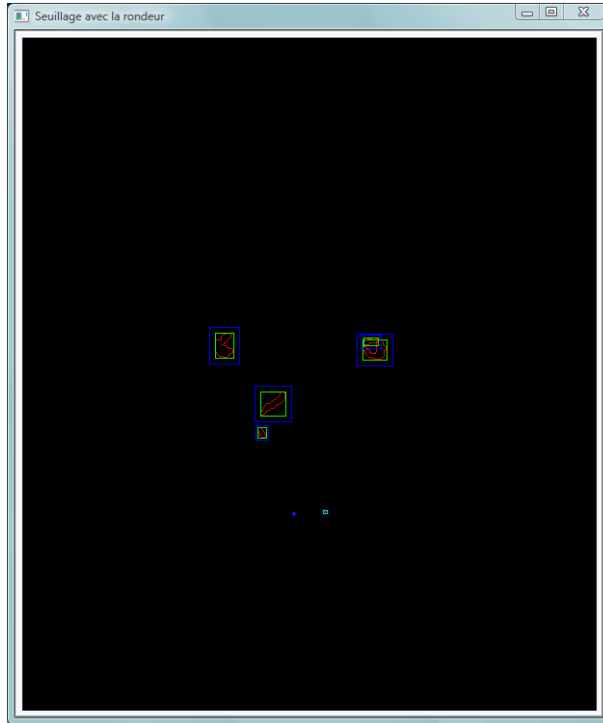
```
//Calcul de la rondeur de la zone détecté:  
  
roundness = (4*Pi*area)/(perimetre*perimetre);
```



Critère de rondeur appliqué sur : photo1_pirate.jpg



Critère de rondeur appliqué sur : photo2_oeil_lunettes.jpg



Critère de rondeur appliqué sur : photo3_rouge_leger.jpg

C. Filtrage par critère de surface :

La dernière étape de filtrage qu'on utilise est celle qui nous permet d'éliminer les zones possédant une toute petite surface ; en effet l'œil est d'une taille moyen par rapport aux autres parties qui nous restent après les étapes précédentes de filtrage.

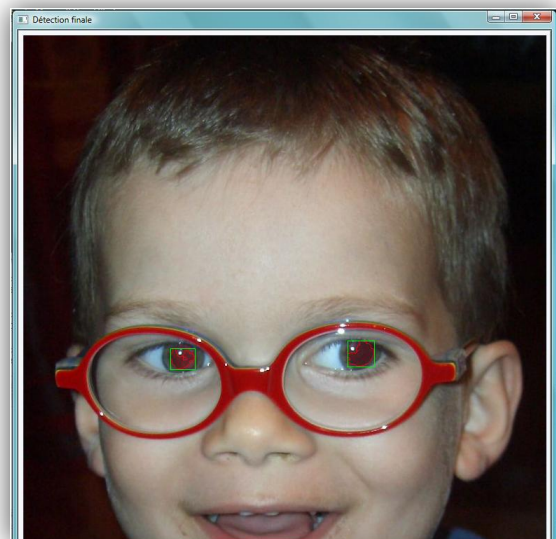
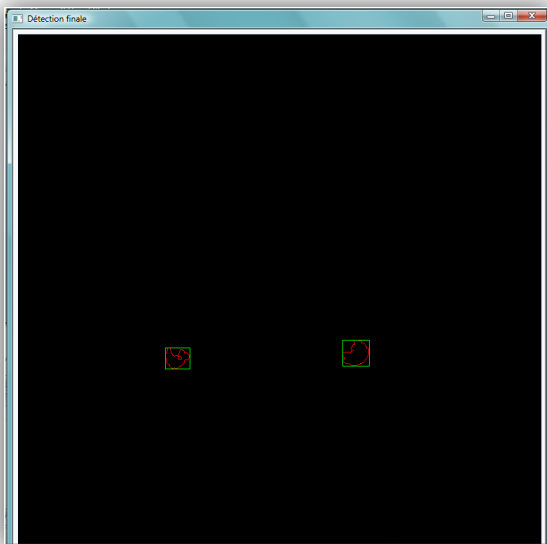
On récupère les surfaces des zones restantes à l'aide de la procédure suivante :

```
//Récupérer la surface incluse dans le contour:
area = fabs (cvContourArea (c,CV_WHOLE_SEQ) );
```


Donc cette étape fournit le résultat final de la phase de la détection, et on peut affirmer qu'on a eu de bons résultats sur chacune des trois photos qu'on travaille avec :



Résultat final de la phase de la détection sur : photo1_pirate.jpg



Résultat final de la phase de la détection sur : photo2_oel_lunettes.jpg

III. PHASE DE LA CORRECTION :

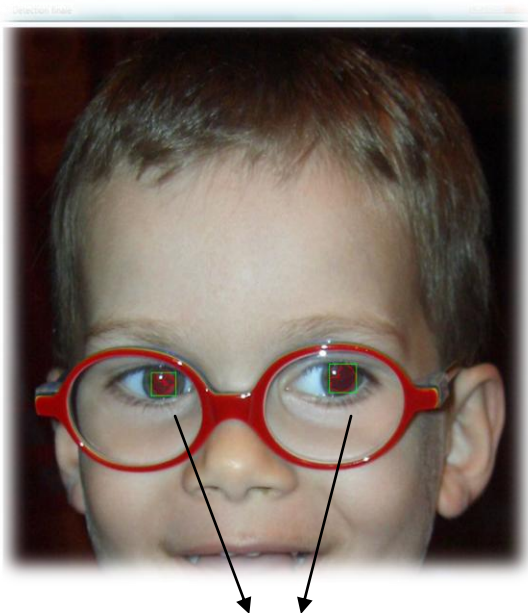
Après la phase de détection, la rétine de l'œil est marquée par un contour rouge. Dans cette étape, on va essayer de corriger cette zone et de récupérer la couleur originale de la pupille. L'approche de notre algorithme se base sur le fait de diminuer la composante rouge. Celle-ci est fortement importante avant la correction, alors que pour les deux autres composantes : bleu et vert, leurs valeurs sont moins importantes.



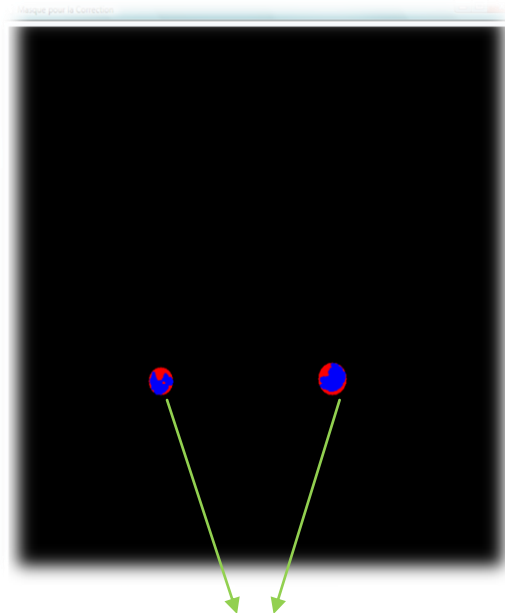
1. ENTOURER LA ZONE DE CORRECTION :

Comme on a mentionné précédemment, la zone détectée par notre algorithme n'est pas parfaitement ronde. Pour limiter exactement la tâche à corriger, le premier essai était de prendre le plus petit rectangle contenant l'œil rouge. Il s'est avéré, lors de notre travail, que pour bien entourer la zone et rendre la correction parfaite, il est préférable de travailler avec le plus petit cercle contenant cette zone.

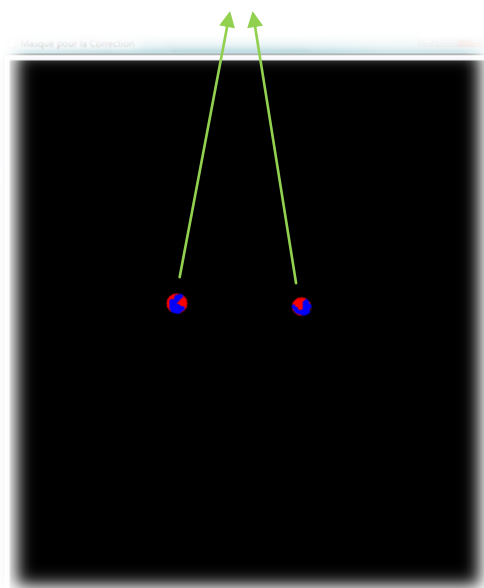
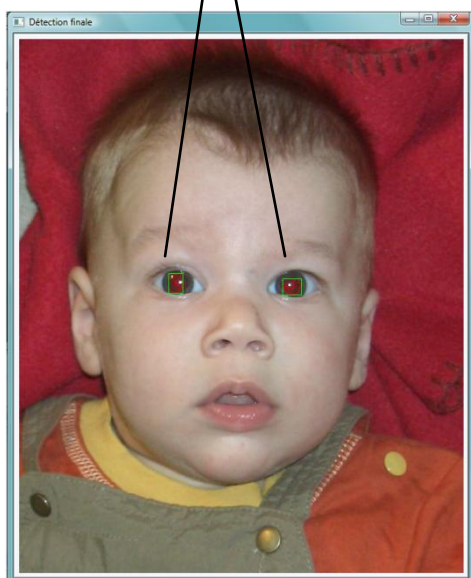
On a besoin d'un masque contenant les deux cercles avec les deux zones détectées afin de faciliter la correction. En effet, on exécute notre correction dans l'image initiale, plus précisément dans les zones marquées dans le masque en rouge (les cercles qui entourent les zones détectées dans la phase précédente).



Les yeux rouges détectés



Les deux plus petits cercles contenant les zones détectées



2. CORRIGER LA COULEUR DES YEUX ROUGES :

La seconde partie est plus artistique et par là même plus subjective, plus difficile à évaluer : elle consiste à trouver un type de correction qui donne un rendu naturel à l'œil rouge détecté.

Nous avons étudiés des outils statistiques que nous avons mis en place pour évaluer ces algorithmes, et enfin, nous avons comparés les différents types de correction envisageables et celle que nous avons choisie afin d'obtenir le rendu le plus naturel possible.

En effet, notre approche utilisée se base sur la diminution de la composante « rouge » et l'augmentation des composantes « bleue » et « verte » selon ces calculs :

```
//cercle rouge

if((redMask == 250)&&(greenMask == 0)&&(blueMask ==0))

{

    //changer le pixel de la zone rouge:

    pixel.val[2] = red/2;

    pixel.val[1] = green ;

    pixel.val[0] = blue ;

    cvSet2D(img, y, x, pixel);

}
```

La 1^{ère} Photo Initiale



La photo Corrigée



La 2^{ème} Photo Initiale



La Photo Corrigée



La 3^{ème} Photo Initiale



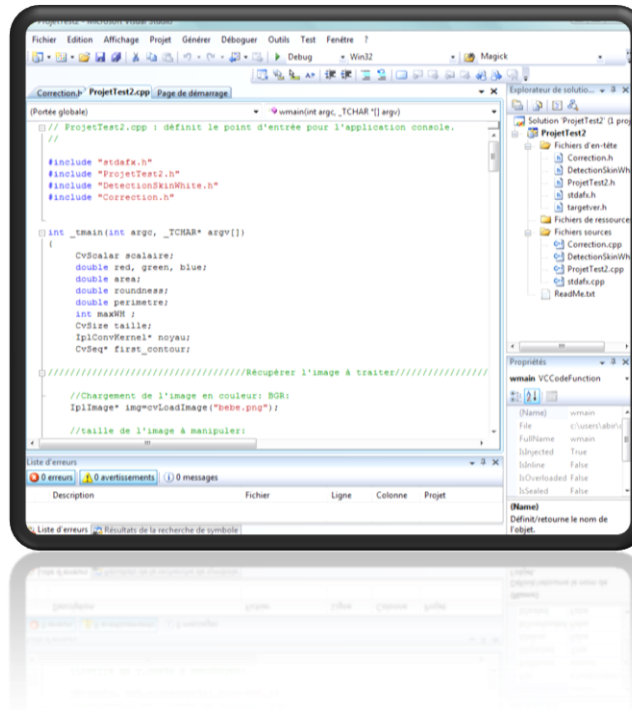
La Photo Corrigée





IV. ANALYSE DES RESULTATS EXPERIMENTAUX :

Notre algorithme a été implémenté sur Visual Studio avec le langage de programmation C++ en utilisant OpenCV qui nous a facilités la tâche en utilisant des fonctions prédéfinies (pour la détection de contour des composantes connexes, le traçage des plus petits rectangles contenant les zones connexes).



En appliquant les différentes étapes de l'algorithme défini précédemment, on a réussi à corriger les yeux rouges pour les images proposées dans l'énoncé d'une manière efficace. Pourtant, on remarque que pour la partie de correction, le choix de couleurs reste un point trop sensible. Pour valider cet algorithme, il faut mieux le tester sur des centaines d'images et de voir la combinaison la plus convenables pour le rapport entre les couleurs de corrections.

CONCLUSION:

Le problème des yeux rouges est très classique dans le domaine du traitement d'images. Cependant, plusieurs algorithmes ont été proposés et des différents outils ont été créés pour résoudre cette problématique. Notre algorithme illustre la détection automatique qui se base sur plusieurs critères (seuillage, détection de la peau, détection du blanc de l'œil...) et il propose aussi la correction de ces yeux en choisissant un rapport bien déterminé entre les différentes composantes.

Quelques résultats peuvent être améliorés en ayant recours à d'autres méthodes de traitements d'images. Pour conclure, ce projet nous a permis de découvrir « OpenCV » qui nous a facilité l'implémentation de notre algorithme. En plus, on a eu l'opportunité d'enrichir nos connaissances dans le domaine de traitement d'images.



BIBLIOGRAPHIE

❖ Les Articles :

- Automates red-eye detection and correction in digital photographs:
 - Lei Zhang, Yanfeng Sun, Mingjing Li, Hongjiang Zhang

- Automatic Red-Eye Removal based on Sclera and Skin Tone Detection:
 - Flavien Volken, Johann Terrier, Patrick Vandewalle

❖ Les Sites Internet:

<http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html>

www.wikipedia.org

www.developpez.com