

MATLAB
The Language of Technical Computing



ETUDIANTS :

Romain POLETTE
Colin BOUVRY

TUTEUR:

Stéphane BINCZAK

COMPENSATION DES COMPOSANTES RGB EN FONCTION DE LA COULEUR D'ÉCLAIRAGE



1	INTRODUCTION.....	3
1.1	Généralités.....	3
1.2	Exemple concret	3
2	ETUDE PRELIMINAIRE.....	4
2.1	Approche globale matérielle.....	4
2.2	Décomposition de l'étude.....	4
3	ETUDE VIRTUELLE (Théorique).....	5
3.1	Introduction	5
3.2	Mise en situation	5
4	ACQUISITION D'IMAGES PAR CAMEA CCD COULEUR.....	7
5	TRAITEMENT D'IMAGES AVEC MATLAB.....	8
5.1	Généralités.....	8
5.2	Binarisation	8
5.3	Agrégation.....	8
5.4	Erosion.....	9
5.5	Moyenne des composantes RGB de chaque région.....	10
5.6	Calcul et tracé des courbes.....	10
6	ETUDE EN FONCTION D'UN ECLAIRAGE BLEU.....	11
6.1	Effet de la luminosité.....	11
6.2	Etude des composantes RGB.....	12
6.3	Normalisation des composantes R et G et courbes dérivées	12
7	PROBLEMES MATERIELS.....	14
7.1	Le Vidéoprojecteur.....	14
7.2	La Caméra CCD	14
8	CONCLUSION	14

1 INTRODUCTION

1.1 Généralités

Le but global de notre projet est d'étudier la variation des composantes RGB de l'ensemble d'un objet en fonction de la lumière extérieure.

En effet, tout le monde peut constater que lorsque l'on éclaire un objet quelconque avec des lumières de styles différents, cet objet n'apparaît pas avec les mêmes couleurs qu'il aurait eu s'il était éclairé avec une lumière blanche (comme en plein jour par exemple).

Notre travail va consister à simuler ces différentes variations lumineuses, récupérer les variations des composantes RGB de l'objet et les interpréter par différents tracés de courbes. On voudra ensuite établir des équations afin de pouvoir compenser ces dites variations de lumière extérieures, afin que l'objet soit visible avec ses couleurs d'origine (lumière naturelle) sur un écran vidéo.

Les différentes méthodologies employées seront détaillées dans la suite de ce rapport.

1.2 Exemple concret

Pour donner un exemple concret d'utilisation de ce projet en milieu naturel, on peut citer la vidéo sous-marine.

En effet, lorsque l'on filme avec une caméra traditionnelle en milieu immergé, les différents objets n'apparaissent pas avec leurs couleurs réelles. Ceci est dû au phénomène de la diffusion de la lumière sous l'eau. Ainsi tous les objets apparaîtront plus ou moins bleus. De plus, il faut tenir compte de la profondeur et de la distance à laquelle on film l'objet.



Ces différents paramètres ajoutent de nouvelles difficultés pour obtenir les couleurs originelles, mais avec une étude élaborée on pourrait concevoir une caméra qui compenserait automatiquement les variations de couleurs sous l'eau afin d'observer le monde sous-marin en couleurs réelles.

Notre étude sera bien sûr plus simplifiée qu'une étude en milieu réelle, comme nous allons le démontrer au cours des différents chapitres de ce rapport.

2 ETUDE PRELIMINAIRE

2.1 Approche globale matérielle

Pour mener à bien notre projet, nous disposons du matériel suivant :

- PC + Windows XP
- Caméra CCD Couleur Ueye
- Vidéoprojecteur Sanyo
- Logiciel d'acquisition d'image U Eye
- Logiciel MatLab 7 pour le traitement d'images

Pour décrire en quelques mots la mise en place de notre projet, on peut définir l'utilisation du matériel utilisé comme suivant :

- Le vidéoprojecteur va simuler la lumière extérieure
- La caméra CCD permet d'acquérir l'image de l'objet éclairé
- Matlab va nous permettre d'opérer les différents traitements d'images acquises par le logiciel IDS U Eye

La description du matériel est succincte et nous détaillerons l'utilisation de ce dernier au fur et à mesure des chapitres.

2.2 Décomposition de l'étude

Nous commencerons tout d'abord par faire une étude dite virtuelle pour estimer les différents résultats que l'on désire. Cette approche virtuelle se fera uniquement par une simulation sous MatLab.

Ensuite, après une approche très théorique, nous étudierons la mise en place de notre matériel pour faire une expérience avec acquisition d'image par caméra CCD Couleur.

Après avoir fait les différentes acquisitions dont nous avons besoin, on concevra un algorithme de traitement des images acquises et de tracé de courbes grâce au logiciel MatLab 7.

Enfin, nous interpréterons les différentes courbes que nous avons tracées et nous verrons les différentes conclusions que l'on peut en tirer.

Il serait également intéressant de pouvoir retrouver la couleur de l'éclairage à partir d'une image acquise par caméra.

3 ETUDE VIRTUELLE (Théorique)

3.1 Introduction

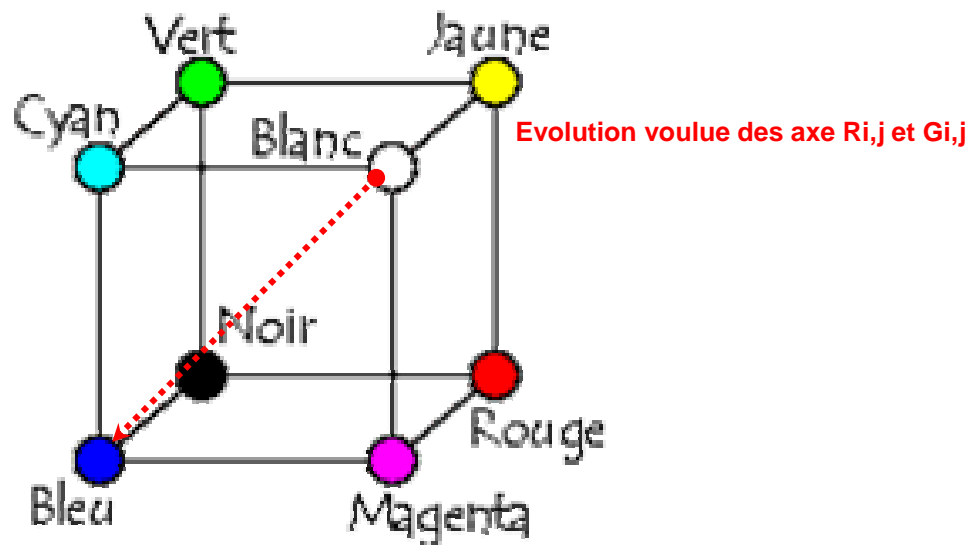
Cette étude virtuelle va nous permettre d'obtenir des résultats purement théoriques. Ces résultats vont permettre de se rendre compte de l'évolution des couleurs en fonction d'un éclairage "virtuel". On en déduira des courbes qui, on le verra plus tard, ne reflètent pas la réalité.

3.2 Mise en situation

Pour se mettre en situation, il faut tout d'abord raisonner dans l'espace RGB. C'est en faisant varier ces paramètres R G B que l'on va simuler une variation de la coloration de la lumière ambiante.

Prenons par exemple une variation d'éclairage qui va de la lumière blanche RGB [255 255 255] vers une lumière purement bleue RGB [0 0 255].

Pour pouvoir obtenir cet éclairage purement bleu, il faut faire décroître les composantes R(i,j) et G(i,j) de l'éclairage en même temps.



Ainsi, pour simuler une variation d'éclairage on appliquera une transformation à chaque pixel de l'image à traiter.

Cette transformation est la suivante:

$$P'(i, j) = P(i, j) * \begin{pmatrix} \alpha r \\ \alpha g \\ \alpha b \end{pmatrix}$$

P (i,j) = Pixel d'origine

P'(i,j) = Pixel transformé

$\alpha x(i,j)$ = coefficient de transformation avec $\alpha r = \alpha g \in [0 : 1]$ et $\alpha b = 1 = \text{cste}$

On commence avec une image théorique créée sous Paintbrush. Ce sera notre image de référence sur laquelle on simulera un éclairage bleu.

On peut visualiser le résultat de la simulation sur quelques images théoriques en fonction de l'éclairage bleu simulé par les différents $\alpha_x (i,j)$:

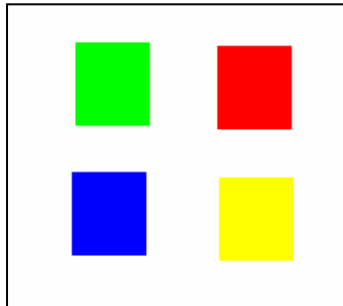


Image de référence
 $\alpha_r = \alpha_g = \alpha_b = 1$

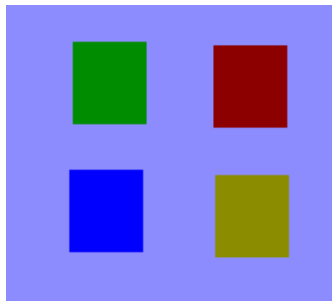


Image Transformée
 $\alpha_r = \alpha_g = 0.55$
 $\alpha_b = 1$

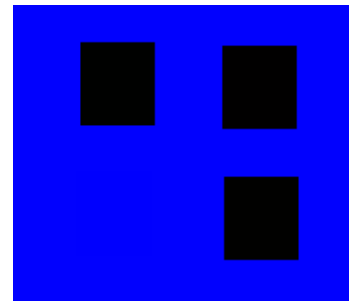


Image Finale
 $\alpha_r = \alpha_g = 0$
 $\alpha_b = 1$

Juste en faisant varier les paramètres α_x , on peut constater l'évolution finale voulue avec un éclairage variant de la lumière naturelle vers une lumière à composante bleue unique.

On remarque donc que chaque pixel de l'image a bien subi une transformation. Cette transformation est celle que l'on aimerait avoir avec une simulation en milieu réel.

Appliquons maintenant le même procédé de transformation d'une image acquise avec notre caméra CCD.

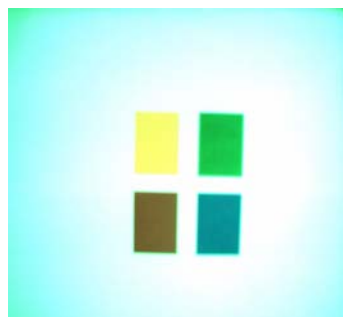


Image de référence
 $\alpha_r = \alpha_g = \alpha_b = 1$



Image Transformée
 $\alpha_r = \alpha_g = 0.62$
 $\alpha_b = 1$

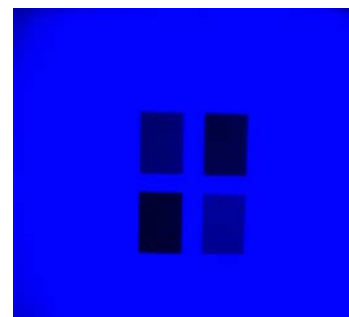
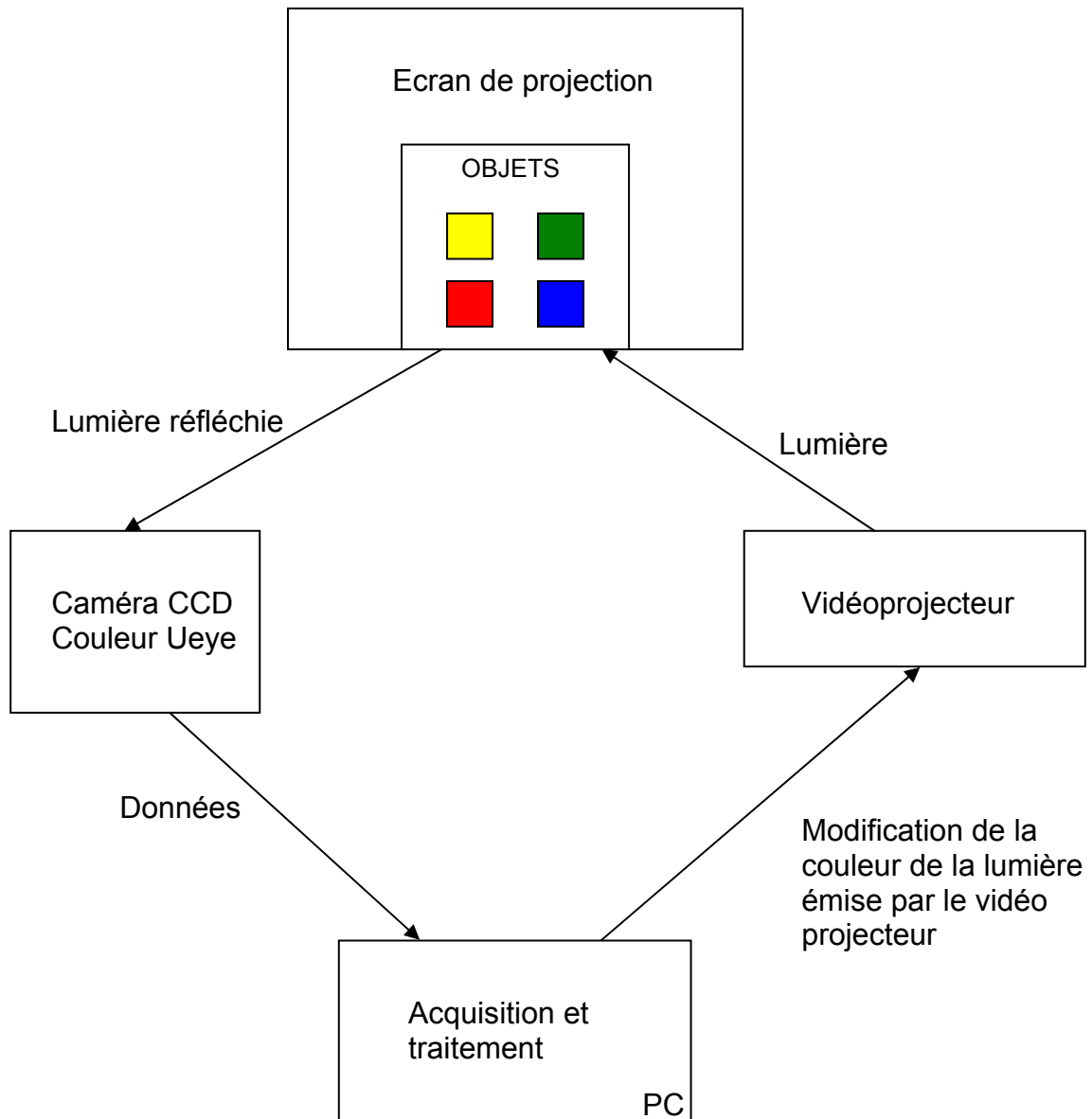


Image Finale
 $\alpha_r = \alpha_g = 0$
 $\alpha_b = 1$

On constate que même en appliquant la même transformation, les résultats visibles sur une image acquise ne sont pas les mêmes que sur une image théorique.

Juste par ce constat, on peut dire que l'étude va se compliquer, car il faudra tenir compte de plusieurs paramètres (que nous verrons par la suite) pour avoir des résultats convenables sur une image acquise par caméra numérique.

4 ACQUISITION D'IMAGES PAR CAMEA CCD COULEUR



Le PC va nous permettre de définir la couleur que le vidéoprojecteur va émettre. Ensuite, le vidéoprojecteur diffusera la lumière vers un écran blanc sur lequel on aura posé les objets que l'on veut étudier. Après, la caméra captera la lumière réfléchie par les objets. Ainsi, le logiciel d'acquisition IDS Ueye nous permettra de stocker les images sur le PC en vue d'un traitement.

5 TRAITEMENT D'IMAGES AVEC MATLAB

5.1 Généralités

Pour le projet, nous n'allons pas utiliser un objet réel, mais plutôt de simples carrés de couleurs imprimés sur une feuille blanche. Chaque carré représentera une région d'un objet.

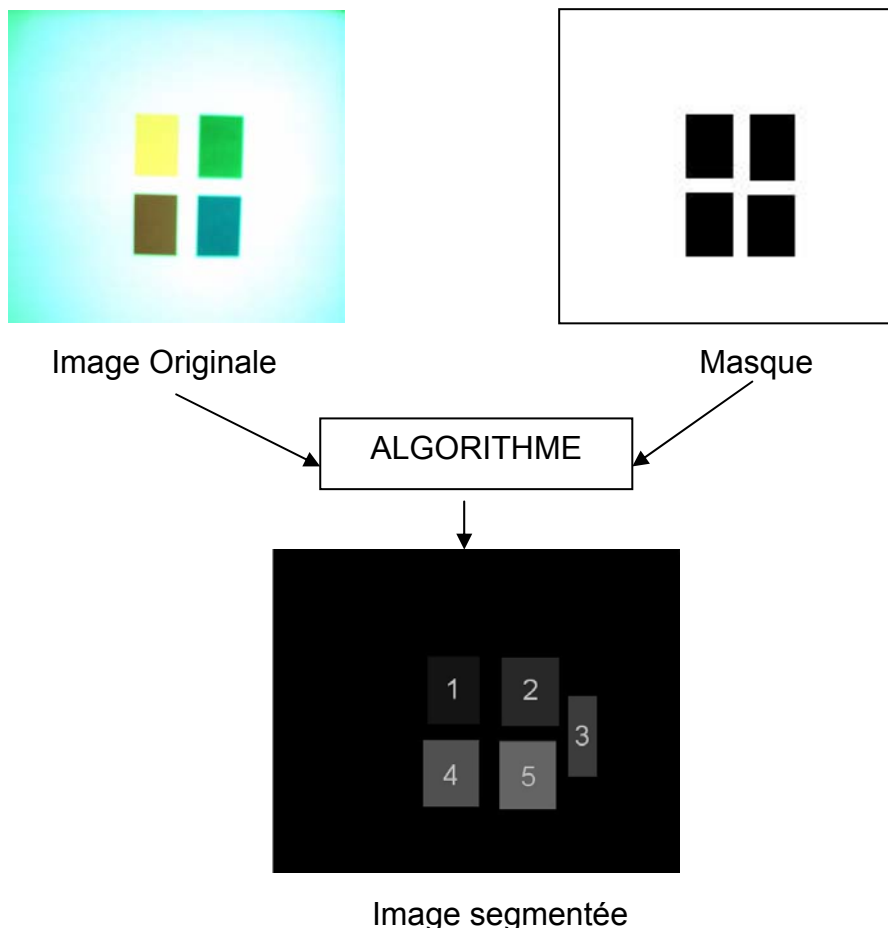
L'image reçue de la caméra est une image couleur RGB 24 bits. Le but pour l'instant est de pouvoir tracer l'évolution des composantes RGB de chaque région d'un objet en fonction d'un éclairage qui passe progressivement du blanc au bleu. Pour cela, il faut opérer des traitements de l'image afin de pouvoir séparer les différentes régions d'un objet, de récupérer les moyennes des composantes RGB de ces régions et de tracer les courbes des moyennes.

5.2 Binarisation

On convertit d'abord notre image couleur en image de niveaux de gris grâce à la fonction RGB2GRAY de MatLab. Ensuite cette image est binarisée (cf. Algorithme en Annexe). Au final, l'image est donc uniquement composée de niveaux Noir 0 ou Blanc 255.

5.3 Agrégation

Afin de faire une agrégation précise de notre image de référence, on utilise un masque. Ce masque consiste à superposer des carrés noirs sur nos carrés d'origine. Grâce à cela, le procédé de détection des régions va se faire plus facilement. En effet, les régions que notre programme aura détectées sur le masque correspondront aux régions de notre objet.



Chaque numéro de région correspondra à une couleur de notre objet d'origine.

Note : La région n° 3 correspond à la couleur du fond de l'objet, autrement dit, cette région représente la lumière émise par notre vidéoprojecteur.

Cette technique est la plus pratique et la plus rapide que nous connaissions pour faire de la détection de région sur une image couleur.

Donc :

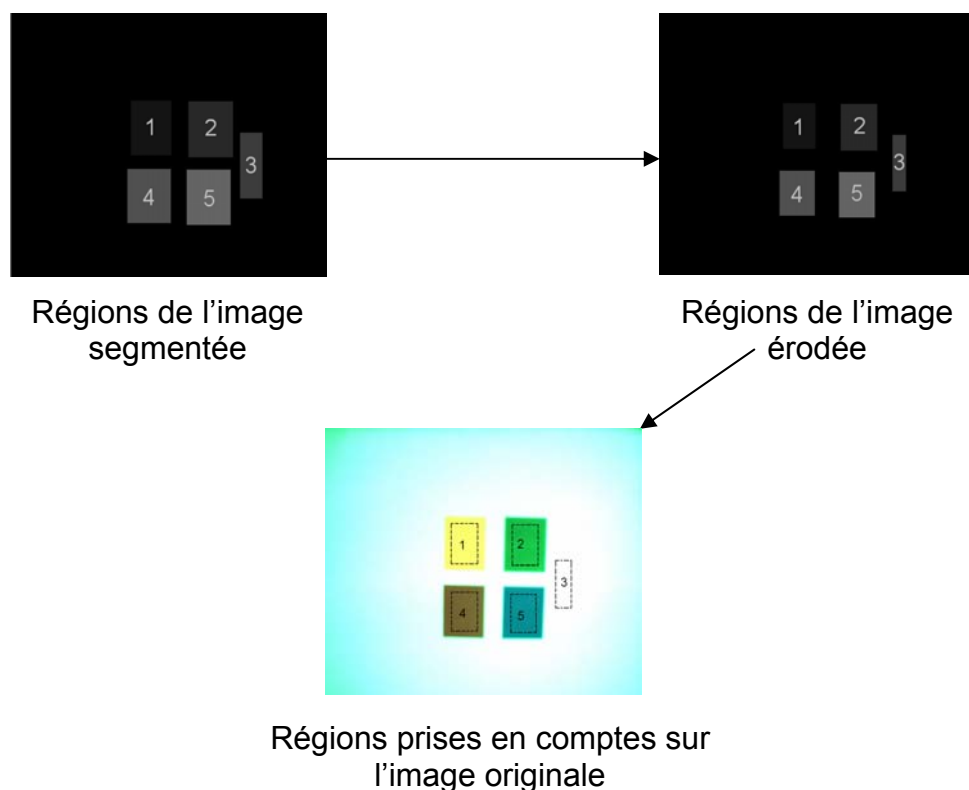
- Région 1 = couleur Jaune
- Région 2 = couleur verte
- Région 3 = fond de l'image = couleur de l'éclairage
- Région 4 = couleur rouge
- Région 5 = couleur bleue

Mais, faire uniquement une agrégation ne suffit pas. En effet, il existe une certaine incertitude du fait que l'on se contente de superposer un masque sur notre image d'origine. Par exemple, les contours de notre masque ne sont pas forcément précis et il peut y avoir un décalage entre deux acquisitions d'images et de ce fait, les contours du masque ne se superposent pas exactement avec la région originale. Les régions ainsi détectées peuvent inclure des pixels qui sont hors de la région couleur d'origine.

Pour remédier à ce problème, il suffit de faire une érosion.

5.4 [Erosion](#)

Afin de recentrer les régions de l'image segmentée au milieu de la région d'origine, il faut appliquer une érosion. Ainsi, on sera sûr de ne prendre en compte les bonnes couleurs de chaque région. Cette opération est nécessaire afin que l'on puisse calculer la moyenne globale de chaque composante RGB de chaque région, afin de pouvoir tracer des courbes par la suite.



Après avoir fait l'érosion, on peut donc calculer la moyenne de la couleur de chaque région sans erreurs.

5.5 Moyenne des composantes RGB de chaque région

Ce procédé va nous permettre de calculer la moyenne des composantes RGB de l'ensemble des pixels de chaque région. En effet, lorsque l'on fait l'acquisition d'une image avec la caméra CCD, on peut constater que tous les pixels d'une même région ne sont pas exactement tous de la même couleur. Pour palier ces défauts, on va faire la moyenne de l'ensemble des composantes RGB de la région, pour homogénéiser la région.

5.6 Calcul et tracé des courbes

Le programme créé sous MatLab va permettre d'effectuer le tracé de l'évolution de la moyenne des composantes RGB pour chaque région en fonction de la variation de l'éclairage.

Nous allons d'abord commencer par faire plusieurs acquisitions avec 3 éclairages de couleurs différentes (Bleu, Rouge, Vert). On notera que la luminosité du vidéoprojecteur est réglée sur 20 (ceci est important comme on pourra le constater dans le paragraphe 5.1).

Cf. ANNEXE A

Commentaires sur Annexe A :

A première vue, on peut constater que l'évolution de chaque composante est non linéaire contrairement à la théorie. On remarque qu'il y a une saturation entre [0:100] pour la région Fond et entre [0 : 70] pour la région de couleur jaune.

En théorie, les composantes autres que la couleur de l'éclairage devraient évoluer de la même façon. Par exemple, pour l'éclairage Bleu, les composantes Rouge et Verte devraient théoriquement évoluer en même temps sans saturation.

On constate également que pour un éclairage Rouge, l'évolution des composantes RGB est erronée. Par exemple, pour la région Fond la composante R devrait rester constante (ou quasi) comme pour l'éclairage Bleu ou Vert.

Conclusions sur Annexe A :

Les courbes obtenues sont donc difficilement interprétables. Ceci est dû en grande partie aux différentes erreurs de chaque composant de notre chaîne de mesures : vidéo projecteur + caméra + couleur des objets.

6 ETUDE EN FONCTION D'UN ECLAIRAGE BLEU

Pour approfondir notre étude, on se contente d'étudier la variation des composantes RGB en fonction d'un éclairage Bleu.

6.1 Effet de la luminosité

Si on veut se mettre en condition de simulation de mesures sous-marines, on sait que la luminosité varie en fonction de la profondeur. On va donc faire plusieurs mesures en jouant sur la luminosité du vidéoprojecteur.

Cf. ANNEXE B

Commentaires sur Annexe B :

Différents test sont fait avec 3 valeurs de luminosité (5, 40 et 60). Ces trois valeurs sont réglées par le vidéoprojecteur.

On constate qu'avec une luminosité de 60, les composantes RGB des régions sont saturées pour un éclairage bleu compris entre [0 : 60] pour les régions Rouge, Verte et Bleu. Les régions Fond et Jaune saturent pour un éclairage compris entre [0 : ~100] : cela est normal car ces régions sont composées de couleurs claires, donc plus sensibles à la variation d'un éclairage bleu.

On descend ensuite la luminosité à 40. Pour les régions Bleu, Verte et Rouge, on constate que le phénomène de saturation a disparu. Mais, pour les régions Jaune et Fond, le phénomène est encore présent pour un éclairage compris entre [0 : ~100]. On en déduit que cela perturbe nos mesures et il faut donc encore diminuer la luminosité de l'éclairage.

Avec une luminosité de 5, le phénomène de saturation n'est plus présent (sauf pour la région Fond où la saturation est encore visible au début des courbes), mais les images acquises par la caméra CCD sont mauvaises. Par exemple avec éclairage blanc + luminosité 5, le fond de notre image n'est pas blanc comme il devrait l'être, on ne peut donc pas exploiter ces courbes si l'éclairage est de mauvaise qualité dès le départ.

Conclusion sur Annexe B :

Même avec peut de points sur les courbes, on constate que les données acquises ne peuvent être exploitée à 100%.

Il faut donc trouver un compromis pour effectuer la suite des mesures. On fixe donc la luminosité du vidéoprojecteur a une valeur de 21. Cette valeur nous semble la mieux appropriée pour effectuer des mesures correctes.

6.2 [Etude des composantes RGB](#)

La luminosité de l'éclairage est donc maintenant de 21. Le programme MatLab va permettre de tracer la moyenne des composantes RGB de chaque région en fonction des variations de l'éclairage.

Matlab permet de tracer les courbes directement sous forme polynomiale.

Par la suite, on va comparer les courbes acquises avec les courbes que l'on a obtenu par l'étude virtuelle (Cf. 3).

Cf. Annexe C

Commentaire sur Annexe C :

Le fait de programmer directement les courbes sous forme polynomiale va permettre de calculer leurs dérivées respectives plus facilement grâce aux fonctions de Matlab (Cf. programme MatLab en Annexe).

Malgré le fait que l'on limite la luminosité à 21, on aperçoit encore le phénomène de saturation pour les régions Jaune et Fond.

Si on ne tient pas compte de la saturation, on peut estimer que l'évolution des composantes RGB de chaque région semble évoluer dans le sens des résultats que l'on a obtenu avec la théorie.

Conclusion sur Annexe C :

Bien que les courbes obtenues par acquisition caméra tendent à se rapprocher de la théorie, on constate qu'elles évoluent de façon non linéaire.

Afin d'étudier plus en détail les variations des composantes R et G, on décide de les normaliser par rapport à la composante B.

6.3 [Normalisation des composantes R et G et courbes dérivées](#)

La normalisation des composantes R et G par rapport à la composante B va permettre de déterminer l'évolution de R et G en fonction de l'éclairage Bleu.

Cf. Annexe D

Commentaires sur Annexe D :

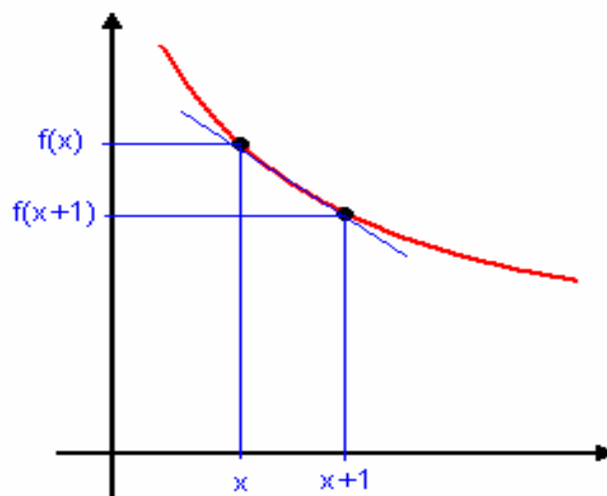
Normaliser les composantes R et G revient à les diviser par la composante B. Cette composante B est censée être constante, comme on a pu le voir sur les courbes théorique de l'Annexe C. Le problème est que sur les courbes réelles, la composante Bleue n'est pas constante.

On constate qu'en ce qui concerne les régions Bleue, Verte et Rouge, les courbes de normalisations ne varient pas de façon totalement linéaire. Pour ces 3 régions, c'est uniquement une partie des courbes qui tend à avoir un comportement linéaire.

On constate alors un problème en ce qui concerne les régions Jaune et Fond. Les courbes de normalisation de ces 2 régions ont un comportement totalement non linéaire. On observe même une régression pour la région Jaune et un phénomène de saturation pour la région Fond.

En ce qui concerne les courbes de dérivées, elles sont directement obtenues grâce à la fonction MatLab Polyder (Cf. Programme en Annexe), qui permet de calculer directement la dérivée d'une fonction polynôme (c'est pourquoi nos courbes sont directement mis sous une équation polynomiale).

Les dérivées sont censées donner une idée de la compensation à apporter pour corriger les variations des composantes RGB afin qu'une région d'un objet retrouve ces composantes d'origines sous un éclairage bleu. En effet, on considère que la compensation à apporter est en fait la pente qui relie 2 points d'une courbe de composante.



On appelle $\delta(t)$ cette compensation. On se ramène donc à la formule fondamentale de la dérivée.

$$\delta(t) = \frac{f(x) - f(x+1)}{x - (x+1)}$$

On constate que les courbes que l'on a acquises ne sont pas exactes. Donc, les courbes dérivées obtenues seront forcément fausses.

Conclusion sur Annexe D :

On peut conclure globalement que les résultats obtenus, tant au point de vue de la normalisation que des dérivées, sont inexacts. Il est donc assez difficile de pouvoir exploiter ces courbes de l'Annexe D et d'en tirer des conclusions pertinentes afin de pouvoir aller plus loin dans l'étude. En ce qui concerne les dérivées, il faudrait trouver un moyen plus efficace de les calculer, car les courbes obtenues sont totalement fausses.

7 PROBLEMES MATERIELS

Tout le long du projet, nous avons eu certains problèmes en ce qui concerne l'exploitation des courbes d'acquisition caméra. Cela est surtout dû à la mauvaise qualité des mesures. Cette qualité dépend surtout des problèmes de matériel.

En effet, on a pu constater que plusieurs erreurs s'accumulaient dans la chaîne de mesure.

7.1 Le Vidéoprojecteur

Le vidéoprojecteur ne reproduit pas exactement les composantes RGB de l'éclairage voulu. D'après une étude réalisée par des étudiants en Master (Gabrielle MENU et Loïc PEIGNÉ) un vidéoprojecteur ne restitue pas une luminance linéaire. Dans notre cas, il est donc normal d'obtenir des courbes de composantes RGB non linéaire quand on fait varier l'éclairage du blanc au bleu.

7.2 La Caméra CCD

Les erreurs de mesures sont également dues au fait que la caméra CCD ne restitue pas fidèlement les couleurs. On ne sait pas exactement si cela provient de l'objectif ou du capteur CCD de la caméra. En fonction de l'ouverture du diaphragme, il y peut y avoir saturation de la lumière. Il faut donc veiller à ce que le diaphragme ait une ouverture correcte dès le début des acquisitions. Les erreurs sont flagrantes lorsque l'on éclaire un objet avec une lumière rouge, en effet, on constate que le fond de l'objet n'apparaît pas en rouge mais en nuances de rouge-gris. Les courbes obtenues n'auront donc aucun rapport avec la réalité.

8 CONCLUSION

Il ne faut rien considérer comme acquis. Même si les résultats peuvent sembler convenables à première vue, on constate par la suite que les résultats obtenus ne sont pas assez pertinents pour pouvoir tirer des conclusions. Il est donc difficile d'aller plus loin dans l'étude du projet.

Le principal problème du projet a été que toutes les erreurs de la chaîne de mesure (Vidéoprojecteur + Ecran de projection + caméra) s'additionnent et donnent donc des résultats erronés.

Pour avoir des mesures correctes qui se rapprocheraient de la réalité, il faudrait refaire l'étude avec un nouveau type de matériel (surtout en ce qui concerne la caméra) ou revoir en partie la méthode de mesure et d'analyse.

ETUDIANTS :

Romain POLETTE
Colin BOUVRY

TUTEUR:

Stéphane BINCZAK

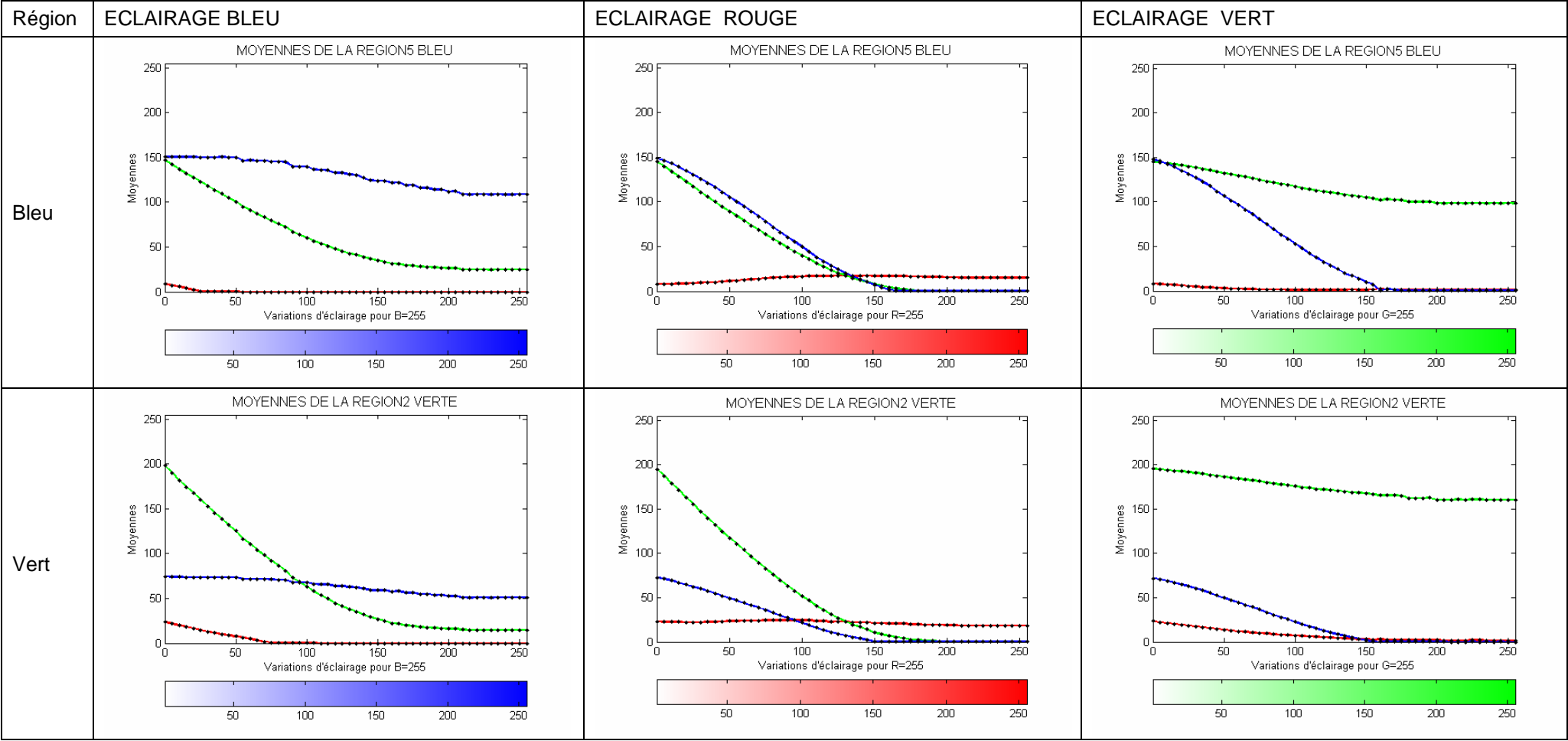
COMPENSATION DES COMPOSANTES RGB EN FONCTION DE LA COULEUR D'ECLAIRAGE

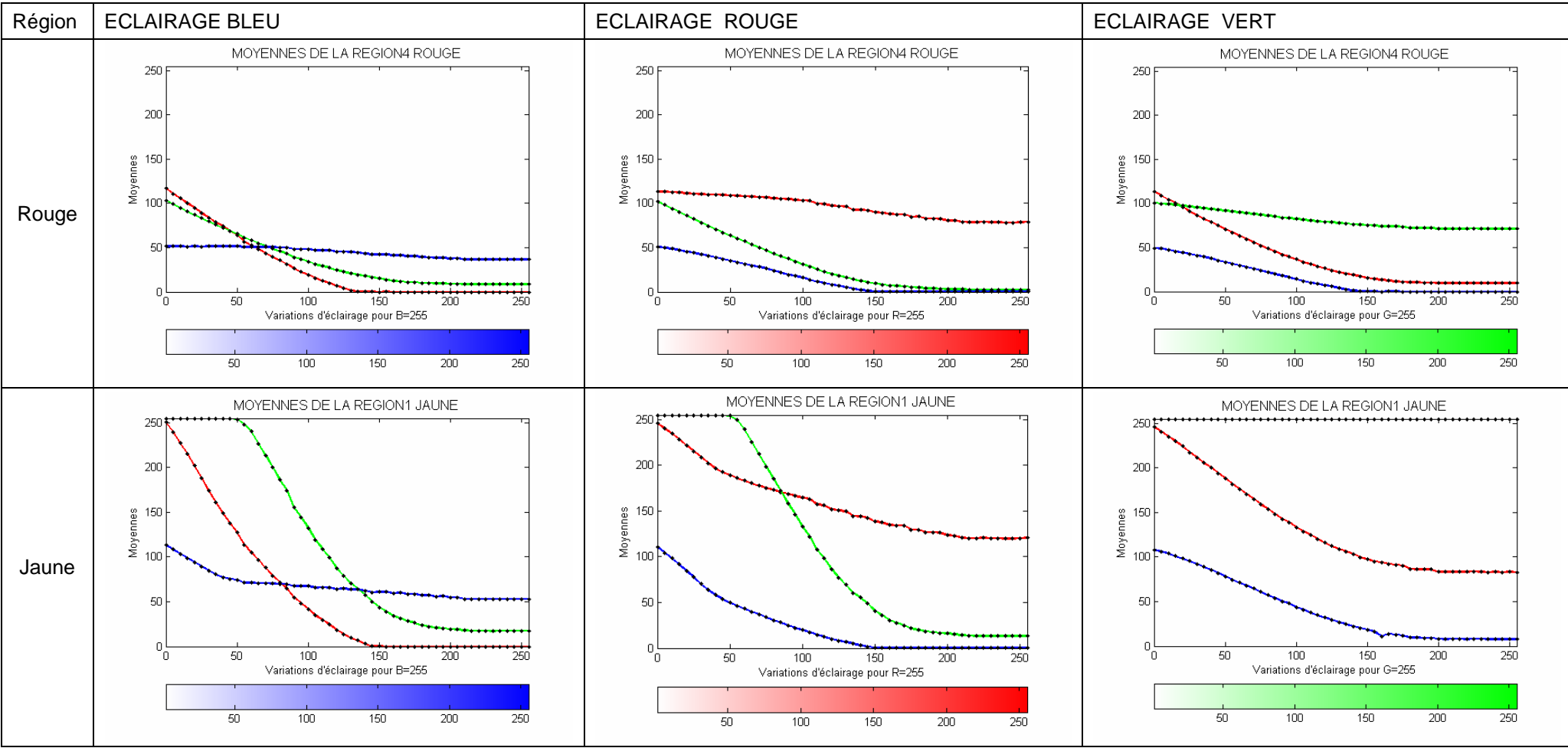
ANNEXES

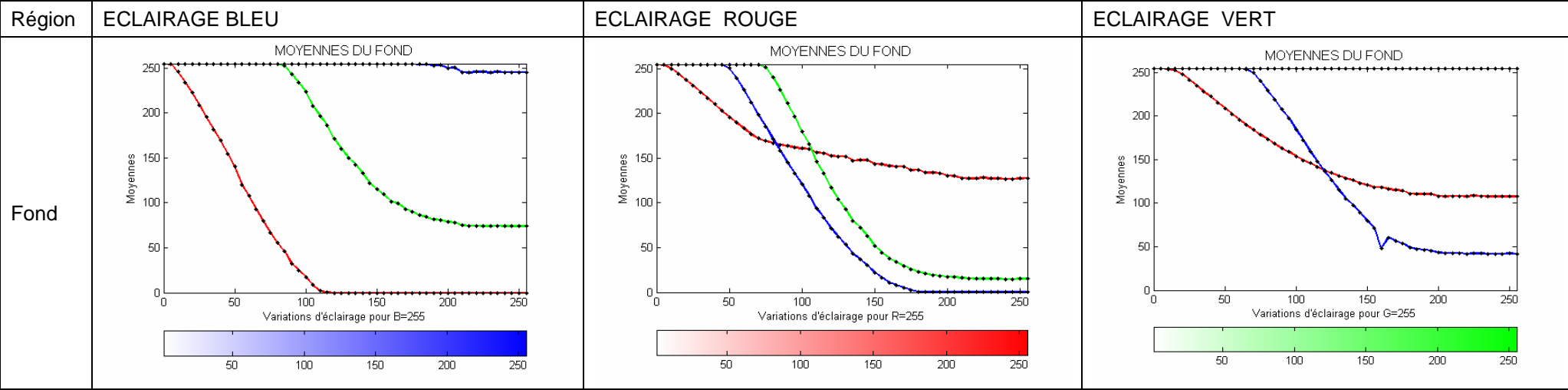


ANNEXE A

Variation des composantes RGB en fonction
de différents éclairages

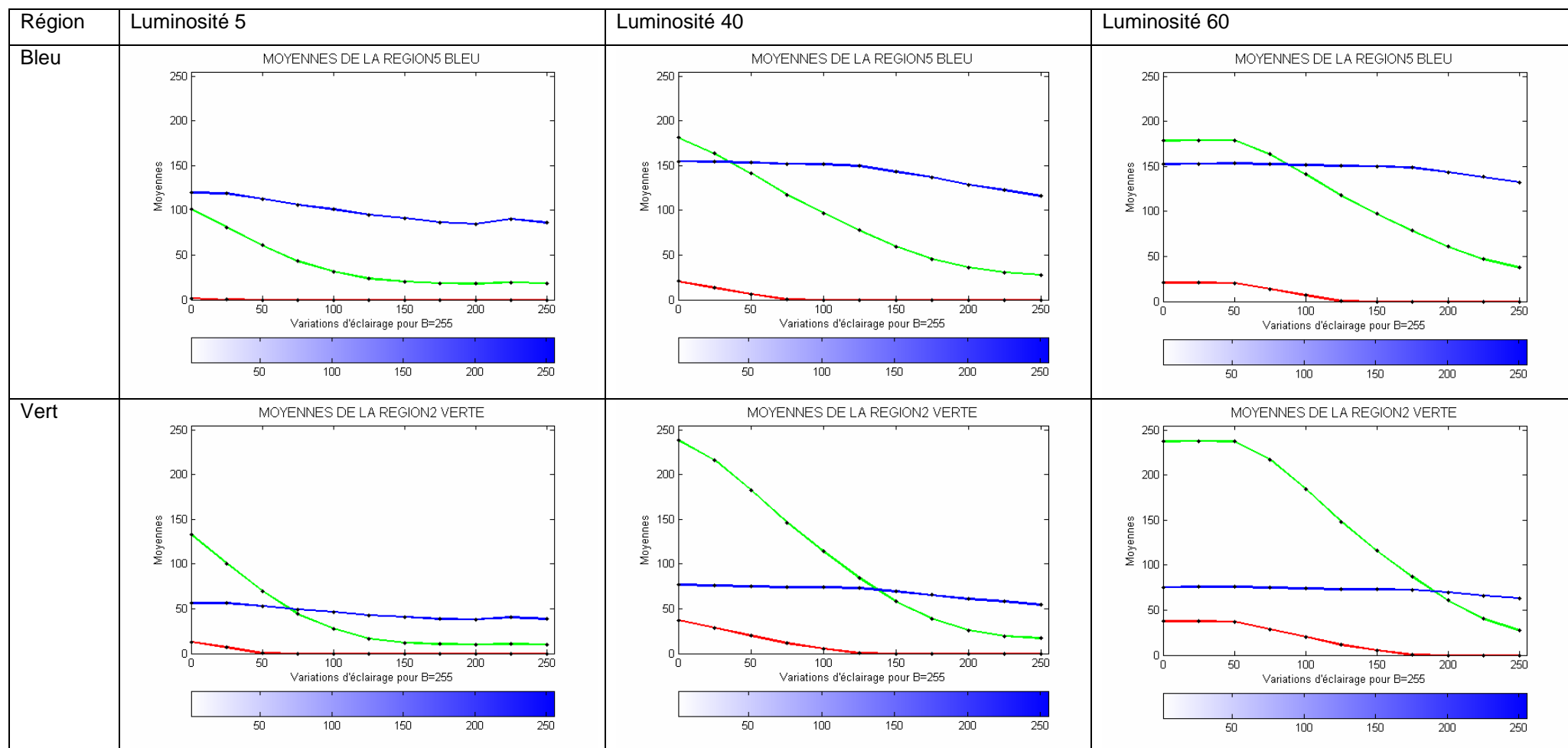


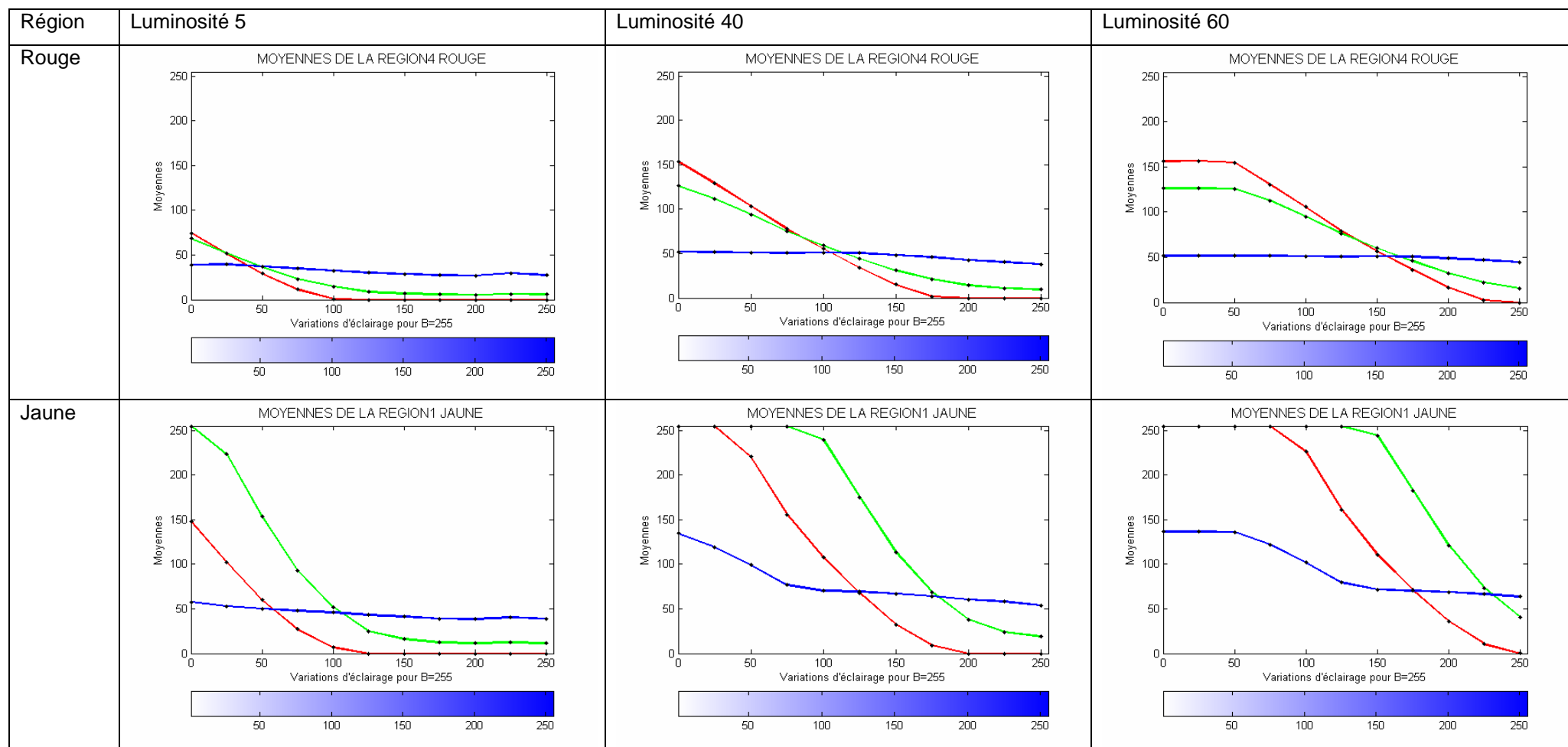


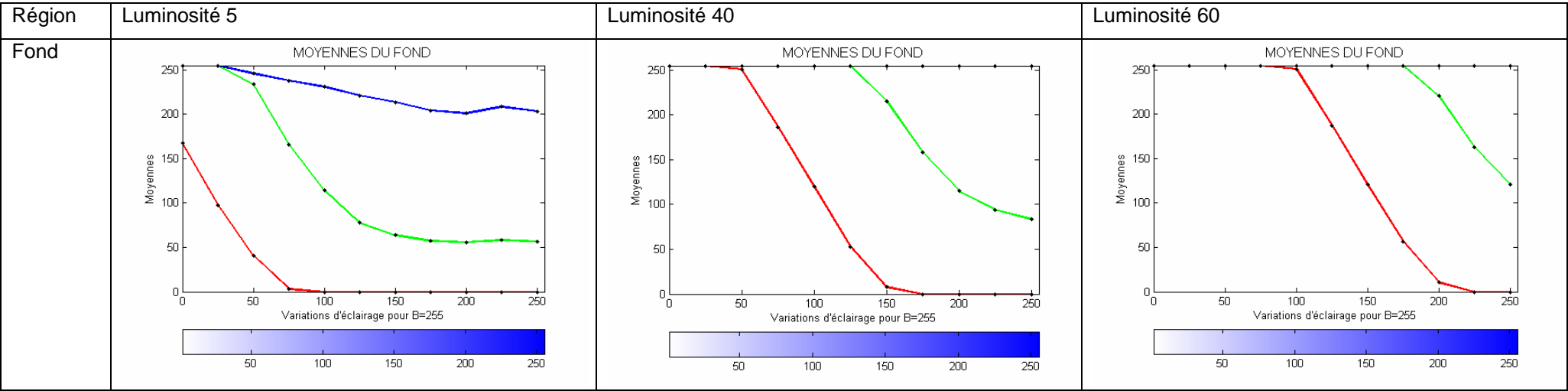


ANNEXE B

Variations des composantes RGB en fonction
de différentes luminosités

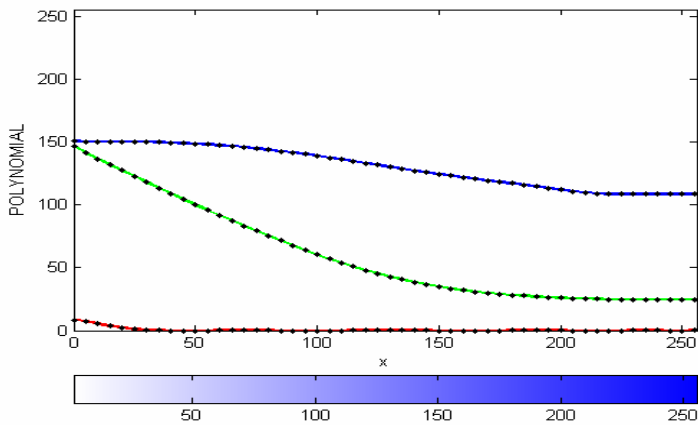
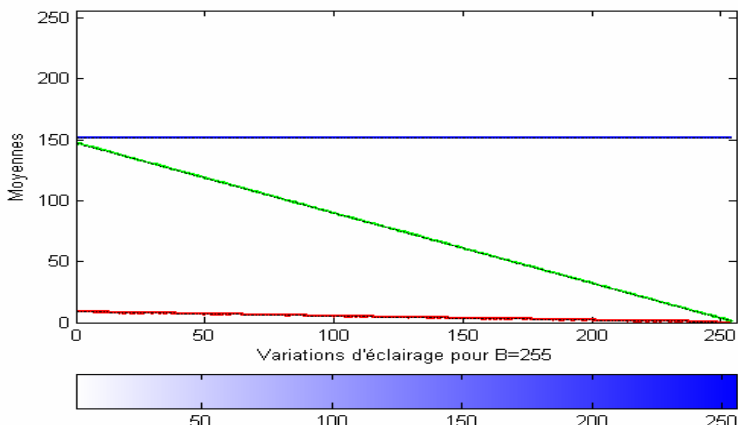
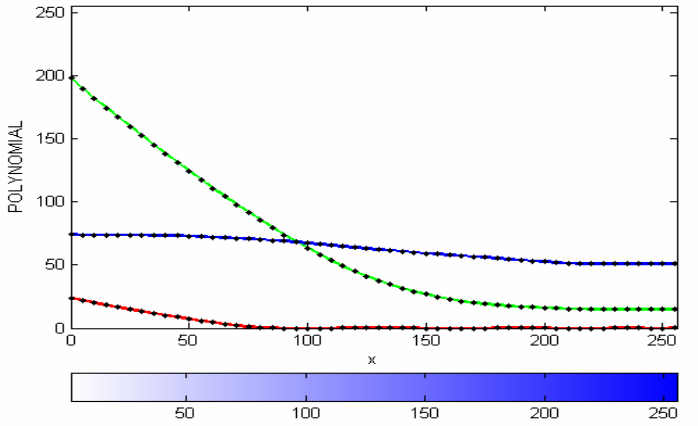
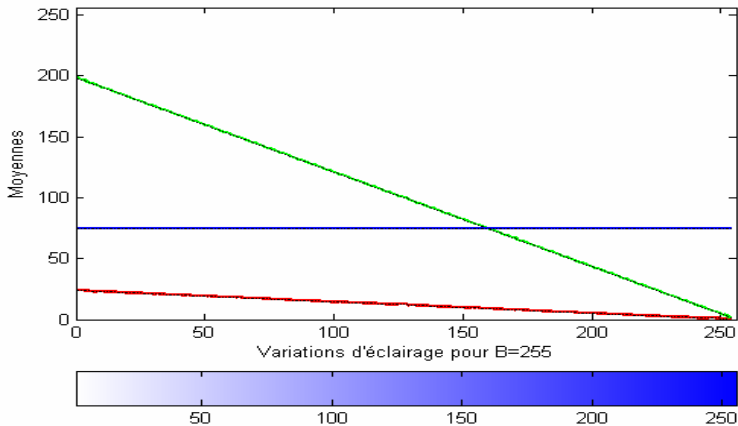


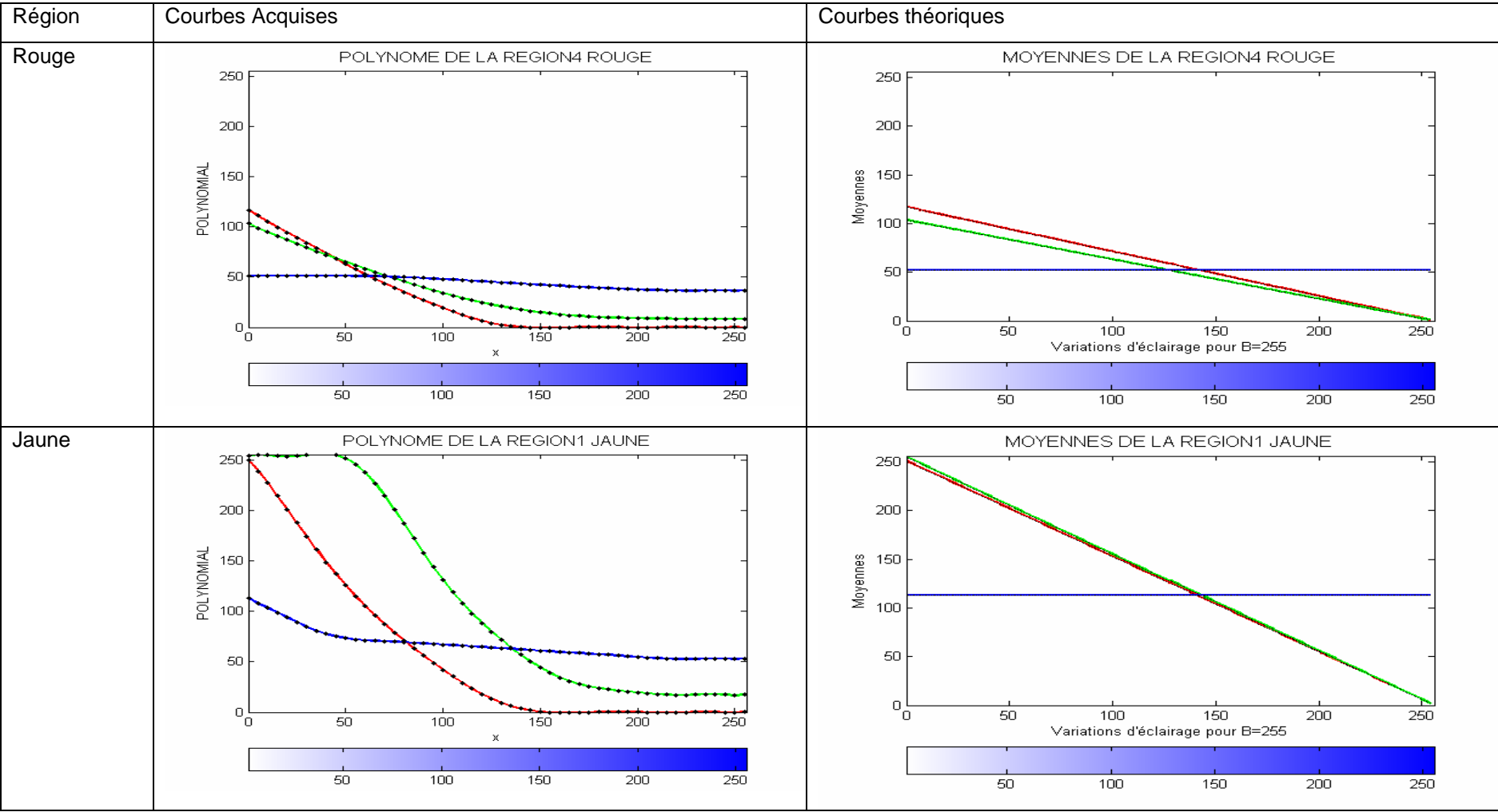


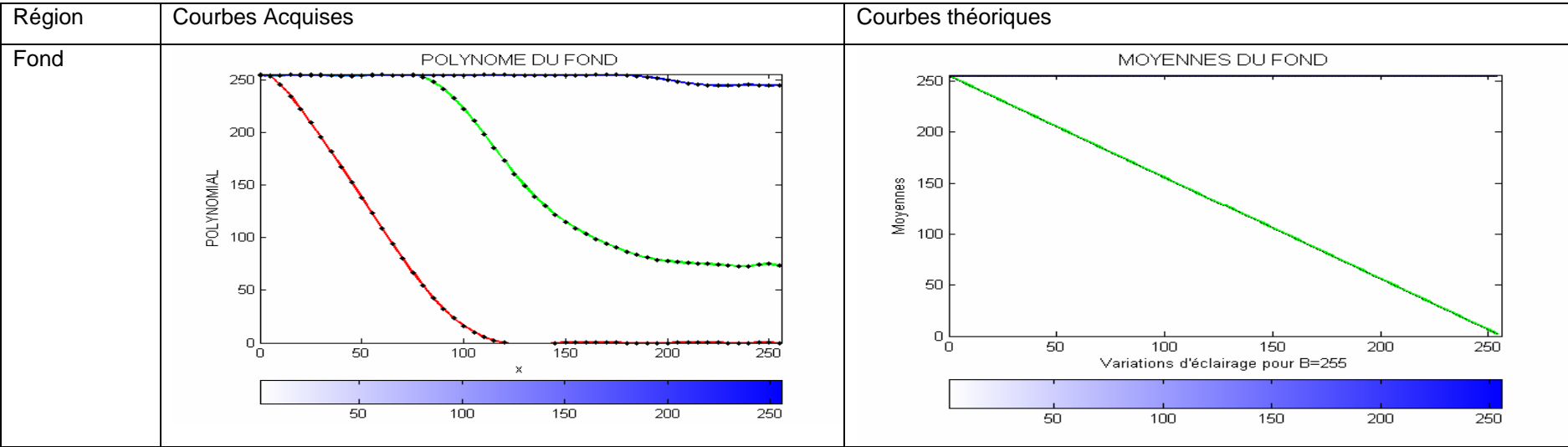


ANNEXE C

Comparaison entre courbes acquises et
courbes théories

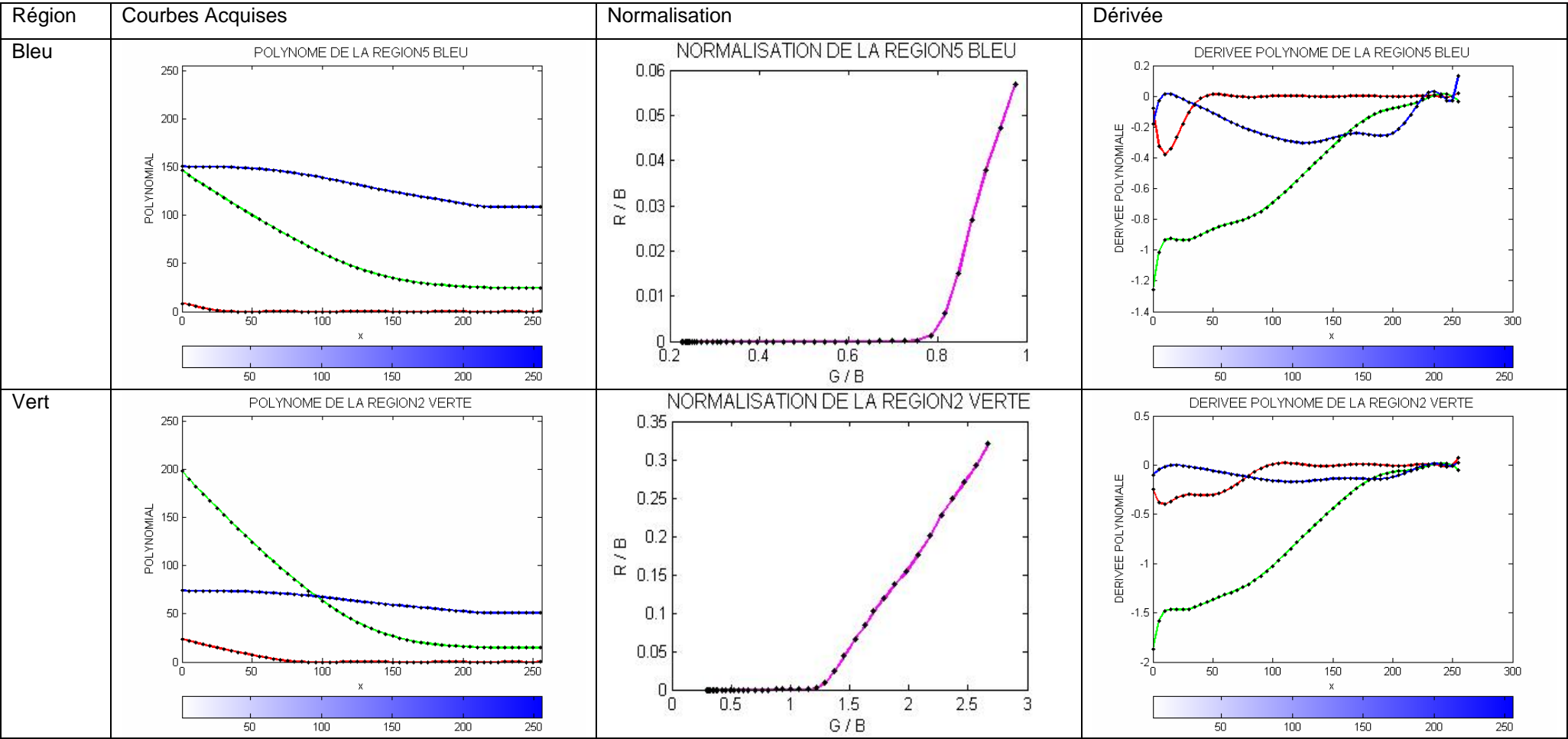
Région	Courbes Acquises	Courbes théoriques
Bleu	<p>POLYNOME DE LA REGION5 BLEU</p> 	<p>MOYENNES DE LA REGION5 BLEU</p> 
Vert	<p>POLYNOME DE LA REGION2 VERTE</p> 	<p>MOYENNES DE LA REGION2 VERTE</p> 

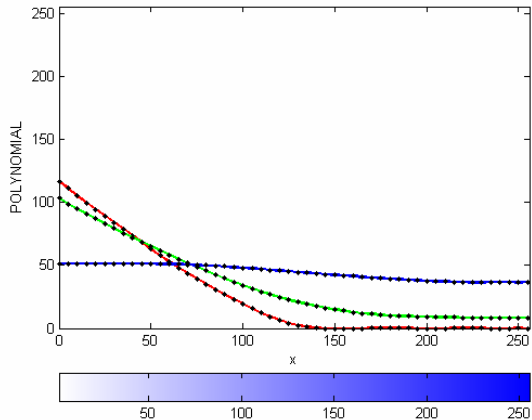
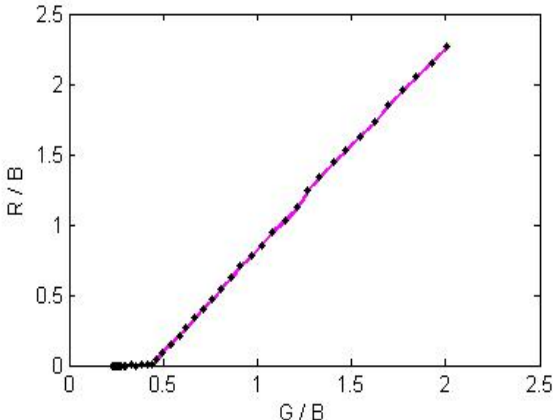
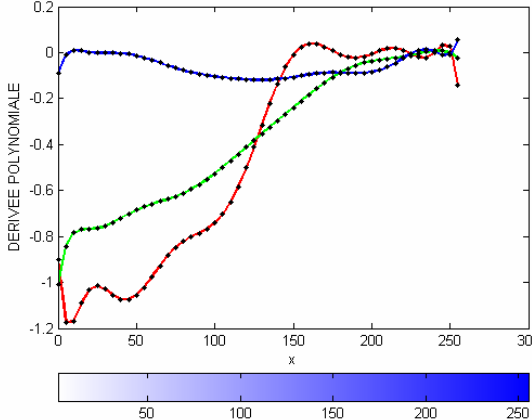
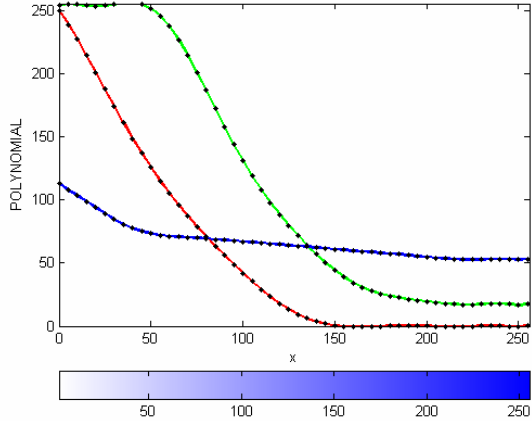
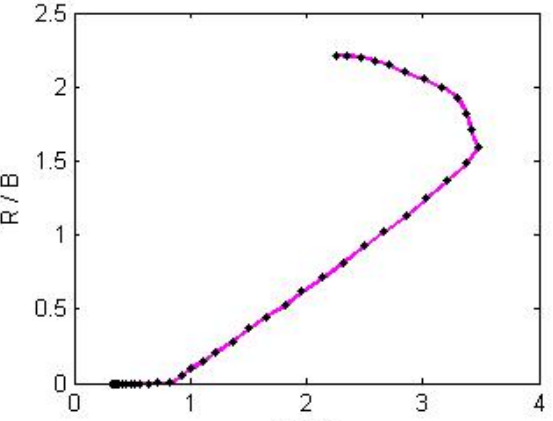
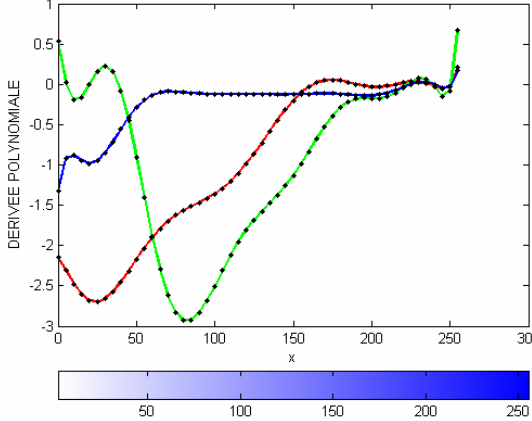




ANNEXE D

Courbes de normalisation et dérivées



Région	Courbes Acquis	Normalisation	Dérivée
Rouge	<p>POLYNOME DE LA REGION4 ROUGE</p> 	<p>NORMALISATION DE LA REGION4 ROUGE</p> 	<p>DERIVEE POLYNOME DE LA REGION4 ROUGE</p> 
Jaune	<p>POLYNOME DE LA REGION1 JAUNE</p> 	<p>NORMALISATION DE LA REGION1 JAUNE</p> 	<p>DERIVEE POLYNOME DE LA REGION1 JAUNE</p> 

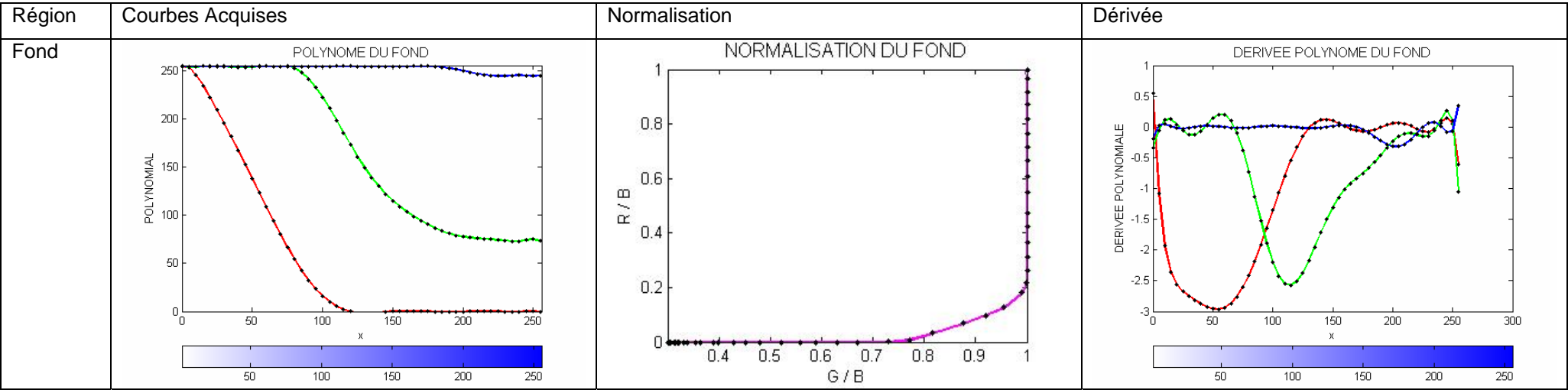
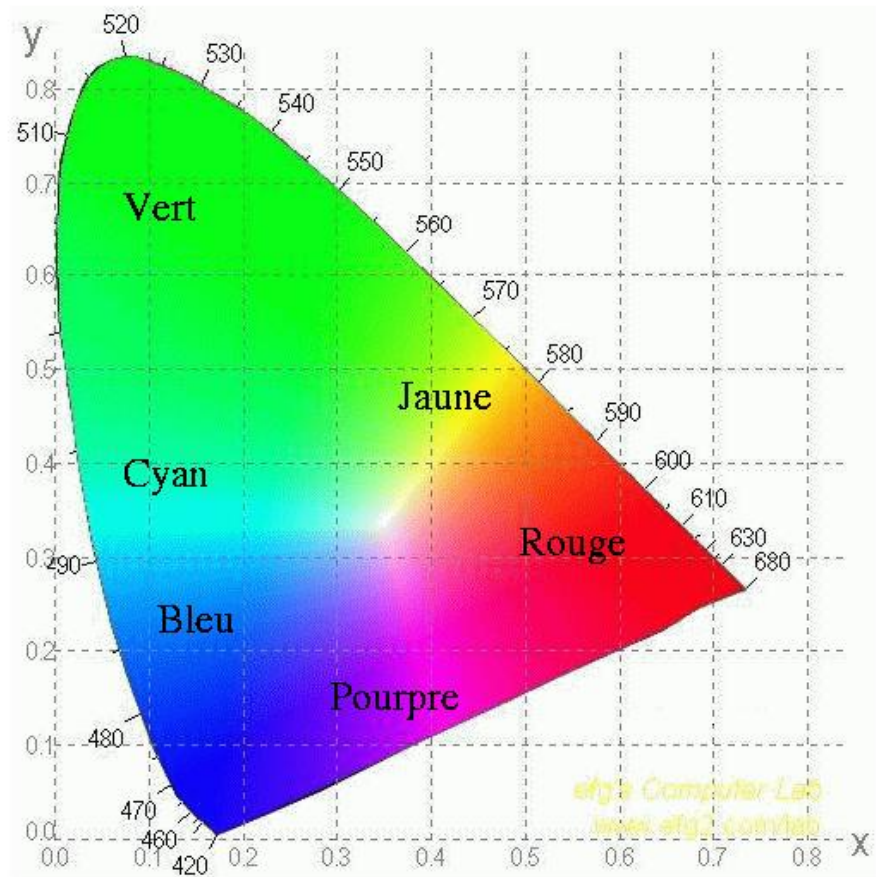


DIAGRAMME DE CHROMACITÉ

Le spectre visible dans le diagramme de chromaticité de la CIE (1931)



PROGRAMMES PRINCIPAUX DE TRAITEMENT SOUS MATLAB

1. Programme de calcul et de tracé

```
%% Init

close all
clear all

IMG = imread
('D:\PROJET_2005\Acquisitions_eclairage\bleu\RGB_REF.bmp');

figure(1)
subplot(3,3,1);
imshow(uint8(IMG));
title('Image de référence')

%% CONVERSION RGB to GRAY

GR = double(rgb2gray(IMG));
subplot(3,3,2);
imshow(uint8(GR));
title('Conversion en gris')

%% Binarisation des niveaux gris

SeuilBin=5;

[nl,nc] = size(GR);
GRBin=zeros(nl,nc);    %Initialise GRBin = Matrice de 0

for l=1:nl
    for c=1:nc
        if GR(l,c)>SeuilBin
            GRBin(l,c)=255;
        end
    end
end

colormap(gray);
subplot (3,3,3);
imshow(uint8(GRBin));
title('Binarisation')

%% SEGMENTATION

% AGGREGATION

RES = zeros(nl,nc); %init
valregion = 0; % init
IncrementRegion = 20;

seuil = 10; % seuil du critère d'adjacence (distance)
```

```

for i=2 : nl-1
    for j=2 : nc-1
        d(1) = abs(GRBin(i,j)-GRBin(i,j-1));
        d(2) = abs(GRBin(i,j)-GRBin(i-1,j-1));
        d(3) = abs(GRBin(i,j)-GRBin(i-1,j));
        d(4) = abs(GRBin(i,j)-GRBin(i-1,j+1));

        [mind, indiced] = min(d);

        if mind < seuil

            switch indiced
                case 1
                    RES(i,j) = RES(i,j-1);
                case 2
                    RES(i,j) = RES(i-1,j-1);
                case 3
                    RES(i,j) = RES(i-1,j);
                case 4
                    RES(i,j) = RES(i-1,j+1);
            end;% switch
        else
            valregion = valregion + IncrementRegion;
            RES(i,j) = valregion;
        end;%if
    end;%for j
end;%for i

nbregions = valregion/IncrementRegion;

colormap(gray);
subplot(3,3,4);
imshow(uint8(RES));
title('Aggrégation')

%% EROSION
% cela permettra de faire une marge d'erreur de région pour
éviter de
% faire une moyenne sur les bords ou pour éviter les bords si
la caméra bouge
% ATTENTION : ON DOIT ADAPTER LA DIMENSION DE L'EROSION EN
FONCTION DE LA
% TAILLE DE L'IMAGE

S = [0 1 0
      1 1 1
      0 1 0];
REROS = imerode(RES,S);
for i =0 : 15 % régler le nombre d'érosion en fonction de la
résolution de l'image

```

```

    REROS = imerode(REROS,S);
end;

figure;
colormap(gray);
subplot(1,1,1);
imshow(uint8(REROS));
title('Erosion')

%% DILATATION

S = [0 1 0
      1 1 1
      0 1 0];

RDIL = imdilate(RES,S);

for i =0 : 5 % régler le nombre de dilatation en fonction de
la résolution de l'image
    RDIL = imdilate(RDIL,S);    %RDIL = Image Dilatée
end;
colormap(gray);
subplot(3,3,6);
imshow(uint8(RDIL));
title('Dilatation')

%% CALCUL

%initialisation

z = 255;
pas = 5; %Pas entre 2 images
m=mod(255,pas); % permet de tronquer le nombre d'image
if m >= (pas/2)
    NbImages = uint8(255/pas);
else
    NbImages = uint8(255/pas)+1;
end;

SommeRegionR=zeros(NbImages,nbreregions);
SommeRegionG=zeros(NbImages,nbreregions);
SommeRegionB=zeros(NbImages,nbreregions);
MoyenneRegionR=zeros(NbImages,nbreregions);
MoyenneRegionG=zeros(NbImages,nbreregions);
MoyenneRegionB=zeros(NbImages,nbreregions);
NbPix=zeros(1,nbreregions);

SommeFondR = zeros(1,NbImages);
SommeFondG = zeros(1,NbImages);

```

```

SommeFondB = zeros(1,NbImages);
NbPixFond = 0;
% boucle

for NumImage = 1 : NbImages
    lien = sprintf
('D:\\PROJET_2005\\Acquisitions_eclairage\\bleu\\RGB_%d_%d_255
.bmp',z,z);
    IMG = imread(lien);
    x(NumImage)=255-z;
    z=z-pas; % Pas pour passer d'une image à l'autre

%% MOYENNEUR

% MOYENNEUR DES REGIONS

[nl,nc] = size(REROS);

for l=1 : nl
    for c=1 : nc
        region = REROS(l,c)/IncrementRegion;
        if region ~= 0 % car region = 0 : c'est le fond de
l'image
            SommeRegionR(NumImage,region) = double(IMG(l,c,1))
+ SommeRegionR(NumImage,region);
            SommeRegionG(NumImage,region) = double(IMG(l,c,2))
+ SommeRegionG(NumImage,region);
            SommeRegionB(NumImage,region) = double(IMG(l,c,3))
+ SommeRegionB(NumImage,region);
            if NumImage == 1 %le calcul du nombre de pixels
par région est pareil pour chaque image
                NbPix(region) = NbPix(region) +1 ;
            end;
        end;
    end;
end;

for region =1 : nbregions
    MoyenneRegionR(NumImage,region) =
SommeRegionR(NumImage,region)/(NbPix(region));
    MoyenneRegionG(NumImage,region) =
SommeRegionG(NumImage,region)/(NbPix(region));
    MoyenneRegionB(NumImage,region) =
SommeRegionB(NumImage,region)/(NbPix(region));
end;

% MOYENNEUR DU FOND DE REFERENCE

for l=1 : nl
    for c=1 : nc
        region = RDIL(l,c)/IncrementRegion;

```

```

        if region == 0
            SommeFondR(NumImage) = double(IMG(1,c,1)) +
SommeFondR(NumImage);
            SommeFondG(NumImage) = double(IMG(1,c,2)) +
SommeFondG(NumImage);
            SommeFondB(NumImage) = double(IMG(1,c,3)) +
SommeFondB(NumImage);
            if NumImage == 1 % un seul calcul suffit
                NbPixFond = NbPixFond +1 ;
            end;
        end;
    end;
end;

MoyenneFondR(NumImage) = SommeFondR(NumImage)/NbPixFond;
MoyenneFondG(NumImage) = SommeFondG(NumImage)/NbPixFond;
MoyenneFondB(NumImage) = SommeFondB(NumImage)/NbPixFond;

end; % fin CALCUL

%% Calcul des courbes

% Creation de la Matrice map (permet d'initialiser la colorbar
du blanc
% vers le rouge), comprise entre 0 et 1 et non entre 0 et 255.
for j=1:255
    map(256-j,1)=j/255;
    map(256-j,2)=j/255;
    map(256-j,3)=1;
end;

%% Forme Polynomiale

% Calcul des courbes sous forme de polynômes

for region=1 :nbregions

for image = 1:NbImages

    y1(image) = MoyenneRegionR(image,region);
    y2(image) = MoyenneRegionG(image,region);
    y3(image) = MoyenneRegionB(image,region);
end;

    PolyR = polyfit(x,y1,12);
    PolyG = polyfit(x,y2,12);
    PolyB = polyfit(x,y3,12);

    fR = polyval (PolyR,x);

```

```

fG = polyval (PolyG,x);
fB = polyval (PolyB,x);

PolyDerR = polyder (PolyR);
PolyDerG = polyder (PolyG);
PolyDerB = polyder (PolyB);

fderR = polyval (PolyDerR,x);
fderG = polyval (PolyDerG,x);
fderB = polyval (PolyDerB,x);

% TRACE DES COURBES
figure ;
plot(x,y1,'-r',x,y2,'-g',x,y3,'-b','LineWidth',2,...
      'Marker','.',...
      'MarkerEdgeColor','k',...
      'MarkerFaceColor','g',...
      'MarkerSize',10)

ylabel('MOYENNES');
xlabel('x');

switch(region)
    case 1
        titre = sprintf('MOYENNE DE LA REGION1 JAUNE');
    case 2
        titre = sprintf('MOYENNE DE LA REGION2 VERTE');
    case 3
        titre = sprintf('MOYENNE DU FOND');
    case 4
        titre = sprintf('MOYENNE DE LA REGION4 ROUGE');
    case 5
        titre = sprintf('MOYENNE DE LA REGION5 BLEU');
end;

title(titre,'FontSize',12);
title(titre,'FontSize',12);
colormap(map);
colorbar('SouthOutside');

% TRACE DES COURBES SOUS FORMES POLYNÔMIALES
figure ;
plot(x,fR,'-r',x,fG,'-g',x,fB,'-b','LineWidth',2,...
      'Marker','.',...
      'MarkerEdgeColor','k',...
      'MarkerFaceColor','g',...
      'MarkerSize',10)

ylabel('POLYNOMIAL');
xlabel('x');

```

```

switch(region)
    case 1
        titre = sprintf('POLYNOME DE LA REGION1 JAUNE');
    case 2
        titre = sprintf('POLYNOME DE LA REGION2 VERTE');
    case 3
        titre = sprintf('POLYNOME DU FOND');
    case 4
        titre = sprintf('POLYNOME DE LA REGION4 ROUGE');
    case 5
        titre = sprintf('POLYNOME DE LA REGION5 BLEU');
end;

title(titre,'FontSize',12);
title(titre,'FontSize',12);
colormap(map);
colorbar('SouthOutside');

%TRACE DES COURBES DES DERIVEES DES POLYNÔMES
figure;
plot(x,fderR,'-r',x,fderG,'-g',x,fderB,'-b','LineWidth',2,...
      'Marker','.',...
      'MarkerEdgeColor','k',...
      'MarkerFaceColor','g',...
      'MarkerSize',10)

ylabel('DERIVEE POLYNOMIALE');
xlabel('x');

switch(region)
    case 1
        titre = sprintf('DERIVEE POLYNOME DE LA REGION1
JAUNE');
    case 2
        titre = sprintf('DERIVEE POLYNOME DE LA REGION2
VERTE');
    case 3
        titre = sprintf('DERIVEE POLYNOME DU FOND');
    case 4
        titre = sprintf('DERIVEE POLYNOME DE LA REGION4
ROUGE');
    case 5
        titre = sprintf('DERIVEE POLYNOME DE LA REGION5
BLEU');
end;

title(titre,'FontSize',12);
title(titre,'FontSize',12);
colormap(map);
colorbar('SouthOutside');

```



```

end;
figure;
colormap(gray);
imshow(uint8(RES));
title('Aggrégation');

```

2. Programme pour l'étude virtuelle

% Ce programme permet de générer des images théoriques à partir d'une seule image

```

close all
clear all

```

```

%IMG = imread('D:\Projet POLETTE BOUVRY\Base1.bmp');
IMG = imread('D:\Projet POLETTE
BOUVRY\Etude_Virtuelle\bleuacq\RGB_255_255_255.bmp');

```

```

figure(1);
subplot(1,1,1);
image(IMG);
title('Image de référence');

```

%% SIMULATION par transformation

```

[nl,nc,coul] = size(IMG);
RES = zeros(nl,nc,coul);

```

```

% boucle
ar = 1;
ag = 1;
ab = 1;

```

```

for z=1:256
    for l=1:nl
        for c=1:nc
            RES(l,c,1) = IMG(l,c,1) * ar;
            RES(l,c,2) = IMG(l,c,2) * ag;
            RES(l,c,3) = IMG(l,c,3) * ab;
        end;
    end;
    nom = sprintf('RGB_%d_%d_255.bmp',ar*256-1,ag*256-1);
    imwrite(uint8(RES),nom);

    ar=ar-1/256;
    ag=ag-1/256;
end;

```