



Traitement numérique des images

Notes de cours (C++)

Steven Pigeon

Professeur

Département de mathématiques,
informatique et génie
UQAR

Traitement numérique des images
Notes de cours (C++)

Traitement numérique des images

Notes de cours (C++)

par

Steven Pigeon, professeur

Département de mathématiques,
informatique et génie
UQAR

Steven Pigeon
Département de mathématiques, informatique, et génie
Université du Québec à Rimouski
Rimouski, Québec
G5L 3A1
steven_pigeon@uqar.ca

Le document a été typographié par l'auteur avec la police libre URW-Garamond par Mathdesign et le système L^AT_EX. Les illustrations, sauf indications contraires, sont de la main de l'auteur. Les autres illustrations sont libres de droits ou utilisées selon les dispositions de l'article 29 de la loi sur le droit d'auteur. La photo « *Inoxtubes* » de la couverture est © Steven Pigeon.

V 0.9 Juillet 2020

005.7xxx
QA76.xx 2018
ISBN 978-0-0000-0000-2

© 2020 Steven Pigeon

Tous droits de traduction totale ou partielle réservés pour tous les pays. La reproduction d'un extrait quelconque de ce livre, sauf selon les dispositions prévue par la loi, par quelque procédé que ce soit, tant électronique que mécanique, en particulier par photocopie, est interdite sans l'autorisation écrite de l'auteur.

Dernièrement, un ami [...] me disait ceci : « Le savoir, si tu te donnes complètement à lui, il te donne un peu. Si tu lui donnes un peu, il ne te donne rien ».

Ali Benmakhlof
À quoi sert le savoir ?

Table des matières

Table des matières	i
Table des figures	v
Liste des tableaux	ix
Table des notations	xi
Conventions typographiques	xiii
1 Structures des images	1
1.1 Les pixels	1
1.1.1 Systèmes de couleurs	1
1.1.1.1 Monochromie et tons de gris	1
1.1.1.2 Dichromie	2
1.1.1.3 Trichromie	2
1.1.1.4 Quadrichromie	3
1.1.1.5 Multichromie	3
1.1.1.6 Couleurs hyperspectrales	3
1.1.1.7 Bande dynamique étendue et non-linéarité	6
1.1.2 Encodage de la couleur	8
1.1.2.1 <i>Direct Colors et Bit Depth</i>	8
1.1.2.2 Palettes	12
1.2 Organisation des pixels dans l'image	13
1.2.1 Résolutions	13
1.2.2 Organisation en mémoire	15
1.2.3 Sous-échantillonnage	18
1.3 Exemples de formats d'image	19
1.4 Remarques bibliographiques	23
2 Saisie des images	25
2.1 Introduction	25
2.2 Senseurs et couleurs	25
2.2.1 Technologies de senseurs	25
2.2.2 Prismes et filtres	27
2.2.3 Systèmes à un seul senseur	28
2.2.3.1 Bayer et ses amis	29
2.3 Remarques bibliographiques	31
3 Espaces de couleurs	33
3.1 Introduction	33
3.1.1 Des couleurs primaires	33
3.1.2 Familles d'espaces de couleurs	36
3.2 L'espace de couleurs CIÉ 1931	37

3.2.1	De l'origine du triangle des couleurs	38
3.2.2	L'espace des couleurs CIÉ comme un espace vectoriel	40
3.2.3	Contraintes imposées sur l'espace CIÉ 1931	41
3.2.4	CIÉ 1931 et les autres espaces de couleurs.	41
3.2.5	Conversions entre CIÉ $X Y Z$ 1931 et RGB	42
3.3	Les espaces de couleurs linéaires	42
3.3.1	Kodak 1	43
3.3.2	Kodak <i>Lossless</i>	45
3.3.3	Kodak <i>YCC</i>	46
3.3.4	Xerox <i>YES</i>	48
3.3.5	Espaces Ohta	48
3.3.6	$YC_o C_g$	50
3.3.7	JPEG 2000 : $Y V U - R$	53
3.3.8	Série <i>S</i>	54
3.3.9	Les espaces de couleurs de la télévision	61
3.3.9.1	<i>Transmission primaries</i>	61
3.3.9.2	<i>YUV</i> et <i>YIQ</i>	63
3.3.9.3	Télévision numérique : BT.601, BT.709 et sRGB	66
3.3.9.4	$YC_b C_r$	68
3.3.9.5	$YP_b P_r$	69
3.3.9.6	$YD_b D_r$	70
3.3.9.7	$Y'C'_b C'_r$ alias $YC_b C_r$, HDTV	71
3.3.9.8	BT.2020 UHDTV	71
3.3.9.9	BT.2100 HDRTV	74
3.3.9.10	Un portrait de famille	76
3.3.10	<i>CMY</i> et <i>CMYK</i>	77
3.4	Les espaces de couleurs non linéaires	82
3.4.1	<i>HSV</i> et <i>HSL</i>	83
3.4.1.1	<i>HSV</i>	83
3.4.1.2	<i>HSL</i>	88
3.4.2	CIÉ $L^* a^* b^*$, $L^* u^* v^*$ et CIÉDE2000	91
3.4.2.1	$L^* a^* b^*$	92
3.4.2.2	$L^* u^* v^*$	94
3.4.2.3	CIÉDE2000	95
3.4.3	Autres espaces de couleurs	95
3.4.3.1	Munsell	95
3.4.3.2	Ostwald	96
3.4.3.3	NCS	98
3.5	Remarques bibliographiques	99
4	Discrétisation des couleurs	105
4.1	Introduction	105
4.2	Métriques et fidélité	106
4.2.1	Métriques	106
4.2.2	Estimation de la fidélité	108
4.3	Méthodes directes	109
4.3.1	16/15/12/9/8/6/3 bits par pixel	110
4.3.2	Méthode de Paeth	112
4.3.3	666, la palette du diable	114
4.4	Méthodes adaptatives	116
4.4.1	Popularité	117
4.4.2	Choix Aléatoire	118
4.4.3	<i>Median-Cut</i> et <i>Min-Variance</i>	118
4.4.4	Octree	121
4.4.5	<i>K-means</i>	122
4.4.6	Agglomération hiérarchique	124
4.4.7	Programmation dynamique et analyse en composantes principales	127

4.4.8	Comparaison des algorithmes adaptatifs	128
4.5	Remarques bibliographiques	131
5	Tramage et diffusion d'erreur	135
5.1	Introduction	135
5.2	Tramage régulier et quasi-régulier	136
5.2.1	Tramage régulier	136
5.2.2	Tramage par ordinateur : aspects historiques	138
5.2.3	Tramage ordonné	140
5.2.3.1	Tramage de Julesz	142
5.2.3.2	Tramage de Bayer	142
5.2.3.3	Tramage d'Allebach et Liu	144
5.2.3.4	Tramage de Bryngdahl	145
5.2.3.5	Tramage de Hawley	145
5.2.3.6	Tramage de Schumacher	148
5.2.3.7	Tramage d'Adler <i>et al.</i>	149
5.3	Diffusion d'erreur	150
5.3.1	Boucle de rétroaction pour la correction d'erreur	152
5.3.2	Matrices de diffusion	153
5.3.2.1	Méthode de Floyd-Steinberg	154
5.3.2.2	Méthode d'Atkinson	155
5.3.2.3	Méthode de Jarvis, Judice et Ninke	156
5.3.2.4	Méthodes de Stucki	156
5.3.2.5	Méthode de Burkes	157
5.3.2.6	Méthodes de Sierra	158
5.3.2.7	Créer une méthode de diffusion d'erreur	159
5.3.2.8	Diffusion d'erreur et autres géométries de pixels	161
5.3.2.9	Autres Méthodes	162
5.4	Remarques bibliographiques	166
Bibliographie		169
Index		189

TABLE DES MATIÈRES

Table des figures

1.1.1	Dithering et halftoning	2
1.1.2	Trois couleurs	2
1.1.3	Tramage	4
1.1.4	Les piliers de la création, Hubble	6
1.1.5	Les piliers de la création, JWST	6
1.1.6	Correction gamma.	7
1.1.7	Corrections plus souples	7
1.1.8	Contraste simultané.	8
1.1.9	Faux contours	10
1.1.10	La réponse psychovisuelle, d'après [149].	11
1.1.11	Palettes CGA	12
1.1.12	Le roi Toutânkhamon par Avril Harrison (1986), version Amiga. Source : Retroshowcase [29].	13
1.2.1	L'organisation en plage contiguë.	15
1.2.2	Organisation en plans de bits	16
1.2.3	L'organisation en plans avec transparence.	17
1.2.4	Organisation Entrelacée.	17
1.2.5	Sous-échantillonnage des couleurs	19
1.2.6	YC_bC_r et les couleurs reconstruites	20
1.2.7	Des images sous-échantillonées en 4:2:0	20
1.4.1	Caractères semigraphiques	23
2.2.1	Rolling shutter	26
2.2.2	Prisme de type Philips	27
2.2.3	Prismes chroïques	28
2.2.4	Prisme dichroïque	28
2.2.5	Prisme trichroïque de caméra	29
2.2.6	Senseur « pleine couleur »	29
2.2.7	Senseur et microlentilles	29
2.2.8	Motifs de Bayer	30
2.2.9	Image saisie avec un motif de Bayer	30
3.1.1	Sensibilité de l'œil aux longueurs d'ondes	34
3.1.2	Triangle de couleurs primaires	35
3.1.3	Les cônes et les bâtonnets combinent leurs signaux pour donner la luminance, les différences rouge-vert et jaune-bleu [35].	37
3.2.1	Les fonctions colorimétriques CIÉ	38
3.2.2	Triangle de couleurs primaires CIÉ	39
3.2.3	Triangle $X Y Z$ de la CIÉ	40
3.2.4	Pyramide $X Y Z$ de la CIÉ	40
3.3.1	L'espace de couleurs RGB.	43
3.3.2	L'espace de couleurs Kodak-1.	44
3.3.3	L'espace de couleurs Kodak <i>Lossless</i>	45
3.3.4	L'espace de couleurs Kodak <i>Lossless</i> , seconde variante.	46

3.3.5	L'espace de couleurs Kodak <i>YCC</i>	47
3.3.6	L'espace de couleurs Xerox <i>YES</i>	49
3.3.7	Les axes principaux d'un patatoïde.	49
3.3.8	Les espaces de couleurs Ohta.	50
3.3.9	L'espace de couleurs YC_gC_g	51
3.3.10	L'espace de couleurs $YVU-R$	54
3.3.11	La construction de l'espace S_1	56
3.3.12	Les espaces de couleurs S_1 et S_2	56
3.3.13	Le cube <i>RGB</i> « compressé » par les poids ω_r , ω_g et ω_b	58
3.3.14	L'espace de couleurs S_3	59
3.3.15	L'espace de couleurs S'_1	60
3.3.16	Une image télévision couleur décodée par un poste noir et blanc.	62
3.3.17	Les espaces de couleurs <i>YUV</i> et <i>YIQ</i>	64
3.3.18	L'espace de couleurs YC_bC_r	68
3.3.19	Les espaces de couleurs YC_bC_r et $Y'C'_bC'_r$ comparés.	72
3.3.20	L'espace de couleurs $Y'C'_{BC}C'_{RC}$	73
3.3.21	Une image en grande gamme dynamique (HDR).	74
3.3.22	L'espace de couleurs IC_TC_P transformant <i>LMS</i> transformant <i>RGB</i> , sans les corrections non linéaires.	76
3.3.23	La famille BT.	77
3.3.24	L'espace de couleurs <i>CMYK</i> , attentes et résultats!	78
3.3.25	L'espace de couleurs <i>CMY</i>	79
3.3.26	L'espace de couleurs <i>CMYK</i> (les <i>K</i> sont montrés par translations).	81
3.4.1	Le cercle chromatique de l'espace <i>HSV</i>	84
3.4.2	Géométries de l'espace <i>HSV</i>	84
3.4.3	<i>HSV</i> et une projection du cube <i>RGB</i>	85
3.4.4	La projection sur l'hexagone <i>HSV</i>	85
3.4.5	Sur l'intervalle $[-\frac{1}{2}, \frac{1}{2}]$, $\tan^{-1}x \approx x$	86
3.4.6	Les régions de couleurs dans l'hexagone <i>HSV</i>	87
3.4.7	Une implémentation <i>Mathematica</i> des conversions de <i>RGB</i> à <i>HSV</i>	88
3.4.8	Les variantes de l'espace de couleurs <i>HSV</i>	88
3.4.9	Les variantes de l'espace de couleurs <i>HSL</i>	89
3.4.10	<i>HSL</i> , dans le brevet de Bergstedt [60].	89
3.4.11	Une implémentation <i>Mathematica</i> des conversions de <i>RGB</i> à <i>HSL</i>	91
3.4.12	Les ellipses de MacAdam (dessinées 10 fois plus grandes qu'en réel), d'après [245]. Source : Wikipedia.	92
3.4.13	La portion <i>RGB</i> de l'espace de couleurs $L^*a^*b^*$	93
3.4.14	La portion <i>RGB</i> de l'espace $L^*u^*v^*$	95
3.4.15	Le cercle chromatique de l'espace de couleurs de Munsell. Source : Wikipedia.	96
3.4.16	L'espace de couleurs de Munsell (couleurs réalisables en <i>RGB</i>).	96
3.4.17	L'arbre des couleurs de Munsell, tiré de [93] (domaine public).	97
3.4.18	Le cercle chromatique d'Ostwald	97
3.4.19	L'espace de couleurs Ostwald.	98
3.4.20	Les coordonnées Ostwald	98
3.4.21	L'espace de couleur NCS.	99
3.5.1	L'espace d'Ostwald et sa correction logarithmique.	103
4.3.1	L'image de référence « coucher de soleil ». Image : Steven Pigeon	110
4.3.2	Les effets d'une discréétisation directe, avec un nombre variable de bits par pixel.	111
4.3.3	Le cube de Paeth.	112
4.3.4	La méthode de Paeth.	112
4.3.5	Les palettes <i>web safe</i> , ou presque.	114
4.3.6	Les palettes <i>web safe</i> , ou presque, appliquées aux images.	114
4.4.1	L'algorithme popularité.	117
4.4.2	Palettes aléatoires.	118
4.4.3	<i>Median-Cut</i>	119
4.4.4	Plan de coupe pour une boîte dans <i>Median-Cut</i> . Le plan de coupe est perpendiculaire à l'axe le plus long.	119
4.4.5	<i>Min-variance</i>	120
4.4.6	Division de l'espace en <i>octree</i>	121

4.4.7	Discrétisation par <i>octree</i>	122
4.4.8	Discrétisation avec <i>K-Means</i>	123
4.4.9	Une implémentation possible de <i>K-Means</i>	124
4.4.10	Division de l'espace par <i>K-means</i> (source : Wikipedia).	125
4.4.11	Les régions produites par une agglomération hiérarchique (« concept d'artiste »).	125
4.4.12	Discrétisation avec agglomération hiérarchique.	125
4.4.13	La distribution des couleurs dans le cube RGB pour notre image de référence.	127
4.4.14	Le graphe pour l'algorithme de Wu	127
4.4.15	Les effets des différents algorithmes de discrétisation adaptative, 16 couleurs.	129
4.4.16	Les effets des différents algorithmes de discrétisation adaptative, 256 couleurs.	130
4.5.1	L'espace de couleur YJK de la puce Yamaha V9938.	131
5.2.1	Tramage régulier et densité variable.	136
5.2.2	Les angles typiques utilisés dans le tramage régulier.	137
5.2.3	Un exemple de tramage en demi-teinte, avec détail.	137
5.2.4	Tramage pour le tissage.	138
5.2.5	Tramage ordonné inspiré de Beaumont.	138
5.2.6	Matrice de seuillage inspirée du tramage présenté dans Beaumont.	139
5.2.7	Tramage ordonné inspiré de Knowlton.	139
5.2.8	Motifs du tramage BEFLIX.	139
5.2.9	Matrice de seuillage inspirée du tramage BEFLIX.	140
5.2.10	Les huit couleurs de base utilisées pour le tramage et la diffusion d'erreur.	141
5.2.11	Tramage ordonné de Julesz.	141
5.2.12	Matrice de seuillage de Julesz, cité dans [212].	141
5.2.13	Tramage aléatoire de Bayer.	142
5.2.14	Matrice de seuillage en spirale de Bayer.	143
5.2.15	Matrice de seuillage « optimale » de Bayer.	143
5.2.16	Tramage ordonné de Bayer.	144
5.2.17	Matrice de seuillage en losanges d'Allebach et Liu.	144
5.2.18	Matrice de seuillage en grille d'Allebach et Liu.	145
5.2.19	Tramage ordonné de Allebach et Liu.	146
5.2.20	Matrices de Bryngdahl.	146
5.2.21	Tramage ordonné de Bryngdahl.	147
5.2.22	Tramage ordonné de Hawley.	147
5.2.23	Matrice de seuillage en quinconce de Schumacher.	148
5.2.24	Tramage ordonné de Schumacher.	148
5.2.25	Motif 3×3 d'Adler <i>et al.</i>	148
5.2.26	Matrices d'Adler <i>et al.</i>	149
5.2.27	Tramage ordonné de Adler <i>et al.</i>	149
5.3.1	Tramage aléatoire.	150
5.3.2	Bruit pseudo-aléatoire de Roberts pour quatre niveaux.	151
5.3.3	La boucle de rétroaction pour la correction d'erreur en une dimension.	152
5.3.4	La diffusion d'erreur doit se faire sur les pixels voisins en 2D.	153
5.3.5	La matrice de diffusion d'erreur de Floyd et Steinberg (diviseur : 16).	154
5.3.6	Diffusion d'erreur de Floyd et Steinberg.	154
5.3.7	La matrice de diffusion d'erreur de Crocker <i>et al.</i> (diviseur : 8).	155
5.3.8	Diffusion d'erreur de Crocker <i>et al.</i>	155
5.3.9	La matrice de diffusion d'erreur d'Atkinson (diviseur : 8).	155
5.3.10	Diffusion d'erreur d'Atkinson.	156
5.3.11	La matrice de diffusion d'erreur de Jarvis, Judice et Ninke (diviseur : 46).	156
5.3.12	Diffusion d'erreur de Jarvis, Judice et Ninke.	157
5.3.13	Matrices de diffusion d'erreur de Stucki.	157
5.3.14	Diffusion d'erreur de Stucki en noir et blanc.	157
5.3.15	Diffusion d'erreur de Stucki en 8 couleurs.	158
5.3.16	La matrice de diffusion d'erreur de Burkes (diviseur : 32).	158
5.3.17	Diffusion d'erreur de Burkes.	158
5.3.18	Matrices de diffusion d'erreur de Sierra.	159

TABLE DES FIGURES

5.3.19	Diffusion d'erreur de Sierra en noir et blanc	159
5.3.20	Diffusion d'erreur de Sierra en 8 couleurs.	159
5.3.21	La matrice de diffusion d'erreur de Pigeon [300].	160
5.3.22	Les matrices de diffusion d'erreur de Pigeon, version 3.	160
5.3.23	Les matrices de diffusion d'erreur de Pigeon, versions 4 et 5.	161
5.3.24	Diffusion d'erreur de Pigeon, variantes 1 et 2.	162
5.3.25	Diffusion d'erreur de Pigeon, variantes 3.	163
5.3.26	Diffusion d'erreur de Pigeon, variantes 4 et 5.	164
5.3.27	Diffusion d'erreurs et géométries variées.	165
5.3.28	Matrice de seuillage de Knuth	165
5.4.1	La momie d'Héraclide, 120-140 ap. J.-C.(détail, image de domaine public).	166

Liste des tableaux

1.1.1	Longueurs d'ondes et phénomènes physiques	5
1.1.2	Quelques exemples d'encodage de couleurs	9
1.2.1	Quelques résolutions communes	14
1.2.2	Quelques résolutions communes pour la télévision	14
1.2.3	CIF et SIF	15
4.3.1	Différentes résolutions de couleurs	110
4.3.2	Couleurs de la méthode de Paeth.	113
4.4.1	PSNR en dB des méthodes directes.	128
4.4.2	PSNR en dB des méthodes adaptatives.	128

LISTE DES TABLEAUX

Table des notations

\mathbb{N}	Les nombres naturels, $\{1, 2, 3, \dots\}$.
\mathbb{N}_0	Les naturels incluant le zéro, $\{0, 1, 2, 3, \dots\}$.
\mathbb{Z}	Les entiers, $\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$.
\mathbb{Z}^*	Les entiers non négatifs, équivalent à \mathbb{N}_0 .
\mathbb{Z}^+	Les entiers positifs, équivalent à \mathbb{N} .
\mathbb{Q}, \mathbb{Q}'	Les rationnels, les irrationnels.
\mathbb{R}	Les réels.
\mathbb{R}^*	Les réels non négatifs.
\mathbb{R}^+	Les réels (strictement) positifs.
$ x $	Valeur absolue de x .
$\ x\ $	Norme de x (on suppose euclidienne).
$\ x\ _p$	p -Norme de x .
$\lfloor x \rfloor$	Le <i>plancher</i> de x : le plus grand entier plus petit ou égal à x .
$\lceil x \rceil$	Le <i>plafond</i> de x : le plus petit entier plus grand ou égal à x .
$\{x_i\}_{i=a}^b$	La séquence $\{x_a, x_{a+1}, \dots, x_{b-1}, x_b\}$.
\perp	Séquence vide, mot vide.
$ A $	Cardinalité de l'ensemble A , longueur de l'objet A .
$A \times B$	Si A et B sont des ensembles, produit cartésien.
A^n	Si A est un ensemble, la puissance cartésienne.
A^*	$\{\perp\} \cup A \cup A^2 \cup A^3 \cup \dots$
$A \cup B$	Union de A et B .
$A \cap B$	Intersection de A et B .
$A \subseteq B, A \subset B$	Inclusion, inclusion stricte.
$A^c, \complement_U A$	Complément de A , complément de A dans U .
$P(S)$	L'ensemble de toutes les parties d'un ensemble.
$\ln x$	Logarithme naturel (base e) de x .
$\log_{10} x$	Logarithme à base 10 de x .

$\lg x$	Logarithme à base 2 de x .
$\log x$	Logarithme à base indifférente.
$\exists, \exists!, \nexists$	Quantificateurs existentiels : il existe, il n'existe qu'un seul, il n'existe aucun.
\forall	Quantificateur universel : pour tous.
$a \ll b$	a est beaucoup plus petit que b .
$a \gg b$	a est beaucoup plus grand que b .
$a \sim b$	a est de la forme b , a est semblable à b .
$\binom{n}{m}$	Nombre de façons de choisir m parmi n , $\binom{n}{m} = \frac{n!}{m!(n-m)!}$.
$\binom{n}{n_1 n_2 \dots n_k}$	Nombre de façons de choisir n_1, n_2, \dots, n_k symboles indistinguables parmi n .
ssi	Si, et seulement si.
γ	La constante d'Euler-Mascheroni, $\gamma = 0.577215664901532\dots$
e	La constante d'Euler, $e = 2.718281828459045\dots$
π	La constante d'Archimède, $\pi = 3.141592653589793\dots$
\hat{x}	Un estimateur (prédiction) pour x .
\tilde{x}	Une discréétisation de x .

Conventions typographiques

Le texte normal est typographié avec la police libre URW-Garamond par Mathdesign. Les programmes et les éléments de programmation sont typographiés avec la police ASCII de Syropoulos et Nickalls, qui rappelle une police de terminal. Les éléments ordinaires sont typographiés ainsi alors que les mots-clefs du langage sont typographiés en gras, comme pour **while**, **if**, **for**, etc.

Les librairies utilisées par les programmes sont tirés de la librairie standard. Par convention, les librairies introduites par `#include <librairie>` font partie du système et sont en principe disponibles en même temps que le compilateur C++. Les librairies introduites par des guillemets, comme par exemple `#include "librairie"` sont les librairies fournies par l'utilisateur, c'est-à-dire par le programme plutôt que par la librairie standard.

Les références, notées par exemple [299], vous amènent à la fin de l'ouvrage où elles sont présentées en ordre alphabétique d'auteurs, et non en ordre d'apparition.

Les démonstrations se trouvent presque toujours dans un encadré intitulé comme tel. Pour une première lecture ou pour un cours moins avancé, ces démonstrations peuvent être omises, mais nous invitons tout de même le lecteur à s'y attarder. Comme pour les démonstrations, les exemples sont encadrées.

1

Structure des images

Sommaire. Dans ce chapitre, nous nous intéresserons à la représentation interne des images. Bien entendu, nous avons tous une idée assez intuitive de ce qu'est une image, mais savons-nous bien comment elles sont représentées dans l'ordinateur? Nous présenterons d'abord les pixels, avec leurs précision et résolutions différentes, puis nous nous intéresserons à représenter les pixels dans la mémoire de l'ordinateur. Nous verrons les représentations usuelles, c'est-à-dire les organisations contiguës, entrelacées et en plants de bits. Nous discuterons aussi du sous-échantillonnage qui exploite les propriétés du système visuel humain pour détruire intelligemment de l'information et ainsi sauver de la mémoire.

1.1 Les pixels

Commençons par nous intéresser aux pixels et aux systèmes de couleurs qui sont communément utilisés, autant à l'imprimé qu'à l'écran.

1.1.1 Systèmes de couleurs

1.1.1.1 Monochromie et tons de gris

Les images monochromes (étymologiquement « une couleur ») sont en fait toujours composées de deux couleurs, dont une fait office d'arrière-plan et l'autre d'avant-plan. Typiquement, ces deux couleurs sont le noir et le blanc, mais il peut s'agir d'autres couleurs — comme deux tons de vert épinard ou d'« ambre »! Comme chaque pixel ne peut prendre que deux couleurs, les tons dégradés nécessaires au rendu agréable d'images naturelles sont simulés par diffusion d'erreur (*dithering*) ou par tramage (*halftone*). Ces techniques pour simuler les tons de gris sont montrées à la fig. 1.1.1.

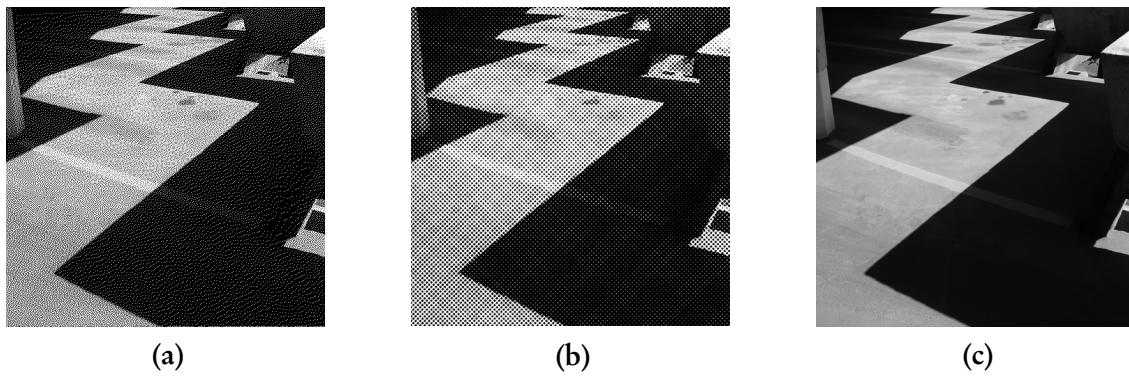


FIGURE 1.1.1 — Diffusion d'erreur (a) et tramage (b). En (c), l'image en tons de gris.

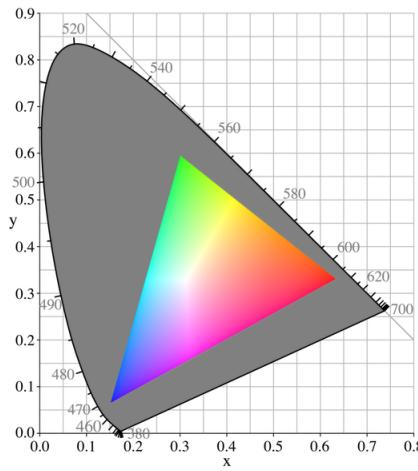


FIGURE 1.1.2 — La plage de couleurs réalisable à partir de trois couleurs primaires. Source : Wikipedia.

Lorsque l'image est en tons de gris (*grayscale*) nous trouvons n (souvent $n = 2^k$) tons également répartis du noir au blanc. Si le nombre de tons est suffisamment grand, nous pouvons obtenir l'illusion d'une image naturelle en noir et blanc.

1.1.1.2 Dichromie

Nous pourrions imaginer des systèmes de couleurs avec deux couleurs primaires, mais il semble que ces systèmes s'avèrent insuffisants pour représenter des couleurs qui paraissent naturelles et convaincantes. Bien qu'il soit facile de simuler un système dichrome avec un matériel trichrome, il semble qu'il n'existe pas de système fondamentalement dichrome.

1.1.1.3 Trichromie

Il semble que trois composantes suffisent pour tromper l'œil et lui faire voir une plage intéressante de couleurs. Si elles sont judicieusement choisies, ces composantes peuvent être mélangées de façon

à former un grand nombre de couleur, et peut-être un nombre assez vaste pour permettre une illusion convaincante d'avoir de « vraies couleurs ». La fig. 1.1.2 montre, en gris, toutes les couleurs perceptibles pour un humain typique, tandis que les couleurs simulées se retrouvent dans le triangle coloré.

Typiquement — mais ni forcément, ni tout le temps — les images sont construites à partir de trois couleurs primitives : le rouge, le vert et le bleu. Les longueurs d'ondes des couleurs primitives dépendent de la technologie (comme la télévision analogique) ou des standards (pour le cinéma 8K, par exemple).

1.1.1.4 Quadrichromie

Certains systèmes proposent une quatrième couleur primaire, comme par exemple le jaune ajouté aux primaires rouge, verte et bleue. Cela a pour effet d'accroître le nombre de couleurs représentables et le triangle des couleurs réalisables (c.f. fig. 1.1.2) devient un quadrilatère. Le nombre de couleurs nouvellement représentables variera en fonction de la nouvelle composante de couleur, mais on peut tout aussi bien accroître le nombre de couleurs représentables en choisissant mieux nos trois composantes primaires — peut-être même au point d'inclure la quatrième composante primaire.

1.1.1.5 Multichromie

Certains procédés utilisent un grand nombre de « couleurs pures » pour le rendu des images. C'est notamment le cas en imprimerie où, s'il est toujours possible de simuler les couleurs par tramage de quatre couleurs de base (magenta, jaune, cyan et noir, comme on le voit à la fig. 1.1.3 (a), et comme nous en discuterons plus longuement au chapitre 5), il est souvent préférable d'utiliser des couleurs exactes pré-mélangées. Avec des couleurs pures, les effets de tramage disparaissent et il est possible de représenter des couleurs impossibles à rendre à partir des couleurs de base choisies. Cependant, les exemples de la fig. 1.1.3 montrent que le rendu par tramage peut être satisfaisant.

1.1.1.6 Couleurs hyperspectrales

Pour les images hyperspectrales, les couleurs ne sont plus représentées par seulement trois composantes mais par un grand nombre de composantes, chacune correspondant à une longueur d'onde spécifique. Les images hyperspectrales sont souvent le résultat d'un passage au spectromètre, où chaque pixel est décomposé en un « arc-en-ciel » de quelques dizaines ou quelques centaines de longueurs d'onde. Lorsqu'on a une petite dizaine de longueurs d'ondes, on parlera alors d'image

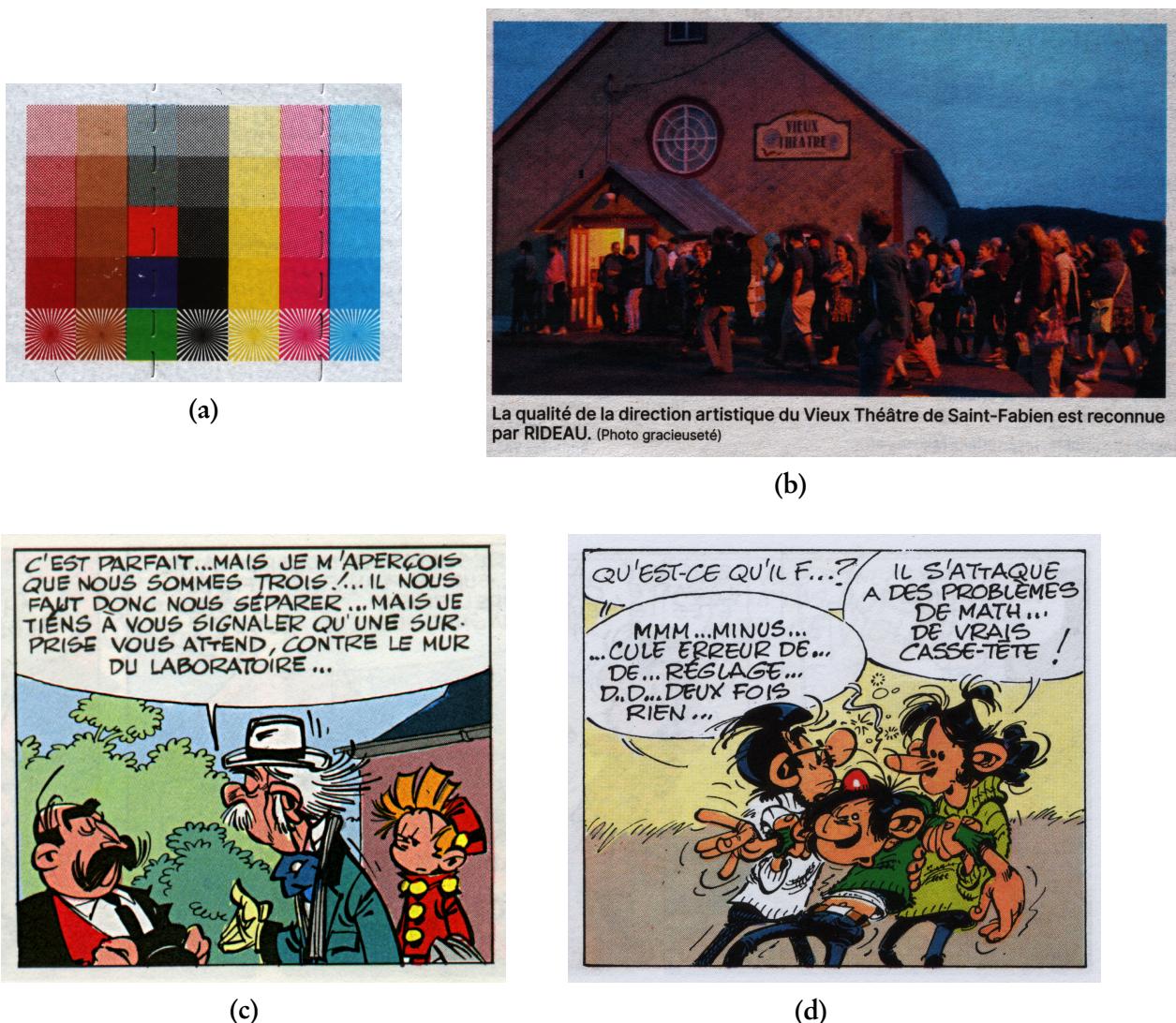


FIGURE 1.1.3 — Technique du tramage. a) À gauche, un étalon trouvé sur une boîte de poulet surgelé. b) À droite, une photographie rendue par tramage (source : *L'avantage*, vol 24 n° 34, p.13). c) Tramage d'une image tirée de Spirou, d) de Gaston Lagaffe [136, p. 7].

multiparticulaire; bien que la distinction entre hyperspectrale et multispectrale soit floue et varie d'un auteur à l'autre.

Les bandes de fréquences¹ peuvent couvrir le spectre visible (400nm, violet, à 700nm, rouge profond) ou un spectre bien plus étendu, depuis l'« infrarouge profond » (*deep infrared*) jusqu'aux ultraviolets ou même les rayons X! La table 1.1.1 présente quelques bandes de longueurs d'ondes et leur correspondance physique.

Ces images jouent un rôle important en télédétection (soit à partir d'un avion, soit un satellite),

1. Longueur d'onde et fréquence sont liées. À une longueur d'onde λ correspond une fréquence $f = c/\lambda$ et inversement $\lambda = c/f$. Les fréquences sont exprimées en Hz, les longueurs d'ondes en nanomètres, c est la vitesse de la lumière.

Nom	Longueurs d'onde	Application
Rayons X Ultra-violet	0.03nm–3nm 10nm–400nm	Radiographie Spectroscopie, photolithographie
Violets	380 ou 400nm–450nm	?
Bleus	450nm–520nm	Atmosphère, eaux (moins de 50m)
Verts/Jaune	515 ou 520nm–600nm	Végétation, eaux (moins de 30m)
Jaune/Rouges	600nm–690nm	Structures artificielles, eaux (moins de 10m)
Proche infrarouge	750nm–900nm	Végétation
Moyen infrarouge	1550nm–1750nm	Végétation, humidité des sols, feux de forêt
Infrarouge lointain	2080nm–2350nm	Végétation, humidité des sols, feux de forêt, silicates, argiles, feux
Infrarouge thermique	10000nm–12500nm	Chaleur, température des eaux, feux, « vision nocturne »

TABLE 1.1.1 — Longueurs d'ondes et phénomènes physiques.

et permettent de segmenter les images selon les différentes longueurs d'ondes. Il est ainsi bien plus facile de détecter les plans d'eau, d'estimer leur profondeur, la couverture végétale, la quantité d'eau contenue dans celle-ci... voire détecter les vilaines cultures illégales.

Si nous avons des bandes qui correspondent au spectre visible, on pourra toujours reconstruire une image visible en trois composantes en utilisant, par exemple, la sensibilité de l'œil aux différentes longueurs d'ondes¹. Cette sensibilité est montrée à la fig. 3.1.1.

Par contre, les images multi- ou hyperspectrale sont souvent rendues en *fausse couleurs*, où chaque bande reçoit une couleur (visible) aléatoire ou encore codifiée — jaune pour le sable, bleu pour l'eau, vert pour la végétation, gris pour les argiles, etc. Si les couleurs sont assignées en fonction d'une échelle, comme par exemple un gradient du bleu au rouge pour la température, nous parleront alors de *pseudo-couleurs*.

Par exemple, les superbes photographies multicolores de nébuleuses lointaines (voir fig. 1.1.4) du télescope Hubble sont composées en pseudo-couleurs en fonction de leur composition atomique : bleu-vert pour l'oxygène, rouge pour l'hydrogène, etc. [46, 99, 313, 360]. Le grand télescope JWST (*James Webb Space Telescope*), avec son miroir composite de 6.5 m, observe quant à lui l'univers dans les bandes infrarouge proches, moyennes et lointaines : une quarantaine de bandes allant de 600 nm (ou 0.6 μm ²) à 25500 nm (ou 25.5 μm) [53, 62, 67, 143, 151, 155, 156, 206, 220, 302, 315, 317, 318, 373, 376].

1. . Notons par ailleurs que la couleur est *perçue* par notre cerveau, et ne correspond en rien à des caractéristiques *a priori* physiques.

2. Il semblerait que la littérature favorise les μm lorsqu'on discute des longueurs d'ondes correspondant aux infrarouges, par ex. [271].



FIGURE 1.1.4 — Les « pilliers de la création ». Image composée de trois bandes : O-III (502 nm), H- α (657 nm), S-II (673 nm). Source : NASA, ESA/Hubble and the Hubble Heritage Team.



FIGURE 1.1.5 — Même région qu'à la fig. 1.1.4, mais composée en fausses couleurs à partir des images de deux instruments du *James Webb Space Telescope*, MIRI (infrarouge moyen et lointain) et NIRCam (proche infrarouge). Source : NASA/ESA/CSA.

1.1.1.7 Bande dynamique étendue et non-linéarité

La réponse du médium peut être linéaire ou non. Si la réponse est linéaire (d'aucuns pourraient penser qu'il s'agit d'une réponse idéale), l'intensité rendue sera identique à l'entrée, peut-être fois une constante de proportionnalité, c'est-à-dire quelque chose comme $f(x) = \alpha x$. Il est souvent pratique de supposer que c'est le cas.

Si la réponse est non-linéaire (et c'est presque toujours le cas), elle pourra atténuer ou accentuer certains niveaux, ou transformer toutes les valeurs. Dans les cas les plus simples, on parlera d'une fonction ou d'une *correction gamma* qui prend la forme $f(x) = ax^\gamma$. Les grandes valeurs de γ

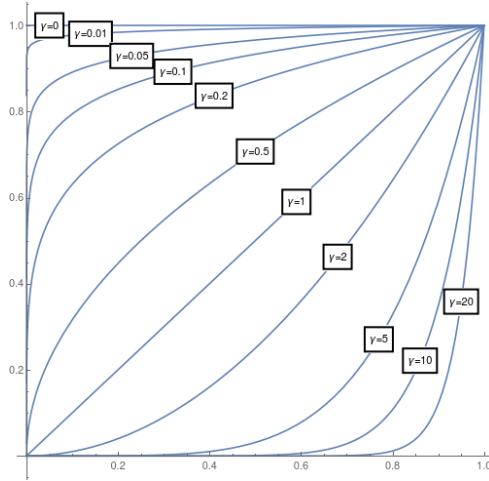


FIGURE 1.1.6 — Correction gamma.

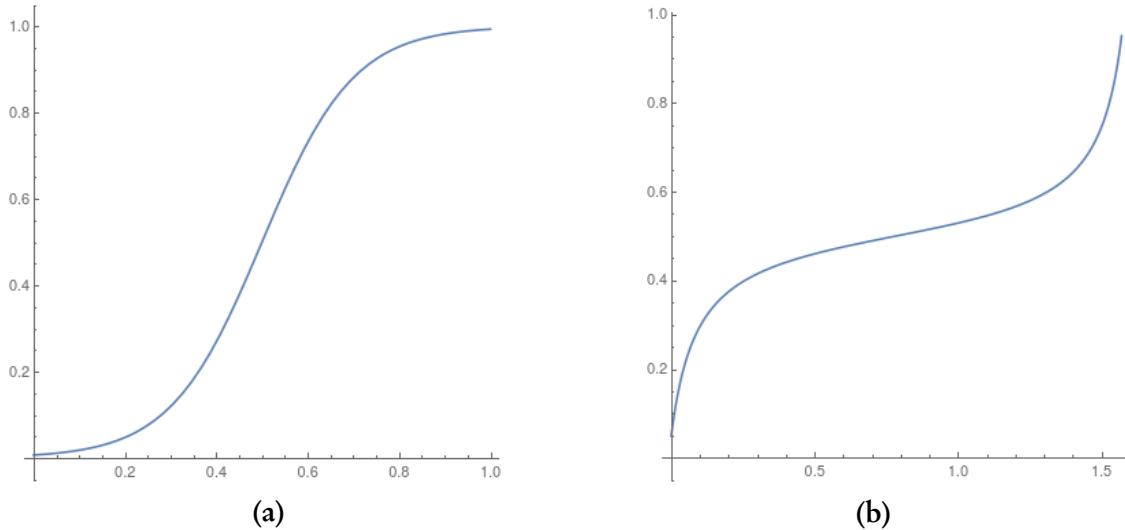


FIGURE 1.1.7 — Corrections plus souples. (a) À gauche, une fonction de contraste. (b) À droite, correction de bande dynamique.

augmentent le contraste mais obscurcissent l'image; les petites adoucissent le contraste et rendent l'image plus claire. Les corrections résultant de plusieurs valeurs de γ sont montrées à la fig. 1.1.6.

La correction gamma est quand même assez peu expressive car les formes qu'elle produit sont peu variées. D'autres lui préféreront des fonctions dites sigmoïdes (« en forme de s »). Nous en trouvons deux à la fig. 1.1.7. Ces fonctions sont utiles pour « améliorer » le contraste en obscurcissant davantage les couleurs sombres et en augmentant les couleurs claires (comme on le voit à la fig. 1.1.7 (a)).

Nous pouvons aussi utiliser ces fonctions pour simuler la réponse de l'œil aux scènes fortement contrastées. Contrairement à ce que nous pourrions penser, nous ne percevons pas une scène dans sa totalité d'un seul regard : il nous faut déplacer notre attention sur ses différents éléments pour



FIGURE 1.1.8 — Contraste simultané. (a) À gauche. En vert, les régions où l’œil perçoit bien les détails. En rouge, une région où l’ombre semble impénétrable. (b) À droite. La même scène corrigée par la fonction de la fig. 1.1.7 b).

avoir l’impression de la percevoir dans son ensemble. Lorsque nous portons notre attention sur une (petite) région de la scène, nous pouvons voir clairement des éléments dont la luminosité relative ne diffère que d’environ deux ordres de grandeurs (soit 100:1), qu’il s’agisse d’ombres ou de lumières. Si le contraste dépasse (localement) ce ratio, l’ombre nous paraît impénétrable. La fig. 1.1.8 (a) nous montre une de ces images. À la fig. 1.1.8 (b), nous avons la même image, mais cette fois-ci, nous avons appliqué la fonction de la fig. 1.1.7 (b). Nous retrouvons l’effet perceptuel sans pour autant augmenter le ratio de contraste maximal de l’écran. Notons, de plus, que cette correction n’est pas nécessairement appliquée en logiciel : l’écran peut très bien prendre en charge cette correction.

1.1.2 Encodage de la couleur

Présentons maintenant comment les couleurs sont encodées au niveau du matériel. Nous verrons comment la richesse des couleurs — et de leur encodage — a évolué dans le temps, des premiers ordinateurs domestiques jusqu’aux machines courantes.

1.1.2.1 *Direct Colors et Bit Depth*

Lorsque l’encodage d’un pixel donne directement les composantes de couleur, on dira que l’encodage est de type *direct color*. Si l’encodage réfère à une table, alors nous parlerons d’encodage par palettes — nous en discuterons en détail dans la prochaine section.

Si l’encodage est de type direct, alors un certain nombre de bits sont dévolus à l’encodage de la composante rouge, de la composante verte et de la composante bleue. Le nombre de bits utilisés pour

Bits	Mode	Exemple
3/4	D	[u,r g,b]
6/7	D	[B,r _b g _b b _b I,r _f g _f b _f] CGA avant- et arrière-plan (I pour intensité, B pour <i>blink</i>) [95, 96]
6	D	[u,u r,r,g,g b,b] EGA
9	P	[u,u u,u u,r r,r u,g g,g u,b b,b] Atari ST [295, p 2-13]
12	P	[u,u u,u r,r,r r,g g,g b,b,b,b] Amiga 1000 [294, p 2-13]
15	D	[u,r r,r r,r g,g g,g b,b,b,b,b] SVGA
16	D	[r,r r,r r,r g,g g,g g,g b,b,b,b,b] SVGA
18	P	[u,u r,r r,r r u,u g,g g,g g u,u b,b b,b,b,b] VGA
24	P	[r,r r,r r,r r g,g g,g g,g b,b b,b b,b] VESA

TABLE 1.1.2 — Quelques exemples d'encodage de couleurs. Légende : u (inutilisé), r,g,b, bits qui participent respectivement aux composantes rouges, vertes, bleues. P indique un mode palette, D un mode direct.

encoder la couleur est le *bit depth*, ou encore le *color depth* (traduit assez naïvement par *profondeur de couleur*). Si, pour arriver à un alignement compatible avec le matériel (typiquement, le nombre de bits est arrondi au prochain octet), l'encodage laisse quelques bits disponibles, ils peuvent encoder une quatrième composante, par exemple la transparence (universellement nommée α , probablement depuis 1984 [301], car avant, il semble qu'il ne soit question que de transparence [366]), ou encore un effet autre, comme la demi-brillance [229, p. 258] ou le clignotement. La table 1.1.2 montre quelques exemples d'encodage de couleur.

Lorsque le nombre de bits par pixel est de 15 ou 16, on parlera de mode *high color* ou de « milliers de couleurs ». Si le nombre de bits par pixel est de 24, on parlera alors de *True Color* (« vraies couleurs ») ou de « millions de couleurs » (ce qui peut être ambigu, puisque 20 bits suffisent pour donner « des millions » de couleurs).

*
* * *

La table 1.1.2 montre des encodages où le nombre de niveaux par composante est une puissance de deux. Cela est le résultat direct de l'utilisation d'un nombre entiers de bits pour encoder chaque composante. Ainsi, nous avons 2,8,16, 256, etc., niveaux, mais nous ne sommes pas contraints aux puissances de deux. Rien ne nous empêche d'avoir 19 ou 42 niveaux par composantes, ou même des nombres de niveaux différents d'une composante à l'autre !

La *web safe palette* (alias *Netscape Color Cube*) est un exemple d'un tel encodage. Chaque composante varie selon 6 niveaux, pour un total de $6 \times 6 \times 6 = 216$ couleurs. À l'époque où la plupart des cartes graphiques ne pouvaient que rendre 256 couleurs en même temps en haute résolution (souvent via un mode palette), les applications ne disposaient pas d'un grand nombre de couleurs

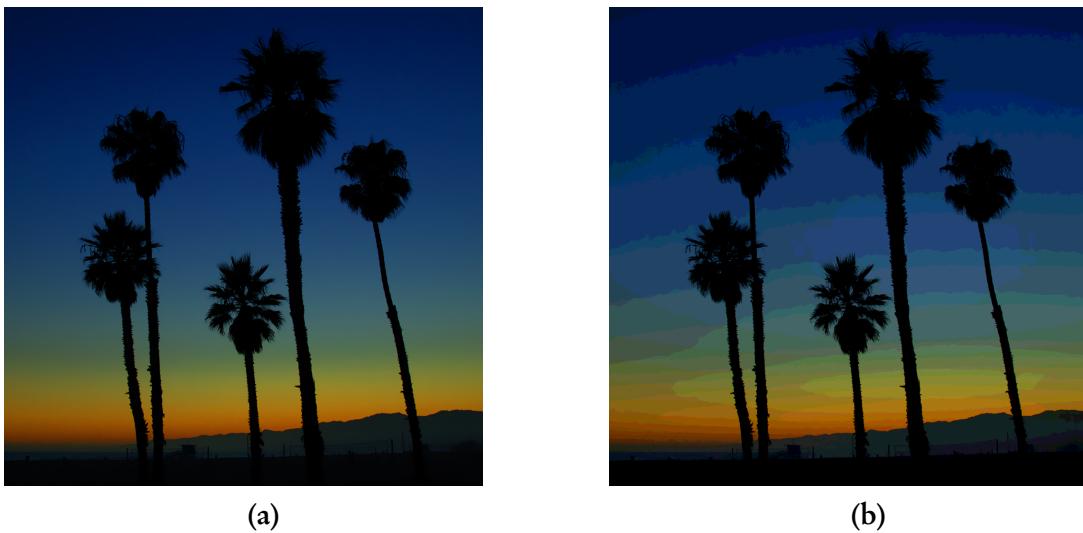


FIGURE 1.1.9 — Effet de faux contours dû à l’insuffisance de résolution de couleurs. (a) image originale (24 bits, 16777216 couleurs), (b) image avec 16 niveaux par composante (12 bits, 4096 couleurs, sans diffusion d’erreur).

libres, car certaines étaient réservées et fixes pour assurer que l’interface graphique demeure lisible en tout temps. Heureusement, les couleurs réservées pour l’interface étaient les 16 ou 32 premières couleurs, laissant ainsi les autres aux applications qui pouvaient les reprogrammer — et nous offrir un camaïeu de couleurs farfelues lorsque qu’on passait d’une application à l’autre.

*
* * *

Comme nous venons de le faire remarquer, le nombre de niveaux par composantes n’est pas nécessairement contraint. Souvent il est choisi pour exploiter efficacement le matériel sous-jacent (par exemple, un octet par composante), mais le matériel évolue constamment comme son coût de production chute constamment. Ainsi, nous voyons émerger des systèmes avec 10, 12, voire 16 bits par composantes, nous donnant des pixels sur 30, 36, voire 48 bits (on parlera alors de modes *deep color*). Cependant, même si en principe le matériel rend les couleurs sur plus de 24 bits, il est quand même encore hautement probable que le rendu final par l’écran soit bien en deçà des capacités théoriques du matériel — un œil exercé verra les limitations de l’écran où les tons en principe continus sont rendus par des couleurs discrétisées, faisant apparaître de « faux contours », comme à la fig. 1.1.9.

*
* * *

Mais alors, combien de bits seraient nécessaires pour un rendu parfait de n’importe quelle image? Pour répondre adéquatement, il faut d’abord comprendre comment l’œil réagit à l’intensité

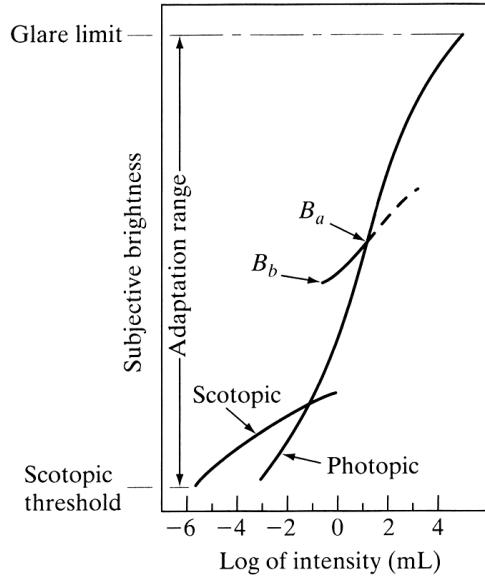


FIGURE 1.1.10 — La réponse psychovisuelle, d'après [149].

lumineuse. Certains estiment que cette bande dynamique s'étend sur 10 ordres de grandeur, du seuil de la perception jusqu'à l'éblouissement [149], d'autres plutôt sur 16 [35, 121]. La fig. 1.1.10 montre la courbe de réponse de l'œil à l'intensité lumineuse. En lumière normale, ce sont les cônes qui forment la vision en couleur (la vision photopique), tandis qu'en obscurité, ce sont les bâtonnets qui forment l'image (la vision scotopique). Pour représenter les intensités lumineuses avec précision sur D ordres de grandeur, il nous faut

$$\log_2 k 10^D = \log_2 k + D \log_2 10 \approx \log_2 k + 3.322D \quad (1.1.1)$$

bits. Ici, k est une constante de finesse qui peut être ajustée en fonction de la différence minimale perceptible entre deux intensités. Si on suppose $D = 10$, on trouvera

$$\log_2 k 10^{10} \approx \log_2 k + 33.22$$

bits, tandis qu'avec $D = 16$,

$$\log_2 k 10^{16} \approx \log_2 k + 53.15$$

bits. Cela semble indiquer qu'il nous faille utiliser un grand nombre de bits pour représenter une image avec la même précision qu'avec laquelle l'œil la perçoit. Revenons à la fig. 1.1.10. Au milieu de la figure, nous avons un point indiqué B_a , qui est, pour une scène donnée, l'illumination moyenne. Autour de cette illumination moyenne, l'adaptation de l'œil (qui ne se fait qu'après un moment) ne permet de résoudre qu'une certaine bande dynamique dont la marge inférieure est notée B_b dans la figure (et, par symétrie, une marge supérieure indiquée par le pointillé). Les luminosités plus faibles

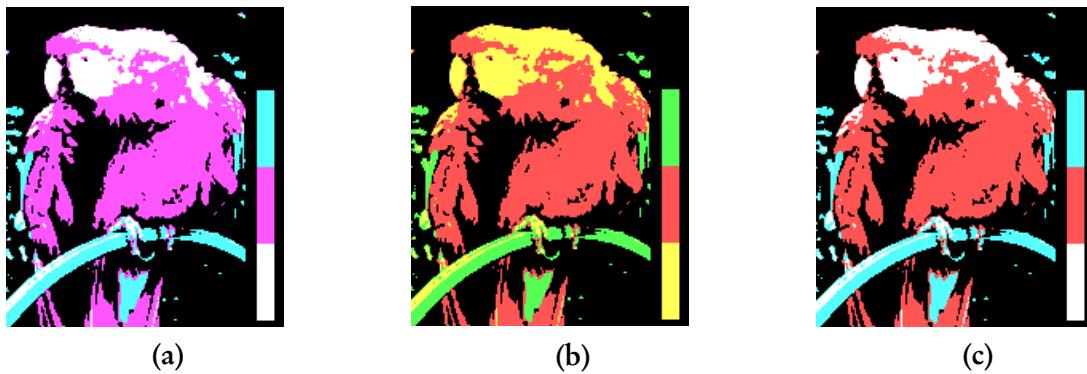


FIGURE 1.1.11 — Palettes CGA. (a) mode 4, palette 1, (b) mode 4, palette 2 et (c), mode 5. Source : Wikipedia.

que B_b sont perçues comme étant noires, et les luminosités plus grandes que son point symétrique comme « blanc », c'est-à-dire une luminosité d'où il nous est impossible de résoudre les détails.

Cette bande dynamique d'adaptation est bien plus restreinte que la bande dynamique de toutes les intensités visibles, que de toute façon, nous ne pouvons pas voir simultanément en entier. Cela nous donne donc l'idée de choisir judicieusement un B_a fixe, satisfaisant pour la télévision, ou pour l'infographie, et ajuster la bande dynamique entre B_b et le point symétrique B_c . La figure nous donne 4 ou 5 ordres de grandeur pour cette bande dynamique, ce qui, en utilisant l'éq. (1.1.1), nous donne $\approx \log_2 k + 16$ bits. En posant $k = 1$, nous arrivons à ≈ 16 bits par composante¹.

1.1.2.2 Palettes

Le mode palette est un compromis entre la mémoire utilisée pour stocker l'image et le nombre de couleurs possibles. Chaque pixel, plutôt que coder directement la couleur à afficher, devient un index dans une table de couleurs. Ainsi, on peut avoir des index assez courts (un octet, ou moins) qui pointent dans une table de couleurs riches (9, 12, 16 bits ou plus). Ce compromis permet de réduire l'espace mémoire nécessaire pour retenir l'image (et aussi la largeur du bus nécessaire pour y accéder) sans sacrifier trop de l'expressivité de l'image.

La palette peut être sous le contrôle d'un programme ou être fixée par le matériel. Ainsi, les modes graphiques CGA 320×200 offraient des modes à quatre couleurs, dont une seule programmable (noir par défaut); le choix du mode déterminait les 3 autres couleurs (en deux modes : faible et haute intensité) [95, 96]. Heureusement, d'autres matériels étaient capables de bien mieux ! Par exemple, l'Amiga 1000 était capable de modes avec une palette de 32 couleurs choisies parmi 4096. La

1. C'est encore un peu simplifier ; la luminosité d'une couleur est une fonction de toutes ses composantes, en plus des corrections non-linéaires qui ont pu y être apportées. Nous reviendrons sur le rôle des composantes dans la luminosité au chapitre 3, sur les espaces de couleurs.



FIGURE 1.1.12 — Le roi Toutânkhamon par Avril Harrison (1986), version Amiga. Source : Retroshowcase [29].

fig. 1.1.12 montre une telle image. Comme changer les couleurs contenues dans la palette n’implique pas forcément de recoder l’image, le mode palette permet de réaliser des animations (simples) à peu de frais : changer cycliquement certaines couleurs peut donner des effets de chatoiement ou encore donner l’illusion du mouvement.

1.2 Organisation des pixels dans l’image

1.2.1 Résolutions

Nous entendons parfois *résolution* et *définition* utilisés de façon indifférente pour parler du nombre de pixels qui forment une image. Cependant, la *définition* d’une image désigne plutôt la qualité (précision) de chaque pixel, tandis que la *résolution* en détermine le nombre.

Les résolutions typiques ont évolué au gré des technologies. La table 1.2.1 montre les résolutions d’écran communes (ou qui l’ont été). Il en existe évidemment beaucoup d’autres, dictées par les dimensions physiques des écrans, des technologies du moment, des lignes de produits — Apple, par exemple, offre des écrans de 1366×768 pixels sur certains de ses modèles. Pour la télévision, numérique comme analogique, les résolutions sont fortement standardisées pour assurer l’interopérabilité des signaux [54, 124, 125, 148, 157, 305]. La table 1.2.2 montre les résolutions pour la télévision. Notons que certaines résolutions sont *entrelacées* : pour des raisons techniques liées à la bande passante limitée, ces résolutions affichent en alternance les lignes impaires d’une image puis les lignes paires de l’image suivante. Cela permet de doubler le nombre d’images *perçues* par seconde sans doubler la bande passante, camouflant ainsi une partie des saccades de la vidéo.

Résolution	Sigle	Aspect	Nom
320×200	CGA	16:10	Color Graphics Adapter
320×240	QVGA	4:3	Quarter VGA
640×480	VGA	4:3	Video Graphics Array
720×358	Hercules	4:3	Monochrome
800×600	SVGA	4:3	Super VGA
1024×768	XGA	4:3	Extended Graphics Array
1280×720	HD	16:9	HDTV
1280×1024	SXGA	5:4	Super XGA
1440×900	WSXGA	16:10	Wide SXGA
1600×1200	UXGA	4:3	Ultra SXGA
1920×1080	HD 1080	16:9	Full HDTV
2048×1536	QXGA	4:3	Quad XGA
2560×1440	QHD	16:10	Quad HD
3840×2160	UHD	16:9	Ultra HD, 4K
7680×4320	UHD2	16:9	Ultra HD 2, 8K

TABLE 1.2.1 — Quelques résolutions communes

Résolution	Sigle	Aspect	Modes
704×576	PAL/SÉCAM	4:3	Entrelacé (analogique)
704×576	PAL/SÉCAM	16:9	Entrelacé (analogique)
720×576	PAL/SÉCAM	4:3	Entrelacé (numérique)
720×576	PAL/SÉCAM	16:9	Entrelacé (numérique)
440×486	NTSC	4:3	Entrelacé
720×480	NTSC	4:3	Entrelacé (480i)
1280×720	HD	16:9	HDTV, entrelacé (720i) ou progressif (720p)
1920×1080	HD 1080	16:9	Full HD, entrelacé (1080i) ou progressif (1080p)
3840×2160	UHD	16:9	Ultra HD, 4K, progressif
7680×4320	UHD2	16:9	Ultra HD 2, 8K, progressif

TABLE 1.2.2 — Quelques résolutions communes pour la télévision

L'interopérabilité était un souci : les différents standards de télévision n'étaient pas tous exactement compatibles entre eux. Le nombre d'images par seconde variait d'une région à l'autre (25 ou 30 images par secondes, décomposées en 50 ou 60 demi-images); ce qui demande des adaptations de la vidéo grâce à une gymnastique compliquée d'interpolation et de désentrelacement. Comme si cela n'était pas déjà assez compliqué, les résolutions différaient aussi, nécessitant, parfois, des formats intermédiaires facilement convertibles à l'une ou l'autre résolution standard. Les formats CIF (*Common Intermediate Format*) et SIF (*Source Input Format*), définis par la norme H.261 (MPEG-1), tentent de résoudre ce problème [6]. La table 1.2.3 montre les résolutions proposées par le standard CIF/SIF.

Résolution	Sigle	Nom
128×96	SQCIF	Sub QCIF
176×144	QCIF	Quarter CIF
352×240	SIF	Source Input Format
352×288	CIF	Common Intermediate Format
704×480	4SIF	Quad SIF
704×576	4CIF	Quad CIF
1408×1152	16CIF	

TABLE 1.2.3 — CIF et SIF

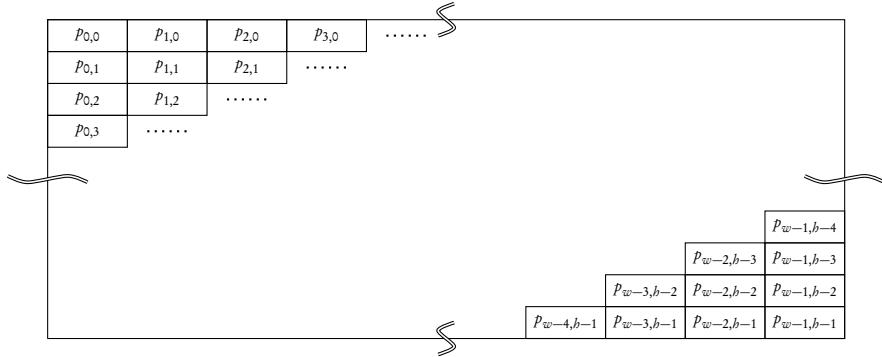


FIGURE 1.2.1 — L'organisation en plage contiguë.

Si les écrans — et la télévision — ont plutôt tendance à imposer des résolutions particulières, les images, comme les vidéos, peuvent utiliser n’importe quelle résolution. L’ordinateur s’accommode fort bien des images minuscules ou immenses, et la plupart des *media players* sont indifférents à la dimension de l’image vidéo, se contentant de les redimensionner en temps réel au besoin.

1.2.2 Organisation en mémoire

L’organisation la plus naturelle est de placer les pixels les uns à la file des autres dans une plage de mémoire contiguë et sans sauts. La fig. 1.2.1 montre schématiquement cet arrangement. Traditionnellement, le pixel en haut à gauche reçoit l’adresse (relative) 0. Cela facilite le calcul d’adresse à l’intérieur de l’image car il devient simplement

$$a(i, j) = p + s(iw + j),$$

où p est l’adresse de base, s est la taille (en octet) d’un pixel, i est la rangée et j la colonne, tous deux commençant à zéro. Cette convention quasi-universelle comporte certaines exceptions, comme, par exemple, le format d’image BMP de Microsoft qui, en plus d’une multitude de complications inutiles, commence par le pixel en bas à gauche [27, 205, 270].

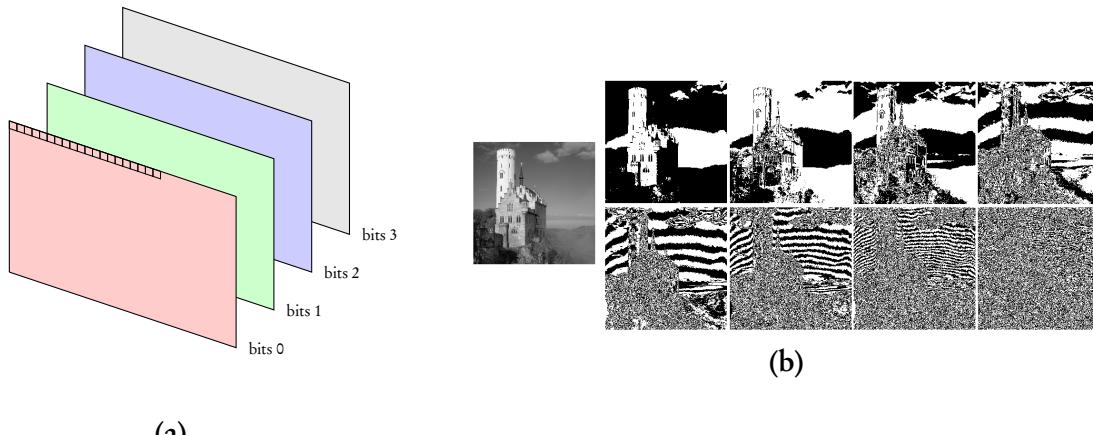


FIGURE 1.2.2 — Organisation en plans de bits des pixels en mémoire. (a) Plans de bits, (b) Image décomposée en plans de bits (Source : Wikipedia).

L'organisation en plans de bits (*bit planes*) regroupe tous les bits de même poids dans une même plage de mémoire. La fig. 1.2.2 détaille cette organisation. Cela permettrait, en principe, d'ajouter des plans et d'augmenter la profondeur des pixels; ou en avoir moins et obtenir naturellement une image avec moins de profondeur. À la limite, on pourrait supposer que cela uniformise la gestion de l'image, plus ou moins profonde en fonction de la mémoire disponible.

Dans une organisation en plans de bits, changer un pixel demande de modifier des bits dans plusieurs plans, ce qui demande un calcul d'adresse par bit, ce qui peut se révéler assez peu efficace. En effet, l'adresse de l'octet qui contient le bit b à la position (i, j) est donnée par

$$\begin{aligned} a(i, j, b) &= p + b \left\lceil \frac{w}{8} \right\rceil h + i \left\lceil \frac{w}{8} \right\rceil + \left\lfloor \frac{j}{8} \right\rfloor \\ &= p + \left\lceil \frac{w}{8} \right\rceil (bh + i) + \left\lfloor \frac{j}{8} \right\rfloor, \end{aligned}$$

où w est la largeur de l'image en pixels, h sa hauteur et p le pointeur vers le début de la plage de mémoire.

*
* * *

Les plans peuvent aussi être utilisés avec de la transparence, comme à la fig. 1.2.3. La transparence est gérée automagiquement par le matériel : une « couleur » spéciale est réservée pour coder la transparence, et la lecture d'un pixel transparent provoque la lecture du pixel correspondant dans le plan suivant. S'il est lui aussi transparent, on passe au prochain plan, et ainsi de suite, jusqu'à concurrence du nombre de plans supporté par le matériel. Cela permet de simuler une profondeur de champs, surtout avec les jeux vidéo. Pour simuler l'effet de parallaxe, il suffit de déplacer le plan

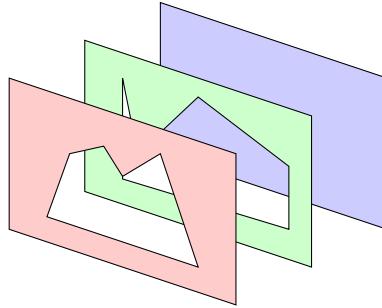


FIGURE 1.2.3 — L'organisation en plans avec transparence.

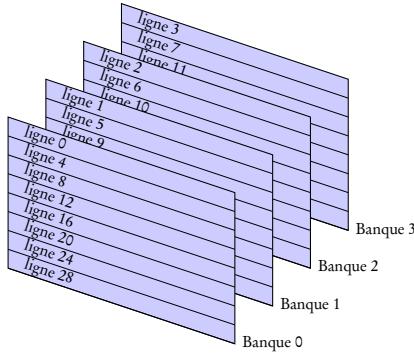


FIGURE 1.2.4 — Organisation Entrelacée.

le plus profond plus lentement que le plan médian, lui aussi plus lent que le premier plan. Cela rend l'illusion du mouvement tel qu'on le percevrait en train : la montagne lointaine bouge peu dans notre champ de vision mais l'avant plan défile à toute allure.

*
* * *

Le dernier arrangement que nous considérerons ici est l'organisation entrelacée des pixels. Dans cette organisation, l'image est décomposée en $n = 2^k$ banques. La première banque contient toutes les lignes de l'image qui sont congruentes à 0 modulo 2^k (donc $0, 2^k, 2 \times 2^k, 3 \times 2^k, \dots$), la seconde les lignes congruentes à 1 modulo 2^k ($1, 2^k + 1, 2 \times 2^k + 1, 3 \times 2^k + 1, \dots$), la troisième les lignes congruentes à 2 modulo 2^k , et ainsi de suite. La fig. 1.2.4 montre une organisation en quatre plans.

Cette organisation qui paraît compliquée a pourtant une utilité. La mémoire vidéo est accédée par deux côtés : d'un côté, le processeur qui peut aller y lire et y écrire, et de l'autre, le générateur d'image qui doit y accéder en lecture (seulement) pour produire l'image vidéo. Si la mémoire ne supporte pas les accès simultanées, écrire à (ou lire de) la mémoire bloque la lecture par le générateur d'image. C'est d'ailleurs ce qu'on avait avec les premières cartes graphiques : écrire trop rapidement à la mémoire vidéo se traduisait par des lignes de « neige » dans l'écran ! Pour palier les limitations

de la mémoire, les premières cartes graphiques utilisaient une organisation entrelacée de la mémoire — une solution maintes fois reprise pour la mémoire partagée [173, 191, 292]. Comme les lignes de l'image doivent être générées dans l'ordre de l'image (à cause des limitations des moniteurs d'alors [157]), l'organisation linéaire de la mémoire était présentée du côté du générateur vidéo, laissant l'organisation entrelacée du côté du processeur. Ainsi, écrire dans une banque n'affecte qu'une ligne sur quatre (comme dans la fig. 1.2.4) et si on écrit assez rapidement, on limite la quantité de neige dans l'écran [95].

1.2.3 Sous-échantillonnage

Jusqu'à maintenant, nous n'avons considéré que les couleurs *RGB*. Si ces trois couleurs primaires sont suffisantes pour recréer des couleurs réalistes, nous les percevons pas individuellement. Si l'œil humain est muni de cônes qui détectent le rouge, le vert et le bleu et de bâtonnets qui sont sensibles à l'intensité de la lumière, le signal envoyé au cerveau n'est pas directement celui mesuré par les cônes et les bâtonnets, mais un calcul qui estime la luminosité, la différence rouge-vert et la différence jaune-bleu [35]. Ainsi, l'œil est très sensible aux différences de luminosité, mais beaucoup moins aux différences de teinte et de saturation. Les formats d'images exploitent souvent cette caractéristique et utilisent plutôt un espace de couleur où une composante représente la luminosité, et où les deux autres correspondent — au moins approximativement — à la teinte et à la saturation, mais plus souvent à la différence rouge-vert et jaune-bleu. De plus, comme nous sommes beaucoup moins sensibles aux deux dernières composantes, nous pouvons nous permettre de les sous-échantillonner, c'est-à-dire d'avoir une résolution beaucoup plus faible pour ces composantes que pour la luminosité. JPEG, par exemple, utilise cette technique [297]. L'avez-vous remarqué? Non? *My point exactly.*

Le sous-échantillonnage est noté par trois (parfois 4) nombres, de la forme $l:c_1:c_2$ ou $l:c_1:c_2:\alpha$ [305, p. 90], où :

- l est le nombre d'échantillons horizontaux de luminance. Conventionnellement 4 à cause d'une contrainte technique liée au signal de télévision, mais on trouve — rarement! — d'autres valeurs, comme 3. On suppose que l'image est toujours « pleine résolution » dans le domaine de la luminance.
- c_1 est le nombre de changements de chrominance pour l pixels horizontaux.
- c_2 est le nombre de changements de chrominance d'une ligne à l'autre.

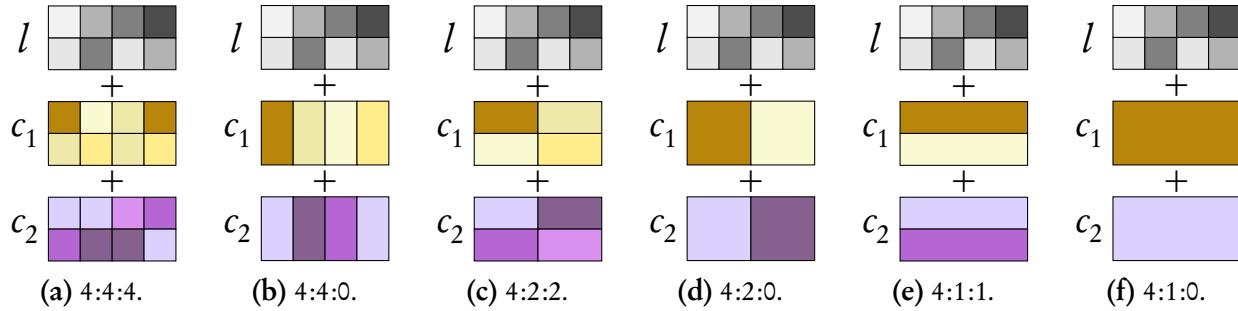


FIGURE 1.2.5 — Principaux motifs de sous-échantillonnage des couleurs.

- α , lorsque présent, représente les facteurs de transparence ; habituellement le même que l , mais pas nécessairement.

Les sous-échantillonnages les plus fréquents sont

- 4:4:4 : Chaque pixel a une composante de luminance et deux composantes de chrominance — il n'y a pas de sous-échantillonnage. Supporté pour la vidéo haute définition « *studio profile* » (MPEG-4, Part 2).
- 4:2:2 : Hérité de la télévision analogique [13, 102, 105, 124, 125, 157], se retrouve encore dans les standards HDTV [21], et correspond historiquement à la transmission entrelacée.
- 4:2:0 : présente un bon compromis entre la résolution de luminance et de chrominance. Ce sous-échantillonnage est supporté par un grand nombre de formats d'images (comme JPEG [297]) et de vidéo (MPEG, H.264, HEVC, AV1, etc.).

La fig. 1.2.5 montre les sous-échantillonnages que l'on retrouve dans les standards et les produits commerciaux. La fig. 1.2.5 (d) nous montre une image sous-échantillonnée en mode 4:2:0, tandis que la fig. 1.2.6 nous montre comment les couleurs changent avec ce sous-échantillonnage. Nous reviendrons, au chapitre 3, sur les motivations de l'utilisation du sous-échantillonnage de couleur.

1.3 Exemples de formats d'image

L'organisation en mémoire des images reflète souvent l'architecture de la machine sous-jacente, mais les formats de fichiers qui sauvegardent les images s'efforcent, au contraire, de s'en absoudre

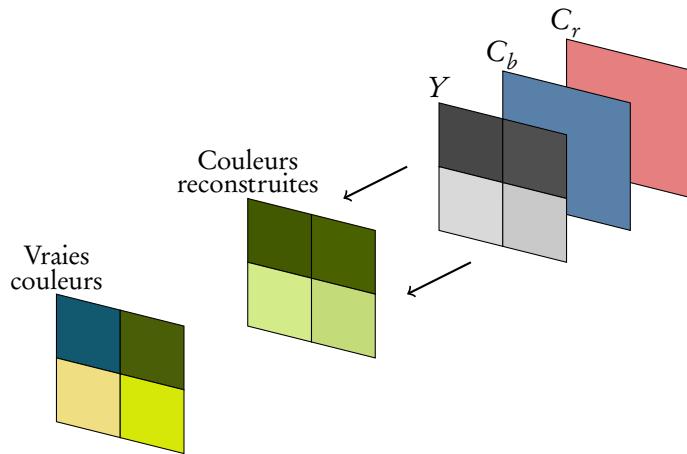


FIGURE 1.2.6 — Les couleurs sous-échantillonnées en mode 4:2:0 changent légèrement car les pixels partagent les composantes de chrominance et seule la composante luminance est individuelle. La transformée de couleur utilisée est YC_bC_r (voir § 3.3.9.4), et les couleurs de la figure sont précises.



(a)



(b)

FIGURE 1.2.7 — Des images sous-échantillonnées en 4:2:0. (a) En haut, l'image reconstruite (Gus et MIPS). (b) En Bas, les composantes : en pleine résolution, la luminance, à moitié résolution, les composantes de chrominance.

et d'assurer un rendu cohérent sur les différentes plate-formes. Les formats de fichiers proposent, souvent de façon différente, de balancer la taille du fichier avec la qualité du rendu, c'est-à-dire que la compression peut être avec ou sans perte.

La compression sans perte restitue, bit pour bit, les données originales. Ce stockage sans perte est perçu comme essentiel pour les applications médicales, pour l'archivage, ou pour la masterisation. La compression avec perte essayera de détruire l'information peu perceptible afin de réduire davantage la taille du fichier. Nous reviendrons plus longuement sur la compression au chapitre ??.

Quelques formats usuels :

- PNM : alias *netpbm*, pour *portable any-map format*. Un format extrêmement simple avec une entête qui annonce le mode (noir et blanc, *RGB*, *RGBA*) et la résolution. Introduit par Jef Poskanzer en 1988, ce format est demeuré marginal, mais sans disparaître.
- GIF (*graphics interchange format*¹). Ce format est spécialement conçu pour les images par palette, avec au plus 8 bits par pixel. La compression utilise une variante de l'encodage LZ78 [387], LZW, que l'on doit à Welch [371, 375]. Le format prévoit aussi un mode entrelacé pour permettre d'afficher une image approximative même lorsque la transmission n'est pas complétée (un *must* lorsque la vitesse de transfert se calcule en kilobits/s!). Prévoit aussi un mode « multi-image » pour les animations. La limitation la plus importante est le nombre de couleurs, limité à 256 par image.
- TIFF (*Tagged Image File Format*) est un format originalement proposé par les fabricants de numérisateurs de documents, mais a été rapidement adopté pour la publication et l'info-graphisme. TIFF supporte un grand nombre d'espaces de couleurs, dont *RGB*, *CYMK*, $YC_r C_b$, CIÉ $L^* a^* b^*$, ..., avec différentes profondeurs. Ce format supporte aussi plusieurs algorithmes de compression, comme RLE, CCITT G3 [16], LZW et même des modes avec perte utilisant JPEG [5].
- JPEG (*Joint Photographic Experts Group*), du nom du comité ISO/IEC chargé de développer un nouveau standard) [297]. Le format JPEG commence par changer l'espace de couleur *RGB* vers $YC_r C_b$, puis sous-échantillonne (de façon paramétrable) les composantes C_r et C_b (qui correspondent à la différence rouge-vert et jaune-bleu, respectivement). Ces plans sous-échantillonnés sont ensuite transformés dans un domaine

1. D'après l'auteur, GIF se prononce « jiffe ». Tout le monde dit « guif », comme *gift* ou *graphics*.

fréquentiel où les données jugées imperceptibles sont détruites. Suit enfin un codage entropique, de style Huffman ou codage arithmétique.

Le successeur de JPEG, JPEG2000, utilise les ondelettes comme transformée et un format d'encodage beaucoup plus sophistiqué [352]. Cependant, la relative complexité de l'implémentation et l'efficacité, ainsi que le support universel du format JPEG original ont grandement freiné l'adoption de ce nouveau standard.

- **PNG (*Portable Network Graphics*)**. Ce format sans perte supporte plusieurs espaces de couleurs, la détection d'erreur, et la compression grâce aux prédicteurs [17, 70]. L'algorithme de compression, comme celui de GIF, est un dérivé de l'algorithme LZ78, LZSS [342]. Ce format supporte aussi l'entrelacement¹ pour les connexions lentes.
- **Kodak PhotoCD**. Originalement conçu pour la photographie numérique haute résolution (6144×4096 par exemple), le format est maintenant tombé en désuétude [365]. La compression utilisée est relativement simple. Après une transformation vers un espace de couleur propriétaire, KodakYCC, l'image est codée par prédiction : l'image est réduite à $1/4$ de la résolution (réduction par 2 dans les deux dimensions) puis réinterpolée à pleine résolution pour servir de prédicteur, à partir duquel on code les différences [122].
- **RAW**. Il existe un grand nombre de formats pour les images issues de caméras numériques (ou de numériseurs de document). Ces formats supportent des profondeurs de couleurs plus grandes qu'à l'habitude, et stockent les valeurs sans perte (parfois sans compression). De plus, ils incluent les méta-données de la prise de vue (heure, date, géolocalisation, temps d'exposition, température de couleur, correction de couleur, etc.). Les méta-données sont stockées selon le standard EXIF [22], tandis que les données elles-mêmes sont stockées selon l'algorithme choisi pour le format particulier (par ex. [9]) et de façon à accommoder la géométrie du senseur — sur laquelle nous reviendrons au chapitre 2. Le standard DNG (*digital negative*) d'Adobe se veut un format *raw* universel [23].

Il existe encore un grand nombre de formats de fichiers. Pour en savoir plus, vous pouvez consulter [74, 205, 270].

1. L'entrelacement Adam7 a été retenu. Pourquoi ce nom ? C'est le 7^e essai... d'Adam M. Costello, un des contributeurs originaux au projet PNG.



FIGURE 1.4.1 — Les caractères semigraphiques du TRS-80 Color Computer. Source : Wikipedia.

1.4 Remarques bibliographiques

Le format PNG a été conçu pour remplacer GIF qui risquait de ne plus pouvoir être utilisé librement. En effet, le propriétaire de l'algorithme de compression, la société Unisys (originalement Sperry Corporation), voulait exiger licences et royaumes, grâce au brevet qu'elle détenait [372]. Cela en contraria plusieurs, et dès 1995, un petit groupe se mit au développement d'un format libre de brevets [234, ch. 7]. Dès l'année suivante, les bases étaient jetées [70]. Ultimement, PNG finit par s'imposer, mais GIF continuera d'être utilisé par tous impunément et — surtout — impécuniairement.

*
* * *

Les vieux ordinateurs (pour lesquels, par ailleurs, je n'éprouve *aucune* nostalgie) nous enseignent plein de trucs astucieux d'économie de mémoire et de parcimonie d'encodage. Certains ordinateurs permettaient, par exemple, de changer la palette à chaque ligne de l'image grâce à une interruption matérielle ; d'autres encore simulaient un mode graphique grâce aux caractères « semigraphiques ». Par exemple, le TRS-80 Color Computer combinait à la fois forme et couleur des caractères (voir fig. 1.4.1) [32]. Des astuces similaires se retrouvent encore dans les algorithmes de compression de texture.

*
* * *

Les images multi- ou hyperspectrales occupent une place prépondérante en exploration terrestre [154] comme spatiale [46, 221, 239, 360].

2

Saisies des images

Sommaire. Dans ce chapitre, nous nous intéresserons à la formation d'image dans une caméra. Nous discuterons des senseurs, de la séparation des couleurs, et finalement des *color filter arrays*.

2.1 Introduction

Dans ce chapitre, nous nous intéresserons à la formation d'image, mais du point de vue des senseurs seulement. Bien que la quincaillerie d'un appareil photo ou d'un numérisateur, les lentilles, la mise au point automatique (*autofocus*), les diaphragmes, les obturateurs, etc., soient tous des sujets fort intéressants, nous n'en parlerons pas ici.

2.2 Senseurs et couleurs

2.2.1 Technologies de senseurs

Il existe deux grandes familles de (photo)senseurs :

- Les senseurs actifs (souvent dits CMOS¹, à cause de la technologie d'intégration de circuits utilisée). Ce type de senseur utilise une grille de photodiodes (ou de photorésistances) pour

1. Pour *Complementary Metal-Oxyde Semiconductor*.



FIGURE 2.2.1 — L'effet du *rolling shutter*, résultant du jeu entre le déplacement de l'objet et l'ordre de lecture des pixels. Source : Wikipedia [316].

capturer l'image. Or, pour mesurer la quantité de lumière reçue par un pixel en particulier, la photodiode (ou photorésistance) doit être traversée par un courant. Ce courant est activé au moment de la lecture du pixel; ce qui implique que l'image n'est pas capturée dans son ensemble d'un coup, mais séquentiellement pixel par pixel [133, 134, 314]. Cette technologie présente plusieurs avantages :

- Grande densité réalisable. Des senseurs de taille très réduite sont idéaux pour les applications où on peut se contenter d'une qualité d'image moindre, comme les appareils photo intégrés aux téléphones cellulaires.
- Grande sensibilité. Tandis qu'un film argentique typique offre une sensibilité entre 100 et 10000 ISO [4, 15], les senseurs CMOS peuvent aller jusqu'à 4000000 ISO (un senseur fabriqué par Canon).
- Le fait que les pixels doivent être activés/adressés individuellement pour être lus a pour effet secondaire de permettre de lire un sous-ensemble arbitraire des pixels. Cela permet de saisir des images à plus faible résolution (soit en espaçant régulièrement les pixels lus, soit en restreignant la région lue à un rectangle) plus rapidement.

Par contre, comme chaque pixel est lu séquentiellement, les images saisies par ce type de senseur peuvent être déformées. L'effet le plus connu est celui du *rolling shutter* [28], dont un exemple est montré à la fig. 2.2.1.

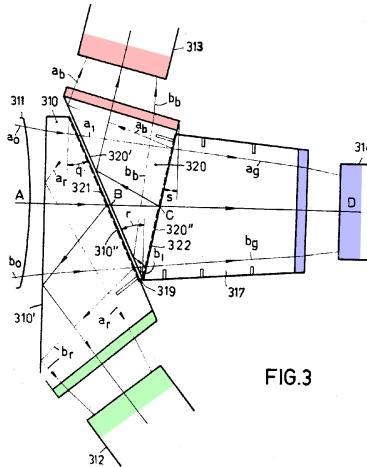


FIG.3

FIGURE 2.2.2 — Trois prismes séparant la lumière entrante en trois faisceaux, chacun filtré pour ne laisser passer qu'une plage de couleurs. Tiré de [104], couleurs ajoutées.

- Les senseurs à charge couplée. Ces senseurs utilisent une charge électrique isolée du senseur à proprement parler, et ce sont les photons qui traversent le pixel qui charrent les électrons de la réserve vers le détecteur. Conçus par Boyle et Smith en 1969 [73] et rapidement mis en œuvre par Amelio *et al.* [42], ces senseurs permettent la saisie de l'image dans son ensemble au même instant, puisque ce sont les photons reçus qui transfèrent les charges. Par contre, pour sortir les valeurs des pixels, il faut quand même les lire séquentiellement (grâce à une mécanique compliquée de registre à décalage).

Cependant, ces senseurs sont monochromatiques — ou plus exactement *panchromatiques* — et doivent être aidés de filtres et/ou de prismes pour saisir des images en couleur.

2.2.2 Prismes et filtres

voidComme les senseurs eux-mêmes sont typiquement monochromatiques, il faut soit scinder la lumière en plusieurs bandes de longueurs d'onde, soit appliquer des filtres ne laissant passer que certaines longueurs d'onde devant les senseurs, voire les deux !

La première solution — séparer la lumière en bandes — est montrée à la fig. 2.2.2 [104]. Ici, des prismes séparent et reflètent la lumière, créant trois chemins qui se terminent par un filtre de couleur. La lumière qui atteint un senseur est alors monochromatique (ou presque) : rouge, verte, bleue.

Une seconde solution (qui paraît plus récente) est d'utiliser des prismes dichroïques (étymologiquement *deux couleurs*) ou trichroïques (*trois couleurs*) pour séparer la lumière blanche en couleurs choisies, possiblement polarisée [167]. La fig. 2.2.3 montre quelques unes des configurations. La

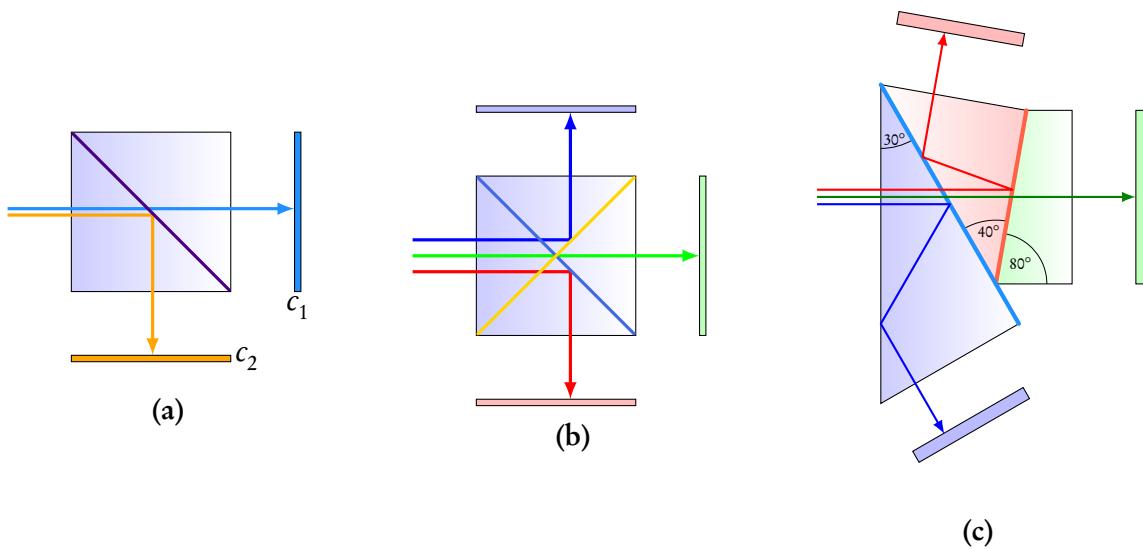


FIGURE 2.2.3 — Prismes pour la séparations de couleur. (a) prisme dichroïque, (b) trichroïque (*cross dichroic*) et (c) trichroïque de style Philips.



FIGURE 2.2.4 — Un prisme dichroïque (taille $\approx 1\text{cm}^3$) du type montré à la fig. 2.2.3 (b). Photo : Steven Pigeon.

fig. 2.2.3 (a) montre une configuration où deux couleurs sont séparées et dirigées vers deux senseurs. Les configurations qui nous intéressent le plus sont montrées aux figs. 2.2.3 (b) et 2.2.3 (c), cette dernière reprenant la configuration de la fig. 2.2.2, tandis que la première montre le schéma d'un prisme semblable à celui montré aux figs. 2.2.4 et 2.2.5. Pour toutes astucieuses qu'elles soient, les solutions basées sur des prismes di- ou trichroïque ont pour désavantage principal d'être encombrantes, en plus de demander un alignement précis des senseurs pour que les images séparées correspondent sur tous les senseurs. Il est donc très difficile de construire ces assemblages dans des appareils aussi compacts que des téléphones mobiles! C'est pourquoi on leur préfère souvent d'autres solutions, comme les CFAs (*color filter arrays*), sujets de la section suivante.

2.2.3 Systèmes à un seul senseur

Si nous voulons nous restreindre à un seul senseur, il faut que chaque pixel mesure les trois composantes de couleur. Cela mène à un arrangement similaire à la fig 2.2.6 où chaque région du



FIGURE 2.2.5 — Un exemple de caméra utilisant l’assemblage de type Philips. Source : Wikipedia [385].

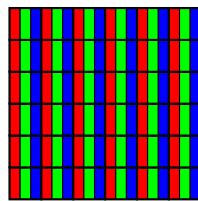


FIGURE 2.2.6 — Un senseur « pleine couleur », où chaque pixel possède trois types de détecteurs.

senseur (donc un « pixel ») possède trois détecteurs séparés, chacun réagissant à une couleur. La géométrie la plus évidente — mais pas la seule possible! — est d’avoir trois régions rectangulaires qui correspondent aux composantes du système d’imagerie (voir ch. 1). Cependant, cette solution comporte ses inconvénients. Comme la densité des pixels augmente, la taille de chacune des régions est d’autant plus réduite qu’elle n’occupe qu’un tiers de pixel, ce qui en réduit la sensibilité. De plus, il devient difficile de créer les lentilles qui concentrent la lumière reçue par la région rectangulaire sur le senseur lui-même (comme à la fig. 2.2.7).

2.2.3.1 Bayer et ses amis

Plutôt que d’utiliser un senseur où chaque pixel comporte trois senseurs (un pour chaque couleur), Bayer proposa d’utiliser une grille où chaque pixel n’aurait qu’une lentille et un senseur, mais pour une seule des couleurs primaires. Pour s’assurer de mesurer (au moins approximativement) les

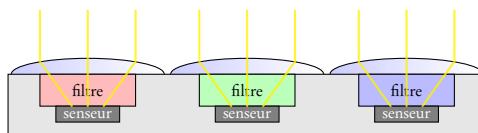


FIGURE 2.2.7 — Un senseur avec filtres de couleur et microlentilles.

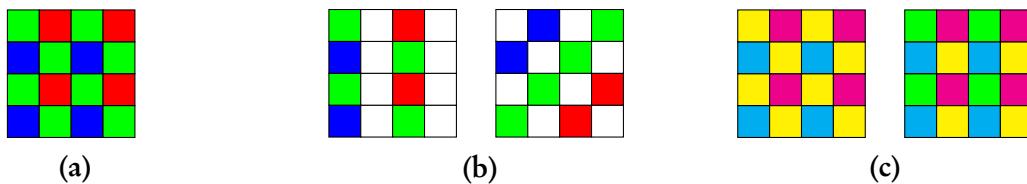


FIGURE 2.2.8 — Motifs de Bayer. (a) Motif original de Bayer [51], (b) variantes Kodak [94], (c) variantes avec d’autres primitives de couleur (cyan, magenta, jaune et cyan, magenta, jaune, vert).

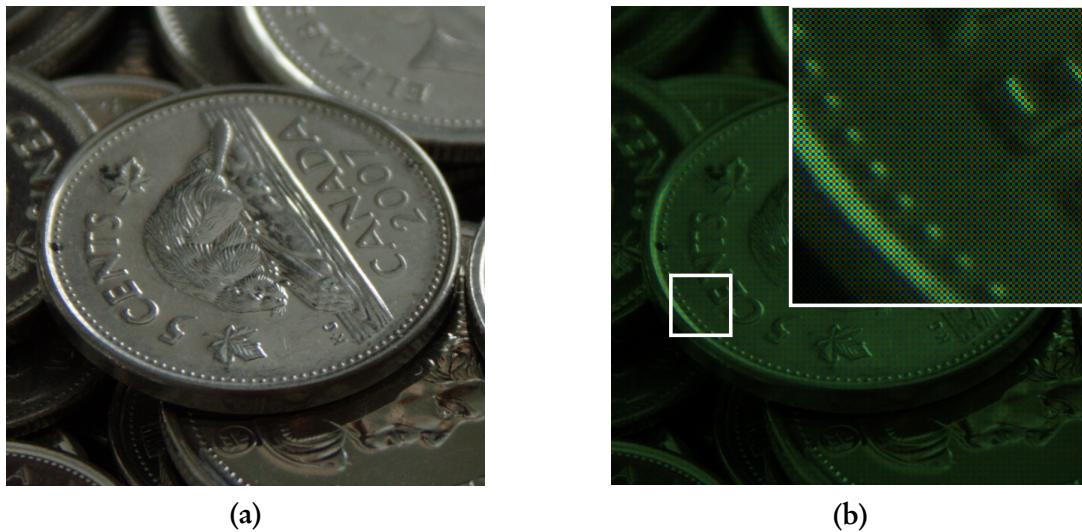


FIGURE 2.2.9 — Image saisie avec un motif de Bayer. (a) Image désirée, (b) image telle que saisie par le senseur, en encadré, la zone agrandie.

couleurs dans les trois composantes, les filtres de couleurs sont dispersés comme nous le montre la fig. 2.2.8 (a) [51]. La fig. 2.2.8 montre aussi quelques variantes où certains senseurs, montrés en blanc, sont « panchromatiques », c'est-à-dire qui réagissent à la lumière selon un spectre assez large de longueur d'ondes ; d'autres encore utilisent des primitives différentes, comme le cyan, le jaune, etc.

Le problème évident avec cette technique, c'est que chaque pixel, au sortir du senseur d'image, ne comporte qu'une des trois composantes de couleur ! C'est bien embêtant et il faut utiliser un algorithme pour « inventer » les couleurs manquantes. Certains se contentent d'algorithmes assez simples — voisin le plus près, interpolation linéaire, etc. — mais d'autres préféreront des algorithmes qui préservent la teinte et les rebords tout en évitant les effets de moiré. Parmi les algorithmes se retrouvant fréquemment dans les logiciels de traitement de photographies (dans un ordre quelconque) :

- AHD (*Adaptive Homogeneity Directed Demosaicing Algorithm*), par Hirakawa et Parks [176–180]. Cet algorithme estime les variations de couleurs par direction et utilise la direction de variation minimale pour interpoler, puis utilise un filtre d'antialiasage

sélectif pour adoucir les « escaliers ».

- PPG (*Patterned Pixel Grouping*), par Lin (2003) [233]. Cet algorithme estime les gradients, mais dans l'espace de la luminance. Il prend ensuite une décision en fonction du gradient le plus petit, préférant les plages au rebords.
- VNG (*Variable Number of Gradients*), proposé par Chang *et al.* [89]. Cet algorithme estime les gradients dans huit directions pour procéder heuristiquement au choix des valeurs manquantes. Cet algorithme tente de préserver les rebords et leur direction. Il existe une variante, VNG4, où les pixels verts sont traités comme deux couleurs distinctes : on a rouge, bleu, vert-1 et vert-2.

On trouve plusieurs autres algorithmes, certains assez quelconques, d'autres qu'on ne connaît qu'à travers leur implémentation, n'ayant pas fait l'objet de « vraies » publications. Parmi ceux-ci, on trouve RCD (*Ratio-Corrected Demosaicing*), DCB par Jacek Góźdż et AMaZE (*Aliasing Minimization and Zipper Elimination*) par Emil J. Martinec.

2.3 Remarques bibliographiques

L'idée de Bayer a eu un impact initial limité, et la littérature des vingt-cinq années qui suivirent ce premier brevet [51] est assez modeste [109, 211, 349]. Par contre, un peu avant les années 2000, l'intérêt renaît et la littérature explose [14, 37–39, 43, 81, 87, 89, 90, 92, 94, 97, 113, 114, 119, 120, 132, 142, 150, 158, 160–162, 170, 171, 176–180, 186, 190, 207, 208, 231–233, 238, 240–242, 254, 262, 263, 267–269, 272, 275, 277, 278, 296, 308–312, 333, 346, 351, 355, 363, 364, 380, 386, 388]! Mais pourquoi? Simplement parce que l'évolution de la technologie rendait la photographie numérique réalisable à peu de frais, promettant ainsi des applications nouvelles et peu dispendieuses!

*
* * *

Les algorithmes de mise au point automagiques (*autofocus*) sont extrêmement mal documentés! En fait, ce n'est pas exact : comme les bons algorithmes de mise au point automatiques constituent des avantages compétitifs intéressants — voire essentiels! — entre les différents modèles et manufacturiers, la plupart sont plus ou moins des secrets industriels que l'on ne connaît qu'à travers les brevets qui les protègent [34, 146, 147, 166, 210, 219, 326, 330, 331] et quelques articles de recensement [49].

3

Espaces de Couleurs

Sommaire. Dans ce chapitre, nous nous intéresserons aux espaces de couleurs. Nous verrons qu'il existe deux grandes familles d'espaces de couleurs, les espaces linéaires et (sans surprise) les espaces non linéaires. Nous insisterons un peu plus sur les espaces de couleurs reliés à la télévision et au cinéma. Enfin, nous terminerons avec les remarques bibliographiques.

3.1 Introduction

Pourquoi avons-nous besoin d'autres espaces de couleurs que l'espace *RGB* (dont nous avons déjà discuté au ch. 1)? En partie parce que l'œil (normal) fonctionne avec quatre types de capteurs, les bâtonnets, qui sont achromatiques et réagissent à l'intensité de la lumière, et trois types de cônes qui captent chacun le rouge, le vert et le bleu, mais l'information qui est transmise au cerveau est encodée en termes de la luminosité, de la différence vert-rouge et de la différence jaune-bleu. Il tombe alors sous le sens de tenter d'exploiter ce trait physiologique et d'encoder les couleurs dans un espace qui correspond à cette sensibilité ; c'est-à-dire avec l'information principalement concentrée dans la luminosité, et avec moins d'information dans le *chroma* (c'est-à-dire la « couleur » avec sa teinte et sa pureté) auquel nous sommes beaucoup moins sensibles.

3.1.1 Des couleurs primaires

Les différents types de cônes ne répondent pas à une seule longueur d'onde de lumière spécifique, plutôt ils répondent différemment à différentes longueurs d'ondes, avec, il est vrai, une réponse plus

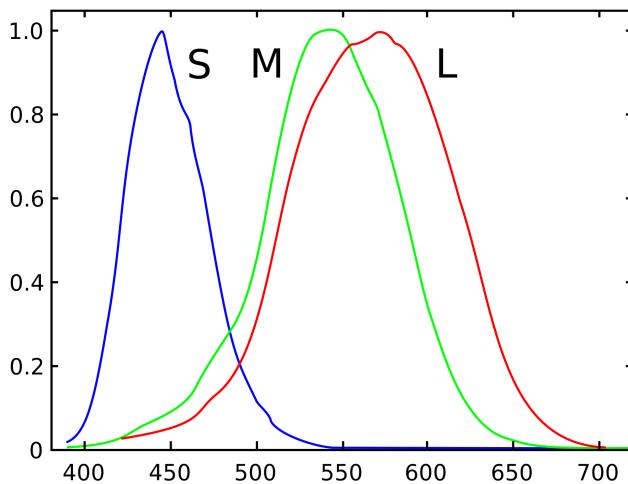


FIGURE 3.1.1 — Sensibilité de l’œil aux longueurs d’ondes. Source : Wikipedia.

forte autour de certaines longueurs d’ondes que nous associons aux couleurs rouge, vert et bleu. La fig. 3.1.1 montre les courbes de réponses normalisées.

Comme les courbes se recoupent de façon importante, cela implique qu’une longueur d’onde donnée (on parle d’une couleur monochromatique ou de « couleur spectrale ») peut stimuler plus d’un type de cône; nous avons alors ça sensation d’une couleur « complexe ». Par contre, deux (ou trois) longueurs d’ondes judicieusement choisies et mélangées avec des intensités appropriées pourraient stimuler les cônes exactement de la même façon ! On décrit toutes les distributions spectrales — tous les mélanges de longueur d’ondes — qui résultent dans la même couleur *perçue* comme des couleurs métamères (ou couleurs homochromes).

Il s’agit alors de choisir un petit nombre de longueurs d’ondes suffisantes, par métamérisme, pour produire une gamme de couleurs assez riche pour être intéressante. Comme l’œil dispose de trois types de cônes dont les réponses diffèrent significativement, choisir trois *couleurs primaires* paraît indiqué, et sans doute devraient-elles correspondre aux maximums des courbes de réponses de la fig. 3.1.1. Pourrait-on reproduire les couleurs de façon satisfaisante avec moins de couleurs primaires ?

Clairement, une seule couleur serait bien insuffisante et ne donnerait qu’une seule couleur en intensité variable, ce serait une image monochrome. Deux couleurs paraissent aussi insuffisantes. En effet, deux couleurs ne pourraient qu’exprimer des dégradés entre ces deux couleurs, par exemple, passer du rouge au bleu en générant différents tons de mauve, fuchsia, etc., ce qui serait bien étrange et dérangeant. Par contre, trois couleurs primaires permettent des combinaisons bien plus intéressantes. Si nous considérons l’espace des couleurs comme un volume en trois dimensions où une dimension correspond à la luminance perçue et les deux autres à des propriétés comme la teinte et la pureté (des caractéristiques que nous expliquerons plus en détail dans un instant), une unique couleur primaire correspond à un segment de droite à quelque part dans le volume orienté parallèlement à la luminance.

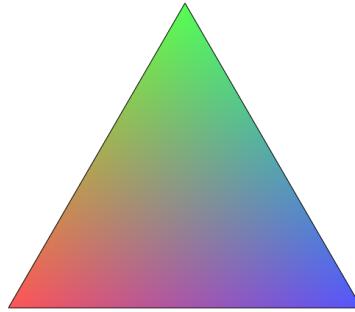


FIGURE 3.1.2 — Trois couleurs primaires sont mélangées pour donner une multitude de couleurs intermédiaires. Vers le centre, nous avons des tons de gris, et des couleurs maximalement saturées sur les rebords.

Deux couleurs primaires correspondent à un plan coupant le volume à un angle donné. Ce n'est qu'avec les trois composantes que nous pouvons remplir le volume.

Nous pouvons aussi visualiser les combinaisons des trois couleurs primaires comme un diagramme ternaire, dont une approximation visuelle est donnée à la fig. 3.1.2. Un diagramme ternaire (en anglais *ternary plot*) affiche un point de composantes originales a , b et c entre 0 et 1 inclusivement aux coordonnées

$$\alpha = \frac{a}{a+b+c},$$

$$\beta = \frac{b}{a+b+c},$$

$$\gamma = \frac{c}{a+b+c}$$

dans le diagramme ; lesquelles sont converties sur le plan par

$$\begin{bmatrix} 0 & \frac{1}{2} & -\frac{1}{2} \\ 1 & \frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$

où la matrice est composée des vecteurs colonnes qui correspondent aux orientations des côtés du triangle équilatéral dont le coin inférieur gauche est aux coordonnées $(0,0)$. Cette représentation est importante car nous la réutiliserons à la § 3.1.2.

La fig. 3.1.2 devrait nous convaincre que les combinaisons de trois couleurs primaires créent des tons variés et intéressants. Au centre, nous trouvons des tons de gris, le blanc lorsque l'intensité des couleurs est suffisante et du noir lorsque l'intensité est nulle¹. Le problème devient alors

1. Une intensité suffisante ne signifie pas nécessairement maximale ; à partir d'une certaine intensité, tous les cônes saturent et même les couleurs monochromatiques apparaissent blanches. Si l'intensité dépasse un certain seuil, les dommages à l'œil sont permanents. Nous considérerons donc que l'intensité est entre des limites « modérées » : noir est une intensité de lumière insuffisante pour stimuler les cônes, et blanc une combinaisons de couleurs suffisamment intense pour donner une *sensation* de blanc. Les intensités sont normalisées arbitrairement entre 0 et 1, ou quelques autres limites imposées par l'espace de couleurs considéré.

de judicieusement choisir ces trois couleurs primaires — pas forcément monochromatique. Mais comment les choisir puis les décrire? C'est le problème que nous explorerons à la section 3.2. Nous verrons aussi que le choix des couleurs primaires varient d'un standard d'image à l'autre, évoluant selon les technologies disponibles.

3.1.2 Familles d'espaces de couleurs

Les espaces de couleurs (ou colorimétriques) sont donc des représentations des couleurs qui permettent d'encoder les couleurs (stimuli perçus) de façon numérique, préféablement d'une manière utile pour les diverses opérations que nous ferons subir aux images.

Nous distinguons deux grandes familles d'espaces de couleur :

- Les *espaces linéaires* expriment les couleurs en termes de combinaisons linéaires des couleurs primaires. C'est donc qu'il est possible de les exprimer comme des produits matrice/vecteur; et les « traductions » d'un espace de couleurs linéaire à un autre s'expriment aussi par des matrices réversibles. C'est-à-dire que s'il existe une matrice T d'un espace A à un espace B , nous avons

$$T \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad \text{et} \quad T^{-1} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Typiquement, l'un de ces deux espaces est l'espace *RGB* pour la raison très pragmatique que c'est typiquement l'encodage des couleurs directement supporté par le matériel de rendu.

- Les *espaces non-linéaires*. Ces espaces expriment les couleurs en utilisant des transformations généralisées, habituellement pour organiser les couleurs selon des caractéristiques qui sont elles-mêmes non linéaires, telles que la luminosité perçue, la teinte, la saturation, etc. Ces transformations peuvent être complexes et difficiles à calculer, par exemple, à cause de l'emploi de fonctions logarithmiques, trigonométriques, etc. La complexité de ces espaces les rend peu attrayants pour les applications où le temps de calcul doit demeurer modeste (comme la compression d'image), mais peuvent quand même être utilisés lorsque les propriétés de ces espaces les rendent intéressants pour un traitement en particulier.

Qu'ils soient linéaires ou non, les espaces de couleurs vont tenter d'exploiter les caractéristiques du système visuel humain. En particulier, comme nous le montre la fig. 3.1.3, que si les couleurs perçues sont bien captées par les cônes et les bâtonnets dans l'espace de couleurs primaires rouge, vert et bleus en plus de la luminance, l'information envoyée au cerveau est un encodage de la luminance,

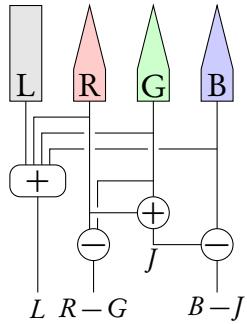


FIGURE 3.1.3 — Les cônes et les bâtonnets combinent leurs signaux pour donner la luminance, les différences rouge-vert et jaune-bleu [35].

d'une différence rouge-vert ($R-G$) et jaune-bleu ($B-J$). Nous ne sommes pas également sensibles à ces trois composantes. En effet, nous sommes beaucoup plus sensibles aux variations en luminance qu'aux variations dans les différences rouge-vert ou jaune-bleu. Cela pourra être exploité judicieusement pour coder la couleur avec moins de précision, mais sans dégradation visible.

*
* * *

Les prochaines sections exploreront ces deux grandes familles plus en détail, tout en tissant les différents liens de parenté entre les espaces de couleurs présentés.

3.2 L'espace de couleurs CIÉ 1931

Rappelons qu'à la base, la lumière n'est composée que de photons de différentes énergies¹ qui sont ensuite détectés par les capteurs de l'œil, les cônes des trois types et les bâtonnets. Ces capteurs ont des réponses différentes en fonction de l'énergie des photons. Pour modéliser la réponse de l'œil aux couleurs (qui n'existent par ailleurs que dans notre esprit), il faut donc estimer les *courbes de réponse (matching functions)* des différents capteurs de l'œil. C'est ce qu'à fait la Commission internationale de l'éclairage (ci-après, CIÉ) dans les années 1920–1930 [159, 303, 337, 377]. Les courbes obtenues par la CIÉ sont montrées à la fig. 3.2.1.

Le modèle (qui ne tient pas compte de la réponse des bâtonnets qui varie grandement en fonction de l'intensité de l'éclairage) a été ajusté sur un relativement petit nombre d'observations (dix sujets

1. Rappelons aussi que des photons de haute énergie possèdent des longueurs d'ondes plus courtes, car la fréquence f et longueur d'onde λ sont liées par la formule $f = c/\lambda$ (et donc $\lambda = c/f$), et que l'énergie contenue dans un photon de longueur d'onde λ est donnée par $E = h^c/\lambda$, où $h \approx 6.626 \times 10^{-34}$ Js est la constante de Planck.

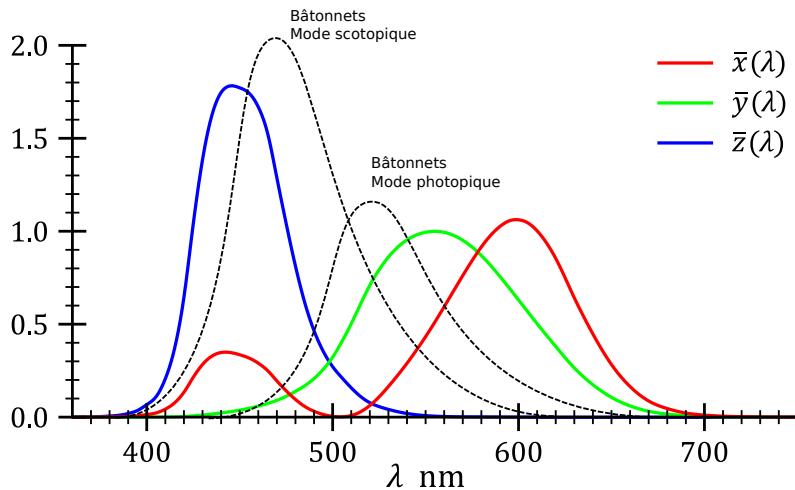


FIGURE 3.2.1 — Les fonctions colorimétriques de l’observateur standard selon le modèle CIÉ 1931. Modifié d’après Wikipedia pour les bâtonnets, échelles très approximatives.

dans l’étude de Wright [377] et sept dans celle de Guild [159]) et on doit s’attendre à des variations significatives d’un observateur à l’autre. Cependant, la réponse n’est pas entièrement subjective, elle est forcément proportionnelle, en intensité, à l’énergie contenue dans la lumière. Nous avons donc une réponse visuelle

$$V \propto \int v(\lambda)E(\lambda) \, d\lambda,$$

où $v(\lambda)$ est la sensibilité de l’œil à la longueur d’onde λ et $E(\lambda)$ l’énergie de la longueur d’onde λ [68]. Par hypothèse, si nous avons deux sources de lumière $E_1 = v(\lambda_1)E(\lambda_1) = v(\lambda_2)E(\lambda_2) = E_2$, alors elles sont de même brillance (intensité) perçue. De même, pour tous les $0 \leq \alpha \leq 1$, la mixture

$$\alpha E_1 + (1 - \alpha)E_2$$

est de brillance perçue constante (même si la couleur varie) [216].

3.2.1 De l’origine du triangle des couleurs

En mélangeant trois couleurs spectrales (monochromatiques, c'est-à-dire composées d'une seule longueur d'onde) judicieusement choisies dans des proportions calculées, nous devrions pouvoir simuler n'importe quelle couleur visible. Pour des raisons technologiques¹, ces trois couleurs judicieusement choisies par le CIÉ sont

Rouge	700 nm
Vert	546.1 nm
Bleu	435.8 nm

1. Dans les années 1920, les lampes à vapeur de mercure semblaient être un choix raisonnable, d'autant plus que trois des bandes d'émission correspondent exactement aux couleurs retenues, sauf pour le rouge, à 650 nm.

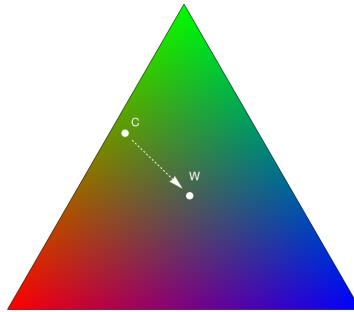


FIGURE 3.2.2 — Trois couleurs primaires sont mélangées. Vers le centre, nous avons un point perçu comme un gris neutre (W). Une couleur C près du bord est perçue comme saturée. Se déplacer vers le gris neutre produit des couleurs de moins en moins saturées.

Les couleurs que l'on peut obtenir par combinaisons linéaires avec des coefficients normalisés entre 0 et 1 inclusivement, sont contenues dans un triangle (revoir la fig. 3.1.2, p. 35, et la section 3.1.1) dont les sommets sont les couleurs primaires choisies. Considérez maintenant la fig. 3.2.2. Dans ce diagramme triangulaire, un point, pour une brillance moyenne donnée, sera perçu comme un gris neutre (c'est-à-dire sans teinte). Dans la fig. 3.2.2, c'est le point « blanc » marqué par un W (pour White). Ce point, en anglais *white point*, variera en fonction des couleurs primaires choisies. Une autre couleur (marquée par un C dans la figure) paraîtra plus riche, plus saturée, si elle se trouve près des bords de la figure. Faire glisser cette couleur vers le point blanc donne des couleurs de moins en moins saturées (mais notez que suivre la flèche en ligne droite ne préserve pas la *teinte*!).

Pour produire des couleurs en dehors du triangle de la fig. 3.2.2, il faudrait avoir recours à des couleurs qui dépassent l'intensité maximale permise (par hypothèse de construction du triangle) ou encore négatives, ce qui, bien entendu, est impossible physiquement¹, mais *mathématiquement cohérent*. Nous pourrions alors choisir trois couleurs primitives « imaginaires », X , Y , Z qui englobent non seulement le triangle de la fig. 3.2.2 mais aussi *toutes* les couleurs que nous pouvons percevoir.

Considérons maintenant la fig. 3.2.3. Nous trouvons maintenant un triangle dont les sommets sont les couleurs imaginaires X , Y et Z . Dans la figure, le triangle montre toutes les couleurs que l'on pourrait obtenir par des combinaisons des couleurs X , Y et Z . En gris, ce sont les couleurs impossibles à obtenir physiquement. En blanc, les couleurs visibles. Dans le triangle de couleur, les couleurs que l'on peut représenter comme une combinaison des trois couleurs primaires. La zone en blanc représente des couleurs visibles, mais impossibles à représenter à partir des trois couleurs primaires (si le blanc eût été coloré, il eusse dû l'être avec de fausses couleurs!). Il existe plusieurs

1. Et avant qu'un petit malin suggère d'utiliser des antiphotons, sachez que les photons, comme les bosons de jauge (autres que W^- et W^+), sont leurs propres antiparticules.

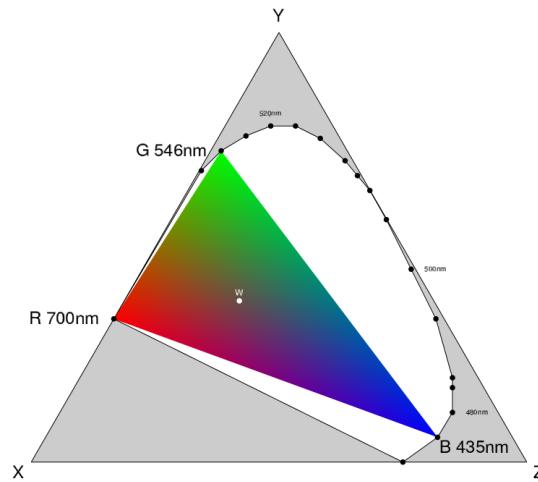


FIGURE 3.2.3 — Le triangle XYZ de la CIÉ 1931 (schéma très approximatif).

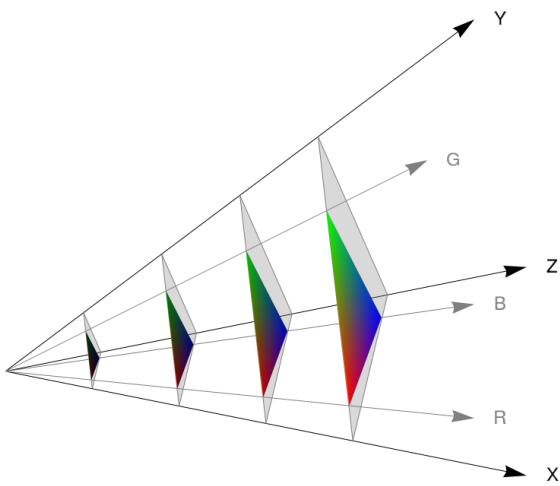


FIGURE 3.2.4 — La pyramide XYZ de la CIÉ 1931.

représentations équivalentes du triangle CIÉ, comme par exemple la fig. 1.1.2, p. 2, où le triangle n'est plus équilatéral mais rectangle et où l'axe X change de direction.

3.2.2 L'espace des couleurs CIÉ comme un espace vectoriel

Si on impose comme condition qu'aucune des couleurs (imaginaires ou réelles) qui correspondent aux sommets du triangle ne peut être une combinaison des deux autres (elles seraient colinéaires sinon), elles peuvent former un espace vectoriel à trois dimensions où chacun des vecteurs de la base correspond aux intensités d'une couleur primaire. Ainsi, une couleur est décrite de façon non ambiguë comme une coordonnée dans un espace à trois dimensions.

La fig. 3.2.4 montre différents plans formés par des couleurs de luminosité perçue égale. Les régions en gris correspondent aux triangles XYZ pour des luminosités données, y compris les couleurs impossibles à réaliser. Inclus dans chacun de ces plans, les triangles des couleurs réalisables à partir des trois couleurs primaires (réalisables aussi) R, G et B . À la pointe, la luminosité est nulle et toutes les couleurs sont confondues dans le noir. Lorsque X, Y et Z sont suffisamment grands, un plan apparaît avec un point vraiment blanc (plutôt que gris).

3.2.3 Contraintes imposées sur l'espace CIÉ 1931

Pour que l'espace de couleurs soit vraiment utile, il doit se comporter non seulement comme un espace vectoriel, mais aussi posséder des caractéristiques intuitives, par exemple :

- Pour une couleur C , varier sa luminosité déplace la couleur le long de la droite passant par l'origine et C (changer la luminosité ne change pas la teinte);
- Les couleurs spectrales (monochromatiques) sont sur le pourtour de la forme en « voile de bateau »;
- Les couleurs maximalement saturées sont aussi sur le pourtour de la « voile »;
- Les couleurs réalisables à partir des couleurs primaires sont maximamente saturées sur les rebords du triangle;
- Mélanger deux couleurs C_1 et C_2 par une combinaison linéaire crée des couleurs intermédiaires sur le segment de droite dont les extrémités sont C_1 et C_2 ;

On aimerait bien aussi que

- Si on mélange le « blanc » et une couleur on obtient une version moins saturée de la couleur ; mais ce n'est pas ce que l'on observe, car la couleur se rapproche du point blanc mais en traversant des *teintes* différentes, ce qui est bien embêtant. Dans cet espace, le chemin entre le point blanc et la couleur désirée qui conserve la teinte est *courbé*.
- Qu'il soit facile d'avoir une intuition de la couleur qui résulte de X, Y et Z , mais ce n'est pas le cas.

3.2.4 CIÉ 1931 et les autres espaces de couleurs.

Tous les autres espaces de couleurs standards que nous verrons dans ce chapitre définissent (au moins) trois couleurs primaires. Si ces couleurs primaires correspondent toutes assez bien aux

couleurs primaires rouge, vert et bleu, elles ne sont pas forcément monochromatiques. Le standard CIÉ impose le rouge monochromatique à 700nm, le vert à 546.1nm et le bleu à 435.8nm, mais d'autres standards vont utiliser des couleurs primaires complexes. Pour les définir sans ambiguïté, et avec grande précision, ils les définissent très souvent en termes des coordonnées xyz obtenues à partir des coordonnées XYZ de la façon suivante :

$$\begin{aligned}x &= \frac{X}{X + Y + Z} \\y &= \frac{Y}{X + Y + Z} \\z &= \frac{Z}{X + Y + Z} = 1 - x - y\end{aligned}$$

et donc le z peut être laissé implicite.

3.2.5 Conversions entre CIÉ XYZ 1931 et RGB .

La conversion entre XYZ et RGB CIÉ 1931 (tel que défini par les couleurs monochromatiques décrite à la § 3.2.1) pour $0 \leq r, g, b \leq 1$ est donnée [68, p. 100–101] par

$$\begin{bmatrix} 0.48698 & 0.31001 & 0.20000 \\ 0.17696 & 0.81240 & 0.01064 \\ 0 & 0.01 & 0.98999 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \quad (3.2.1)$$

et l'inverse par

$$\begin{bmatrix} 2.3647 & -0.90302 & -0.47143 \\ -0.51884 & 1.42781 & 0.08947 \\ 0.00524 & -0.01442 & 1.00921 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad (3.2.2)$$

Vous pouvez vous interroger sur les valeurs suspectement précises utilisées étant donné que les observations ont été obtenues avec assez peu de sujets; d'autres auteurs ne s'en bâorent pas et arrondissent considérablement les valeurs [118].

3.3 Les espaces de couleurs linéaires

Dans cette section, nous présentons quelques exemples d'espaces de couleurs linéaires, c'est-à-dire ceux dont la transformation de et vers RGB (montré à la fig. 3.3.1) peut s'exprimer en termes de produits matrice/vecteur. Les espaces de couleurs sont aussi présentés approximativement selon une gradation de la complexité des idées qui ont mené à leur création. Enfin, en fin de section, nous reviendrons sur ce que nous avons appris et nous présenterons une « généalogie » des espaces de couleurs linéaires qui clarifiera, nous l'espérons, les relations entre les différents standards.

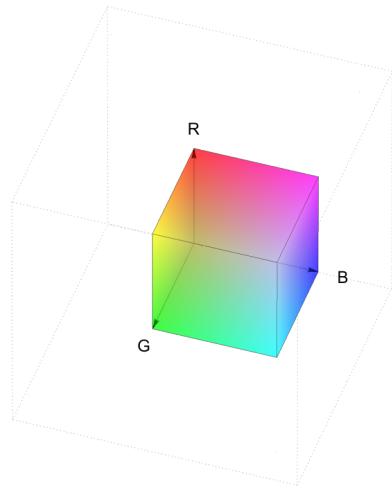


FIGURE 3.3.1 — L'espace de couleurs RGB.

3.3.1 Kodak 1

La matrice de transformation de *RGB* à Kodak-1 est donnée¹ par

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} \quad (3.3.1)$$

et de Kodak-1 à *RGB* par

$$\frac{1}{2} \begin{bmatrix} 1 & 0 & 1 \\ 0 & -1 & -1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad (3.3.2)$$

Cet espace de couleur n'est pas idéal, car k_1 est approximativement la luminance, k_2 est une différence jaune-bleu, mais k_3 est une différence rouge-cyan (plutôt que rouge-vert) comme attendu pour bien modéliser la réponse visuelle. La transformation du cube *RGB* dans l'espace de couleurs Kodak-1 est montré à la fig. 3.3.2.

Il néglige aussi de compacter les bits : si r , g et b sont des valeurs allant de 0 à 255, nous avons

$$\begin{aligned} 0 &\leq k_1 \leq 765 = 3 \times 255, \\ -510 &\leq k_2 \leq 255, \\ -510 &\leq k_3 \leq 255, \end{aligned}$$

1. J'ai pris note de cette transformation dans un cahier de laboratoire pendant mon séjour à Bell Labs en 1996. Naturellement, je n'ai pas retenu la référence de la source et il semble qu'elle soit maintenant « tombée de l'Internet », car je n'ai pu la retrouver. Mon moi du passé a appris de ses erreurs ; n'oubliez pas de noter correctement vos références !

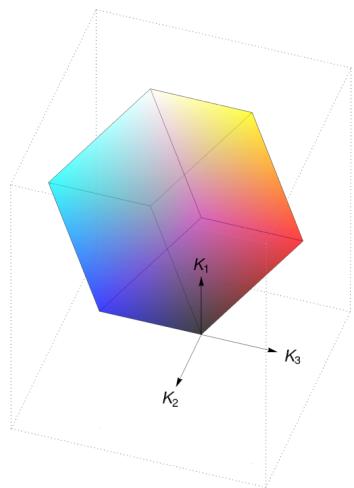


FIGURE 3.3.2 — L'espace de couleurs Kodak-1.

ce qui ne tient plus sur 8 bits par composantes, mais sur 10 bits, ce qui est bien embêtant. De plus, les coordonnées dans l'espace Kodak-1 sont assez peu intuitives : si le noir est bien à $(0,0,0)$, le blanc est à la position $(3,-1,-1)$. Cependant, il a l'avantage de se calculer en nombres entiers seulement et d'être parfaitement réversible, c'est-à-dire qu'on peut partir de RGB , aller dans Kodak-1 et revenir exactement aux mêmes valeurs RGB .

Démonstration 3.3.1. *Réversibilité de Kodak-1.* La démonstration de la réversibilité est directe :

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} r+g+b \\ -r-g+b \\ r-g-b \end{bmatrix}$$

puis

$$\begin{aligned} \frac{1}{2} \begin{bmatrix} 1 & 0 & 1 \\ 0 & -1 & -1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} r+g+b \\ -r-g+b \\ r-g-b \end{bmatrix} &= \frac{1}{2} \begin{bmatrix} (r+g+b)+(r-g-b) \\ (-r-g+b)+(r-g-b) \\ (r+g+b)+(-r-g+b) \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} 2r \\ 2g \\ 2b \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \end{aligned}$$

□

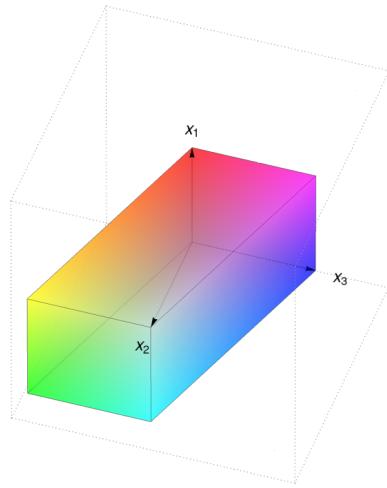


FIGURE 3.3.3 — L'espace de couleurs Kodak *Lossless*.

3.3.2 Kodak *Lossless*

L'intérêt pour les espaces de couleurs parfaitement réversibles est clair : cela permet de compresser sans perte des images. Un de ces espaces de couleurs se retrouve dans le standard JPEG sans perte [307, 356, 374]. La transformation de *RGB* vers l'espace Kodak *Lossless* (« sans perte »), montré à fig. 3.3.3, est donnée par

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (3.3.3)$$

et de Kodak *Lossless* à *RGB* par

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad (3.3.4)$$

Il est alors facile de vérifier que la transformation est exactement réversible (sur le modèle de la démonstration 3.3.1). Il existe une variante qui joue sur les propriétés de l'arithmétique entière (qu'utilisent des langages de programmation comme C et C++) et qui, pour cette raison, ne s'exprime pas bien comme un produit matrice/vecteur. Nous avons

$$\begin{aligned} x_1 &= r - g \\ x_2 &= g \\ x_3 &= b - \left\lfloor \frac{r + g}{2} \right\rfloor \end{aligned} \quad (3.3.5)$$

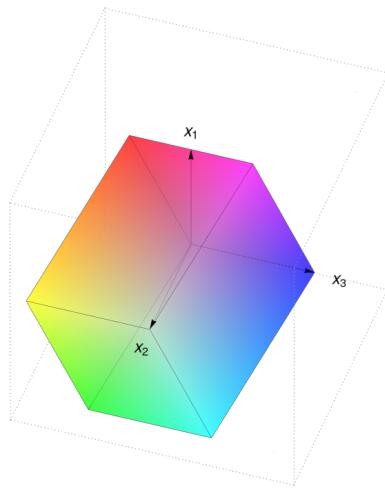


FIGURE 3.3.4 — L'espace de couleurs Kodak *Lossless*, seconde variante.

et l'inverse est donné par

$$\begin{aligned} r &= x_1 + x_2 \\ g &= x_2 \\ b &= x_3 + \left\lfloor \frac{x_1 + 2x_2}{2} \right\rfloor = x_3 + \left\lfloor \frac{x_1}{2} \right\rfloor + x_2 \end{aligned} \tag{3.3.6}$$

La transformation du cube *RGB* qui résulte de cette seconde transformation est montrée à la fig. 3.3.4. L'une comme l'autre donne des coefficients tels que

$$\begin{aligned} -255 &\leq x_1 \leq 255, \\ 0 &\leq x_2 \leq 255, \\ -255 &\leq x_3 \leq 255, \end{aligned}$$

ce qui ne préserve pas le nombre original de bits.

3.3.3 Kodak YCC

Les transformées précédentes donnent une pondération égale à chaque composante de couleur. Cependant, comme nous le montre la fig. 3.2.1, p. 38, l'œil n'y est pas sensible également. Il convient donc d'inclure ces pondérations dans l'espace de couleurs [13, 21, 116]. Celles-ci sont

$$\begin{aligned} \omega_r &= 0.299, \\ \omega_g &= 0.587, \\ \omega_b &= 0.114. \end{aligned} \tag{3.3.7}$$

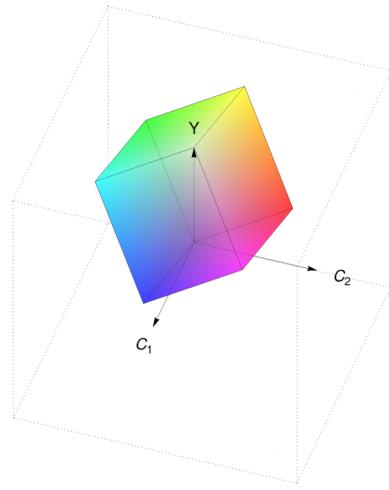


FIGURE 3.3.5 — L'espace de couleurs Kodak YCC.

Il paraît alors raisonnable d'inclure ces sensibilités dans l'espace de couleurs, et si possible d'une façon « équilibrée », c'est-à-dire de telle façon que les coordonnées dans l'espace de couleurs soient aussi centrées sur zéro que possible.

L'espace Kodak *YCC*, montré à la fig. 3.3.5, est un espace simple qui fait usage de ces pondérations [7, 270, 365]. Strictement, l'espace Kodak *YCC* n'est pas linéaire : les composantes *RGB* sont transformées par une espèce de correction gamma. Pour la composante *c*, on a

$$c' = \begin{cases} 1.099c^{0.45} - 0.099 & \text{si } c \geq 0.018 \\ 4.5c & \text{sinon.} \end{cases} \quad (3.3.8)$$

Une fois les composantes *r*, *g* et *b* « corrigées » en *r'*, *g'* et *b'*, la transformation est donnée par

$$\begin{bmatrix} \omega_r & \omega_g & \omega_b \\ -\omega_r & -\omega_g & 1-\omega_b \\ 1-\omega_r & -\omega_g & -\omega_b \end{bmatrix} \begin{bmatrix} r' \\ g' \\ b' \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.299 & -0.587 & 0.886 \\ 0.701 & -0.587 & -0.114 \end{bmatrix} \begin{bmatrix} r' \\ g' \\ b' \end{bmatrix} = \begin{bmatrix} Y \\ C_1 \\ C_2 \end{bmatrix} \quad (3.3.9)$$

et l'inverse est donné par

$$\begin{bmatrix} \omega_r & \omega_g & \omega_b \\ -\omega_r & -\omega_g & 1-\omega_b \\ 1-\omega_r & -\omega_g & -\omega_b \end{bmatrix}^{-1} \begin{bmatrix} Y \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & -\frac{\omega_b}{\omega_g} & -\frac{\omega_r}{\omega_g} \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} Y \\ C_1 \\ C_2 \end{bmatrix}$$

$$\approx \begin{bmatrix} 1 & 0 & 1 \\ 1 & -0.194 & -0.509 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} Y \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} r' \\ g' \\ b' \end{bmatrix}. \quad (3.3.10)$$

Cet espace de couleur utilise non seulement les pondérations pour la sensibilité aux couleurs, mais

aussi tente de maintenir l'espace balancé, c'est-à-dire approximativement centré sur l'axe Y . Ainsi, nous avons $1 - \omega_r$, qui s'oppose à $-\omega_g - \omega_b$ car $1 - \omega_r = \omega_g + \omega_b$, comme nous avons $1 - \omega_b$ qui s'oppose à $-\omega_r - \omega_g$, pour la même raison.

3.3.4 Xerox YES

Cet espace de couleurs spécifie des couleurs primaires particulières en terme de leurs coordonnées CIÉ ($x, y, z = 1$) [8, 203],

$$\begin{array}{ll} R & x = 0.630 \quad y = 0.340 \\ G & x = 0.310 \quad y = 0.595 \\ B & x = 0.155 \quad y = 0.070 \end{array}$$

probablement pour « bien jouer » avec le matériel de reproduction d'image de Xerox. La transformation de RGB (avec leurs couleurs primaires spéciales) à YES est donnée par

$$\begin{bmatrix} 0.253 & 0.684 & 0.063 \\ \frac{1}{2} & -\frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} r_x \\ g_x \\ b_x \end{bmatrix} = \begin{bmatrix} Y \\ E \\ S \end{bmatrix} \quad (3.3.11)$$

tandis que l'inverse est donné par

$$\begin{bmatrix} 1 & 1.431 & 0.126 \\ 1 & -0.569 & 0.126 \\ 1 & 0.431 & -1.874 \end{bmatrix} \begin{bmatrix} Y \\ E \\ S \end{bmatrix} = \begin{bmatrix} r_x \\ g_x \\ b_x \end{bmatrix}. \quad (3.3.12)$$

L'espace de couleurs YES est montré à la fig. 3.3.6.

3.3.5 Espaces Ohta

Alors que la plupart des autres espaces de couleurs veulent modéliser, au moins approximativement, la réponse de l'œil aux couleurs, les espaces d'Ohta (ce n'est pas un acronyme mais un patronyme) proposent une approche radicalement différente. L'idée d'Ohta est que l'espace de couleurs devrait supporter efficacement la distribution des couleurs dans l'image et non la réponse de l'œil. Pour ce faire, nous trouvons les axes principaux du nuage de points formé par les couleurs de l'image grâce au calcul des composantes principales (voir § 3.5). Ces axes sont orthogonaux entre eux, de longueur proportionnelle à la variance du nuage de point dans la direction qui leur correspond et la fig. 3.3.7 montre un exemple en 2 dimensions.

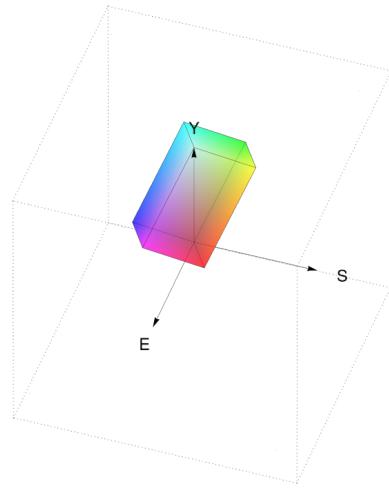


FIGURE 3.3.6 — L'espace de couleurs Xerox YES.

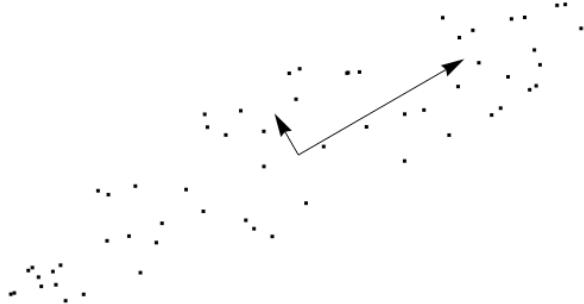


FIGURE 3.3.7 — Les axes principaux d'un patatoïde.

Or, plutôt que de trouver ces vecteurs optimaux pour chaque nouvelle image (ce qui s'avère pas très compliqué, mais pas trivial non plus [198]), Ohta propose simplement de choisir des vecteurs de base qui sont, en général, à peu près dans les bonnes directions. Il propose les deux espaces de couleurs suivants [282, 283]. Le premier espace qu'il propose, nommons-le Ohta-1, est donné par

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \pm\frac{1}{2} & 0 & \mp\frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} \quad (3.3.13)$$

avec son inverse (considérant la seconde rangée $(\frac{1}{2}, 0, -\frac{1}{2})$), car il propose d'utiliser « l'une ou l'autre » combinaisons de signes indifféremment [283, p. 228]) :

$$\begin{bmatrix} 1 & 1 & -\frac{2}{3} \\ 1 & 0 & \frac{4}{3} \\ 1 & -1 & -\frac{2}{3} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \quad (3.3.14)$$

Il prétend par ailleurs que cet espace de couleurs remplace les axes calculés exactement pour

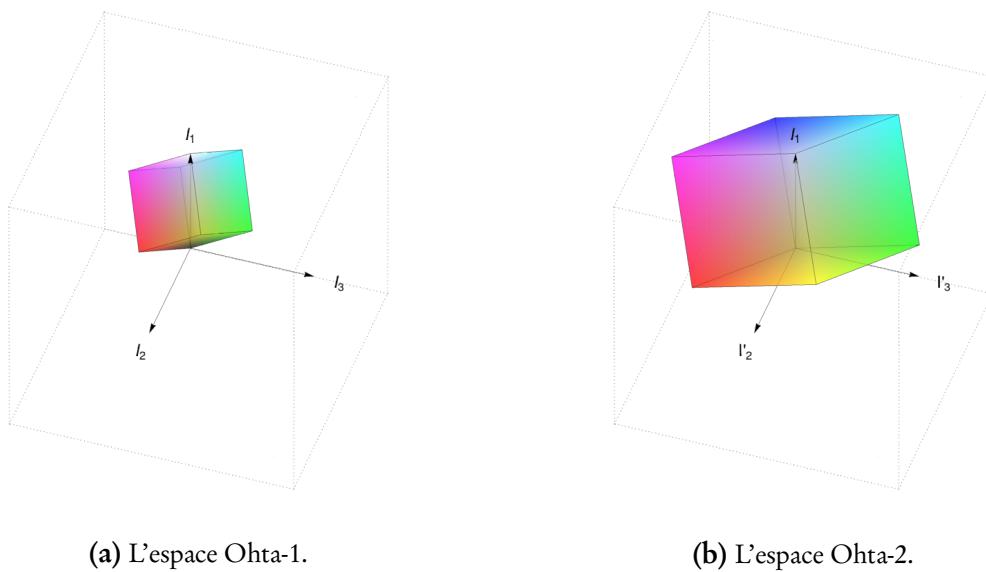


FIGURE 3.3.8 — Les espaces de couleurs Ohta.

le nuage de point « sans dégradation » [282, p. 28], mais sans véritablement offrir d’arguments quantifiés¹. Le second espace de couleurs qu’il propose, Ohta-2, utilise la transformation suivante :

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 1 & 0 & -1 \\ -\frac{1}{2} & 1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} I_1 \\ I'_2 \\ I'_3 \end{bmatrix} \quad (3.3.15)$$

où vous aurez compris que c’est essentiellement une mise à l’échelle de deux rangées de la matrice de transformation pour le premier espace. L’inverse est donné par

$$\begin{bmatrix} 1 & \frac{1}{2} & -\frac{1}{3} \\ 1 & 0 & \frac{2}{3} \\ 1 & -\frac{1}{2} & -\frac{1}{3} \end{bmatrix} \begin{bmatrix} I_1 \\ I'_2 \\ I'_3 \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \quad (3.3.16)$$

Enfin, les cubes *RGB* transformés dans les deux espaces d’Ohta sont montrés à la fig. 3.3.8.

3.3.6 YC_oC_g

Proposé par Malvar *et al.*, l’espace de couleur YC_oC_g répond à deux besoins distincts : d’une part, un calcul extrêmement rapide de la transformation de YC_oC_g à *RGB*, d’autre part une possibilité de rendre la conversion de *RGB* à YC_oC_g puis de nouveau à *RGB* sans perte [255–257, 348].

1. Mais pour être rigoureux, et obtenir les vraies composantes principales, il faudrait aussi appliquer un biais sur les couleurs, c’est-à-dire leur soustraire la couleur moyenne. Ohta triche encore un peu en plaçant ce « centre » à (0,0,0).

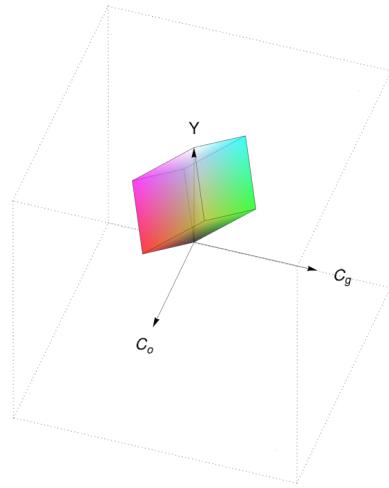


FIGURE 3.3.9 — L'espace de couleurs YC_oC_g .

Contrairement aux espaces Ohta qui donnent une pondération égale aux composantes rouges, vertes et bleues, l'espace YC_oC_g donne prépondérance au vert :

$$\begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} Y \\ C_o \\ C_g \end{bmatrix}. \quad (3.3.17)$$

La transformée inverse, comme promis, est particulièrement simple :

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} Y \\ C_o \\ C_g \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \quad (3.3.18)$$

L'espace de couleurs YC_oC_g est montré à la fig. 3.3.9. Curiosité, les indices o et g sont censés représenter orange et vert, respectivement, mais C_o n'est pas une « différence orange », mais une différence rouge-bleue ; C_g est la différence vert-magenta.

Par contre, la transformation de RGB à YC_oC_g n'est pas immédiatement parfaitement réversible, car il faut garder deux bits après le point (à cause des divisions par 4). Sachant que pour une matrice A et une constante $k \neq 0$, nous avons que

$$(kA)^{-1} = A^{-1}k^{-1} = \frac{1}{k}A^{-1},$$

il nous est possible de mettre les deux matrices (transformée et inverse) à l'échelle pour obtenir de l'arithmétique en nombres entiers seulement. Certes :

$$4 \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & -2 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} Y' \\ C'_o \\ C'_g \end{bmatrix}. \quad (3.3.19)$$

et l'inverse est donné par

$$\frac{1}{4} \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} Y' \\ C'_o \\ C'_g \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \quad (3.3.20)$$

Ce qui nous donne bien une transformation parfaitement réversible (mais dont la bande dynamique est augmentée à 10 bits par composante).

Démonstration 3.3.2.

Réversibilité de YC_oC_g mis à l'échelle. Encore une fois, la démonstration est directe :

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & -2 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} r+2g+b \\ 2(r-b) \\ -r+2g-b \end{bmatrix},$$

et

$$\begin{aligned} & \frac{1}{4} \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} r+2g+b \\ 2(r-b) \\ -r+2g-b \end{bmatrix} \\ &= \frac{1}{4} \begin{bmatrix} (r+2g+b)+2(r-b)-(-r+2g-b) \\ (r+2g+b)+(-r+2g-b) \\ (r+2g+b)-2(r-b)-(-r+2g-b) \end{bmatrix} \\ &= \frac{1}{4} \begin{bmatrix} 4r \\ 4g \\ 4b \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \end{aligned}$$

□

Enfin, Malvar *et al.* proposent une version réversible par « lifting », essentiellement une astuce qui rétablit la parité du résultat sans avoir recours à des bits après point. Pour deux entiers x et y , posons

$$d = x - y,$$

$$m = y + \left\lfloor \frac{d}{2} \right\rfloor.$$

Nous pouvons retrouver exactement x et y car

$$y = m - \left\lfloor \frac{d}{2} \right\rfloor,$$

$$x = y + d.$$

En appliquant cette astuce :

$$\begin{aligned} C_o &= r - b, \\ t &= b + \left\lfloor \frac{C_o}{2} \right\rfloor, \\ C_g &= g - t, \\ Y &= t + \left\lfloor \frac{C_g}{2} \right\rfloor, \end{aligned}$$

ce qui nous permet de retrouver nos composantes *RGB* originales :

$$\begin{aligned} t &= y - \left\lfloor \frac{C_g}{2} \right\rfloor, \\ g &= C_g + t, \\ b &= t - \left\lfloor \frac{C_o}{2} \right\rfloor, \\ r &= b + C_o. \end{aligned}$$

3.3.7 JPEG 2000 : *YVU-R*

JPEG 2000 est un standard développé pour remplacer JPEG. Cependant, pour toutes sortes de raisons, et malgré que ce soit un meilleur standard de compression, il n'a pas su s'imposer. Il propose d'utiliser les espaces de couleurs YC_bC_r (dont nous discuterons plus loin) et $YVU-R$, une transformation exactement réversible (d'où le $-R$). Cette transformée est presque linéaire, car, comme YC_oC_g , elle exploite les propriétés de l'arithmétique entière.

La transformation de *RGB* à *YVU-R* est donnée par [335, p. 40] :

$$\begin{aligned} Y_r &= \left\lfloor \frac{r + 4g + b}{4} \right\rfloor, \\ V_r &= r - g, \\ U_r &= b - g, \end{aligned} \tag{3.3.21}$$

et de *YVU-R* à *RGB* par

$$\begin{aligned} g &= Y_r - \left\lfloor \frac{V_r + U_r}{4} \right\rfloor, \\ r &= V_r + g, \\ b &= U_r + g. \end{aligned} \tag{3.3.22}$$

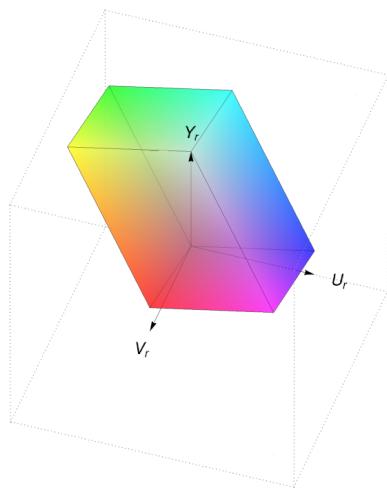


FIGURE 3.3.10 — L'espace de couleurs $YVU\text{-}R$.

Cette transformée est réversible mais demande un bit de plus pour les composantes V_r et U_r . L'espace $YVU\text{-}R$ est montré à la fig. 3.3.10.

Démonstration 3.3.3. *Réversibilité de $YVU\text{-}R$.* Il suffit de montrer que nous récupérons la composante g :

$$\begin{aligned}
 g &= Y_r - \left\lfloor \frac{V_r + U_r}{4} \right\rfloor \\
 &= \left\lfloor \frac{r + 2g + b}{4} \right\rfloor - \left\lfloor \frac{(r - g) + (b - g)}{4} \right\rfloor \\
 &= \left\lfloor \frac{r + 2g + b}{4} \right\rfloor - \left\lfloor \frac{r - 2g + b}{4} \right\rfloor \\
 &= \left\lfloor \frac{r + 2g + b}{4} \right\rfloor + \left\lfloor \frac{-r + 2g - b}{4} \right\rfloor \\
 &= \left\lfloor \frac{4g}{4} \right\rfloor \\
 &= g.
 \end{aligned}$$

□

3.3.8 Série S

Notes : Les espaces S viennent de mes notes préparatoires pour le doctorat, développés pour explorer le problème des espaces de couleurs linéaires. Même s'ils ne sont pas « standards », ils serviront ici à comprendre des manipulations nécessaires à la construction d'autres espaces de couleurs.

Parmi les propriétés désirables pour un espace de couleur, on trouve qu'une des composantes devrait correspondre directement à la luminosité de la couleur, et en particulier aux tons de gris lorsque les deux autres composantes sont nulles — cela permet, par exemple, de créer une image en noir et blanc à partir d'une seule composante de l'espace de couleur. Préférablement, les deux autres composantes devraient contenir moins d'information et correspondre, au moins vaguement, aux différences rouge-vert et jaune-bleu ; la teinte et la saturation n'étant vraiment réalisables qu'avec des transformations non-linéaires.

Il est alors possible de prendre le cube *RGB*, lui appliquer une rotation adéquate pour ramener la grande diagonale du cube — à laquelle correspondent les tons de gris — sur un des axes, disons s_{11} ; et peut-être lui appliquer une rotation supplémentaire pour qu'un des axes correspondent à la différence jaune-bleu ou rouge-vert. Mais pour ce faire, nous aurons besoin des matrices de rotations en trois dimensions (que nous donnons ici sans démonstration). La matrice de rotation en sens *antihoraire*¹ autour de l'axe des R est donnée par

$$R_r(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha \\ 0 & -\sin \alpha & \cos \alpha \end{bmatrix}, \quad (3.3.23)$$

celle autour de l'axe des G par

$$R_g(\alpha) = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{bmatrix}, \quad (3.3.24)$$

tandis que celle autour de l'axe des b est donnée par

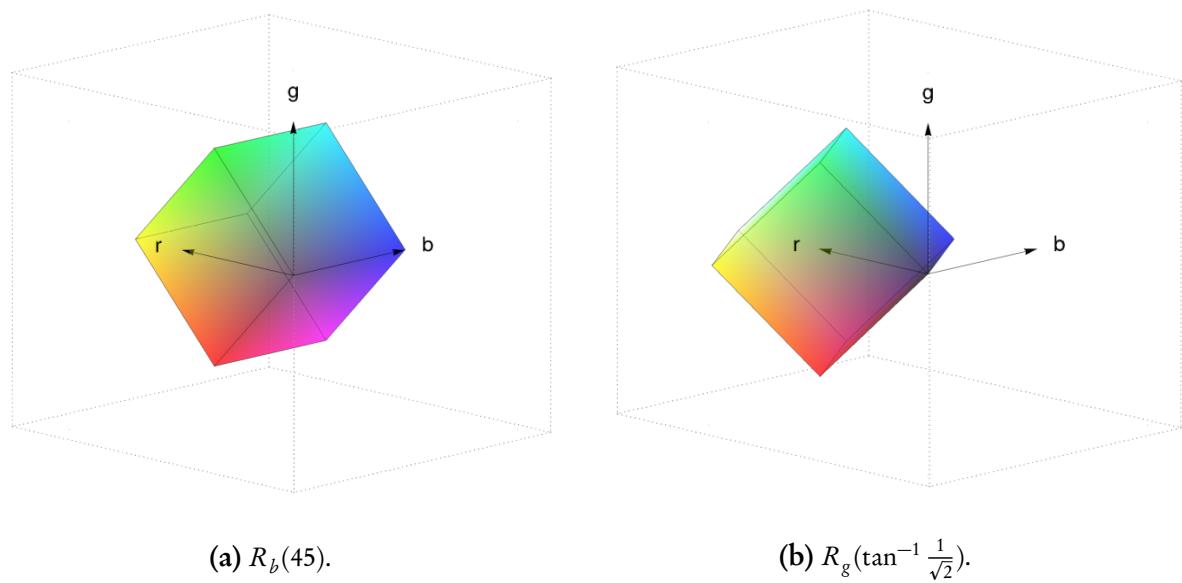
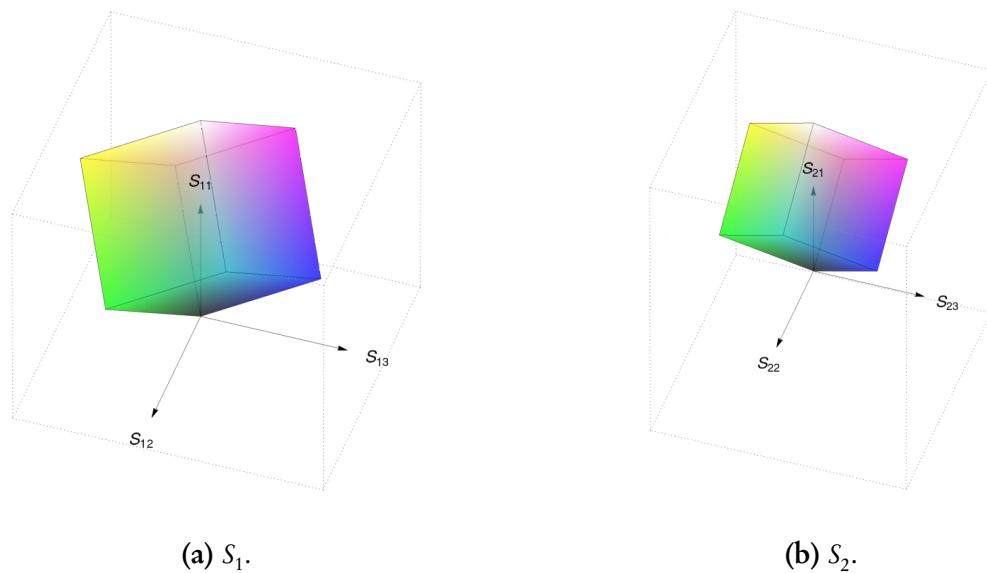
$$R_b(\alpha) = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3.3.25)$$

Utilisons donc ces rotations pour ramener la grande diagonale du cube *RGB* sur l'un des axes de notre espace — le premier. Cette rotation composée est donnée par

$$S_1 = R_g\left(\tan^{-1}\frac{1}{\sqrt{2}}\right)R_b(45^\circ). \quad (3.3.26)$$

Cette transformation applique d'abord une rotation au cube *RGB* sur l'axe B de façon à amener $(1, 1, 0)$ (rouge et vert maximum) à $(\sqrt{2}, 0, 0)$ (voir fig. 3.3.11 (a)), puis une rotation sur l'axe des G pour ramener le coin $(\sqrt{2}, 0, 1)$ à $(\sqrt{3}, 0, 0)$ (voir fig. 3.3.11 (b)). Le résultat final, en plaçant l'axe S_{11} à

1. C'est-à-dire le « bon » sens !

**FIGURE 3.3.11** — La construction de l'espace S_1 .**FIGURE 3.3.12** — Les espaces de couleurs S_1 et S_2 .

la verticale, est montré à la fig. 3.3.12. La matrice est donc

$$S_1 = R_g(\tan^{-1} \frac{1}{\sqrt{2}})R_b(45^\circ) = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & \sqrt{\frac{2}{3}} \end{bmatrix} \quad (3.3.27)$$

donc

$$\begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & \sqrt{\frac{2}{3}} \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} s_{11} \\ s_{12} \\ s_{13} \end{bmatrix} \quad (3.3.28)$$

Une propriété intéressante des matrices de rotations, c'est qu'elles sont orthonormales; et une composition de matrices de rotation est une matrice de rotation. La matrice de transformation de l'espace S_1 est donc orthonormale, et nous avons $S_1^{-1} = S_1^T$. La conversion de S_1 à RGB est donc

$$\begin{bmatrix} \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & \sqrt{\frac{2}{3}} \end{bmatrix} \begin{bmatrix} s_{11} \\ s_{12} \\ s_{13} \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \quad (3.3.29)$$

Le premier axe, S_{11} correspond à la luminosité et si S_{13} correspond à la différence jaune-bleu, S_{12} correspond à la différence aquamarine-fuchsia, une caractéristique très quelconque. On pourrait vouloir (pour une raison ou une autre) une orientation différente, par exemple avoir un axe qui correspond à la différence rouge-cyan et un autre à la différence mauve-lime. Il faudrait alors appliquer une rotation autour de S_{11} de 30° . La transformation serait alors

$$S_2 = R_r(30^\circ)R_g(\tan^{-1}\frac{1}{\sqrt{2}})R_b(45^\circ) = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ -\sqrt{\frac{2}{3}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \quad (3.3.30)$$

et donc

$$\begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ -\sqrt{\frac{2}{3}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} s_{21} \\ s_{22} \\ s_{23} \end{bmatrix}. \quad (3.3.31)$$

Puisque S_2 est orthonormale, $S_2^{-1} = S_2^T$, et

$$\begin{bmatrix} \frac{1}{\sqrt{3}} & -\sqrt{\frac{2}{3}} & 0 \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} s_{21} \\ s_{22} \\ s_{23} \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \quad (3.3.32)$$

L'espace de couleur S_2 est montré à la fig. 3.3.12 (b).

Nous avons conçu deux espaces de couleurs qui ont des propriétés généralement désirables pour les espaces de couleurs : la grande diagonale du cube RGB est aligné avec un des axes de l'espace de couleurs et les autres dimensions correspondent, au moins partiellement, aux caractéristiques de

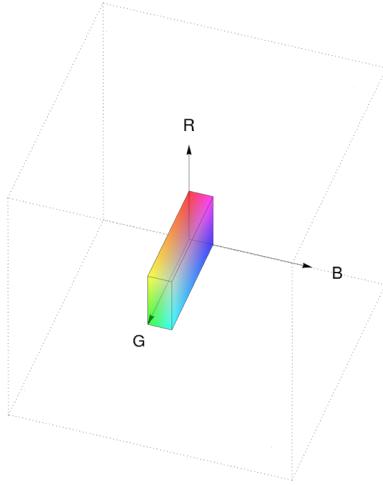


FIGURE 3.3.13 — Le cube *RGB* « compressé » par les poids ω_r , ω_g et ω_b .

l’œil. Cependant, les valeurs maximales ne sont pas bornées de façon intéressante. Par exemple, pour S_1 nous avons

$$\begin{aligned} 0 &\leq s_{11} \leq \sqrt{3}, \\ -\frac{1}{\sqrt{2}} &\leq s_{12} \leq \frac{1}{\sqrt{2}}, \\ -\sqrt{\frac{2}{3}} &\leq s_{13} \leq \sqrt{\frac{2}{3}}, \end{aligned}$$

ce qui est numériquement quelconque. De plus, ni S_1 ni S_2 ne tiennent compte de la sensibilité de l’œil aux différentes couleurs primaires. L’éq. (3.3.7), p. 46, donne les poids ω_r , ω_g et ω_b aux trois couleurs primaires *RGB*. La fig. 3.3.13 montre l’importance relative des couleurs primaires en utilisant le cube *RGB* « compressé » par les poids ω_r , ω_g et ω_b .

Cet espace maintenant mis à l’échelle ne correspond pas encore tout à fait à nos besoins : il faut encore amener la grande diagonale du parallélépipède rectangle (ou « brique ») sur un des axes de notre espace. Nous n’avons plus un cube de dimension $1 \times 1 \times 1$ mais $\omega_r \times \omega_g \times \omega_b$, ce qui complique un peu les choses. Posons donc

$$S_3 = \frac{1}{\|(\omega_r, \omega_g, \omega_b)\|} R_g \left(\tan^{-1} \frac{\omega_b}{\|(\omega_g, \omega_r)\|} \right) R_b \left(\tan^{-1} \frac{\omega_g}{\omega_r} \right) \begin{bmatrix} \omega_r & 0 & 0 \\ 0 & \omega_g & 0 \\ 0 & 0 & \omega_b \end{bmatrix}, \quad (3.3.33)$$

où $\|(\omega_g, \omega_b)\|$ est la longueur de la diagonale de la face vert-rouge du parallélépipède et $\|(\omega_r, \omega_g, \omega_b)\|$ est la longueur de la grande diagonale du parallélépipède. L’espace de couleurs

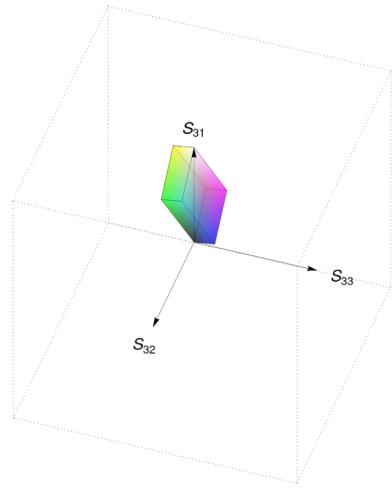


FIGURE 3.3.14 — L'espace de couleurs S_3 .

S_3 est montré à fig. 3.3.14. Cela nous donne les deux transformations

$$S_3 \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} 0.200 & 0.771 & 0.029 \\ -0.399 & 0.399 & 0 \\ -0.035 & -0.133 & 0.168 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} s_{31} \\ s_{32} \\ s_{33} \end{bmatrix} \quad (3.3.34)$$

et

$$S_3^{-1} \begin{bmatrix} s_{31} \\ s_{32} \\ s_{33} \end{bmatrix} = \begin{bmatrix} 1 & -1.990 & -0.173 \\ 1 & 0.516 & -0.173 \\ 1 & -0.006 & 5.780 \end{bmatrix} \begin{bmatrix} s_{31} \\ s_{32} \\ s_{33} \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \quad (3.3.35)$$

Nous avons donc maintenant un espace de couleurs où la luminosité est normalisée (qui varie de 0 à 1), mais où les deux autres composantes sont quelconques ($-0.399 \leq s_{32} \leq 0.399$ et $-0.168 \leq s_{33} \leq 0.168$).

*
* * *

Si pour une raison technique (par exemple, liée à la nature du canal de transmission ou de l'algorithme de traitement d'images) nous voudrions plutôt avoir un espace de couleurs qui « remplit » notre canal, nous devrions normaliser notre espace de façon à ce que les coordonnées occupent une boîte d'encombrement (*bounding box*) donnée. Prenons par exemple l'espace S_1 . Nous

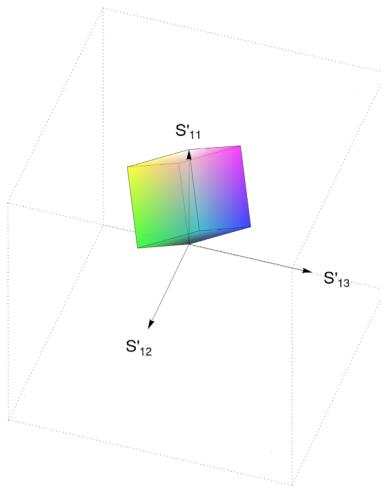


FIGURE 3.3.15 — L'espace de couleurs S'_1 .

avions remarqué que les coordonnées étaient

$$\begin{aligned} 0 &\leq s_{11} \leq \sqrt{3}, \\ -\frac{1}{\sqrt{2}} &\leq s_{12} \leq \frac{1}{\sqrt{2}}, \\ -\sqrt{\frac{2}{3}} &\leq s_{13} \leq \sqrt{\frac{2}{3}}. \end{aligned}$$

Pour normaliser S_1 de façon à ce que la boîte occupée varie de 0 à 1 pour s_{11} , et de $-\frac{1}{2}$ à $\frac{1}{2}$ pour les deux autres composantes, il suffit de poser

$$S'_1 \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{2\sqrt{\frac{2}{3}}} \end{bmatrix} S_1 \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{4} & -\frac{1}{4} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} s'_{11} \\ s'_{12} \\ s'_{13} \end{bmatrix} \quad (3.3.36)$$

Nous remarquons avec un certain bonheur que l'espace S'_1 normalisé a une matrice de transformation somme toute bien sympathique. La première matrice divise par les grandeurs désirées pour obtenir des portées voulues. Le premier coefficient doit s'assurer de normaliser la première composante sur 0 à 1, on divise donc par $\sqrt{3}$. Pour la seconde, la portée désirée est $-\frac{1}{2}$ à $\frac{1}{2}$, il faut donc diviser par deux fois le maximum (et $\frac{1}{2\frac{1}{\sqrt{2}}} = \frac{1}{\sqrt{2}}$). Enfin pour le troisième, on divise par $2\sqrt{\frac{2}{3}}$. Par un heureux concours de circonstance, tout cela se simplifie plutôt bien et donne une matrice de transformation simple.

L'inverse aussi est assez simple et est donnée par :

$$S_1'^{-1} \begin{bmatrix} s'_{11} \\ s'_{12} \\ s'_{13} \end{bmatrix} = \begin{bmatrix} 1 & -1 & -\frac{2}{3} \\ 1 & 1 & -\frac{2}{3} \\ 1 & 0 & \frac{4}{3} \end{bmatrix} \begin{bmatrix} s'_{11} \\ s'_{12} \\ s'_{13} \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \quad (3.3.37)$$

*

* * *

La présentation de la série S nous a permis de faire le tour de la boîte à outils de création des espaces de couleurs linéaires : rotations, mise à l'échelle, normalisation. Dans ce qui suit, nous rencontrerons d'autres espaces de couleurs qui s'expliquent fort bien par la combinaisons de ces transformations de base. Il est, à notre avis, bien plus intéressant de comprendre comment un espace de couleurs est construit que de simplement connaître ses deux matrices de transformation.

3.3.9 Les espaces de couleurs de la télévision

Dans cette section, nous allons jeter un œil aux espaces de couleurs issues de la télévision, autant analogique (au temps des dinosaures) que numérique.

3.3.9.1 *Transmission primaries*

Très tôt dans l'histoire de la télévision (qui commence aussi loin que les années 1920 [123]) on a arrêté le choix des couleurs primaires [187]. Le NTSC (*National Television System Committee*, l'organisme de standardisation formé en 1941) a proposé en 1953 un standard pour la télévision en couleur et définissait les trois couleurs primaires comme étant

Rouge R_{470}	$x = 0.67$	$y = 0.33$
Vert G_{470}	$x = 0.21$	$y = 0.71$
Bleu B_{470}	$x = 0.14$	$y = 0.08$
Blanc C	$x = 0.31$	$y = 0.316$

en coordonnées xyz ($z = 1$) du CIÉ 1931. Cette proposition est plus tard devenue la recommandation BT.470 du ITU [13]. Le blanc est le blanc de l'« illuminant C », une source de lumière censée correspondre à la lumière du jour, à 6774°K [13, 75, 84]. Ils établissent aussi les sensibilités relatives



FIGURE 3.3.16 — Une image télévision couleur décodée par un poste noir et blanc.

aux couleurs primaires, $\omega_r = 0.299$, $\omega_g = 0.587$, $\omega_b = 0.114$, déjà mentionnées à la section 3.3.3, p. 46.

Les couleurs des pixels sont représentées comme des combinaisons linéaires des couleurs primaires qu'il faut transmettre. Dans le cas particulier de la télévision couleur, il était nécessaire d'assurer la compatibilité avec la télévision noir et blanc, alors la seule télévision. Il nous faut donc transmettre les couleurs de façon à ce que le signal qui contient l'information de couleur puisse être ignoré par les appareils déjà en place, et rendre une image noir et blanc potable. Il faut donc transformer l'espace de couleurs *RGB* (avec les couleurs primaires ci-dessus) en un espace de couleurs où la luminance (l'image en noir et blanc) est séparée du signal de couleur. Ainsi, un poste en couleur affichera une image couleur après avoir décodé les trois canaux, tandis qu'un poste en noir et blanc ne faisant pas la distinction des canaux, décodera une image satisfaisante, mais bruitée. En effet, le signal de couleur, codé à plus faible résolution, était poussé dans les très hautes fréquences du signal de l'image, à basse amplitude. Cela avait pour effet de produire une image en noir et blanc avec un bruit en damier (voir fig. 3.3.16). Le détail de l'encodage dépasse le cadre du présent ouvrage, mais pour en savoir plus, consultez, par exemple, [102, 105, 116, 148, 157].

Il faut donc transformer les couleurs primaires en « couleurs primaires de transmission » BT.470, en *transmission primaires*. Ces *transmission primaires* sont données par

$$\begin{aligned} Y &= \omega_r r + \omega_g g + \omega_b b, \\ P_2 &= R - Y = (1 - \omega_r)r - \omega_g g - \omega_b b, \\ P_3 &= B - Y = -\omega_r r - \omega_g g + (1 - \omega_b)b, \end{aligned} \tag{3.3.38}$$

soit donc

$$T_{p_{470}} \begin{bmatrix} r_{470} \\ g_{470} \\ b_{470} \end{bmatrix} = \begin{bmatrix} \omega_r & \omega_g & \omega_b \\ 1 - \omega_r & -\omega_g & -\omega_b \\ -\omega_r & -\omega_g & 1 - \omega_b \end{bmatrix} \begin{bmatrix} r_{470} \\ g_{470} \\ b_{470} \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.701 & -0.587 & -0.114 \\ -0.299 & -0.587 & 0.886 \end{bmatrix} \begin{bmatrix} r_{470} \\ g_{470} \\ b_{470} \end{bmatrix} = \begin{bmatrix} Y \\ P_2 \\ P_3 \end{bmatrix}. \quad (3.3.39)$$

La transformée inverse, qui nous ramène à l'espace *RGB* n'est pas difficile à calculer :

$$T_p^{-1} \begin{bmatrix} Y \\ P_2 \\ P_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & -0.509 & -0.194 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} Y \\ P_2 \\ P_3 \end{bmatrix} = \begin{bmatrix} r_{470} \\ g_{470} \\ b_{470} \end{bmatrix}. \quad (3.3.40)$$

Par la suite, d'autres transformations seront appliquées pour accommoder les différentes stratégies de transmission et l'évolution des standards.

3.3.9.2 *YUV* et *YIQ*

Les *transmission primaries* sont d'abord normalisées pour permettre un encodage ultérieur par un signal en quadrature de phase [18, 108]. Les *transmission primaires* sont d'abord transformées de *RGB* (avec les couleurs primaires BT.470) vers l'espace de couleurs *YUV*.

Nous définissons¹

$$\begin{aligned} U_{\max} &= 0.436, \\ V_{\max} &= 0.615, \end{aligned}$$

et les poids

$$\begin{aligned} W_Y &= 1, \\ W_U &= \frac{U_{\max}}{1 - \omega_b}, \\ W_V &= \frac{V_{\max}}{1 - \omega_r}. \end{aligned}$$

Nous définissons aussi une matrice de permutation (car l'ordre est différent des *transmission primaries*),

$$P_{YUV} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \quad (3.3.41)$$

1. L'origine exacte des « constantes magiques » semble liée aux circuits analogiques d'encodage et de décodage en quadrature. Cependant, bien que j'aie cherché, je n'ai pas trouvé de justification explicite pour ces constantes. Poynton donne une formule pour les calculer, mais sans la justifier [304]. Il semblerait que les constantes aient été choisies pour une compatibilité avec une unité de mesure d'amplitude, les IRE, qui varient de 0 (pour noir) à 100 (pour blanc), les valeurs négatives étant réservées pour des signaux de contrôle [20].

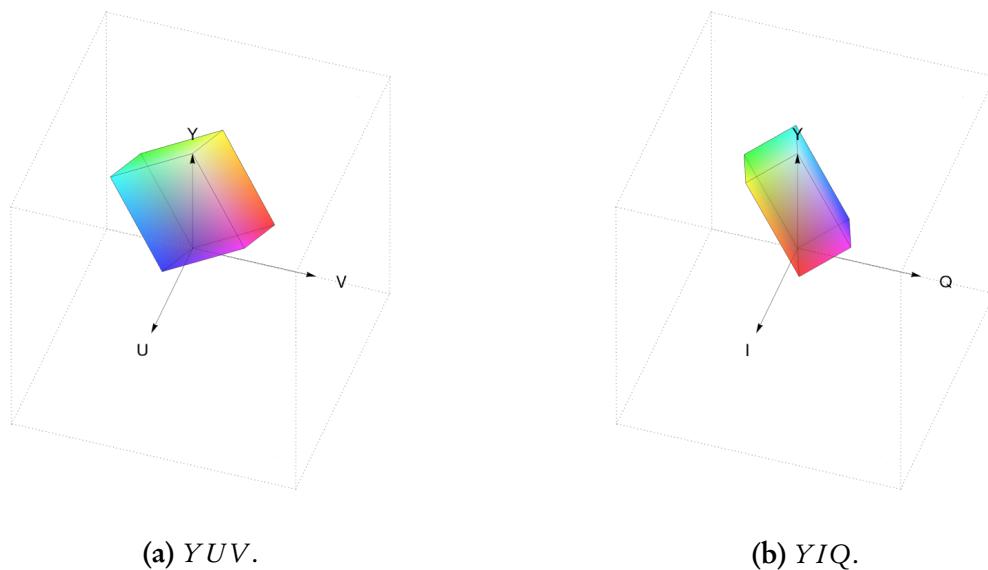


FIGURE 3.3.17— Les espaces de couleurs YUV et YIQ.

Enfin, nous pouvons définir la transformation de *RGB* à *YUV* :

$$\begin{aligned}
T_{YUV} \begin{bmatrix} r_{470} \\ g_{470} \\ b_{470} \end{bmatrix} &= \underbrace{\begin{bmatrix} W_Y & 0 & 0 \\ 0 & W_U & 0 \\ 0 & 0 & W_V \end{bmatrix}}_{\text{normalisation pour } YIQ} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{P_{YUV}} \underbrace{\begin{bmatrix} \omega_r & \omega_g & \omega_b \\ 1 - \omega_r & -\omega_g & -\omega_b \\ -\omega_r & -\omega_g & 1 - \omega_b \end{bmatrix}}_{T_{p_{470}}} \begin{bmatrix} r_{470} \\ g_{470} \\ b_{470} \end{bmatrix} \\
&= \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ -0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} r_{470} \\ g_{470} \\ b_{470} \end{bmatrix} \\
&= \begin{bmatrix} Y \\ U \\ V \end{bmatrix}. \tag{3.3.42}
\end{aligned}$$

L'inverse est donné par

$$T_{YUV}^{-1} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.140 \\ 1 & -0.394 & -0.581 \\ 1 & 2.032 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} r_{470} \\ g_{470} \\ b_{470} \end{bmatrix}. \quad (3.3.43)$$

L'espace YIQ transforme YUV en un espace compatible avec l'encodage par quadrature, mais il ne s'agit pas d'une simple mise à l'échelle pour occuper une boîte donnée : non ! il faut encore permuter deux rangées de YUV et lui appliquer une rotation de 33° en sens horaire. Nous revenons

donc aux *transmission primaries* :

$$\begin{aligned}
 T_{YIQ} &= R_r(-33^\circ) \begin{bmatrix} W_Y & 0 & 0 \\ 0 & W_V & 0 \\ 0 & 0 & W_U \end{bmatrix} T_p \\
 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos -33^\circ & \sin -33^\circ \\ 0 & -\sin -33^\circ & \cos -33^\circ \end{bmatrix} \underbrace{\begin{bmatrix} W_Y & 0 & 0 \\ 0 & W_V & 0 \\ 0 & 0 & W_U \end{bmatrix} \begin{bmatrix} \omega_r & \omega_g & \omega_b \\ 1 - \omega_r & -\omega_g & -\omega_b \\ -\omega_r & -\omega_g & 1 - \omega_b \end{bmatrix}}_{\text{une permutation de } YUV} \quad (3.3.44) \\
 &\approx \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.522 & 0.311 \end{bmatrix}.
 \end{aligned}$$

Enfin, nous avons :

$$T_{YIQ} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.522 & 0.311 \end{bmatrix} \begin{bmatrix} r_{470} \\ g_{470} \\ b_{470} \end{bmatrix} = \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}, \quad (3.3.45)$$

et l'inverse

$$T_{YIQ}^{-1} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.621 \\ 1 & -0.275 & -0.647 \\ 1 & -1.107 & 1.704 \end{bmatrix} \begin{bmatrix} r_{470} \\ g_{470} \\ b_{470} \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}. \quad (3.3.46)$$

Les deux espaces de couleurs *YUV* et *YIQ* sont montrés à la fig. 3.3.17.

*
* * *

Pourquoi faire simple quand on peut faire des standards? Le FCC (l'équivalent américain du CRTC) propose de modifier légèrement les matrices de *YIQ* [20, § 73.682], juste « parce que » :

$$T_{YIQ_{FCC}} = \begin{bmatrix} 0.3 & 0.59 & 0.11 \\ 0.599 & -0.2773 & -0.3217 \\ 0.213 & -0.5251 & 0.3121 \end{bmatrix} \quad (3.3.47)$$

et l'inverse est maintenant

$$T_{YIQ_{FCC}}^{-1} \approx \begin{bmatrix} 1 & 0.947 & 0.624 \\ 1 & -0.275 & -0.636 \\ 1 & -1.109 & 1.709 \end{bmatrix}. \quad (3.3.48)$$

3.3.9.3 Télévision numérique : BT.601, BT.709 et sRGB

La télévision analogique, sauf encore pour quelques rares stations, s'est éteinte. Elle a été remplacée par la télévision numérique haute définition (HDTV), rendue possible par la standardisation d'algorithmes de compression vidéo (MPEG2 qui assure la compression) et de transmission numérique (MPEG-TS qui en assure le transport par ondes ou par câble). Les résolutions supportées par la télévision numérique sont le 720p (des images de 1280×720 pixels, transmises progressivement) et le 1080p/i (des images de 1920×1080 pixels, transmises progressivement ou en mode entrelacé).

La recommandation BT.601, amendée par BT.709, définit de nouvelles couleurs primaires et de nouveaux espaces de couleurs, affranchis des mystérieux bricolages analogiques, choisis pour accommoder les nouvelles technologies des écrans plats de toutes sortes. BT.601, qui définit les paramètres pour la transition de la télévision analogique à la télévision numérique, définit les couleurs primaires de transfert [21] :

	625 lignes	525 lignes
	SÉCAM/PAL (Europe)	NTSC (Amér./Japon)
Rouge	$x = 0.64$	$x = 0.63$
Vert	$y = 0.33$	$y = 0.34$
Bleu	$x = 0.29$	$x = 0.31$
	$y = 0.60$	$y = 0.595$
	$x = 0.15$	$x = 0.155$
	$y = 0.06$	$y = 0.07$

C'est le standard BT.709 qui définit les nouvelles couleurs primaires [25] :

Rouge R_{709}	$x = 0.64$	$y = 0.33$
Vert G_{709}	$x = 0.30$	$y = 0.60$
Bleu B_{709}	$x = 0.15$	$y = 0.06$
Blanc D_{65}	$x = 0.3127$	$y = 0.3290$

Le blanc utilisé est l'illuminant D_{65} ($x = 0.3127$ et $y = 0.3290$). Cet espace *RGB* avec ces nouvelles couleurs primaires forment l'espace *sRGB* (pour *standard RGB*) ou encore noté RGB_{709} . Il faut donc convertir de l'espace de couleurs analogique vers RGB_{709} à partir des couleurs primaires définies selon le standard à 625 ou 525 lignes, c'est-à-dire calculer une conversion de coordonnées de BT.470 à BT.709, avant d'appliquer la transformation de couleurs vers l'espace YC_bC_r .

*
* * *

Les couleurs primaires de BT.709 servent de base à l'espace de couleurs standard des téléviseurs mais aussi des écrans d'ordinateurs récents, *sRGB* [11, 12]. Le standard IEC/4WD 61966-2-1 prévoit d'autres paramètres (généralement ignorés) comme la luminosité de l'écran (à 80 cd/m^2) et la couleur et l'intensité l'éclairage de la pièce environnante (350 lx de blanc D_{50} , soit $x = 0.3457$, $y = 0.3585$). Les transformations de xyz à *sRGB* et de *sRGB* à xyz sont données par

$$\begin{bmatrix} r_{sRGB} \\ g_{sRGB} \\ b_{sRGB} \end{bmatrix} = \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0587 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (3.3.49)$$

et

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} r_{sRGB} \\ g_{sRGB} \\ b_{sRGB} \end{bmatrix}. \quad (3.3.50)$$

Le standard prévoit aussi des transformations de type correction gamma. Soient donc r' , g' et b' les valeurs *sRGB* linéaires $0 \leq r, g, b \leq 1$ qui seront ajustées de façon non linéaire en r , g et b ¹. La transformation non-linéaire est, pour chacune des composantes, donnée par

$$\gamma_{sRGB}(u) = \begin{cases} u^{\frac{24}{323}} & \text{si } u \leq 0.04045 \\ \left(\frac{u + \frac{11}{200}}{1 + \frac{11}{200}}\right)^{2.4} & \text{sinon.} \end{cases} \quad (3.3.51)$$

On applique la transformation à chaque composante :

$$\begin{aligned} r &= \gamma_{sRGB}(r'), \\ g &= \gamma_{sRGB}(g'), \\ b &= \gamma_{sRGB}(b'). \end{aligned}$$

Le choix de l'exposant, 2.4 est tel qu'il est assez près de 2.2 pour maintenir une compatibilité approximative avec le matériel plus ancien [11, p. 7]. L'inverse de la fonction $\gamma_{sRGB}(\cdot)$, l'éq. (3.3.51), est donné par

$$\gamma_{sRGB}^{-1}(u) = \begin{cases} u^{\frac{323}{25}} & \text{si } u \leq 0.0031308 \\ \sqrt[2.4]{(1 + \frac{11}{200})u - \frac{11}{200}} & \text{sinon.} \end{cases} \quad (3.3.52)$$

Les transformations données par les éqs. (3.3.51) et (3.3.52) sont bricolées pour éviter que la correction ne donne des valeurs trop près de zéro lorsque la composante est petite, tout en appliquant une correction de type gamma lorsque les valeurs sont assez grandes. On remarquera aussi le grand nombre de « constantes magiques » qui ne sont ni justifiées, ni expliquées dans le standard.

1. C'est la notation préconisée par [11].

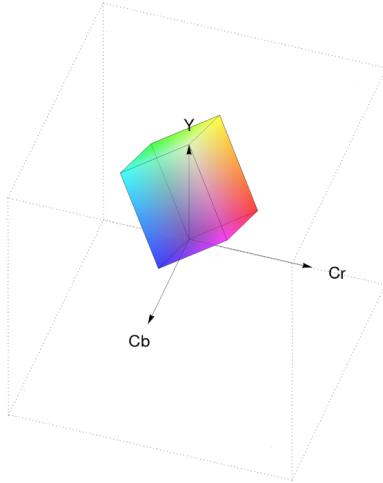


FIGURE 3.3.18 — L'espace de couleurs YC_bC_r

3.3.9.4 YC_bC_r

L'espace de couleurs YC_bC_r , défini dans le standard BT.709 [25] est un espace dérivé de l'espace YUV mais cette-fois ci normalisé pour que nous ayons $0 \leq Y \leq 1$, $-\frac{1}{2} \leq C_b \leq \frac{1}{2}$ et $-\frac{1}{2} \leq C_r \leq \frac{1}{2}$. Comme nous avons déjà fait ce genre de normalisation avec S_3 (un espace de couleurs de la série S , ci-dessus), on déduit rapidement que la formule pour transformer de RGB_{709} à YC_bC_r est donnée par :

$$\begin{aligned}
 T_{YC_bC_r} \begin{bmatrix} r_{709} \\ g_{709} \\ b_{709} \end{bmatrix} &= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2(1-\omega_b)} & 0 \\ 0 & 0 & \frac{1}{2(1-\omega_r)} \end{bmatrix}}_{\text{normalisation}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{P_{YUV}} \underbrace{\begin{bmatrix} \omega_r & \omega_g & \omega_b \\ 1-\omega_r & -\omega_g & -\omega_b \\ -\omega_r & -\omega_g & 1-\omega_b \end{bmatrix}}_{T_p} \begin{bmatrix} r_{709} \\ g_{709} \\ b_{709} \end{bmatrix} \\
 &= \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} r_{709} \\ g_{709} \\ b_{709} \end{bmatrix} \\
 &= \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix},
 \end{aligned} \tag{3.3.53}$$

tandis que l'inverse est donnée par

$$T_{YC_bC_r}^{-1} \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.334 & -0.714 \\ 1 & 1.772 & 0 \end{bmatrix} \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} r_{709} \\ g_{709} \\ b_{709} \end{bmatrix}. \tag{3.3.54}$$

L'espace de couleurs est montré à la fig. 3.3.18.

*
* *

Il existe une asymétrie entre la production de contenu (image ou vidéo) et sa consommation. Si on fait la supposition que contenu est créé une fois, mais qu'il est consommée plusieurs fois, on peut vouloir un encodage aussi dispendieux que l'on veut, mais on tiendra quand même à un décodage aussi rapide et aussi léger que possible. Cela permet de réaliser des décodeurs numériques à faible coût. Une des façons d'accélérer le décodage des images, c'est d'avoir une transformée de l'espace de couleurs intermédiaire à RGB rapide. Avec YC_bC_r , on remarque que

$$\begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.334 & -0.714 \\ 1 & 1.772 & 0 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 1.5 \\ 1 & -0.25 & -0.75 \\ 1 & 1.75 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \frac{3}{2} \\ 1 & -\frac{1}{4} & -\frac{3}{4} \\ 1 & \frac{7}{4} & 0 \end{bmatrix},$$

et il s'avère que des divisions par deux ou quatre se font bien en arithmétique entière grâce aux opérations de décalage. En effet, $\frac{3}{4}x$ c'est $\frac{x}{2} + \frac{x}{4}$, que l'on écrit $(x \gg 1) + (x \gg 4)$. Ces instructions de décalage sont presque tout le temps *beaucoup* plus rapides que des divisions entières, et considérablement plus rapides que les divisions en virgule flottante, ce qui rend l'astuce intéressante. Il ne reste plus qu'à calculer l'inverse de l'inverse pour trouver la transformée. Si

$$\tilde{T}_{YC_bC_r}^{-1} \begin{bmatrix} \tilde{Y} \\ \tilde{C}_b \\ \tilde{C}_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & \frac{3}{2} \\ 1 & -\frac{1}{4} & -\frac{3}{4} \\ 1 & \frac{7}{4} & 0 \end{bmatrix} \begin{bmatrix} \tilde{Y} \\ \tilde{C}_b \\ \tilde{C}_r \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}, \quad (3.3.55)$$

alors

$$(\tilde{T}_{YC_bC_r}^{-1})^{-1} = \tilde{T}_{YC_bC_r} = \begin{bmatrix} 0.304 & 0.609 & 0.087 \\ -0.174 & -0.348 & 0.522 \\ 0.464 & -0.406 & -0.058 \end{bmatrix},$$

et donc

$$\tilde{T}_{YC_bC_r} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} 0.304 & 0.609 & 0.087 \\ -0.174 & -0.348 & 0.522 \\ 0.464 & -0.406 & -0.058 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} \tilde{Y} \\ \tilde{C}_b \\ \tilde{C}_r \end{bmatrix}. \quad (3.3.56)$$

J'ai proposé cet espace de couleurs pour le standard DjVu (prononcé *déjà vu*) en 1997 ou à peu près. C'est encore l'espace de couleurs utilisé [299, p. 265] et [19, p. 44]. La différence avec YC_bC_r demeure petite.

3.3.9.5 YP_bP_r

C'est un encodage analogique (pour des câbles « composantes ») de l'espace YC_bC_r , avec une correction de type gamma (*power function*) sur les composantes. Pour r , g et b entre 0 et 1

inclusivement, on applique d'abord la transformation

$$\begin{aligned} r' &= r^{0.45}, \\ g' &= g^{0.45}, \\ b' &= b^{0.45}. \end{aligned}$$

avant de poursuivre à travers YC_bC_r pour la télévision « définition standard » avec RGB_{601} ou $Y'C'_bC'_r$ pour la télévision haute définition avec RGB_{709} (nous reparlerons de $Y'C'_bC'_r$ dans un moment) [305, p. 303–318].

Ce n'est donc pas exactement un espace de couleurs linéaire. L'inverse de la transformation d'espace de couleurs est donnée par l'inverse de YC_bC_r (ou de $Y'C'_bC'_r$). L'inverse de la correction gamma est étrange. Si nous avons $r' = r^{0.45}$, l'inverse devrait être $r = r'^{\frac{20}{9}} = r'^{2.2}$, puisque $0.45 = \frac{9}{20}$. Cependant, l'inverse est donné par $r = r'^{\frac{5}{2}}$ [304].

3.3.9.6 YD_bD_r

Le système de télévision analogique SÉCAM (pour Séquentiel couleur à mémoire¹) est, avec PAL (qui est encodé par YUV), l'un des standards compétiteurs de NTSC. SÉCAM était utilisé en France, en Afrique francophone et en Russie, et dans quelques autres pays [148]. SÉCAM utilise les couleurs primaires de BT.601 pour les images à 625 lignes (voir ci-haut). SÉCAM module les signaux de façon plus simple que NTSC :

$$\begin{aligned} T_{YD_bD_r} \begin{bmatrix} r_{\text{SÉCAM}} \\ g_{\text{SÉCAM}} \\ b_{\text{SÉCAM}} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{4}{3} \frac{1}{1-\omega_b} & 0 \\ 0 & 0 & \frac{4}{3} \frac{1}{1-\omega_r} \end{bmatrix} P_{YUV} T_p \begin{bmatrix} r_{\text{SÉCAM}} \\ g_{\text{SÉCAM}} \\ b_{\text{SÉCAM}} \end{bmatrix} \\ &= \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.450 & -0.833 & 1.333 \\ -1.333 & -1.116 & -0.217 \end{bmatrix} \begin{bmatrix} r_{\text{SÉCAM}} \\ g_{\text{SÉCAM}} \\ b_{\text{SÉCAM}} \end{bmatrix} \\ &= \begin{bmatrix} Y \\ D_b \\ D_r \end{bmatrix}. \end{aligned} \tag{3.3.57}$$

L'inverse est donné par

$$T_{YC_bC_r}^{-1} \begin{bmatrix} Y \\ D_b \\ D_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0.526 \\ 1 & -0.129 & -0.268 \\ 1 & 0.665 & 0 \end{bmatrix} \begin{bmatrix} Y \\ D_b \\ D_r \end{bmatrix} = \begin{bmatrix} r_{\text{SÉCAM}} \\ g_{\text{SÉCAM}} \\ b_{\text{SÉCAM}} \end{bmatrix} \tag{3.3.58}$$

1. Et non, comme le prétendent certaines langues bifides, le *système élégant contre les Américains!*

3.3.9.7 $Y'C'_bC'_r$ alias YC_bC_r HDTV

La recommandation BT.709 introduit, en plus de nouvelles couleurs primaires, un nouvel espace de couleur, $Y'C'_bC'_r$, défini plus précisément avec des pondérations différentes pour les couleurs primaires. Les nouvelles pondérations sont

$$\begin{aligned}w_r &= 0.2126, \\w_g &= 0.7152, \\w_b &= 0.0722.\end{aligned}$$

Les coefficients sont donnés avec plus de précision pour accommoder des valeurs sur 8 et 10 bits. Comme pour YP_bP_r , les valeurs RGB sont ajustées par la même correction que Kodak YCC, l'éq. (3.3.8), avant d'être transformées vers l'espace de couleurs $Y'C'_bC'_r$. Ce dernier est bâti de la même façon que l'espace YC_bC_r , à la différence qu'on remplace ω_r , ω_g et ω_b par w_r , w_g et w_b :

$$\begin{aligned}T_{Y'C'_bC'_r}^{-1} \begin{bmatrix} r_{709} \\ g_{709} \\ b_{709} \end{bmatrix} &= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2(1-w_b)} & 0 \\ 0 & 0 & \frac{1}{2(1-w_r)} \end{bmatrix}}_{\text{normalisation}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}}_{P_{YUV}} \underbrace{\begin{bmatrix} w_r & w_g & w_b \\ 1-w_r & -w_g & -w_b \\ -w_r & -w_g & 1-w_b \end{bmatrix}}_{T_{p709}} \begin{bmatrix} r_{709} \\ g_{709} \\ b_{709} \end{bmatrix} \\&= \begin{bmatrix} 0.2126 & 0.7152 & 0.0722 \\ -0.1146 & -0.3854 & 0.5 \\ 0.5 & -0.4542 & -0.0458 \end{bmatrix} \begin{bmatrix} r_{709} \\ g_{709} \\ b_{709} \end{bmatrix} \\&= \begin{bmatrix} Y' \\ C'_b \\ C'_r \end{bmatrix},\end{aligned}\tag{3.3.59}$$

L'inverse est donné par

$$T_{Y'C'_bC'_r}^{-1} \begin{bmatrix} Y' \\ C'_b \\ C'_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.5748 \\ 1 & -0.1874 & -0.4681 \\ 1 & 1.8556 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ C'_b \\ C'_r \end{bmatrix} = \begin{bmatrix} r_{709} \\ g_{709} \\ b_{709} \end{bmatrix}.\tag{3.3.60}$$

3.3.9.8 BT.2020 UHDTV

Pour la télévision 4K (3840×2160 pixels) et 8K (7640×4320 pixels), la recommandation BT.2020 définit *encore* de nouvelles couleurs primaires, de nouveaux poids, et un nouvel espace de couleurs [24]. Les couleurs primaires sont

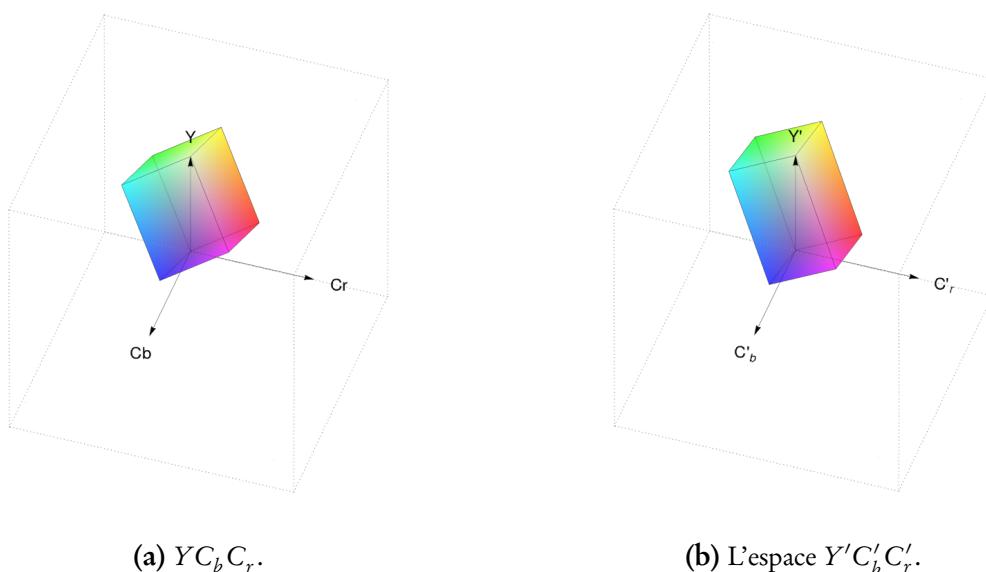


FIGURE 3.3.19 — Les espaces de couleurs YC_bC_r et $Y'C'_bC'_r$ comparés.

Rouge R_{2020}	$x = 0.708$	$y = 0.292$
Vert G_{2020}	$x = 0.170$	$y = 0.797$
Bleu B_{2020}	$x = 0.131$	$y = 0.046$
Blanc D_{65}	$x = 0.3127$	$y = 0.3290$

tandis que les pondérations sont données par

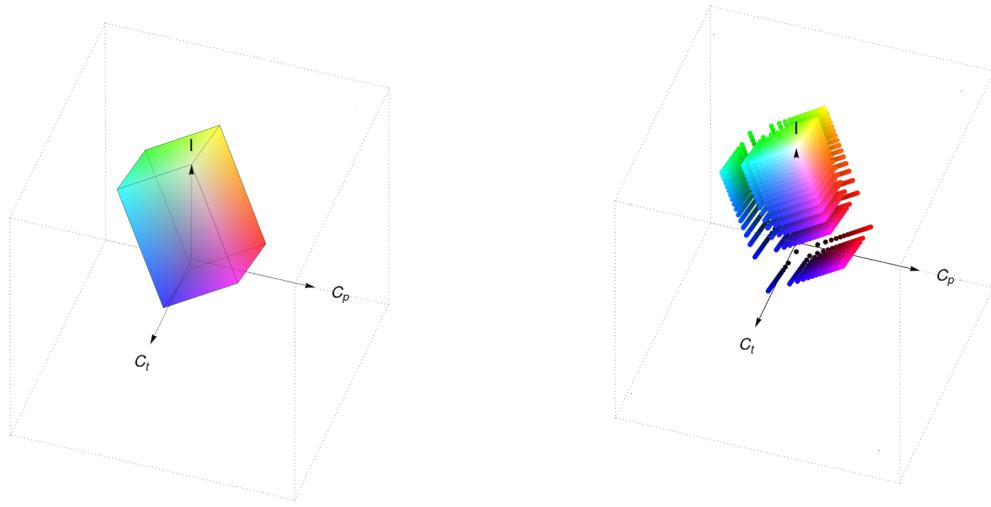
$$\begin{aligned}W_r &= 0.2627, \\W_g &= 0.6780, \\W_b &= 0.0593.\end{aligned}$$

La transformée de couleurs est aussi passablement plus compliquée que les précédentes. Comme pour BT.709 et KodakYCC, on commence par une correction de type gamma :

$$c' = \begin{cases} \alpha c^{0.45} - (\alpha - 1) & \text{si } \beta \leq c \geq 1 \\ 4.5c & \text{sinon.} \end{cases} \quad (3.3.61)$$

où $\alpha = 1.099$ et $\beta = 0.018$ pour les images en 10 bits par composantes, tandis que nous aurons $\alpha = 1.0993$ et $\beta = 0.0181$ pour les images en 12 bits. Après la correction (qui a pour effet de donner peu de codes aux couleurs très sombres et plus aux couleurs intenses, comme nous le montre la fig. 3.3.20 (b)), nous obtenons r' , g' et b' . La transformée de couleur est

$$Y'_c = (W_r, W_g, W_b)^T(r', g', b') = 0.2627r' + 0.6780g' + 0.0593b', \quad (3.3.62)$$



(a) L'espace \$Y'C_b'C_r'\$.

(b) La densité de l'espace \$Y'C_b'C_r'\$ (effet exagéré).

FIGURE 3.3.20 — L'espace de couleurs \$Y'C_{BC}'C_{RC}'\$

puis

$$C'_{BC} = \begin{cases} \frac{b' - Y'_C}{-2N_B} & \text{si } N_B \leq b' - Y'_C \leq 0 \\ \frac{b' - Y'_C}{2P_B} & \text{sinon,} \end{cases} \quad (3.3.63)$$

et

$$C'_{RC} = \begin{cases} \frac{r' - Y'_C}{-2N_R} & \text{si } N_R \leq r' - Y'_C \leq 0 \\ \frac{r' - Y'_C}{2P_R} & \text{sinon,} \end{cases} \quad (3.3.64)$$

où

$$\begin{aligned} P_B &= \alpha(1 - 0.0593^{0.45}) \approx 0.7910 \\ N_B &= \alpha(1 - 0.9407^{0.45}) \approx -0.9720 \\ P_R &= \alpha(1 - 0.2627^{0.45}) \approx 0.4969 \\ P_N &= \alpha(1 - 0.7373^{0.45}) \approx -0.8591. \end{aligned}$$

Les indices en « C » (comme \$C'_{RC}\$) sont censés indiquer « luminosité constante », dans la mesure où ils préservent, avec une bonne précision, les valeurs \$r'\$, \$g'\$, \$b'\$. L'espace \$Y'C_{BC}'C_{RC}'\$ est montré à la fig. 3.3.20. La recommandation BT.2020 dit par ailleurs que pour une approximation, on pourra utiliser \$Y'C_b'C_r'\$, mais avec les poids \$W_r\$, \$W_g\$ et \$W_b\$; la différence entre les deux espaces est négligeable — si on néglige le fait qu'ils n'utilisent pas les mêmes couleurs primaires.



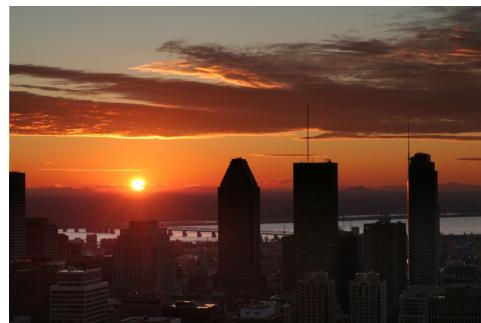
(a) Image exposée pour les lumières intenses.



(b) Image exposée pour les lumières moyennes.



(c) Image exposée pour les lumières faibles.



(d) Image HDR recomposée.

FIGURE 3.3.21 — Une image en grande gamme dynamique (HDR).

3.3.9.9 BT.2100 HDRTV

Le standard BT.2100 spécifie les caractéristiques pour un téléviseur à grande gamme dynamique (en anglais HDR pour *High Dynamic Range*), censé s'approcher de l'étendue de la vision humaine. Rappelez-vous la fig. 1.1.10, p. 11, où l'on montre la réponse perçue à l'intensité de lumière reçue. Les images HDR présentent, au moins en principe, une très grande différence de luminosité entre la luminosité maximale (au seuil de l'aveuglement) et minimale (au seuil de la perception).

Dans la réalité, la bande dynamique demeure modeste, mais des traitements sont apportés aux couleurs pour donner l'impression d'une grande bande dynamique. La fig. 3.3.21 explique comment : on prend plusieurs images de la même scène, une en exposant pour les lumières intenses (fig. 3.3.21 (a)), une pour les lumières moyennes (fig. 3.3.21 (b)), et une pour les lumières faibles (fig. 3.3.21 (c)). On les combine ensuite en choisissant les pixels en fonction de l'image que notre œil percevrait — il y a un flou artistique pour réussir à produire des images vraisemblables et agréables. C'est ce processus que les standards HDR tentent d'automatiser, en approximant la courbe de réponse de la fig. 1.1.10, p. 11, en « boostant » les intensités les plus faibles et en atténuant les plus fortes.

BT.2100 définit le standard pour la télévision HDR [26]. Les transformations de couleurs sont non linéaires, mais BT.2100 est listé dans cette section par ce que le cœur de la transformée est linéaire. Il se distingue aussi des autres standards en ce qu'il utilise une composition de transformées, censée rendre la réponse du système visuel [82].

La première matrice de transformation (après des étapes de correction de type gamma que nous ne décrirons pas ici mais qui se trouvent fort bien détaillées dans la recommandation [26, p. 4+]) convertit les r , g et b (avec les couleurs primaires de BT.2020) dans l'espace LMS grâce à la transformation

$$T_{LMS} \begin{bmatrix} r_{2020} \\ g_{2020} \\ b_{2020} \end{bmatrix} = \frac{1}{4096} \begin{bmatrix} 1688 & 2146 & 262 \\ 683 & 2951 & 462 \\ 99 & 309 & 3688 \end{bmatrix} \begin{bmatrix} r_{2020} \\ g_{2020} \\ b_{2020} \end{bmatrix} = \begin{bmatrix} l \\ m \\ s \end{bmatrix}. \quad (3.3.65)$$

L'espace LMS (pour *Long, Medium, Short Wavelengths*) est censé reproduire la réponse moyenne des cônes rouges, verts et bleus. Le diviseur 4096 normalise les valeurs sur 12 bits (car le standard prévoit des images sur 10 ou 12 bits).

Par la suite, une seconde transformée est appliquée à l'espace LMS pour l'amener dans l'espace $IC_T C_p$, dont les composantes représentent la luminance, la différence jaune-bleu et la différence rouge-vert/cyan. La différence jaune-bleu est dite tritanopique, tandis que la différence rouge-vert est protanopique¹, d'où le T et le P dans $IC_T C_p$.

La transformation de $L'M'S'$ (car encore une fois une correction de type gamma est appliquée) à $IC_T C_P$ est donnée par

$$\begin{aligned} T_{IC_T C_P} \begin{bmatrix} l' \\ m' \\ s' \end{bmatrix} &= \frac{1}{4096} \begin{bmatrix} 2048 & 0 & 2048 \\ 6610 & -13613 & 7003 \\ 17933 & -17933 & -542 \end{bmatrix} \begin{bmatrix} l' \\ m' \\ s' \end{bmatrix} \\ &= \frac{1}{4096} \begin{bmatrix} 2048 & 0 & 2048 \\ 6610 & -13613 & 7003 \\ 17933 & -17390 & -542 \end{bmatrix} \sigma \left(\frac{1}{4096} \begin{bmatrix} 1688 & 2146 & 262 \\ 683 & 2951 & 462 \\ 99 & 309 & 3688 \end{bmatrix} \begin{bmatrix} r_{2020} \\ g_{2020} \\ b_{2020} \end{bmatrix} \right) \quad (3.3.66) \\ &= \begin{bmatrix} I \\ C_T \\ C_P \end{bmatrix}. \end{aligned}$$

1. Pour être strict, il faudrait parler de tritanopie et de protanopie que lorsque nous sommes *incapables* de voir ces différences, comme par exemple, si nous souffrons d'une des formes de daltonisme. Il existe plusieurs autres conditions, dépendamment de quels récepteurs font défaut. Voir, par exemple [35, 259].

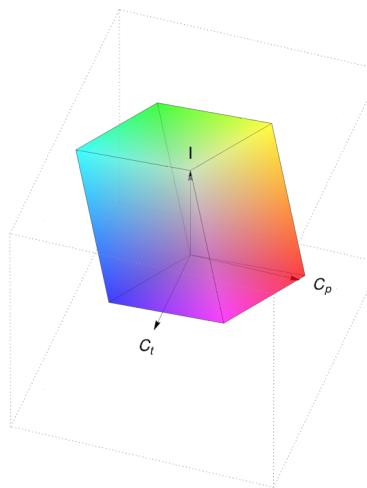


FIGURE 3.3.22 — L'espace de couleurs $IC_T C_P$ transformant LMS transformant RGB , sans les corrections non linéaires.

où $\sigma(\cdot)$ représente la fonction de transfert non linéaire. La transformée inverse est donnée par

$$T_{IC_T C_P}^{-1} \begin{bmatrix} I \\ C_T \\ C_P \end{bmatrix} = \begin{bmatrix} 0.9999 & -0.2757 & 0.2158 \\ 1 & -0.2929 & -0.0062 \\ 1.0001 & 2757 & -0.2158 \end{bmatrix} \begin{bmatrix} I \\ C_T \\ C_P \end{bmatrix} = \begin{bmatrix} l' \\ m' \\ s' \end{bmatrix}, \quad (3.3.67)$$

puis enfin, après l'inverse de la transformation non linéaire qui nous amène de $L'M'S'$ à LMS , on peut revenir à RGB

$$T_{LMS}^{-1} \begin{bmatrix} l \\ m \\ s \end{bmatrix} = \begin{bmatrix} 3.4361 & -2.5058 & 0.0697 \\ -0.7909 & 1.9831 & -0.1922 \\ -0.0261 & -0.0987 & 1.1248 \end{bmatrix} \begin{bmatrix} l \\ m \\ s \end{bmatrix} = \begin{bmatrix} r_{2020} \\ g_{2020} \\ b_{2020} \end{bmatrix}. \quad (3.3.68)$$

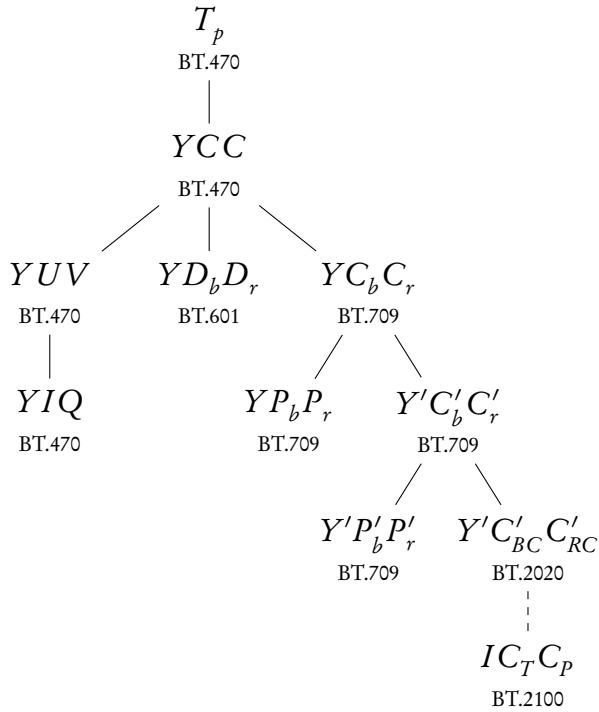
L'espace de couleurs $IC_T C_P$, sans les corrections non linéaires, est montré à la fig. 3.3.22.

*
* * *

Vous aurez compris que les transformations non linéaires de RGB à LMS et de LMS à $IC_T C_P$ sont censé modéliser la courbe de réponse de la fig. 1.1.10, et ainsi rendre l'illusion d'une image à grande gamme dynamique. Cependant, l'image reste ultimement limitée par la gamme relativement modeste de l'écran.

3.3.9.10 Un portrait de famille

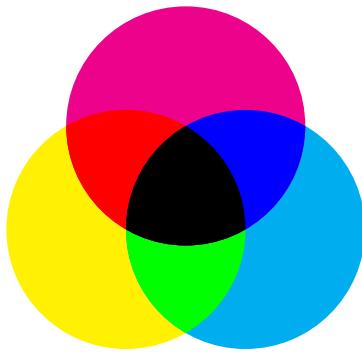
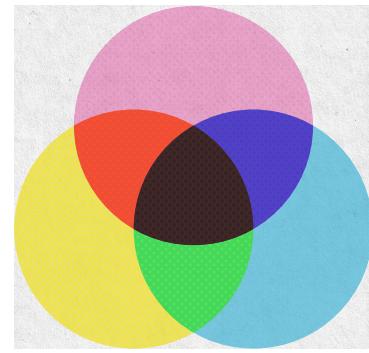
La figure 3.3.23 montre les filiations entre les divers espaces de couleur de la télévision. Dans l'arbre généalogique, les liens directs se limitent à des transformations simples dans la structure des

**FIGURE 3.3.23** — La famille BT.

espaces de couleurs (changement des couleurs primaires ou de leur pondération, échanger des lignes de la matrice des *transmission primaries*, changement de normalisation, etc.). Quelques fois, l'ajout d'une correction de type gamma vient bouleverser la linéarité stricte des espaces de couleurs. Nous avons vu aussi que pour certains espaces (comme YP_bP_r), ces corrections non linéaires sont présentes pour accommoder un transport analogue, ou une autre contrainte. Parfois, (comme BT.2100), elles jouent un rôle fondamental dans l'espace de couleur. Pour en savoir plus sur ces espaces de couleurs et leurs interactions avec les différents standards de télévision, consultez [54, 124, 125, 305].

3.3.10 CMY et CMYK

Nous avons déjà rencontré l'espace de couleurs *CMYK* au chapitre 1. Il s'agit d'un espace linéaire, mais *soustractif*, c'est-à-dire que les « pixels » (des points imprimés) filtrent la lumière reçue plutôt qu'en émettent — ce qui fait des autres espaces de couleurs des espaces *additifs*, ils émettent de la lumière. Les quatre couleurs de base sont le cyan, *C*, le magenta, *M*, le jaune, *Y* (qu'il ne faut pas confondre ici avec le *Y* de luminance dans d'autres espaces de couleurs), et le noir, *K*. On pourrait penser que *K* a été tiré du *k* final de *blacK* pour le désambiguïser du bleu *B*, mais non! *K* vient de *key*, la « clef d'alignement » en impression — qui n'est pas forcément noire! Les couleurs sont souvent alignées contre une référence qui délimite les contours (voir, par exemple, les fig. 1.1.3 (c) et 1.1.3 (d) pour comprendre pourquoi et comment).

(a) L'espace *CMYK* : résultats désirés.(b) L'espace *CMYK* : résultats obtenus!FIGURE 3.3.24 — L'espace de couleurs *CMYK*, attentes et résultats !

Si nous disposions d'encre *parfaites* sur papier *transcendant*, nous obtiendrions, à l'imprimé, des couleurs vives et précises, comme à la fig. 3.3.24 (a). Or, les encres et les papiers utilisés sont toujours en deçà des espérances, et les couleurs obtenues sont moins brillantes et moins précises, on obtient un résultat semblable à la fig. 3.3.24 (b), où, en plus d'utiliser des encres moins que parfaites, le tramage a été utilisé. Dans un monde idéal, le mélange des encres donnerait la couleur secondaire avec précision, or, le magenta et le cyan mélangés ne donnent pas un bleu pur, mais un bleu vaguement prussien¹, et le mélange des trois couleurs ne donne pas noir, mais un brun vaguement scatochrome. C'est une des raisons pourquoi le noir est ajouté, et assez souvent aussi des *spot colors*, des encres avec une couleur spécifique (voir fig. 1.1.3 (a)). Nous verrons comment l'ajout du noir a aussi pour effet de permettre une économie des encres colorées.

*
* * *

Si on suppose que *CMY* sont les couleurs parfaitement *complémentaires* à *RGB* (car $C = G + B$, et donc $W - R = G + B = C$) on peut obtenir une première conversion (linéaire) et naïve :

$$\begin{aligned} c &= 1 - r, \\ m &= 1 - g, \\ y &= 1 - b, \end{aligned}$$

que l'on peut exprimer comme un produit matrice/vecteur :

$$\begin{bmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \\ 1 \end{bmatrix} = \begin{bmatrix} 1 - r \\ 1 - g \\ 1 - b \\ 1 \end{bmatrix} = \begin{bmatrix} c \\ m \\ y \\ 1 \end{bmatrix}. \quad (3.3.69)$$

1. Le *bleu de Prusse*, *Preußischblau* en Allemand, est un bleu chimique à base de fer proche de l'indigo [86, p. 92–98].

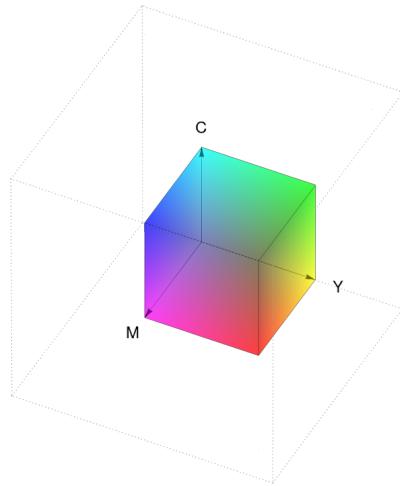


FIGURE 3.3.25 — L'espace de couleurs *CMY*.

L'inverse est donné par

$$r = 1 - c,$$

$$g = 1 - m,$$

$$b = 1 - y,$$

c'est-à-dire

$$\begin{bmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} c \\ m \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 - c \\ 1 - m \\ 1 - y \end{bmatrix} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad (3.3.70)$$

où on ignore joyeusement k . On remarque que l'inverse n'est plus une matrice carrée : comme c'est une projection d'un espace à 4 dimensions sur un espace à 3, la matrice est rectangulaire 3×4 ¹. L'éq. (3.3.69) ne se réduit pas à une matrice 4×3 à cause de la « translation » : il faut alors utiliser des coordonnées homogènes. L'espace de couleurs *CMY* est montré à la fig. 3.3.25.

Mais la transformée naïve de *RGB* à *CMYK* ne fonctionne qu'à l'écran ! En imprimé, on ne peut pas avoir un cyan foncé en ajoutant plus d'encre cyan ! Un peu d'encre cyan filtre un peu de jaune, beaucoup d'encre filtrera beaucoup de jaune, mais ne devient pas cyan foncé. Il faut donc inclure le noir dans notre modèle pour permettre de faire des couleurs foncées — et les couleurs pâles

1. On pourrait voir toute matrice de transformation comme une projection d'un espace à n dimensions vers un autre espace à m dimensions. Lorsque que nous projetons un espace à $n > m$ dimensions en m dimensions, la matrice sera rectangulaire horizontalement et $m \times n$, c'est-à-dire avec m rangées de n colonnes. Si nous projetons un espace à $n < m$ dimensions dans un espace en m dimensions, la matrice sera encore rectangulaire, cette fois verticalement, $m \times n$.

sont mélangées avec du blanc, mais le blanc, c'est le papier! Il nous faut donc maintenant inclure la composante K que nous supposerons noire¹.

Pour ajouter le noir à CMY pour obtenir $CMYK$, on pose

$$\begin{aligned} k &= \min(c, m, y) \\ &= \min(1 - r, 1 - g, 1 - b) \\ &= 1 - \max(r, g, b) \\ &= 1 - \gamma. \end{aligned}$$

Autrement dit, la composante K , c'est ce qu'on a ôté à blanc. Alors, si $k \neq 1$, nous avons

$$\begin{aligned} c' &= \frac{c - k}{1 - k}, \\ m' &= \frac{m - k}{1 - k}, \\ y' &= \frac{y - k}{1 - k}, \\ k &= \min(c, m, y). \end{aligned} \tag{3.3.71}$$

sinon nous posons $(c', m', y', k) = (0, 0, 0, 1)$.

On peut comprendre que les composantes (c', m', y', k) (avec des apostrophes pour les distinguer de CMY) sont encodées différentiellement par rapport au niveau de noir qu'elles comportent déjà. Les différences sont normalisées sur $1 - k$ (une étape qui ne paraît pas essentielle, mais qui est traditionnellement présente). Pour revenir à CMY , on applique les transformations inverses :

$$\begin{aligned} c &= (1 - k)c' + k, \\ m &= (1 - k)m' + k, \\ y &= (1 - k)y' + k, \end{aligned} \tag{3.3.72}$$

que l'obtient facilement à partir des éqs. (3.3.71). On peut ensuite revenir à RGB .

*
* * *

Passer par l'espace intermédiaire — et par ailleurs plutôt inutile — CMY pour se rendre de RGB à $CMYK$ paraît bien superflu. Créons donc la transformée directe de RGB à $CMYK$ (et son inverse, de $CMYK$ à RGB). Rappelons : nous avons

$$k = 1 - \max(r, g, b) = 1 - \gamma.$$

1. Comme nous l'avons mentionné précédemment, le *key* est la couleur d'alignement, et elle n'est pas forcément noire.

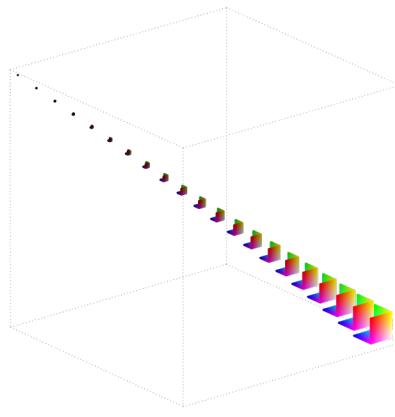


FIGURE 3.3.26 — L'espace de couleurs CMYK (les K sont montrés par translations).

Nous avons suite que

$$\begin{aligned} c' &= \frac{c-k}{1-k} \\ &= \frac{(1-r)-k}{1-k} \\ &= \frac{(1-r)-(1-\gamma)}{1-(1-\gamma)} \\ &= \frac{\gamma-r}{\gamma} = 1 - \frac{r}{\gamma}. \end{aligned}$$

En faisant la même transformation pour les autres composantes, si $\gamma \neq 0$, nous trouvons

$$\begin{aligned} c' &= \frac{\gamma-r}{\gamma} = \frac{(1-k)-r}{1-k}, \\ m' &= \frac{\gamma-g}{\gamma} = \frac{(1-k)-g}{1-k}, \\ y' &= \frac{\gamma-b}{\gamma} = \frac{(1-k)-b}{1-k}, \\ k &= 1-\gamma, \end{aligned} \tag{3.3.73}$$

sinon nous posons $(c', m', y', k) = (0, 0, 0, 1)$. Pour trouver l'inverse, on utilise les éqs. (3.3.73) pour isoler les composantes RGB , et on obtient :

$$\begin{aligned} r &= (1-k)(1-c'), \\ g &= (1-k)(1-m'), \\ b &= (1-k)(1-y'). \end{aligned} \tag{3.3.74}$$

*

* *

Il est impossible de visualiser directement l'espace $CMYK$ en « cube des couleurs » puisque celui-ci est en quatre dimensions. Si on dessine, comme à la fig. 3.3.26, pour chaque K les trois autres dimensions, on découvre une structure qui rappelle les poupées russes, où chaque surface CMY est une petite *matriochka*. On voit aussi que correctement empilées, les « coquilles » correspondront au cube RGB .

3.4 Les espaces de couleurs non linéaires

Les espaces non linéaires se divisent en deux grandes familles. D'abord, nous avons la famille des espaces non linéaires seulement par leur système de coordonnées, souvent une variation sur le thème des coordonnées polaires ou sphériques. Puis nous avons la famille des espaces où les coordonnées sont non linéaires pour répondre à un critère perceptuel, par exemple, s'assurer qu'un déplacement dans l'espace des coordonnées correspond à une variation proportionnelle dans la perception. Cette dernière famille est plus difficile à concevoir : il s'agit de bien modéliser la réponse de l'observateur. Nous présenterons des espaces de chaque famille.

*

* *

Établissons le vocabulaire que nous utiliserons pour ce qui suit :

- La *teinte*, c'est-à-dire la couleur « pure ». La plupart des espaces de couleurs utilisent un système de coordonnées polaires ou sphériques où un angle donne la teinte, avec une teinte spécifique choisie pour correspondre à l'angle zéro (typiquement rouge) ; et d'autres pour des angles spécifiques (Munsell utilise cinq couleurs également distribuées autour du cercle chromatique, Ostwald en utilise quatre, *HSV/HSL* en utilisent six). Les couleurs primaires sont choisies en fonction des applications spécifiques.
- La *saturation* ou la *pureté*, c'est-à-dire la portion de la couleur pure mélangée au gris neutre de même valeur. La plupart des systèmes dénotent la pureté comme étant maximale lorsque la couleur est le plus éloigné de l'axe central, auxquels correspondent les tons de gris ; la pureté varie donc entre la couleur maximamente distinguée et un gris neutre de même valeur.

- La *valeur*. La luminance mesure la puissance de la lumière visible, elle n'a pas limite supérieure évidente ; cependant, les espaces de couleurs la supposent limitée entre 0, noir, et 1, blanc, que l'on suppose tous deux idéaux — dans la réalité, toutefois, ni l'un ni l'autre ne sont parfaits, mais seulement limités par l'équipement de rendu.

La plupart des systèmes (mais pas tous) profitent du fait qu'une couleur ne peut pas être à la fois très saturée et très brillante, car elle est perçue comme étant « blanche » et qu'une couleur ne peut être très saturée et très foncée en même temps, car elle est perçue comme « noircie », pour restreindre l'espace de couleurs dans ces régions. Ostwald propose un système en double cône à cet effet, tandis que l'espace de couleurs *C M Y K* ne donne beaucoup de place qu'aux couleurs maximalement saturées et très peu aux couleurs foncées (voir la fig. 3.3.26).

3.4.1 *HSV* et *HSL*

Les espaces (et différents systèmes de classement) de couleurs sont depuis longtemps (voir § 3.5) basés sur un cercle chromatique où un certain nombre de couleurs primaires sont placées, les teintes intermédiaires entre elles, aussi sur le cercle. Certains étendent le modèle à un disque, où couleurs maximalement saturées (« pures ») sont placé sur le rebord et les couleurs peu saturées près du centre, laissant le centre exact pour les gris achromatiques. La fig. 3.4.1 montre un tel arrangement, celui que *HSV* et *HSL* utilisent.

L'idée est donc de découper le cercle (ou le disque) chromatique en secteurs, où chaque angle spécifique indique une teinte, et où la distance au centre donne la saturation, les plus loin du centre étant les plus saturés. Un autre axe, perpendiculaire aux deux autres donne la luminosité de la couleur. Comme nous l'avons fait remarqué précédemment, une couleur ne peut être à la fois très saturée et très lumineuse, car elle est perçue comme blanche, de même qu'une couleur foncée n'apparaît justement que foncée, sans grand-couleur. Les systèmes *HSV* et *HSL* diffèrent principalement dans la façon dont ils exploitent cette observation.

3.4.1.1 *HSV*

Revenons à la fig. 3.4.1. L'espace *HSV* est composé de trois dimensions :

- *H*, la teinte (*hue*), est un angle (dépendamment des implémentations, de 0° à 360° , de 0 à 2π , ou même 0 à 1) relatif à la couleur de départ, presque tout le temps le rouge.
- *S*, la saturation, ou la pureté, est la distance au centre. Typiquement, elle variera de 0 (achromatique) à 1 (maximalement saturée).

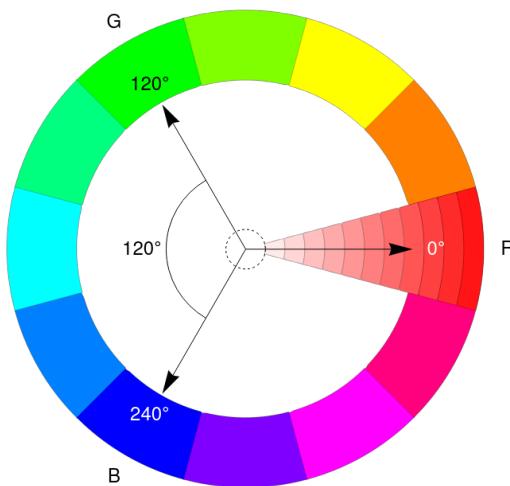


FIGURE 3.4.1 — Le cercle chromatique de l'espace HSV.

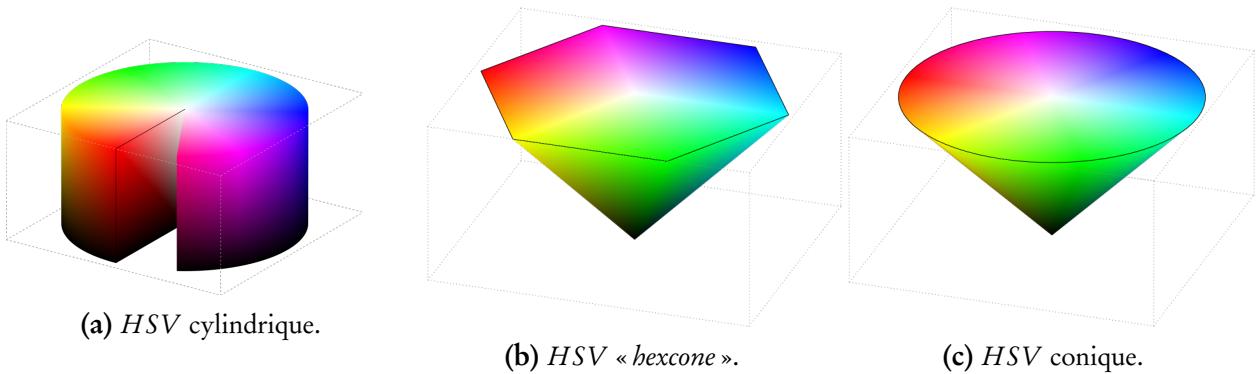
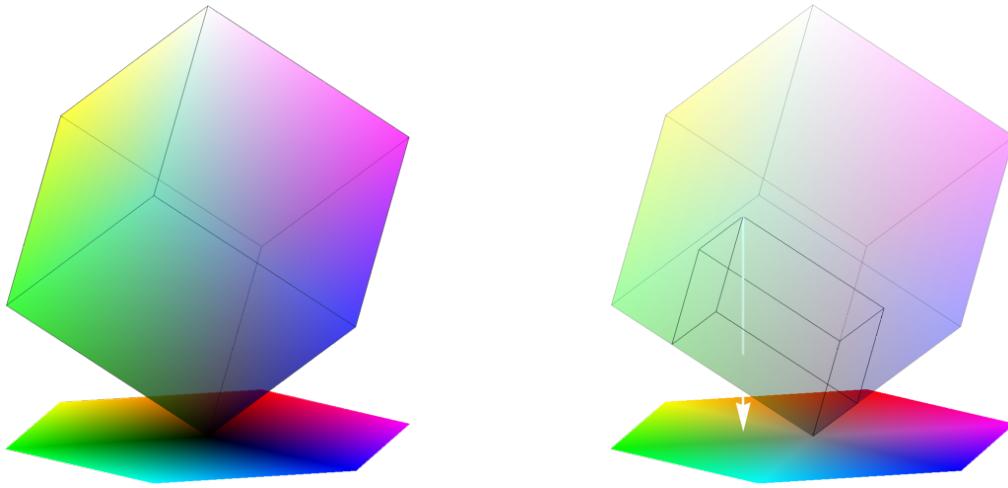


FIGURE 3.4.2 — Géométries de l'espace HSV.

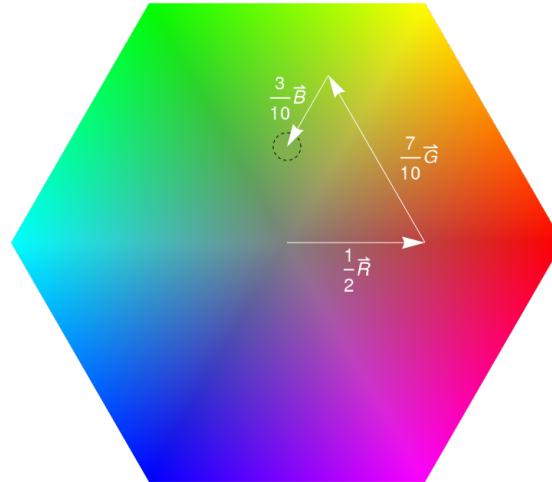
- V , la valeur, la luminosité perçue de la couleur, aussi normalisée entre 0 (noir) et 1 (blanc).

Comment implémenter cet espace de couleurs? Probablement qu'une approche naïve nous donnerait un espace comme celui montré à la fig. 3.4.2 (a), où les trois dimensions sont indépendantes. Ce cylindre, qui n'est pas sans rappeler le pudding à l'arsenic avec sa tranche retirée, donne beaucoup de codes aux couleurs très foncées difficiles à distinguer, même s'il place les tons de gris le long de son axe central. La fig. 3.4.2 (b) nous montre une projection du cube RGB contre un hexagone et compresse les couleurs difficiles à distinguer ensemble, pour former une pointe dont le sommet est le noir. La fig. 3.4.3 (a) nous montre qu'appliquer une rotation au cube RGB de façon à rendre sa grande diagonale verticale nous permet une projection qui forme un hexagone régulier. Mais comment faire cette projection?

Considérons trois vecteurs (colonnes) unitaires $\vec{r} = (r_1, r_2)$, $\vec{g} = (g_1, g_2)$ et $\vec{b} = (b_1, b_2)$ dans le plan, chacun à 120° des autres. Si ces vecteurs sont les projections des vecteurs \vec{R} , \vec{G} et \vec{B} dans le plan, alors les couleurs qui sont des combinaisons linéaires de \vec{R} , \vec{G} et \vec{B} (dans le cube RGB) ont leur projection

(a) Une projection du cube *RGB*.

(b) La projection d'une couleur en particulier.

FIGURE 3.4.3 — *HSV* et une projection du cube *RGB*.**FIGURE 3.4.4** — La projection sur l'hexagone *HSV*.

dans la combinaison linéaire de mêmes coefficients de \vec{r} , \vec{g} et \vec{b} ! Autrement dit,

$$\begin{bmatrix} \vec{r} & \vec{g} & \vec{b} \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} r_1 & g_1 & b_1 \\ r_2 & g_2 & b_2 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix}.$$

Cette projection n'est pas encore tout à fait complète : nous perdons l'information de luminance et plusieurs couleurs sont projetées au même endroit ! Les coordonnées (u, v) nous donnent, au moins, l'information sur la teinte, puisque nous pouvons obtenir l'angle que forme (u, v) avec l'origine. La saturation est proportionnelle à la norme $\|(u, v)\|$ (la longueur du vecteur) et la valeur, la luminosité, doit être une fonction de (r, g, b) . Mais ces opérations (\tan^{-1} , $\sqrt{u^2 + v^2}$, etc.) sont dispendieuses en temps de calcul, donc on cherchera à les éviter en construisant la transformée différemment.

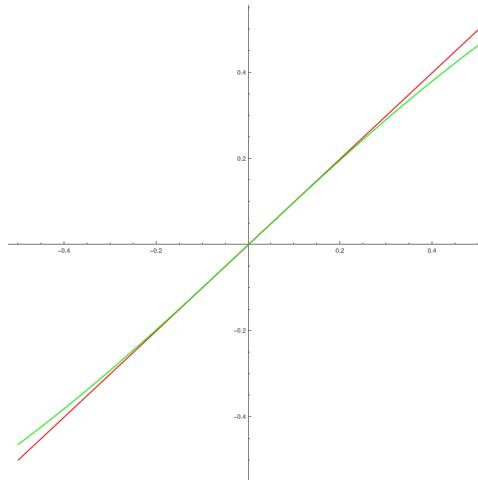


FIGURE 3.4.5 — Sur l'intervalle $[-\frac{1}{2}, \frac{1}{2}]$, $\tan^{-1} x \approx x$.

Clairement, calculer exactement l'angle et la saturation (la distance au centre) pour une couleur (r, g, b) est dispendieux computationnellement. Nous allons donc tricher un peu et changer le calcul tout en conservant l'esprit de la transformation. Commençons par remarquer que près de $x = 0$, on a

$$\tan^{-1} x \approx x,$$

et comme nous le montre la fig. 3.4.5, l'erreur sur l'intervalle $[-\frac{1}{2}, \frac{1}{2}]$ demeure modeste; au plus $\pm 0.0363524\dots$, une erreur relative d'au plus $\approx 7\%$. Cela nous permet de prendre la tangente x pour son angle — c'est bien entendu inexact, mais l'erreur est tolérable — et de nous dispenser de la vilaine fonction trigonométrique \tan .¹

Ensuite, remarquons, avec l'aide de la fig. 3.4.6, que l'espace des couleurs est divisé en trois régions, où l'une des couleurs primaires domine. Dans la région 0, c'est le rouge (nous expliquerons les indices dans un instant), dans la région 2, c'est le vert, dans la 4, le bleu. Nous pourrons alors choisir comme base du triangle qui sert à calculer la pente comme la couleur dominante; et la hauteur comme la différence entre les deux autres couleurs. Plus exactement, nous utiliserons les différences au gris neutre de même luminosité (qui est au centre de la figure). Nous estimerons que la luminosité est donnée par

$$M = \max(r, g, b),$$

tandis que la portion de gris neutre est donnée par

$$m = \min(r, g, b),$$

1. Les instructions trigonométriques peuvent prendre des dizaines ou des centaines de cycles, même sur les processeurs les plus récents et les plus sophistiqués. Agner Fog rapporte de 11 à 82 cycles pour l'architecture Ryzen, alors qu'une addition en virgule flottante, en comparaison, ne prend qu'un cycle [130, p. 93].

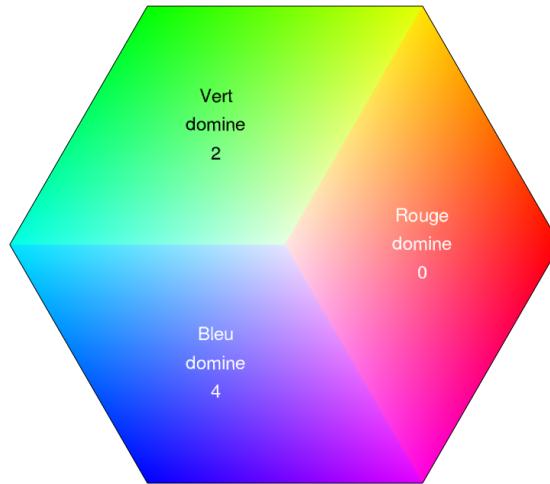


FIGURE 3.4.6 — Les régions de couleurs dans l’hexagone *HSV*.

et on pourra dire que la saturation est $\approx M - m$.

Ainsi, nous pouvons écrire une couleur (r, g, b) comme $m + (r - m, g - m, b - m)$; comme nous l’avions fait avec *CYMK*. La portion $(r - m, g - m, b - m)$ peut être interprétée comme la différence au gris (m, m, m) . Prenons par exemple une couleur où le rouge domine, nous aurons $M = r$, et la pente (qu’on estime être \approx l’angle) est donné par

$$\frac{(g - m) - (b - m)}{r - m} = \frac{g - m - b + m}{r - m} = \frac{g - b}{r - m} = \frac{g - b}{M - m}.$$

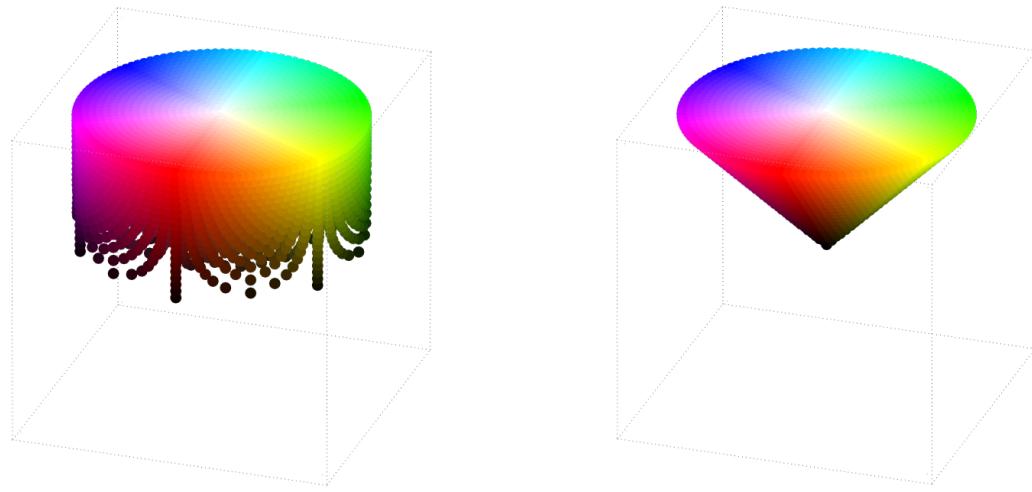
Pour cette couleur, nous avons

$$\begin{aligned} h &\approx \frac{g - b}{r - m}, \\ s &\approx M - m, \\ v &\approx M. \end{aligned}$$

On pourra par la suite normaliser h de façon à tenir, par exemple, entre 0 et 1, ou 0 et 6 (c’est un détail d’implémentation). Pour une implémentation complète, il faut bien entendu tester dans quel secteur se trouve la couleur dominante. Pour donner l’illusion d’un angle, on pourra avoir les « angles » dans $]-1, 1]$ pour le secteur rouge, dans $]1, 3]$ pour le secteur vert, et dans $]3, 5[$ pour le secteur bleu. On peut ajuster les « angles » rouges de façon à ce qu’ils soient soient entre -1 et 1 , ou entre 0 et 2 ; et tous les angles entre -1 et 5 , ou, comme nous le faisons ici, entre 0 et 6 . La fig. 3.4.7 montre une implémentation complète en *Mathematica*. L’implémentation est fortement inspirée de [126, 336]. Le mapping des couleurs *RGB* dans l’espace *HSV* cylindrique est montré à la fig. 3.4.8 (a); tandis que la fig. 3.4.8 (b) montre l’espace conique.

```
(* HSV cylindrique *)
RGBtoHSV[{r_, g_, b_}] :=
Module[{M = Max[r, g, b], m = Min[r, g, b], h,
s, v, d},
v = M;
s = If[M != 0,  $\frac{M-m}{M}$ , 0];
(* ôter la normalisation sur M donne le
modèle conique *)
d = M - m;
h = If[s == 0, 0,
If[r == M, Mod[(g - b) / 6, 6],
(* façon de faire +6 si négatif *)
If[g == M, 2 + (b - r) / d,
4 + (r - g) / d]]];
{s, v, v} (* normalisation de h ici...
ou non *)]
]

HSVtoRGB[{h_, s_, v_}] := Module[{i, f, p, q, t},
i = Floor[h];
f = h - i;
p = v (1 - s); (* si s=0, alors achromatique *)
q = v (1 - s f);
t = v (1 - s (1 - f));
Which[
i == 0, {v, t, p},
i == 1, {q, v, p},
i == 2, {p, v, t},
i == 3, {p, q, v},
i == 4, {t, p, v},
i == 5, {v, p, q}]
]
```

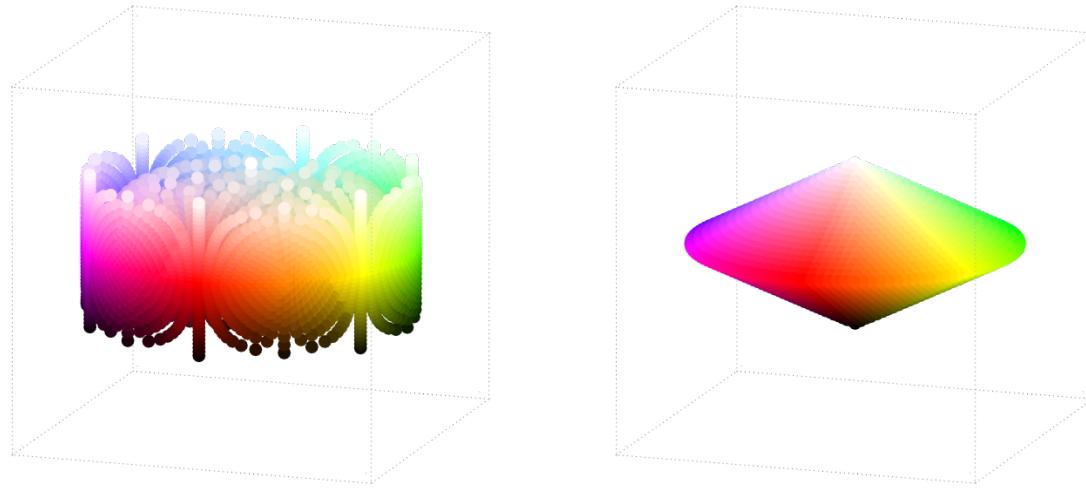
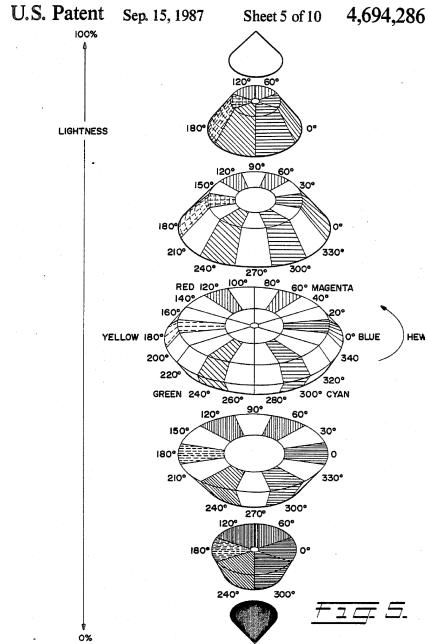
(a) Conversion de *RGB* à *HSV*.(b) Conversion de *HSV* à *RGB*.**FIGURE 3.4.7** — Une implémentation *Mathematica* des conversions de *RGB* à *HSV*.(a) *HSV* cylindrique(b) *HSV* conique.**FIGURE 3.4.8** — Les variantes de l'espace de couleurs *HSV*.

3.4.1.2 *HSL*

L'espace de couleurs *HSL* est si semblable à l'espace *HSV* qu'il semble que certains le confondent. En fait, même s'ils sont très semblables, ils diffèrent en deux points importants : comment ils estiment la luminosité et la saturation. Dans l'espace *HSV*, tel que présenté par Fishkin [126], on calcule la luminosité comme

$$l = \frac{1}{2}(\max(r, g, b) - \min(r, g, b)) = \frac{1}{2}(M - m)$$

tandis que d'autres implémentations posent plutôt $l = \frac{1}{3}(r + g + b)$. Le calcul de la saturation est

(a) *HSL cylindrique*(b) *HSL conique*.**FIGURE 3.4.9** — Les variantes de l'espace de couleurs *HSL*.**FIGURE 3.4.10** — *HSL*, dans le brevet de Bergstedt [60].

passablement plus compliqué :

$$s = \begin{cases} \frac{M-m}{M+m} & \text{si } l \leq \frac{1}{2} \\ \frac{M-m}{2-M-n} & \text{sinon.} \end{cases}$$

La transformation du cube *RGB* par l'algorithme de Fishkin disperse les points comme à la

fig. 3.4.9 (a), en espèce de couronne, où les couleurs très pâles ou très foncées sont dispersées dans des régions ténues de l'espace *HSL*. Au contraire, le centre se trouve densément peuplé de couleurs saturées. Ce que nous voulons, c'est avoir un espace également densément peuplé, comme à la fig. 3.4.9 (b) ou encore comme à la fig. 3.4.10.

Pour ce faire, nous allons modifier le calcul de la saturation, laissant le calcul de la luminosité inchangé. Si on pose plutôt

$$s = \max(r, g, b) - \min(r, g, b) = M - m,$$

nous obtenons bien la fig. 3.4.9 (b), mais il nous faut encore modifier le calcul de l'inverse. Il nous faut retrouver M et m à partir de l et s . Nous avons deux équations, deux inconnues,

$$\begin{aligned} l &= \frac{1}{2}(M - m) \\ s &= M - m. \end{aligned}$$

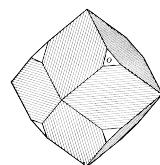
qu'il nous faut résoudre pour M et m . On trouve

$$\begin{aligned} M &= \frac{1}{2}(2l + s) \\ m &= \frac{1}{2}(2l - s). \end{aligned}$$

Il ne reste plus qu'à corriger l'implémentation de Fishkin pour obtenir le code de la fig. 3.4.11. Notons que l'implémentation revenir à l'original en changeant le calcul de s pour le code en commentaire dans la fig. 3.4.11 (a) et l'assignation de v pour le code en commentaire pour la fig. 3.4.11 (b).

*
* * *

Mais pourquoi insister sur les transformées coniques et ne pas se contenter des transformées cylindriques mêmes si elles n'utilisent pas très efficacement l'espace? Si on veut utiliser ces espaces de couleurs en compression de données, on ne veut pas assigner des codes à des combinaisons qui ne sont pas susceptibles d'être observées. Au contraire, on veut minimiser le nombre de bits émis, et donc adresser un espace aussi compact que possible. D'autres considérations entrent en jeu, comme la compaction d'énergie, dont nous discuterons amplement au chapitre ??.



```

Clear[HSLtoRGB]
HSLtoRGB[{h_, s_, l_}] :=
Module[{v, m1, m2, M1, M2, ss, i, f},
v =  $\frac{1}{2} (2 l + s)$ ;
(*If[l <=  $\frac{1}{2}$ , l(1+s), l+s(1-l)];*)
If[v == 0, {0, 0, 0},
m1 = 2 l - v;
ss = 1 -  $\frac{m1}{v}$ ;
i = Floor[h];
f = h - i;
M1 = m1 + v ss f;
M2 = v (1 - ss f);
Which[
i == 0, {v, M1, m1},
i == 1, {M2, v, m1},
i == 2, {m1, v, M1},
i == 3, {m1, M2, v},
i == 4, {M1, m1, v},
i == 5, {v, m1, M2}
]
]
]
]

(* Fishkin, VIII.14 dans Graphics Gems (I),
p. 448+ *)
(* modifiée pour double cone *)
Clear[RGBtoHSL]
RGBtoHSL[{r_, g_, b_}] :=
Module[{M = Max[r, g, b], m = Min[r, g, b], l, s, h},
l = (M + m) / 2;
If[M == 0, {0, 0, 0},
s = M - m; (* If[M!=m, If[l <=  $\frac{1}{2}$ ,  $\frac{M-m}{M+m}$ ,  $\frac{M-m}{2-M-m}$ ], 0];*)
h = If[s == 0, 0,
If[r == M, Mod[(g - b) / (r - m), 6],
(* façon de faire +6 si négatif *)
If[g == M, 2 + (b - r) / (g - m),
4 + (r - g) / (b - m)]]];
{l, s, h}
]
]
]

(a) Conversion de RGB à HSL.

```

(b) Conversion de *HSL* à *RGB*.

FIGURE 3.4.11 — Une implémentation *Mathematica* des conversions de *RGB* à *HSL*.

3.4.2 CIÉ $L^*a^*b^*$, $L^*u^*v^*$ et CIÉDE2000

Les espaces de couleurs présentés jusqu'à maintenant tentaient simplement de représenter les couleurs dans un espace où une des composantes représente la luminosité tandis que les deux autres donnent de l'information sur la saturation et la teinte, soit sous la forme de différentes jaune-bleu et rouge-vert, soit sous la forme explicite de teinte et de saturation. Mais ces espaces ne sont pas perceptuellement uniformes, c'est-à-dire que la magnitude du changement perçu dans la couleur dépend de la direction du changement. Autrement dit, ces espaces ne garantissent pas une variation d'égale magnitude selon qu'un petit changement Δ soit fait dans la première, la seconde ou la troisième composante.

Ce problème a été remarqué il y a déjà fort longtemps et plusieurs ont tenté de mesurer, le plus précisément possible, la quantité de changement nécessaire pour qu'on remarque une variation dans la luminosité ou la teinte [76, 77, 103, 258, 378, 382]. Ce plus petit changement perçu est la JND (*just noticeable difference*), et dépend de la couleur, comme nous le montre la fig. 3.4.12, créée à partir des données de MacAdam [245–251], et maintes fois revisitée [91, 153, 214]. L'expérience consistait « simplement » à faire varier x et y à partir d'une couleur de départ jusqu'à ce qu'une variation soit perçue, et marquer le changement nécessaire. Ensuite, une ellipse est estimée pour chaque couleur initiale. L'orientation et la forme des ellipses nous montrent clairement que l'espace de couleur

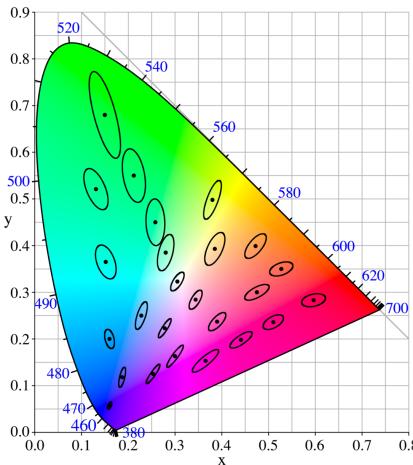


FIGURE 3.4.12 — Les ellipses de MacAdam (dessinées 10 fois plus grandes qu'en réel), d'après [245]. Source : Wikipedia.

CIÉ XYZ (ou xyz) n'est pas perceptuellement uniforme : on doit « marcher » moins loin dans une direction que dans une autre pour atteindre les limites des ellipses.

Un espace perceptuellement uniforme ne présenterait pas des ellipses, mais des cercles (ou des sphères si on considère les trois dimensions). Pour rendre l'espace de couleurs uniforme, il faut estimer les JND un peu partout dans cet espace, et compenser pour les déformations et les ramener à des cercles/sphères. Les espaces de couleurs $L^*a^*b^*$ et $L^*u^*v^*$ du CIÉ proposent justement une déformation de l'espace XYZ qui le rend « isotrope ».

3.4.2.1 $L^*a^*b^*$

Accepté comme standard par le CIÉ en 1976, l'espace $L^*a^*b^*$ est une transformation de l'espace XYZ pour le rendre uniforme [260, 261, 320]. L'espace $L^*a^*b^*$ dépend d'une fonction non linéaire f donnée par [145, 320] :

$$f(x) = \begin{cases} \sqrt[3]{x} & \text{si } x > \left(\frac{6}{29}\right)^3 \\ \frac{x}{3} \left(\frac{29}{6}\right)^2 + \frac{4}{29} & \text{sinon.} \end{cases} \quad (3.4.1)$$

tandis que l'inverse est donnée par

$$f^{-1}(z) = \begin{cases} z^3 & \text{si } z > \frac{6}{29} \\ 3\left(\frac{6}{29}\right)^2 \left(z - \frac{4}{29}\right) & \text{sinon.} \end{cases} \quad (3.4.2)$$

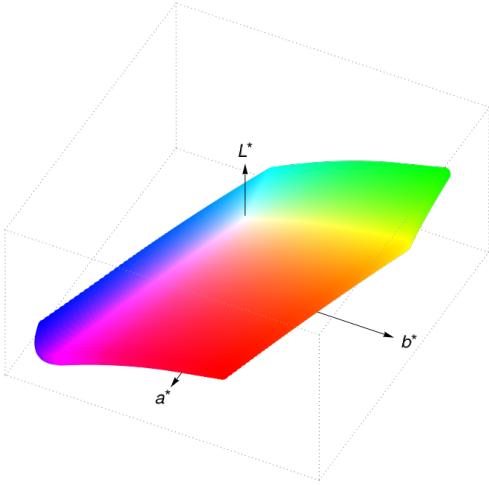


FIGURE 3.4.13 — La portion *RGB* de l'espace de couleurs $L^*a^*b^*$

En fonction d'un point blanc, par exemple D_{65} , dont les coordonnées XYZ sont $x_n = 95.4212$, $y_n = 100$ et $z_n = 18.8840$, et une couleur (x, y, z) on a les coordonnées (l^*, a^*, b^*) telles que

$$\begin{aligned} l^* &= 116f\left(\frac{y}{y_n}\right) - 16, \\ a^* &= 500\left(f\left(\frac{x}{x_n}\right) - f\left(\frac{y}{y_n}\right)\right), \\ b^* &= 200\left(f\left(\frac{y}{y_n}\right) - f\left(\frac{z}{z_n}\right)\right), \end{aligned} \quad (3.4.3)$$

où l^* est la luminosité, a^* correspond approximativement à la différence rouge-vert et b^* à la différence jaune-bleu. On revient à XYZ grâce à

$$\begin{aligned} x &= x_n f^{-1}\left(\frac{l^* + 16}{116} + \frac{a^*}{500}\right), \\ y &= y_n f^{-1}\left(\frac{l^* + 16}{116}\right), \\ z &= z_n f^{-1}\left(\frac{l^* + 16}{116} - \frac{b^*}{200}\right). \end{aligned} \quad (3.4.4)$$

Ces fonctions sont bricolées pour s'ajuster aux données recueillies pour mesurer les différences perçues et on été choisies parmi plusieurs alternatives [85, 261].

La portion *RGB* de l'espace de couleurs $L^*a^*b^*$ est montrée à la fig. 3.4.13. Cet espace, donc, est censé être perceptuellement uniforme. C'est-à-dire que toutes les modifications à l^* , a^* et b^* qui donnent la même différence

$$\Delta E^* = \sqrt{(l_1^* - l_2^*)^2 + (a_1^* - a_2^*)^2 + (b_1^* - b_2^*)^2}$$

donne la même sensation de différence. De plus, dans cet espace, $\Delta E^* \approx 2.3$ correspond à la JND d'un humain normal.

3.4.2.2 $L^*u^*v^*$

L'espace de couleurs CIÉ $L^*u^*v^*$, en quelque sorte une version simplifiée de l'espace $L^*a^*b^*$, est une évolution des espaces de couleurs (dont nous ne discuterons pas plus avant) CIÉ 1960 et CIÉ 1964 [199, 244] et [324, p. 81]. Pour convertir de XYZ à $L^*u^*v^*$, nous avons d'abord

$$\begin{aligned} u' &= \frac{4x}{12y - 2x + 3} \\ u'_n &= 0.2009 \\ v' &= \frac{9y}{12y - 2x + 3} \\ v'_n &= 0.4610 \end{aligned}$$

où u'_n et v'_n sont des constantes choisies en fonction du blanc C, puis

$$\begin{aligned} l^* &= 116f\left(\frac{y}{y_n}\right) - 16, \\ u^* &= 13l^*(u' - u'_n) \\ v^* &= 13l^*(v' - v'_n) \end{aligned} \tag{3.4.5}$$

où $f(x)$ est donnée par l'éq. (3.4.1) (et son inverse par l'éq. (3.4.2)¹) et y_n le y du blanc C. L'inverse est donné par

$$\begin{aligned} u' &= \frac{u^*}{13l^*} + u'_n \\ v' &= \frac{v^*}{13l^*} + v'_n \end{aligned}$$

puis

$$\begin{aligned} y &= y_n f^{-1}\left(\frac{l^* + 16}{116}\right) \\ x &= y \frac{9u'}{4v'} \\ z &= y \frac{12 - 3u' - 20v'}{4v'} \end{aligned} \tag{3.4.6}$$

La portion RGB de l'espace de couleurs $L^*u^*v^*$ est montrée à la fig. 3.4.14.

1. La version de cette fonction que l'on trouve dans Wikipedia (juillet 2020) est cassée ; le seuil à 8 (qui n'existe pas les versions du CIÉ) demande une constante supplémentaire pour que la partie cubique se joigne correctement à la partie linéaire.

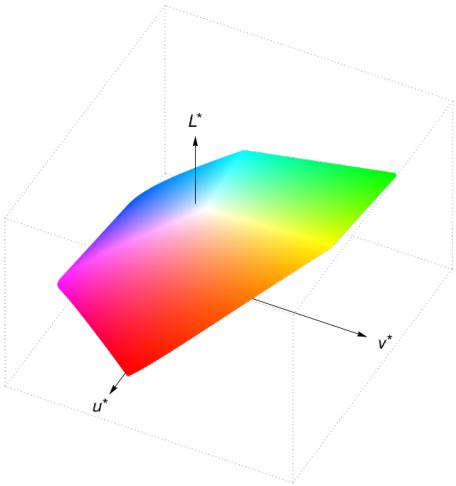


FIGURE 3.4.14 — La portion *RGB* de l'espace $L^*u^*v^*$

3.4.2.3 CIÉDE2000

Ce n'est pas à proprement parler un espace de couleurs, mais plutôt une métrique de différences de couleurs dans l'espace de couleurs $L^*a^*b^*$ [40, 243, 327]. La métrique est passablement plus compliquée que le ΔE^* de $L^*a^*b^*$ (elle fait une vingtaine d'équations comparativement à simple une distance euclidienne) mais est censée rendre encore beaucoup mieux les différences perçues lorsqu'on applique une petite différence de luminosité, de teinte, ou de saturation. Cette métrique a des visées industrielles où, dans certains processus, il peut être crucial de bien vérifier que les couleurs demeurent inchangées ou au moins à l'intérieur d'une certaine variation acceptable.

3.4.3 Autres espaces de couleurs

3.4.3.1 Munsell

Créé dans le but de donner des noms sans ambiguïté et très précisément aux couleurs, l'espace de couleurs de Munsell est un espace non linéaire qui découpe les couleurs selon un schéma classique de teinte, saturation (qu'il nomme « concentration ») et valeur [200, 264–266, 298]. Seule surprise, les couleurs primaires sont au nombre de 5 (rouge (R), jaune (Y), vert (G), bleu (B) et (P) pourpre) auxquelles viennent s'ajouter 5 couleurs intermédiaires (jaune-rouge (YR), jaune-vert (YG), bleu-vert (BG), bleu-pourpre (BP), et (PR) pourpre-rouge). Le cercle chromatique formé par ces couleurs est montré à la fig. 3.4.15. L'ensemble des couleurs (réalisables en *RGB*) est montré à la fig. 3.4.16, créée en utilisant les données du *Munsell Color Science Lab*, Rochester Institute of Technology.

En principe, le système de classement des couleurs dans l'espace de Munsell est méthodique (comme veut l'illustrer l'« arbre des couleurs », fig. 3.4.17), mais il comporte un défaut sérieux, du

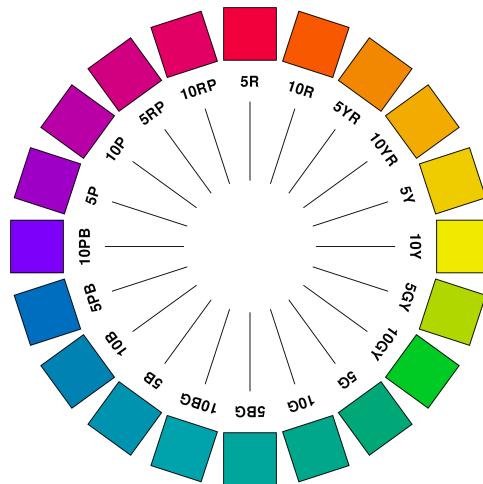


FIGURE 3.4.15 — Le cercle chromatique de l'espace de couleurs de Munsell. Source : Wikipedia.

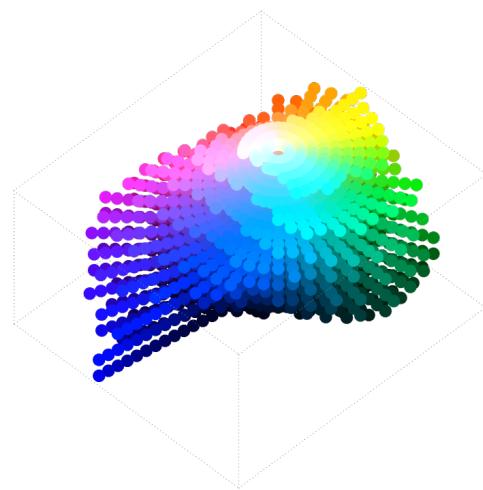


FIGURE 3.4.16 — L'espace de couleurs de Munsell (couleurs réalisables en *RGB*).

moins computationnellement : il n'y a pas de formule de conversion explicite, seulement une table établie entre codes de Munsell et des couleurs exprimées en coordonnées xyY (une variante de $X\bar{Y}Z$ CIÉ 1931). Il faut alors construire une table de RGB à xyY à Munsell, ce qui est assez peu pratique [274].

3.4.3.2 Ostwald

Dans les années 1900–1920, Ostwald a tenté d'établir l'étude des couleurs comme une discipline scientifique sujette à la formalisation mathématique [152, 285–287]. Il divise les couleurs en teintes selon un cercle chromatique dominé par quatre couleurs primaires. Il utilise les couleurs *geld* (jaune), *rot* (rouge), *ultramarin blau* (ultra-marin) et (cyan-verdâtre) *seegrün*, « vert marin »). Le cercle chromatique est montré à la fig. 3.4.18, tandis que le volume est montré à la fig. 3.4.19 (a). Ces couleurs

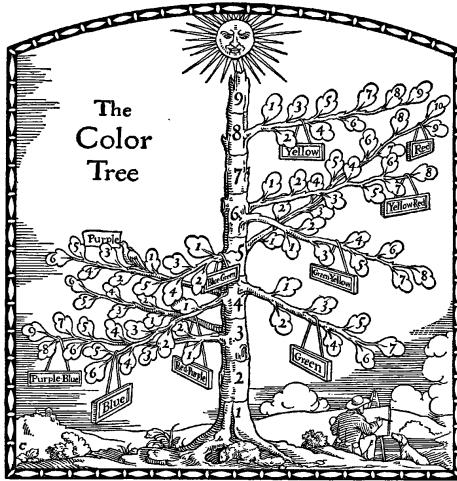


FIGURE 3.4.17 — L’arbre des couleurs de Munsell, tiré de [93] (domaine public).

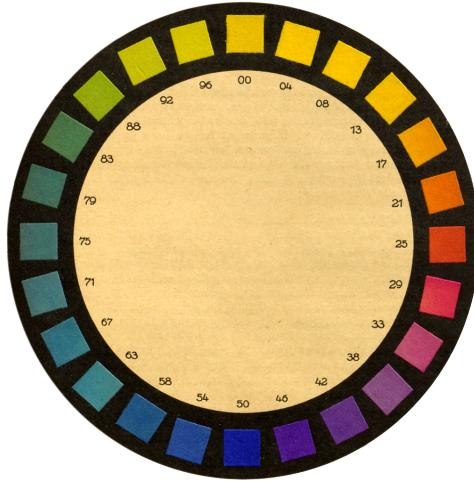


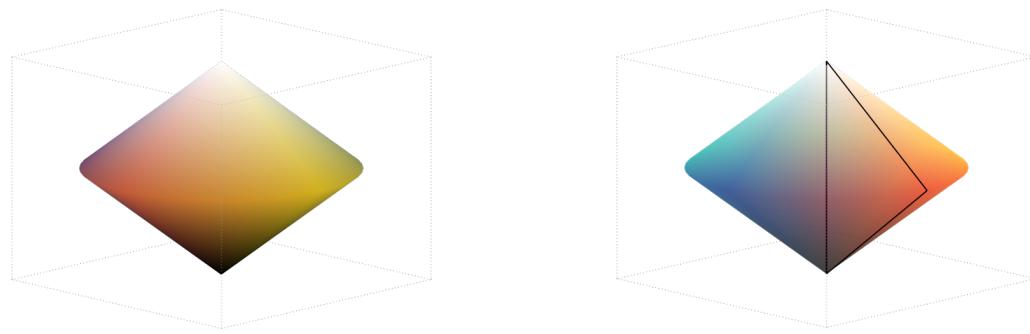
FIGURE 3.4.18 — Le cercle chromatique d’Ostwald, tiré de [287].

ont été vraisemblablement choisies en fonction des technologies de reproduction de couleurs qui lui étaient accessibles à l’époque.

Pour une teinte, il utilise un triangle équilatéral où un sommet est occupé par la couleur pure, la teinte, un autre par le noir (s pour *schwartz*, noir en Allemand) et le troisième par le blanc (w pour *weiß*, blanc en Allemand), ce triangle, comme le triangle de la fig. 3.1.2, interpole entre ces trois couleurs. Il impose, comme à la § 3.1.1, une normalisation des coordonnées. Ainsi, le mélange de couleur pure r , de blanc w et de noir s obéit à la contrainte

$$r + w + s = 100.$$

Les coordonnées dans le triangle sont exprimées par un système complexe et arbitraire : l’axe vertical est « numéroté » de a (blanc) à p (noir, p est la 16^e lettre de l’alphabet), la distance de la couleur pure est aussi notée de a (extérieur de la figure) à p (gris achromatique de même valeur),



(a) L'espace de couleurs Ostwald.

(b) Le triangle équilatéral.

FIGURE 3.4.19 — L'espace de couleurs Ostwald.

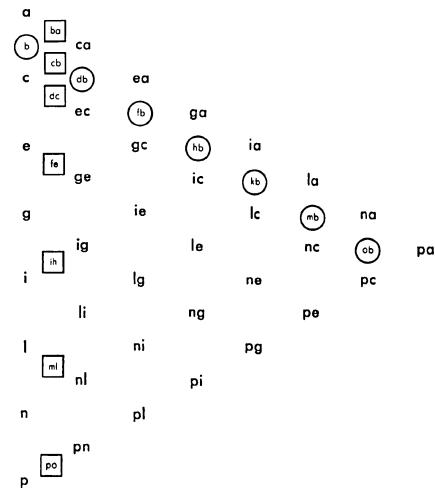


FIGURE 3.4.20 — Le système de coordonnées de l'espace Ostwald, d'après [152].

auxquelles s'ajoute un code pour la teinte. La fig. 3.4.20 donne une idée du système. Ce triangle, génère un volume de révolution (voir fig. 3.4.19 (b)) où le haut est dominé par le blanc, le bas par le noir, et l'« équateur » par les couleurs maximalement saturées.

3.4.3.3 NCS

L'espace de couleur NCS (pour *Natural Color System*) est basé sur quatre couleurs primaires issues de la théorie des couleurs complémentaires : une paire rouge-vert et une paire jaune-bleu [188, 189]. Comme pour *HSL*, l'espace forme un double cône dont les extrémités sont le blanc et le noir, et où les couleurs maximalement saturées sont réparties sur l'« équateur ». Les quatre couleurs primaires (qui ne sont pas sans rappeler les couleurs d'Ostwald) sont montrées à la fig. 3.4.21 (a) et sont données par

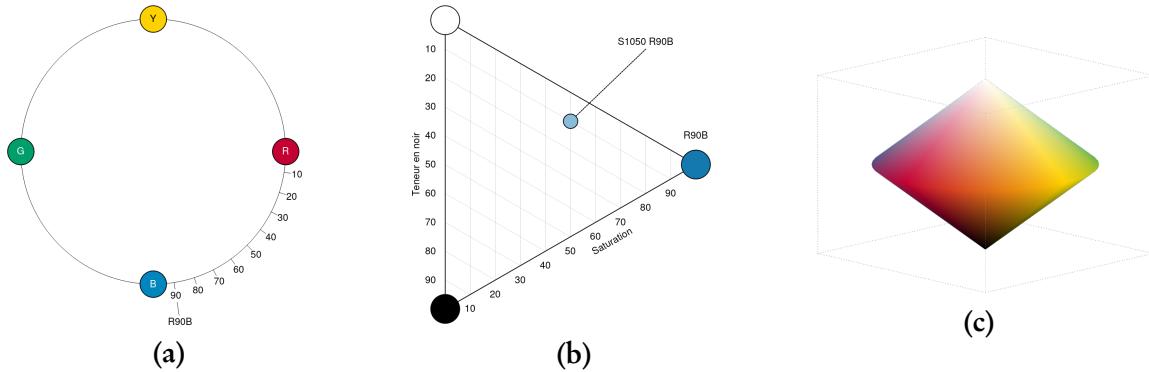


FIGURE 3.4.21 — L'espace de couleur NCS.

	RGB	Hexa
Blanc	1, 1, 1	ffffff
Rouge	0.77, 0.01, 0.2	c40233
Vert	0, 0.62, 0.42	009f6b
Bleu	0, 0.5, 0.74	0087bd
Jaune	1, 0.83, 0	ffd300
Noir	0, 0, 0	000000

Comme pour Ostwald, le système de coordonnées est basé sur la teinte, donnée par une des couleurs primaires R , Y , G , B . Ainsi la teinte $R90B$ représente 90 parts de B pour 10 de R , comme nous le montre la fig. 3.4.21 (a) qui définit un triangle entre le blanc, le noir et la couleur maximalement saturée. À l'intérieur de ce triangle (placé dans le volume de la fig. 3.4.21 (c)) comme à la fig. 3.4.19 (b)), on établit des coordonnées triangulaires, bien que plus simplement que pour le système d'Ostwald. NCS établit une grille où un axe est saturation et l'autre la teneur en noir; donnant des couleurs comme S1050 R90B qu'il faut décoder comme 10% noir, 50% saturée, de la teinte R90B.

3.5 Remarques bibliographiques

Les espaces de couleurs CIÉ $L^*a^*b^*$ et $L^*u^*v^*$ perceptuellement uniformes ne sont pas les premiers espaces « équisensibles ». En effet, un certain nombre d'essais les ont précédés. Certains, avant MacAdam, ont tenté d'estimer la sensibilité de l'œil aux variations de couleur et d'intensité [306, 358], tandis que d'autres encore ont proposé des reparamétrisations de l'espace de couleur CIÉ de façon à rendre mieux compte de la sensibilité visuelle [199]. En particulier, le système de coordonnées « en racine cubique » qui survivra jusque dans les espaces de couleurs CIÉ 1976 $L^*a^*b^*$ et $L^*u^*v^*$ [145].

*

* *

Pour un détail de l'histoire du développement de la télévision jusqu'à 1940, on pourra s'en référer à Fink [123]. La standardisation de la télévision couleur est arrivée en 1953, issue de travaux qui avaient débuté bien avant, dans les années 1930 [172, 389]. La standardisation a permis l'essor de la télévision, car auparavant, il y avait une pléthore de méthodes, farfelues ou astucieuses, mais toutes incompatibles entre elles [57, 357]. Après un développement des techniques qui permettent la transmission de la couleur [361, 362], il ne restait plus qu'à décider d'un standard pour rendre interopérables stations et récepteurs, indépendamment du fabriquant. Malgré tout, plusieurs standards se sont imposés : NTSC en Amérique du nord et Japon, SÉCAM en France, Russie et quelques pays francophones, et PAL ça et là [148].

*

* *

Les espaces de couleurs dont la transformée peut être calculée sans perte, c'est-à-dire qu'une couleur *RGB* est restituée à l'identique après un passage par un second espace de couleurs, sont intéressants dans certains cas particuliers où il est primordial de restituer l'image exactement à la décompression. S'il n'est pas nécessaire de sauvegarder vos photos de voyage au bit près, d'autres images demanderont d'être restituées *exactement*. Pour les applications d'archivage muséal, médicales, de télédétection ou spatiales, les images acquises à grand coût, riches en information, doivent être stockées et compressées sans perte. Par conséquent, ces applications ne peuvent s'accommoder de transformées de couleurs approximatives. Pour un traitement plus général des transformées de couleurs sans perte, voir [343, 344].

*

* *

Les théories sur la couleur ne datent pas d'hier. Déjà, au XII^e siècle Robert Grosseteste (c. 1168–1253), évêque de Lincoln de 1235 jusqu'à sa mort, s'est intéressé aux couleurs. Vers 1220 il écrit *De Colore*, un traité sur les couleurs où il propose essentiellement un modèle à deux cônes, avec le blanc et le noir aux sommets, mais avec sept couleurs primaires, les couleurs de l'arc-en-ciel sur son équateur [110, 338, 339]. Leon Battista Alberti (1404–1472), surtout connu pour son chiffre [202, 334], s'y est aussi intéressé. En 1435, dans *Della pittura*, il propose une double pyramide dont chaque coin correspond à l'une des quatre couleurs primaires, rouge, vert, bleu et jaune et dont les deux pointes correspondent au noir et au blanc [64]. L'organisation des couleurs en cercle chromatique (en « roue

des couleurs ») découle peut-être d'une observation judicieusement vague de Newton [276], reprise et formalisée par Boutet, qui sera le premier à donner explicitement le cercle chromatique [71].



La formulation explicite du concept de couleurs primaires serait peut-être due à Boyle, au XVII^e siècle [72]. L'idée d'utiliser des couleurs primaires en imprimerie s'est imposée assez tôt. Dès 1725, le Francfortois Jakob Christoffel Le Blon fait paraître, d'abord en anglais, puis en français, « L'art d'imprimer des tableaux » où il préconise l'utilisation des couleurs cyan, magenta et jaune, en plus du noir [227, 228]. L'idée, qui s'avérera néanmoins adéquate, repose tout de même alors sur des bases pifométriques. Le premier à s'interroger et à formaliser le problème des couleurs primaires optimales sera Benson, dans les années 1860 [58]. Pour une histoire plus-que-complète sur les espaces et modèles de couleurs, voyez *Color Ordered : A Survey of Color Order Systems, from Antiquity to the Present* [218].

*
* * *

La théorie des espaces de couleurs (qui nous intéresse ici par son application aux images numériques) n'est en fait qu'un sous-domaine du domaine plus vaste de la vision et de la perception des couleurs. Certains s'y intéressent à un niveau anatomique et biologique [35, 45], psychologique [106, 107, 135], physique [68, 323], esthétique voire frivole [33, 138], en général [79, 107], pratique [111], sous une perspective mathématique rigoureuse [115].

*
* * *

L'idée d'Ohta (voir § 3.3.5) d'utiliser les axes principaux du nuage de couleurs pour construire les vecteurs de base de son espace de couleur n'est pas bête du tout, mais computationnellement non triviale, et il opte plutôt pour une approximation qu'il dit être satisfaisante. Cependant, il existe des techniques pour obtenir exactement les axes optimaux d'un nuage de points en un nombre quelconque de dimensions : il s'agit de l'analyse en composante principale, aussi connue sous le nom de transformée de Karhunen-Loève et transformée d'Hotelling selon les auteurs et les domaines [198, 204, 237, 293]. On pourrait argumenter que sur une machine à peu près moderne, calculer les composantes principales d'un nuage de points en trois dimensions n'est plus prohibitif, car il ne s'agit que de trouver les vecteurs propres (*eigenvectors*) d'une matrice de covariance 3×3 .

*
* * *

Pourquoi JPEG 2000 n'a-t-il pas su s'imposer malgré sa supériorité technique? D'une part, le standard JPEG 2000 est nettement plus complexe que le standard JPEG « classique », et son implémentation efficace plus difficile. D'autre part, les librairies ont mis du temps à être disponibles, étaient peu portables ou propriétaires (par opposition à « en logiciel libre »), et demandaient, pour l'époque, au tournant du millénaire, trop de mémoire et étaient lentes. Même aujourd'hui, le support pour JPEG2000 est incomplet et difficile à mettre en œuvre... Enfin, pour la plupart des gens, JPEG2000 n'apporte aucun avantage évident par rapport à JPEG. En effet, pour s'échanger des images par e-mail, JPEG fait bien l'affaire.

*
* * *

Le système NCS utilise quatre couleurs car Hering croyait que le jaune était une couleur primaire et non un mélange d'autres couleurs [174]. Le choix des couleurs m'est plus nébuleux pour l'espace de Munsell [279–281], que l'on peut voir en toutes couleurs [93]. On pourra aussi se surprendre que les couleurs primaires choisies ne soient pas tout à fait vives : *seegrün* et *geld* sont des couleurs que l'on voit comme composées dès le départ.

L'espace d'Ostwald, que nous avons présenté à la section § 3.4.3.2, est ordinairement présenté comme à la fig. 3.4.19 (a), mais si on observe attentivement les documents, on voit que l'axe vertical, qui correspond à la luminosité est toujours montré selon une échelle logarithmique [152], malgré qu'Ostwald lui-même ait initialement utilisé une échelle linéaire [285–287]. Si on montre l'espace d'Ostwald dans sa version 1942 selon une échelle linéaire, on obtient l'oignon de la fig. 3.5.1 (b), qui est nettement plus étrange.

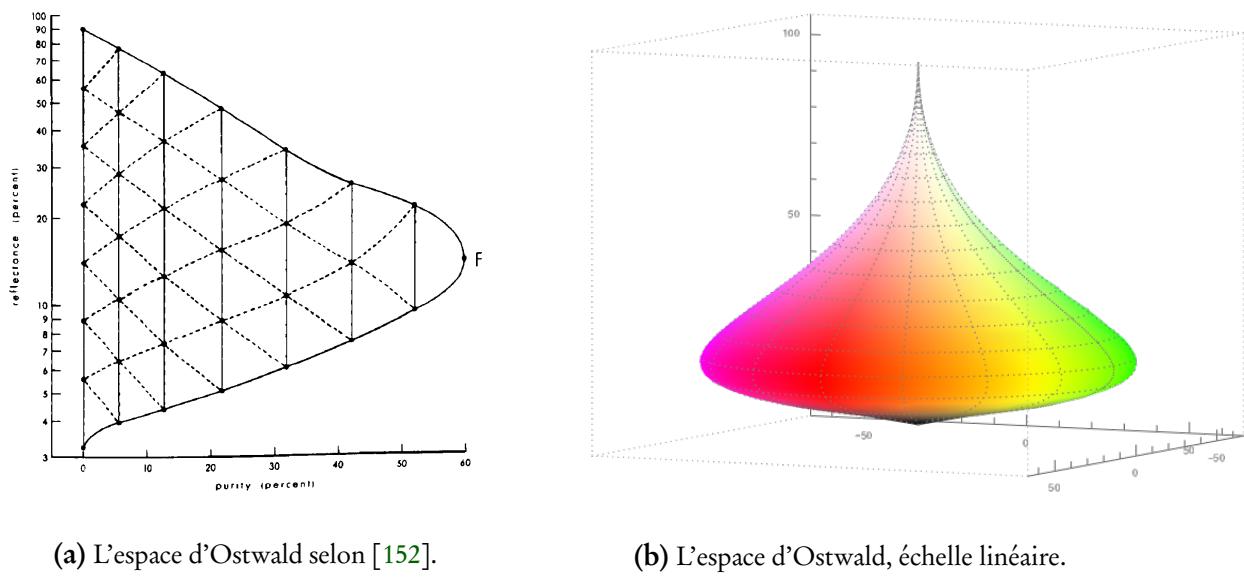


FIGURE 3.5.1 — L'espace d'Ostwald et sa correction logarithmique.

4

Discrétisation des couleurs

Sommaire. Dans ce chapitre, nous explorons le problème de la discrétisation des couleurs, où il s'agit de trouver k bonnes couleurs représentatives pour une image qui peut en comporter un grand nombre. Nous commencerons par discuter des métriques, nécessaires aux algorithmes adaptatifs. Nous présenterons les méthodes de discrétisation directes, qui se contentent de mettre en correspondance les pixels en haute résolution de couleurs avec des couleurs limitées et fixes. Nous verrons ensuite les méthodes adaptatives qui tirent profit (pas toujours très bien) de la distribution des couleurs dans les images pour obtenir une palette réduite mais optimisée.

4.1 Introduction

Au chapitre 1, nous avons vu que certains modes graphiques ne permettent d'afficher seulement qu'un petit nombre de couleurs simultanément, et que ces couleurs peuvent être tirées d'une palette fixe (sur le matériel plus ancien) ou d'une palette adaptative tirée d'une gamme de couleurs assez vaste. Donc si nous avons une image en « millions de couleurs » que nous devons afficher avec l'un de ces modes, il faudra réduire le nombre de couleurs de façon à être compatible avec le matériel.

Mais la question se pose : comment choisir, parmi ces millions de couleurs, quelques dizaines, ou quelques centaines de couleurs — typiquement 256 — pour rendre une image tout de même fidèle à l'original? Ce chapitre tentera de répondre à cette question. Nous verrons que les méthodes sont divisées en essentiellement deux catégories, soit les méthodes directes, qui réduisent la résolution des couleurs pour se plier directement au matériel, et les méthodes adaptatives qui calculeront une palette de couleurs idoine à l'image considérée. Ces dernières méthodes sont notamment plus complexes, mais donnent de bien meilleurs résultats.

Les méthodes adaptatives que nous verrons ici, expressément conçues pour la discrétisation des couleurs, se généralisent, pour la plupart, à un nombre quelconque de dimensions — plutôt que seulement trois — et s'insèrent dans la problématique plus vaste de la discrétisation de vecteurs sous un critère de fidélité.

Or, justement, ce critère de fidélité, qui mesure la qualité de la discrétisation, suppose que nous puissions mesurer une « distance » entre deux couleurs, c'est-à-dire que nous puissions établir une *métrique*. Ce critère de fidélité peut être généralisé à l'image en entier, comme nous le verrons à la § 4.2.2

4.2 Métriques et fidélité

4.2.1 Métriques

Un *espace métrique* est un ensemble (ou un espace vectoriel) où une notion de distance est définie entre deux éléments de cet espace. Cette fonction de distance, pour être une *métrique*, doit satisfaire les conditions suivantes :

1. $d(x, y) = 0$ ssi $x = y$ (« indiscernabilité des identiques »),
2. $d(x, y) = d(y, x)$ (symétrie),
3. $d(x, z) \leq d(x, y) + d(y, z)$ (inégalité triangulaire),
4. $d(x, y) \geq 0$ (pas de distances négatives).

La métrique peut par ailleurs satisfaire des propriétés supplémentaires (comme modifier l'inégalité triangulaire par $d(x, z) \leq \max(d(x, y), d(y, z))$ pour obtenir une *ultramétrique*). Si n'importe quelle fonction qui satisfasse les quatre conditions ci-dessus est une métrique, on se limite généralement aux métriques suivantes :

1. L_1 , dite « distance de Manhattan » (en anglais on trouve *Manhattan distance* mais aussi *taxis distance*¹). On a

$$d(x, y) = \|x - y\|_1 = |x - y| = \sum_{i=1}^n |x_i - y_i|$$

où $|u|$ est la valeur absolue.

1. C'est censé rappeler les distances entre deux points dans une ville où les rues sont arrangées en grille orthogonales, comme New York [217]

2. L_2 , dite « euclidienne », c'est la distance « ordinaire »,

$$\begin{aligned} d(x, y) &= \|x - y\|_2 \\ &= \sqrt{(x - y)^2} \\ &= \sqrt{(x - y)^T(x - y)} \\ &= \sqrt{\sum_{i=1}^n (x_i - y_i)^2}. \end{aligned}$$

3. L_∞ , dite « uniforme »,

$$d(x, y) = \|x - y\|_\infty = \max\{|x_i - y_i|\}_{i=1}^n.$$

4. L_p , « norme généralisée »,

$$d(x, y) = \|x - y\|_p = \sqrt[p]{\sum_{i=1}^n |x_i - y_i|^p}.$$

On voit comment la norme L_p a pour cas spéciaux L_1 , L_2 et L_∞ .

Ces normes supposent un espace isotropique, c'est-à-dire dans lequel il n'y a pas de dimension préférée. Or, avec les espaces de couleurs, qui forment un espace à trois dimensions pour la plupart, ce n'est généralement pas le cas puisque, comme nous l'avons vu au chapitre 3, nous sommes plus sensibles aux variations dans le vert que dans le rouge ou le bleu. Si nous sommes encore dans l'espace RGB, nous pourrions pondérer les dimensions avec des coefficients qui reflètent notre sensibilité à chaque couleur primitive. Posons α , le coefficient pour le rouge, β pour le vert et γ pour le bleu. Nous pouvons créer

$$W = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix},$$

avec des valeurs particulières pour un espace de couleur considéré, par exemple RGB BT.709, c'est-à-dire sRGB (voir § 3.3.9.3, p. 66)

$$W_{BT.709} = \begin{bmatrix} 0.2126 & 0 & 0 \\ 0 & 0.7156 & 0 \\ 0 & 0 & 0.0722 \end{bmatrix}.$$

Si nous restons dans L_2 , la distance devient

$$\begin{aligned} d(x, y) &= d(Wx, Wy) \\ &= \sqrt{(W(x - y))^2} \\ &= \sqrt{(W(x - y))^T(W(x - y))} \\ &= \sqrt{(x - y)^T W^T W (x - y)}. \end{aligned}$$

Évidemment, cette transformation peut être différente selon l'espace de couleur considéré... ou même inexistante si l'espace de couleur est uniforme, comme c'est le cas pour $L^*a^*b^*$.

4.2.2 Estimation de la fidélité

Pour estimer la qualité d'une image dont les couleurs ont été discrétisées relativement à l'image originale, il faut en quelque sorte calculer la distance entre ces deux images dans un espace vectoriel dont le nombre de dimensions est proportionnel au nombre de pixels fois le nombre de composantes par pixel; c'est bien sûr souvent compliqué de procéder ainsi. C'est pourquoi en général, nous allons calculer la différence entre deux images comme une fonction de la somme de la différences des pixels individuels.

C'est exactement ce que l'erreur quadratique moyenne, le MSE (*mean square error*) propose de faire. Pour une image originale A de dimensions $n \times m$ et une image discrétisée \tilde{A} de même dimensions, nous avons

$$MSE(A, \tilde{A}) = \frac{1}{nm} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \|A_{ij} - \tilde{A}_{ij}\|_2^2, \quad (4.2.1)$$

où $\|x - y\|_2$ est la distance euclidienne, la norme L_2 .

Dans le domaine du traitement de signal, et jusqu'à assez récemment pour les images, on utilisait plus souvent le rapport du signal au bruit (en anglais, SNR, pour *signal to noise ratio*). Plus précisément, c'est le rapport de la puissance du signal à la puissance du bruit, toutes deux des mesures quadratiques¹.

Cette mesure est donnée par

$$SNR(A, \tilde{A}) = \frac{\mathbb{E}[A^2]}{\mathbb{E}[(A - \tilde{A})^2]} = \frac{P_{\text{signal}}}{MSE(A, \tilde{A})} = \frac{\mu^2}{\sigma^2}, \quad (4.2.2)$$

où $\mathbb{E}[X]$ est l'espérance mathématique². Si on veut exprimer le SNR en décibels (voir la discussion § 4.5), on utilisera

$$SNR_{\text{db}}(A, \tilde{A}) = 10 \log_{10} SNR(A, \tilde{A}). \quad (4.2.3)$$

Nous trouvons aussi souvent le PSNR (*peak signal to noise ratio*, qu'il faut lire comme (*peak signal*) *to noise ratio*). Ce dernier est donné par

$$PSNR(A, \tilde{A}) = \frac{\max(A^2)}{MSE(A, \tilde{A})}, \quad (4.2.4)$$

1. La puissance du signal est souvent exprimée en Watts, lesquels sont proportionnels au carré du voltage. En effet, si P est la puissance, I le courant en ampères, V le voltage et $R = \frac{V}{I}$ la résistance en ohms, nous avons $P = IV = I^2R = \frac{V^2}{R}$.
2. Pour un échantillon fini, l'espérance est estimée à partir de la moyenne d'échantillon.

où $\max(A^2)$ est le plus grand A_{ij}^2 , ou encore la valeur maximale possible pour un pixel (comme 255 par exemple). En décibels, on aura

$$\begin{aligned} PSNR_{\text{db}}(A, \tilde{A}) &= 10 \log_{10} PSNR(A, \tilde{A}) \\ &= 10 \log_{10} \frac{\max(A^2)}{MSE(A, \tilde{A})} \\ &= 20 \log_{10} \frac{\max(A)}{\sqrt{MSE(A, \tilde{A})}}. \end{aligned} \quad (4.2.5)$$

*
* * *

Si les mesures de fidélité telles que le SNR et le PSNR sont fréquemment rencontrées dans la littérature, celles-ci ne sont pas spécialement efficaces pour mesurer la qualité perçue des images, car elles sont sensibles à certaines transformations qui n'affectent pas notre perception de la qualité de l'image.

Par exemple, si on ajoute 1 à chaque pixel ou qu'on décale toute l'image d'un pixel à gauche, il est peu probable qu'un observateur moyen s'objecte à la transformation. L'image, de son point de vue, demeure parfaite. Cependant, le SNR chutera significativement. C'est pourquoi, au fil des ans, des mesures plus sophistiquées ont été développées. Ces nouvelles techniques, comme la similarité structurelle [209, 328, 329, 368, 369], vont chercher à estimer la qualité perçue en réduisant, ou éliminant, les effets des transformations comme ± 1 et les petits décalages dans l'image.

*
* * *

Ceci étant dit, la plupart des méthodes adaptatives que nous présenterons ici minimiseront l'erreur quadratique moyenne, c'est-à-dire le MSE, simplement parce que c'est computationnellement plus simple.

4.3 Méthodes directes

Dans les sections et les qui suivent, nous verrons comment les méthodes de discréétisation des couleurs affectent les images. Nous l'utiliserons l'image « coucher de soleil », montrée à la fig. 4.3.1, comme image-étalon.



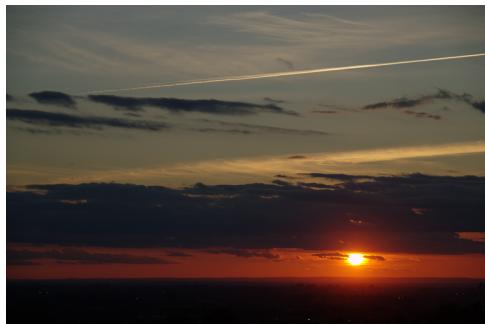
FIGURE 4.3.1 — L'image de référence « coucher de soleil ». Image : Steven Pigeon

Bits	Organisation	Exemple
3	u r g b	Teletex Level 1 (1976) [10]
6	u,u r,r g,g b,b	EGA
8	r,r r,g g,g b,b	MSX-2 [3]
9	u,u u,u u,r r,r u,g g,g u,b b,b	Atari ST [295, p 2-13]
12	u,u u,u r,r r,r g,g g,g b,b b,b	Amiga 1000 [294, p 2-13]
15	u,r r,r r,r g,g g,g b,b b,b b	SVGA
16	r,r r,r r,r g,g g,g g,g b,b b,b b	SVGA
24	r,r r,r r,r r,r g,g g,g g,g b,b b,b b,b	VESA

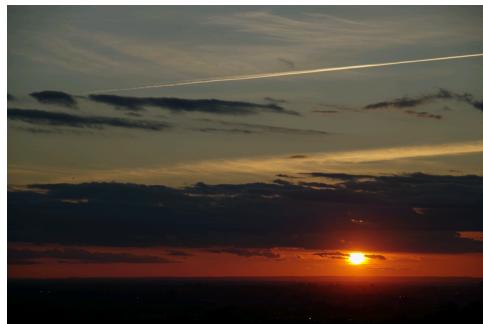
TABLE 4.3.1 — Différentes résolutions de couleurs. Légende : u (inutilisé), r,g,b, bits qui participent respectivement aux composantes rouges, vertes, bleues.

4.3.1 16/15/12/9/8/6/3 bits par pixel

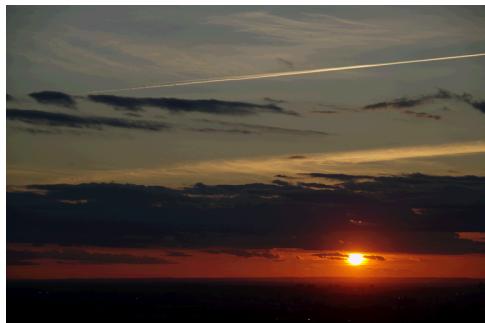
La table 4.3.1 nous montre quelques résolutions de couleurs susceptibles d'être supportées directement par le matériel. Une méthode de discréétisation directe des couleurs réduirait directement le nombre de bits par composante, en conservant les bits les plus significatifs, pour correspondre à la résolution de couleur désirée. La fig. 4.3.2 nous montre le résultat de la discréétisation d'une image 24 bits (fig. 4.3.2 (a)), en plusieurs résolutions de couleurs. On y voit qu'à 16, 15, et 12 bits par pixel (figs. 4.3.2 (b) à 4.3.2 (d)), les images sont assez satisfaisantes, bien qu'à 12 bits, les faux contours sont bien visibles — nous verrons dans un chapitre ultérieur comment réduire l'effet des faux contours grâce à la diffusion d'erreur, une technique absolument nécessaire lorsque la résolution, ou le nombre, de couleurs est faible. Sans surprise, plus la résolution des couleurs descend (figs. 4.3.2 (e) à 4.3.2 (h)), plus l'image est abîmée.



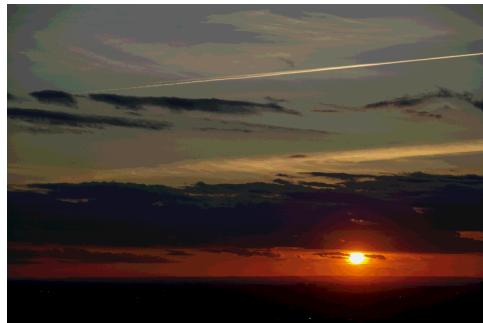
(a) Image originale en 24 bits par pixel.



(b) Image en 16 bits par pixel (5:6:5).



(c) Image en 15 bits par pixel (1:5:5:5).



(d) Image en 12 bits par pixel (4:4:4).



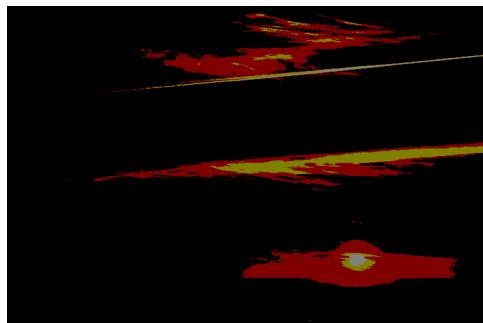
(e) Image en 9 bits par pixel (3:3:3).



(f) Image en 8 bits par pixel (3:3:2).

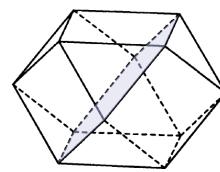
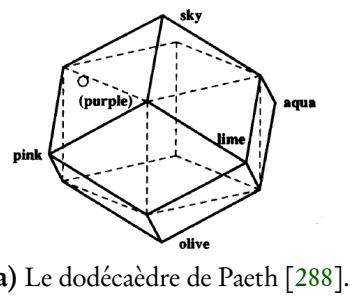


(g) Image en 6 bits par pixel (2:2:2).



(h) Image en 3 bits par pixel (1:1:1).

FIGURE 4.3.2 — Les effets d'une discréétisation directe, avec un nombre variable de bits par pixel.



(b) Le polyèdre dual, le cuboctaèdre, montrant un plan de coupe [288].

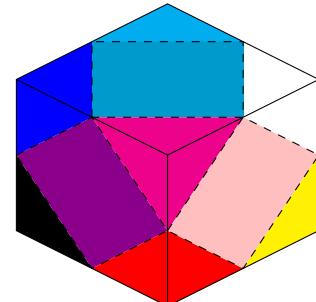
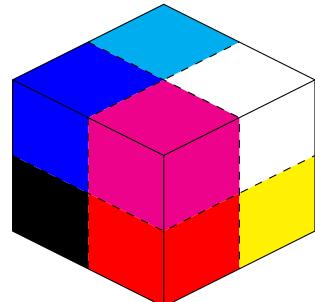


FIGURE 4.3.3 — Le cube de Paeth.

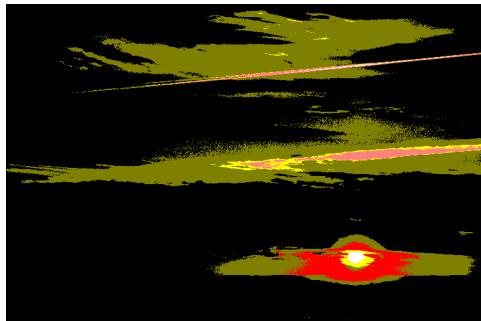


FIGURE 4.3.4 — La méthode de Paeth.

4.3.2 Méthode de Paeth

Pour discréteriser rapidement les images, Paeth propose une méthode originale et simple à implémenter [144, 288]. Il remarque qu'un cube RGB avec seulement un bit par composante (ce qui nous donne seulement 8 couleurs, comme on le voit à la fig. 4.3.3 (c)) est un peu trop cru pour le rendu des images. Il propose d'augmenter le cube en ajoutant à chaque face une région centrale qui correspond à une couleur intermédiaire, ce qui nous donne maintenant 14 couleurs — le centre du cube n'est pas utilisé.

Paeth propose d'utiliser une astuce basée sur les polyèdres duals pour effectuer la conversion

b_3	b_2	b_1	b_0	R	G	B	Nom
0	0	0	0	0	0	0	Noir
0	0	0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	Olive
0	0	1	0	$\frac{1}{2}$	0	$\frac{1}{2}$	Pourpre
0	0	1	1	1	0	0	Rouge
0	1	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	Aqua
0	1	0	1	0	1	0	Vert
0	1	1	0	0	0	1	Bleu
0	1	1	1	-	-	-	Inutilisé
1	0	0	0	-	-	-	Inutilisé
1	0	0	1	1	1	0	Jaune
1	0	1	0	1	0	1	Magenta
1	0	1	1	1	$\frac{1}{2}$	$\frac{1}{2}$	Rose
1	1	0	0	0	1	1	Cyan
1	1	0	1	$\frac{1}{2}$	1	$\frac{1}{2}$	Lime
1	1	1	0	$\frac{1}{2}$	$\frac{1}{2}$	1	Ciel
1	1	1	1	1	1	1	Blanc

TABLE 4.3.2 — Couleurs de la méthode de Paeth.

efficacement. Le cube augmenté de la fig. 4.3.3 (d) pourrait être vu comme un dodécaèdre rhombique (fig. 4.3.3 (a)) dont le polyèdre dual est le cuboctaèdre (fig. 4.3.3 (b)). Il présente une méthode assez compliquée pour couper l'espace en plans pour tester dans quelle région du dodécaèdre rhombique se trouve la couleur afin d'effectuer la discréétisation. Heureusement, il y a beaucoup plus simple! Si on organise les 14 couleurs comme à la table 4.3.2, on peut faire l'assignation

$$\begin{aligned} b_3 &= (r + g + b) > \frac{3}{2} \\ b_2 &= (g + b - r) > \frac{1}{2} \\ b_1 &= (r + b - g) > \frac{1}{2} \\ b_0 &= (r + g - b) > \frac{1}{2} \end{aligned}$$

en considérant que la comparaison retourne 1 si elle est vérifiée et 0 sinon.

La fig. 4.3.4 (a) montre l'image de la fig. 4.3.2 (a) discréétisée par l'algorithme de Paeth. Sans surprise, même avec 14 couleurs, la méthode reste trop crue pour être utilisée telle quelle, sans diffusion d'erreur. La fig. 4.3.4 (b) nous montre le résultat avec la diffusion d'erreur. Ce sont les mêmes 14 couleurs, mais l'erreur générée par la discréétisation d'un pixel est distribuée sur ses voisins qui sont maintenant susceptibles d'être discréétisés différemment.

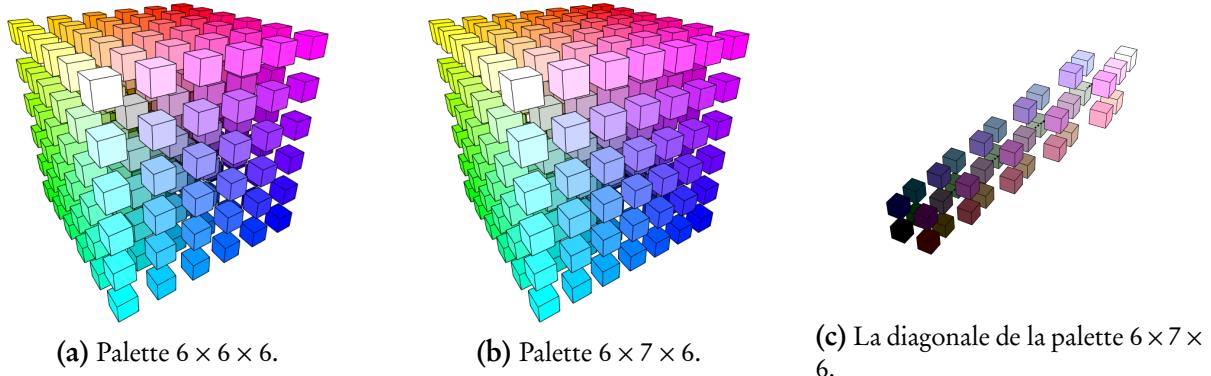


FIGURE 4.3.5 — Les palettes *web safe*, ou presque.

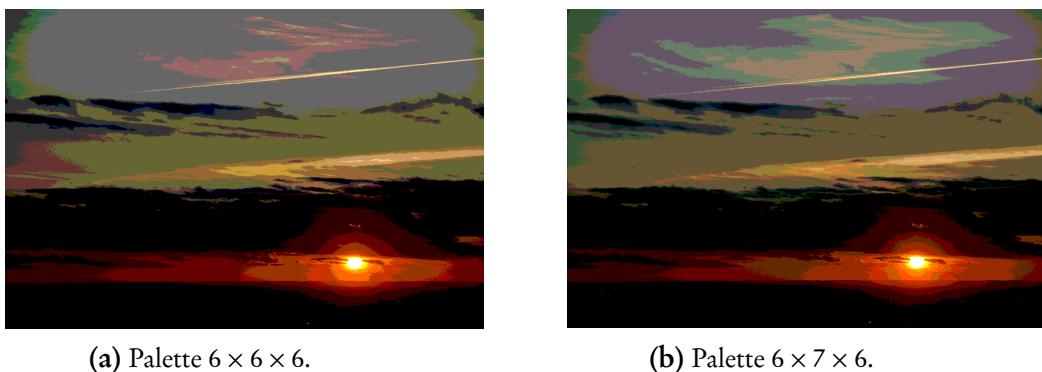


FIGURE 4.3.6 — Les palettes *web safe*, ou presque, appliquées aux images.

4.3.3 666, la palette du diable

Nous avons déjà rencontré la *web safe palette* au chapitre 1. Cette palette, qui espace régulièrement les couleurs dans un cube de $6 \times 6 \times 6 = 216$ couleurs, a été conçue pour permettre un affichage en mode palette de 256 couleurs, tout en laissant 40 couleurs réservées pour l'interface graphique. Cela permet d'afficher des images sans avoir à reprogrammer les couleurs de l'interface, un compromis élégant pour les modes graphiques en 256 couleurs avec palette. La palette est montrée à la fig. 4.3.5 (a), et son résultat à la fig. 4.3.6 (a).

S'il est évident que $6 \times 6 \times 6 = 216$ est un compromis acceptable lorsqu'on veut conserver quelques couleurs libres, il est aussi évident que 216 couleurs n'utilisent pas maximalement les 8 bits nécessaires pour les encoder — puisque 7 bits seraient insuffisants pour représenter toutes les combinaisons. Comme un code optimal devrait contenir *tout juste assez* de bits pour représenter ces 216 combinaisons, $\log_2 216 \approx 7.75$ bits devraient suffire ! En utilisant 8 bits, nous « gaspillons » donc ≈ 0.25 bits par rapport à un code optimal ! Si nous voulons mieux utiliser nos 8 bits, il nous faut une palette avec plus de couleurs — idéalement 256. Or, Une variante de la palette *web safe*, présente $6 \times 7 \times 6 = 252$ couleurs, avec un niveau supplémentaire pour le vert. Cette palette s'approche plus

de 256 couleurs que la palette $6 \times 6 \times 6$, et en ce sens, utilise beaucoup mieux les 8 bits du code de couleur, en plus de mettre un peu plus de finesse dans le vert (mais au détriment des tons de gris, comme nous le montre la fig. 4.3.5 (c)). En effet, nous trouvons $\log_2 252 \approx 7.98$ bits ! Cette nouvelle palette est montrée à la fig. 4.3.5 (b), et son résultat à la fig. 4.3.6 (b).

Voyons maintenant comment procéder à la conversion. Supposons que nous commençons avec une image 24 bits, avec 256 niveaux par composantes. Pour la palette *web safe* de base, il faut commencer par réduire les 256 niveaux de chaque composante à seulement 6, en prenant soin de faire correspondre les zéros, et 255 à 5. On pourrait remarquer que $\frac{255}{5} = 51$ et diviser la valeur originale par 51. Or, cela, à cause de l'arithmétique en nombre entier, ne distribue pas également les 256 niveaux originaux sur les 6 ; puisque seul 255 peut donner 5, ce qui aura pour effet d'assombrir l'image.

Pour s'assurer de bien distribuer les valeurs, nous utiliserons

$$\tilde{r} = \left\lfloor \frac{6r}{256} \right\rfloor, \quad \tilde{g} = \left\lfloor \frac{6g}{256} \right\rfloor, \quad \tilde{b} = \left\lfloor \frac{6b}{256} \right\rfloor.$$

Pour obtenir le code final, on « replie » ces valeurs dans un octet :

$$c = (\tilde{r}6 + \tilde{g})6 + \tilde{b}.$$

C'est comme si nous faisions un décalage par un nombre fractionnaire de bits, par $\log_2 6$ bits¹. Pour décoder, nous ferons, tout en arithmétique entière,

$$\begin{aligned} \tilde{b} &= c \bmod 6, \\ c &= \left\lfloor \frac{c}{6} \right\rfloor, \\ \tilde{g} &= c \bmod 6, \\ \tilde{r} &= \left\lfloor \frac{c}{6} \right\rfloor, \end{aligned}$$

puis enfin

$$\hat{r} = \left\lfloor \frac{\tilde{r}255}{5} \right\rfloor, \quad \hat{g} = \left\lfloor \frac{\tilde{g}255}{5} \right\rfloor, \quad \hat{b} = \left\lfloor \frac{\tilde{b}255}{5} \right\rfloor,$$

1. Lorsqu'on multiplie par une puissance de deux, cela correspond directement à un décalage à gauche des bits ; multiplier par 8, c'est décaler à gauche de 3 bits. Or, nous avons bien que $3 = \log_2 8 = \log_2 2^3$; mais si nous multiplions par 6, cela devrait donc correspondre à un décalage de $\log_2 6 \approx 2.5849$ bits. Les divisions correspondront aux décalages dans l'autre sens, à droite. Cela nous permet donc d'encoder des valeurs avec des *fractions de bits*. Nous y reviendrons lorsque nous discuterons de la compression de données appliquée aux images.

nous donnent les couleurs décodées.

Pour la palette $6 \times 7 \times 6 = 252$, c'est essentiellement le même algorithme, sauf pour la composante verte, qui utilise maintenant 7 niveaux. L'algorithme devient

$$\tilde{r} = \left\lfloor \frac{6r}{256} \right\rfloor, \quad \tilde{g} = \left\lfloor \frac{7g}{256} \right\rfloor, \quad \tilde{b} = \left\lfloor \frac{6b}{256} \right\rfloor,$$

et

$$c = (\tilde{r}7 + \tilde{g})6 + \tilde{b};$$

qu'on décode avec

$$\begin{aligned} \tilde{b} &= c \bmod 6, \\ c &= \left\lfloor \frac{c}{6} \right\rfloor, \\ \tilde{g} &= c \bmod 7, \\ \tilde{r} &= \left\lfloor \frac{c}{7} \right\rfloor, \end{aligned}$$

et finalement

$$\hat{r} = \left\lfloor \frac{\tilde{r}255}{5} \right\rfloor, \quad \hat{g} = \left\lfloor \frac{\tilde{g}255}{6} \right\rfloor, \quad \hat{b} = \left\lfloor \frac{\tilde{b}255}{5} \right\rfloor.$$

4.4 Méthodes adaptatives

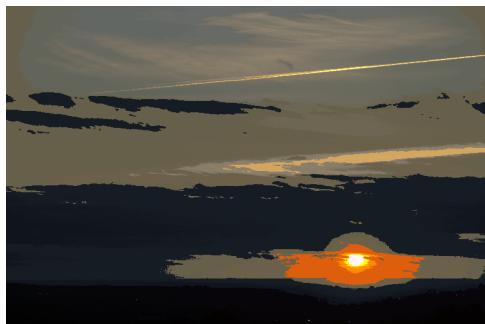
Les méthodes de la section précédente sont trop directes et n'utilisent pas toute l'information de l'image pour bien choisir les couleurs, elles se contente de trouver la couleur la plus près dans une palette limitée. Si nous devons établir une palette limitée mais avec une grande résolution de couleurs (par exemple, 16 couleurs de 24 bits), il faudra utiliser l'information contenue dans l'image, et en particulier la distribution des couleurs. Cette section présente quelques méthodes adaptatives de discréétisation de couleurs.



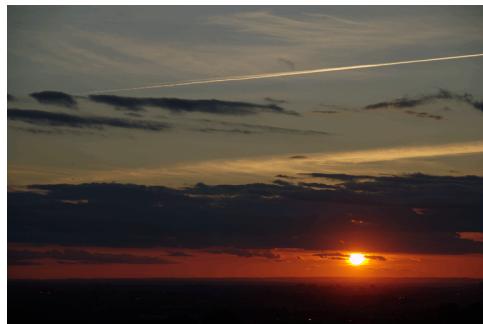
(a) Popularité, 16 couleurs.



(b) Popularité, 16 couleurs, seuil de 10.



(c) Popularité, 256 couleurs.



(d) Popularité, 256 couleurs, seuil de 10.

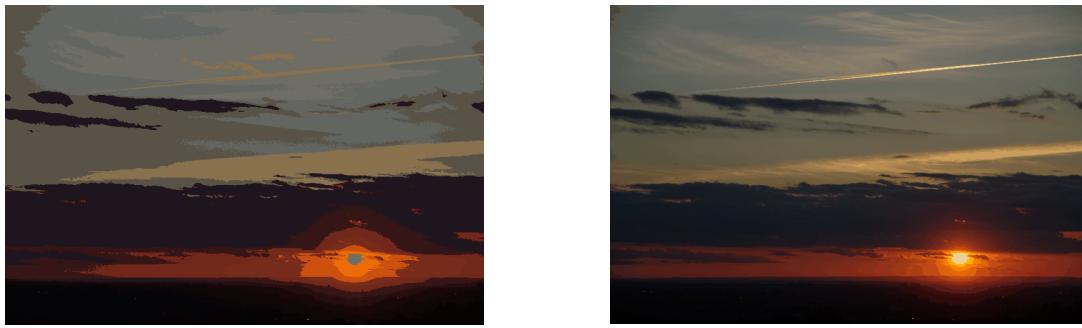
FIGURE 4.4.1 — L'algorithme popularité.

4.4.1 Popularité

Cet algorithme, extrêmement simple, discrétise les couleurs de l'image en ramassant les k couleurs les plus fréquentes dans l'image pour former la palette [168, 169]. En gros, on utilise la valeur de chaque pixel comme un index dans une structure de données (un dictionnaire, ou une autre structure associative) pour aller incrémenter un compteur à chaque fois que cette couleur est observée. Une fois tous les pixels traités, on récupère les k couleurs les plus fréquentes (les pixels avec les plus grands comptes) pour former la palette.

C'est une stratégie simple, mais ultimement vouée à l'échec pour la plupart des images naturelles, car il y a fort à parier que les k couleurs les plus fréquentes soient des variations d'une couleur dominante. Par exemple, pour une photo de paysage montrant un ciel bleu, nous n'aurons pas qu'un seul bleu, mais un nuage de couleurs proches du bleu principal avec des variations minimes. Avec seulement 16 couleurs, nous obtenons la fig. 4.4.1 (a), qui malgré les apparences, n'est pas complètement uniforme. Avec 256, nous obtenons un meilleur résultat, mais encore assez peu satisfaisant, comme nous le montre la fig. 4.4.1 (c). En observant bien l'image, on voit que les « plages » uniformes ne le sont pas, elles sont bien composées de variantes quasiment identiques.

Pour éviter qu'une teinte ne « gagne » toute l'image, nous allons procéder par fusion des couleurs



(a) Palette aléatoire à 16 couleurs.

(b) Palette aléatoire à 256 couleurs.

FIGURE 4.4.2 — Palettes aléatoires.

voisines. Plutôt que de compter toutes les couleurs individuellement, nous allons fusionner les couleurs qui sont « assez près » pour leur faire partager le même compteur et nous calculerons la moyenne des couleurs qui finissent dans la même boîte pour augmenter la résolution de couleur. Cela réduit grandement le nombre de couleurs, et laisse une chance aux couleurs moins présentes tout en conservant de bonnes couleurs moyennes. La fig. 4.4.1 (b) montre le résultat pour 16 couleurs arrondis à la dizaine près. La fig 4.4.1 (d) montre le résultat pour 256 couleurs avec le même arrondi. Nous concluons que l'algorithme de popularité de peut pas bien fonctionner sans l'arrondi. Malgré ces limitations, l'algorithme demeure malgré tout utilisé [41], et peut même donner de bons résultats !

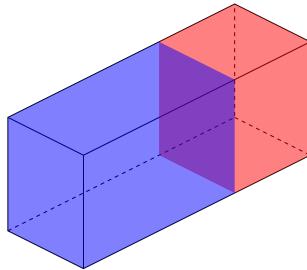
4.4.2 Choix Aléatoire

Une autre possibilité, c'est de simplement choisir k couleurs au hasard dans l'image pour former la palette. La stratégie n'est pas totalement mauvaise, car nous choisirons une couleur avec une probabilité proportionnelle à sa fréquence dans l'image, c'est-à-dire que même si le tirage est aléatoire selon une loi uniforme, il obéira à la distribution des pixels dans l'image. Si c'est en principe vrai pour un (très) grand nombre de « tirages », il n'y a cependant aucune garantie qu'un tirage en particulier choisisse bien les couleurs — le choix peut même être très mauvais !

Un algorithme de discréétisation aléatoire pourrait donc procéder à plusieurs tirages (une dizaine?) et choisir le meilleur résultat. C'est ainsi que j'ai procédé pour la fig. 4.4.2. La fig. 4.4.2 (a) montre le résultat avec 16 couleurs, tandis que la fig. 4.4.2 (b) montre le résultat pour 256 couleurs. Si ces résultats semblent encourageant, la plupart des essais ressemblaient plutôt à la fig. 4.4.1 (c).

4.4.3 *Median-Cut* et *Min-Variance*

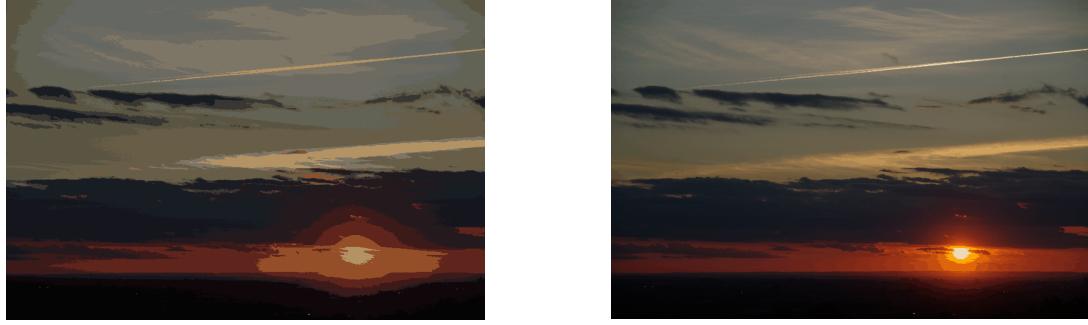
La méthode de *median-cut*, proposée par Heckbert en 1980 procède par coupes successives de l'espace de couleurs [168, 169]. Avec cet algorithme, nous commençons avec un seul volume

(a) *Median-Cut* à 16 couleurs.(b) *Median-Cut* à 256 couleurs.**FIGURE 4.4.3** — *Median-Cut*.**FIGURE 4.4.4** — Plan de coupe pour une boîte dans *Median-Cut*. Le plan de coupe est perpendiculaire à l'axe le plus long.

contenant toutes les couleurs et on calcule sa boîte englobante (*bounding box*) orthogonale aux axes. Cette boîte englobante aura des dimensions différentes dans l'axe des r , des g , des b . Pour choisir comment séparer la boîte en deux, nous trouvons dans quel axe elle s'étend le plus, et nous coupons perpendiculairement à cet axe les points en deux ensembles contenant le même nombre de points en utilisant la médiane selon l'axe choisié (voir fig. 4.4.4).

Ainsi, si c'est dans le rouge que la boîte s'étend le plus, nous trions les pixels selon le rouge (il faudra avoir une fonction pour ordonner les pixels au moins partiellement par une des composantes) et nous prenons la médiane pour les séparer en deux : nous formerons deux nouvelles boîtes en détruisant la boîte originale. Notons que la coupe n'est pas nécessairement au centre de la boîte ; elle peut être très asymétrique car nous coupons en deux selon la distribution des points dans la boîte. Puis nous recommençons avec la boîte la plus étendue (dans n'importe quelle direction), jusqu'à ce que nous ayons le même nombre de boîtes que le nombre de couleurs désirées. Les couleurs finales sont obtenues en calculant la moyenne des couleurs contenues dans chaque boîte.

L'algorithme *median-cut* original procède par raffinements successifs en coupant à chaque fois la boîte la plus étendue pour séparer les couleurs, mais une variante qui semble donner de meilleurs résultats procède en coupant à chaque fois la boîte la plus lourde, c'est-à-dire celle qui contient le plus de couleurs.

FIGURE 4.4.5 — *Min-variance*.

Couper selon la médiane revient à utiliser implicitement une métrique L_1 . Si on veut utiliser une métrique L_2 — aux moindres carrés — il faut couper en utilisant la moyenne et la variance. L'algorithme *min-variance* coupe en premier les boîtes avec les plus grandes variances [367, 381]. Comme pour *median-cut*, l'algorithme regarde les axes r , g et b un par un mais choisit l'axe qui présente la plus grande variance et coupe perpendiculairement à celui-ci. La coupe est choisie de façon à minimiser la somme des variances des boîtes résultantes.

Wan *et al.* proposent de minimiser explicitement la variance [367]. Reprenons leur notation. Si on choisit une direction (disons rouge), nous allons calculer la variance de la boîte dans cette direction, que nous appellerons simplement σ^2 . Nous allons, toujours le long de cette direction, chercher un point de coupe t tel que

$$(w_1 + w_2)\sigma^2 - (w_1\sigma_1^2(t) + w_2\sigma_2^2(t)),$$

la réduction de la variance, est maximisée. Ici, w_1 et w_2 sont les poids relatifs des ensembles produits par la coupe au point t , $\sigma_1^2(t)$ et $\sigma_2^2(t)$ sont les variances de ces nouveaux ensembles. Ils cherchent t optimal, mais en limitant les valeurs testées aux valeurs près de la moyenne dans cette direction.

Comme pour *median-cut*, on procède de façon vorace en choisissant à chaque round la boîte qui a la plus grande réduction de la variance, jusqu'à ce qu'on ait le nombre de boîtes désiré. Les couleurs finales sont obtenues en calculant la moyenne des couleurs contenues dans chaque boîte. La fig. 4.4.5 montre des exemples de *min-variance*.

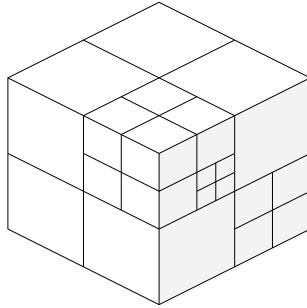


FIGURE 4.4.6 — Division de l'espace en *octree*.

Une dernière variante, *binary split*, va utiliser des plans de coupe arbitraires en trouvant les directions principales par PCA [69, 284]. L'analyse en composantes principales permet de trouver les axes perpendiculaires qui expliquent le mieux un nuage de points (voir la fig. 3.3.7, p. 49). Ces axes remplacent les axes orthogonaux pour les coupes successives. Dans *binary split*, le plan de coupe est perpendiculaire au plus grand vecteur propre trouvé pour le nuage de point et passe par son centroïde.

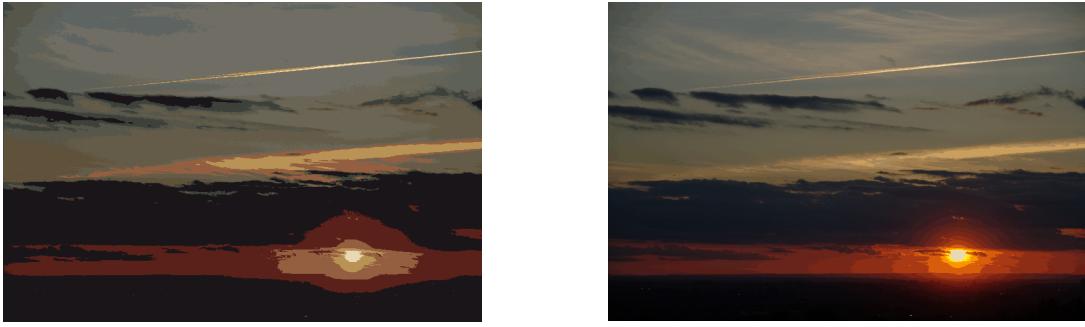
4.4.4 Octree

Cet algorithme de discréétisation va créer un *octree* (voir fig. 4.4.6) dans lequel on insérera toutes les couleurs de l'image [140, 141]. L'arbre, d'arité 8, sera par la suite progressivement élagué, en fusionnant les feuilles, jusqu'à ce nous n'ayons plus que k feuilles, lesquelles contiendront les couleurs finales.

Au début, l'octree est vide. Nous pouvons limiter artificiellement la profondeur de l'arbre pour limiter la quantité de mémoire, mais si nous avons n bits par composantes, l'arbre sera naturellement de profondeur n . Nous allons donc insérer chaque pixel, un par un, dans l'octree. Les feuilles contiendront la couleur insérée, mais aussi un compte du nombre de fois que cette même couleur a été insérée.

Heureusement, savoir laquelle des 8 branches prendre n'est pas compliqué. Au début, à la racine, on prend le bit le plus significatif de chaque composante, ce qui nous donne trois bits. Ces trois bits, concaténés, donnent directement un nombre entre 0 et 7, c'est le numéro de la branche à prendre. Au premier fils, on extrait les deuxièmes bits les plus significatifs des composantes et on recommence. Éventuellement, tous les bits auront été utilisés et nous arrivons à une feuille. Si on a `maxbits` bits par composantes et que nous sommes à la profondeur `profondeur` (en commençant par zéro), l'index est :

```
int index=
```



(a) Octree à 16 couleurs.

(b) Octree à 256 couleurs.

FIGURE 4.4.7 — Discréétisation par *octree*.

```
((p.r >> (maxbits-1-profondeur)) & 1 ) << 2)
| (((p.g >> (maxbits-1-profondeur)) & 1 ) << 1)
| ( (p.b >> (maxbits-1-profondeur)) & 1 );
```

Pour élaguer, on commence par une fouille en profondeur qui nous permettra d'ajouter chaque feuille dans une liste de feuilles, mais aussi, pour chaque nœud interne de l'arbre, de calculer la somme des couleurs dans le sous-arbre. Au terme de cette fouille, nous avons une liste des feuilles et chaque nœud interne contient la couleur moyenne contenue dans le sous-arbre. Ensuite commence la phase d'élagage comme telle. En utilisant la liste de feuilles, on trouve la feuille la plus « légère », celle qui contient le moins de pixels, et on l'enlève. On répète jusqu'à ce que nous n'ayons plus que k feuilles, qui nous donnent directement les k couleurs de la palette. Le résultat est montré à la fig. 4.4.7.

L'algorithme peut procéder en élaguant une seule feuille, ou tuer toute la fratrie pour ne conserver que le parent, qui devient alors une feuille. L'élagage est alors beaucoup plus rapide, mais moins précis. L'algorithme peut aussi limiter la profondeur de l'arbre, ce qui économise la mémoire, mais qui peut avoir pour effet de ne pas faire une division de l'espace où il y aurait dû en avoir une, fusionnant de force des couleurs qui auraient pu rester distinctes.

4.4.5 *K-means*

L'algorithme *K-means* est un algorithme plus général que les autres algorithmes que nous avons vus jusqu'à maintenant, car il permet de résoudre le problème de la *discréétisation de vecteurs*, et non seulement spécifiquement le problème de la discréétisation des couleurs. L'algorithme est d'une telle importance qu'il aura été réinventé plusieurs fois (voir § 4.5). Non seulement l'algorithme calculera une très bonne palette, mais il est en plus assez simple.

Au début, nous choisissons aléatoirement k couleurs distinctes, elles deviendront les *prototypes*. Ensuite, pour chacune des couleurs de l'image, on trouve quel prototype est le plus près (dans l'espace

(a) *K-means* à 16 couleurs.(b) *K-Means* à 256 couleurs.**FIGURE 4.4.8** — Discréétisation avec *K-Means*.

de couleurs) et on assigne la couleur à ce prototype. Ensuite, pour chaque prototype, on calcule la moyenne des couleurs qui lui ont été assignées, et cette couleur moyenne deviendra la nouvelle valeur du prototype. On répète jusqu'à ce que les prototypes ne changent plus.

Il ne reste plus qu'à choisir un critère de convergence. Il pourrait s'agir d'un nombre d'itérations, ou encore que la réduction d'erreur relative passe sous un certain seuil, ou encore que les prototypes ne bougent plus. La fig. 4.4.9 montre une implémentation possible, avec un nombre d'itérations fixe. La fig. 4.4.8 montre les images discréétisées par *K-means*.

*
* * *

L'algorithme est bien étudié. Bottou et Bengio montrent que *K-means* et ses variantes convergent très rapidement, c'est-à-dire de façon au moins quadratique, puisqu'il s'agit d'une version de l'algorithme de Newton [65, 66, 112]. L'algorithme est cependant sensible aux conditions initiales. Selon le choix aléatoire des prototypes, l'algorithme prend des décisions différentes et peut converger à une solution différente. Il se peut aussi que l'algorithme converge rapidement, mais vers une solution sous-optimale. Ainsi, nous pourrions recommencer quelques fois avec des prototypes choisis différemment pour choisir la meilleure solution, celle qui minimise l'erreur.

Cependant, les régions qui correspondent aux prototypes trouvés par *K-means* sont convexes avec des facettes planes (ce sont des *polytopes convexes*), comme le montre la fig. 4.4.10. Ces régions de coupes sont quand même plus générales que les coupes orthogonales, mais sont quand même moins souples que les régions obtenues (au moins en principe) par les algorithmes d'agglomération hiérarchique que nous verrons à la prochaine section.

```

///////////
palette k_means(const bitmap<rgb24> & image, const options & opt)
{
    std::mt19937_64 rand(opt.seed); // mersenne 19937_64 generator
    palette protos;
    std::set<size_t> picked;

    size_t nb_pixels=image.width()*image.height();
    for (size_t i=0;i<opt.nb_colors;i++)
    {
        size_t r;
        do
        {
            r=rand() % nb_pixels;
        }
        while (picked.count(r));
        protos.push_back(image.linear_pixel(r));
    }

    std::vector< std::pair< RGB<size_t>, size_t > > sums(opt.nb_colors);
    for (size_t it=0; it<opt.max_iter; it++)
    {
        for (size_t p=0;p<nb_pixels;p++)
        {
            rgb24 q=image.linear_pixel(p);
            size_t i=closest(protos,q);
            sums[i].first.r+=q.r;
            sums[i].first.g+=q.g;
            sums[i].first.b+=q.b;
            sums[i].second++;
        }

        // renorm
        for (size_t i=0;i<opt.nb_colors;i++)
        {
            if (sums[i].second)
                protos[i]=rgb24( sums[i].first.r/sums[i].second,
                                sums[i].first.g/sums[i].second,
                                sums[i].first.b/sums[i].second );
            else
                protos[i]=image.linear_pixel(rand() % nb_pixels);

            sums[i]={0,0}; // si on ne remets pas les sommes à
                           // zéro, ça donne une "inertie" aux
                           // prototypes, qui se "souviennent"
                           // des valeurs précédentes.
        }
    }

    return protos;
}

```

FIGURE 4.4.9 — Une implémentation possible de *K-Means*.

4.4.6 Agglomération hiérarchique

L'algorithme d'agglomération hiérarchique est coûteux, mais très simple à mettre en œuvre [253, 321, 383, 384]. Il s'agit de trouver les deux couleurs qui sont les plus proches selon notre métrique, de les retirer du nuage de couleurs, et d'y réinsérer la couleur qui résulte de leur fusion. On recommence

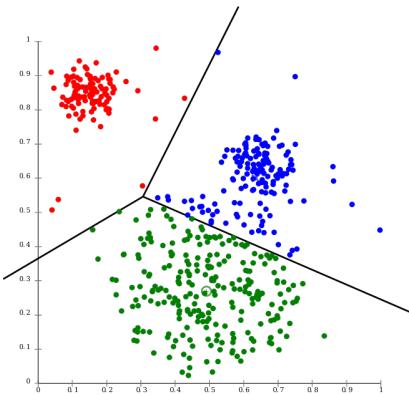


FIGURE 4.4.10 — Division de l'espace par K -means (source : [Wikipedia](#)).

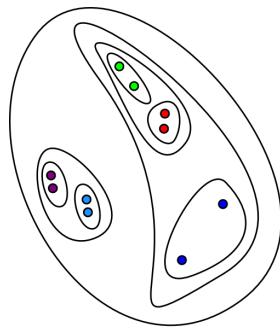


FIGURE 4.4.11 — Les régions produites par une agglomération hiérarchique (« concept d'artiste »).

jusqu'à ce qu'on n'ait plus que k couleurs, qui seront les couleurs de la palette. On voit les résultats à la fig. 4.4.12.

Cet algorithme très simple permet cependant des régions complexes, comme nous le montre la fig. 4.4.11, beaucoup plus générales que les coupes planes de K -means et des autres.



(a) Agglomération hiérarchique à 16 couleurs.



(b) Agglomération hiérarchique à 256 couleurs.

FIGURE 4.4.12 — Discréétisation avec agglomération hiérarchique.

```

fonction agglo( $\{x_i\}_{i=1}^n$  : couleurs,  $k$  : nombre de couleurs)
     $\{w_i\}_{i=1}^n = 1$       { Tous les poids à 1 au début }
    tant que  $n > k$  faire
        Trouver  $x_i$  et  $x_j$ , les deux couleurs les plus proches
        Récupérer  $w_i$  et  $w_j$ , les poids correspondants
        Retirer  $x_i$  et  $x_j$  des  $\{x_l\}_{l=1}^n$ 
        Retirer  $w_i$  et  $w_j$  des  $\{w_l\}_{l=1}^n$ 
        Ajouter  $x_i + x_j$  à  $\{x_l\}_{l=1}^n$ 
        Ajouter  $w_i + w_j$  à  $\{w_l\}_{l=1}^n$ 
         $n := n - 1$ 
    retourner  $\{x_i/w_i\}_{i=1}^k$     { La palette }

```

Algorithme 4.4.1 — Pseudocode pour l’agglomération hiérarchique.

Le pseudocode à l’algorithme 4.4.1 donne une idée de l’algorithme d’agglomération hiérarchique. La stratégie est directe : on trouve les deux couleurs les plus près et on les fusionne. Cependant, pour qu’on puisse calculer la fusion en minimisant l’erreur d’arrondi, on va, comme pour plusieurs autres algorithmes, calculer la somme des couleurs et diviser par le nombre de couleurs fusionnées pour obtenir la couleur représentative. La fonction qui va trouver les deux couleurs les plus proches doit en tenir compte correctement.

La complexité de l’algorithme est élevée. Une implémentation naïve est $O(n^3)$ puisque l’étape de trouver les deux couleurs les plus proches, sans structures de données supplémentaires, est $O(n^2)$ (il faut comparer toutes les paires) et il faut répéter l’opération $O(n)$ fois. On peut cependant grandement accélérer les calculs. D’abord, on précalcule toutes les distances, au coût de $O(n^2)$. Ensuite, on remarque que lorsque que deux couleurs fusionnent, elles disparaissent de la liste des couleurs, mais les seules autres distances qui sont affectées, sont celles où ces deux couleurs participent (autrement dit, une couleur x_k qui avait x_i ou x_j comme ami). On peut alors maintenir un tableau qui dit quelle est la couleur la plus proche pour chaque couleur ; tableau qu’on peut balayer en $O(n)$ pour trouver toutes les couleurs qui doivent se trouver un nouvel ami ; et trouver l’ami se fait en au plus $O(n)$ [59, 63]. Comme on peut espérer que seulement un petit nombre de couleurs seront affectées, on peut espérer que la mise à jour est au plus $O(n)$. On fera $O(n)$ fois cette mise à jour, ce qui ramène la complexité à $O(n^2)$. On peut aussi réduire n en construisant une liste où seules les couleurs effectivement distinctes sont conservées (avec leurs poids correspondant à leur multiplicités).

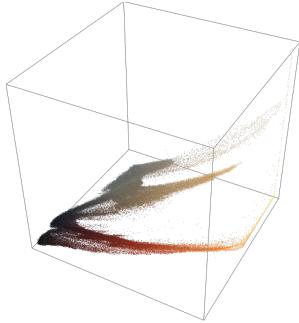


FIGURE 4.4.13 — La distribution des couleurs dans le cube RGB pour notre image de référence.

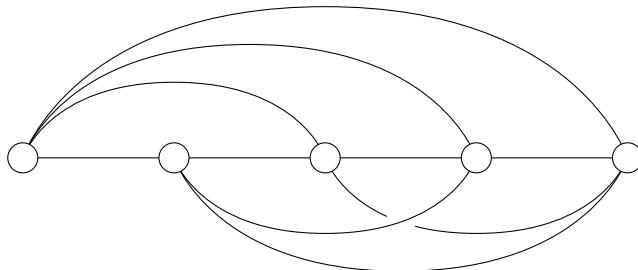


FIGURE 4.4.14 — Le graphe pour l'algorithme de Wu

4.4.7 Programmation dynamique et analyse en composantes principales

L'algorithme de Wu va transformer les couleurs de façon à les ramener autant que possible dans un plan parallèle aux axes pour ensuite les découper optimalement avec algorithme par programmation dynamique [379].

Nous avons remarqué plusieurs fois que les coupes orthogonales/parallèles aux axes ne sont pas forcément optimales, et qu'elles sont certainement plus restrictives que des plans de coupes arbitraires. Sauf que trouver ces plans de coupes optimaux pour chaque sous-nuage de couleurs est dispendieux. Ce que nous devrions faire, c'est d'espérer que notre nuage de couleurs est approximativement dans un plan (comme à la fig. 4.4.13), et de rendre ce plan parallèle à une paire d'axe. Nous pourrions alors procéder aux coupes orthogonales. Nous l'avons déjà mentionné, la façon de procéder pour opérer cette transformation, c'est de trouver les vecteurs propres du nuage de point et calculer la transformée inverse — c'est ce qu'on appelle la transformée de Karhunen-Loève (KLT) ou encore de Hotelling.

Une fois le nuage de couleurs ramené à un plan parallèle aux axes, on procédera à la découpe optimale par programmation dynamique, avec un algorithme qui est une variation sur le thème des plus courts chemins. Nous allons construire un graphe (semblable schématiquement au graphe de la fig. 4.4.14) où chaque nœud correspond à une coupe (dans le rouge, à telle place, par exemple) et où les arcs a_{ij} correspondent à l'erreur encourue si on passe de la coupe i à la coupe j (puisque toutes

Méthode	PSNR (dB)
16	37.036
15	35.689
12	29.317
9	23.261
8	21.073
6	17.754
3	10.902
Paeth	11.927
666	22.385
676	22.519

TABLE 4.4.1 — PSNR en dB des méthodes directes.

Méthode	16	256
Popularité	10.628	24.568
Popularité, seuil 10	23.487	41.978
Aléatoire	26.565	37.686
<i>Median-Cut</i>	22.956	31.841
<i>Min-variance</i>	28.270	39.136
<i>Octree</i>	26.167	37.920
<i>K-Means</i>	30.224	40.039
Agglomération hiérarchique	26.491	34.453

TABLE 4.4.2 — PSNR en dB des méthodes adaptatives.

les couleurs entre les points de coupe sont fusionnées). Il s'agit ensuite de résoudre pour le chemin le plus court de longueur $k - 1$ avec notre algorithme préféré, ce qui nous donne k régions et donc nos k couleurs formant la palette.

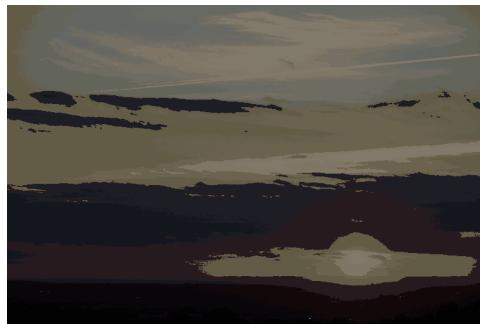
4.4.8 Comparaison des algorithmes adaptatifs

La question que vous vous posez maintenant est « quel est le meilleur algorithme? » Parmi les algorithmes implémentés, et en examinant la table 4.4.2 ainsi que les figs. 4.4.15 et 4.4.16, nous serions tenté de répondre *K-means*, même si l'algorithme de popularité avec fusion donne un résultat suspectueusement adéquat pour l'image test. Quoi qu'il en soit, les algorithmes adaptatifs se montrent pour la plupart bien supérieurs aux méthodes directes, si on en juge avec la table 4.4.1.

Un bon algorithme de discréétisation des couleurs ne devrait pas fonctionner bien seulement lorsque le nombre de couleurs est grand, mais aussi lorsque le nombre de couleurs est réduit, comme 16 ou 4. Là encore, *K-means* semble faire mieux, si on en juge avec la fig. 4.4.15.



(a) Popularité.



(b) Popularité, seuil de 10.



(c) Palette aléatoire.



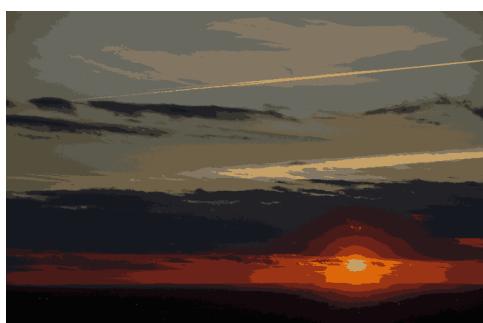
(d) Median-cut.



(e) Min-variance.



(f) Octree.

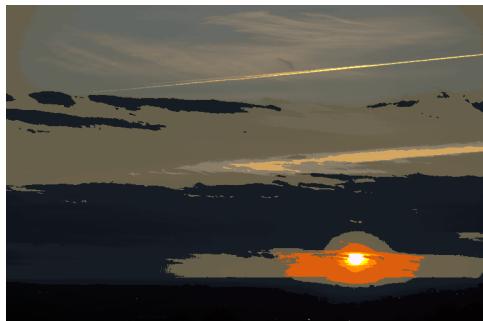


(g) K-Means.

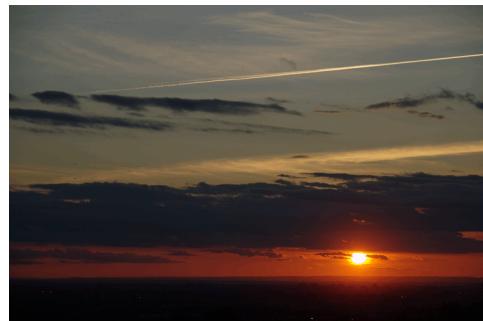


(h) Agglomération hiérarchique.

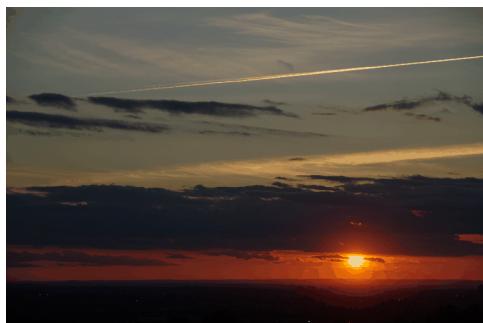
FIGURE 4.4.15 — Les effets des différents algorithmes de discréétisation adaptative, 16 couleurs.



(a) Popularité.



(b) Popularité, seuil de 10.



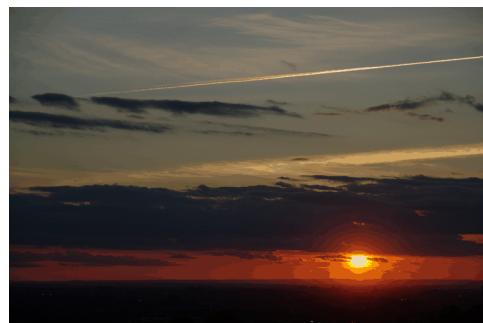
(c) Palette aléatoire.



(d) Median-cut.



(e) Min-variance.



(f) Octree.



(g) K-Means.



(h) Agglomération hiérarchique.

FIGURE 4.4.16 — Les effets des différents algorithmes de discréétisation adaptative, 256 couleurs.

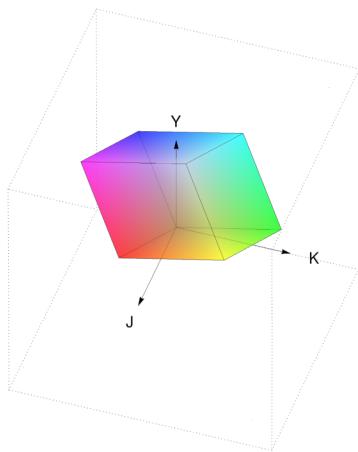


FIGURE 4.5.1 — L'espace de couleur YJK de la puce Yamaha V9938.

4.5 Remarques bibliographiques

L'algorithme *K-means* a été réinventé indépendamment plusieurs fois, dans des contextes très différents [131, 236, 252, 340].

*
* * *

Dans la section sur les méthodes directes, § 4.3, nous avons supposé que les couleurs se discrétaient directement d'un espace RGB à 24 bits vers un espace à 8 bits ou moins grâce à des manipulations assez simples de bits, ou tout au plus avec un algorithme trivial qui met les couleurs RGB en correspondance avec une palette fixe. Or, ce n'est pas toujours ce qui arrive, car les images ne sont pas toujours dans un espace RGB, elles sont parfois dans un espace de couleur plus étrange, voire dédié à un matériel en particulier, ou bidouillé pour être compatible avec un certain type de signal vidéo. Par exemple, les circuits vidéo des premiers ordinateurs personnels (ou « familiaux ») généraient un signal compatible avec le téléviseur, en NTSC ou en PAL/SÉCAM selon les régions. Ces choix de conception se reflètent parfois dans l'espace de couleur utilisé par l'ordinateur. Si on regarde les puces Yamaha V9938 [3], TMS9918A [1], ou MC6847 [2], l'espace de couleur utilisé est *YJK* (nommé Y , $R - Y$, $B - Y$, pour Texas Instruments et Y , ϕA , ϕB pour Motorola, montré à la fig. 4.5.1), ce qui simplifie (très probablement) la circuiterie sous-jacente (cf. avec *YUV* et *YIQ*, § 3.3.9.2).

*
* * *

Les méthodes d'agglomération hiérarchiques existent dans d'autres domaines, comme la biologie, où elles sont utilisées pour créer des « arbres de la vie » en utilisant une métrique comme la proximité génétique entre les organismes considérés. D'autres utilisent le terme d'agglomération hiérarchiques pour des algorithmes différents, plutôt *top-down*, dont il est difficile d'assurer l'optimalité [47, 48, 117, 235]

*
* * *

Les techniques de programmation dynamique [55, 56] ne sont qu'une des nombreuses stratégies pour résoudre des problèmes d'optimisation combinatoire [226, 289].

*
* * *

Dans les années 1920, on a commencé à s'intéresser aux communications câblées à longue distance [197]. L'atténuation du signal était modélisée comme une fonction exponentielle en la longueur du câble, une relation de la forme

$$\tilde{I} \sim I e^{\theta},$$

soit encore $\theta = \ln \frac{\tilde{I}}{I}$, d'où son nom d'« atténuation naturelle ».

Les paramètres de l'atténuation dépendant surtout du câble lui-même, on a proposé le *mile of standard cable* (MSC), qui suppose 88 ohms de résistance uniforme et une capacitance de $0.054 \mu\text{F}$ par mile. Ce modèle étend le modèle d'atténuation naturelle avec de nouveaux paramètres :

$$\tilde{I} = I e^{pN_{MSC}},$$

où MSC représente les constantes du *mile of standard cable*, p est la constante de propagation et N le nombre de miles. Si on brasse l'équation un peu, on trouvera

$$\ln \frac{\tilde{I}}{I} = pN_{MSC}.$$

À partir du milieu des années 1920, on a cherché des méthodes pour mesurer l'atténuation qui de dépendait pas directement des caractéristiques du câble. On a donc proposé les *new telephone transmission units*, les TU, définies par

$$N_{TU} = 10 \log_{10} \frac{W_1}{W_2},$$

où les W_i sont des puissances, donc des quantités quadratiques¹. Les TU sont encore conçus pour mesurer les atténuations, ce qui bien une mesure de fidélité du signal, mais qui, d'une certaine façon, ne dit rien sur l'utilité du signal. En effet, un signal atténué peut encore être parfaitement décodable, c'est pourquoi nous nous intéresserons au rapport entre la puissance du signal et la puissance du bruit (mais nous n'y sommes pas encore).

Hartley remarque que puisque nous percevons les signaux de telle façon que pour un changement perceptuel d'une quantité fixe ΔS , il faille multiplier le signal par une constante k , cette réponse se modélise par une fonction (vaguement) logarithmique [164]. Il prend pour exemple la loi de Weber (ou Weber-Fechner), qui dit que la différence de stimulus tout juste remarquable (la *just noticeable difference*, JND) obéit à une loi de la forme

$$dS = kSW,$$

où S est le stimulus, dS sa variation perçue, W sa puissance et k les constantes appropriées [44, 194]. Si on résout l'équation différentielle ci-dessus, on trouve une loi de forme

$$S \sim k \ln W,$$

où W est la puissance du signal, et S est la magnitude la sensation perçue. Hartley considère qu'il est alors complètement justifié d'avoir une relation logarithmique si on s'intéresse aux effets de l'atténuation sur la perception d'un signal. Faisant référence aux recommandation d'un comité (le *international advisory committee*, dont il ne dit rien d'autre), il propose deux mesures, à savoir,

- Le système népérien, qui pose la variation comme étant proportionnelle à $\frac{1}{2} \ln \frac{P_1}{P_2}$, où P_1 et P_2 sont des puissances, et la constante $\frac{1}{2}$ n'est utile que pour absorber l'exposant 2 dans les puissances ;
- Le bel, où la variation est proportionnelle à $\log_{10} \frac{P_1}{P_2}$, qui peut être encore divisée en *décibels*,

$$dB = 10 \log_{10} \frac{P_1}{P_2}.$$

Dorénavant, affirme Hartley, ce seront les décibels qui seront utilisés à Bell Laboratories, remplaçant définitivement les TU et autres MSC. Ce n'est qu'un peu plus tard (quand?) qu'on a commencé à regarder le rapport de la puissance du signal à la puissance du bruit pour juger de la possibilité de décoder correctement le signal.

*

* * *

1. Certes, rappelons-nous, une puissance électrique est définie par $P = IV = I^2R = \frac{V^2}{R}$, où $R = \frac{V}{I}$ est la résistance en ohms, I le courant en ampères et V le voltage. La puissance est donc quadratique dans le voltage.

Comme pour *K-means*, l'analyse en composantes principales a été réinventée plusieurs fois, avec des applications et des contextes légèrement différents [184, 185, 204, 237, 293]. Pour certains, il ne s'agit que d'extraire les vecteurs propres et les valeurs propres qui leur correspondent; d'autres vont aller une étape plus loin et utiliser la « eigenbasis » (la « base propre ») pour dériver une matrice de transformation qui ramène les vecteurs propres parallèles aux axes naturels, d'autres vont encore normaliser de façon à transformer un saucissoïde en sphéroïde, et ainsi éliminer tout effet indésirable d'échelle (par exemple quand les variables ne sont pas dans les mêmes unités, comme l'âge et le salaire qui n'ont pas d'échelle commune).

5

Tramage et diffusion d'erreur

Sommaire. Dans ce chapitre, nous présentons les techniques de diffusion d'erreur qui nous aident à camoufler le faible nombre de couleurs disponibles pour donner l'illusion d'un grand nombre de couleurs ou encore de tons de gris continus. Ces techniques se divisent en deux grandes familles, le tramage régulier où un motif judicieusement choisi est répété et la diffusion d'erreur qui perturbe les pixels voisins de façon presque aléatoire pour créer l'illusion de tons intermédiaires.

5.1 Introduction

Le chapitre 4 nous a montré comment on peut réduire le nombre de couleurs dans une image de façon à obtenir un rendu raisonnable, mais nous avons aussi vu qu'une assez forte discréétisation mène à des effets bien visibles, que ça soit des couleurs erronées ou des faux contours. Les techniques de tramage et de diffusion d'erreur sont utilisées pour améliorer le rendu visuel — et parfois même la qualité objective — des images dont les couleurs sont discréétisées.

Ces techniques se divisent en deux catégories principales, à savoir

- Le tramage régulier ou quasi-régulier. Ces techniques utilisent des points espacés régulièrement mais dont la taille varie, ces variations permettent d'approximer les tons. Si les points sont répartis selon une grille fixe, on parlera de tramage ordonné (*ordered dithering*) ou de demi-tons (*halftoning*).
- La diffusion d'erreur. Ces techniques corrigent, grâce à une boucle de rétroaction (possiblement assez compliquée), le rendu de l'image discréétisée de façon à ramener le niveau local moyen aussi près que possible de l'image originale.

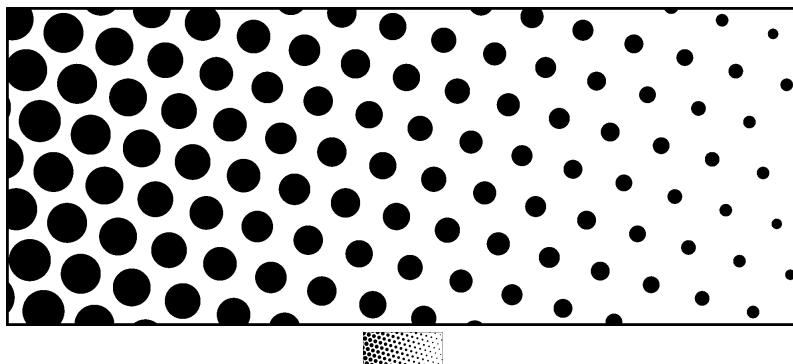


FIGURE 5.2.1 — Tramage régulier et variation de la taille des points. (Source : [Wikipédia](#).)

Évidemment, ces techniques s’appliquent autant aux images monochromes qu’aux images en tons de gris ou en couleurs. Les méthodes de tramage ou de diffusion d’erreur peuvent être modifiées pour préserver les détails (comme les rebords) ou encore éliminer les motifs visuellement dérangeants qui résultent du jeu de la discréétisation et du tramage.

5.2 Tramage régulier et quasi-régulier

Les techniques de tramage régulier et quasi-régulier, utilisées depuis longtemps en imprimerie, reposent essentiellement sur la même technique. On crée une grille de points régulièrement espacés (ou légèrement perturbés pour casser le motif) et on varie la taille des points pour obtenir une impression plus ou moins foncée. On peut y ajouter la couleur, et comme le tramage est habituellement destiné à l’impression, les couleurs choisies sont le cyan, le magenta, le jaune et le noir (revoir la § 3.3.10 au besoin).

5.2.1 Tramage régulier

La fig. 5.2.1 nous montre le principe de base de la technique *halftone* (en français, on trouve demi-teinte et similigravure comme synonymes). Nous commençons avec une grille de points régulièrement espacés, et la grille peut avoir une orientation quelconque. Pour simuler des tons plus foncés, on grossit — et donc rapproche — les points, et pour des tons plus pâles, on les réduit, possiblement jusqu’à leur disparition pour le blanc (ou, plus exactement, pour la couleur du papier).

La technique de la demi-teinte peut être étendue à la couleur. Comme il s’agit souvent d’impression, les couleurs choisies sont les couleurs primaires soustractive, c’est-à-dire le cyan, le magenta, le jaune et une couleur d’alignement, typiquement, mais pas toujours, le noir. Cependant, si les grilles de couleurs étaient orientées de la même façon et alignées, les points se chevaucheraient

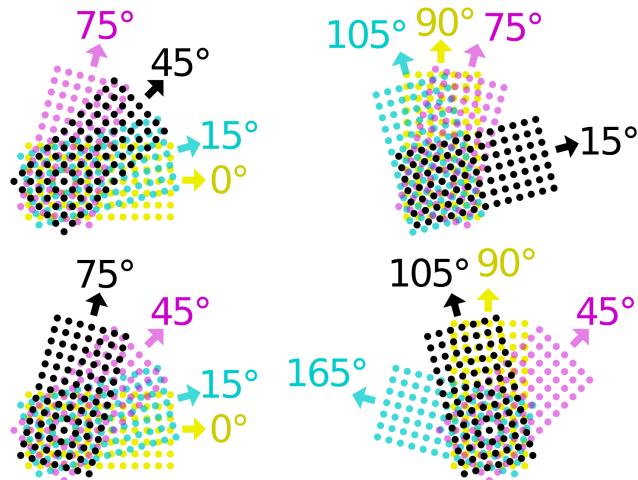


FIGURE 5.2.2 — Les angles typiques utilisés dans le tramage régulier. (Source : [Wikipédia](#).)

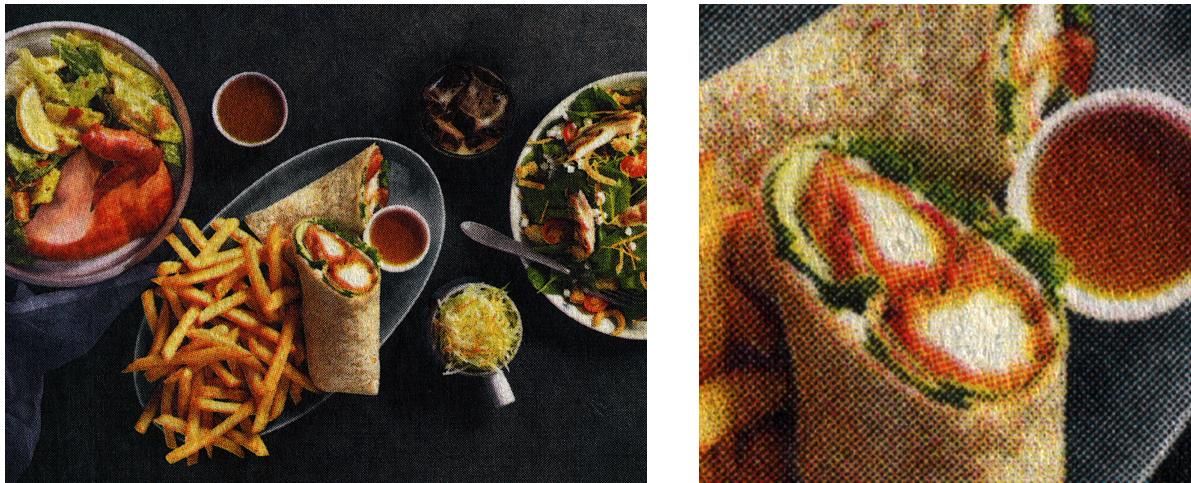


FIGURE 5.2.3 — Un exemple de tramage en demi-teinte, avec détail.

et donneraient des couleurs mélangées et ternes. Pour éviter cela, nous allons changer l'orientation et l'alignement de chacune des grilles de points, selon des angles divers, comme nous le montre la fig. 5.2.2.

Ces variations d'angles minimisent les effets de moiré, mais peuvent toutefois en créer. Si on examine la fig. 5.2.2, on remarque que trois des quatre alignements créent une rosette bien visible, et seule la variation en haut à droite n'en produit pas. Un exemple d'image rendue par demi-teinte (tiré d'un dépliant publicitaire) est montré à la fig. 5.2.3. À distance normale, l'image est satisfaisante, mais si on observe de près, le tramage apparaît clairement. L'effet fonctionne seulement grâce à la résolution angulaire limitée de l'œil. À partir d'une certaine distance, les détails deviennent trop fins pour être perçus individuellement et une « intégration » survient : nous ne percevons plus qu'une moyenne des couleurs, ce qui nous donne l'illusion de couleurs vraisemblables.

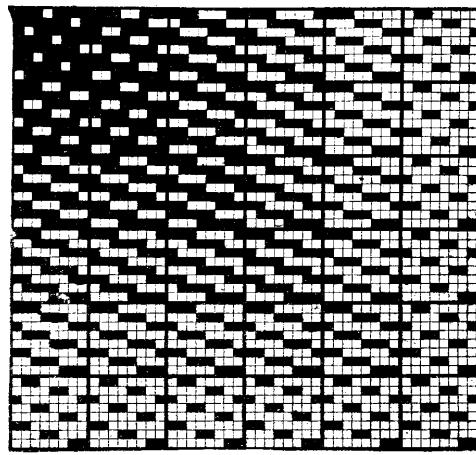
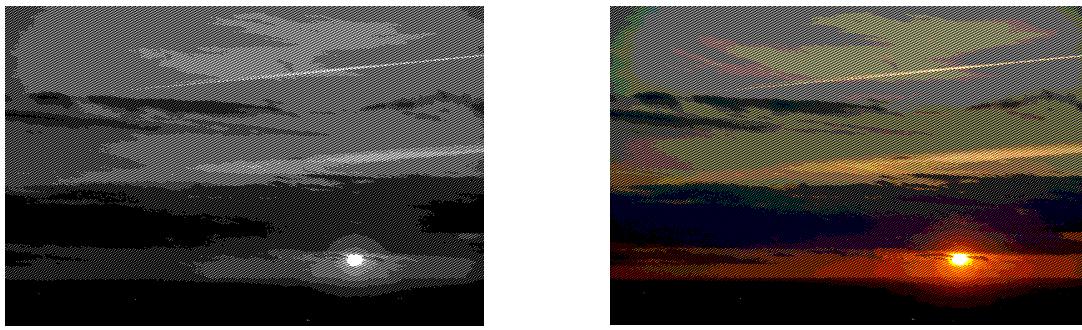


Fig. 24.

FIGURE 5.2.4 — Motifs pour le tissage d'étoffes [52, p. 178], cité dans [212].

(a) Trame de Beaumont.

(b) Trame de Beaumont (couleur).

FIGURE 5.2.5 — Trame ordonné inspiré de Beaumont.

5.2.2 Tramage par ordinateur : aspects historiques

Aussi étonnant que cela puisse être, les premières idées pour simuler des couleurs plus fines à partir de quelques couleurs seulement ne sont pas arrivées avec l'imprimerie couleur, mais avec le tissage d'étoffes — ce qui nous donne l'origine du mot français, *tramage*. Les techniques ont commencé au XVI^e siècle ou avant (voir § 5.4). On trouve chez Beaumont un guide des motifs pour le tissage en couleur, dont on voit un exemple à la fig. 5.2.5 (a) [52]. On en trouve d'autres exemples chez Watson [370]. Ces auteurs sont cités par Knowlton et Harmon, mais il y en a d'autres qui décrivent comment créer motifs et couleurs [215, 290].

Un tramage inspiré de la fig. 5.2.5 (a) peut être appliqué à une image pour obtenir une version en noir et blanc, ou en couleur. Cela produit des images assez agréables (sans toutefois être très fidèles à l'original).

Le motif de la fig. 5.2.5 (a) est représenté grâce à la matrice de seuillage montrée à la fig. 5.2.6. Voyons comment ces matrices fonctionnent. Les niveaux de couleur sont d'abord normalisés de 0 à

6	7	0	1	2	3	4	5
3	4	5	6	7	0	1	2
0	1	2	3	4	5	6	7
5	6	7	0	1	2	3	4
2	3	4	5	6	7	0	1
7	0	1	2	3	4	5	6
4	5	6	7	0	1	2	3
1	2	3	4	5	6	7	0

FIGURE 5.2.6 — Matrice de seuillage inspirée du tramage présenté dans Beaumont.



(a) Tramage de Knowlton.



(b) Tramage de Knowlton (couleur).

FIGURE 5.2.7 — Tramage ordonné inspiré de Knowlton.

7 (comme les nombres de la matrice vont de 0 à 7, la normalisation dépend de la valeur maximale contenue dans la matrice). Si le pixel à la position (x, y) , correspondant à l'entrée $M_{1+(x \bmod 8), 1+(y \bmod 8)}$ de la matrice de seuillage M , est plus foncé (de valeur inférieure ou égale) au nombre dans la matrice, il est marqué en noir; sinon il est laissé blanc. C'est cette correspondance directe entre les pixels de l'image et les coefficients de la matrice qui donne le nom d'« ordonné » à cette classe de techniques. Avec les images en couleurs, on procède similairement : pour chaque composante, rouge, vert, bleu, normalisée, on fait le même test. Cela donne un tramage naïf, mais direct, en huit couleurs de base¹. Le résultat pour la matrice de Beaumont est montré à la fig. 5.2.5.

Cette technique issue du tissage n'a pas été immédiatement appliquée aux images par ordinateur.

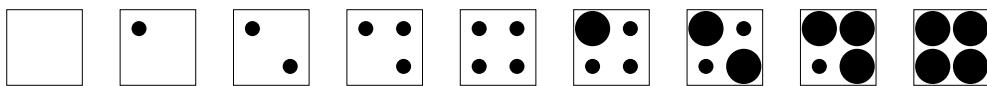


FIGURE 5.2.8 — Motifs du tramage BEFLIX, repris de [212].

1. Quoique l'on pourrait utiliser des fils de couleurs spécifiques et judicieusement choisie pour le tissage, nous nous intéressons ici au rendu d'image par ordinateur!

8	4	8	12	10	6	10	12
4	0	4	8	6	2	6	10
8	4	8	12	10	6	10	12
12	8	12	14	13	9	13	14
11	7	11	13	9	5	9	13
7	3	7	9	5	1	5	9
11	7	11	13	9	5	9	13
12	8	12	14	13	9	13	14

FIGURE 5.2.9 — Matrice de seuillage inspirée du tramage BEFLIX.

Knowlton et Harmon, avec leur système de « films par ordinateur » proposent d’abord une technique beaucoup moins élaborée, probablement pour accommoder un matériel trop simple. Dans leur système BEFLIX, commencé dans les années 1960, ils utilisent un tramage très cru, montré à la fig. 5.2.9) [212]. Tel quel, cela produit des images de très faible qualité, puisque la résolution n’est que de 63×46 « caractères » composés de quatre points. Le système BEFLIX a été utilisé pour produire des court-métrages expérimentaux, entre autre la série *PoemField* avec Stan Vanderbeek, de 1965 à 1972, et avec Lilian F. Schwartz, comme *Pixillation* (1970) d’une durée de 4 minutes et *UFO* (1971), d’une durée de 3 minutes.

Si la méthode originale de Knowlton et Harmon est trop crue pour un usage moderne, nous pouvons quand même nous en inspirer pour créer une matrice de seuillage. Nous la trouvons à la fig. 5.2.9. Comme nous ne disposions pas vraiment du jeu de caractère bizarroïde de BEFLIX, nous avons créé une variation décente, seuillée sur 15 niveaux, de 0 à 14. Nous commençons par les « noyaux » (comme à la fig. 5.2.8) dans les quartiers différents de la matrice. Autour des points de niveaux 0, 1, 2, et 3, nous ajoutons des « couronnes » avec les niveaux 4, 5, 6 et 7; à celles-ci nous en ajoutons d’autres, pour les niveaux 8, 9, 10 et 11, etc., jusqu’à ce que nous remplissions complètement la matrice. Les couronnes sont montrées en couleur, pour montrer la structure de la matrice. Notons de plus que la matrice est considérée comme un tore : si on dépasse d’un côté, on entre de l’autre côté. Le résultat de cette matrice de seuillage est montré à la fig. 5.2.7.

5.2.3 Tramage ordonné

Les premiers essais de tramage, comme ceux de Knowlton et Harmon dans leur système BEFLIX, ou d’Adler *et al.*, étaient trop simples et donnaient des résultats en deçà des attentes. Le premier exemple de tramage ordonné sur ordinateur semble être la méthode proposée par Julesz (1971) [201],

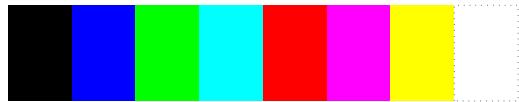
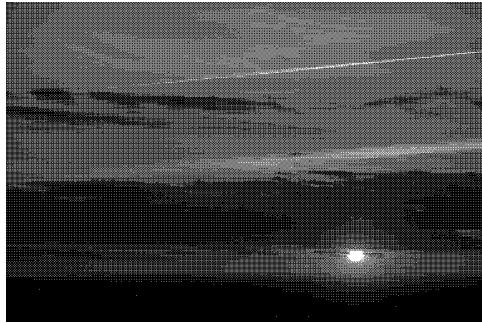


FIGURE 5.2.10 — Les huit couleurs de base utilisées pour le tramage et la diffusion d'erreur.



(a) Tramage de Julesz.



(b) Tramage de Julesz (couleur).

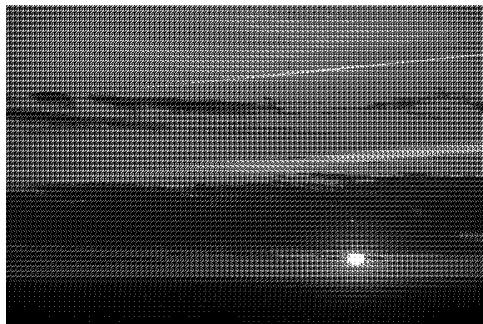
FIGURE 5.2.11 — Tramage ordonné de Julesz.

dont nous discuterons dans un instant à la § 5.2.3.1. Puis, en 1973, Bayer introduit deux matrices, cette fois-ci avec des contraintes explicites. Cependant, les idées de Bayer semblent avoir eu un impact limité car pendant longtemps encore, on a cherché des matrices qui imitent les *halftones* classiques, avec des points ou d'autres formes de base. Les techniques de tramage ordonné — présentées ici en ordre chronologique — furent éventuellement supplantées par la diffusion d'erreur, le sujet de la prochaine section.

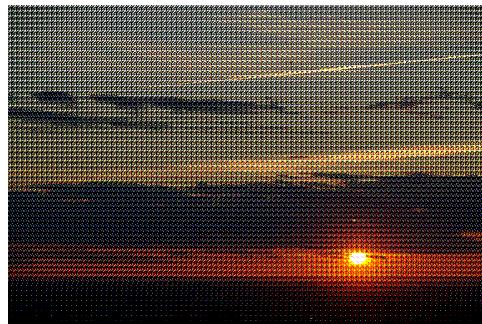
Pour le tramage (et la diffusion d'erreur) en noir et blanc, un seuillage simple est appliqué ; si la luminosité du pixel dépasse le seuil prévu par la méthode de tramage, le pixel est blanc, sinon il est noir. Pour le tramage (et la diffusion d'erreur) en couleur, nous utiliserons la palette de huit couleurs montrée à la fig. 5.2.10. Cela revient à appliquer un seuillage sur chacune des composantes rouge, verte et bleue de l'image — un seuillage par composante.

6	8	4	11
14	2	12	1
5	10	7	9
13	0	15	3

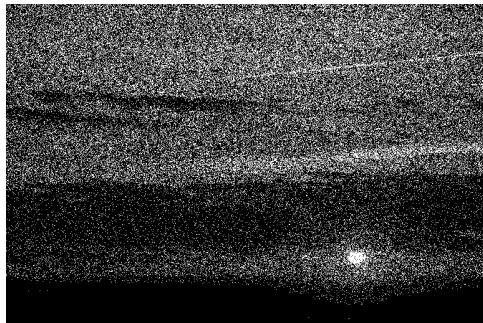
FIGURE 5.2.12 — Matrice de seuillage de Julesz, cité dans [212].



(a) Tramage avec matrice aléatoire de Bayer (noir et blanc).



(b) Tramage avec matrice aléatoire de Bayer (couleur).



(c) Tramage aléatoire.



(d) Tramage aléatoire (couleur).

FIGURE 5.2.13 — Tramage aléatoire de Bayer.

5.2.3.1 Tramage de Julesz

Le premier exemple de tramage ordonné sur ordinateur semble être la méthode proposée en 1971 par Julesz [201], citée par [212]. Plutôt que d'avoir un motif complètement « raisonné », il propose une matrice où les éléments sont heuristiquement distribués de façon à remplir progressivement la plus grande surface encore libre (ou, de façon équivalente, le prochain niveau est aussi loin que possible des précédents dans la matrice). La matrice de seuillage, seulement 4×4 , est montrée à la fig. 5.2.12. Les résultats obtenus sont montrés à la fig. 5.2.11.

5.2.3.2 Tramage de Bayer

Bayer¹ commence par considérer le cas du tramage aléatoire ordonné grâce à une matrice aléatoire [50]. Cependant, comme il ne donne pas cette matrice 8×8 , nous pouvons en générer une qui est une permutation aléatoire des niveaux de 0 à 63. Cela donne des images semblables aux figs. 5.2.13 (a) et 5.2.13 (b), où on voit clairement des motifs très répétitifs². Il aurait été préférable de

1. Oui, le même Bayer des senseurs d'images en mosaïques, voir § 2.2.3.1.

2. La numérisation de l'article dont je dispose est tellement mauvaise qu'elle ne permet pas de conclure grand-chose.

6	5	4	15	16	17	18	19
7	0	3	14	27	28	29	20
8	1	2	13	26	31	30	21
9	10	11	12	25	24	23	22
16	17	18	19	6	5	4	15
27	28	29	20	7	0	3	14
26	31	30	21	8	1	2	13
25	24	23	22	9	10	11	12

FIGURE 5.2.14 — Matrice de seuillage en spirale de Bayer.

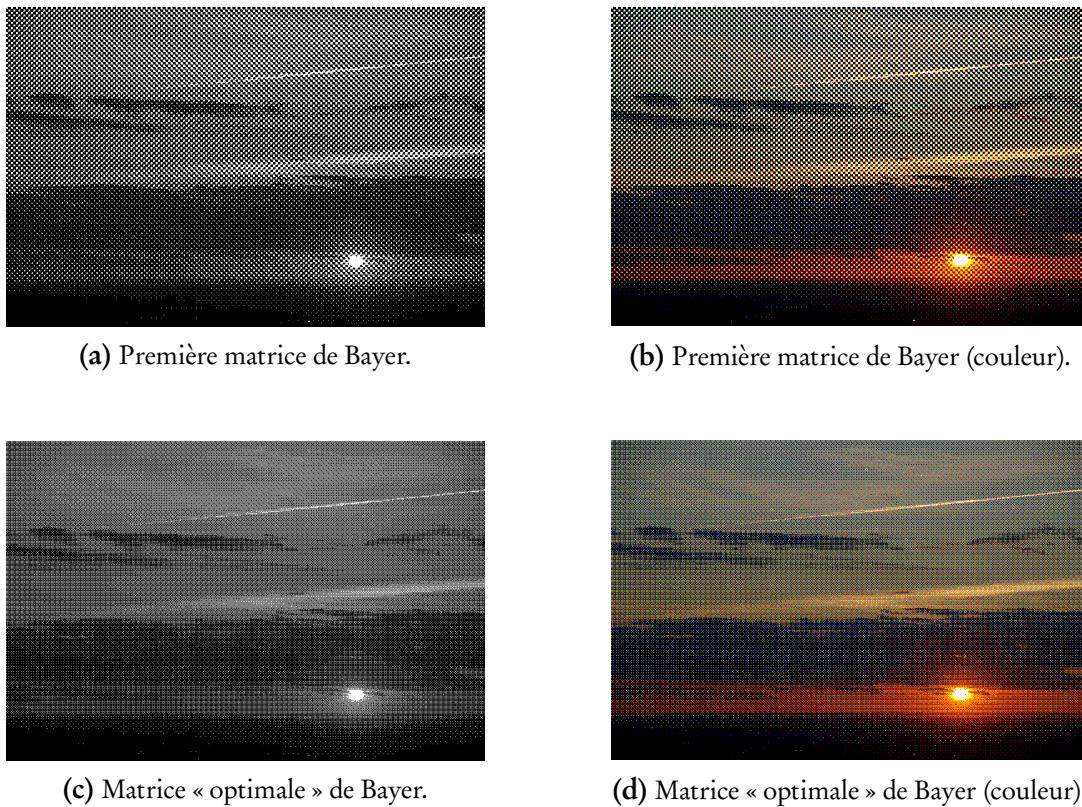
0	16	4	20	1	17	5	21
24	8	28	12	25	9	29	13
6	22	2	18	7	23	3	19
30	14	26	10	31	15	27	11
1	17	5	21	0	16	4	20
25	9	29	13	24	8	28	12
7	23	3	19	6	22	2	18
31	15	27	11	30	14	26	10

FIGURE 5.2.15 — Matrice de seuillage « optimale » de Bayer.

tirer pour chaque point une quantité aléatoire (uniforme, indépendante) pour le seuil et cela aurait donné des images semblables aux figs. 5.2.13 (c) et 5.2.13 (d).

Bayer remarque qu'on pourrait plutôt chercher à simuler les effets de *halftoning* grâce à une matrice qui simule la croissance d'un point. À cet effet, il propose la matrice de la fig. 5.2.14, avec des motifs en spirale ; l'effet est satisfaisant, si on en juge avec les figs. 5.2.16 (a) et 5.2.16 (b). Cependant, il ne s'agit encore que d'une approximation de procédés existants, et non d'une optimisation explicite en terme de précision des tons.

Une bonne matrice de seuillage, raisonne-t-il, devrait être alignée sur les fréquences contenues dans l'image. Après avoir défini une mesure d'erreur basée sur les séries de Fourier, où il fait correspondre les pixels originaux et les pixels seuillés, Bayer propose la matrice « optimale » de la fig. 5.2.15, où quelques couleurs sont présentes pour mettre la structure en évidence. Le résultat est présenté aux figs. 5.2.16 (c) et 5.2.16 (d).

**FIGURE 5.2.16** — Tramage ordonné de Bayer.

6	5	4	3	4	5	6	7
5	4	3	2	3	4	5	6
4	3	2	1	2	3	4	5
3	2	1	0	1	2	3	4
4	3	2	1	2	3	4	5
5	4	3	2	3	4	5	6
6	5	4	3	4	5	6	7
7	6	5	4	5	6	7	8

FIGURE 5.2.17 — Matrice de seuillage en losanges d'Allebach et Liu.

5.2.3.3 Tramage d'Allebach et Liu

Allebach et Liu reviennent d'abord au tramage inspiré de l'imprimerie, mais cette fois-ci avec des motifs en losanges [36]. Ce motif est montré à la fig. 5.2.17, et les résultats aux figs. 5.2.19 (a) et 5.2.19 (b). Ils remarquent que cette matrice peut créer des motifs indésirables pour certaines images où l'échelle des variations ne correspond pas à l'espacement naturel des losanges.

0	4	2	7	1	4	2	7
5	3	7	3	5	3	7	5
2	7	2	5	2	7	2	5
7	3	6	3	7	3	6	3
1	4	2	7	1	4	2	7
5	3	7	3	5	3	7	3
2	7	2	5	2	7	2	5
7	3	6	3	7	3	6	3

FIGURE 5.2.18 — Matrice de seuillage en grille d’Allebach et Liu.

Pour résoudre ce problème, ils proposent une matrice en grille (*grating dot profile matrix*) censée maximiser le contenu en fréquences, c'est-à-dire le nombre de transitions possibles¹. Cette matrice (qui n'est que partiellement spécifiée dans leur article) offrirait donc plus de transitions qui auraient de meilleures chances de correspondre aux variations dans l'image. Cette matrice ressemblerait à la matrice montrée à la fig. 5.2.18. Le résultat est montré aux figs. 5.2.19 (c) et 5.2.19 (d).

5.2.3.4 Tramage de Bryngdahl

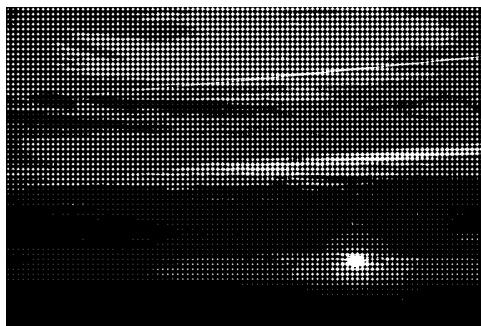
Bryngdahl, qui cherche aussi à maximiser la distribution des points (en « maximisant la longueur de la bordure entre les tons pâles et les tons foncés »), propose des motifs qui rappellent les rosettes du tramage régulier [80]. Pour ce faire, il propose d'abord une matrice 5×5 d'apparence aléatoire (fig. 5.2.20 (a)), qui, une fois composée par des rotations de 90° en sens horaire, forment une matrice 10×10 présentant des motifs vaguement concentriques (fig. 5.2.20 (b)). Le résultat de la matrice 10×10 est montré à la fig. 5.2.21.

5.2.3.5 Tramage de Hawley

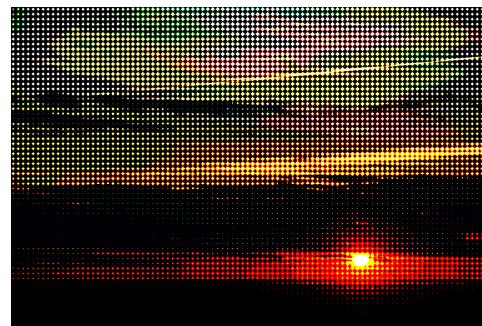
Hawley propose quant à lui une procédure pour générer, au moins en principe, des matrices de seuillage dont les dimensions sont des puissances de deux quelconques [165]. Il commence par poser la « zéroième » matrice

$$M_{n,0} = [0],$$

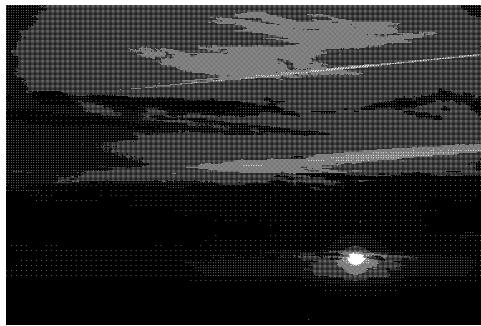
1. Nous reviendrons sur le concept de « fréquences » pour les images au chapitre ??.



(a) Tramage d'Allebach et Liu.



(b) Tramage d'Allebach et Liu (couleur).



(c) Tramage en grille d'Allebach et Liu.



(d) Tramage en grille d'Allebach et Liu (couleur).

FIGURE 5.2.19 — Tramage ordonné de Allebach et Liu.

8	0	11	9	6
2	22	20	13	23
12	16	3	5	17
10	18	7	1	15
4	24	14	19	21

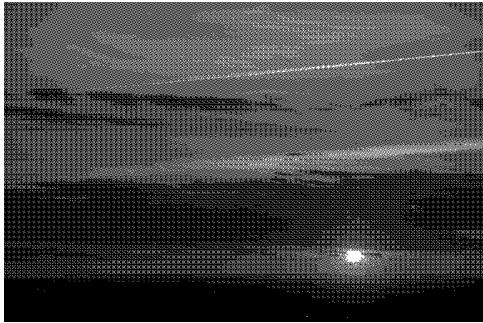
(a) Matrice de seuillage 5×5 de Bryngdahl.

10	0	15	7	4	11	4	7	15	0
2	9	12	19	16	22	18	21	14	8
15	14	24	3	6	10	5	1	24	12
7	21	1	13	20	23	17	13	3	19
4	18	5	17	8	0	9	20	6	16
11	22	10	23	2	11	2	23	10	22
4	16	6	20	9	0	8	17	5	18
7	19	3	13	17	23	20	13	1	21
15	12	24	1	5	10	6	3	24	14
2	8	14	21	18	22	16	19	12	9

(b) Matrice de seuillage 10×10 de Bryngdahl.**FIGURE 5.2.20** — Matrices de Bryngdahl.

puis la relation de récurrence

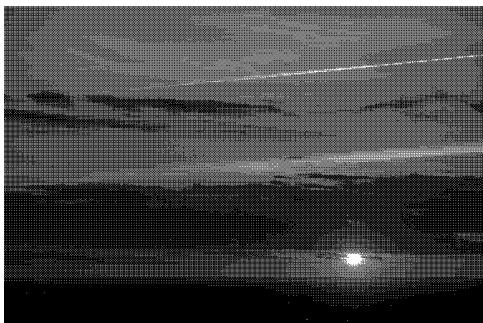
$$M_{n,k} = \begin{bmatrix} M_{n,k-1} & 3 \times 2^{2k-2} + M_{n,k-1} \\ 2 \times 2^{2k-2} + M_{n,k-1} & 2^{2k-1} + M_{n,k-1} \end{bmatrix}.$$



(a) Tramage de Bryngdahl.



(b) Tramage de Bryngdahl (couleur).

FIGURE 5.2.21 — Tramage ordonné de Bryngdahl.

(a) Tramage de Hawley.



(b) Tramage de Hawley (couleur).

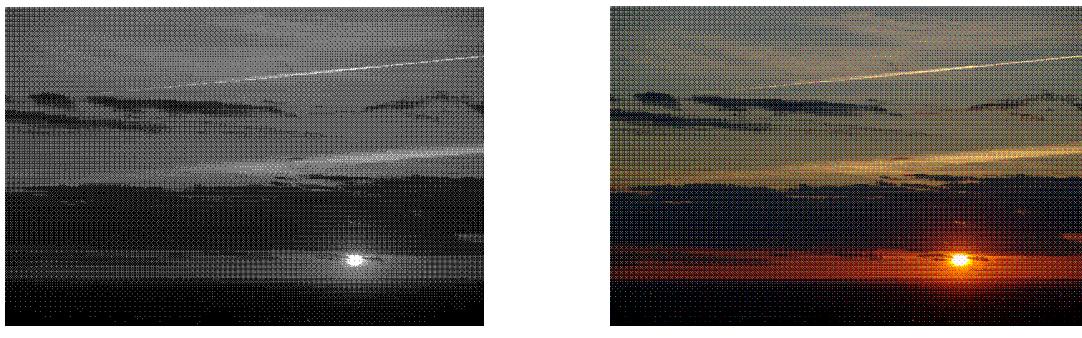
FIGURE 5.2.22 — Tramage ordonné de Hawley.

En posant $n = 6$ et commençant avec $k = 3$, on obtient la matrice

$$M_{8,3} = \begin{bmatrix} 0 & 48 & 12 & 60 & 3 & 51 & 15 & 63 \\ 32 & 16 & 44 & 28 & 35 & 19 & 47 & 31 \\ 8 & 56 & 4 & 52 & 11 & 59 & 7 & 55 \\ 40 & 24 & 36 & 20 & 43 & 27 & 39 & 23 \\ 2 & 50 & 14 & 62 & 1 & 49 & 13 & 61 \\ 34 & 18 & 46 & 30 & 33 & 17 & 45 & 29 \\ 10 & 58 & 6 & 54 & 9 & 57 & 5 & 53 \\ 42 & 26 & 38 & 22 & 41 & 25 & 37 & 21 \end{bmatrix}.$$

La matrice de Hawley (qui n'est peut-être pas la sienne, comme il ne cite aucune source) favorise la dispersion des points en quinconces, ce qui évite en partie les motifs dérangeants, mais crée aussi des régions plus uniformes.

0	32	8	40	2	34	10	42
48	16	26	24	50	18	58	26
12	44	4	36	14	46	6	38
60	28	52	20	62	30	54	22
3	35	11	43	1	33	9	41
51	19	59	27	49	17	57	25
15	47	7	39	13	45	5	37
63	31	55	23	61	29	53	21

FIGURE 5.2.23 — Matrice de seuillage en quinconce de Schumacher.

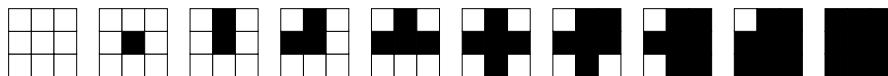
(a) Tramage de Schumacher.

(b) Tramage de Schumacher (couleur).

FIGURE 5.2.24 — Tramage ordonné de Schumacher.

5.2.3.6 Tramage de Schumacher

Schumacher généralise la distribution des points en quinconce avec la matrice de la fig. 5.2.23 [325]. Mieux distribuée que la matrice de Hawley, la matrice de Schumacher réduit les plages uniformes, comme on peut le voir à la fig. 5.2.24. Réduire les plages uniforme a pour heureux effet de mieux représenter les détails, ou, du moins, de présenter subjectivement une image plus agréable.

**FIGURE 5.2.25** — Motif 3×3 d'Adler *et al.*

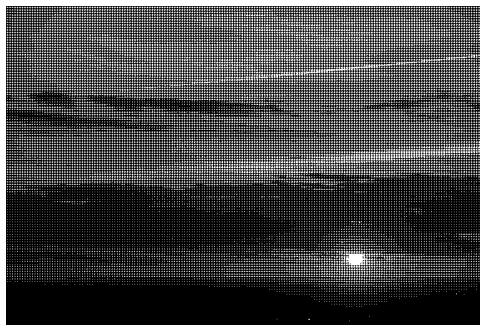
15	14	9	11
13	8	1	5
10	2	0	3
12	7	4	6

(a) Matrice de seuillage 4×4 inspirée d'Adler *et al.*

7	6	5	5	5	5	6	7
6	5	4	3	3	4	5	6
5	4	2	1	1	2	4	5
5	3	1	0	0	1	3	5
5	3	1	0	0	1	3	5
5	4	2	1	1	2	4	5
6	5	4	3	3	4	5	6
7	6	5	5	5	5	6	7

(b) Matrice de seuillage 8×8 inspirée d'Adler *et al.*

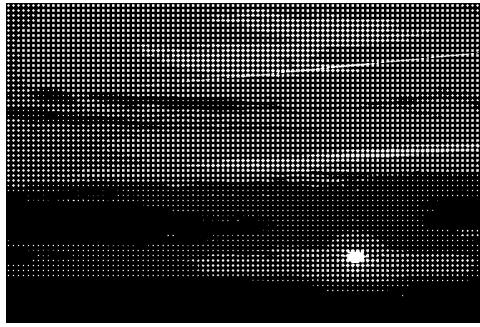
FIGURE 5.2.26 — Matrices d'Adler *et al.*



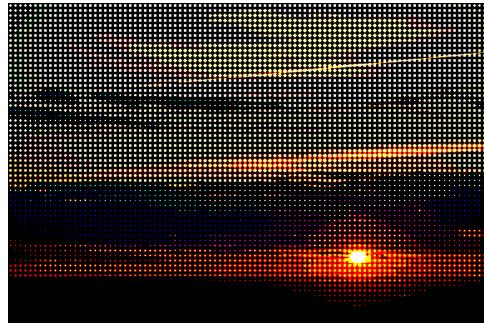
(a) Tramage de Adler *et al.* 4×4 .



(b) Tramage de Adler *et al.* 4×4 (couleur).



(c) Tramage de Adler *et al.*

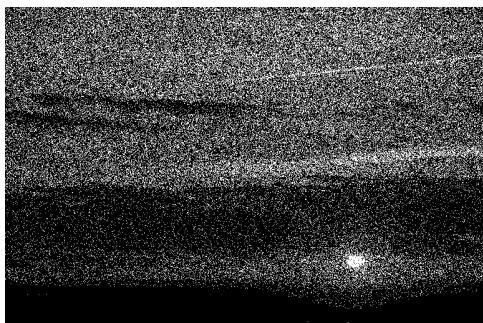


(d) Tramage de Adler *et al.* (couleur).

FIGURE 5.2.27 — Tramage ordonné de Adler *et al.*

5.2.3.7 Tramage d'Adler *et al.*

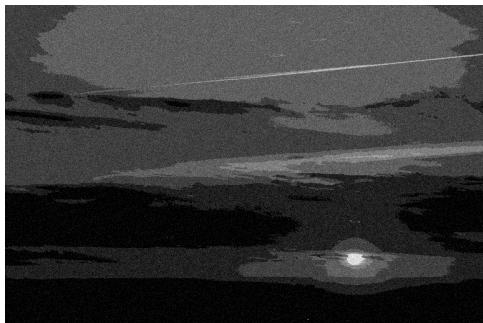
Adler *et al.* reviennent au tramage classique en simulant des points de différentes tailles [31]. Ils proposent une matrice 3×3 , dont les motifs sont montrés à la fig. 5.2.25 (qui n'est pas sans rappeler le tramage BEFLIX, voir fig. 5.2.8), et (pour des raisons d'implémentation dans le programme utilisé pour générer les tramages), c'est la méthode avec une matrice 4×4 , montrée à la fig. 5.2.26 (a), dont on



(a) Tramage aléatoire.



(b) Tramage aléatoire (couleur).



(c) Tramage aléatoire de Roberts, 8 niveaux.



(d) Tramage aléatoire de Roberts, 8 niveaux (couleur).

FIGURE 5.3.1 — Tramage aléatoire.

voit les résultats aux figs. 5.2.27 (a) et 5.2.27 (b). Une version avec de plus gros points, dont la matrice est montrée à la fig. 5.2.26 (b), donne sans doute des images plus agréables, pour peu que les points soient très petits, comme on peut les faire à l'impression. Nous en voyons l'effet aux figs. 5.2.27 (c) et 5.2.27 (d).

5.3 Diffusion d'erreur

Les matrices de seuillage tentent soit de simuler le tramage classique, hérité de l'imprimerie, soit d'approximer localement les couleurs avec une grille fixe, en souhaitant que, vu d'assez loin ou en assez haute résolution, la densité des points nous permettent de percevoir une bonne approximation des couleurs désirées.

Il y a deux problèmes avec cette approche. Premièrement, la matrice est fixe et donc susceptible de former des motifs répétitifs et visibles (comme le tramage de Bryngdahl, § 5.2.3.4, pour lequel les motifs sont très apparents). Deuxièmement, si la densité des points est faible, ou que les niveaux de seuillages sont trop crus, l'approximation locale des tons ne pourra pas être très bonne.

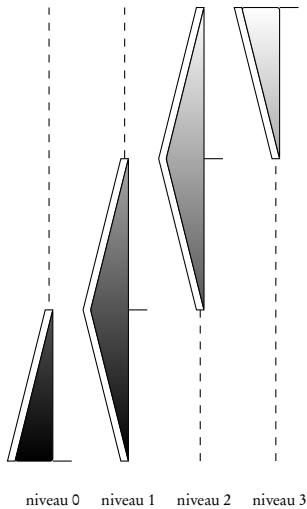


FIGURE 5.3.2 — Bruit pseudo-aléatoire de Roberts pour quatre niveaux.

Si une matrice induit nécessairement un motif répétitif, ne pourrait-on pas simplement seuiller au hasard pour chaque point, évitant ainsi tout motif? Il serait simple de tirer une valeur aléatoire entre 0 et 255 (ou peu importe les valeurs valides pour un pixel) et si cette valeur aléatoire est inférieure à la valeur du pixel, on émet un point noir, sinon, on émet un point blanc. Pour les pixels en couleur, on pourrait appliquer la même stratégie chaque composante, émettant « zéro rouge » ou « plein rouge » selon que la valeur choisie aléatoirement soit inférieure ou supérieure à la valeur rouge, et de même pour les deux (ou plus) autres composantes. Cela donnerait des images semblables aux figs. 5.3.1 (a) et 5.3.1 (b). Elles sont très bruitées à la résolution présentée ici, mais si nous avions une résolution *très* élevée, la qualité pourrait paraître raisonnable, malgré que les images couleurs soient formées d'une palette de seulement huit couleurs. Évidemment, à résolution moyenne ou faible, la qualité est très peu satisfaisante¹.

Roberts propose un compromis pour « dédiscrétiser » une image en augmentant artificiellement le nombre de niveaux avec du bruit (pseudo)aléatoire [319]. Supposons que nous ayons au départ une image en huit niveaux que nous voulons rendre sur 256 niveaux. Nous commençons par calculer comment les huit niveaux originaux s'étalent sur nos 256 niveaux (voir § 4.3.3 pour la méthode), mais, pour simuler plus de niveaux, nous allons perturber ces huit valeurs avec un bruit aléatoire tiré selon une loi triangulaire, comme nous le montre la fig. 5.3.2. Ainsi, le niveau 2, par exemple, pourrait être rendu par n'importe quel niveau contenu dans le triangle, avec une probabilité proportionnelle

1. Par contre, si la résolution de l'image une fois discrétisée est *nettement* supérieure à la résolution de l'image originale, on pourrait imaginer promouvoir un pixel de l'image originale à une matrice de points dans l'image discrétisée et choisir chacun des pixels discrétisés dans cette matrice de façon à obtenir une bonne approximation de la couleur originale. Par exemple, un pixel en ton de gris dans l'image originale pourrait devenir une tuile de 4×4 pixels noirs et blancs, et de ces 16, choisir un certain nombre (proportionnel à l'intensité du pixel original) qu'on mettra blancs en laissant les autres noirs.

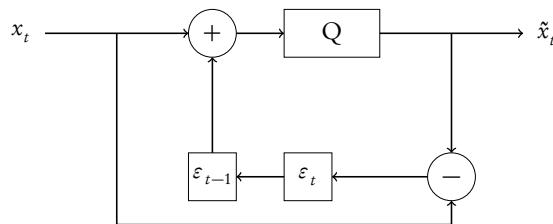


FIGURE 5.3.3 — La boucle de rétroaction pour la correction d'erreur en une dimension.

à la hauteur du triangle¹. Le résultat pour 8 niveaux de gris est montré à la fig. 5.3.1 (c), tandis que le résultat pour 8 niveaux de rouge, de vert et de bleu est montré à la fig. 5.3.1 (d). Le résultat semble plus heureux pour la version en couleur que pour l'image en tons de gris; mais la méthode n'élimine pas les faux contours dus à la discréétisation puisque le bruit aléatoire, tirée d'une loi triangulaire, va majoritairement choisir des valeurs près des niveaux originaux — il aurait peut-être alors été préférable d'utiliser une loi uniforme. De plus, le bruit, étant aléatoire et indépendant, ne cherche pas à corriger les tons vers les couleurs originales, mais seulement à les perturber.

5.3.1 Boucle de rétroaction pour la correction d'erreur

Le tramage de Roberts, nous l'avons vu, utilise simplement le hasard pour perturber les niveaux trop crus afin de dissimuler, ou au moins d'amoindrir, les faux contours. Sauf que cette tactique, pire que le tramage ordonné, n'assure pas que la moyenne des points dans une région approxime de façon satisfaisante la vraie couleur, inaccessible, perdue par la discréétisation. Le tramage ordonné tente explicitement d'approximer localement cette moyenne (tout en évitant autant que faire se peut les motifs dérangeants), mais le tramage est fait indépendamment pour chacun des points, sans contrôle de l'erreur — ou plutôt, une erreur approximativement contrôlée sur une région de l'image aussi grande que la « tuile » de tramage.

Ce que nous devrions plutôt faire, c'est compenser l'erreur due à la discréétisation de façon à ce qu'une erreur excessive sur un pixel soit corrigée par un pixel voisin, et ainsi conserver localement une couleur moyenne plus près de la vraie couleur. La fig. 5.3.3 montre une telle boucle de rétroaction pour une image en une dimension. La variable x_t représente la vraie valeur du t^{e} pixel. Sans rétroaction, ce pixel passerait directement par la boîte Q qui symbolise la discréétisation² et on émettrait sa version discréétisée \tilde{x}_t . Or, il se peut, en fonction de ce qui se trouve dans la boîte Q , que \tilde{x}_t soit passablement différent de x_t , et que les valeurs successives entre deux niveaux de discréétisation soient toutes approximées par la même valeur discréétisée, créant ainsi des faux contours

1. *Quête secondaire* pour le lecteur : établissez une méthode de génération de nombres aléatoires qui respecte une loi triangulaire.

2. Il est habituel de représenter la discréétisation par la lettre Q , puisqu'en anglais, on parle de *quantization*.

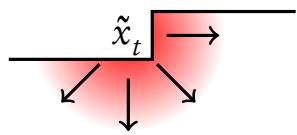


FIGURE 5.3.4 — La diffusion d'erreur doit se faire sur les pixels voisins en 2D.

et des plages uniformes alors qu'elles auraient dû montrer un dégradé. Il s'agit alors de « corriger le tir » en conservant une mesure d'erreur. Si $\tilde{x}_t \leq x_t$, alors nous avons « sous-évalué » la valeur du pixel, nous devrions alors « sur-évaluer » la valeur du prochain pixel pour qu'en moyenne, les valeurs soient approximativement correctes — et, inversement, si $\tilde{x}_t \geq x_t$, le prochain pixel devrait être « sous-évalué ». Ce mécanisme est montré à la fig. 5.3.3 par l'opération de différence et les deux boîtes qui permettent un délai d'un pixel pour l'erreur. L'erreur calculée est $\varepsilon_t = x_t - \tilde{x}_t$, ainsi, si on sous-estime, le prochain pixel aura sa valeur augmentée ; sinon, elle sera diminuée.

Ce mécanisme en une dimension est simple mais on peut s'y objecter en faisant remarquer qu'une image est typiquement en deux dimensions, et donc que la propagation d'erreur devrait l'être aussi — si on dispose d'assez de mémoire pour stocker les prochaines lignes de l'image. On peut imaginer que la propagation se fasse à partir du pixel discréteisé vers ses voisins dans l'image, de façon radiale, comme le suggère la fig. 5.3.4. Mais comment y arriverons-nous ?

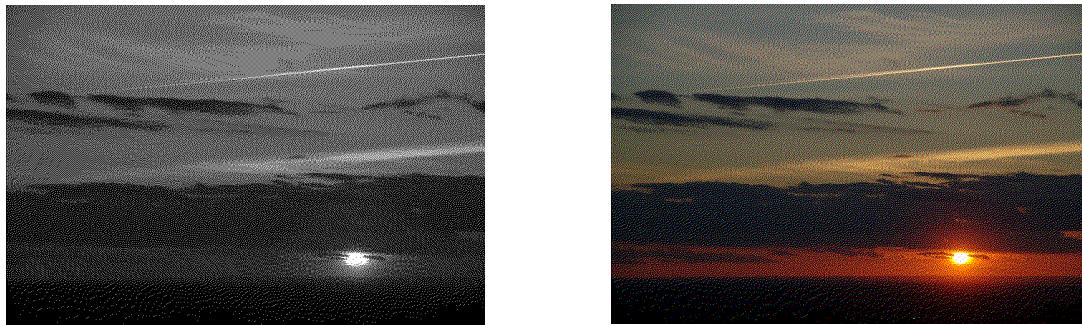
5.3.2 Matrices de diffusion

L'approche habituelle est d'utiliser une matrice de diffusion (en anglais, *dithering matrix*) qui spécifie les proportions de l'erreur que reçoit chaque pixel voisin. Cette matrice est souvent de taille modeste (parce qu'on veut limiter la propagation d'erreur autant que limiter la quantité de calculs), et ses coefficients exprimés comme des nombres entiers, chacun correspondant à la part relative de l'erreur reçue. La somme des coefficients donne le diviseur¹.

Dans ce qui suit, nous montrerons l'effet des diverses matrices de diffusion d'erreur sur des images seuillées en noir et blanc et sur des images en huit couleurs — et pouvoir ainsi comparer justement avec les méthodes de tramage ordonné. Les huit couleurs sont montrées à la fig. 5.2.10, p. 141. Cette palette revient à seuiller indépendamment les composantes rouge, verte et bleue de l'image.

\tilde{x}_t	7
3	5
1	

FIGURE 5.3.5 — La matrice de diffusion d'erreur de Floyd et Steinberg (diviseur : 16).



(a) En noir et blanc.

(b) En 8 couleurs.

FIGURE 5.3.6 — Diffusion d'erreur de Floyd et Steinberg.

5.3.2.1 Méthode de Floyd-Steinberg

Historiquement, la première méthode de diffusion d'erreur est due à Floyd et Steinberg qui proposent une matrice somme toute réduite, que l'on voit à la fig. 5.3.5 [129]. La fig. 5.3.6 reprend l'image du coucher de soleil et nous montre l'effet de la diffusion d'erreur de Floyd et Steinberg en noir et blanc et en huit couleurs.

La matrice de Floyd et Steinberg diffuse l'erreur sur quatre pixels voisins et les coefficients sont exprimés en 16^{ièmes}. Le choix de 16 comme dénominateur est sans doute motivé par la facilité du calcul¹. Les coefficients sont trouvés *mostly by trial and error* dans le but d'éviter la génération de motifs dérangeants tout en s'assurant que les tons de gris moyens (i.e., 50%) donnent des motifs en damier [129, p. 74].

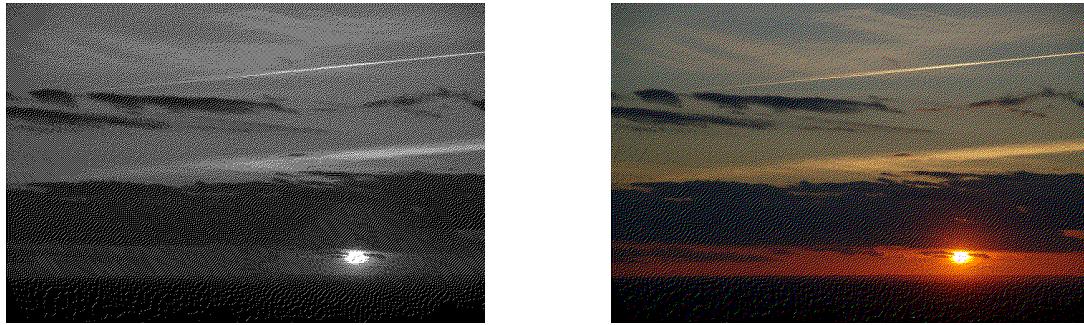
Crocker *et al.* mentionnent le « faux filtre » de Floyd et Steinberg (montré à la fig. 5.3.7) comme s'il existait déjà, mais sans donner de référence. Cette matrice de diffusion est en effet souvent présentée comme l'invention de Crocker *et al.*, mais ça ne semble pas être le cas, si on en croit le texte original (un fichier texte de 1989, dhalf.txt, qui vient probablement d'Usenet [100]). Ils remarquent par ailleurs que cette méthode de diffusion est loin d'être aussi bonne que la méthode de Floyd et

1. On peut sur- ou sous-estimer le diviseur. Un diviseur plus grand que la somme des coefficients atténuerait l'erreur, un diviseur plus grand, l'exagérera.

1. En arithmétique en nombres entiers, effectuer un décalage vers la droite de 4 bits est équivalent à diviser par 16 : $x \gg 4$ remplacera $x / 16$. On suppose les décalages *beaucoup* plus rapides que les multiplications et surtout les divisions. Par ailleurs, certaines multiplications sont aussi « faciles » à faire avec des décalages : $5 * x$ devient $(x \ll 2) + x$, $7 * x$ devient $(x \ll 3) - x$, etc.

\hat{x}_t	3	
	3	2

FIGURE 5.3.7 — La matrice de diffusion d'erreur de Crocker *et al.* (diviseur : 8).



(a) En noir et blanc.

(b) En 8 couleurs.

FIGURE 5.3.8 — Diffusion d'erreur de Crocker *et al.*

Steinberg comme elle n'a pas assez de coefficients et que l'erreur est diffusée trop crûment. Le résultat de la diffusion d'erreur de Crocker, Boulay et Morra est montré à la fig. 5.3.8.

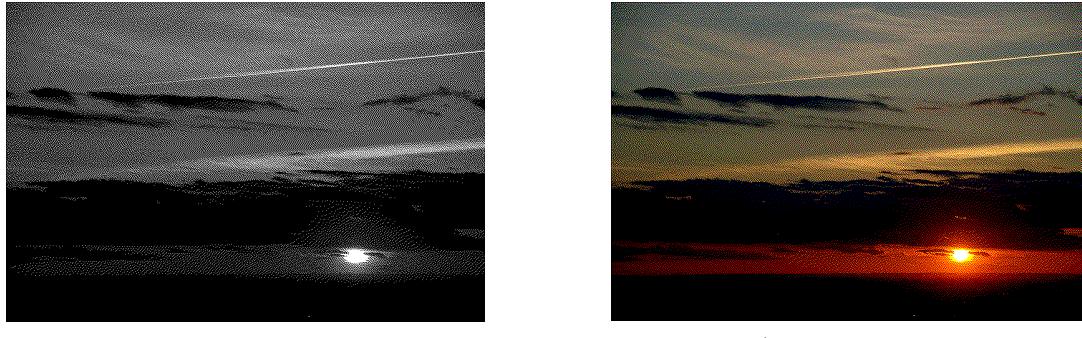
5.3.2.2 Méthode d'Atkinson

Utilisée par le logiciel MacPaint (1983), la diffusion d'erreur d'Atkinson se veut simple et efficace, sans coefficients compliqués, comme nous le montre la fig. 5.3.9. D'abord perçue comme une simple variation de la diffusion d'erreur de Floyd et Steinberg [175, p. 240], la diffusion d'erreur d'Atkinson apporte cependant un élément original : n'utiliser qu'une partie de l'erreur. En effet, comme nous le montre la fig. 5.3.9, la somme des coefficients est 6, mais le diviseur est 8, ce qui veut dire qu'au plus $\frac{3}{4}$ de l'erreur est utilisée¹. Cela a pour but de limiter la perturbation des pixels voisins et donc, on l'espère, préserver les détails. De plus, la disposition des coefficients, plus étendue, est choisie pour aider à propager l'erreur dans toutes les directions.

\hat{x}_t	1	1	
	1	1	1
		1	

FIGURE 5.3.9 — La matrice de diffusion d'erreur d'Atkinson (diviseur : 8).

1. Environ, car, n'oublions pas, nous faisons les calculs en arithmétique entière!



(a) En noir et blanc.

(b) En 8 couleurs.

FIGURE 5.3.10 — Diffusion d'erreur d'Atkinson.

$$\begin{array}{c} \tilde{x}_t \sqrt{7 \ 5} \\ \hline 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{array}$$

FIGURE 5.3.11 — La matrice de diffusion d'erreur de Jarvis, Judice et Ninke (diviseur : 46).

5.3.2.3 Méthode de Jarvis, Judice et Ninke

La matrice de diffusion de Jarvis, Judice et Ninke, présentée à la fig. 5.3.11, propose une diffusion radiale et proportionnelle à la distance : les voisins les plus près de \tilde{x}_t ont les plus gros coefficients [192]. On remarque que ni le diviseur ni les coefficients sont des puissances de deux : ils sont arbitraires. En effet, Jarvis *et al.* écrivent que le choix des coefficients n'affecte pas significativement l'image qui résulte de la diffusion d'erreur (ce qui est bien entendu *faux!*) [192, p. 37]. Quoi qu'il en soit, le résultat de la diffusion est montré à la fig. 5.3.12.

La matrice de diffusion, montrée à la fig. 5.3.11 est présentée dans [192] inversée : le raisonnement étant que ce n'est pas l'erreur sur le pixel à discréteriser qui passe aux voisins, mais la somme des erreurs *reçues* des voisins discréétisés qui est calculée. Naturellement, cela ne change rien au résultat, mais c'est une façon inusitée de présenter le calcul — qui sera reprise par Stucki.

5.3.2.4 Méthodes de Stucki

Stucki propose les méthodes MECCA (*Multiple-Error Correction Computation Algorithm*) pour l'impression en noir et blanc [345]. En fait, il s'agit de trois matrices de diffusion d'erreur, montrées à la fig. 5.3.13. Les tailles des matrices correspondent à différentes « étendues spatiales » censées correspondre à certaines réponses spectrales (c'est-à-dire, dans le domaine fréquence), mais les coefficients sont donnés sans justification. Les résultats de la diffusion d'erreur selon les trois matrices sont montrés à la fig. 5.3.14 pour le noir et blanc et à la fig. 5.3.15 pour les images en 8 couleurs.



(a) En noir et blanc.



(b) En 8 couleurs.

FIGURE 5.3.12 — Diffusion d'erreur de Jarvis, Judice et Ninke.

\tilde{x}_t	8	2
2	8	2
2		

(a) Petite matrice (diviseur 24).

\tilde{x}_t	8	4
2	4	8
1	2	4

(b) Moyenne matrice (diviseur 42).

\tilde{x}_t	8	4	2
2	4	8	8
2	4	4	4

(c) Grande matrice (diviseur 88).

FIGURE 5.3.13 — Matrices de diffusion d'erreur de Stucki.

5.3.2.5 Méthode de Burkes

La fig. 5.3.16 nous montre la matrice de diffusion d'erreur proposée par Burkes. Cette méthode serait présentée dans un communiqué que tout le monde cite, mais que personne n'a [83]; on doit s'en remettre à des sources secondaires [100]. Burkes tente, d'après Crocker *et al.*, d'améliorer la méthode de Stucki. Les coefficients sont des puissances de deux, permettant l'utilisation de décalages pour remplacer les multiplications et les divisions (voir la note en bas de la p. 154).



(a) Petite matrice.



(b) Moyenne matrice.



(c) Grande matrice.

FIGURE 5.3.14 — Diffusion d'erreur de Stucki en noir et blanc.

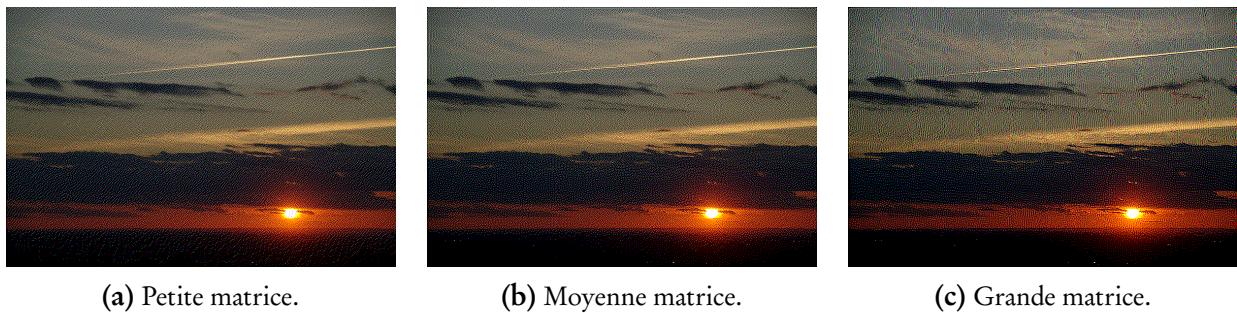


FIGURE 5.3.15 — Diffusion d'erreur de Stucki en 8 couleurs.

\tilde{x}_t	8	4	
2	4	8	4
	2		

FIGURE 5.3.16 — La matrice de diffusion d'erreur de Burkes (diviseur : 32).

5.3.2.6 Méthodes de Sierra

Frankie Sierra¹ propose en 1989 la matrice de diffusion « Sierra trois lignes » (montrée à la fig. 5.3.18 (a)), qui sera suivie, un an plus tard, de deux variantes, « Sierra deux lignes » et « *Sierra lite* » (montrées aux figures 5.3.18 (b) et 5.3.18 (c)), censées donner de meilleurs résultats que la diffusion d'erreur de Floyd et Steinberg. Dans les faits, seule la variante « Sierra trois lignes » est intéressante, tandis que les variantes — surtout la variante « *lite* » — montrent des motifs dérangeants.

Ici aussi on doit se contenter de sources secondaires [100, 195, 354]; la méthode aurait été divulguée sur un BBS qui, bien entendu, a été rejoindre le grand disque rigide dans le ciel depuis longtemps.

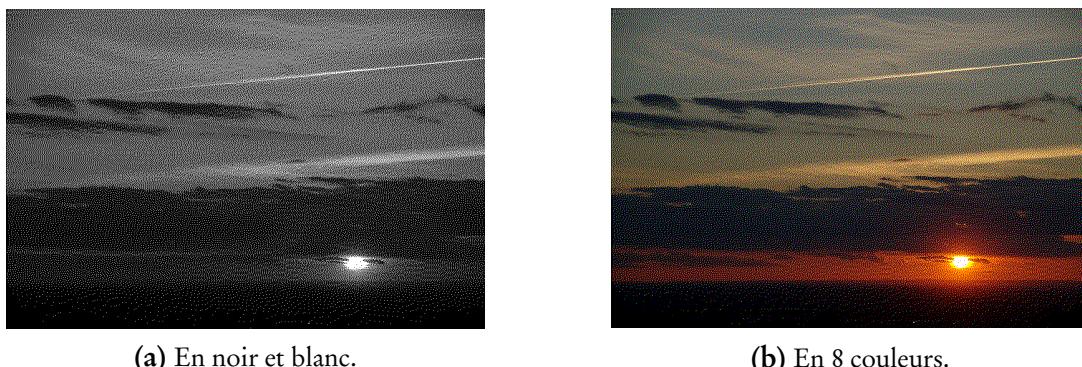


FIGURE 5.3.17 — Diffusion d'erreur de Burkes.

1. Sans relation avec Sierra On-Line, à qui nous devons la célèbre série de jeux d'aventure *King's Quest*.

\tilde{x}_t	5	3
2	4	5
4	2	
2	3	2

\tilde{x}_t	4	3
1	2	3
2	1	

\tilde{x}_t	2
1	1

(a) Matrice Sierra 3 (diviseur 32). (b) Matrice Sierra 2 (diviseur 16). (c) Matrice *Sierra Lite* (diviseur 4).

FIGURE 5.3.18 — Matrices de diffusion d'erreur de Sierra.

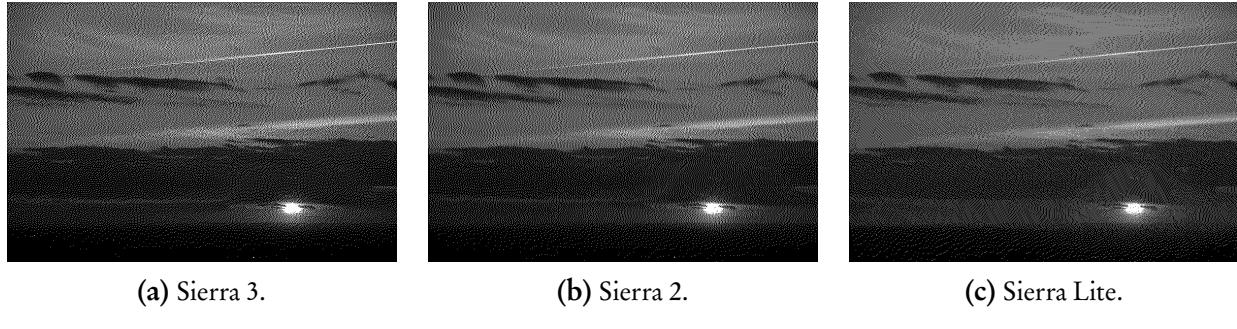


FIGURE 5.3.19 — Diffusion d'erreur de Sierra en noir et blanc.

5.3.2.7 Créer une méthode de diffusion d'erreur

Les méthodes de diffusion d'erreur que nous avons présenté jusqu'à maintenant sont des méthodes essentiellement *ad hoc*, c'est-à-dire bricolées de façon heuristique, où ni la forme du masque ni les coefficients ne sont choisis pour minimiser ou maximiser des contraintes explicites.

On voudrait quand même que

- La diffusion soit isotropique, c'est-à-dire qu'elle n'ait pas de direction de propagation préférée (ce qui nous éviterait, comme le font certaines méthodes, d'alterner le sens du balayage d'une ligne à l'autre).
- La diffusion soit compacte et préserve les détails; une matrice de diffusion trop grande va « lisser » l'image car les erreurs sont propagées plus loin.

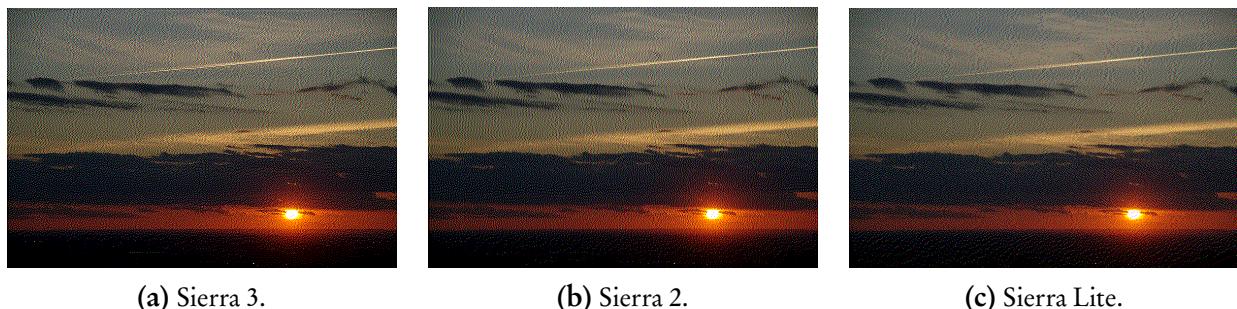


FIGURE 5.3.20 — Diffusion d'erreur de Sierra en 8 couleurs.

\tilde{x}_t	2	1
2	2	2
1	1	1

(a) Diffusion d'erreur de Pigeon, version 1
(diviseur : 12).

\tilde{x}_t	2	1
2	2	2
1	1	1

(b) Diffusion d'erreur de Pigeon, version 2
(diviseur : 14).

FIGURE 5.3.21 — La matrice de diffusion d'erreur de Pigeon [300].

\tilde{x}_t	1
1	1
1	1

(a) Diffusion d'erreur de Pigeon, version 3 (diviseur : 6).

\tilde{x}_t	1
1	1
1	1

(b) Diffusion d'erreur de Pigeon, version 3.1 (diviseur : 8).

\tilde{x}_t	1
1	1
1	1
1	1

(c) Diffusion d'erreur de Pigeon, version 3.2 (diviseur : 12).

FIGURE 5.3.22 — Les matrices de diffusion d'erreur de Pigeon, version 3.

- La diffusion ne crée pas de motifs évidents. En plus d'être visuellement dérangeants, ces motifs masqueront des détails dans l'image.

On veut par ailleurs que

- La géométrie de la matrice soit « facile » (peu de lignes, des coefficients simples, qu'il soit possible d'exploiter le jeu d'instruction de la machine façon efficace, etc.).
- Les coefficients soient de la même échelle que les valeurs et que la taille des entiers utilisés (par exemple, $7/8193$, c'est zéro en arithmétique 8 bits!).

Si la diffusion est isotropique, alors la matrice de diffusion doit contenir des coefficients qui sont disposés en motifs circulaires, symétriques, et décroissant rapidement (mais pas trop!), comme le suggère la fig. 5.3.4, p. 153.

En décembre 2013 j'ai proposé sur mon blog la méthode de diffusion montrée à la fig. 5.3.21 (a) [300]. La méthode a été reprise par d'autres [353]. L'heuristique — *c'est toujours une heuristique* — est de conserver la diffusion près de la source de l'erreur avec *un peu* de diffusion un peu plus loin. La fig. 5.3.21 (b) montre une variation où les « antennes » poussent les erreurs un peu plus loin, dans des directions opposées, dans l'espoir de déstructurer la propagation d'erreur et éviter la création de motifs dérangeants. La fig. 5.3.21 montre les résultats de ces deux matrices.

La fig. 5.3.22 montre des matrices où l'on cherche à diffuser l'erreur un peu plus loin, dans le but de « flouter » un peu l'image. Ces diffusions créent des motifs plus étendus et plus

\tilde{x}_t	16	4	2		32	8	4		32	8	4
2	4	8	16	8	4	2		3	7	20	32
2	4	4	4	2	2	4	6	3	20	20	7
2	2	2			2	2	3	2	7	5	3

(a) Diffusion d'erreur de Pigeon, version 4 (diviseur : 88). (b) Diffusion d'erreur de Pigeon, version 5 (diviseur : 184). (c) Diffusion d'erreur de Pigeon, version 5.1 (diviseur : 200).

FIGURE 5.3.23 — Les matrices de diffusion d'erreur de Pigeon, versions 4 et 5.

structurés spatialement, tout en étant suffisamment chaotiques pour éviter de générer des répétitions dérangeantes visuellement. Les résultats de ces matrices sont montrés à la fig 5.3.25.

Comme on l'a mentionné, on veut que la diffusion d'erreur soit telle que la plus grande part de l'erreur soit diffusée sur les voisins immédiats — ce qui va préserver les détails, on l'espère — et que le reste de l'erreur soit diffusé sur des pixels un peu plus loin, le tout en s'assurant de ne pas préférer de direction dans la propagation d'erreur. Cela suggère des matrices où les coefficients correspondent à une forme « en cloche », mais comment choisir les coefficients? On peut procéder heuristiquement (comme à la fig. 5.3.23 (a)) et y aller avec des coefficients qui paraissent approximer raisonnablement des cercles concentriques de poids qui décroissent en fonction de la distance, ou on peut calculer explicitement les coefficients en utilisant une métrique qui dépend de la distance (l'inverse de la distance au carré, ou à une autre puissance) normalisée par une constante qui dépend des valeurs admissibles pour les entiers utilisés pour faire le calcul, c'est-à-dire

$$W_p = \frac{s}{w\|p\|_2^2},$$

pour des positions p relatives à \tilde{x}_t (aux coordonnées $(0,0)$), une mise à l'échelle s , $s = 32$ par exemple, et une constante w qui dépend de la résolution ($w = 1$ semble raisonnable). La fig. 5.3.23 montre des matrices de diffusion avec une norme $\|x\|_2$ (fig. 5.3.23 (b)) et $\|x\|_3$ (fig. 5.3.23 (c)). Les deux utilisent $w = 1$ et $s = 32$. La figures 5.3.26 montre le résultat de ces matrices.

5.3.2.8 Diffusion d'erreur et autres géométries de pixels

Les méthodes que nous avons présenté jusqu'à maintenant supposent une grille de pixels approximativement carrés, comme les omniprésents écrans 16 : 9, mais ce n'est pas forcément le cas pour tous les écrans, ni pour toutes les technologies d'affichage [21]. Par exemple, pour la télévision analogique, on s'attend à des pixels légèrement plus larges que hauts (dans une proportion qui tourne autour de 12 : 11). Cette légère différence ne perturberait probablement pas significativement la diffusion d'erreur.

D'autres technologies, par contre, montrent des organisation de pixels différentes. Dans le cas d'une grille hexagonale, on pourra revoir les coefficients pour reprendre une organisation spatiale

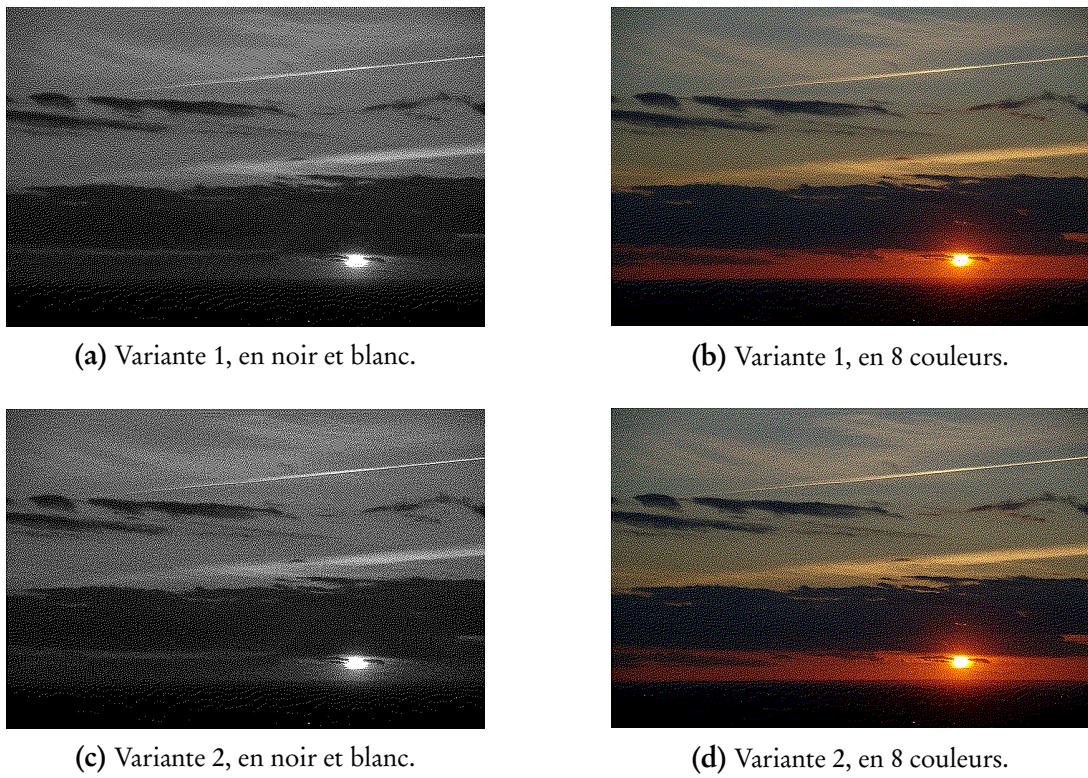


FIGURE 5.3.24 — Diffusion d'erreur de Pigeon, variantes 1 et 2.

cohérente, par exemple, comme proposé à la fig. 5.3.27 (a) [341]. Ici aussi, il s'agit de valeurs obtenues heuristiquement.

Si la géométrie des pixels est radicalement différente (comme ont les voit aux figs 5.3.27 (b) [322] et 5.3.27 (d) [78, 88]), on peut ignorer le problème et souhaiter que le pilote d'écran calcule une approximation raisonnable de l'image pour la matrice d'affichage, soit on peut intégrer la géométrie de l'écran dans la diffusion d'erreur [98]. Cela semble être un sujet de recherche encore actif.

5.3.2.9 Autres Méthodes

Si les matrices de diffusion d'erreur sont la méthode de prédilection, il existe bien entendu d'autres méthodes. Par exemple, Knuth fait remarquer que la diffusion d'erreur classique ne diffuse l'erreur que dans une direction (vers les pixels « futurs ») mais qu'il serait préférable de diffuser l'erreur dans toutes les directions [213]. Pour se faire, il propose d'utiliser une matrice de seuillage, un peu comme pour le tramage ordonné, composée d'une permutation (aléatoire?) des nombres 1 à 64, comme le montre la fig. 5.3.28.

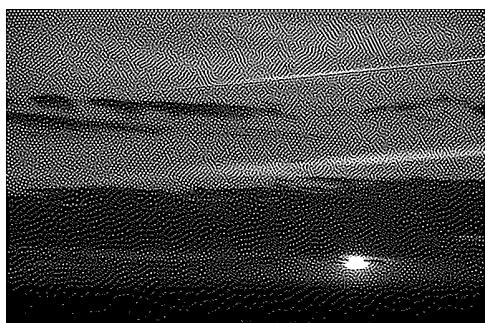
Ensuite, pour un pixel à la position (x, y) , on regarde dans la matrice à la rangée $1 + y \bmod 8$, colonne $1 + x \bmod 8$. On regarde les voisins de cette case. Si la valeur du voisin est plus grande, on



(a) Variante 3, en noir et blanc.



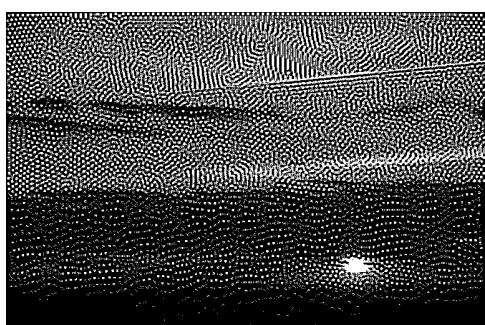
(b) Variante 3, en 8 couleurs.



(c) Variante 3.1, en noir et blanc.



(d) Variante 3.1, en 8 couleurs.



(e) Variante 3.2, en noir et blanc.



(f) Variante 3.2, en 8 couleurs.

FIGURE 5.3.25 — Diffusion d'erreur de Pigeon, variantes 3.

l'inclut dans la diffusion, sinon on ne l'inclut pas. Supposons que nous considérons un pixel qui corresponde à la rangée 5, colonne 2. La valeur est 35. En rouge, les voisins rejetés, en vert, les voisins retenus. Les voisins en vert reçoivent une portion de l'erreur qui dépend de la direction (nord, sud, est, ouest reçoivent $\frac{1}{6}$ ^{es} de l'erreur, les diagonales $\frac{1}{12}$ ^{es}). Cette méthode, bien qu'astucieuse, s'avère décevante, créant des motifs dérangeants et répétitifs — un défaut du tramage régulier!

Certains proposent plutôt d'utiliser une diffusion d'erreur sur des blocs plus larges que des pixels [101]. Ainsi, on pousse l'erreur qui serait normalement destinée à un seul pixel sur un bloc de 2×2 pixels, et c'est le bloc complet qui est seuillé et rendu grâce à l'un des motifs de base choisi d'avance.



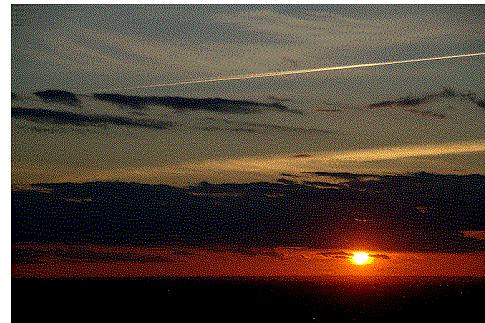
(a) Variante 4, en noir et blanc.



(b) Variante 4, en 8 couleurs.



(c) Variante 5, en noir et blanc.



(d) Variante 5, en 8 couleurs.



(e) Variante 5.1, en noir et blanc.

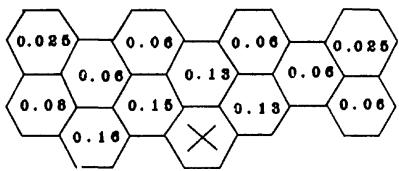


(f) Variante 5.1, en 8 couleurs.

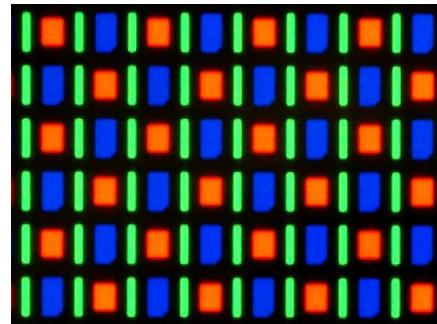
FIGURE 5.3.26 — Diffusion d'erreur de Pigeon, variantes 4 et 5.

Jarvis et Roberts proposent, quant à eux, un seuillage basé sur la moyenne locale dans une fenêtre de $n \times n$ pixels (leur implémentation est toutefois limitée aux fenêtres de 3×3 pixels) de façon à conserver le contraste local, en particulier dans les images textuelles [193]. La moyenne locale est obtenue via une pondération gaussienne et les seuils établis à partir des écarts-types.

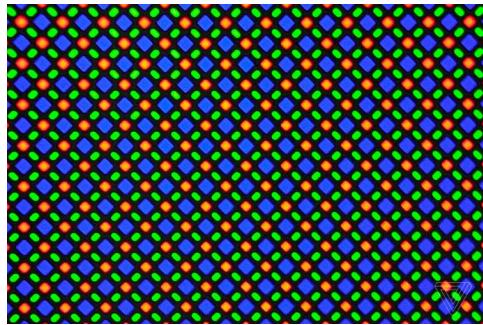
D'autres enfin proposent une diffusion d'erreur déportée en fonction de la position du pixel. Ici, on prend un bloc de 2×2 pixels et on distribue l'erreur sur son pourtour [128].



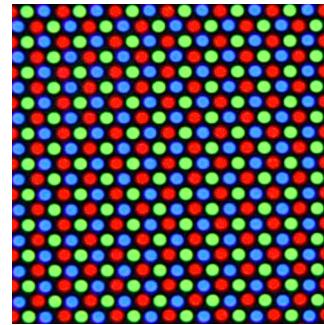
(a) Diffusion hexagonale de Stevenson et Arce [341].



(b) Écran OLED du téléphone Nexus One (image : Matthew Rollings, CC 3.0 [322]).



(c) Écran OLED du téléphone Galaxy S21.



(d) Écran cathodique d'ordinateur (image : Marcin Floryan, domaine public [127]).

FIGURE 5.3.27 — Diffusion d'erreurs et géométries variées.

25	21	13	39	47	57	53	45
48	16	56	24	50	18	58	26
12	44	4	36	14	46	6	38
60	28	52	20	62	30	54	22
3	35	11	43	1	33	9	41
51	19	59	27	49	17	57	25
15	47	7	39	13	45	5	37
63	31	55	23	61	29	53	21

FIGURE 5.3.28 — Matrice de seuillage de Knuth

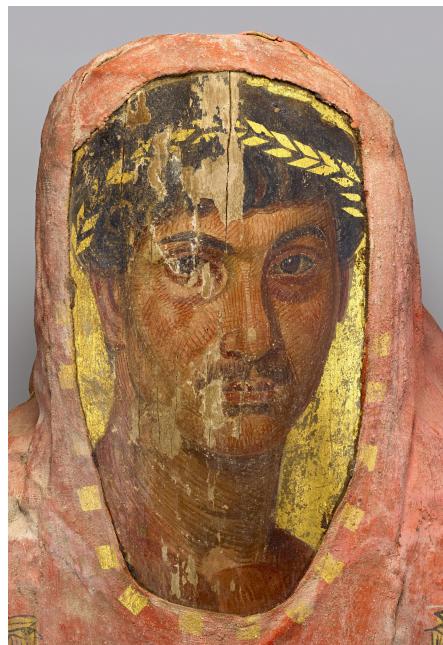


FIGURE 5.4.1 — La momie d'Héraclide, 120-140 ap. J.-C. (détail, image de domaine public).

5.4 Remarques bibliographiques

L'idée du tramage pour l'impression n'est pas récente, car c'est une invention du XIX^e siècle! Les graines de l'idée sont plantées dans les années 1840 par William Henry Fox Talbot (1800–1870). Il obtient un premier brevet pour un procédé d'impression des images sur un papier photographique [350], mais ce n'est pas encore du tramage. Les premiers à mettre en œuvre du véritable tramage sont deux Montréalais : George-Édouard-Amable Desbarats (1838–1893) [137] et William Augustus Leggo (1830–1915) [61]. En 1869, ils lancent les *Canadian Illustrated News*, qui montrent, pour la première fois une reproduction tramée d'une photographie (du duc Arthur de Connaught et Strathearn). Il faudra encore attendre quelques années pour que la technique se perfectionne et s'accommode de la couleur [181–183]. Gervais présente une histoire de la similigravure et du tramage dans [139].

*
* * *

L'idée d'utiliser des hachures ou des mouchetures pour obtenir des tons intermédiaires n'est certes pas nouvelle. En effet, on constate la technique sur de nombreuses œuvres d'art et même sur des portraits funéraires, comme le montre la momie d'Héraclide (*Herakleides*) que l'on peut voir au musée Getty [30]. Ici, des traits rouges sur un fond brun léger lui donnent l'aspect d'un mort très en santé avec de belles joues roses, comme on peut le voir à la fig. 5.4.1.

*

* *

Pour en savoir plus le tramage et les techniques de demi-teinte, voir Lau et Arce [224], ou encore Ulichney [359]. Pour une étude sur la visibilité des effets de demi-teinte, voir par exemple Näsänen [273]. Les points, trop réguliers, ou formant des rosettes, sont dérangeants visuellement. Pour mitiger ces effets, on peut perturber les points, c'est-à-dire les déplacer légèrement pour casser la régularité, une technique envisagée par Sullivan [347]. Ultimement, les techniques basées sur le bruit bleu (un bruit de haute fréquence) s'assurent que les perturbations ne créent pas de regroupements de points mais les espacent toujours suffisamment [163, 196, 222, 223, 225, 291].

*

* *

La méthode de diffusion d'erreur de MacPaint est souvent attribuée à Atkinson, mais il faut souvent s'en remettre aux sources secondaires. Hertzfeld, par exemple, considère la méthode d'Atkinson comme un simple « *modified Floyd-Steinberg Dithering* », qui aurait été utilisée dans le prédecesseur de MacPaint, MacSketch pour l'ordinateur Lisa, c. 1983 [175, 230]. Toutefois, une source solide — le code source du programme MacPaint [332] — pose définitivement Atkinson comme l'auteur de la méthode.

Bibliographie

La bibliographie qui suit contient les références, présentées en ordre alphabétique des auteurs. En principe, aucun des ouvrages cités n'est un « introuvable ». Pour plusieurs références, surtout les références aux articles relativement récents, nous vous invitons à vérifier auprès de votre institution quelles ententes ont été établies auprès des éditeurs. Les journaux de l'IEEE, de l'ACM et d'Elsevier sont souvent accessibles sans frais par l'intermédiaire de votre établissement scolaire, collégial ou universitaire. Plusieurs auteurs auront aussi, comme vous le découvrirez si vous utilisez un engin de recherche, mis à disposition leurs publications sur leurs pages personnelles. D'autres références, enfin, contiendront des URL qu'il vous suffira de suivre pour atteindre l'information.

- [1] *TMS9918A/TMS9928A/TMS9929A Video Display Processors* — Rapport technique, Texas Instruments (1982).
- [2] *MC6847/MC6847Y Video Display Generator (VDG)* — Rapport technique, Motorola Semiconductors (1984).
- [3] *V9938 MSX-Video Technical Data Book* — Rapport technique, Yamaha Nippon Gakki / ASCII Corporation (1985).
- [4] *Photography — Colour Negative Films for Still Photography — Determination of ISO Speed* — Rapport technique, ISO/IEC (novembre 1987).
- [5] *TIFF Revision 6 (FINAL)* — Rapport technique, Aldus Comporation (1992).
- [6] *ITU-T Recommendation H.261 : Video Codec For Audiovisual Services at $p \times 64$ kbits* — Rapport technique, International Telecommunication Union (1993).
- [7] *PhotoYCC Color Encoding and Compression Schemes* — Rapport technique n° 4, Kodak (1994).
- [8] *Xerox Color Encoding Standard XNSS 289005* — Rapport technique, Xerox (1994).
- [9] *CIFF Specification on Image Data File V. 1 r.4* — Rapport technique, Canon Inc. (1997).
- [10] *Enhanced Teletext Specification* — Rapport technique n° ETS 300 706, European Telecommunication Standard (1997).
- [11] *IEC/4WD 61966-2-1 : Colour Measurement and Management in Multimedia Systems and Equipment — Part 2-1 — Default Colour Space — sRGB* — Rapport technique, IEC (1998).
- [12] *ISO/IEC 61966-2-2 : Multimedia Systems and Equipment — Colour Measurement and Management — Part 2-2 — Colour Management — Extended Colour Space — scRGB* — Rapport technique, IEC (1998).
- [13] *Rec. ITU-R BT.470-6 : Conventional Television System* — Rapport technique, International Telecommunication Union (1998).

- [14] *Color Correction for Image Sensors* — Rapport technique n° MTD/PS-0534-2, (2003).
- [15] *Photography – Colour Reversal Camera Films – Determination of ISO Speed* — Rapport technique, ISO/IEC (octobre 2003).
- [16] *Rec. ITU-T.T.4 : Standardization of Group 3 Facsimile Terminals for Document Transmission* — Rapport technique, International Telecommunication Union (juillet 2003).
- [17] *ISO/IEC 15948 :2004 – Information Technology, Computer Graphics and Image Processing – Portable Network Graphics (PNG) Functional Specification* — Rapport technique, ISO/IEC (2004).
- [18] *SMPTE Standard for Television : Composite Analog Video Signal : NTSC for Studio Applications* — Rapport technique, SMPTE (2004).
- [19] *LizardTech DjVu Reference* — Rapport technique, LizardTech (2005).
- [20] *Code of Federal Regulation (CFF) §73.682 TV Transmission Standards* — Rapport technique, US Government Publications (2010).
- [21] *Rec. ITU-R BT.601.7 : Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide-Screen 16:9 Aspect Ratios* — Rapport technique, International Telecommunication Union (2011).
- [22] *CIPA DC-008-2012 Exchangeable Image File Format for Digital Still Cameras : EXIF Version 2.3* — Rapport technique, Camera & Imaging Products Association (2012).
- [23] *Digital Negative (DNG) Specification v1.4.0.0* — Rapport technique, Adobe Systems Inc. (2012).
- [24] *Rec. ITU-R BT.2020-2 : Parameter Values for the Ultra-High Definition Television Systems for Production and International Programme Exchange* — Rapport technique, International Telecommunication Union (2015).
- [25] *Rec. ITU-R BT.709-6 : Parameter Values for the HDTV Standards for Production and International Programme Exchange* — Rapport technique, International Telecommunication Union (2015).
- [26] *Rec. ITU-R BT.2100-1 : Image Parameter Values for High-Dynamic Range Television for Use in Production and International Programme Exchange* — Rapport technique, International Telecommunication Union (2017).
- [27] *Windows Metafile Format V 14.0* — Rapport technique, Microsoft (2017).
- [28] *Rolling Shutter* — https://en.wikipedia.org/wiki/Rolling_shutter (2020).
- [29] *The Atari ST and Amiga computing evolution!* — <https://retroshowcase.gr/index.php?p=article&artid=6> (2020).
- [30] *Mummy of Herakleides, A.D. 120–140* — <https://www.getty.edu/art/collection/object/103WN2> (juillet 2024).
- [31] R. L. Adler, B. P. Kitchens, M. Martens, C. P. Tresser, C. W. Wu — *The Mathematics of Halftoning* — IBM J. of Research and Development, vol. 47, n° 1 (janvier 2003), p. 5–15.
- [32] Tim Ahrens, Jack Browne, Hunter Scales — *What's Inside Radio Shack's Color Computer?* — BYTE, vol. 6, n° 3 (mars 1981), p. 90–130.
- [33] Josef Albers — *Interaction of color : 50th Anniversary Edition* — Yale University Press (2013).
- [34] Felix Albu, Vlad Paenaru, Alexandru Drimbarean — *US Patent 8,508,652 B2 : Autofocus Method* — US Patent Office (2013).
- [35] M. A. Ali, M. A. Klyne — *La vision chez les vertébrés* — Décarie/Masson (1986).

- [36] Jan P. Allebach, B. Liu — *Analysis of Halftone Dot Profile and Aliasing in the Discrete Binary Representation of Images* — J. of the Optical Society of America, vol. 67, n° 8 (septembre 1977), p. 1147–1154.
- [37] David Alleysson, Sabine Süsstrunk, Jeanny Hérault — *Color Demosaicing by Estimating Luminance and Opponent Chromatic Signals in the Fourier Domain* — dans Procs IS&T/SID 10th Color Imaging Conference, (novembre 2002), p. 331–336.
- [38] David Alleysson, Sabine Süsstrunk, Jeanny Hérault — *Linear Demosaicing Inspired by the Human Visual System* — IEEE Trans. on Image Processing, vol. 14, n° 4 (avril 2005), p. 439–449.
- [39] David Alleysson, Sabine Süsstrunk, Joanna Marguier — *Influence of Spectral Sensitivity Functions on Color Demosaicing* — dans Procs. Color and Imaging Conference, (2003), p. 351–357.
- [40] D. H. Alman, *et al.* (éds) — *Improvement to Industrial Colour-Difference Evaluation* — Rapport technique n° CIE 142-2001, CIÉ Central Bureau (2001).
- [41] Yas Abbas K. Alsultanny, Nidal Shilbayeh — *Applying Popularity Quantization algorithms on Color Satellite Images* — Pakistan J. of Applied Science, vol. 1, n° 4 (2001), p. 530–533.
- [42] G. F. Amelio, M. F. Tompsett, G. E. Smith — *Experimental Verification of the Charge Coupled Device Concept* — Bell System Technical Journal, vol. 49, n° 4 (1970), p. 593–600.
- [43] Deepak Aneja — *Edge Strength Based Fuzzification of Color Demosaicking Algorithms* — Mémoire de maîtrise, Dept. Information Technology, Delhi Technological University, (2013).
- [44] Gleb Anfilov — *Physics and Music* — MIR Publishers (1966).
- [45] David A. Atchison, George Smith — *Optics of the Human Eye* — Butterworth Heinemann (2000).
- [46] David Baker — *NASA Hubble Space Telescope : 1990 Onwards (Including All Upgrades), Owner's Workshop Manual* — Haynes Publishing (2015).
- [47] Raja Balasubramanian, Jan P. Allebach — *A New Approach to Palette Selection for Color Images* — Procs. SPIE, vol. 1453 (juin 1991), p. 58–69. (Human Vision, Visual Processing, and Digital Display II).
- [48] Raja Balasubramanian, Jan P. Allebach, Chales A. Bouman — *Color-Image Quantization with Use of a Fast Binary Splitting technique* — J. of the Optical Society of America A, vol. 11, n° 11 (novembre 1994), p. 2777–2786.
- [49] Octavian Baltag — *History of Automatic Focusing Reflected by Patents* — Science Innovation, vol. 4, n° 1 (2015), p. 1–17.
- [50] Bryce Bayer — *An Optimum Method for Low-Level Rendition of Continuous Tone Pictures* — dans Procs. IEEE Int. Conf. on Communications, (juin 1973), p. 26-11–26-15.
- [51] Bryce E. Bayer — *US Patent 3,971,065 : Color Imaging Array* — US Patent Office (1976).
- [52] Roberts Beaumont — *Colour in Woven Design* — George Bell & Sons (1890).
- [53] Charles A. Beichman, Marcia Rieke, Daniel Eisenstein, Thomas P. Greene, John Krist, Don McCarthy, Michael Meyer, John Stansberry, *et al.* — *Science Opportunities with the Near-IR Camera (NIRCam) on the James Webb Space Telescope (JWST)* — Procs. SPIE, vol. 8442 (septembre 2012), p. 8442N-1–8442N-11. (Space Telescopes and instrumentation : Optical, Infrared, and Millimeter Wave.).
- [54] Philippe Bellaïche — *Les secrets de l'image vidéo* — 11^e éd., Eyrolles (2018).
- [55] Richard Bellman — *The Theory of Dynamic Programming* — Rapport technique n° P-550, Rand Corporation (1954).

- [56] Richard Bellman — *Dynamic Programming* — Holt, Rinehart & Winston (1957). (*reprint chez Dover (1972)*).
- [57] Raymond Snyder Bennett W. Cooke, Silas C. Narland — *Coyne Television Cyclopedia* — Coyne Electrical and Television-Radio School (1951).
- [58] William Benson — *Principles of the Science of Colour, concisely stated to Aid and Promote Their Useful Application in the Decorative Arts* — Chapman & Hall (1868).
- [59] Marc de Berg, Mark van Kreveld, Mark Overmars, Otfried Schwarzkopf — *Computational Geometry : Algorithms and Applications* — 2^{de} éd., Springer (1998).
- [60] Gar A. Bergstedt — *US Patent 4,694,286 : Apparatus and Method for Modifying Displayed Color Images* — US Patent Office (septembre 1987).
- [61] Harry Black — *Canadian Scientists and Inventors : Biographies of People who Shaped our World* — 2^{de} éd., Pembroke Publishers (2008).
- [62] A. Boccaletti, P.-O. Lagage, P. Baudoz, C. Beichman, et al. — *The Mid-Infrared Instrument for the James Webb Space Telescope, V : Predicted Performance of the MIRI Coronographs* — Publications of the Astronomical Society of the Pacific, vol. 127, n° 953 (2015), p. 633–645.
- [63] Jean-Daniel Boissonnat, Mariette Yvinec — *Géométrie Algorithmique* — Édisciences (1995).
- [64] Dott. Anicio Bonucci — *Opere Volgari di Leon Battista Alberti (4 Vols)* — Typografia Galileiana (1843).
- [65] Léon Bottou — *Une approche théorique de l'apprentissage connexioniste ; applications à la reconnaissance de la parole* — Thèse de doctorat, Centre d'Orsay, Université Paris XI (février 1991).
- [66] Léon Bottou, Yoshua Bengio — *Convergence Properties of the K-Means Algorithm* — dans *Advances in Neural Information Processing Systems*, (1995), p. 585–592.
- [67] Patrice Bouchet, Macarena García-Marín, P.-O. Lagage, Jérôme Amiaux, et al. — *The Mid-Infrared Instrument for the James Webb Space Telescope, III : MIRIM, the MIRI Imager* — Publications of the Astronomical Society of the Pacific, vol. 127, n° 953 (2015), p. 612–622.
- [68] P. J. Bouma — *Physical Aspects of Color* — N. V. Philips Gloeilampenfabrieken (1947).
- [69] Charles A. Bouman, Michael T. Orchard — *Color Image Display with a Limited Palette Size* — Procs. SPIE, vol. 1199 (1989), p. 522–533. (Visual Communication and Image Processing IV).
- [70] T. Boutell, et al. — *RFC 2083 : PNG (Portable Network Graphics) Specification* — Rapport technique, IETF (1997).
- [71] Claude Boutet — *Traité de la peinture en Mignature pour apprendre aisément à Peindre sans Maître* — Louïs & Henry van Dole, La Haye (1708).
- [72] Robert Boyle — *Experiments and Considerations Touching Colours : The Beginning of an Experimental History of Colours* — Henry Herringman (1664).
- [73] W. S. Boyle, G. E. Smith — *Charge Coupled Semiconductor Devices* — Bell System Technical Journal, vol. 49, n° 3 (1970), p. 587–593.
- [74] C. Wayne Brown, Barry J. Shepherd — *GRaphics File Formats : Reference and Guide* — Manning (1995).
- [75] George Brown, D. G. C. Luck — *Principles and Development of Color Television Systems* — RCA Review, vol. XIV, n° 2 (juin 1953), p. 144–204.
- [76] W. R. J. Brown — *Color Discrimination of Twelve Observers* — J. of the Optical Society of America, vol. 47, n° 2 (février 1957), p. 808–834.

- [77] W. R. J. Brown, David L MacAdam — *Visual Sensitivities to Combined Chromaticity and Luminance Differences* — J. of the Optical Society of America, vol. 39, n° 10 (octobre 1949), p. 808–834.
- [78] C. H. Brown Elliot, T. L. Credelle, S. Han, M. H. Im, M. F. Higgins, P. Higgins — *Development of the Pentile™ Color AMLCD Subpixel Architecture and Rendering Algorithms* — J. of the Society for Information Display, vol. 11, n° 1 (mars 2003), p. 89–98.
- [79] Manlio Brusatin — *Histoire des couleurs* — Champs/Flammarion (1986).
- [80] Olof Bryngdahl — *Halftone Images : Spatial Resolution and Tone Reproduction* — J. of the Optical Society of America, vol. 68, n° 3 (mars 1978), p. 416–422.
- [81] Antoni Buades, Bartomeu Coll, Jean-Michel Morel, Catalina Sbert — *Self-Similarity Driven demosaicing* — Image Processing Online, vol. 1 (2011), p. 51–56.
- [82] Dan Budny — *What is IC_TC_P?* — Rapport technique n° V071, Dolby Systems (avril 2016).
- [83] Daniel Burkes — *Presentation of the Burkes Error Filter for Use in Preparing Continuous-Tone Images for Presentation in Bi-Level Devices* — Rapport technique n° LIB15, CIS Graphics Support Forum (septembre 1988).
- [84] Velma I. Burns — *Design and Calibration of Luminance Standards for the Three Color Phosphors used in Color Television Tubes* — Rapport technique n° 4870, National Bureau of Standards (octobre 1956).
- [85] Velma I. Burns — *CIÉ Colorimetry Committee—Working Program on Color Differences* — J. of the Optical Society of America, vol. 64, n° 6 (juin 1974), p. 896–897.
- [86] Gunter Buxbaum, *et al.* — *Colored Pigments* — dans Hans G. Völz, *et al.* (éds), *Ullmann's Encyclopedia of Industrial Chemistry*, vol. 2, John Wiley & Sons (2000), chapitre 3, p. 24–97.
- [87] Peter B. Catrysse, Bryan A. Wandell, Abbas El Gamal — *Comparative Analysis of Color Architectures for Image Sensors* — dans *Proc. Sensors, Cameras, and Applications of Digital Photography (Electronic Imaging '99)*, (1999), p. 26–35.
- [88] Sung-Ho Chae, Cheol-Hwan Yoo, Jee-Young Sun, Mun-Cheon Kang, Sung Jea Ko — *Subpixel Rendering for the Pentile Display Based on the Human Visual System* — IEEE Trans. on Consumer Electronics, vol. 63, n° 4 (novembre 2017), p. 401–409.
- [89] Ed Chang, Shiufun Cheung, Davis Pan — *Color Filter Array Recovery Using a Threshold-Based Variable Number of Gradients* — dans *Proc. Sensors, Cameras, and Applications of Digital Photography (Electronic Imaging '99)*, (1999), p. 36–43.
- [90] Lanlan Chang, Yap-Peng Tan — *Hybrid Color Filter Array Demosaicing for Effective Artifact Suppression* — Journal of Electronic Imaging, vol. 15, n° 1 (2006), p. 013003-1–013003-17.
- [91] Emmanuel Chevallier, Ivar Farup — *Interpolation of the MacAdam Ellipses* — Rapport technique n° HAL-01625921v2, HAL Archives Ouvertes (2018).
- [92] King-Hong Chung, Yuk-Hee Chan — *Color Demosaicing Using Variance of Color Differences* — IEEE Trans. on Image Processing, vol. 15, n° 10 (octobre 2006), p. 2944–2955.
- [93] T. M. Cleland — *A Grammar of Color : Arrangements of Strathmore Papers, in a Variety of Printed Color Combinations according to the Munsell Color System* — Strathmore Paper Company (1921).
- [94] John T. Compton, John F. Hamilton, Jr — *US Patent 8,139,130 B2 : Image Sensor with Improved Light Sensitivity* — US Patent Office (2012).
- [95] IBM Computers — *IBM PC Technical Reference* — Personal Computer Hardware Reference Library, IBM (1983).

- [96] IBM Computers — *IBM Color/Graphics Monitor Adapter* — Personal Computer Hardware Reference Library, IBM (1984).
- [97] Guy Côté, E. Frederiksen — *US Patent 8,638,342B2 : System and Method for Demosaicing Image Data Using Weighted Gradients* — US Patent Office (2014).
- [98] Julien Couillaud, Alain Horé, Djemel Ziou — *Nature-Inspired Color-Filter Array for Enhancing the Quality of Images* — J. of the Optical Society of America, Series A., vol. 29, n° 8 (août 2012), p. 1580–1587.
- [99] Michael A. Covington — *Astrophotography for the Amateur* — Cambridge University Press (1999).
- [100] Lee Daniel Crocker, Paul Poulay, Mike Morra — DHALF.TXT — (janvier 1989). (*Peut-être originalement un post USENET?*).
- [101] Niranjan Damera-Venkata, Brian L. Evans — *FM Halftoning via Block Error Diffusion* — dans *Proc. Int. Conf. on Image Processing (ICIP)*, (octobre 2001), p. 1081–1084.
- [102] Jan Davidse — *Transmission and Decoding in Colour Television* — Thèse de doctorat, Technische Hogeschool, Technische Uniseriteit Eindhoven (1964).
- [103] Hugh R. Davidson — *Calculation of Color Differences from Visual Sensitivity Ellipsoids* — J. of the Optical Society of America, vol. 41, n° 12 (décembre 1951), p. 1052–1056.
- [104] Hendrik de Lang, Gijsbertus Bouwhuis — *US Patent 3,202,039 : Optical System for a Color Television Camera* — US Patent Office (1965).
- [105] F. W. de Vrijer — *Fundamentals of Color Television* — Philips Technical Review, vol. 19, n° 3 (1957/1958), p. 86–97.
- [106] André Delorme — *Psychologie de la perception* — Université de Montréal (1995).
- [107] Maurice Déribéré — *La couleur* — n° 220 dans la collection *Que sais-je?*, Presses universitaires de France (1964).
- [108] V. G. Devereux — *Limiting of YUV Digital Video Signals* — Rapport technique n° BBC RD 1987/22, BBC Research Department (1987).
- [109] Peter L. P. Dillon, David M. Lewis, Frank G. Kaspar — *Color Imaging System Using a Single CCD Area Array* — IEEE Trans. on Consumer Electronics, vol. ED-25, n° 2 (février 1978), p. 102–107.
- [110] Greti Dinkova-Bruun, Giles E. M. Gasper, Micahel Huxatble, Tom C. b. McLeish, Cecilia Panti, Hannah E. Smithson — *The Dimensions of Colour : Robert Goressetest's De Colore* — Institute of Medieval and Renaissance Studies, Durham University (2013).
- [111] Jean Dourgnon, Paul Kowaliski — *La reproduction des couleurs* — n° 472 dans la collection *Que sais-je?*, Presses universitaires de France (1966).
- [112] Xavier Driancourt — *Optimisation par descente de gradient stochastique de systèmes modulaires combinant réseaux de neurones et programmation dynamique. Applications à la reconnaissance de la parole.* — Thèse de doctorat, Centre d'Orsay, Université Paris XI (février 1994).
- [113] Éric Dubois — *Frequency-Domain Methods for Demosaicing of Bayer-Sampled Color Images* — IEEE Signal Processing Letters, vol. 12, n° 12 (2005), p. 847–850.
- [114] Éric Dubois — *Filter Design for Adaptive Frequency-Domain Bayer Demosaicing* — dans *IEEE Int. Conf. on Image Processing (ICIP)*, (2006), p. 2705–2708.
- [115] Éric Dubois — *The Structure and Properties of Color Spaces and the Representation of Color Images* — Morgan & Claypool (2010).

- [116] Harold E Ennes — *Television Broadcasting : Equipment, Systems, and Operating Fundamentals* — Howard S. Sams & Co. (1970).
- [117] William H. Equitz — *A New Vector Quantization Clustering Algorithm* — IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. 37, n° 11 (octobre 1989), p. 1568–1575.
- [118] Hugh S. Fairman, Michael H. Brill, Henry Hemmendinger — *How the CIÉ 1931 Color-Matching Functions Were Derived from Wright-Guild Data* — Color Research and Application, vol. 32, n° 1 (février 1997), p. 11–13.
- [119] Sina Farsiu, Michael Elad, Peyman Milanfar — *Multi-Frame Demosaicing and Super-Resolution of Color Images* — IEEE Trans. on Image Processing, vol. 15, n° 1 (janvier 2006), p. 141–159.
- [120] Sina Farsiu, Dirk Robinson, Michael Elad, Peyman Milanfar — *Advances and Challenges in Super-Resolution* — Int. Journal of Imaging Systems and Technology, vol. 14, n° 2 (janvier 2004), p. 47–57.
- [121] Alan Fein, Ete Zoltan Suzts — *Photoreceptors : Their Role in Vision* — Cambridge university Press (1982).
- [122] Ted Felix — *Image Pac compression and JPEG compression : What's the Difference?* — <http://tedfelix.com/PhotoCD/ImagePacvsJPEG.html> (2020).
- [123] Donald G. Fink (éd.) — *Television Standards and Practice : Selected Papers form the Proceedings of the National Television System Committee and its Panels* — McGraw-Hill (1941).
- [124] Walter Fischer — *Digitale Fernsehtechnik in Theorie und Praxis* — Springer (2006).
- [125] Walter Fischer — *Digital Video and Audio Broadcasting Technology* — 3^e éd., Springer (2010).
- [126] Ken Fishkin — *A Fast HSL-to-RGB Transform* — dans Andrew S. Glassner (éd.), *Graphics Gems*, Academic Press (1990), chapitre VIII.14, p. 448–449.
- [127] Marcin Floryan — *Magnified view of a delta-gun shadow mask color CRT* — https://commons.wikimedia.org/wiki/File:CRT_screen._closeup.jpg (janvier 2007).
- [128] Marcin Floryan — *Libcaca : Error diffusion* — <http://caca.zoy.org/study/part3.html> (décembre 2009).
- [129] Robert W. Floyd, Louis Steinberg — *An Adaptive Algorithm for Spatial Grayscale* — Procs. of the Society for Information Display, vol. 17, n° 2 (Second Quarter 1976), p. 75–77.
- [130] Agner Fog — *Optimization Manuals, Vol 4 : Instruction Tables* — Technical University of Denmark (2019).
- [131] Edward W. Forgy — *(Abstract) Cluster Analysis of Multivariate Data : Efficiency versus Interpretability of Classifications* — Biometrics, vol. 21, n° 3 (septembre 1965), p. 768–769.
- [132] A. Forsey, S. gungor — *Demosaicing Images from Colour Cameras for Digital Image Correlation* — Optics and Lasers in Engineering, vol. 86 (2016), p. 20–28.
- [133] Eric R. Fossum — *Active Pixel Sensors : Are CCDs Dinosaurs?* — dans Procs. *Charged Compled Devices and Solid-State Optical Sensors III*, (1993), p. 2–14. (SPIE vol. 1900).
- [134] Eric R. Fossum, Donald B. Hondongwa — *A Review of the Pinned Photodiode for CCD and CMOS Image Sensors* — IEEE Journal of the Electron Devices Society, vol. 2, n° 3 (mai 2014), p. 33–43.
- [135] Robert Francès — *La perception* — n° 1076 dans la collection *Que sais-je?*, Presses universitaires de France (1963).
- [136] Franquin — *Gaston Lagaffe : Le gang des gaffeurs* — Dupuis (1973). (reprint 2016).

- [137] Claude Galarneau — *Les Desbarats : une dynastie d'imprimeur-éditeurs (1794-1893)* — Les cahiers des dix, vol. 46 (1991), p. 125–149.
- [138] Amandine Gallienne — *Les 100 mots de la couleur* — n° 4081 dans la collection *Que sais-je?*, Presses universitaires de France (2017).
- [139] Thierry Gervais — *La similigravure : le récit d'une invention (1878-1893)* — Nouvelles de l'estampe, n° 223 (mai 2010), p. 8–23.
- [140] Michael Gervautz, Werner Purgathofer — *A Simple Method for Color Quantization : Octree Quantization* — dans N. Magnenat-Thalmann, *et al.* (éds), *New Trends in Computer Graphics*, Springer (1988), p. 219–231.
- [141] Michael Gervautz, Werner Purgathofer — *A Simple Method for Color Quantization : Octree Quantization* — dans Andrew S. Glassner (éd.), *Graphics Gems*, Academic Press Professional (1990), chapitre IV.13, p. 287–293.
- [142] Pascal Getreuer — *Malvar-He-Cutler Linear Image Demosaicking* — Image Processing Online, vol. 1, p. 2011.
- [143] Alistair Glasse, G. H. Rieke, E. Bauwens, Macarena García-Marín, *et al.* — *The Mid-Infrared Instrument for the James Webb Space Telescope, IX : Predicted Sensibility* — Publications of the Astronomical Society of the Pacific, vol. 127, n° 953 (2015), p. 686–695.
- [144] Andrew S. Glassner (éd.) — *Graphics Gems* — Academic Press Professional (1990).
- [145] L. G. Glassner, A. H. McKinney, C. D. Reilly, P. D. Schnelle — *Cube-Root Color Coordinate System* — J. of the Optical Society of America, vol. 48, n° 10 (novembre 1958), p. 736–740.
- [146] Micha Galor Gluskin — *US Patent 100,044,959 B2 : Mask-Less Phase Detection Autofocus* — US Patent Office (2018).
- [147] Micha Galor Gluskin — *US Patent 10,044,926 B2 : Optimized Phase Detection Autofocus (PDAF) Processing* — US Patent Office (2018).
- [148] Pierre Godou — *Télévisions du monde : guide pratique de la réception longue distance* — Eyrolles (1983).
- [149] Rafael C. Gonzalez, Richard E. Woods — *Digital Image Processing* — 2^{de} éd., Prentice-Hall (2002).
- [150] Bart Goossens, Hiep Luong, Jan Aelterman, Alexandra Pižurica — *An Overview of State-of-the-Art Denoising and Demosaicing Techniques : Toward a Unified Framework for Handling Artifacts During Image Reconstruction* — dans *Procs. Int. Image Sensor Workshop (IISW)*, (2015). Session 12.
- [151] Karl D. Gordon, C. H. Chen, Rachel E. Anderson, Ruymán Azzollini, *et al.* — *The Mid-Infrared Instrument for the James Webb Space Telescope, X : Operations and Data Reductions* — Publications of the Astronomical Society of the Pacific, vol. 127, n° 953 (2015), p. 696–711.
- [152] Walter C. Granville — *The Color Harmoni Manual, a Color Atlas Based on the Ostwald Color System* — Color Research and Application, vol. 19, n° 2 (avril 1994), .
- [153] Jens Gravesen — *The Metric of Colour Space* — Graphical Models, vol. 82 (novembre 2015), p. 77–86.
- [154] Robert O. Green, Michael L. Eastwood, Charles M. Sarture, Thomas G. Chrien, Mikael Aronsson, Bruce J. chippendale, Jessica A. Faust, Betina E. Pavri, Christopher J. Chovit, Manuel Solis, Martin R. Olah — *Imaging Spectroscopy and the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)* — Remote Sensing of Environment, vol. 65, n° 3 (1998), p. 277–320.
- [155] Thomas P. Greene, Laurie Chu, Eiichi Egami, Klaus W. Hodapp an dDouglas M. Kelly, Jarron Leisenring, Marcia Rieke, Massimo Roberto, Everett Schlawin, John Stansberry — *Slitless Spectroscopy with*

- the James Webb Space Telescope Near-Infrared Camera (JWST NIRCam)* — Procs. SPIE, vol. 9904 (août 2016), p. 99040E-1-99040E-12. (Space Telescopes and instrumentation : Optical, Infrared, and Millimeter Wave.).
- [156] Matthew A. Greenhouse — *The JWST Science Instrument Payload : Mission Context and Status* — Procs. SPIE, vol. 8860 (octobre 2013), p. 886004-1-886004-11. (Space Telescopes and instruments : Innovative Technologies and Concepts VI.).
- [157] Bernard Grob — *Basic Television : Principles and Servicing* — MgGraw-Hill (1975).
- [158] Liron D. Grossmann, Yonina C. Eldar — *Enhancement of Color Images by Efficient Demosaicing* — Rapport technique n° CCIT #508, Technion-Israel Institute of Technology (2004).
- [159] J. Guild — *The Colorimetric Properties of the Spectrum* — Philosophical Trans. of the Royal Society of London, Series A : Containing Papers of a Mathematical or Physical Character, vol. 230 (1932), p. 149–187.
- [160] Bahadir K. Gunturk, Yucel Altunbasak, Russell M. Mersereau — *Color Plane Interpolation Using Alternating Projections* — IEEE Trans. on Image Processing, vol. 11, n° 9 (septembre 2002), p. 997–1013.
- [161] Bahadir K. Gunturk, John Glotzbach, Yucel Altunbasak, Ronald W. Schafer, Russell M. Mersereau — *Demosaicking : Color Filter Interpolation* — IEEE Signal Processing Magazine, vol. 22, n° 1 (janvier 2005), p. 44–54.
- [162] Maya R. Gupta, Ting Chen — *Vector Color Filter Demosaicing* — dans *Sensors and Camera Systems for Scientific, Industrial and Digital Applications II*, (2001). (SPIE Vol. 4306).
- [163] Dmitri Gusev — *Anti-Correlation Digital Halftoning* — Thèse de doctorat, Computer Science Dept., Indiana University Bloomington (1993).
- [164] R. V. L. Hartley — “TU” Becomes “Decibel” — Bell Laboratories Record, vol. 7, n° 4 (décembre 1928), p. 137–139.
- [165] Stephen Hawley — *Ordered Dithering* — dans Andrew S. Glassner (éd.), *Graphics Gems*, Academic Press Professional (1990), chapitre III.5, p. 176–178.
- [166] Kengo Hayasaka, Kenji Yamamoto — *Japan Patent JP2011-171858A : Image Processing Apparatus, Image Processing Method, Image Processing Program and Imaging Device* — Japan Patent Office (2011).
- [167] Eugene Hecht — *Optics* — 5^e éd., Pearson (2017).
- [168] Paul S. Heckbert — *Color Image Quantization for Frame Buffer Display* — Mémoire de maîtrise, Dept. Mathematics, MIT, (1980). (Thèse de bac?).
- [169] Paul S. Heckbert — *Color Image Quantization for Frame Buffer Display* — Computer Graphics, vol. 16, n° 3 (juillet 1992), p. 297–307. (SIGGRAPH’82).
- [170] Yacov Hel-Or — *US Patent 6,404,918(B1) : Image Demosaicing Method Utilizing Directional Smoothing* — US Patent Office (2002).
- [171] Yacov Hel-Or — *The Canonical Correlations of Color Images and Their Use for Demosaicing* — Rapport technique n° HPL-2003-164(R.1), HP Laboratories (février 2004).
- [172] Scott Helt — *Practical Television Engineering* — 2^{de} éd., Rinehart Books (1953).
- [173] John L. Hennessy, David A. Patterson — *Computer Architecture : A Quantitative Approach* — Morgan Kaufmann (2006).
- [174] Ewald Hering — *Zur lehre von Lichtsinne* — Druck und Verlag Carl Gerold’s Sohn (1878).

- [175] Andy Hertzfeld — *Revolution in The Valley : The Insanely Great Story of How the Mac Was Made* — O'Reilly (2005).
- [176] Keigo Hirakawa, Thomas W. Parks — *Adaptive Homogeneity-Directed Demosaicing Algorithm* — IEEE Trans. on Image Processing, vol. 14, n° 3 (mars 2005), p. 360–369.
- [177] Keigo Hirakawa, Thomas W. Parks — *Joint Demosaicing and Denoising* — dans *IEEE Int. Conf. on Image Processing (ICIP)*, (2005).
- [178] Keigo Hirakawa, Thomas W. Parks — *Joint Demosaicing and Denoising* — IEEE Trans. on Image Processing, vol. 15, n° 8 (août 2006), p. 2146–2157.
- [179] Keigo Hirakawa, Patrick J. Wolfe — *Second-Generation Color Filter Array and Demosaicking Designs* — dans *Procs. Visual Communications and Image Processing*, (2008), p. 68221P-1–68221P-12. (SPIE Vol. 6822).
- [180] Keigo Hirakawa, Patrick J. Wolfe — *Spatio-Spectral Color Filter Array Design for Enhanced Image Fidelity* — dans *Procs. IEEE Int. Conf. on Image Processing (ICIP)*, (2010), p. II-81–II-94.
- [181] August Hoen — *US Patent 27,981 : Composition for Etching Stone* — US Patent Office (avril 1860).
- [182] August Hoen — *US Patent 227,782 : Lithographic Process* — US Patent Office (mai 1880).
- [183] August Hoen — *US Patent 227,730 : Lithographic Process* — US Patent Office (mai 1883).
- [184] Harold Hotelling — *Analysis of a Complex of Statistical Variables into Principal Components* — J. of Educational Psychology, vol. 24, n° 6 (1933), p. 417–441.
- [185] Harold Hotelling — *Relations Between Two Sets of Variates* — Biometrika, vol. 28, n° 4 (décembre 1936), p. 321–377.
- [186] Jong-Uk Hou, Han-Ul Jang, Heung-Kyu Lee — *Hue Modification Estimation Using Sensor Pattern Noise* — dans *Procs. IEEE Int. Conf. on Image Processing (ICIP)*, (2014), p. 5281–5291.
- [187] P. H. Howells — *The Concept of Transmission Primaries in Color Television* — Procs. of the IRE, vol. 42, n° 1 (janvier 1954), p. 134–138.
- [188] Anders Hård, Lars Sivik, Gunnar Tonnquist — *NCS, Natural Color System – From Concept to Research and Applications, Part I* — Color Research and Application, vol. 21, n° 3 (juin 1996), p. 180–205.
- [189] Anders Hård, Lars Sivik, Gunnar Tonnquist — *NCS, Natural Color System – From Concept to Research and Applications, Part II* — Color Research and Application, vol. 21, n° 3 (juin 1996), p. 206–220.
- [190] Yizhen Huang, Yangjing Long — *Demosacking Recognition with Application in Digital Photo Authentication Based on a Quadratic Pixel Correlation Model* — dans *Procs. IEEE Conf. on Computer Vision and Pattern Recognition*, (2008), p. 1–8.
- [191] Kai Hwang — *Advanced Computer Architecture : Parallelism, Scalability, Programmability* — McGraw-Hill (1993).
- [192] J. F. Jarvis, C. N. Judice, W. H. Ninke — *A Survey of Techniques for the Display of Continuous Tone Pictures on Bilevel Displays* — Computer Graphics and Image Processing, vol. 5, n° 1 (mars 1976), p. 13–40.
- [193] J. F. Jarvis, C. S. Roberts — *Concise Papers : A New Technique for Displaying Continuous Tone Images on a Bilevel Displays* — IEEE Trans. on Communications, vol. 24, n° 8 (août 1976), p. 891–898.
- [194] Sir James Jeans — *Science & Music* — The Macmillan Company (1937).
- [195] Lü Jiangzhao, Da Feipeng, zheng Dongliang — *Projector Defocussing Profilometry Based on Sierra Lite Dithering Algorithm* — Acta Optica Sinica, vol. 34, n° 3 (mars 2014), p. 0312004-1–0312004-9. En chinois.

- [196] Pierre-Marc Jodoin, Victor Ostromoukhov — *Error-Diffusion with Blue-Noise Properties for Mid-tones* — Procs. SPIE, vol. 4663 : Color Imaging : Device Independent Color, Color Hardcopy, and Applications VII (décembre 2001), p. 293–301.
- [197] K. S. Johnson — *Transmission Circuits for Telephonic Communications : Methods of Analysis and Design* — D. van Nostrand Co. (1925).
- [198] I.T. Jolliffe — *Principal Component Analysis* — Springer-Verlag (1986).
- [199] Deane B. Judd — *A Maxwell Triangle Yielding Uniform Chromaticity Scales* — J. of the Optical Society of America, vol. 25, n° 1 (janvier 1935), p. 24–35.
- [200] Deane B. Judd, Dorothy Nickerson — *One Set of Munsell Re-renotations* — Rapport technique n° 192693, National Bureau of Standards (juin 1967).
- [201] Bela Julesz — *Foundation of Cyclopean Perception* — University of Chicago Press (1971).
- [202] David Kahn — *The Codebreakers* — MacMillan (1967).
- [203] Henry R. Kang — *Color Technology for Electronic Imaging Devices* — SPIE Optical Engineering Press (1996).
- [204] Kari Karhunen — *Über Lineare Methoden in der Wahrscheinlichkeitsrechnung* — Suomalainen Tiedeakatemia (1947).
- [205] David C. Kay, John R. Levine — *Graphics File Formats* — TAB Books (1992).
- [206] Sarah Kendrew, Silvia Scheithauer, Patrice Bouchet, Jérôme Amiaux, et al. — *The Mid-Infrared Instrument for the James Webb Space Telescope, IV : The Low-Resolution Spectrometer* — Publications of the Astronomical Society of the Pacific, vol. 127, n° 953 (2015), p. 623–632.
- [207] Daniel Keren, Maragarita Osadchy — *Restoring Subsampled Color Images* — Machine Vision and Applications, vol. 11, n° 4 (décembre 1999), p. 197–202.
- [208] Ron Kimmel — *Demosaicing : Image Reconstruction from Color CCD Samples* — IEEE Trans. on Image Processing, vol. 8, n° 9 (septembre 1999), p. 1221–1228.
- [209] Ihor O. Kirenko, Ling Shao — *Local Objective Metrics of Blocking Artifacts Visibility for Adaptive Repair of Compressed Video Signals* — dans *IEEE Int. Conf. on Multimedia and Expo*, (2007), p. 1303–1306.
- [210] Timothy Knight, Colving Pitts, Kurt Akely, Yurly Romanenko, Carl (Warren) Craddock — *US Patent 9,300,932 B2 : Optimization of Optical Systems for Improved Lightfield Capture and Manipulation* — US Patent Office (2016).
- [211] Karl Knop, Rudolf Morf — *A New Class of Mosaic Color Encoding Patterns for Single-Chip Camera* — IEEE Trans. on Electronic Devices, vol. ED-32, n° 8 (août 1985), p. 1390–1395.
- [212] Ken Knowlton, Leon Harmon — *Computer-Produced Grey Scales* — Computer Graphics and Image Processing, vol. 1, n° 1 (avril 1972), p. 1–20.
- [213] Donald E. Knuth — *Digital Halftones by Dot Diffusion* — ACM Trans. on Graphics, vol. 6, n° 4 (octobre 1987), p. 245–273.
- [214] Jan Koenderink, Andrea van Doorn, Karl Gegenfurtner — *Graininess of RGB-Display Space* — iPerception, vol. 9, n° 5 (2018), p. 1–46.
- [215] Erich Kolbenheyer — *Motive Der Hausindustriellen Stickerei in der Bukowina / Motifs de la broderie paysanne en Bukovine / Design of the Home-Industry Embroideries in Bukovina* — Ministère des travaux publics et de la diète de la Bukovine (1912).

- [216] Arthur König — *Über den Helligkeitswert der Spektralfarben bei Verschiedener absoluter Intensität* — Verlag von Leopold Voss (1891).
- [217] Eugene F. Krause — *Taxicab Geometry : An Adventure in Non-Euclidean Geometry* — Addison-Wesley (1975).
- [218] Rolf G. Kuehni, Andreas Schwarz — *Color Ordered : A Survey of Color Order Systems, from Antiquity to the Present* — Oxford University Press (2008).
- [219] Yosuke Kusaka — *US Patent 5589909 : Automatic Focus Adjustment Camera with High-Speed Sequential Photography Capability and Method of Automatic Focus Adjustment Therefor* — US Patent Office (1996).
- [220] A. S. Kutyrev, N. Collins, J. Chambers, S. H. Moseley, D. Rapchun, et al. — *Microshutter Arrays : High Contrast Programmable Field Masks for JWST NIRSpec* — Procs. SPIE, vol. 2008 (août 2008), p. 8860070103D-1–70103D-10. (Space Telescopes and instrumentation : Optical, Infrared, and Millimeter Wave.).
- [221] Emily Lakdawalla — *The Design and Engineering of Curiosity : How the Mars Rover Performs its Job* — Springer (2018).
- [222] Daniel L. Lau, Gonzalo R. Arce — *Digital Halftoning by Means of Green-Noise Masks* — J. of the Optical Society of America, vol. 16, n° 7 (juillet 1999), p. 1575–1586.
- [223] Daniel L. Lau, Gonzalo R. Arce — *Digital Halftoning using Green-Noise Masks* — dans *Proc. 15th Int. Conf. on Digital Printing Technologies (NIP15)*, (octobre 1999), p. 358–361.
- [224] Daniel L. Lau, Gonzalo R. Arce — *Modern Digital Halftoning* — CRC Press (2008).
- [225] Daniel L. Lau, Robert Ulichney — *Blue-Noise Halftoning for Hexagonal Grids* — Rapport technique n° HPL-2004-232, Hewlett-Packard (décembre 2004).
- [226] Eugene Lawler — *Combinatorial Optimization : Networks and Matroids* — Dover (2001).
- [227] Jakob Christoffel Le Blon — *L'art d'imprimer les tableaux* — P. G. Le Mercier, Imprimeur-libraire (1756).
- [228] Jakob Christoffel Le Blon — *Coloritto; or the Harmony of Colouring in Painting, reduced to the Mechanical Practice under Easy Precepts, and Infallible Rules, together with some Colour'd Figures, in order to render the said precepts and rules intelligible, not only to painters but even to all lovers of Painting* — (inconnu) (c. 1725).
- [229] Sheldon Leemon — *Inside Amiga Graphics* — Compute! Publications (1986).
- [230] Phil Lemmons — *An Interview : The Macintosh Design Team* — Byte, vol. 9, n° 2 (février 1984), p. 58–64,66,68,70,72,74,76,78,80.
- [231] Xin Li — *Demosaicing by Successive Approximations* — IEEE Trans. on Image Processing, vol. 14, n° 3 (mars 2005), p. 370–379.
- [232] Xin Li, Bahadir K. Gunturk, Lei Zhang — *Image Demosaicing : A Systematic Survey* — dans *Proc. Visual Communications and Image Processing*, (2008), p. 68221J-1–68221J-15. (SPIE Vol. 6822).
- [233] Chuan-Kai Lin — *Demosaic* — <https://sites.google.com/site/chklin/demosaic/> (2020).
- [234] Chuan-Kai Lin — *History of the Portable Network Graphics Format* — <http://www.libpng.org/pub/png/book/chapter07.html> (2020).
- [235] Tasnn-Shyong Liu, Long-Wen Chang — *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)* — dans *Greedy Tree Growing for Color Image Quantization*, (1994), p. v-97–v-100.

- [236] Stuart P. LLoyd — *Least Square Quantization in PCM* — IEEE Trans. on Information Processing, vol. 28, n° 2 (mars 1982), p. 129–137.
- [237] Michel Loèv — *Probability Theory* — D. van Nostrand (1955).
- [238] Philippe Longère, Xuemei Zhang, Peter B. Delahunt, David H. Brainhard — *Perceptual Assessment of Demosaicing Algorithm Performance* — Procs. of the IEEE, vol. 90, n° 1 (janvier 2002), p. 132–132.
- [239] Ralph Lorenz — *NASA/ESA/ASI Cassini-Huygens : 1997 Onwards (Cassini Orbiter, Huygens Probe and Future Exploration Concepts)* — Haynes Publishing (2017).
- [240] Olivier Lossen, Ludovic Macaire, Yanquin Yang — *Comparison of Color Demosaicing Methods* — dans *Advances in Imaging and Electronic Physics*, (2010), p. 173–265.
- [241] A. Lukin, D. Kubasov — *High-Quality Algorithm for Bayer Pattern Interpolation* — Programming and Computer Software, vol. 30, n° 6 (2004), p. 347–358.
- [242] Alexey Lukin, Denis Kubasov — *An Imprived Demosaicing Algorithm* — dans *Procs. 14th INT. Conf. on Computer Graphics (GRAPHICON)*, (2004), p. 38–45.
- [243] Ming Ronnier Luo — *New Colour-Difference Formulae for Surface Colours* — Thèse de doctorat, School of Studies in Colour Chemistry and Colour Technology, University of Bradford (1986).
- [244] David L. MacAdam — *Projective Transformations of I.C.I. Color Specifications* — J. of the Optical Society of America, vol. 27, n° 8 (août 1937), p. 294–299.
- [245] David L MacAdam — *Visual Sensitivities to Color Differences in Daylight* — J. of the Optical Society of America, vol. 32, n° 5 (mai 1942), p. 247–274.
- [246] David L MacAdam — *Specification of Small Chromaticity Differences* — J. of the Optical Society of America, vol. 33, n° 1 (janvier 1943), p. 18–26.
- [247] David L. MacAdam — *On the Geometry of Color Space* — J. of the Franklin Institute, vol. 238, n° 3 (septembre 1944), p. 195–210.
- [248] David L MacAdam — *Analytical Approximations for Color Metric Coefficients* — J. of the Optical Society of America, vol. 47, n° 4 (avril 1957), p. 268–274.
- [249] David L MacAdam — *Analytical Approximations for Color Metric Coefficients II : Friele Approximations* — J. of the Optical Society of America, vol. 54, n° 2 (février 1964), p. 249–256.
- [250] David L MacAdam — *Analytical Approximations for Color Metric Coefficients III : Optimization of Parameters in Friele's Formulas* — J. of the Optical Society of America, vol. 54, n° 9 (septembre 1964), p. 1161–1165.
- [251] David L. MacAdam — *Color measurement : Theme and variations* — Springer-Verlag (1981).
- [252] J. MacQueen — *Some Methods for Classification and Analysis of Multivariate Observations* — dans *Procs. Berkeley Symposium on Mathematical Statistics and Probability*, (1967), p. 281–297.
- [253] Oded Maimon, Lior Rokach (éds) — *Data Mining and Knowledge Discovery Handbook* — 2^{de} éd., Springer (2010).
- [254] Henrique Malvar, Li-Wei He, Ross Cutler — *High-Quality Linear Interpolation for Demosaicing of Bayer-Patterned Color Images* — dans *Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, (2004), p. III-485–III-488.
- [255] Henrique S. Malvar, Gary J. Sullivan — *$YC_oC_g - R$: A color Space with RGB Reversibility and Low Dynamic Range* — Rapport technique n° JVT-I014r3, Joint Video Team ISO/IEC (2003).

- [256] Henrique S. Malvar, Gary J. Sullivan — *Lifting-Based Reversible Color Transformations for Image Compression* — dans *Proc. SPIE XXXI (vol. 7073) Application of Digital Image Processing*, SPIE (2008), p. 707307-1–707307-10.
- [257] Pavlos Mavridis, Georgios Papaioannou — *The Compact YC_oC_g Frame Buffer* — *J. of Computer Graphics Techniques*, vol. 1, n° 1 (2012), p. 19–35.
- [258] Joshua R. Maxwell — *Assessing Color Discrimination* — Mémoire de maîtrise, Dept. of Human Factors & Systems, Embry-Riddle Aeronautical University, Daytona Beach (2012).
- [259] Donald McIntyre — *Colour Blindness : Causes and Effects* — Dalton Publishing (2002).
- [260] K. McLaren — *XII : The SDC Recommended Colour-Difference Formula : Change to CIÉLAB* — *J. of the Society of Dyers and Colourists*, vol. 92, n° 9 (septembre 1976), p. 337–338.
- [261] K. McLaren — *XIII : The Development of the CIÉ 1976 ($L^*a^*b^*$) Uniform Colour Space and Colour-Difference Formula* — *J. of the Society of Dyers and Colourists*, vol. 92, n° 9 (septembre 1976), p. 338–341.
- [262] Daniele Menon, Stefano Adriani, Giancarlo Calvagno — *Demosaicing with Directional Filtering and a posteriori Decision* — *IEEE Trans. on Image Processing*, vol. 16, n° 1 (janvier 2017), p. 132–141.
- [263] Daniele Menon, Giancarlo Calvagno — *A Regularized Approach to Demosaicing* — dans *Proc. Visual Communications and Image Processing*, (2008), p. 68221L-1–68221L-11. (SPIE Vol. 6822).
- [264] Albert Henry Munsell — *A Pigment Color System and Notation* — *The American Journal of Psychology*, vol. 23, n° 2 (avril 1912), p. 256–244.
- [265] Albert Henry Munsell — *Atlas of the Munsell Color System* — Munsell Color Company (1915).
- [266] Albert Henry Munsell — *A Color Notation* — Munsell Color Company (1919).
- [267] D. Darian Muresan, Steve Luke, Thomas W. Parks — *Reconstructing of Color Images from CCD Arrays* — dans *TIDSP Fest*, (2000).
- [268] D. Darian Muresan, Thomas W. Parks — *Optimal Recovery Demosaicing* — dans *IASTED Signal And Image Processing*, (2002).
- [269] D. Darian Muresan, Thomas Porter — *Demosaicing Using Optimal Recovery* — *IEEE Trans. on Image Processing*, vol. 14, n° 2 (février 2005), p. 267–278.
- [270] James D. Murray, William van Ryper — *Encyclopedia of Graphics File Formats* — 2^{de} éd., O'Reilly (1996).
- [271] Wayne L. Myers, Ganapati P. Patil (éds) — *Pattern-Based Compression of Multi-Band Image Data for Landscape Analysis* — Springer (2006).
- [272] Junichi Nakamura — *Image Sensors and Signal Processing for Digital Still Camera* — Taylor & Francis (2006).
- [273] Risto Näsänen — *Visibility of Halftone Dot Textures* — *IEEE Trans. on Systems, Man, and Cybernetics*, n° 6 (nov-déc 1984), p. 920–924.
- [274] Sydney M. Newhall, Dorothy Nickerson, Deane B. Judd — *Final Report of the O.S.A. Subcommittee on the Spacing of the Munsell Colors* — *J. of the Optical Society of America*, vol. 33, n° 7 (juillet 1943), p. 385–418.
- [275] Dev R. Newlin, Elwin Chandra Monie — *Edge Sensing Demosaicing Using Adaptive Weighted Interpolation* — *American Journal of Applied Sciences*, vol. 10, n° 4 (2013), p. 418–425.

- [276] Isaac Newton — *Opticks : Or, a Treatise of the Reflexions, Refractions, Inflexions and Colours of Light; also Two Treatises of the Species and Magnitude of Curvilinear Figures* — Smith & Walford, London (1704).
- [277] Nhat Xuan Nguyen — *Numerical Algorithms for Image Superresolution* — Thèse de doctorat, Dept. Scientific Computing and Computational Mathematics, Stanford University (2000).
- [278] Nhat Xuan Nguyen, Peyman Milanfar, Gene Golub — *A Computationally Efficient Superresolution Image Reconstruction Algorithm* — IEEE Trans. on Image Processing, vol. 10, n° 4 (avril 2001), p. 573–583.
- [279] Dorothy Nickerson — *History of the Munsell Color System, Company, and Foundation, I* — Color Research and Application, vol. 1, n° 1 (1976), p. 7–10.
- [280] Dorothy Nickerson — *History of the Munsell Color System, Company, and Foundation, II* — Color Research and Application, vol. 1, n° 2 (1976), p. 69–77.
- [281] Dorothy Nickerson — *History of the Munsell Color System, Company, and Foundation, III* — Color Research and Application, vol. 1, n° 3 (1976), p. 121–130.
- [282] Yu-Ichi Ohta — *A Region-Oriented Image-Analysis System by Computer* — Thèse de doctorat, Department of Information Science, Kyoto University (mars 1980).
- [283] Yu-Ichi Ohta, Takeo Kanada, Toshiyuki Sakai — *Color Information for Region Segmentation* — Computer Graphics and Image Processing, vol. 13, n° 3 (juillet 1980), p. 222–241.
- [284] Michael T. Orchard, Charles A. Bouman — *Color Quantization of Images* — IEEE Trans. on Signal Processing, vol. 39, n° 12 (décembre 1991), p. 2677–2690.
- [285] Wilhelm Ostwald — *Die Farbenlehre, Vol 1 : Mathematische Farbenlehre* — Verlag Unesma (1918).
- [286] Wilhelm Ostwald — *Einführung in die Farbenlehre* — 3^e éd., Verlag von Philipp Reclam (1919).
- [287] Wilhelm Ostwald — *Die Farbenfibel* — 5^e éd., Verlag Unesma (1920).
- [288] Alan W. Paeth — *Mapping RGB Triples onto Four Bits* — dans Andrew S. Glassner (éd.), *Graphics Gems*, Academic Press Professional (1990), chapitre IV.4, p. 233–245.
- [289] Christos H. Papadimitriou, Kenneth Steiglitz — *Combinatorial Optimization : Algorithms and Complexity* — Dover (1998).
- [290] E. Pariset — *Les industries de la soie : sériculture, filature, moulinage, tissage, teinture, histoire & statistique* — Imprimerie Pitrat Aîné (1890).
- [291] Kevin J. Parker, Theophano Mitsa — *US Patent 5,708,518 : Method and Apparatus for Halftone Rendering of a Gray Scale Image using a Blue-Noise Mask* — US Patent Office (janvier 1998).
- [292] David A. Patterson, John L. Hennessy — *Computer Organisation and Design : The Hardware/Software Interface* — 5^e éd., Morgan Kaufmann (2014).
- [293] Karl Pearson — *On Lines and Planes of Closest Fit to Systems of Points in Space* — Philosophical Magazine, vol. 2, n° 6 (1901), p. 559–572.
- [294] Robert Peck, Susan Deyl, Jay Miner — *Amiga Hardware Manual* — Commodore Amiga Inc (1985).
- [295] Katherine Peel — *The Concise Atari ST 68000 Programmer's Reference Guide* — Glentop Press (1986).
- [296] Ibrahim Pekkacuksen, Yucel Altunbasak — *Gradient-Based Threshold-Free Color Filter Array Interpolation* — dans *IEEE Int. Conf. on Image Processing (ICIP)*, (2019), p. 137–140.
- [297] William b. Pennebaker, Joan L. Mitchell — *JPEG : Still Image Data Compression Standard* — van Nostrand Reinhold (1992).

- [298] Walter Petryshyn — *Psychological Hue Relationship of the Munsell Color System* — Mémoire de maîtrise, Department of Psychology, University of Windsor, (1967).
- [299] Steven Pigeon — *Contributions à la compression de données* — Thèse de doctorat, Département d'informatique et de recherche opérationnelle, Faculté des Arts et des Sciences, Université de Montréal (décembre 2001).
- [300] Steven Pigeon — *Dithering* — <https://hbfs.wordpress.com/2013/12/31/dithering/> (décembre 2013).
- [301] Thomas Porter, Tom Duff — *Compositing Digital Images* — Computer Graphics, vol. 18, n° 3 (juillet 1984), p. 253–259.
- [302] W. Posselt, W. Holota, E. Kulinyak, G. Kling, T. Kutschied — *NIRSpec – Near Infrared Spectrograph for the JWST* — Procs. SPIE, vol. 5487 (juin 2004), p. 688–697. (Optical, Infrared, and Millimeter Space Telescopes.).
- [303] Charles Poynton — *A Guided Tour of Color Space* — dans Procs. SMPTE Advanced Television and Electronic Imaging Conference, (1990), p. 167–180.
- [304] Charles Poynton — *FAQ about Color* — Autopublié? (1997).
- [305] Charles Poynton — *Digital Video and HDTV: Algorithms and Interfaces* — Morgan Kaufmann (2006).
- [306] Irwin G. Priest, Ferdinand G. Brickwedde — *Minimum Perceptible Colorimetric Purity as a Function of Dominant Wave-Length* — Rapport technique n° RP1009, National Bureau of Standards (mai 1938) Aussi dans *Journal of the National Bureau of Standards*, vol. 20.
- [307] Majid Rabbani, Craig M. Smith — *US Patent 5,412,427: Electronic Camera Utilizing Image Compression Feedback for Improved Color Processing* — US Patent Office (mai 1995).
- [308] A. Rajeev Ramanath, B. Wesley Snyder, C. David Hinks — *Image Comparison Measure for Digital Still Color Cameras* — dans Procs. Int. Conf. on Image Processing (ICIP), (2002), p. I-629–I-632.
- [309] Rajeev A. Ramanath, Wesley E. Snyder — *Color Restoration in Digital Color Cameras* — dans SPIE OPTO-SouthEast, (2000).
- [310] Rajeev A. Ramanath, Wesley E. Snyder — *Adaptive Demosaicking* — Journal of Electronic Imaging, vol. 12, n° 4 (octobre 2003), p. 633–642.
- [311] Rajeev A. Ramanath, Wesley E. Snyder, Griff L. Bilbro, Willam A. Sander III — *Demosaicking Methods for Bayer Color Arrays* — Journal of Electronic Imaging, vol. 11, n° 3 (juillet 2002), p. 306–315.
- [312] Rajeev A. Ramanath, Wesley E. Snyder, Youngjun Yao, Mark S. Drew — *Color Imaging Processing Pipeline* — IEEE Signal Processing Magazine, vol. 22, n° 1 (janvier 2005), p. 34–43.
- [313] Travis A. Rector, Zoltan G. Levay, Lisa M. Frattare, Jayanne English, Kirk Pu'uohau-Pummil — *Image-Processing Techniques for the Creation of Presentation-Quality Astronomical images* — The Astronomical Journal, vol. 133, n° 2 (février 2007), p. 598–611.
- [314] D. Renshaw, P. B. Denyer, G. Wang, M. Lu — *ASIC Image Sensors* — dans Procs. IEEE Int. Symposium on Circuits and Systems, (mai 1990), p. 3038–3041.
- [315] M. E. Ressler, K. G. Sukhatme, B. R. Franklin, J. C. Mahoney, et al. — *The Mid-Infrared Instrument for the James Webb Space Telescope, VIII : The MIRI Focal Plane System* — Publications of the Astronomical Society of the Pacific, vol. 127, n° 953 (2015), p. 675–685.
- [316] Rich Milliron — *Turboprop Rolling Shutter* — https://commons.wikimedia.org/wiki/File:Turboprop_Rolling_Shutter.jpg (2011).

- [317] G. H. Rieke, M. E. Ressler, Jane E. Morrison, L. Bergeron, *et al.* — *The Mid-Infrared Instrument for the James Webb Space Telescope, VII : The MIRI Detector* — Publications of the Astronomical Society of the Pacific, vol. 127, n° 953 (2015), p. 665–674.
- [318] G. H. Rieke, G. S. Wright, Töker, J. Bouwman, *et al.* — *The Mid-Infrared Instrument for the James Webb Space Telescope, I : Introduction* — Publications of the Astronomical Society of the Pacific, vol. 127, n° 953 (2015), p. 584–594.
- [319] Lawrence Gilman Roberts — *Picture Coding using Pseudo-Random Noise* — IRE Trans. on Information Theory, vol. 8, n° 2 (février 1962), p. 145–153.
- [320] A. R. Robertson, *et al.* (éds) — *Colorimetry Part 4 : CIÉ 1976 L*a*b* Colour Space* — Rapport technique n° DS 014-4.3/E :2007, CIÉ Central Bureau (2007).
- [321] Lior Rokach — *A Survey of Clustering Algorithms* — dans Oded Maimon, Lior Rokach (éds), *Data Mining and Knowledge Discovery Handbook*, Springer 2^{de} éd., (2010), p. 269–298.
- [322] Matthew Rollings — *Magnified image of the AMOLED screen on the Google Nexus One smartphone using the RGBG system of the PenTile matrix family* — https://commons.wikimedia.org/wiki/File:Nexus_one_screen_microscope.jpg (juin 2010).
- [323] Roderick T. Ryan (éd.) — *Principles of Color Sensitometry* — 3^e éd., Society of Motion Picture and Television Engineers (1974).
- [324] János Schanda — *Colorimetry : Understanding the CIE System* — John Wiley & Sons (2007).
- [325] Dale A. Schumacher — *A Comparison of Digital Halftoning Techniques* — dans James Arvo (éd.), *Graphics Gems II*, Academic Press Professional (1995), chapitre II.2, p. 57–71.
- [326] Gal Shabtay, Noy Cohen, Nadav Gera, Oded Gigunshinski — *US Patent 10,084,953 B1 : Thin Multi-Aperture Imaging System with Auto-Focus and MEthod for Using Same* — US Patent Office (2018).
- [327] Gaurav Sharma, Wencheng Wu, Edul N. Nadal — *THE CIÉDE2000 Color-Difference Formula : Implementation Notes, Supplementary Test Data, and Mathematical Observations* — Color Research and Application, vol. 30, n° 1 (février 2005), p. 21–30.
- [328] Hamid Rahim Sheikh, Alan Conrad Bovik — *Image Information and Visual Quality* — IEEE Trans. on Image Processing, vol. 15, n° 2 (février 2006), p. 430–444.
- [329] Hamid Rahim Sheikh, Muhammad Farooq Sabir, Alan Conrad Bovik — *A Statistical Evaluation of Recent Full Reference Image Quality Assessment Algorithms* — IEEE Trans. on Image Processing, vol. 15, n° 11 (novembre 2006), p. 3440–3451.
- [330] Edwin K. Shenk — *US Patent 4,199,244 : Automatic Focusinig Camera* — US Patent Office (1982).
- [331] Edwin K. Shenk — *US Patent 4,309,098 : Autofocus Cine Camera Having Automatic Variable Speed Focusing* — US Patent Office (1982).
- [332] Leonard J. Shustek — *MacPaint and QuickDraw Source Code* — <https://computerhistory.org/blog/macpaint-and-quickeadraw-source-code/> (2010).
- [333] Paul M. Simon — *Single-Shot High Dynamic Range and Multi-Spectral Imaging Based on Properties of Color Filter Arrays* — Mémoire de maîtrise, School of Engineering, Dayton University, (2011).
- [334] Simon Singh — *The Code Book : The Science of Secrecy from Ancient Egypt to Quantum Cryptography* — Anchor Books (2000).
- [335] Athanassios Skodras, Charilaos Christopoulos, Touradj Ebrahimi — *The JPEG 2000 Still Image Compression Standard* — IEEE Signal Processing Magazine, vol. 18, n° 5 (septembre 2001), p. 36–58.

- [336] Alvy Ray Smith — *Color Transform Pairs* — SIGGRAPH Computer Graphics, vol. 12, n° 3 (1978), p. 12–19.
- [337] T. Smith, J. Guild — *The CIÉ Colorimetric Standards and Their Use* — Trans. of the Optical Society, vol. XXXIII, n° 3 (1931–1932), p. 73–134.
- [338] Hannah E. Smithson, Greti Dinkova-Bruun, Giles E. M. Gasper, Michael Huxtable, Tom C. b. McLeish, Cecilia Panti — *A Three Dimensional Color Space from the 13th Century* — J. of the Optical Society of America A, vol. 29, n° 2 (février 2012), p. A356–A352.
- [339] Amelia Carolina Sparavigna — *Robert Grosseteste and the Colours* — Int. Journal of Science, vol. 3, n° 1 (janvier 2014), p. 1–6.
- [340] Hugo Steinhaus — *Sur la division des corps matériels en parties* — Bulletin de l'académie polonaise des sciences (CL III), vol. 4, n° 12 (décembre 1956), p. 801–804.
- [341] R. L. Stevenson, G. R. Arce — *Binary Display of Hexagonally Sampled Continuous-Tone Image* — J. of the Optical Society of America, Series A., vol. 2, n° 7 (juillet 1985), p. 1009–1013.
- [342] James A. Storer, Thomas G. Szymanski — *Data Compression via Textual Substitution* — Journal of the ACM, vol. 29, n° 4 (1982), p. 928–951.
- [343] Tilo Strutz — *Multiplierless reversible Color Transforms and Their Automatic Selection for Image Data Compression* — IEEE Trans. Circuits and Systems for Video Technologies, vol. 23, n° 7 (juillet 2013), p. 1249–1259.
- [344] Tilo Strutz, Alexander Leipnitz — *Reversible Colour Spaces without Increased Bit Depth and Their Adaptive Selection* — IEEE Signal Processing Letters, vol. 22, n° 9 (septembre 2015), p. 1269–1273.
- [345] P. Stucki — *MECCA — A Multiple-Error Correction Computation Algorithm for Bi-Level Image Hardcopy Reproduction* — Rapport technique, IBM Zurich Research Laboratory (février 1981).
- [346] Dan Su, Philip Willis — *Demosaicing of Color Images Using Pixel-Level Data-Dependent Triangulation* — dans *Procs. Theory and Practice of Computer Graphics*, (2003), p. 16–23.
- [347] J. Sullivan, L. Ray, R. Miller — *Design of Minimum Visual Modulation Halftone Patterns* — IEEE Trans. on Systems, Man, and Cybernetics, n° 1 (jan-fév 1991), p. 33–38.
- [348] Shijun Sun — *Residual Color Transform using YC_oC_g* — Rapport technique n° JVT-L014, Joint Video Team ISO/IEC (2004).
- [349] Yoshinori Takizawa, Hiroaki Kotaki, Kikuo Saito, Tadashi Sigiki, Yasuo Takemura — *Field Integration Mode CCD Color Television Camera Using a Frequency Interleaving Method* — IEEE Trans. on Consumer Electronics, vol. CE-29, n° 3 (août 1983), p. 358–364.
- [350] William Henry Fox Talbot — *US Patent 5,171 : Improvement in Photographic Pictures* — US Patent Office (juin 1847).
- [351] Marshall F. Tappen, Bryan C. Russell, William T. Freeman — *Exploiting the Sparse Derivative Prior for Super-Resolution and Image Demosaicing* — dans *3rd Int. Workshop on Statistical and Computational Theories of Vision*, (2003).
- [352] David Taubman, Michael Marcellin — *JPEG 2000 Image Compression : Fundamentals, Standard and Practice* — Springer (2012).
- [353] Timhei — *onebitdithertool* — <https://github.com/timheigames/onebitdithertool> (mai 2022).
- [354] Wang Ting, Lin Bin, Zhang Wanzhen, Chen Bo — *High-Accuracy Three-Dimensional Measurement by Improving the Asymmetry of Dithered Patterns* — J. of Physics : Conference Series, vol. 1229 : 3rd Int. Conf. on Machine Vision and Information Technology (février 2019), p. 012029-1–012029-10.

- [355] Ping-Sing Tsai, Tinku Acharya, Ajay K. Ray — *Adaptive Fuzzy Color Interpolation* — Journal of Electronic Imaging, vol. 11, n° 3 (juillet 2003), p. 293–305.
- [356] Yushing T. Tsai, Scott J. Daly, Majid Rabbani — *US Patent 5,172,227 : Image Compression with Color Interpolation for a Single Sensor Image System* — US Patent Office (décembre 1992).
- [357] Kingdon S. Tyler — *Telecasting and Color* — Harcourt, Brace & Co. (1946).
- [358] E. P. T. Tyndall — *Chromaticity Sensibility to Wave-Length Differences as Function of Purity* — J. of the Optical Society of America, vol. 23, n° 1 (janvier 1933), p. 15–24.
- [359] Robert Ulichney — *Digital Halftoning* — MIT Press (1987).
- [360] Oli Usher, Lars Lindberg Christensen — *The Universe Through the Eyes of Hubble* — ESO & ESA/Hubble (2014).
- [361] Georges Valensi — *Brevet 841-335 : Procédé de télévision en couleurs* — Direction de la propriété intellectuelle (février 1939).
- [362] Georges Valensi — *US Patent 2,375,966 : System of television in Color* — US Patent Office (mai 1945).
- [363] Patrick Vandewalle, Sabine Süsstrunk, Martin Vetterli — *A Frequency-Domain Approach to Super-Resolution Imaging from Shared Low-Resolution Images* — EURASIP Journal on Advances in Signal Processing, (2006), p. 1–14.
- [364] Miguel Vega, Rafael Molina, Aggelos K Katsaggelos — *A Bayesian Superresolution Approach to Demosaicing of Blurred Images* — EURASIP Journal on Advances in Signal Processing, vol. 2006 (2006), p. 1–12. (Article ID 25072).
- [365] Heinz-W. von Bülow, Dirk Paulissen — *Le grand Livre du Kodak PhotoCD* — Éds MicroApplication (1994).
- [366] Bruce A. Wallace — *Merging and Transformation of Raster Images for Cartoon Animation* — Computer Graphics, vol. 15, n° 3 (août 1981), p. 253–262.
- [367] S. J. Wan, S. K. M. Wong, P. Prusinkiewicz — *An Algorithm for Multidimensional Data Clustering* — ACM Trans. on Mathematical Software, vol. 14, n° 2 (juin 1988), p. 153–162.
- [368] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, Eero P. Simoncelli — *Image Quality Assessment : From Error Visibility to Structural Similarity* — IEEE Trans. on Image Processing, vol. 13, n° 4 (avril 2004), p. 600–612.
- [369] Zhou Wang, Alan Conrad Bovik, Eero P. Simoncelli — *Structural Approaches to Image Quality Assessment* — dans Alan Conrad Bovik (éd.), *Handbook of Image and Video Processing*, 2^{de} éd., Elsevier Academic Press (2005), chapitre 8.3, p. 961–974.
- [370] William Watson — *Textile Design and Colour : Elementary Weaves and Figured Fabrics* — 2^{de} éd., Longmans, Green and Co. (1921).
- [371] Terry A. Welch — *A Technique for High-Performance Data Compression* — Computer, vol. 17, n° 6 (1984), p. 8–19.
- [372] Terry A. Welch — *US Patent 4,558,302 : High Speed Data Compression and Decompression Apparatus and Method* — US Patent Office (décembre 1985).
- [373] Martyn Wells, J.-W. Pel, Alistair Glasse, G. S. Wright, et al. — *The Mid-Infrared Instrument for the James Webb Space Telescope, VI : The Medium Resolution Spectrometer* — Publications of the Astronomical Society of the Pacific, vol. 127, n° 953 (2015), p. 646–664.

- [374] Andrej Werner (éd.) — *JPEG 9 Lossless Coding* — Rapport technique n° JTC 1/SC 29WG 1, ISO/IEC (2012).
- [375] Steve Wilhite — *Graphics Interchange Format, Version 89a* — Rapport technique, Compuserve Inc. (1989).
- [376] G. S. Wright, David Wright, G. B. Goodson, G. H. Rieke, *et al.* — *The Mid-Infrared Instrument for the James Webb Space Telescope, II : Design and Build* — Publications of the Astronomical Society of the Pacific, vol. 127, n° 953 (2015), p. 595–611.
- [377] W. D. Wright — *A Re-Determination of the Trichromatic Coefficients of the Spectral Colors* — Trans. of the Optical Society, vol. XXX, n° 4 (1928–1929), p. 73–134.
- [378] W. D. Wright, F. H. G. Pitt — *Hue-Discrimination in Normal Colour-Vision* — Procs. of the Physical Society, vol. 46, n° 3 (1934), p. 459–473.
- [379] Xiaolin Wu — *Color Quantization by Dynamic Programming and Principal Analysis* — ACM Trans. on Graphics, vol. 11, n° 4 (octobre 1992), p. 348–372.
- [380] Xiaolin Wu, Wai Kin Choi, Paul Bao — *Color Restoration from Digital Camera Data By Pattern Matching* — dans *Color Imaging : Device-Independent Color, Color Hard Copy, and Graphics Arts II*, (1997), p. 12–17. (SPIE Vol. 3018).
- [381] Xiaolin Wu, Ian H. Witten — *A Fast K-Means Type Clustering Algortihm* — Rapport technique n° 1985-06-01, University of Calgary (juin 1985).
- [382] Günter Wyszecki, G. H. Fielder — *New Color-Matching Ellipses* — J. of the Optical Society of America, vol. 61, n° 9 (septembre 1971), p. 1135–1152.
- [383] Zhigang Xiang — *Color Image Quantization by Minimizing the Maximum Intercluster Distance* — ACM Trans. on Graphics, vol. 16, n° 3 (juillet 1997), p. 260–276.
- [384] Zhigang Xiang, Gregory Joy — *Color Image Quantization by Agglomerative Clustering* — IEEE Computer Graphics and Applications, vol. 14, n° 3 (mai 1994), p. 44–48.
- [385] Xingbo — *A 3 CCD Imaging Block* — https://en.wikipedia.org/wiki/File:A_3CCD_imaging_block.jpg (2009).
- [386] Fan Zhang, Xiaolin Wu, Xiaokang Yang, Wenjun Zhang — *Improved Color Demosaicking in Weak Spectral Correlation* — dans Procs. Visual Communications and Image Processing, (2008), p. 68221N-1–68221N-10. (SPIE Vol. 6822).
- [387] J. Ziv, A. Lempel — *Compression of Individual Sequences via Variable-Rate Coding* — IEEE Trans. on Information Theory, vol. 24, n° 5 (septembre 1978), p. 530–536.
- [388] Assef Zomet, Shmuel Peleg — *Multi-Sensor Super-Resolution* — dans Procs. 6th IEEE Workshop on Applications of Computer Vision, (2002), p. 27–31.
- [389] V. K. Zworykin, G. A. Morton — *Television : The Electronics of Image Transmission* — John Wiley & Sons (1940).

Index

Les noms propres, les noms des algorithmes, et les concepts se retrouvent ici dans le même index. Les titres de livres apparaissent en italiques. Notez que les entrées sont indexées relativement au début des paragraphes qui abordent le sujet en question.

- Écran ~ Cathodique, 162 ~ OLED, 162
16CIF, 15
4CIF, 15
4SIF, 15
* * *
- 4:1:0 ~ Sous-échantillonnage, 19
4:1:1 ~ Sous-échantillonnage, 19
4:2:0 ~ Sous-échantillonnage, 19
4:2:2 ~ Sous-échantillonnage, 19
4:4:0 ~ Sous-échantillonnage, 19
4:4:4 ~ Sous-échantillonnage, 19
666, 114–116
676, 114–116
* * *
- Adaptive Homogeneity Directed*, 30
Additives ~ Couleurs, 77
Adler *et al.* ~ Tramage de, 149
Adler, R. L., 149
Agglomération hiérarchique, 124–126, 132
AHD, 30
Aléatoire ~ Discréétisation, 118
Aléatoire ~ Tramage, 151
Allenbach et Liu ~ Tramage de, 144
Allenbach, Jan P., 144
Analyse ~ En composantes principales, 102
Analyse en composantes principales, 127, 134
Angles ~ Tramage régulier, 137
Approximation ~ \tan^{-1} , 86
Arc-en-ciel, 100
* * *
- Atkinson, William, 155, 167
Autofocus, 25, 31
* * *
- Bande ~ Dynamique, 6–8
Bayer ~ Tramage de, 142
Bayer Filter, 29–31
Bayer, Bryce, 29, 142
BBS, 158
Beaumont, Roberts, 138
BEFLIX, 139, 140, 149
Bel, 133
Binary Split, 121
Bit ~ Fractions de, 115
bit depth, 8
Blanc C, 61
Bruit ~ Bleu, 167 ~ Pseudoaléatoire, 151
Bryngdahl ~ Tramage de, 145
Bryngdahl, Olof, 145
BT.2020, 71–73 ~ Couleurs primaires, 71
BT.2100, 74–76
BT.470, 61 ~ Couleurs primaires, 61
BT.709 ~ Couleurs primaires, 66
Burkes, Daniel, 157
* * *
- Canadian Illustrated News*, 166
Cathode Ray Tube, 162
Cathodique ~ Écran, 162
CCITT G3, 21
Centroïde, 121
Cercle chromatique, 83, 84, 100
CFA, voir *Color Filter Array*
CFR, 65
CGA, 14
Charge couplée ~ Senseur à, 26
Chats, 20
CIÉ, 21
CIÉ, 37 ~ Fonctions de réponse, 37 ~ *Matching functions*, 37 ~ Propriétés désirables, 41 ~ Triangle des couleurs, 38–40 ~ xyz, 41
CIF, 14, 15
CMOS ~ Senseur, 25
Color depth, 8
Color Filter Array, 27
Common Intermediate Format, 14
Comparaison ~ Méthodes de discréétisation, 128
Compression ~ Et espaces de couleurs, 90, 100 ~ Sans perte, 100 ~ Texture, 23
Constante ~ Magique, 63, 67
Contiguë ~ Organisation, 15
Contours ~ Faux, 10, 135
Correction ~ Gamma, 6–8
Coucher ~ De soleil, 109
Couleurs, voir aussi *Espace de couleurs* ~ Additives, 77 ~ Cercle chromatique, 83, 84 ~ Complémentaires, 78 ~ Et compression, 90, 100 ~ Différence au gris, 87 ~ Discréétisation, 105, 135 ~ Homochromes, 34 ~ Hyperspectrales, 3–5, 23 ~ Luminance, 83 ~ Métamérisme, 34 ~ Mode direct, 8–12 ~ Et Newton, 100 ~ Perception, 101 ~ Pondérations sur les, 46 ~ Primaires, 3, 101 ~ Primaires

- optimales, 101 ~ Primaires, BT.2020, 71 ~ Primaires, BT.470, 61 ~ Primaires, BT.709, 66 ~ Profondeur, 8–12 ~ Pureté, 83 ~ Saturation, 82, 83 ~ Sensibilité aux, 33 ~ Soustractives, 77 ~ Teinte, 82, 83 ~ Théorie de la vision, 101 ~ Théories de la, 100 ~ Triangle des, 35 ~ Valeur, 83, 84
- Crocker, Lee Daniel, 154
- CRT, 162
- CRTC, 65
- Cuboctaèdre, 112
- * * *
- D*₅₀, 67
- D*₆₅, 66, 71, 93
- Damera-Venkata, Niranjan, 163
- Décibel, 133
- Deep Color*, 10
- Demi-teinte, 136
- Demosaicing*, 30–31
- Desbarats, George-Édouard-Amable, 166
- Diagramme ternaire, *voir* Triangle des couleurs
- Dichroïque ~ Prisme, 27
- Dichrome ~ Image, 2
- Dichromie, 2
- Diffusion ~ D'erreur, 1, 135, 150–164 ~ Autres géométries, 161–162 ~ Boucle de rétroaction, 152 ~ Bruit bleu, 167 ~ Compacte, 159 ~ Crée une méthode de, 159–161 ~ D'Atkinson, 155 ~ De Burkes, 157 ~ De Crocker *et al.*, 154 ~ De Damera-Venkata et Evans, 163 ~ de Floyd et Steinberg, 154 ~ De Jarvis et Roberts, 164 ~ De Jarvis, Judice et Ninke, 156 ~ De Knuth, 162 ~ De Pigeon, 160–161 ~ De Roberts, 151 ~ De Sierra, 158 ~ De Stucki, 156 ~ Isotropique, 159 ~ Matrice, 153
- Diffusion d'erreur, 113
- Digital Negative*, 22
- Discretisation ~ Adaptative, 116–128 ~ Agglomération hiérarchique, 124–126, 132 ~ Aléatoire, 118 ~ *Binary Split*, 121 ~ Comparaison des méthodes, 128 ~ Couleurs, 105, 135 ~ Directe, 109–116 ~ *K-Means*, 122–123, 131 ~ *Median-cut*, 118 ~ *Min-variance*, 120 ~ Nombre de bits fixes, 110 ~ *Octree*, 121–122 ~ Méthode de Paeth, 112–113 ~ Popularité, 117–118 ~ Popularité avec fusion, 117–118 ~ Programmation dynamique, 127–128, 132
- Dithering*, 1, 135 ~ *Ordered*, 135
- DNG, 22
- Dodécaèdre, 112
- Dynamique ~ Bande, 6–8
- * * *
- Effet ~ De moiré, 137
- eigenbasis, 134
- eigenvalue, 134
- eigenvector, 134
- Eigenvector*, 102
- Entrelacée ~ Organisation, 17–18
- Erreur ~ Diffusion, 1, 135, 150–164 ~ Boucle de rétroaction, 152 ~ Bruit bleu, 167 ~ Compacte, 159 ~ Crée une méthode de, 159–161 ~ D'Atkinson, 155 ~ De Burkes, 157 ~ De Crocker *et al.*, 154 ~ De Damera-Venkata et Evans, 163 ~ de Floyd et Steinberg, 154 ~ De Jarvis et Roberts, 164 ~ De Jarvis, Judice et Ninke, 156 ~ De Knuth, 162 ~ De Pigeon, 160–161 ~ De Roberts, 151 ~ De Sierra, 158 ~ De Stucki, 156 ~ Isotropique, 159 ~ Matrice, 153
- Espace ~ Isotropique, 107
- Espaces de couleurs ~ CIÉ 1931, 37–42 ~ CMY, 77–82 ~ CMYK, 77–82 ~ Conversion ~ XYZ à RGB, 42 ~ Couleurs primaires, 33–36 ~ *D*₅₀, 67 ~ *D*₆₅, 66, 71, 93 ~ Familles, 36–37 ~ Kodak Lossless, 45–46 ~ Kodak YCC, 46–48 ~ Kodak-1, 43–44 ~ Linéaires, 36, 42–82 ~ NCS, 102 ~ Non linéaires, 36, 82–99 ~ CIÉDE2000, 95 ~ HSL, 88–90 ~ HSL conique, 89–90 ~ HSV, 83–87 ~ *L**^a_b^{*}, 92–94, 99 ~ *L**^u_v^{*}, 94, 99 ~ Munsell, 95–96 ~ NCS, 98–99 ~ Ostwald, 96–98, 102 ~ Perceptuellement uniformes, 91 ~ Ohta, 48–50, 102 ~ Série S, 54–61 ~ sRGB, 67 ~ De la télévision, 61–77 ~ BT.2020, 71–73 ~ BT.2100, 74–76 ~ BT.601, 66–67 ~ BT.709, 66–67 ~ Famille BT, 76–77 ~ *IC_TC_p*, 75–76 ~ LMS, 75–76 ~ Transmission primaires, 61–63 ~ *YC_bC_r*, 21, 68–69 ~
- YC_bC_r*, DjVu, 69 ~ *YC_bC_r*, HDTV, 71 ~ *YC_b'C_r'*, 71 ~ *YD_bD_r*, 70 ~ *YP_bP_r*, 69–70 ~ YUV et YIQ, 63–65 ~ Xerox YES, 48 ~ XYZ, 42 ~ *YC_oC_g*, 50–53 ~ *YC_oC_g* réversible, 51–53 ~ *YVU-R*, 53–54
- Étoffes, 138
- Evans, Brian L., 163
- EXIF, 22
- * * *
- Faux ~ Contours, 10, 135
- FCC, 65
- Fidélité ~ Des images, 108–109 ~ MSE, 108 ~ PSNR, 108, 132 ~ SNR, 108
- Film ~ Par ordinateur, 139
- Filtre ~ Bayer, 29–31 ~ Optique, 27 ~ Reconstruction AHD, 30 ~ Reconstruction mosaïque, 30–31 ~ Reconstruction PPG, 31 ~ Reconstruction VGN, 31
- Floyd, Robert W., 154
- Format ~ GIF, 21 ~ Image, 19–22 ~ Image, DNG, 22 ~ Image, PNG, 23 ~ JPEG, 21 ~ JPEG2000, 22 ~ Kodak PhotoCD, 22 ~ PNG, 22 ~ PNM, 21 ~ RAW, 22 ~ TIFF, 21
- Full HD, 14
- * * *
- Gamma ~ Correction, 6–8
- Gamme ~ Grande gamme dynamique, 74
- GIF, 21
- Graphics Interchange Format*, 21
- Grating Dot Profile*, 145
- grayscale*, *voir* Tons de gris
- Gus, 20
- * * *
- Halftone*, 1, 136
- Halftoning*, 135
- Harmon, Leon, 138, 139
- Hawley ~ Tramage de, 145
- Hawley, Stephen, 145
- HD, 14
- HD 1080, 14
- HDR, 74
- Herakleides* ~ Momie de, 166
- Hercules, 14
- Hierarchic Clustering*, *voir* Agglomération hiérarchique

- High Color*, 9
High Dynamic Range, 74
Homochromes, 34
Hotelling, 127, 134
Hubble, 6
Hyperspectrale ~ Image, 3–5, 23
 * * *
- IEC/4WD 61966-2-1**, 67
Image ~ Dichrome, 2 ~ Formats, 19–22 ~ Hyperspectrale, 3–5, 23 ~ Monochrome, 1 ~ Multichrome, 3 ~ Quadrichrome, 3 ~ Résolution, 13–15 ~ Tons de gris, 1 ~ Trichrome, 2
Images ~ Organisation en mémoire des, 15–18
ISO, 26
Isotropique ~ Espace, 107
 * * *
- Jarvis**, J. F., 156, 164
Joint Photographic Experts Group, 21
JPEG, 21, 53, 102
JPEG 2000, 53, 102
JPEG2000, 22
Judice, C. N., 156
Julez, Bela, 142
James Webb Space Telescope, 6
 * * *
- King Quest**, 158
Kitchens, B. P., 149
KLT, 127, 134
K-Means, 122–123, 131
Knowlton, Ken, 138, 139
Kodak PhotoCD, 22
 * * *
- Lagaffe** ~ Gaston, 4
L'Avantage, 4
Leggo, William Augustus, 166
Libcaca, 164
Lisa (ordinateur), 167
Liu, B., 144
Loi de Weber, 133
Luminance, 84 ~ Définition, 83
LZW, 21
 * * *
- Mémoire** ~ Organisation des images en, 15–18
Métrique, 132
- MacPaint**, 155, 167
MacSketch, 167
Magique ~ Constante, 63, 67
Martens, M., 149
Matrice ~ De diffusion d'erreur, 153 ~ De rotation 3D, 55 ~ Normalisation, 60 ~ De permutation, 63 ~ Rotation, orthonormale, 57
Matriochka, 82
MECCA, 156
Median-cut, 118
Métamérisme, 34
Métrique, 93, 95
Métriques, 106–108 ~ L_1 , 106 ~ L_2 , 107 ~ L_∞ , 107 ~ L_p , 107 ~ Pondérées, 107–108
Mile of standard cable, 132
Min-variance, 120
MIPS, 20
Mode ~ Direct, couleurs, 8–12 ~ Palette, 12–13 ~ Semigraphique, 23
Moiré, 137
Momie ~ D'Héraclide, 166
Monochromatique ~ Senseur, 27, 29
Monochrome ~ Image, 1
Monochromie, 1
Morra, Mike, 154
Mosaïque ~ Filtre, 29–31 ~ Reconstruction, 30–31 ~ Reconstruction AHD, 30 ~ Reconstruction PPG, 31 ~ Reconstruction VGN, 31
Motif ~ de Bayer, 29–31
MSC, 132
MSE, 108
Multichrome ~ Image, 3
Multichromie, 3
Multiple-Error Correction Computation Algorithm, 156
Munsell ~ Espace de couleurs de, 95–96
 * * *
- NCS** ~ Espace de couleurs, 98–99
netpbm, 21
Newton ~ Et couleurs, 100
Ninke, W. H., 156
Normalisation ~ D'un espace de couleurs, 60
Nostalgie ~ lack thereof, 23
NTSC, 14
 * * *
- Octree**, 121–122
- Œil** ~ Sensibilité, 10–12
OLED ~ Écran, 162
Optimisation ~ Combinatoire, 132
Optique ~ Filtre, 27
Ordered dithering, 135
Organisation ~ Contiguë, 15 ~ Des images en mémoire, 15–18 ~ Entre-lacée, 17–18 ~ Plans de bits, 16–17
Ostwald ~ Espace de couleurs d', 96–98, 102
 * * *
- PAL/SÉCAM**, 14
Palette, 12–13 ~ *Web Safe*, 9 ~ *Web Safe*, 114–116
Panchromatique ~ Senseur, 27, 29
Patterned Pixel Grouping, 31
PCA, 127, 134
Perception ~ Des couleurs, 101
PhotoCD, 22
Pixillation, 139
Plans de bits ~ Organisation, 16–17
PNG, 22, 23
PNM, 21
PoemField, 139
Polyèdre, 112 ~ Cuboctaèdre, 112 ~ Dodécaèdre, 112 ~ Duals, 112
Popularité, 117–118
Popularité avec fusion, 117–118
Portable Any-Map Format, 21
Portable Network Graphics, 22, 23
Poulay, Paul, 154
Poulet ~ En demi-teinte, 137
Poulet surgelé, 4
PPG, 31
Prismes, 27–28 ~ Dichroïques, 27 ~ Trichroïques, 27
Profondeur ~ Des couleurs, 8–12
Programmation dynamique, 127–128, 132
Pseudoaléatoire ~ Bruit, 151
PSNR, 108, 132
Pureté, 83
 * * *
- QCIF**, 15
QHD, 14
Quadrichrome ~ Image, 3
Quadrichromie, 3
QVGA, 14
QXGA, 14
 * * *

RAW, 22
Réponse ~ Psychovisuelle, 10–12
Résolution ~ D'image, 13–15
RLE, 21
Roberts, C. S., 164
Rolling Shutter, 26
Rotation ~ Matrice orthonormale, 57 ~ Matrice, 3D, 55
* * *

Saturation, 83
saturation ~ Définition, 82
Schumacher ~ Tramage de, 148
Schumacher, Dale A., 148
Schwartz, Lillian, 139
Semigraphics, 23
Senseur ~ actif, 25 ~ Charge couplée, à, 26 ~ CMOS, 25 ~ Monochromatique, 27, 29 ~ Panchromatique, 27, 29
Sensibilité ~ De l'œil, 10–12
Sierra Lite, 158
Sierra On-Line, 158
Sierra, Frankie, 158
SIF, 14, 15
Signal to Noise Ratio, voir SNR
Similigravure, 136
SNR, 108
Soleil ~ Coucher de, 109
Source Input Format, 14
Sous-échantillonnage, 18–19 ~ 4:1:0, 19 ~ 4:1:1, 19 ~ 4:2:0, 19 ~ 4:2:2, 19 ~ 4:4:0, 19 ~ 4:2:0, 19 ~ 4:2:2, 19 ~ 4:4:4, 19
Soustractives ~ Couleurs, 77
Spirou, 4
SQCIF, 15

Steinberg, Louis, 154
Stucki, P., 156
SVGA, 14
SXGA, 14
* * *
Tagged Image File Format, 21
Talbot, William H. F., 166
 \tan^{-1} , 86
Teinte, 83 ~ Définition, 82
Telephone Transmission Units, 132
Télescope spatial ~ Hubble, 6 ~ James Webb, 6
Télévision ~ Histoire de la, 61, 100
Texture ~ Compression de, 23
Théorie ~ De la vision, 101
TIFF, 21
Tissage, 138
Tons de gris ~ Image, 1
Toutânkhamon, 13
Tramage, 1 ~ Aléatoire, 151 ~ D'Adler *et al.*, 149 ~ D'Allebach et de Liu, 144 ~ De Bayer, 142 ~ De Bryngdahl, 145 ~ De Hawley, 145 ~ De Julesz, 142 ~ De Schumacher, 148 ~ Ordonné, 140–150 ~ Origine du mot, 138 ~ Régulier, 135–137 ~ Régulier, angles, 137 ~ Régulier, problèmes, 150
Transformée ~ Hotelling, 102 ~ Karhunen-Loève, 102
Transformées ~ Linéaires de couleurs, 36
Tresser, C. P., 149
Triangle ~ Des Couleurs, 35 ~ Des Couleurs CIÉ, 38–40
Triche, 86

Trichoïque ~ Prisme, 27
Trichrome ~ Image, 2
Trichromie, 2
TRS-80 Color Computer, 23
True Color, 9
TU, 132
* * *
UHD, 14
UHD2, 14
Unisys, 23
UXGA, 14
* * *
Valeur ~ D'une couleur, 83 ~ D'une couleur, 84
Value ~ Propre, 134
Vanderbeek, Stan, 139
Variable Number of gradients, 31
Vecteur ~ Propre, 102, 121, 127, 134
VGA, 14
VGN, 31
Vision ~ Théorie de la, 101
* * *
Watson, William, 138
Web Safe Palette, 9
Weber ~ Loi de, 133
Web Safe Palette, 114–116
WSXGA, 14
Wu, C. W., 149
* * *
XGA, 14
* * *
 YC_bC_r , 21

Traitement numérique des images

Notes de cours (C++)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam consequat vel sapien in sodales. Mauris tincidunt pulvinar enim ut dictum. Sed lectus massa, maximus a est non, gravida tincidunt mi. Non, ce n'est pas un message secret. Oui, c'est exprès. Mauris molestie fringilla nunc vitae tincidunt. Nam imperdiet mauris volutpat, consectetur orci nec, rhoncus orci. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum vestibulum vel tellus vel lobortis. Suspendisse fermentum malesuada purus malesuada posuere. Nullam aliquet cursus ex, et lacinia ante lobortis quis. Sed vel posuere leo.

